Soft Computing

– Introduction to Machine Learning –

Karl-Heinz Zimmermann

TUHH

July 13, 2021

TUHH
Technische Universität Hamburg-Harburg

Prof. Dr. Karl-Heinz Zimmermann
Hamburg University of Technology
21071 Hamburg
Germany

This monograph is listed in the GBV database and the TUHH library.

https://doi.org/10.15480/882.3652
http://hdl.handle.net/11420/9878
urn:nbn:de:gbv:830-882.0139751

Preface

Soft Computing is a branch of computer science that deals with computationally hard tasks such as NP complete problems for which the known algorithms do not deliver solutions in polynomial time. In opposition to conventional (hard) computing, soft computing provides solutions which may tolerate imprecision, partial truth, uncertainty, and approximation. The paradigm of soft computing is foremost the human mind – learning and inference. The principal fields of soft computing are basically machine learning, fuzzy logic, and evolutionary computation. The applications of soft computing lie in prominent fields such as artificial intelligence and knowledge engineering. The course provides an introduction to soft computing particularly to the increasingly important field of machine learning.

*Starred material can be safely skipping on a first reading without loss of continuity. An index will be separately available.

Hamburg, October 2020 *Karl-Heinz Zimmermann*

Dedication

To my family
for sempiternal
support.

Contents

- Bayesian networks
- Statistical inference and learning
- Artificial neural networks
- Fuzzy logic

Contents - Chapters

1 Introduction to Bayesian networks

2 Hidden Markov model

3 Inference and learning in Bayesian networks

4 Statistical inference and learning

5 Sequence alignment

6 Tree models

7 Artificial neural networks

8 Fuzzy logic

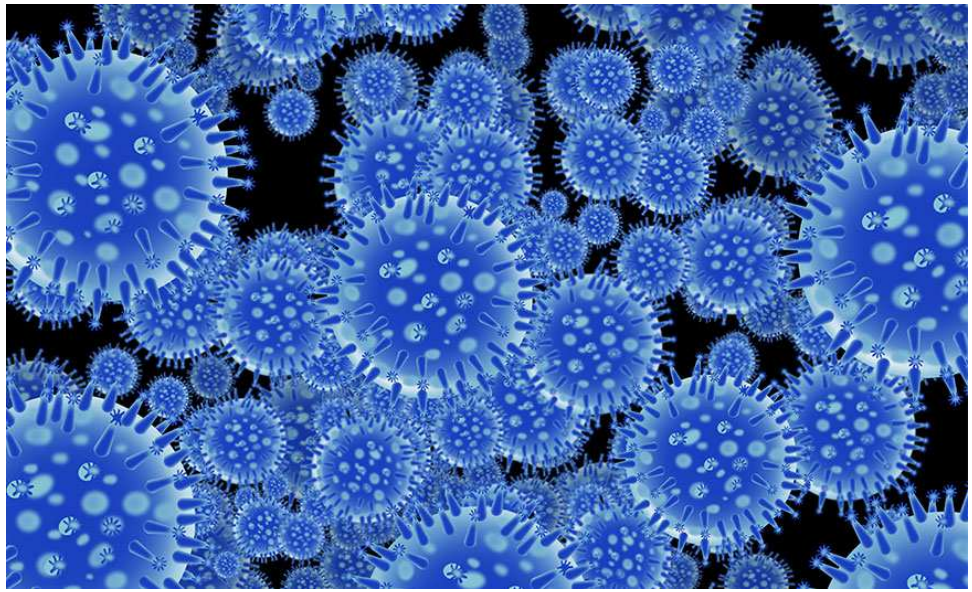
Appendix

9 Computational statistics with R

10 Markov processes

Corona Times

- Lecture is online, not always live.
- Lectures will be recorded for later use.
- Exam is oral in February or March.



Formalities

- *Schedule*:
  - Tuesday, 10:00 - 11:00 am (live)
  - Wednesday (recorded)
- *StudIP:* appointments, manuscript, recorded lecture (media links)
- *Exam:* oral (20-25 min)

Oral Exam (25 min.)

Topics:

- Bayesian networks
- Hidden Markov model
- Statistical inference and learning
- Artificial neural networks
- Fuzzy logic.

*No starred material, no R.

Literature

- David Barber, *Bayes Reasoning and Machine Learning*, Cambridge Univ. Press, Cambridge, 2012.
- Ernst Klement, Radko Mesiar, Endre Pap, *Triangular Norms*, Kluwer, Dordrecht, 2000.
- Timo Koski, John M. Noble, *Bayesian Networks*, Wiley, New York, 2009.
- Hidetoshi Nishimori, *Statistical Physics of Spin Glasses and Information Processing*, Oxford Univ. Press, London, 2001.
- Raul Royas, *Neural Networks*, Springer, Berlin, 1996.
- Lior Pachter, Bernd Sturmfels, *Algebraic Statistics for Computational Biology*, Cambridge Univ. Press, Cambridge, 2005.
- Karl-Heinz Zimmermann, *Algebraic Statistics*, TubDok, Hamburg, 2016.
- Karl-Heinz Zimmermann, *Soft Computing*, Manuscript, Hamburg, 2018.

Digitalization

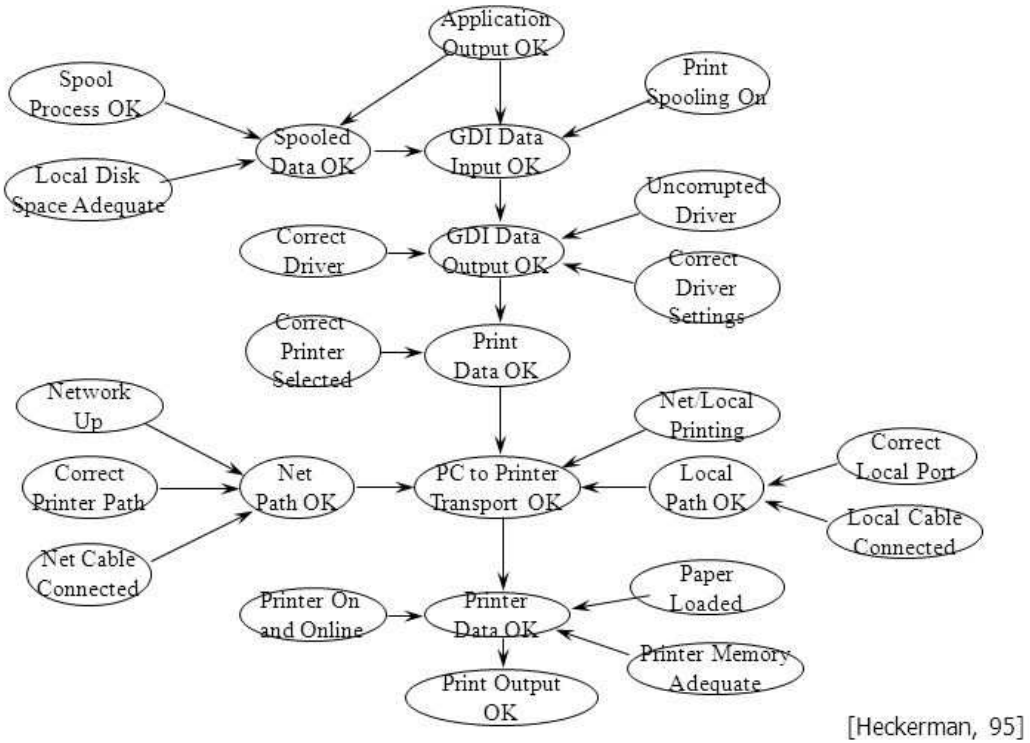
Conversion of analog data to digital data.

- Big data: storage of large data and data retrieval
- Machine learning: structure of data, learning (estimation), inference
- Infrastructure: sensors, actors, communication networks, embedded systems.

Bayesian Networks

Graphical representation of probabilistic relationships between random variables

Example: Printer Troubleshooting (Microsoft Windows 95)



[Heckerman, 95]

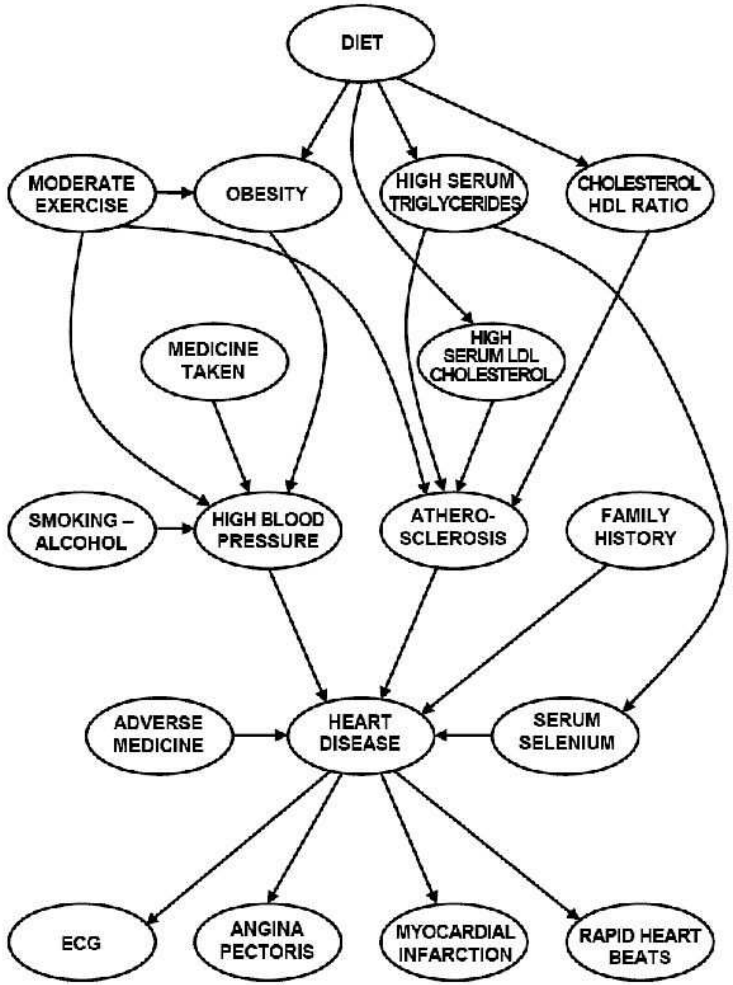
Soft Computing

K.-H. Zimmermann

Preliminaries
Preface
Contents
Formalities
Literature
Digitalization

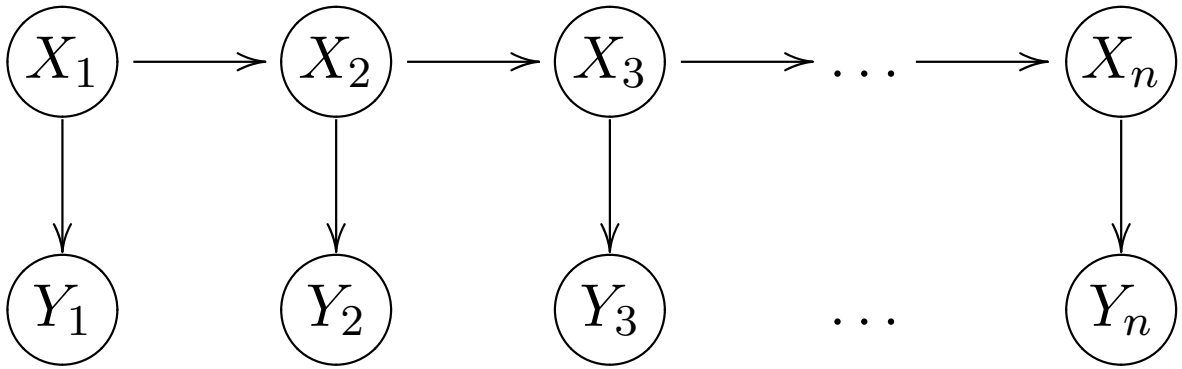
Bayesian Networks

Applications in artificial intelligence, expert systems, machine learning



Hidden Markov Model

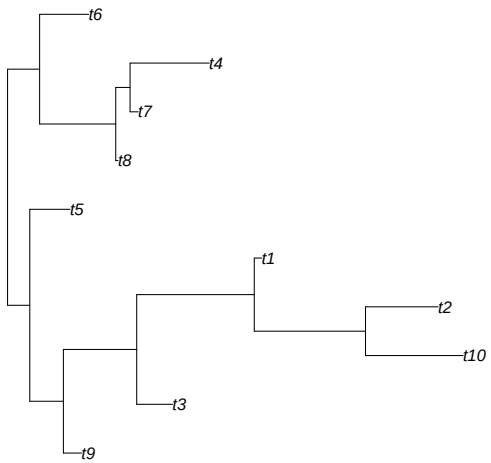
Spatial and temporal pattern analysis



- *Inference:* Given output sequence $y_1 y_2 \ldots y_n$, find most probable state sequence $x_1 x_2 \ldots x_n$ (Viterbi algorithm)
- *Learning:* Estimate transition probabilities (Estimation-Maximation, Baum-Welch algorithm)

Tree Models

Analysis of phylogenetic trees



- *Inference:* Given contemporary species (leaves), find most probable anchestors (Felsenstein algorithm)
- *Learning:* Estimate transition probabilities (Jukes-Cantor model and others)

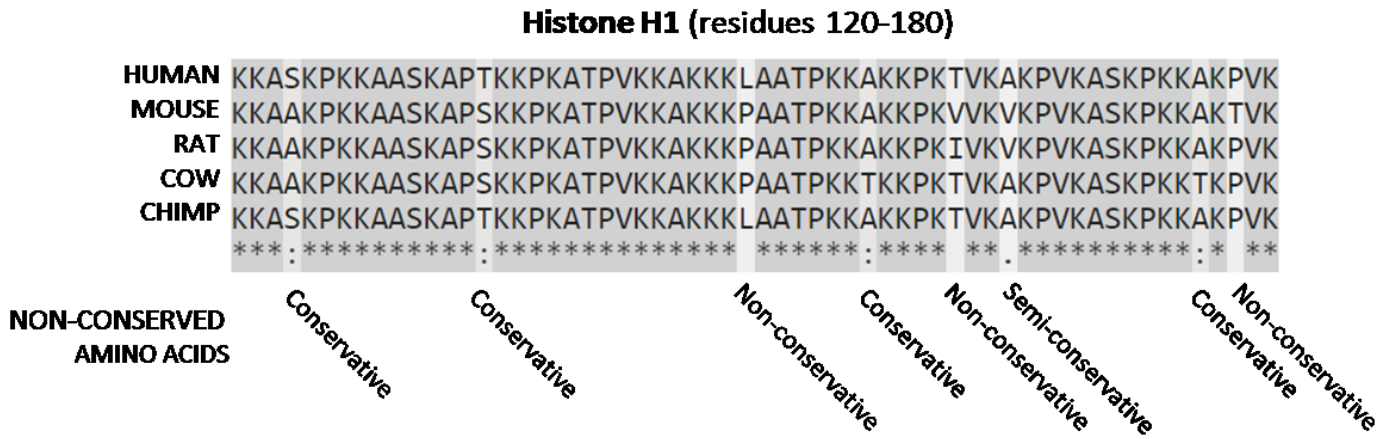
Soft Computing

K.-H. Zimmermann

Preliminaries
Preface
Contents
Formalities
Literature
Digitalization

Sequence Alignment

Sequence alignment is the basic operation of molecular biologists



- *Inference:* Given pairwise scores, find optimal alignment (Needleman-Wunsch algorithm)
- *Global inference:* Find the optimal alignments for all pairwise scores (polytope propagation)
- *Learning:* Estimate pairwise scores (PAM, Blosum)

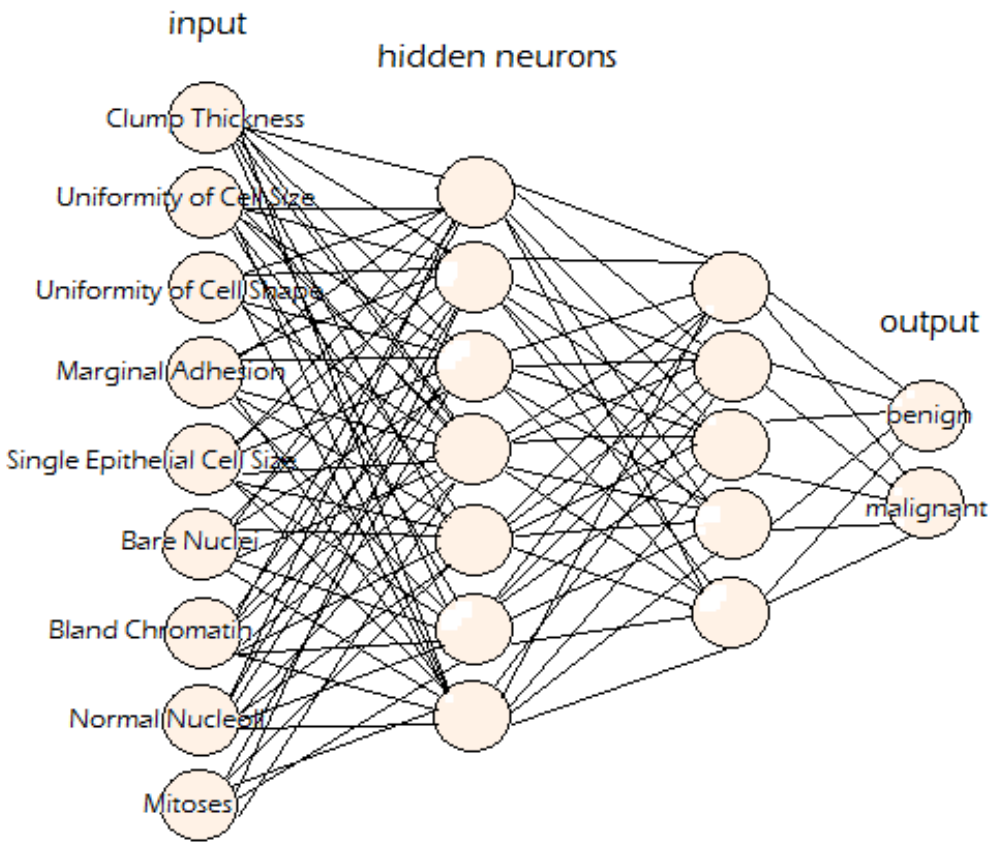
Soft Computing

K.-H. Zimmermann

Preliminaries
Preface
Contents
Formalities
Literature
Digitalization

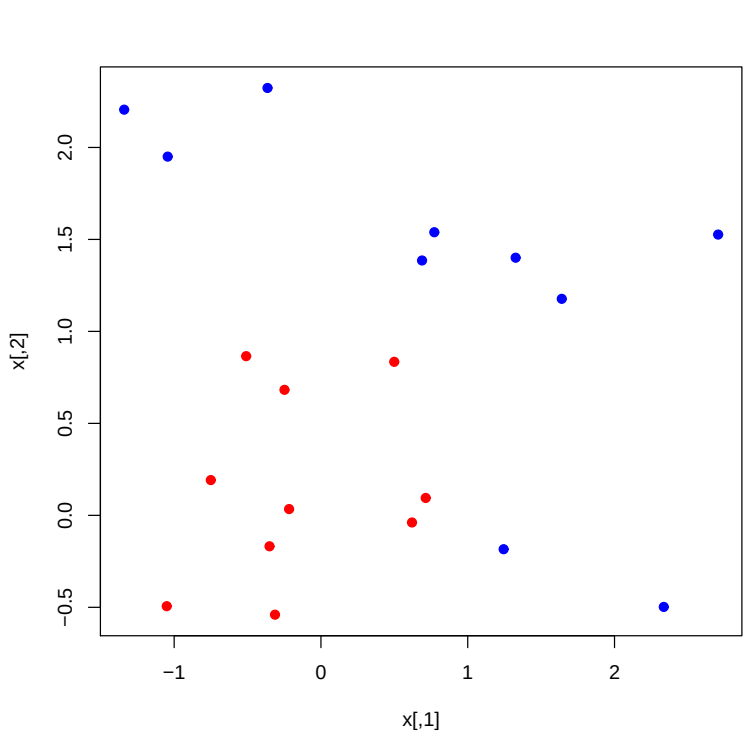
Neural Networks

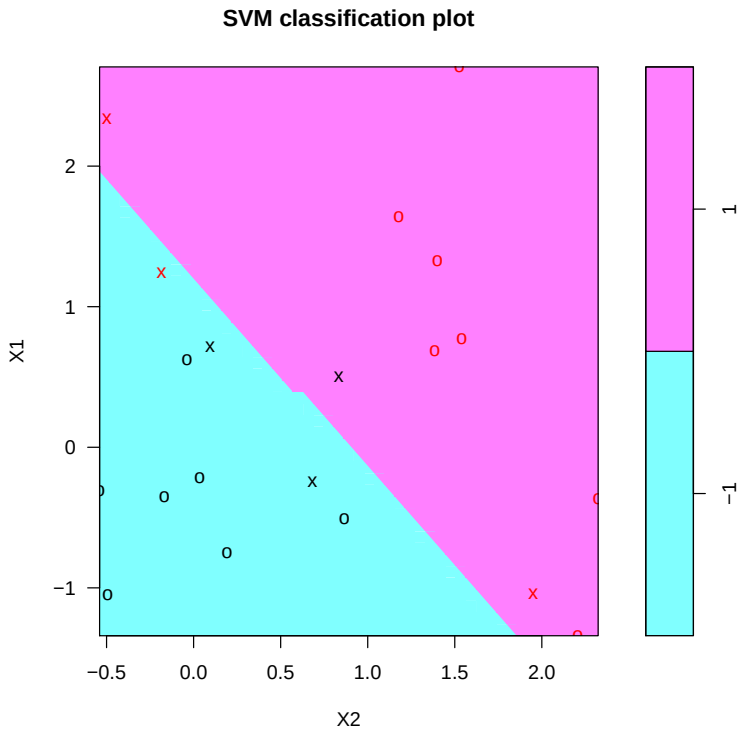
Artificial neural networks can generalize from examples, used in diagnosis, prediction, and recognition. Deep learning is the new black!



Neural Networks

Linear classification of data (perceptron):



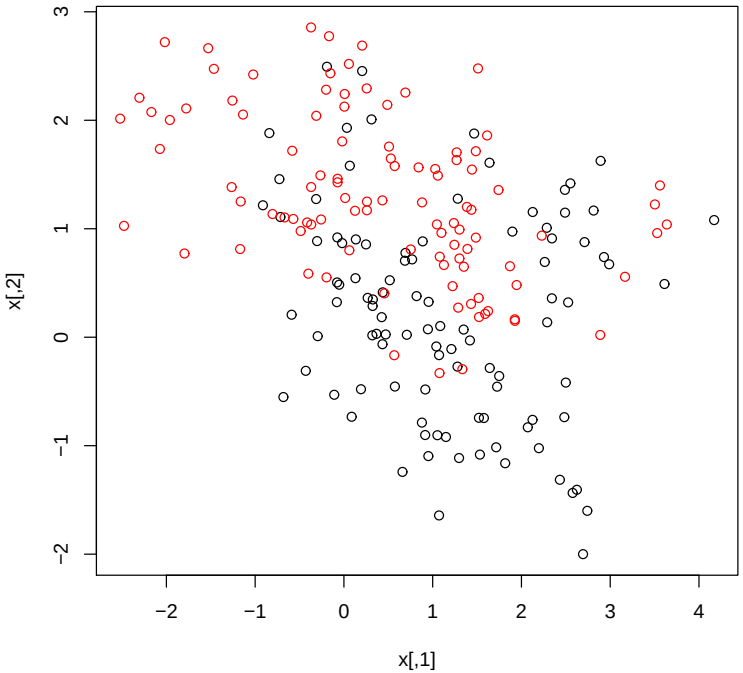


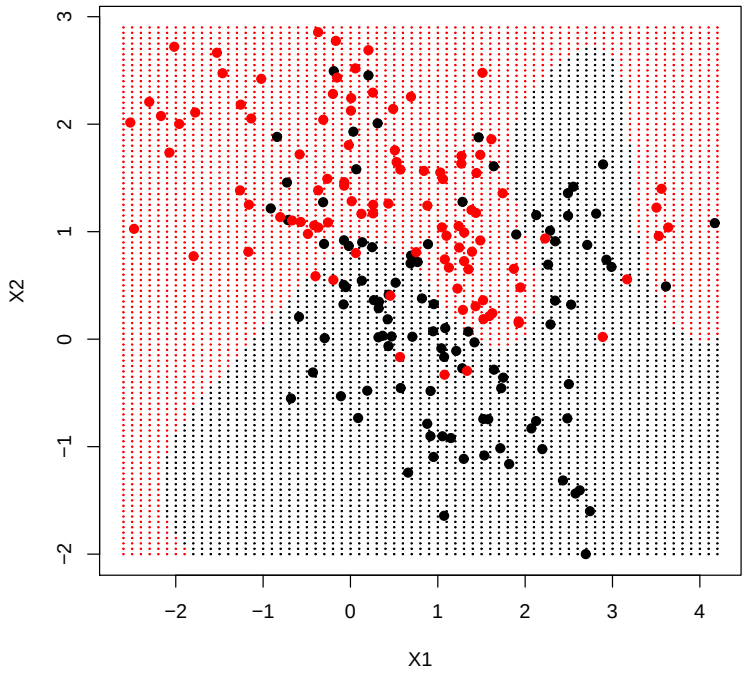
Soft Computing

K.-H. Zimmermann

Preliminaries
Preface
Contents
Formalities
Literature
Digitalization

Vector Support Machine





Neural Networks

Recognition of handwritten digits:

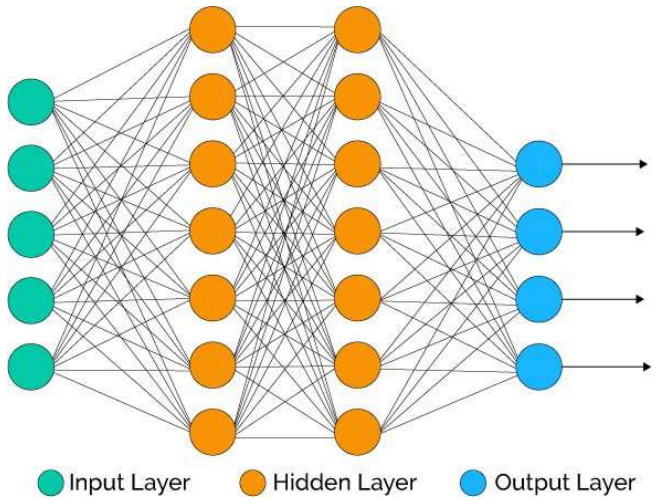
- *MNist*: Database of handwritten digits.
- Database (50 MB) consists of 60,000 digits for training and 10,000 digits for testing.
- Black-white digits consist of $28 \times 28 = 784$ pixels (intensity values 0-255).

Soft Computing

K.-H. Zimmermann

Preliminaries
Preface
Contents
Formalities
Literature
Digitalization

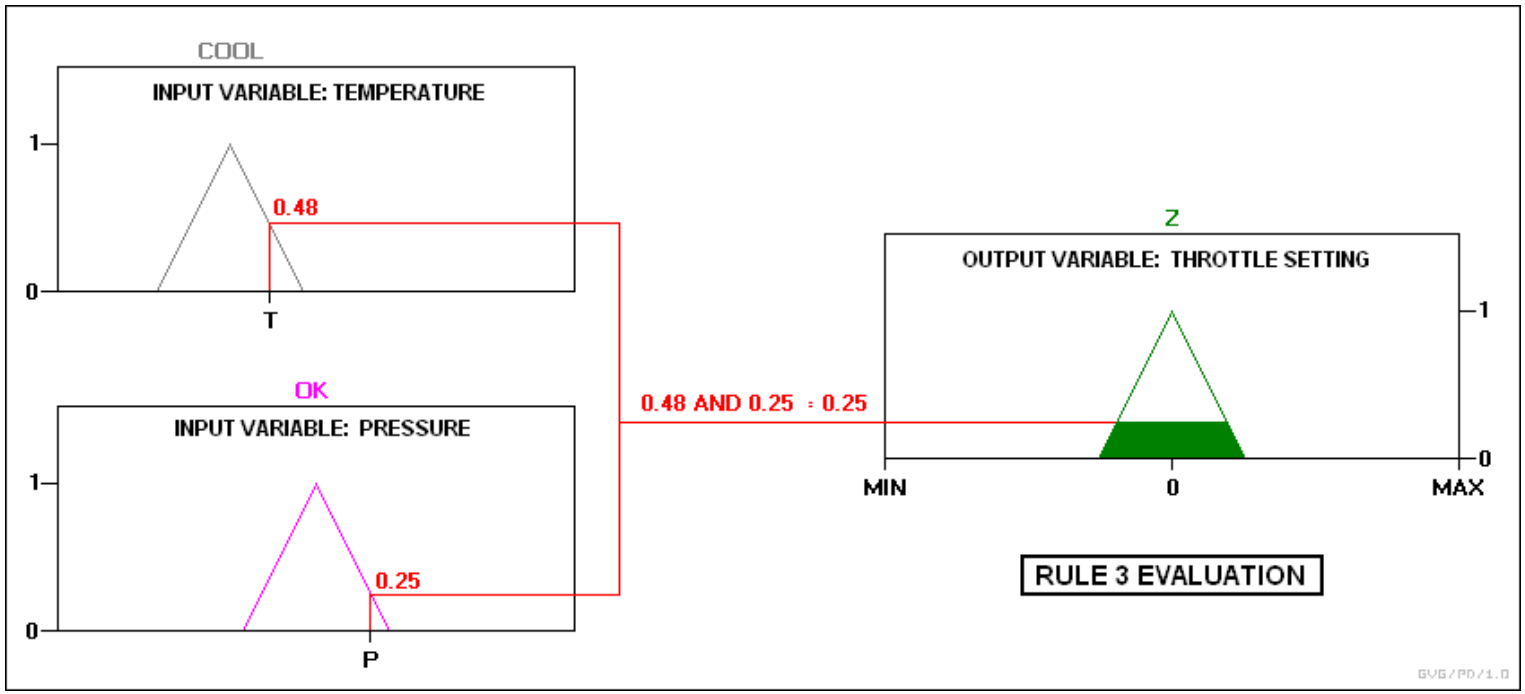
Neural Networks



- Artificial neural network has several layers with a few thousand neurons.
- Hardware: CPU and graphics cards
- Error rate 0,35%, i.e., 35 of 10,000 test digits are wrongly classified.
- Challenges: large number of parameters (12 mio.), efficient implementation.

Fuzzy Logic

A fuzzy control system analyzes analog input values in terms of discrete variables to yield an analog output value.



Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Part I

Introduction to Bayesian Networks

Contents

- Probabilities
- Graph Concepts
- Belief Networks
- Conditional Independence
- Markov Equivalence
- Bayesian Networks in R

Knowledge

- Probabilistic concepts, Bayes' rule
- Basic connections, d-dependence, belief network
- Markov equivalence, essential graph

Skills

- Computation of d-dependence
- Factorization of joint probability distribution along DAG
- Computation of essential graph
- Specification of belief network using R

Probabilities

- Joint probabilities
- Conditional and marginal probabilities
- Bayes' rule
- Independence
- Conditional independence

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Joint Probability Distribution

Given random variables X and Y over finite state sets $\mathcal{X}$ and $\mathcal{Y}$, resp.

- The *joint probability function* $p = p_{X,Y}$ is a probability distribution over $\mathcal{X} \times \mathcal{Y}$; i.e.,

$$p(x, y) \geq 0 \quad \text{for all } (x, y) \in \mathcal{X} \times \mathcal{Y}, \tag{1}$$

and

$$\sum_{(x,y) \in \mathcal{X} \times \mathcal{Y}} p(x, y) = 1. \tag{2}$$

- Example: adult population with X (gender, male or female) and Y (age, 18-100).
- Joint probability distribution contains complete information about the random variables, may not be available. However, conditional probabilities (smaller building blocks of low order) might be known.

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Marginal Distribution

Given joint probability function $p_{X,Y}$ over state set $\mathcal{X} \times \mathcal{Y}$.

- The distribution p_X of X is the *marginal distribution* of $p_{X,Y}$ given by

$$p_X(x) = \sum_{y \in \mathcal{Y}} p_{X,Y}(x,y), \quad x \in \mathcal{X}. \tag{3}$$

- Given random variables X and Y both over state set $\{0,1\}$ and joint probability distribution $p = p_{X,Y}$ given by

p	$Y=0$	$Y=1$
$X=0$	0.06	0.14
$X=1$	0.24	0.56

Then the marginals are

$$p_X(0) = 0.20,\ p_X(1) = 0.80,\ p_Y(0) = 0.30,\ p_Y(1) = 0.70.$$

Conditional Probability Distribution

Given random variables X and Y over $\mathcal{X}$ and $\mathcal{Y}$, resp.

- The *conditional probability* of $x \in \mathcal{X}$ given $y \in \mathcal{Y}$ is defined as

$$p_{X|Y}(x|y) = \frac{p_{X,Y}(x,y)}{p_Y(y)}. \tag{4}$$

  In particular, if $p_Y(y) = 0$, then $p_{X|Y}(x|y)$ is undefined.

- For each $y \in \mathcal{Y}$ with $p_Y(y) \neq 0$, the function $p_{X|Y}(\cdot|y)$ is a probability distribution over $\mathcal{X}$, since $p_{X|Y}(x|y) \geq 0$ for all $x \in \mathcal{X}$ and

$$\begin{aligned} \sum_x p_{X|Y}(x|y) &= \sum_x \frac{p_{X,Y}(x,y)}{p_Y(y)} = \frac{1}{p_Y(y)} \sum_x p_{X,Y}(x,y) \\ &= \frac{p_Y(y)}{p_Y(y)} = 1. \end{aligned}$$

Conditional Probability Distribution – Example

Given random variables X and Y both over state set $\{0, 1\}$ and joint probability distribution $p = p_{X,Y}$ given by

p	$Y = 0$	$Y = 1$
$X = 0$	0.06	0.14
$X = 1$	0.24	0.56

The marginals are

$$p_X(0) = 0.20,\ p_X(1) = 0.80,\ p_Y(0) = 0.30,\ p_Y(1) = 0.70.$$

The conditional probability distribution $p_{X|Y}$ is

$$\begin{aligned} p_{X|Y}(0|0) &= 0.06/0.30 = 0.20, \quad p_{X|Y}(0|1) = 0.14/0.70 = 0.20, \\ p_{X|Y}(1|0) &= 0.24/0.30 = 0.80, \quad p_{X|Y}(1|1) = 0.56/0.70 = 0.80. \end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Bayes' Rule

- For all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$,

$$p_{X,Y}(x,y) = p_{X|Y}(x|y)p_Y(y) \tag{5}$$

and

$$p_{X,Y}(x,y) = p_{Y|X}(y|x)p_X(x). \tag{6}$$

- This gives *Bayes' rule*

$$p_{X|Y}(x|y) = \frac{p_{Y|X}(y|x)p_X(x)}{p_Y(y)}. \tag{7}$$

where

- p_X *prior* probability,
- p_Y *evidence*,
- $p_{Y|X}$ *likelihood*,
- $p_{X|Y}$ *posterior* probability.

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Bayes' Rule – Example

- Animal population: 60% female (F) and 40% male (M). Disease (D): 10% of the females and 5% of the males suffering.
- Random variables X and Y with state sets $\{F, M\}$ and $\{D, \bar{D}\}$, resp.
- Joint probability distribution $p = p_{X,Y}$:

p	$Y = D$	$Y = \bar{D}$
$X = F$	0.06	0.54
$X = M$	0.02	0.38

- Priors: $p_X(F) = 0.60$ and $p_X(M) = 0.40$.

Bayes' Rule – Example (Cont'd)

- Likelihoods:

$$p_{Y|X}(D|F) = \frac{p_{X,Y}(F,D)}{p_X(F)} = 0.10,\ p_{Y|X}(\bar{D}|F) = 0.90,$$

and

$$p_{X|Y}(D|M) = \frac{p_{X,Y}(M,D)}{p_Y(M)} = 0.05,\ p_{X|Y}(\bar{D}|M) = 0.95.$$

- Law of total probability:

$$p_Y(D) = p_X(F) \cdot p_{Y|X}(D|F) + p_X(M) \cdot p_{Y|X}(D|M) = 0.08,$$

so $p_X(\bar{D}) = 0.92$.

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Bayes' Rule – Example (Cont'd)

- By Bayes' rule, the posteriories are

$$p_{X|Y}(F|D) = \frac{p_X(F) \cdot p_{Y|X}(D|F)}{p_Y(D)} = 0.75$$

and so

$$p_{X|Y}(M|D) = 1 - p_{X|Y}(F|D) = 0.25.$$

Independence

Given random variables X and Y over $\mathcal{X}$ and $\mathcal{Y}$, resp.

- X and Y are *(statistically) independent* if

$$p_{X,Y} = p_X \cdot p_Y. \tag{8}$$

  One random variable gives no extra information about the other one.

- The random variables X and Y in the first example are independent:

$$\begin{aligned}
p_{X,Y}(0,0) &= 0.06 = 0.20 \cdot 0.30 = p_X(0)p_Y(0),\\
p_{X,Y}(0,1) &= 0.14 = 0.20 \cdot 0.70 = p_X(0)p_Y(1),\\
p_{X,Y}(1,0) &= 0.24 = 0.80 \cdot 0.30 = p_X(1)p_Y(0),\\
p_{X,Y}(1,1) &= 0.56 = 0.80 \cdot 0.70 = p_X(1)p_Y(1).
\end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Proposition

Given random variables X and Y over $\mathcal{X}$ and $\mathcal{Y}$, resp.
The variables X and Y are independent iff for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ with $p_Y(y) > 0$,

$$p_{X|Y}(x|y) = p_X(x). \tag{9}$$

Proof.

- Let X and Y be independent. Then $p_X(x)p_Y(y) = p_{X,Y}(x,y) = p_{X|Y}(x|y)p_Y(y)$ and so $p_{X|Y}(x|y) = p_X(x)$ since $p_Y(y) > 0$.
- Conversely, let $p_{X|Y}(x|y) = p_X(x)$ for all states $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ with $p_Y(y) > 0$. Since $p_{X,Y}(x,y) = p_{X|Y}(x|y)p_Y(y)$, we have $p_{X,Y}(x,y) = p_X(x)p_Y(y)$.

Observation

Given random variables X, Y, and Z with joint probability function $p_{X,Y,Z}$.
The joint probability distribution $p_{X,Y,Z}$ factors as follows,

$$p_{X,Y,Z} = p_{X|Y,Z} p_{Y|Z} p_Z, \tag{10}$$

since

$$p_{X,Y,Z} = p_{X|Y,Z} p_{Y,Z} \tag{11}$$

and

$$p_{Y,Z} = p_{Y|Z} p_Z. \tag{12}$$

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Proposition

Given random variables $X_1, \ldots, X_n$.
The joint probability function $p_{X_1,\ldots,X_n}$ is given by

$$p_{X_1,X_2,\ldots,X_n} = \prod_{i=1}^{n} p_{X_i|X_{i+1},\ldots,X_n}. \tag{13}$$

Proof.

We have

$$p_{X_1,X_2,\ldots,X_n} = p_{X_1|X_2,\ldots,X_n} p_{X_2,\ldots,X_n}.$$

By induction, the result follows. □

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Conditional Independence

Given random variables X, Y, and Z over state sets $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$, resp.
The variables X and Y are *conditionally independent* given Z, written

$$X \perp Y \mid Z, \tag{14}$$

if for all $(x, y, z) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$,

$$p_{X,Y,Z}(x, y, z) = p_{X|Z}(x \mid z) p_{Y|Z}(y \mid z) p_Z(z). \tag{15}$$

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Proposition

Two random variables X and Y are conditionally independent given the random variable Z iff for all $(x, y, z) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ with $p_{Y|Z}(y \mid z) > 0$ and $p_Z(z) > 0$,

$$p_{X|Y,Z}(x \mid y, z) = p_{X|Z}(x \mid z). \tag{16}$$

Proof.

We have

$$p_{X,Y,Z}(x, y, z) = p_{X|Y,Z}(x \mid y, z) p_{Y|Z}(y \mid z) p_Z(z).$$

The variables X and Y are conditionally independent given Z iff

$$p_{X,Y,Z}(x, y, z) = p_{X|Z}(x \mid z) p_{Y|Z}(y \mid z) p_Z(z).$$

Since $p_{Y|Z}(y \mid z) > 0$ and $p_Z(z) > 0$, we have $p_{X|Y,Z}(x \mid y, z) = p_{X|Z}(x \mid z)$. □

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Proposition

Two random variables X and Y are conditionally independent given the random variable Z iff for all $(x, y, z) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ with $p_Z(z) > 0$,

$$p_{X,Y|Z}(x, y|z) = p_{X|Z}(x|z)p_{Y|Z}(y|z). \tag{17}$$

Proof.

We have

$$p_{X,Y,Z}(x, y, z) = p_{X,Y|Z}(x, y \mid z)p_Z(z).$$

The variables X and Y are conditionally independent given Z iff

$$p_{X,Y,Z}(x, y, z) = p_{X|Z}(x \mid z)p_{Y|Z}(y \mid z)p_Z(z).$$

Since $p_Z(z) > 0$, we have
$p_{X,Y|Z}(x, y|z) = p_{X|Z}(x|z)p_{Y|Z}(y|z)$. □

Coin Experiment

Given random variables X and Y both over state set $\{h, t\}$ denoting the outcome of first and second throwing of fair coin, resp.

- Suppose there is no dependency in the throwing, i.e, the variables X and Y are independent,

$$p_{X,Y} = p_X p_Y.$$

- Let random variable Z over $\{0, 1, 2\}$ denotes the number of heads obtained in both throwings.
- Claim that the variables X and Y are not conditionally independent given Z.

Coin Experiment (Cont'd)

The variables X and Y are not conditionally independent given Z.

Proof.

- By definition, $p_{X,Y|Z}(h,h|1) = 0$.
- By definition, $p_{X|Z}(h|1) = \frac{p_{X,Z}(h,1)}{p_Z(1)}$.
- We have

$$p_{X,Z}(h,1) = p_{X,Y}(h,t) = p_X(h)p_Y(t) = \frac{1}{2^2} = \frac{1}{4}$$

- We have

$$p_Z(1) = p_X(h)p_Y(t) + p_X(t)p_Y(h) = \frac{1}{2^2} + \frac{1}{2^2} = \frac{1}{2}.$$

- So $p_{X|Z}(h|1) = \frac{1}{2}$ and by symmetry $p_{Y|Z}(h|1) = \frac{1}{2}$.
- Thus $p_{X|Z}(h|1)p_{Y|Z}(h|1) > 0$, but $p_{X,Y|Z}(h,h|1) = 0$. □

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Graph Concepts

- Directed and undirected graphs
- Graph data
- Trails and paths
- Connected and complete graphs

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

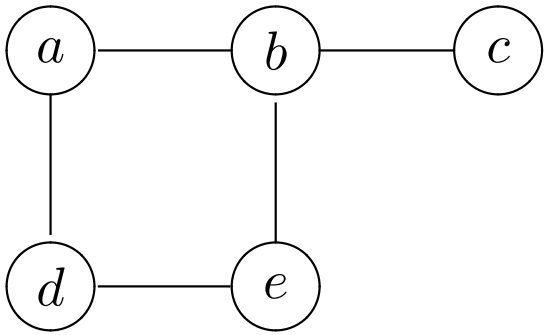
Graph Definition (Classical)

An *undirected graph* $G = (V, E)$ is a pair with

- V finite *node* set and
- E *edge* set with $E \subseteq \binom{V}{2}$,

where $\binom{V}{2}$ denotes the set of all 2-subsets of V.
An edge $\{u, v\}$ in G is denoted by $u - v$.

Example



Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Graph Concepts

- A *directed graph* or *digraph* $G = (V, E)$ is a pair with
  - V finite *node* set and
  - E *edge* set with $E \subseteq (V \times V) \setminus \{(v, v) \mid v \in V\}$.

  The edges are ordered pairs of distinct nodes.
- Let $(u, v) \in E$ be an edge:
  - (u, v) *undirected* if also $(v, u) \in E$, written $u - v$.
  - (u, v) *directed* if $(v, u) \notin E$, written $u \to v$.

  Write $u \cdots v$ if there is an edge of some type between u and v.
- Let G be a graph:
  - G *undirected* if all edges in G are undirected.
  - G *directed* (or *digraph*) if all edges in G are directed.

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

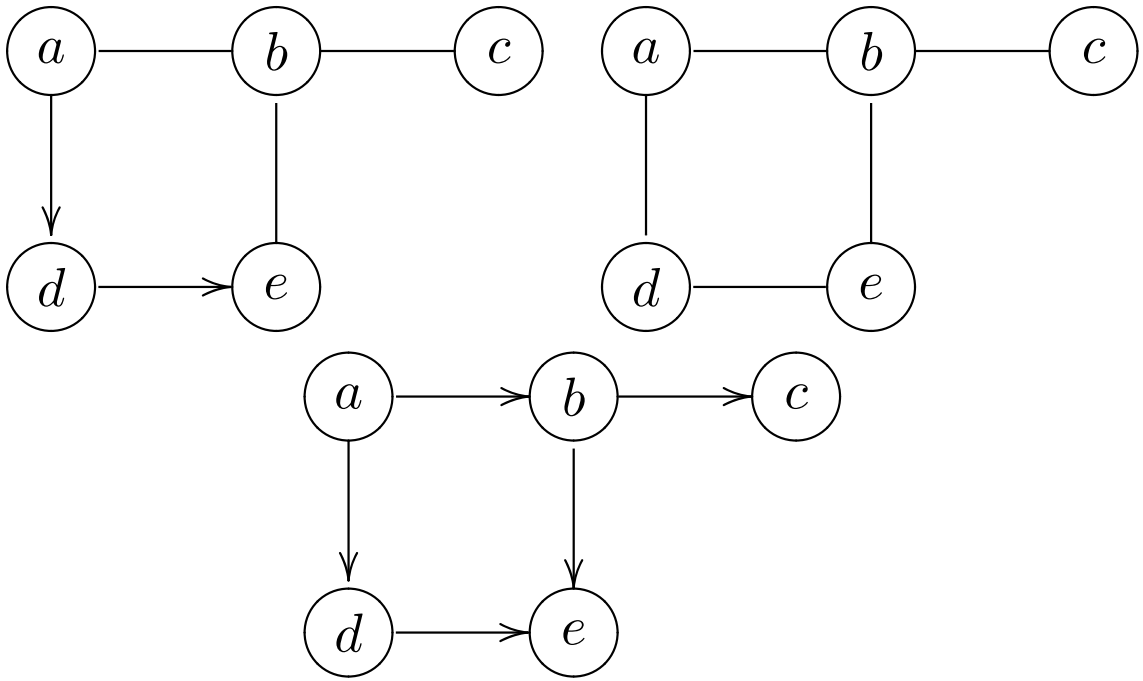
Conditional Independence

Markov Equivalence

Using R

Example

The figure shows graph, undirected graph, and directed graph (digraph):



Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Neighborhood

Let $G = (V, E)$ be a graph and $u, v, w, x \in V$.

- u *parent* of v if $(u, v) \in E$. The parent set of v is

$$\Pi(v) = \{u \in V \mid (u, v) \in E\}. \tag{18}$$

- w *child* of v if $(v, w) \in E$. The set of children of v is

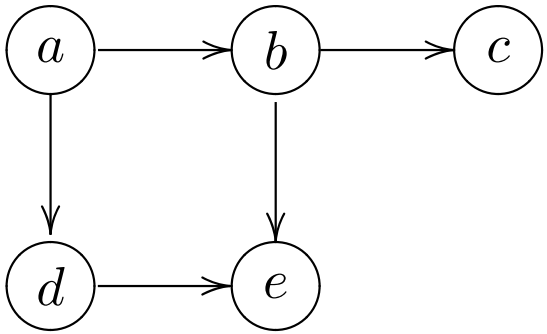
$$\Gamma(v) = \{w \in V \mid (v, w) \in E\}. \tag{19}$$

- x *neighbor* of v if x parent or child of v. The set of neighbors of v is

$$N(v) = \Pi(v) \cup \Gamma(v). \tag{20}$$

Example

Given the digraph



The parent of e are b, d, the children of a are b, d, and the neighbors of b are a, c, e.

Subgraphs

Let $G = (V, E)$ be a graph and U be a subset of V.

- $G_U = (U, E_U)$ is a *subgraph* of G if $E_U \subseteq E \cap (U \times U)$.
  The edges in G_U are edges between nodes of U in G.
- If $E_U = E \cap (U \times U)$, then $G_U = (U, E_U)$ is the subgraph *induced* by U.
  An induced subgraph G_U contains an edge (u, v), $u, v \in U$, iff (u, v) is an edge in G.

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

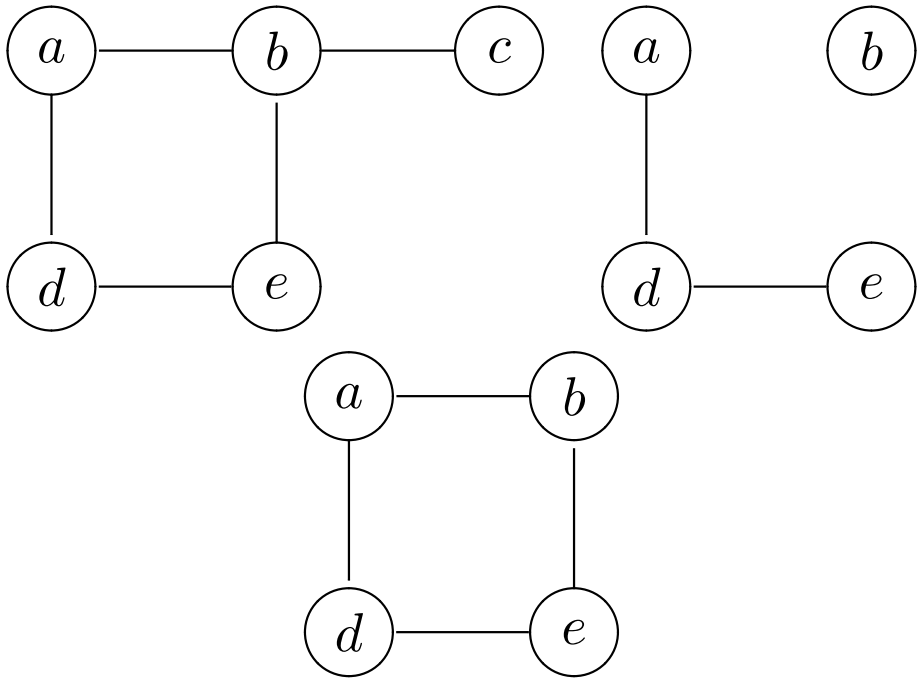
Conditional Independence

Markov Equivalence

Using R

Example

The figure shows graph, subgraph, and induced subgraph.



Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Trails, Paths, and Cycles

Let $G = (V, E)$ be a graph and u, v be distinct nodes in G.

- A *trail* in G of length $k \geq 1$ between u and v is a sequence of nodes $w = (w_0, \ldots, w_k)$ such that $w_0 = u$, $w_k = v$, and $w_i \cdots w_{i+1}$ for each $0 \leq i \leq k - 1$.
- A *undirected path* in G of length $k \geq 1$ from u and v is a trail $w = (w_0, \ldots, w_k)$ in G such that $w_0 = u$, $w_k = v$, and $w_i - w_{i+1}$ for each $0 \leq i \leq k - 1$.
- A *directed path* in G of length $k \geq 1$ from u and v is a trail $w = (w_0, \ldots, w_k)$ in G such that $w_0 = u$, $w_k = v$, and $w_i \to w_{i+1}$ for each $0 \leq i \leq k - 1$.
- A (directed, undirected) *cycle* in G is a (directed, undirected) path $w = (w_0, \ldots, w_k)$ in G with $w_0 = w_k$.

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

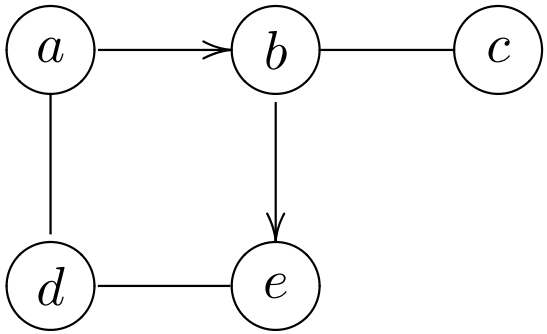
Conditional Independence

Markov Equivalence

Using R

Example

Consider the graph



- Trails: (a, b, c), (a, d, e, b, c)
- Directed path: (a, b, e)
- Undirected path: (a, d, e)

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

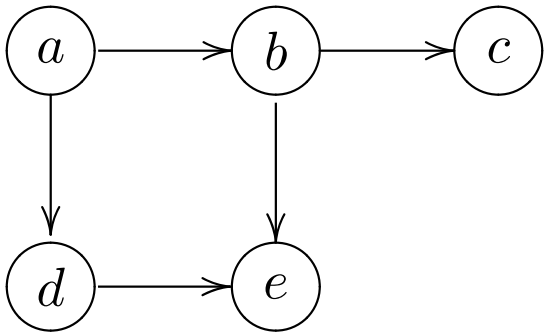
Markov Equivalence

Using R

Anchestors and Descendants

Let $G = (V, E)$ be a graph and u, v be distinct nodes in G.

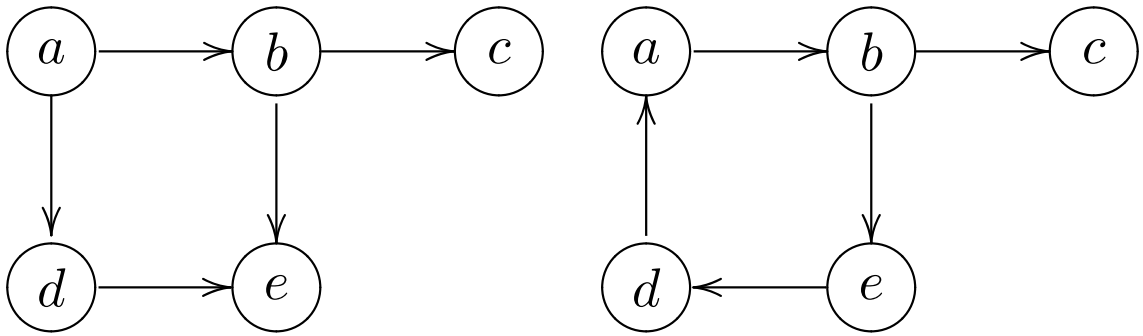
- The node v is a *descendant* of node u if there is a directed path from u to v, but no directed path from v to u.
- The node v is an *ancestor* of node u if u is a descendant of v.
- Consider the graph



The descendants of a are b, c, d, e, the ancestors of c are a, b.

DAG

- A graph $G = (V, E)$ G is a *directed acyclic* graph (DAG) if G is a digraph and there are no directed cycles in G.
- The figure shows DAG and digraph with cycle:



Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Belief Networks

- Ordering of nodes in DAG
- Introduction of belief network

Topological Sorting

Given DAG $G = (V, E)$ with node set $V = \{v_1, \ldots, v_n\}$.

There is an ordering $(v_{\sigma(1)}, \ldots, v_{\sigma(n)})$ of the nodes of G such that for each $1 \leq i \leq n$, the parent set of the node $v_{\sigma(i)}$ is a subset of $\{v_{\sigma(1)}, \ldots, v_{\sigma(i-1)}\}$.

Proof.

The DAG G is cycle-free and thus has at least one node, say v_{j_1}, which has no children (case $k = 1$).
For each $k = 1, \ldots, n$, consider the subgraph of G with $n - k + 1$ nodes. Take a node which has no children. Let v_{j_k} be such a node and put $\sigma(n - k + 1) = j_k$. Remove the node v_{j_k} and all corresponding edges. This provides a subgraph which is a DAG with $n - k$ nodes. Continuing in this way gives the required ordering. □

Example

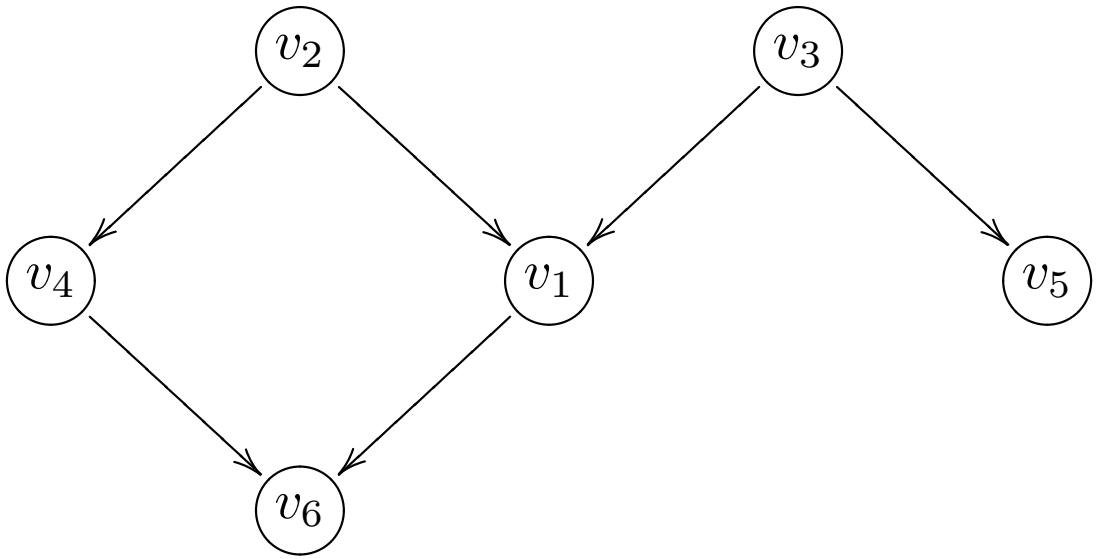
The DAG below gives several orderings satisfying the above condition such as

$$(v_3, v_2, v_1, v_4, v_5, v_6)$$

or

$$(v_3, v_2, v_1, v_4, v_6, v_5).$$

Add nodes from behind which have no children in the shrinking DAG.



Belief Network

Given DAG $G = (V, E)$ with nodes (random variables) $X_1, \ldots, X_n$, and collection p of conditional probability distributions of the random variables.
(G, p) is a *belief network* if the following holds:

- For each node X_i without parent, there is a probability distribution p_{X_i}.
- For each node $X_i \in V$ with parent set $\Pi_i \neq \emptyset$, there is a conditional probability distribution $p_{X_i|\Pi_i}$.
- The joint probability function $p_{X_1,\ldots,X_n}$ factors according to the (spatial) *Markov property*,

$$p_{X_1,\ldots,X_n} = \prod_{i=1}^{n} p_{X_i|\Pi_i}. \tag{21}$$

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

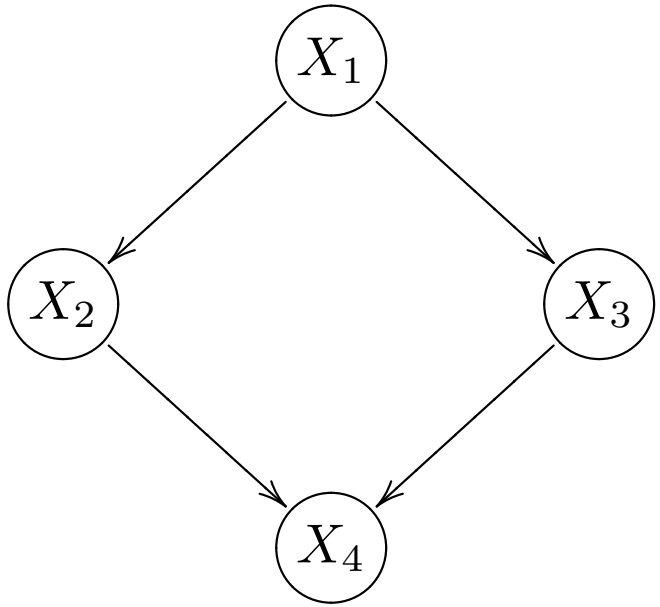
Using R

Example

Consider the belief network by the DAG below.

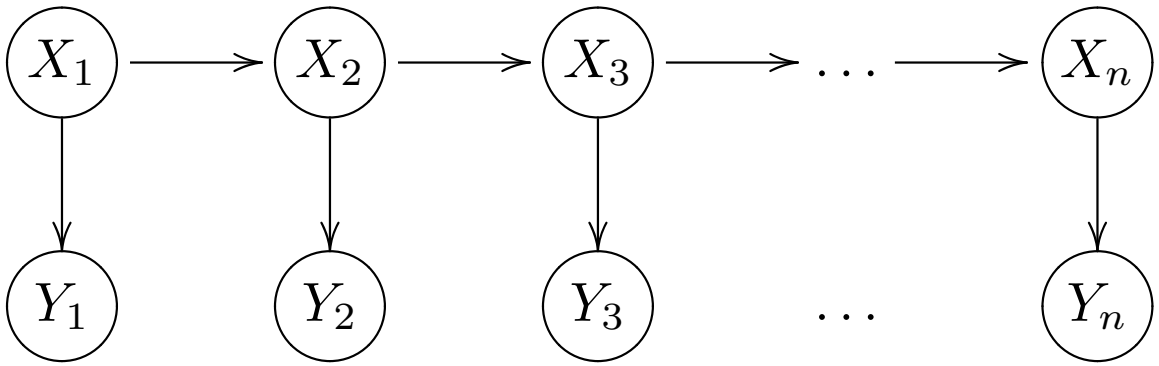
- The random variables $X_1, \ldots, X_4$ have parent sets $\Pi(X_1) = \emptyset$, $\Pi(X_2) = \{X_1\}$, $\Pi(X_3) = \{X_1\}$, and $\Pi(X_4) = \{X_2, X_3\}$.
- Factoring of joint probability function:

$$\begin{aligned} p_{X_1,X_2,X_3,X_4} &= p_{X_1} \cdot p_{X_2|X_1} \cdot p_{X_3|X_2,X_1} \cdot p_{X_4|X_3,X_2,X_1}, \text{ by (13)} \\ &= p_{X_1} \cdot p_{X_2|X_1} \cdot p_{X_3|X_1} \cdot p_{X_4|X_3,X_2}, \text{ by (21)}. \end{aligned}$$



Example: Hidden Markov Model

Belief network with DAG:



Joint probability distribution ($n = 4$):

$$p_{X_1,\ldots,X_4,Y_1,\ldots,Y_4} = p_{X_1} p_{Y_1|X_1} p_{X_2|X_1} p_{Y_2|X_2} p_{X_3|X_2} p_{Y_3|X_3} p_{X_4|X_3} p_{Y_4|X_4}.$$

*Belief Networks – Free Parameters

Given belief network with random variables $X_1, \ldots, X_n$ over common state set $\{0, 1\}$.

- The non-factored joint probability function $p_{X_1,\ldots,X_n}$ has $2^n - 1$ free parameters.
- The factored joint probability function $p_{X_1,\ldots,X_n}$ in (21) has

$$2^{u(X_1)} + \ldots + 2^{u(X_n)} \tag{22}$$

free parameters, where $u(X_i)$ is the number of parents of X_i.

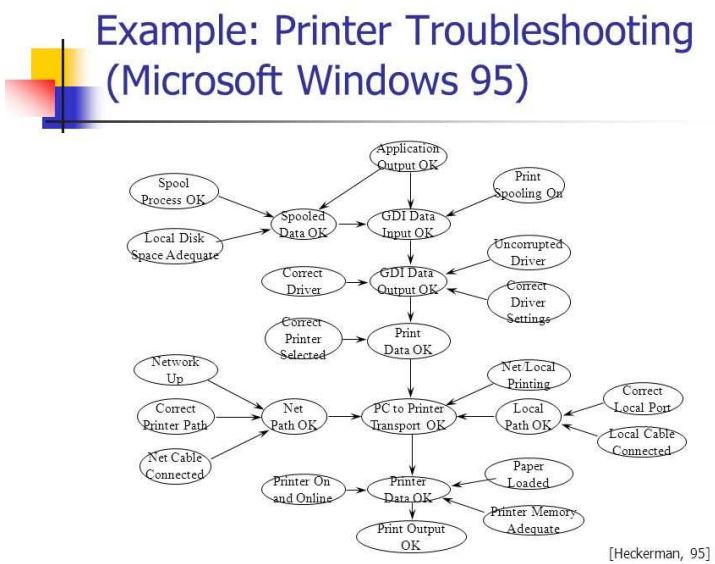
Example

Take random variable X:

- X has no parent (1 free parameter): $p_X(0) = 1 - p_X(1)$.
- X has one parent Y (2 free parameters): $p_{X|Y}(0|0) = 1 - p_{X|Y}(1|0)$ and $p_{X|Y}(0|1) = 1 - p_{X|Y}(1|1)$.
- X has two parents Y, Z (4 free parameters): $p_{X|Y,Z}(0|0,0) = 1 - p_{X|Y,Z}(1|0,0)$, $p_{X|Y,Z}(0|0,1) = 1 - p_{X|Y,Z}(1|0,1)$ $p_{X|Y,Z}(0|1,0) = 1 - p_{X|Y,Z}(1|1,0)$, and $p_{X|Y,Z}(0|1,1) = 1 - p_{X|Y,Z}(1|1,1)$.

Example: Printer Troubleshooting (Microsoft Windows 95)

The belief network below has 26 nodes. Suppose all random variables have binary state sets. Then the joint probability distribution has $2^{26} - 1 = 67.108.863$ parameters. As a belief network, by (22), the number of parameters is $17 \cdot 1 + 0 \cdot 2^1 + 4 \cdot 2^2 + 2 \cdot 2^3 + 3 \cdot 2^4 = 97$. Impressive!



Example: Soft XOR Gate

Hard version and soft version of XOR gate:

X	Y	$Z = X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

X	Y	$p_{Z\|X,Y}(1\|x,y)$
0	0	0.10
0	1	0.98
1	0	0.80
1	1	0.20

Consider the belief network

(23)

with priors $p_X(1) = 0.60$ and $p_Y(1) = 0.80$. Then the joint probability distribution is

$$p_{X,Y,Z} = p_X p_Y p_{Z|X,Y}.$$

Claim that the random variables X and Y are not conditionally independent given Z.

Example: Soft XOR Gate (Cont'd)

Joint probability distribution

$$
\begin{aligned}
p_{X,Y,Z}(0,0,0) &= 0.40 \cdot 0.20 \cdot 0.90 = 0.072, \\
p_{X,Y,Z}(0,0,1) &= 0.40 \cdot 0.20 \cdot 0.10 = 0.008, \\
p_{X,Y,Z}(0,1,0) &= 0.40 \cdot 0.80 \cdot 0.02 = 0.0064, \\
p_{X,Y,Z}(0,1,1) &= 0.40 \cdot 0.80 \cdot 0.98 = 0.3136, \\
p_{X,Y,Z}(1,0,0) &= 0.60 \cdot 0.20 \cdot 0.20 = 0.024, \\
p_{X,Y,Z}(1,0,1) &= 0.60 \cdot 0.20 \cdot 0.80 = 0.096, \\
p_{X,Y,Z}(1,1,0) &= 0.60 \cdot 0.80 \cdot 0.80 = 0.384, \\
p_{X,Y,Z}(1,1,1) &= 0.60 \cdot 0.80 \cdot 0.20 = 0.096.
\end{aligned}
$$

Compute $p_{X|Z}(1|0) \cdot p_{Y|Z}(1|0)$ and $p_{X,Y|Z}(1,1|0)$.

Example: Soft XOR Gate (Cont'd)

By marginalization,

$$\begin{aligned} p_{X,Z}(1,0) &= \sum_y p_{X,Y,Z}(1,y,0) = 0.408, \\ p_{Y,Z}(1,0) &= \sum_x p_{X,Y,Z}(x,1,0) = 0.3904, \\ p_Z(0) &= \sum_{x,y} p_{X,Y,Z}(x,y,0) = 0.4864. \end{aligned}$$

Then

$$\begin{aligned} p_{X|Z}(1|0) &= \frac{p_{X,Z}(1,0)}{p_Z(0)} = \frac{0.408}{0.4864} = 0.8389, \\ p_{Y|Z}(1|0) &= \frac{p_{Y,Z}(1,0)}{p_Z(0)} = \frac{0.3904}{0.4864} = 0.8026, \\ p_{X,Y|Z}(1,1|0) &= \frac{p_{X,Y,Z}(1,1,0)}{p_Z(0)} = \frac{0.384}{0.4864} = 0.7895. \end{aligned}$$

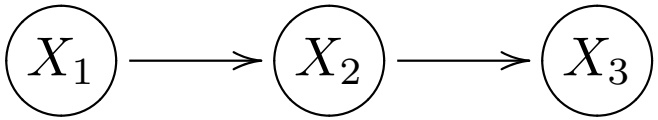
We have $p_{X|Z}(1|0) \cdot p_{Y|Z}(1|0) = 0.6884 \neq 0.7895 = p_{X,Y|Z}(1,1|0)$. Thus the random variables X and Y are not conditionally independent given Z.

Conditional Independence

- Basic connections
- D-separation
- Markov blankets
- Faithful belief networks

Serial Connection

- A *serial connection* of three random variables X_1, X_2, and X_3 has the form



  The variable X_2 is *serial* or the *intermediate cause*.

- As a belief network, the joint probability distribution factors as

$$p_{X_1,X_2,X_3} = p_{X_1} p_{X_2|X_1} p_{X_3|X_2}. \tag{24}$$

  Thus by Bayes' rule,

$$\begin{aligned} p_{X_1,X_3|X_2} &= \frac{p_{X_1,X_2,X_3}}{p_{X_2}} = \frac{\overbrace{p_{X_1} p_{X_2|X_1}} p_{X_3|X_2}}{p_{X_2}} \\ &= \frac{\overbrace{p_{X_2} p_{X_1|X_2}} p_{X_3|X_2}}{p_{X_2}} = p_{X_1|X_2} \cdot p_{X_3|X_2}. \end{aligned} \tag{25}$$

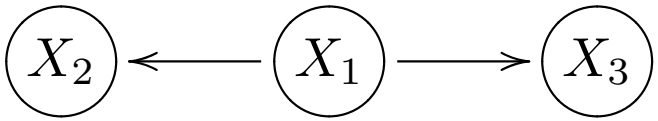
  By (17), X_1 and X_3 are conditionally independent given X_2.

Serial Connection – Example

In the belief network below, social environment S influences education E which in turn affects job J. Then social environment and job are conditionally independent given the education.

Diverging Connection

- A *diverging connection* of three random variables X_1, X_2, and X_3 has the form



The variable X_1 is *diverging* or the *common cause*.

- As a belief network, the probability distribution factors as

$$p_{X_1,X_2,X_3} = p_{X_1} p_{X_2|X_1} p_{X_3|X_1}. \tag{26}$$

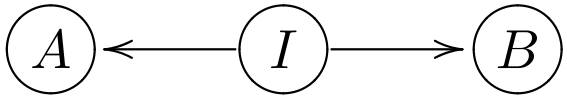
Thus we obtain

$$\begin{aligned} p_{X_2,X_3|X_1} &= \frac{p_{X_1,X_2,X_3}}{p_{X_1}} = \frac{p_{X_1} p_{X_2|X_1} p_{X_3|X_1}}{p_{X_1}} \\ &= p_{X_2|X_1} p_{X_3|X_1}. \end{aligned} \tag{27}$$

By (17), X_2 and X_3 are conditionally independent given X_1.

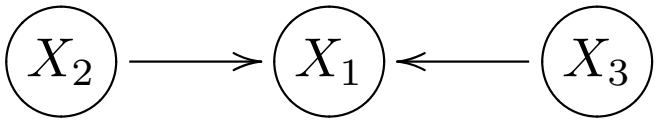
Diverging Connection – Example

In the belief network below, icy roads I influence the crashing of the drivers A and B. Then the crashing of both A and B is conditionally independent given icy roads.



Converging Connection

- A *converging connection* of three random variables X_1, X_2, and X_3 has the form



  The variable X_1 is *converging* or the *common effect*.

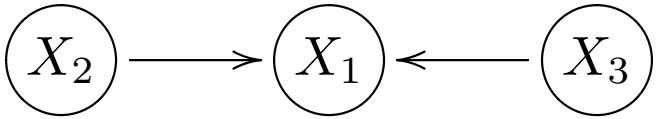
- As a belief network, the probability distribution p_{X_1,X_2,X_3} factors as

$$p_{X_1,X_2,X_3} = p_{X_2} p_{X_3} p_{X_1|X_2,X_3}. \tag{28}$$

- The random variables X_2 and X_3 may not be conditionally independent given X_1 – see Soft XOR gate (23).

Converging Connection

In a converging connection,



if X_1 is unknown, then X_2 and X_3 are independent.

Proof.

For all $x_2 \in \mathcal{X}_2$ and $x_3 \in \mathcal{X}_3$,

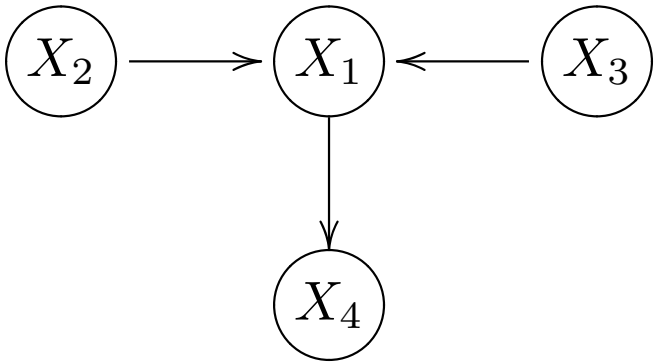
$$\begin{aligned} p_{X_2,X_3}(x_2,x_3) &= \sum_{x_1 \in \mathcal{X}_1} p_{X_1,X_2,X_3}(x_1,x_2,x_3) \\ &= \sum_{x_1 \in \mathcal{X}_1} p_{X_2}(x_2) p_{X_3}(x_3) p_{X_1|X_2,X_3}(x_1 \mid x_2,x_3) \\ &= p_{X_2}(x_2) p_{X_3}(x_3) \sum_{x_1 \in \mathcal{X}_1} p_{X_1|X_2,X_3}(x_1 \mid x_2,x_3) \\ &= p_{X_2}(x_2) p_{X_3}(x_3). \end{aligned}$$

Converging Connection – Example

In the belief network below, rain R and sprinkler S influence the wetness of grass A. If there is no information about the condition of grass A, then R and S are conditionally independent.

Converging Connection

Consider the converging connection



where X_1 is converging and has descendant X_4.
As a belief network, the probability distribution p_{X_1,X_2,X_3,X_4} factors as

$$p_{X_1,X_2,X_3,X_4} = p_{X_2} p_{X_3} p_{X_1|X_2,X_3} p_{X_4|X_1}. \tag{29}$$

Converging Connection

If in the above belief network X_1 and X_4 are unknown, then X_2 and X_3 are independent.

Proof.

For all $x_2 \in \mathcal{X}_2$ and $x_3 \in \mathcal{X}_3$,

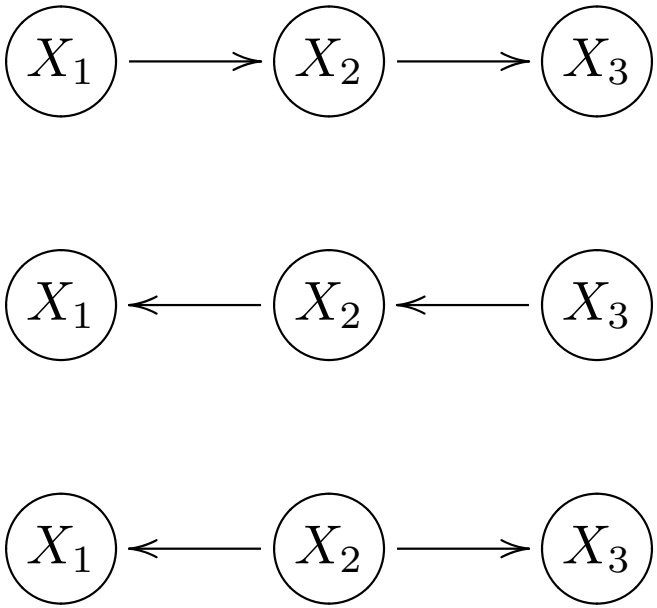
$$
\begin{aligned}
& p_{X_2,X_3}(x_2,x_3) \\
&= \sum_{x_1\in\mathcal{X}_1}\sum_{x_4\in\mathcal{X}_4} p_{X_1,X_2,X_3,X_4}(x_1,x_2,x_3,x_4) \\
&= \sum_{x_1\in\mathcal{X}_1}\sum_{x_4\in\mathcal{X}_4} p_{X_2}(x_2)p_{X_3}(x_3)p_{X_1|X_2,X_3}(x_1 \mid x_2,x_3)p_{X_4|X_1}(x_4 \mid x_1) \\
&= p_{X_2}(x_2)p_{X_3}(x_3)\sum_{x_1\in\mathcal{X}_1} p_{X_1|X_2,X_3}(x_1 \mid x_2,x_3)\sum_{x_4\in\mathcal{X}_4} p_{X_4|X_1}(x_4 \mid x_1) \\
&= p_{X_2}(x_2)p_{X_3}(x_3).
\end{aligned}
$$

□

More generally, if in a converging connection $X_2 \to X_1 \leftarrow X_3$ the converging random variable X_1 and all its descendants are not instantiated, the variables X_2 and X_3 are independent.

Serial and Diverging Connections

Consider the three belief networks



In each case, the joint probability distribution factors as

$$p_{X_1,X_2,X_3} = p_{X_1} p_{X_2|X_1} p_{X_3|X_2}. \tag{30}$$

However, the converging connection is of totally different nature!

Serial and Diverging Connections

In each case, the joint probability distribution factors as

$$p_{X_1,X_2,X_3} = p_{X_1} p_{X_2|X_1} p_{X_3|X_2}. \tag{31}$$

Proof.

- In the first network,
$$p_{X_1,X_2,X_3} = p_{X_1} p_{X_2|X_1} p_{X_3|X_2}.$$
- In the second network, apply Bayes' rule twice,
$$\begin{aligned} p_{X_1,X_2,X_3} &= \overbrace{p_{X_3} p_{X_2|X_3}}\, p_{X_1|X_2} = p_{X_3|X_2} \underbrace{p_{X_2} p_{X_1|X_2}} \\ &= p_{X_3|X_2} p_{X_2|X_1} p_{X_1}. \end{aligned}$$
- In the third network, apply Bayes' rule,
$$p_{X_1,X_2,X_3} = \overbrace{p_{X_2} p_{X_1|X_2}}\, p_{X_3|X_2} = p_{X_1} p_{X_2|X_1} p_{X_3|X_2}.$$

□

D-Separation

Given DAG $G = (V, E)$ with nodes labeled by random variables, $X, Y \in V$ and $S \subseteq V$ with $X, Y \notin S$.

- A trail τ between X and Y is *blocked* by S if there is a node Z on the trail such that one of the following holds:
  - Z is a converging node along the trail τ and neither Z nor any of its descendents belong to S, or
  - Z is a serial or diverging node along the trail τ and Z belongs to S.

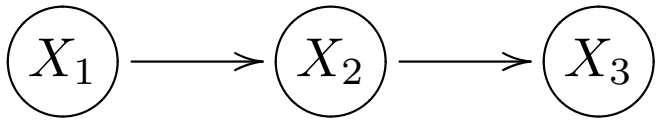
  Then the node Z is *blocking* the trail τ.
- The nodes X and Y are *d-separated* or *blocked* by S if all trails between X, Y are blocked by S, written

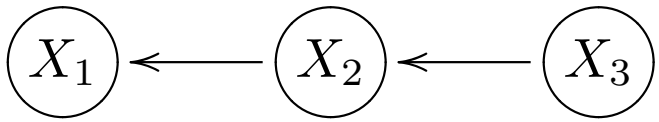
$$X \perp Y \mid_G S. \tag{32}$$

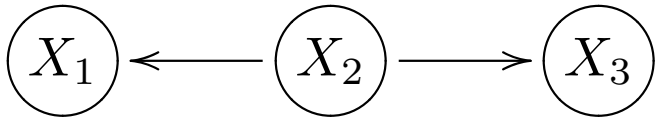
Otherwise, X and Y are *d-connected* or *non-blocked* by S.

D-Separation

Consider the three belief networks

serial 

serial 

diverging 

In each case, the trail between X_1 and X_3 is blocked by $S = \{X_2\}$ and so X_1 and X_3 are d-separated by S.

D-Separation

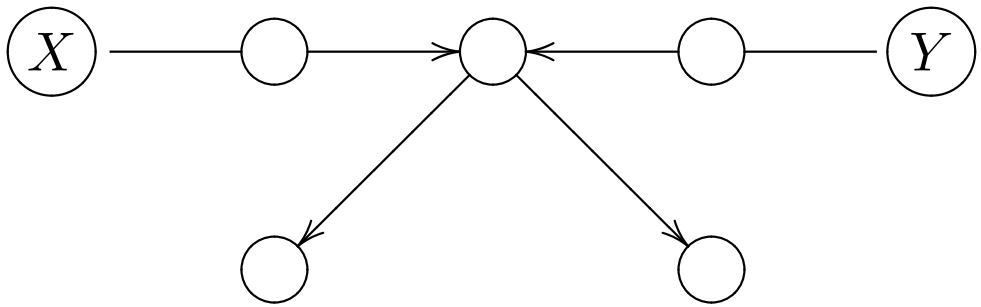
Schematic d-separation of X and Y by S with $Z \in S$:

serial

serial

diverging

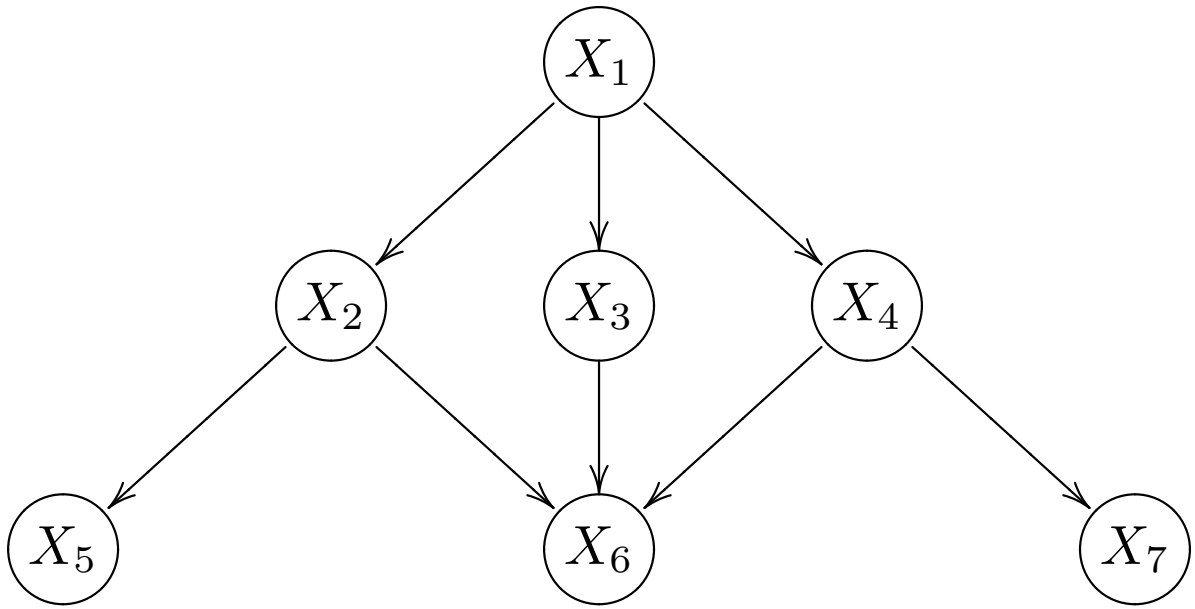
converging



D-Separation – Example

Consider the DAG below. Let $S = \{X_2, X_3, X_4\}$.

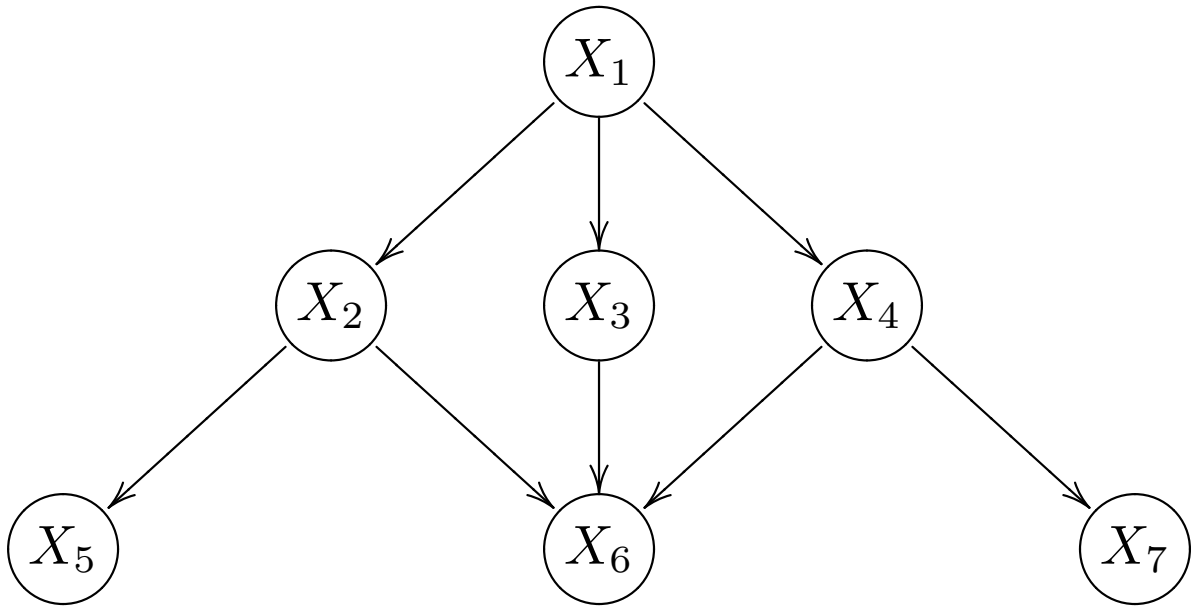
- X_1 and X_6 are d-separated given S; each trail between them has a serial node (X_2, X_3 or X_4) in S.
- X_5 and X_6 are d-separated given S; each trail between them contains the diverging node X_2 in S or a serial node (X_3 or X_4) in S.



D-Separation – Example (Cont'd)

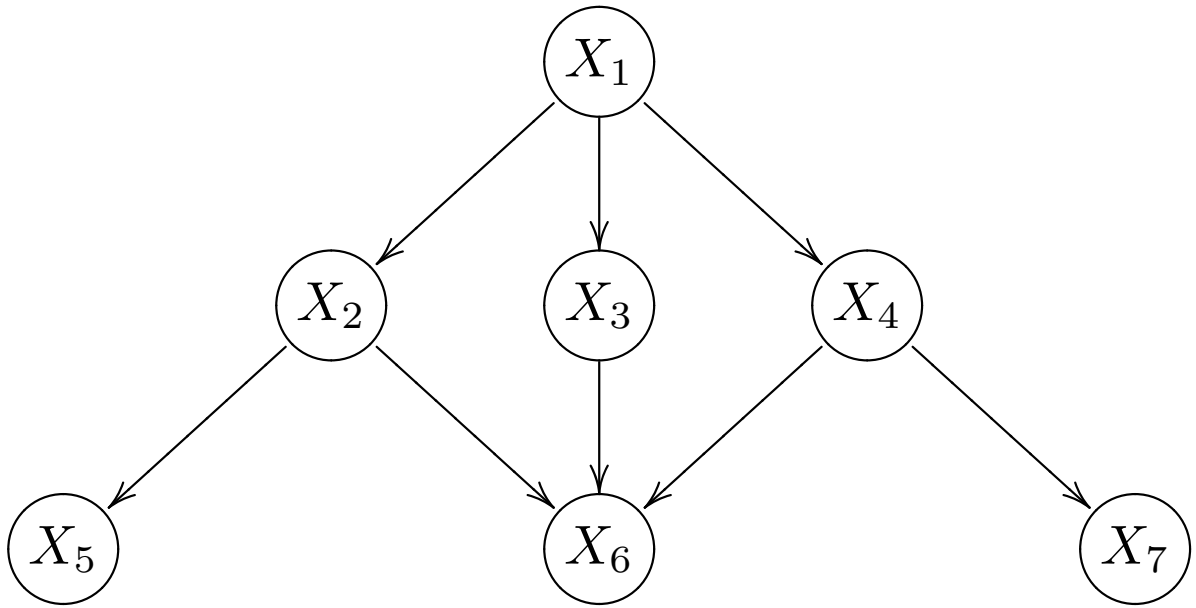
Consider the DAG below. Let $S = \{X_1, X_3\}$.

- X_2 and X_4 are d-separated given S; each trail between them contains the converging node X_6 not in S or the diverging node X_1 in S.



Markov Blanket

- The *Markov blanket* of a node X in a DAG G is the set of nodes composed of its parents, its children, and the parents of its children (not X itself).
- In the DAG below, the Markov blanket of X_3 consists of X_1 (parent), X_6 (child), and X_2, X_4 (parents of children).



Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Markov Blanket

Given a DAG $G = (V, E)$. Let X and Y be nodes in G and S be a subset of nodes not containing X and Y. If S contains the Markov blanket of X, the nodes X and Y are d-separated given S.

Proof.

Let τ be a trail between X and Y.

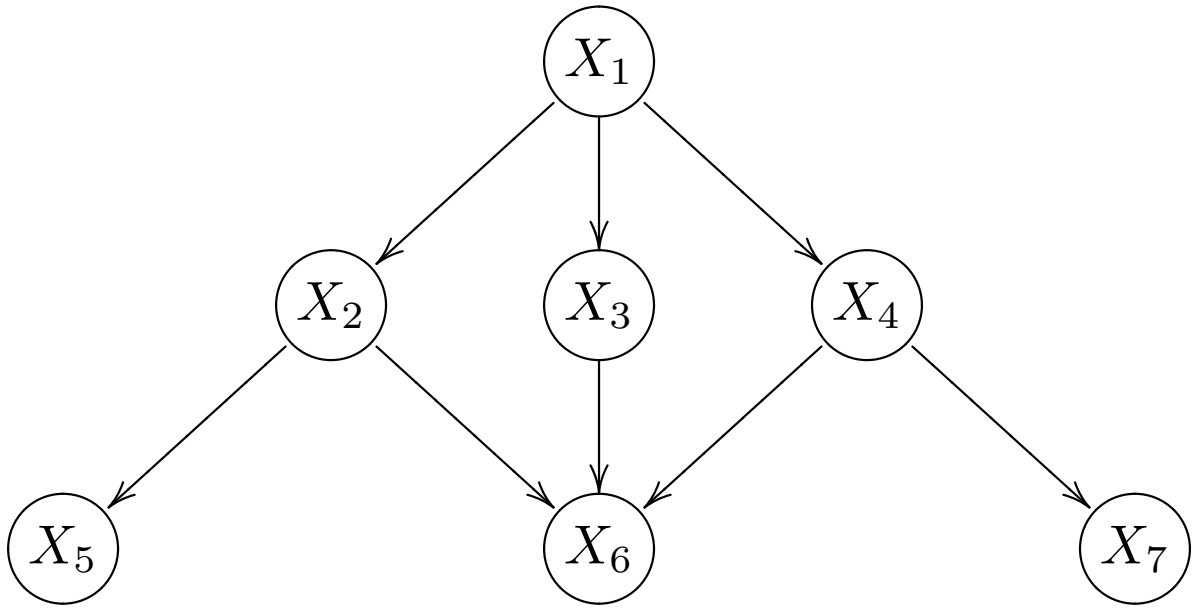
- If $X \to X_1 \to X_2 \ldots$, then $X_1 \in S$ is a serial node.
- If $X \leftarrow X_1 \to X_2 \ldots$, then $X_1 \in S$ is a diverging node.
- If $X \leftarrow X_1 \leftarrow X_2 \ldots$, then $X_1 \in S$ is a serial node.
- If $X \to X_1 \leftarrow X_2 \ldots$, then $X_1 \in S$ is a converging node. Since S contains the Markov blanket of X, we have $X_2 \in S$. The trail may proceed as follows:
  - If $X \to X_1 \leftarrow X_2 \to X_3 \ldots$, then X_2 is a diverging node.
  - If $X \to X_1 \leftarrow X_2 \leftarrow X_3 \ldots$, then X_2 is a serial node.

In each case, the trail τ is blocked by S as required. □

D-Separation – Example

Consider the DAG below.

- The Markov blanket of node X_3 is $S = \{X_1, X_2, X_4, X_6\}$.
- X_3 and X_5 (or X_3 and X_7) are d-separated given S.



Conditional Independence and D-Separation

Let (G, p) be a belief network with random variables $X_1, \ldots, X_n$. For any three disjoint subsets A, B, and S of $\{X_1, \ldots, X_n\}$ if A and B are d-separated given S, then A and B are conditionally independent given S; i.e.,

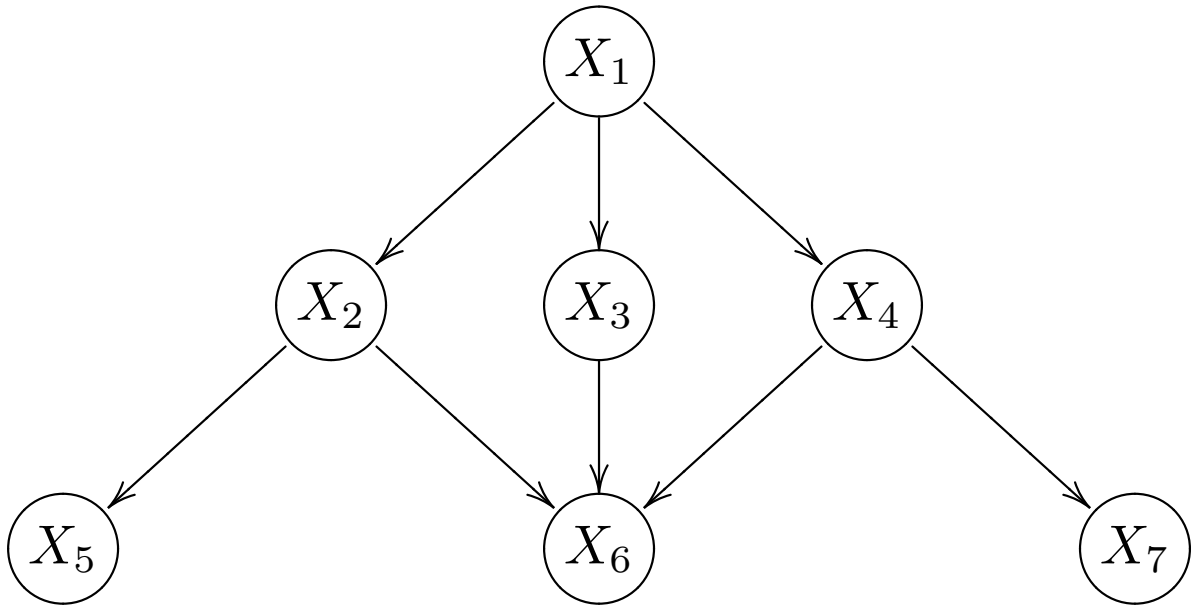
$$A \perp B \mid_G S \quad \Longrightarrow \quad A \perp B \mid S. \tag{33}$$

Conditional independence is purely probabilistic, while d-separation is purely graph-theoretical.

Conditional Independence and D-Separation – Example

Consider the DAG below. Let $S = \{X_1, X_3\}$.

- X_2 and X_4 are d-separated given S.
- Thus X_2 and X_4 are conditionally independent given S.



Conditional Independence and D-Separation

A belief network (G, p) with random variables $X_1, \ldots, X_n$ is *faithful* if for any three disjoint subsets A, B, and S of $\{X_1, \ldots, X_n\}$,

$$A \perp B \,|_G\, S \quad \Longleftrightarrow \quad A \perp B \mid S. \tag{34}$$

A faithful belief network is also called a *Bayesian network*.

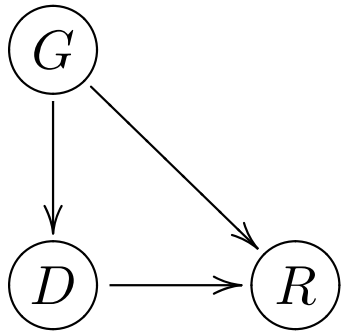
The design of Bayesian networks requires a good analysis of the probabilistic model.

Causality

- Causality comes into play in belief networks when the model contains no explicit temporal information.
- Causality refer to the connection of one process (cause) with another process (effect) and the second depends on the first.
- A joint probability function $p_{X,Y}$ can be written as $p_{X|Y}p_Y$; think of Y causes X.
- There are pitfalls, e.g., Simpson's paradox.

Simpson's Paradox

Consider the belief network of patients with gender (G), male or female, treated by drug (D), yes or no, who recover (R), yes or no.



The joint probability distribution factors as

$$p_{G,D,R} = p_{R|G,D} \cdot p_{D|G} \cdot p_G.$$

Simpson's Paradox (Cont'd)

Sample data:

	Recovered	Not recovered	Rate of recovery
Males			
Given drug	18	12	0.60
Not given drug	7	3	0.70
Females			
Given drug	2	8	0.20
Not given drug	9	21	0.30
Combined			
Given drug	20	20	0.50

Simpson's Paradox (Cont'd)

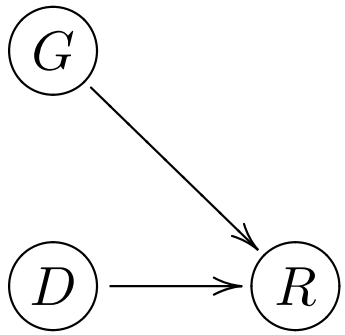
Does the drug cause increased recovery?

- If males or females are considered separately, the answer is no: $0.60 < 0.70$ for males and $0.20 < 0.30$ for females.
- If the gender information is ignored, more people will recover when given the drug than when given not:

$$\frac{18}{30} < \frac{7}{10} \text{ and } \frac{2}{10} < \frac{9}{30}, \text{ but } \frac{18+2}{30+10} > \frac{7+9}{10+30}.$$

Simpson's Paradox (Cont'd)

The model with (causal) link $G \to D$ is different from the model in which this link is removed:



giving the joint probability distribution

$$p_{G,D,R} = p_{R|G,D} \cdot p_G \cdot p_D.$$

Causality comes into play in belief networks with no explicit temporal information referring to the connection of one process (cause) with another process (effect).

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Markov Equivalence

- Skeleton and immoralities
- Essential graphs

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

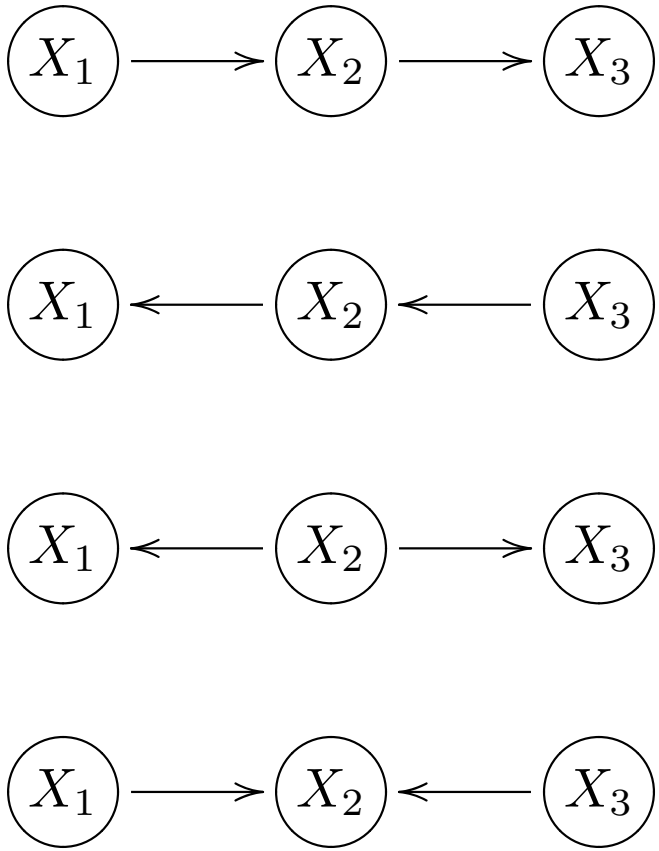
Markov Equivalence

Let $G = (V, E)$ and $G' = (V, E')$ be DAGs with node set V.

- $G \leq G'$ means that each pair of nodes X, Y which is d-separated by a set S of nodes of G with $X, Y \notin S$ is also d-separated by the same set in G'.
- G and G' are *Markov equivalent* if $G \leq G'$ and $G' \leq G$.
- Markov equivalence is an equivalence relation on the set of DAGs with common node set.

Markov Equivalence – Example

Consider the four DAGs

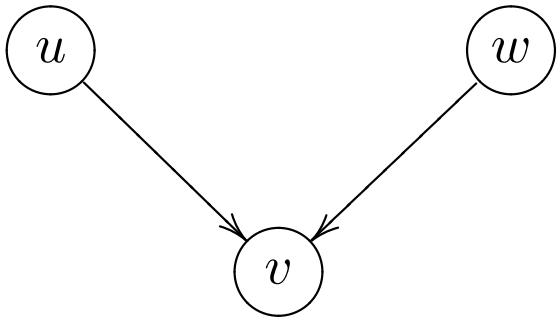


The first three ones are Markov equivalent with $X_1 \perp X_3 \mid_G X_2$, while the fourth one is not Markov equivalent to the other ones.

Skeleton and Immoralities

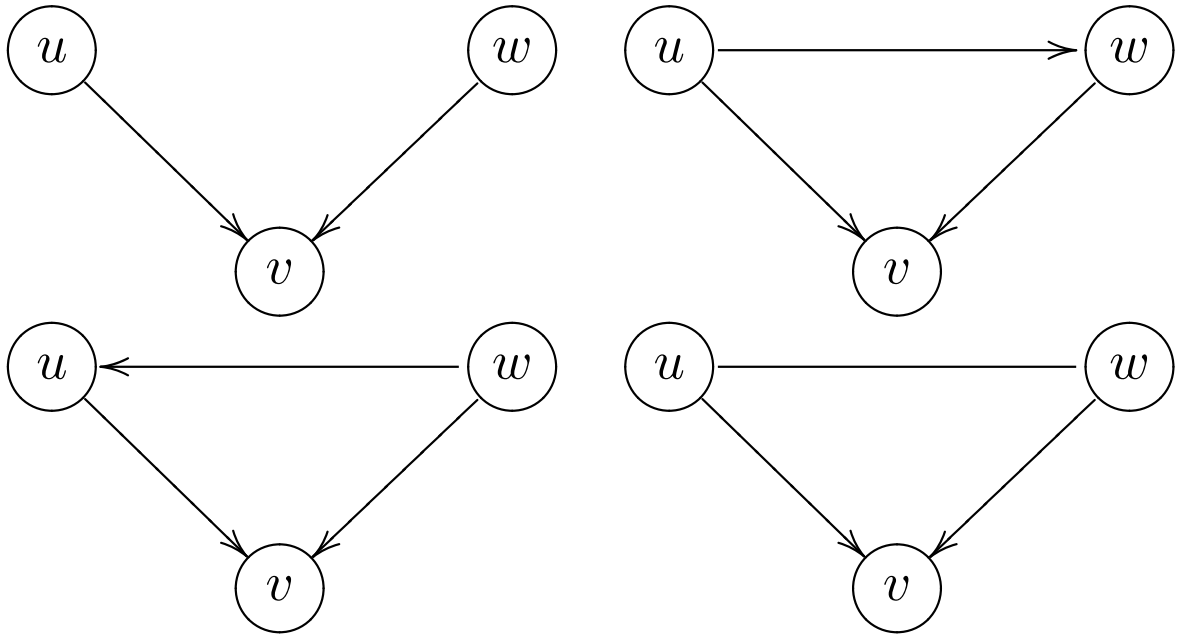
Let $G = (V, E)$ be a DAG.

- The *skeleton* of G is the (undirected) graph $G^u = (V, E^u)$ resulting from G by making all edges undirected.
- An *immorality* in G is a triple (u, v, w) of three distinct nodes in G such that $u \to v$ and $w \to v$, but the edges $u \to w$, $w \to u$, and $u - w$ do not belong to G.



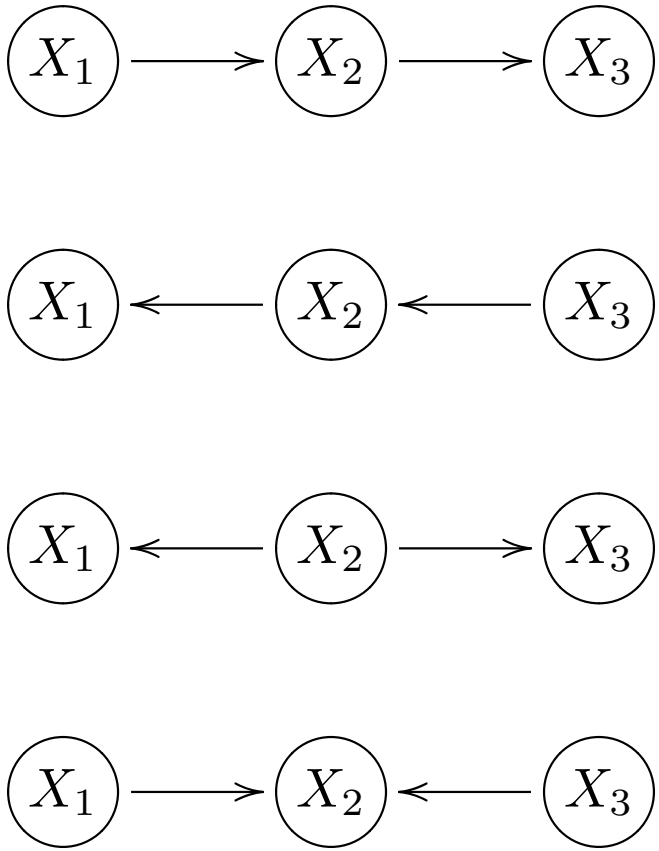
Skeleton and Immoralities – Example

An immorality (u, v, w) and three non-immoralities:



Skeleton and Immoralities – Example

Consider the four DAGs



All DAGs have the same skeleton $X_1 - X_2 - X_3$ but only the last one has an immorality (X_1, X_2, X_3).

Skeleton and Immoralities

Given DAGs $G = (V, E)$ and $G' = (V, E')$ with same skeleton and immoralities. Then for each node $v \in V$, the Markov blanket of v in G is equal to the Markov blanket of v in G'.

*Proof.

Let u be a parent of v in G. Since G and G' have the same skeleton, the node u is a parent or a child of v in G'. Thus u is in the Markov blanket of v in G'. The same holds if u is a child of v in G.

Let u be a parent of a child w of v in G. If u and v are connected by a directed edge, we are back in the above case. Otherwise, we have an immorality $v \to w \leftarrow u$ in G. Since G and G' have the same immoralities, $v \to w \leftarrow u$ is also an immorality in G'. Hence, the node u lies also in the Markov blanket of v in G'. □

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Markov Equivalence

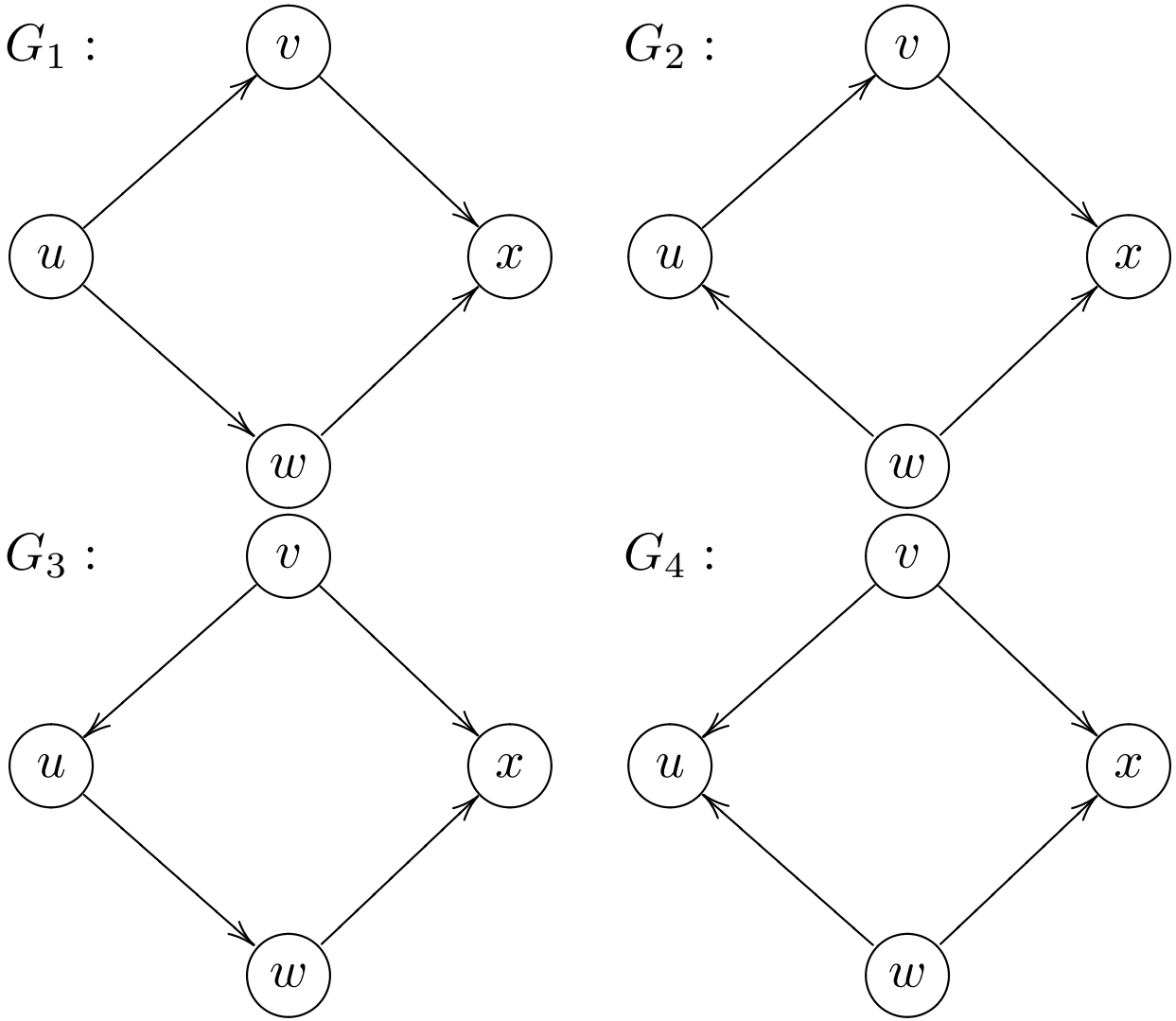
Two DAGs have the same skeleton and the same immoralities iff they are Markov equivalent.

*Proof.

Let G and G' be two DAGs with the same skeleton and immoralities. Then for any two nodes u and v and any subset S of nodes containing the Markov blanket of v but not containing v and u, we have that u and v are d-separated in G given S if and only if u and v are d-separated in G' given S. It follows that G and G' have the same independence structure and hence are Markov equivalent.
Conversely, if two DAGs have the same d-dependence structure, they have the same skeleton and the same immoralities. □

Markov Equivalence – Example

Consider the four DAGs



Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

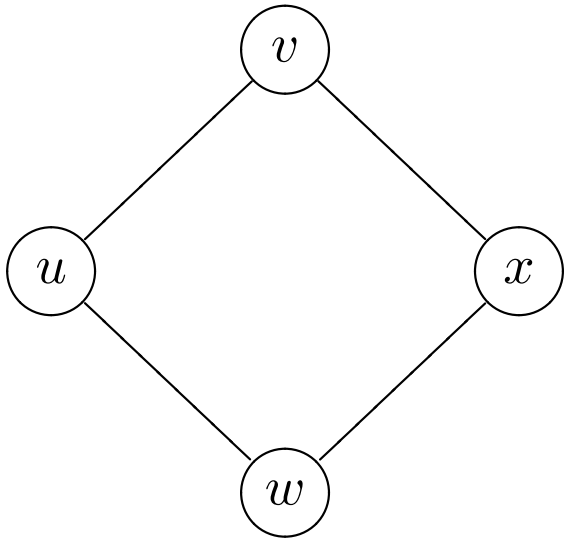
Conditional Independence

Markov Equivalence

Using R

Markov Equivalence – Example (Cont'd)

All DAGs have the skeleton



and they have the immorality (v, x, w) in common.

- The DAGs G_1, G_2, and G_3 have no further immoralities and so are Markov equivalent.
- The DAG G_4 has an additional immorality (v, u, w) and so is not Markov equivalent to the other ones.

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

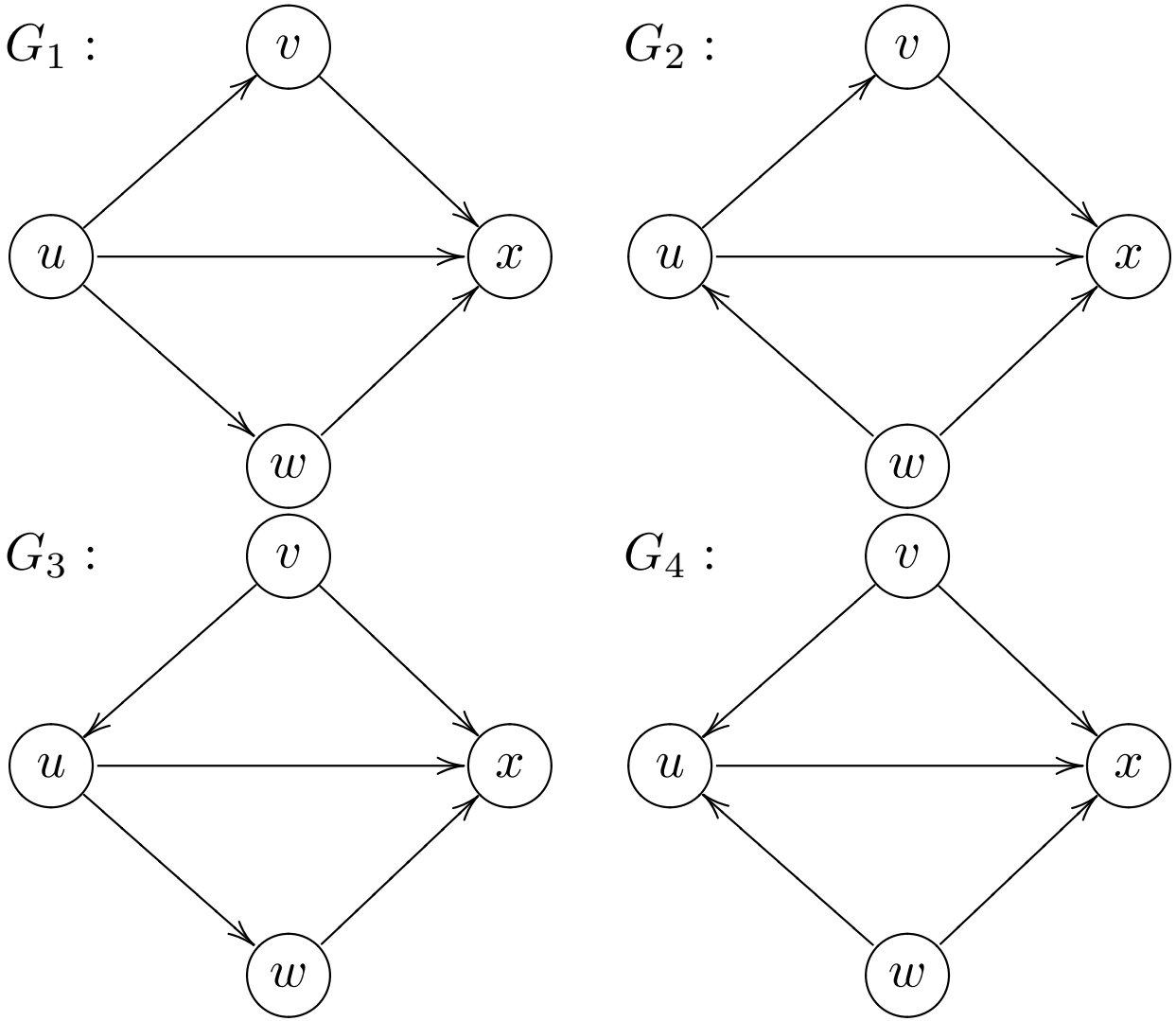
Essential Graphs

The *essential graph* of a DAG G is a graph G^* such that

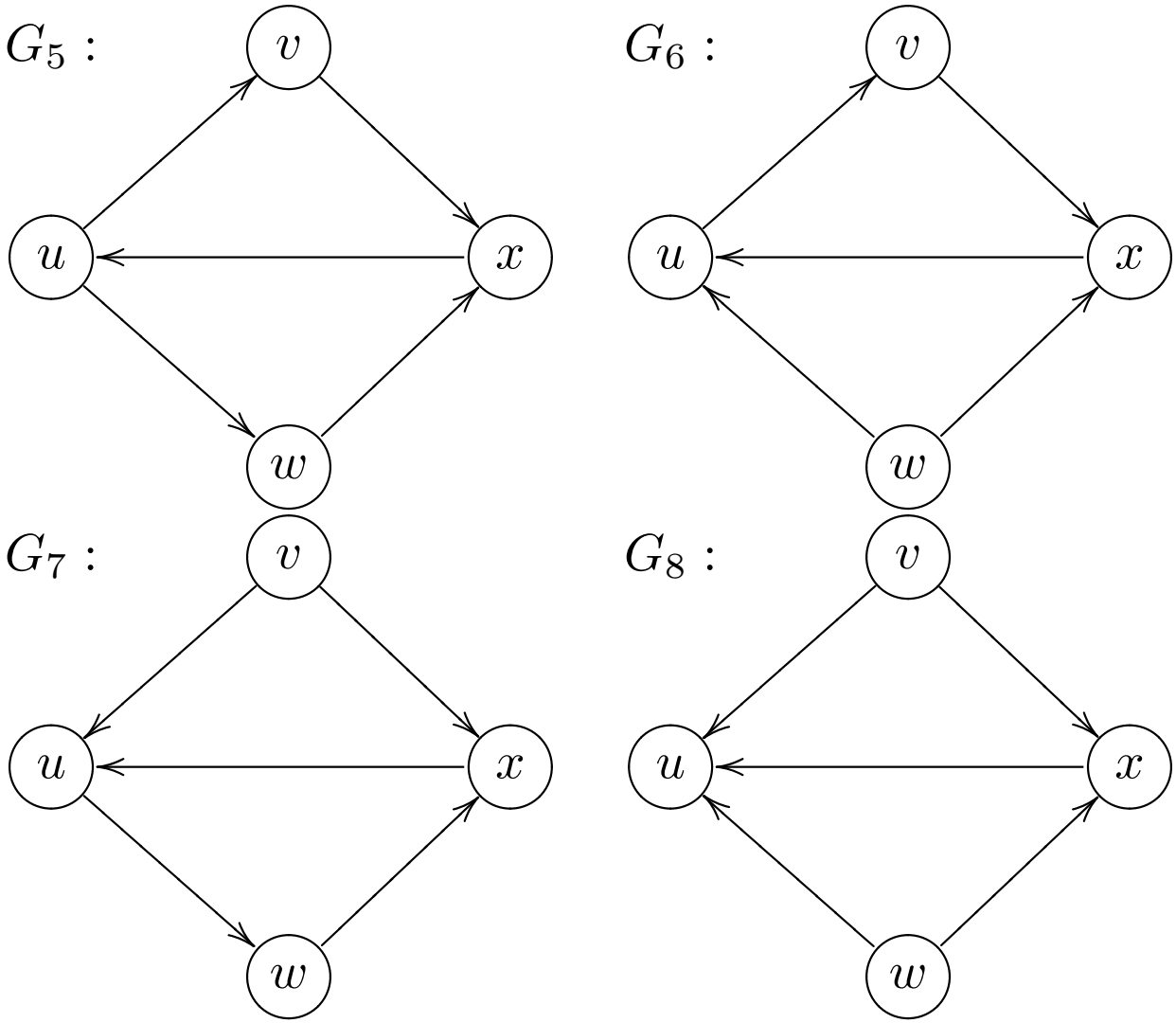
- G and G^* have same skeleton.
- An edge in G^* is directed if it is directed with the same orientation in each DAG which is Markov equivalent to G. Otherwise, the edge is undirected. The directed edges in G^* are the *essential* edges of G.

Essential Graphs – Example

Consider the eight digraphs



Essential Graphs – Example (Cont'd)

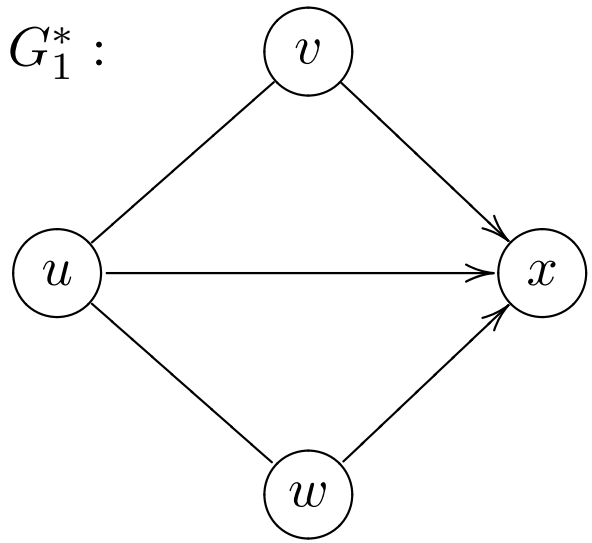


Essential Graphs – Example (Cont'd)

- All digraphs have the same skeleton and share the immorality (v, x, w).
- The directed edges $v \to x$ and $w \to x$ are essential in G_1.
- The other three edges of G_1 can be oriented in $2^3 = 8$ possible ways.
- From these eight digraphs, only five are acyclic (i.e., G_1, G_2, G_3, G_4, G_8) and from these five digraphs, only three (i.e., G_1, G_2, G_3) have the same immorality as in G_1 and no further one.
- The Markov-equivalence class of G_1 is $\{G_1, G_2, G_3\}$.
- The directed edge $u \to x$ occurs in each member of the Markov-equivalence class of G_1 and so is essential too.

Essential Graphs – Example (Cont'd)

Essential graph of G_1:



Belief Networks in R

The package `bnlearn` is an `R` library (2009) for Bayesian networks. The library is loaded by the command

```
> library(bnlearn)
```

A graph structure (`bn` object) can be generated in three different ways:

- edge (arc) set of edge-induced graph,
- adjacency matrix, and
- model formula.

Moreover, random graph structures can be created.

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Graph Generation

Generation of an empty graph with a given node set:

```
> e = empty.graph(LETTERS[1:6])
> class(e)
[1] "bn"
> e
  Random/Generated Bayesian network
  model:
   [A][B][C][D][E][F]
  nodes:                                6
  arcs:                                 0
    undirected arcs:                    0
    directed arcs:                      0
  average markov blanket size:          0.00
  average neighborhood size:            0.00
  average branching factor:             0.00
  generation algorithm:                 Empty
```

Multiple empty graphs can be specified by the `num` argument.

Graph Generation

Generation of graph structure from an edge set:

```
> arc.set = matrix(c("A", "B", "A", "C", "D", "C"),
+                  ncol = 2, byrow = TRUE,
+                  dimnames = list(NULL, c("from", "to")))
> arc.set
     from to
[1,] "A"  "B"
[2,] "A"  "C"
[3,] "D"  "C"
```

Graph Generation

The created arc set can be assigned to a bn object:

```
> arcs(e) = arc.set
> e
  Random/Generated Bayesian network
  model:
   [A][D][E][F][B|A][C|A:D]
  nodes:                                6
  arcs:                                 3
    undirected arcs:                    0
    directed arcs:                      3
  average markov blanket size:          1.33
  average neighborhood size:            1.00
  average branching factor:             0.50
  generation algorithm:                 Empty
```

Undirected arcs can be specified by including arcs in both directions.

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Graph Generation

Generation of graph structure by using an adjacency matrix:

```
> adj = matrix(0L, ncol=6, nrow = 6,
+              dimnames = list(LETTERS[1:6], LETTERS[1:6]))
> adj["A", "B"] = 1L
> adj["A", "C"] = 1L
> adj["D", "C"] = 1L
> adj
  A B C D E F
A 0 1 1 0 0 0
B 0 0 0 0 0 0
C 0 0 0 0 0 0
D 0 0 1 0 0 0
E 0 0 0 0 0 0
F 0 0 0 0 0 0
```

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Graph Generation

The generated arc set can be assigned to a bn object:

```
> amat(e) = adj
> e
  Random/Generated Bayesian network
  model:
   [A][D][E][F][B|A][C|A:D]
  nodes:                                6
  arcs:                                 3
    undirected arcs:                    0
    directed arcs:                      3
  average markov blanket size:          1.33
  average neighborhood size:            1.00
  average branching factor:             0.50
  generation algorithm:                 Empty
```

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Graph Generation

Generation of a graph structure by a model formula:

```
> model2network("[A][B][C|A:B][D|C][E|C:D][F|A:B:C]")
  Random/Generated Bayesian network
  model:
   [A][B][C|A:B][D|C][E|C:D][F|A:B:C]
  nodes:                                6
  arcs:                                 8
    undirected arcs:                    0
    directed arcs:                      8
  average markov blanket size:          3.00
  average neighborhood size:            2.67
  average branching factor:             1.33
  generation algorithm:                 Empty
```

Graph Generation

Generation of a graph structure using the empty (or an existing) graph:

```
> e = empty.graph(LETTERS[1:6])
> modelstring(e) = "[A][B|A][C|B][D|C][E|D][F|E]"
> e
  Random/Generated Bayesian network
  model:
   [A][B|A][C|B][D|C][E|D][F|E]
  nodes:                                6
  arcs:                                 5
    undirected arcs:                    0
    directed arcs:                      5
  average markov blanket size:          1.67
  average neighborhood size:            1.67
  average branching factor:             0.83
  generation algorithm:                 Empty
```

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Graph Generation

Generation of a random graph from an ordered node set. The arcs are sampled independently with a given probability of inclusion.

```
> g = random.graph(LETTERS[1:6], prob = 0.1)
> g
  Random/Generated Bayesian network
  model:
   [A][B][C][E][D|A:C][F|B:E]
  nodes:                                6
  arcs:                                 4
    undirected arcs:                    0
    directed arcs:                      4
  average markov blanket size:          2.00
  average neighborhood size:            1.33
  average branching factor:             0.67
  generation algorithm:                 Full Ordering
  arc sampling probability:             0.1
```

Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

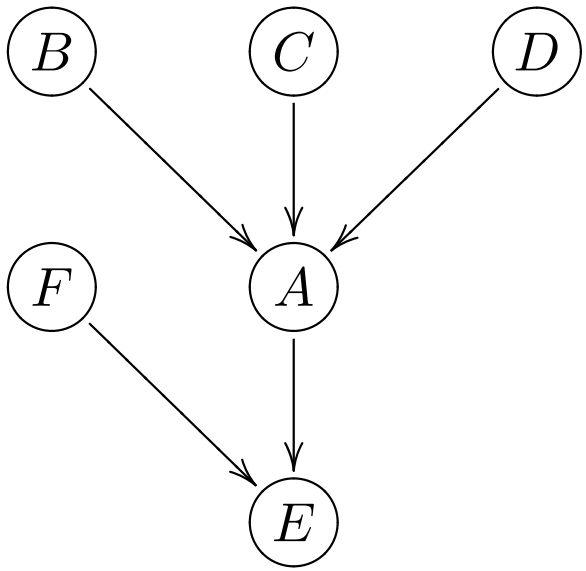
Markov Equivalence

Using R

Graph Plotting

Plotting of a DAG associated with a belief network using the `Rgraphviz` package:

```
> belief.net = model2network(
+                 "[B][C][D][F][A|B:C:D][E|A:F]")
> graphviz.plot(belief.net)
```



Soft Computing

K.-H. Zimmermann

Contents

Probabilities

Graph Concepts

Belief Networks

Conditional Independence

Markov Equivalence

Using R

Graph Information

Extracting information from a graph structure:

```
> parents(belief.net, "A")
[1] "B" "C" "D"
> children(belief.net, "A")
[1] "E"
> nbr(belief.net, "A")
[1] "B" "C" "D" "E"
> mb(belief.net, "A")
[1] "B" "C" "D" "E" "F"
> root.nodes(belief.net)
[1] "B" "C" "D" "F"
> leaf.nodes(belief.net)
[1] "E"
```

Part II

Hidden Markov Model

Contents

- Belief network
- Learning conditional probabilities (fully observed and hidden model)
- Probabilistic inference of output marginals
- Example: Finding CpG islands
- Using R

Knowledge

- Belief network of HMM
- Probabilistic inference
- Learning conditional probability distributions
- Applications of HMM

Skills

- Viterbi algorithm
- EM algorithm
- *Baum-Welch algorithm

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Belief Network

- Structure of belief network
- Joint probability distribution
- Example: dishonest casino dealer

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

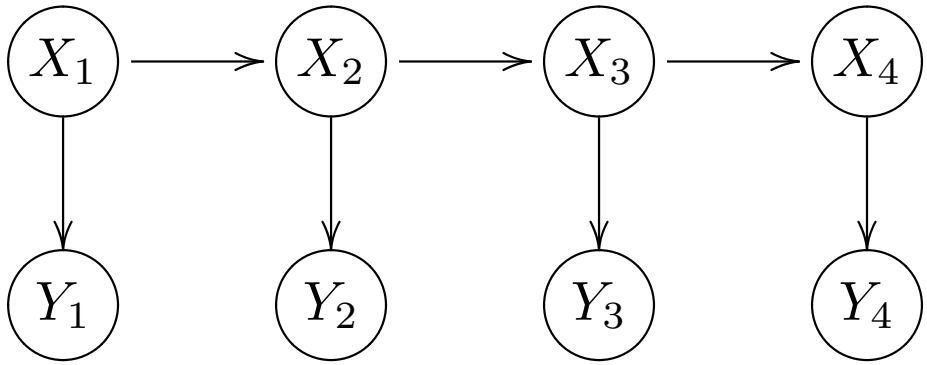
Belief Network

Given integer $n \geq 1$.

- $X_1, \ldots, X_n$ random variables over finite state set Σ.
- $Y_1, \ldots, Y_n$ random variables over finite state set Σ'.
- Σ and Σ' are the *state* and *output* alphabets, resp.
- Put $l = |\Sigma|$ and $l' = |\Sigma'|$.
- Random vectors $X = (X_1, \ldots, X_n)$ and $Y = (Y_1, \ldots, Y_n)$ have state sets Σ^n and Σ'^n, resp.

Belief Network

Belief network (G_n, p) for $n = 4$:



Joint probability distribution factors w.r.t. the network:

$$\begin{aligned} p_{X,Y} &= p_{X_1,\ldots,X_4,Y_1,\ldots,Y_4} \\ &= p_{X_1} p_{Y_1|X_1} p_{X_2|X_1} p_{Y_2|X_2} p_{X_3|X_2} p_{Y_3|X_3} p_{X_4|X_3} p_{Y_4|X_4}. \end{aligned} \tag{35}$$

Belief Network

- Joint probability distribution factors w.r.t. the network:

$$\begin{aligned} p_{X,Y} &= p_{X_1,\ldots,X_n,Y_1,\ldots,Y_n} \\ &= p_{X_1} p_{Y_1|X_1} p_{X_2|X_1} \cdots p_{X_n|X_{n-1}} p_{Y_n|X_n}. \end{aligned} \tag{36}$$

- For simplicity, initial distribution p_{X_1} is uniform,

$$p_{X_1}(x) = \frac{1}{l}, \quad x \in \Sigma. \tag{37}$$

- The network is *homogeneous*, i.e.,
  - $p_{X_{i+1}|X_i}$ for $1 \leq i \leq n-1$ are all equal, and
  - $p_{Y_i|X_i}$ for $1 \leq i \leq n$ are all equal.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Belief Network

- State transition probabilities

$$\theta_{x,x'} = p_{X_{i+1}|X_i}(x'|x), \quad x, x' \in \Sigma,\ 1 \leq i \leq n-1, \tag{38}$$

- Emission probabilities

$$\theta'_{x,y} = p_{Y_i|X_i}(y|x), \quad x \in \Sigma, y \in \Sigma',\ 1 \leq i \leq n. \tag{39}$$

- Joint probability distribution

$$\begin{aligned} & p_{X_1,\ldots,X_n,Y_1,\ldots,Y_n}(x_1,\ldots,x_n,y_1,\ldots,y_n) \\ = \quad & \frac{1}{l}\theta'_{x_1,y_1}\theta_{x_1,x_2}\theta'_{x_2,y_2}\theta_{x_2,x_3}\ldots\theta_{x_{n-1},x_n}\theta'_{x_n,y_n}. \end{aligned} \tag{40}$$

Start with state x_1, emit y_1, move from x_1 to x_2, emit y_2, move from x_2 to x_3, emit y_3 and so on. At the end, move from x_{n-1} to x_n, emit y_n.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Belief Network – Example

Dishonest casino dealer tosses fair (F) or loaded (L) coin with outcomes head (h) or tail (t).

- Loaded coin has probability 0.75 to get head and 0.25 to get tail.
- Dealer changes coins with probability 0.1.
- State alphabet $\Sigma = \{F, L\}$ and emission alphabet $\Sigma' = \{h, t\}$.
- Conditional probabilities

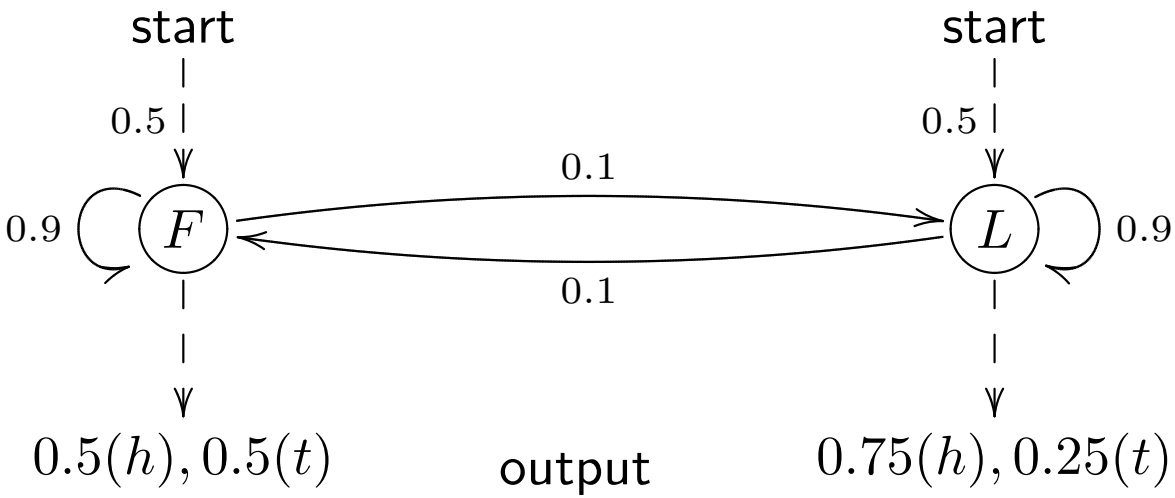
$$\theta_{F,F} = 0.9, \quad \theta_{F,L} = 0.1, \quad \theta_{L,F} = 0.1, \quad \theta_{L,L} = 0.9$$

and

$$\theta'_{F,h} = 0.5, \quad \theta'_{F,t} = 0.5, \quad \theta'_{L,h} = 0.75, \quad \theta'_{L,t} = 0.25.$$

Belief Network – Example (Cont'd)

State transition graph of casino model:



Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Belief Network – Example (Cont'd)

Joint probability distribution

$$\begin{aligned} & p_{X_1,\ldots,X_4,Y_1,\ldots,Y_4}(FFLF, htht) \\ = & \frac{1}{2}\theta'_{F,h}\theta_{F,F}\theta'_{F,t}\theta_{F,L}\theta'_{L,h}\theta_{L,F}\theta'_{F,t} \\ = & 0.5 \cdot 0.5 \cdot 0.9 \cdot 0.5 \cdot 0.1 \cdot 0.75 \cdot 0.1 \cdot 0.5 = 0.00042. \end{aligned}$$

Given a sequence of coin tosses we wish to determine when the dealer has used the biased and fair coins.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

*Bernoulli Experiment

Total probability of n coin tosses (trail) with k heads,

$$P(S_n = k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad 0 \leq k \leq n, \tag{41}$$

where $S_n = \sum_{i=1}^{n} X_i$ and X_i is the binary random variable with success probability $P(X_i = 1) = p$ (head).

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

*Statistical Hypothesis Testing I

Left-sided test:

- Null hypothesis $H_0 : p = p_0$
- Alternative hypothesis $H_1 : p < p_0$
- Accept H_0: $A = \{k+1, \ldots, n\}$
- Reject H_0: $\overline{A} = \{0, \ldots, k\}$
- Type 1 error: $P(S_n \leq k) \leq \alpha$; H_0 is rejected but correct.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

*Statistical Hypothesis Testing I – Example

Left-sided test: ideal six-sided die

- Null hypothesis $H_0 : p = \frac{1}{6}$
- Alternative hypothesis $H_1 : p < \frac{1}{6}$
- Accept H_0: $A = \{k+1, \ldots, 50\}$,
- Reject H_0: $\overline{A} = \{0, \ldots, k\}$
- Type 1 error: $P(S_{50} \leq k) \leq 0.05$ gives $k = 3$ (tabulated)
- Sampling $n = 50$ times has given 4 times the outcome 6
- $4 \in A$ and so H_0 cannot be rejected

*Statistical Hypothesis Testing II

Right-sided test:

- Null hypothesis $H_0 : p = p_0$
- Alternative hypothesis $H_1 : p > p_0$
- Accept H_0: $A = \{0, \ldots, k\}$
- Reject H_0: $\overline{A} = \{k+1, \ldots, n\}$
- Type 1 error: $P(S_n \geq k+1) \leq \alpha$ or $P(S_n \leq k) \geq 1 - \alpha$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Learning in Fully Observed Markov Model

- Structure of fully observed model
- Sufficient statistic
- Maximum likelihood estimate

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Learning in Fully Observed Markov Model

Fully observed Markov model: Observer has access to both state sequence and associated output (emission) sequence.

- Objective is to learn or estimate the conditional probabilities.
- Sample data consist of pairs of state and output sequences.
- Maximum likelihood estimation solves the problem.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Fully Observed Markov Model

Given integer $n \geq 1$.

- Belief network (G_n, p) of HMM with joint probability distribution

$$\begin{aligned} p_{X,Y} &= p_{X_1,\ldots,X_n,Y_1,\ldots,Y_n} \\ &= p_{X_1} p_{Y_1|X_1} p_{X_2|X_1} \cdots p_{X_n|X_{n-1}} p_{Y_n|X_n}. \end{aligned} \tag{42}$$

- Parameters for transition probabilities

$$\Theta = \{\theta = (\theta_{x,x'}) \mid \theta_{x,x'} \geq 0, \sum_{x'} \theta_{x,x'} = 1\} \tag{43}$$

and emission probabilities

$$\Theta' = \{\theta' = (\theta'_{x,y}) \mid \theta'_{x,y} \geq 0, \sum_{y} \theta'_{x,y} = 1\}. \tag{44}$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Fully Observed Markov Model

- Parameters for state transition probabilities

$$\theta_{x,x'} = p_{X_{i+1}|X_i}(x'|x), \quad x, x' \in \Sigma, 1 \leq i \leq n-1, \tag{45}$$

and emission probabilities

$$\theta'_{x,y} = p_{Y_i|X_i}(y|x), \quad x \in \Sigma, y \in \Sigma', 1 \leq i \leq n. \tag{46}$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Fully Observed Markov Model

Estimate the transition probabilities using a sample set.

- Given collection of N independent sequence pairs called *database*,

$$D = (d_1, \ldots, d_N). \tag{47}$$

- The *r-th sample*

$$d_r = (x^{(r)}, y^{(r)}) \tag{48}$$

is given by state sequence $x^{(r)} = x_1^{(r)} \ldots x_n^{(r)} \in \Sigma^n$ and output sequence $y^{(r)} = y_1^{(r)} \ldots y_n^{(r)} \in \Sigma'^n$, $1 \leq r \leq N$.

- Joint probability of sample d_r depending on the parameters,

$$p_{X,Y|\Theta,\Theta'}(d_r \mid \theta, \theta') = \tag{49}$$
$$\frac{1}{l}\theta'_{x_1^{(r)},y_1^{(r)}}\theta_{x_1^{(r)},x_2^{(r)}}\theta'_{x_2^{(r)},y_2^{(r)}}\theta_{x_2^{(r)},x_3^{(r)}}\ldots\theta_{x_{n-1}^{(r)},x_n^{(r)}}\theta'_{x_n^{(r)},y_n^{(r)}}.$$

Fully Observed Markov Model

- $u_{x,y}$ is the number of times the pair $(x, y) \in \Sigma^n \times \Sigma'^n$ is observed in the sample set.
- Thus

$$\sum_{(x,y)} u_{x,y} = N. \tag{50}$$

- Likelihood function

$$\begin{aligned} L(\theta, \theta') &= \prod_{r=1}^{N} p_{X,Y|\Theta,\Theta'}(d_r \mid \theta, \theta') \\ &= \prod_{(x,y)} p_{X,Y|\Theta,\Theta'}(x, y \mid \theta, \theta')^{u_{x,y}}. \end{aligned} \tag{51}$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Fully Observed Markov Model – Example

In view of the occasionally dishonest casino with $n = 4$, given $N = 8$ samples

$$\begin{array}{lll} d_1 = FFFF, hhhh & d_2 = FFFL, hhtt & d_3 = FFFF, hhhh \\ d_4 = FFFL, hhtt & d_5 = FFFF, hhhh & d_6 = FFFL, hhtt \\ d_7 = LLLL, hhhh & d_8 = LLLL, hhtt. & \end{array}$$

Then

$$\begin{array}{ll} u_{FFFF,hhhh} = 3, & u_{FFFL,hhtt} = 3, \\ u_{LLLL,hhhh} = 1, & u_{LLLL,hhtt} = 1. \end{array}$$

Likelihood function:

$$\begin{aligned} L(\theta, \theta') &= \prod_{r=1}^{N} p_{X,Y}(d_r) \\ &= p_{X,Y}(FFFF, hhhh)^3 \cdot p_{X,Y}(FFFL, hhtt)^3 \\ &\quad \cdot p_{X,Y}(LLLL, hhhh) \cdot p_{X,Y}(FFFF, hhtt) \\ &= \frac{1}{2^8} \theta_{FF}^{15} \theta_{FL}^{3} \theta_{LF}^{0} \theta_{LL}^{6} \theta'^{18}_{Fh} \theta'^{3}_{Ft} \theta'^{6}_{Lh} \theta'^{5}_{Lt}. \end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Fully Observed Markov Model

- $v_{a,b}$ is the number of occurrences of $(a,b) \in \Sigma^2$ as a consecutive pair in the sequence x in any of the observed pairs (x,y): $x = \ldots ab \ldots$.
- $v'_{a,c}$ is the number of occurrences of $(a,c) \in \Sigma \times \Sigma'$ at the same positions in the sequences x and y in any of the observed pairs (x,y): $\binom{x}{y} = \ldots \binom{a}{c} \ldots$.
- Likelihood function (up to a constant)

$$L(\theta, \theta') = \prod_{a,b} \theta_{a,b}^{v_{ab}} \cdot \prod_{a,c} {\theta'}_{a,c}^{v'_{ac}}. \tag{52}$$

- Log-likelihood function $\ell(\theta, \theta') = \log L(\theta, \theta')$

$$\ell(\theta, \theta') = \sum_{a,b} v_{a,b} \log(\theta_{a,b}) + \sum_{a,c} v'_{a,c} \log(\theta'_{a,c}). \tag{53}$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Fully Observed Markov Model

- The *sufficient statistic* of the model is given by the data

$$v = ((v_{a,b}), (v'_{a,c})). \tag{54}$$

The likelihood function (52) is seen through these data!

- *Linear transformation* yields v from given data $u = (u_{x,y})$,

$$v = A_{(l,l'),n} \cdot u, \tag{55}$$

where $A_{(l,l'),n}$ is an integral matrix.

- $A_{(l,l'),n}$ has $d = l \cdot l + l \cdot l'$ rows labeled by the pairs $(a, b) \in \Sigma^2$ and in turn by the pairs $(a, c) \in \Sigma \times \Sigma'$.
- $A_{(l,l'),n}$ has $m = l^n \cdot l'^n$ columns labeled by the pairs of words $x \in \Sigma^n$ and $y \in \Sigma'^n$.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Fully Observed Markov Model – Example (Cont'd)

In view of the occasionally dishonest casino with $n = 4$, given $N = 8$ samples

$$\begin{array}{lll} d_1 = FFFF, hhhh & d_2 = FFFL, hhtt & d_3 = FFFF, hhhh \\ d_4 = FFFL, hhtt & d_5 = FFFF, hhhh & d_6 = FFFL, hhtt \\ d_7 = LLLL, hhhh & d_8 = LLLL, hhtt. & \end{array}$$

Then

$$\begin{array}{ll} u_{FFFF,hhhh} = 3, & u_{FFFL,hhtt} = 3, \\ u_{LLLL,hhhh} = 1, & u_{LLLL,hhtt} = 1. \end{array}$$

Likelihood function

$$L(\theta, \theta') = \frac{1}{2^8} \theta_{FF}^{15} \theta_{FL}^{3} \theta_{LF}^{0} \theta_{LL}^{6} \theta'^{18}_{Fh} \theta'^{3}_{Ft} \theta'^{6}_{Lh} \theta'^{5}_{Lt}.$$

We have

$$\begin{array}{llll} v_{FF} = 15, & v_{FL} = 3, & v_{LF} = 0, & v_{LL} = 6, \\ v'_{Fh} = 18, & v'_{Ft} = 3, & v'_{Lh} = 6, & v'_{Lt} = 5, \end{array}$$

and $v = A_{(2,2),4} \cdot u$.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Fully Observed Markov Model – Example (Cont'd)

In view of the occasionally dishonest casino with $n = 4$, the 8×256 matrix $A_{(2,2),4}$ is as follows,

$$A^t_{(2,2),4} = \begin{array}{c} \\ FFFF, hhhh \\ FFFF, hhht \\ FFFF, hhth \\ FFFF, hhtt \\ \vdots \\ LLLL, tttt \end{array} \begin{array}{c} \begin{array}{cccccccc} FF & FL & LF & LL & Fh & Ft & Lh & Lt \end{array} \\ \begin{pmatrix} 3 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 3 & 1 & 0 & 0 \\ 3 & 0 & 0 & 0 & 3 & 1 & 0 & 0 \\ 3 & 0 & 0 & 0 & 2 & 2 & 0 & 0 \\ & & & & & & & \\ & & & & & & & \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 4 \end{pmatrix} \end{array}.$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

*Fully Observed Markov Model

- $A_{(l,l'),n}$ has entry k in row $(a,b) \in \Sigma^2$ and column (x,y) if k is the number of times the word ab is consecutive in the word x,
- $A_{(l,l'),n}$ has entry k' in row $(a,c) \in \Sigma \times \Sigma'$ and column (x,y) if k' is the number of times the symbols a and c occur at the same positions in the words x and y.
- $A_{(l,l'),n}$ has column sums $(n-1)+n = 2n-1$, since each word of length n has $n-1$ consecutive length-2 words and two words of length n pair in n positions.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Fully Observed Markov Model

Given the Likelihood function (up to a constant)

$$L(\theta, \theta') = \prod_{a,b} \theta_{a,b}^{v_{ab}} \cdot \prod_{a,c} \theta'^{v'_{ac}}_{a,c}. \tag{56}$$

The maximum likelihood estimate of $L(\theta, \theta')$ is

$$\hat{\theta}_{a,b} = \frac{v_{a,b}}{\sum_{b' \in \Sigma} v_{a,b'}}, \quad a, b \in \Sigma, \tag{57}$$

and

$$\hat{\theta}'_{a,c} = \frac{v'_{a,c}}{\sum_{c' \in \Sigma'} v'_{a,c'}}, \quad a \in \Sigma, \, c \in \Sigma'. \tag{58}$$

Fully Observed Markov Model – Example (Cont'd)

Reconsider the occasionally dishonest dealer with $n = 4$:

$$\begin{aligned} v_{FF} &= 15, \quad v_{FL} = 3, \quad v_{LF} = 0, \quad v_{LL} = 6, \\ v'_{Fh} &= 18, \quad v'_{Ft} = 3, \quad v'_{Lh} = 6, \quad v'_{Lt} = 5, \end{aligned}$$

The optimal parameters are

$$\begin{aligned} \hat{\theta}_{FF} &= \frac{15}{15+3} = \frac{15}{18}, \quad \hat{\theta}_{FL} = \frac{3}{18}, \\ \hat{\theta}_{LF} &= \frac{0}{0+6} = 0, \quad \hat{\theta}_{LL} = 1, \\ \hat{\theta}'_{Fh} &= \frac{18}{18+3} = \frac{18}{21}, \quad \hat{\theta}'_{Ft} = \frac{3}{21}, \\ \hat{\theta}'_{Lh} &= \frac{6}{6+5} = \frac{6}{11}, \quad \hat{\theta}'_{Lt} = \frac{5}{11}. \end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Fully Observed Markov Model – Example (Derivative)

Reconsider the occasionally dishonest dealer with $n = 4$:

$$\begin{aligned} L(\theta, \theta') &= \frac{1}{2^N} \theta_{FF}^{v_{FF}} \theta_{FL}^{v_{FL}} \theta_{LF}^{v_{LF}} \theta_{LL}^{v_{LL}} \theta'^{v'_{Fh}}_{Fh} \theta'^{v'_{Ft}}_{Ft} \theta'^{v'_{Lh}}_{Lh} \theta'^{v'_{Lt}}_{Lt} \\ &= \frac{1}{2^N} \theta_{FF}^{v_{FF}} (1 - \theta_{FF})^{v_{FL}} \theta_{LF}^{v_{LF}} (1 - \theta_{LF})^{v_{LL}} \\ &\quad \theta'^{v'_{Fh}}_{Fh} (1 - \theta'_{Fh})^{v'_{Ft}} \theta'^{v'_{Lh}}_{Lh} (1 - \theta'_{Lh})^{v'_{Lt}} \end{aligned}$$

In view of the log-likelihood function (up to a constant),

$$\ell(\theta, \theta') = v_{FF} \log \theta_{FF} + v_{FL} \log(1 - \theta_{FF}) + \dots$$

set the derivatives to zero,

$$\frac{\partial \ell}{\partial \theta_{FF}} = \frac{v_{FF}}{\theta_{FF}} - \frac{v_{FL}}{1 - \theta_{FF}} = 0, \text{ i.e. } \hat{\theta}_{FF} = \frac{v_{FF}}{v_{FF} + v_{FL}}.$$

This gives a critical point $(\hat{\theta}, \hat{\theta}')$.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Proof.

Let $\Sigma = \{a_1, \ldots, a_l\}$ and $\Sigma' = \{c_1, \ldots, c_{l'}\}$.

- For each state $a_i \in \Sigma$, $1 \leq i \leq l$, we have $\sum_{j=1}^{l} \theta_{a_i a_j} = 1$.
- The parameters $\theta_{a_i a_j}$ with $1 \leq j \leq l$ appear in the log-likelihood function $\ell(\theta, \theta')$ as the partial sum

$$\ell_i = \sum_{j=1}^{l} v_{a_i a_j} \log(\theta_{a_i a_j}).$$

- Using $\theta_{a_i a_l} = 1 - \sum_{j=1}^{l-1} \theta_{a_i a_j}$, the partial derivative of ℓ_i with respect to $\theta_{a_i a_j}$ becomes

$$\frac{\partial \ell_i}{\partial \theta_{a_i a_j}} = \frac{v_{a_i a_j}}{\theta_{a_i a_j}} - \frac{v_{a_i a_l}}{1 - \sum_{j=1}^{l-1} \theta_{a_i a_j}}.$$

- Equating this expression to 0 gives

$$\hat{\theta}_{a_i a_j} = \frac{v_{a_i a_j}}{\sum_{k=1}^{l} v_{a_i a_k}}.$$

- Similar for the emission probabilitities $\hat{\theta}'_{a_i c_k}$ with $1 \leq k \leq l'$.
- Thus $(\hat{\theta}, \hat{\theta}') = (\hat{\theta}_{a_i a_j}, \hat{\theta}'_{a_i c_k})$ is a critial point of the likelihood function.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Proof (Cont'd).

Claim that this point maximizes the likelihood function. Indeed,

$$
\begin{aligned}
\ell(\theta,\theta') &= \sum_{a\in\Sigma}\sum_{b\in\Sigma} v_{ab}\log\theta_{ab} + \sum_{a\in\Sigma}\sum_{c\in\Sigma'} v'_{ac}\log\theta'_{ac} \\
&= \sum_{a\in\Sigma}\sum_{b\in\Sigma}\sum_{b'\in\Sigma} v_{ab'}\hat\theta_{ab}\log\theta_{ab} + \sum_{a\in\Sigma}\sum_{c\in\Sigma'}\sum_{c'\in\Sigma'} v'_{ac'}\hat\theta'_{ac}\log\theta'_{ac}, \text{ by } (57,58), \\
&= \sum_{a\in\Sigma} v_a \sum_{b\in\Sigma}\hat\theta_{ab}\log\theta_{ab} + \sum_{a\in\Sigma} v'_a \sum_{c\in\Sigma'}\hat\theta'_{ac}\log\theta'_{ac} \\
&= \sum_{a\in\Sigma} v_a \left(\sum_{b\in\Sigma}\hat\theta_{ab}\log\hat\theta_{ab} - \sum_{b\in\Sigma}\hat\theta_{ab}\log\frac{\hat\theta_{ab}}{\theta_{ab}}\right) \\
&\quad + \sum_{a\in\Sigma} v'_a \left(\sum_{c\in\Sigma'}\hat\theta'_{ac}\log\hat\theta'_{ac} - \sum_{c\in\Sigma'}\hat\theta'_{ac}\log\frac{\hat\theta'_{ac}}{\theta'_{ac}}\right) \\
&= \sum_{a\in\Sigma} -v_a\left(H(\hat\theta_a) + D(\hat\theta_a\|\theta_a)\right) + \sum_{a\in\Sigma} -v'_a\left(H(\hat\theta'_a) + D(\hat\theta'_a\|\theta'_a)\right)
\end{aligned}
$$

where $v_a = \sum_{b\in\Sigma} v_{ab}$, $v'_a = \sum_{c\in\Sigma'} v_{ac}$, $\theta_a = (\theta_{ab})_{b\in\Sigma}$, $\theta'_a = (\theta'_{ac})_{c\in\Sigma'}$, $\hat\theta_a = (\hat\theta_{ab})_{b\in\Sigma}$, and $\hat\theta'_a = (\hat\theta'_{ac})_{c\in\Sigma'}$ for each symbol $a \in \Sigma$.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Proof (Cont'd).

We have

$$\begin{aligned}
\ell(\theta,\theta') &= \sum_{a\in\Sigma} -v_a\left(H(\hat\theta_a)+D(\hat\theta_a\|\theta_a)\right)+\sum_{a\in\Sigma} -v'_a\left(H(\hat\theta'_a)+D(\hat\theta'_a\|\theta'_a)\right)\\
&\le -\sum_{a\in\Sigma}\left(v_a H(\hat\theta_a)+v'_a H(\hat\theta'_a)\right)\\
&= \sum_{a\in\Sigma}\left(v_a\sum_{b\in\Sigma}\left(\hat\theta_{ab}\log\hat\theta_{ab}\right)+v'_a\sum_{c\in\Sigma'}\left(\hat\theta'_{ac}\log\hat\theta'_{ac}\right)\right)\\
&= \sum_{a\in\Sigma}\left(\sum_{b\in\Sigma} v_{ab}\log\hat\theta_{ab}+\sum_{c\in\Sigma'} v'_{ac}\log\hat\theta'_{ac}\right),\ \text{by } (57,58),\\
&= \ell(\hat\theta,\hat\theta'),
\end{aligned}$$

where for two distributions $p=(p_1,\ldots,p_n)$ and $q=(q_1,\ldots,q_n)$,

$$H(p)=-\sum_{i=1}^{n} p_i\log(p_i)$$

is the entropy and

$$D(p\parallel q)=\sum_{i=1}^{n} p_i\log\frac{p_i}{q_i}\ge 0$$

is the Kullback-Leibler measure. □

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Probabilistic Inference of Output Marginals

- Probabilistic inference refers to the computation of posterior probabilities.
- Objective is to calculate the marginal probability of the output variables.
- Efficient solution makes use of sum-product decomposition.
- Basis of the Viterbi algorithm.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Probabilistic Inference of Output Marginals

Given integer $n \geq 1$.

- Belief network (G_n, p) of HMM with joint probability distribution

$$\begin{aligned} p_{X,Y} &= p_{X_1,\ldots,X_n,Y_1,\ldots,Y_n} \\ &= p_{X_1} p_{Y_1|X_1} p_{X_2|X_1} \cdots p_{X_n|X_{n-1}} p_{Y_n|X_n}. \end{aligned} \tag{59}$$

- Suppose the conditional probabilities

$$\theta_{x,x'} = p_{X_{i+1}|X_i}(x'|x), \quad x, x' \in \Sigma, 1 \leq i \leq n-1, \tag{60}$$

and

$$\theta'_{x,y} = p_{Y_i|X_i}(y|x), \quad x \in \Sigma, y \in \Sigma', 1 \leq i \leq n, \tag{61}$$

are known.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Probabilistic Inference of Output Marginals

- The probability of output sequence $y = (y_1, \ldots, y_n)$ is given by the *marginal distribution*

$$\begin{aligned} & p_{Y_1,\ldots,Y_n}(y_1, \ldots, y_n) \\ &= \sum_{x_1,\ldots,x_n} p_{X_1,\ldots,X_n,Y_1,\ldots,Y_n}(x_1, \ldots, x_n, y_1, \ldots, y_n) \\ &= \frac{1}{l} \sum_{x_1 \in \Sigma} \cdots \sum_{x_n \in \Sigma} \theta'_{x_1,y_1} \theta_{x_1,x_2} \theta'_{x_2,y_2} \theta_{x_2,x_3} \cdots \theta_{x_{n-1},x_n} \theta'_{x_n,y_n}. \end{aligned} \tag{62}$$

- The straightforward computation has runtime $O(nl^n)$, since the set Σ^n has l^n elements and each joint probability has $2n$ factors.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Probabilistic Inference of Output Marginals – Example

Reconsider the occasionally dishonest dealer with $n = 4$.

- Output marginals

$$p_{Y_1,Y_2,Y_3,Y_4}(y_1, y_2, y_3, y_4) = \frac{1}{2} \sum_{x_1 \in \Sigma} \cdots \sum_{x_4 \in \Sigma} \theta'_{x_1,y_1} \theta_{x_1,x_2} \theta'_{x_2,y_2} \theta_{x_2,x_3} \theta'_{x_3,y_3} \theta_{x_3,x_4} \theta'_{x_4,y_4}.$$

Sum-Product Decomposition

Decomposition of the output marginal by distributivity,

$$
\begin{aligned}
& p_Y(y) \qquad (63)\\
&= p_{Y_1,\ldots,Y_n}(y_1,\ldots,y_n)\\
&= \frac{1}{l}\sum_{x_n\in\Sigma}\theta'_{x_n,y_n}\left(\sum_{x_{n-1}\in\Sigma}\theta_{x_{n-1},x_n}\theta'_{x_{n-1},y_{n-1}}\right.\\
&\qquad \left.\left(\cdots\left(\sum_{x_2\in\Sigma}\theta_{x_2,x_3}\theta'_{x_2,y_2}\left(\sum_{x_1\in\Sigma}\theta_{x_1,x_2}\theta'_{x_1,y_1}\right)\right)\cdots\right)\right).
\end{aligned}
$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Sum-Product Decomposition – Example

Reconsider the occasionally dishonest dealer with $n = 4$.

$$
\begin{aligned}
& p_Y(y) \\
&= \quad p_{Y_1,\ldots,Y_4}(y_1,\ldots,y_4) \\
&= \frac{1}{l} \sum_{x_4 \in \Sigma} \theta'_{x_4,y_4} \left(\sum_{x_3 \in \Sigma} \theta_{x_3,x_4} \theta'_{x_3,y_3} \left(\sum_{x_2 \in \Sigma} \theta_{x_2,x_3} \theta'_{x_2,y_2} \left(\sum_{x_1 \in \Sigma} \theta_{x_1,x_2} \theta'_{x_1,y_1} \right) \right) \right)
\end{aligned}
$$

Computation (from inner to outer brackets):

$$
\begin{aligned}
M[1,x] &= \sum_{x_1 \in \Sigma} \theta_{x_1,x} \cdot \theta'_{x_1,y_1}, \quad x \in \Sigma, \\
M[2,x] &= \sum_{x_2 \in \Sigma} \theta_{x_2,x} \cdot \theta'_{x_2,y_2} \cdot M[1,x_2], \; x \in \Sigma, \\
M[3,x] &= \sum_{x_3 \in \Sigma} \theta_{x_3,x} \cdot \theta'_{x_3,y_3} \cdot M[2,x_3], \; x \in \Sigma, \\
p_Y(y) &= \frac{1}{l} \sum_{x_4 \in \Sigma} \theta'_{x_4,y_4} \cdot M[3,x_4].
\end{aligned}
$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Sum-Product Decomposition

Evaluation of $p_Y(y)$ using an $(n-1) \times l$ table M,

$$
\begin{aligned}
M[1,x] &= \sum_{x_1 \in \Sigma} \theta_{x_1,x} \cdot \theta'_{x_1,y_1}, \quad x \in \Sigma, \\
M[k,x] &= \sum_{x_k \in \Sigma} \theta_{x_k,x} \cdot \theta'_{x_k,y_k} \cdot M[k-1,x_k], \\
&\quad x \in \Sigma,\ 2 \leq k \leq n-1, \\
p_Y(y) &= \frac{1}{l} \sum_{x_n \in \Sigma} \theta'_{x_n,y_n} \cdot M[n-1,x_n].
\end{aligned}
$$

Computation has runtime $O(l^2 n)$, since the table M has $O(ln)$ entries and each entry (summation) is evaluated in $O(l)$ steps.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Viterbi Algorithm

- Calculate the most probable state sequences which produce a given output sequence.
- Named after Andrew Viterbi (1967) who proposed it as decoding algorithm for convolutional codes over noisy channels.
- Deduction by tropicalization of sum-product decomposition.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Semirings

A *semiring* is a set $R \neq \emptyset$ with two binary operations, addition $+$ and multiplication $\cdot$, such that

- $(R, +)$ is a commutative monoid with identity element 0.
- $(R, \cdot)$ is a monoid with identity element 1.
- Multiplication distributes over addition, i.e., for all $a, b, c \in R$,

  $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ and $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$.

- Multiplication with 0 annihilates R, i.e., for all $a \in R$, $a \cdot 0 = 0 = 0 \cdot a$.

A semiring is *commutative* if the multiplication is commutative, i.e., for all $a, b \in R$, $a \cdot b = b \cdot a$.
A semiring is *idempotent* if the addition is idempotent, i.e., for all $a \in R$, $a + a = a$.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Semirings

- A semiring is an algebraic structure similar to a ring, but without the requirement that each element must have an additive inverse. Thus each ring is a semiring.
- The set of natural numbers $\mathbb{N}_0$ forms a commutative semiring with the ordinary addition and multiplication.
- Likewise, the set of non-negative real numbers $\mathbb{R}_{\geq 0}$ forms a commutative semiring with the ordinary addition and multiplication, written $(\mathbb{R}_{\geq 0}, +, \cdot)$.
- The set $\mathbb{R} \cup \{\infty\}$ together with the operations

$$x \oplus y = \min\{x, y\}, \tag{64}$$

$$x \odot y = x + y \tag{65}$$

for all $x, y \in \mathbb{R} \cup \{\infty\}$, with $x + \infty = \infty = \infty + x$, forms an idempotent commutative semiring with additive identity ∞ and multiplicative identity 0.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Semirings

- The semiring $(\mathbb{R} \cup \{\infty\}, \min, +)$ is called min-plus algebra or *tropical algebra*.
- Additive and multiplicative inverses may not exist; e.g., the equations $3 \oplus x = 10$ and $\infty \odot x = 1$ have no solutions $x \in \mathbb{R} \cup \{\infty\}$.
- The mapping

$$\phi : \mathbb{R}_{\geq 0} \to \mathbb{R} \cup \{\infty\} : x \mapsto -\log x \tag{66}$$

is bijective and strictly monotonic decreasing with $\phi(0) = \infty$, $\phi(1) = 0$, and

$$\phi(x \cdot y) = \phi(x) \odot \phi(y), \quad x, y \in \mathbb{R}_{\geq 0}. \tag{67}$$

This mapping is the *tropicalization* of the ordinary semiring $(\mathbb{R}_{\geq 0}, +, \cdot)$.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

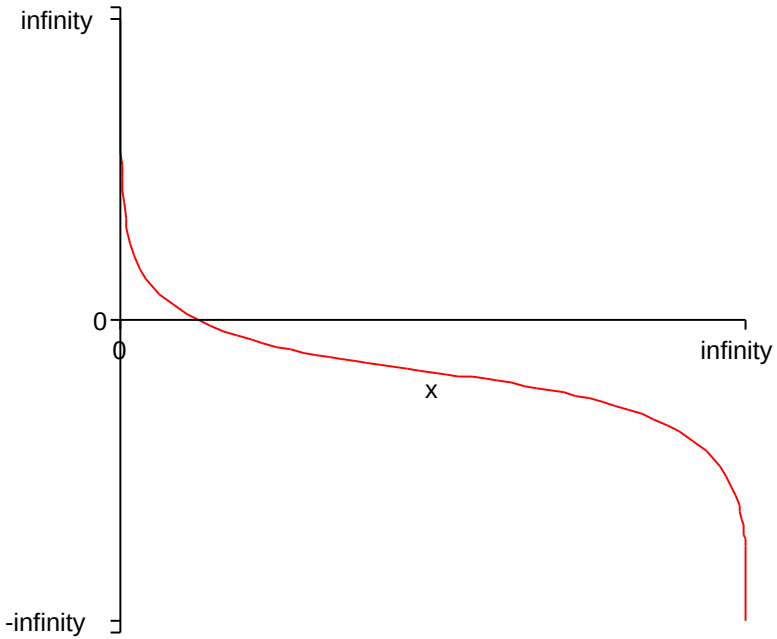
*BW Algorithm

CpG Islands

Using R

Semirings

Tropicalization $x \mapsto -\log x$.



Tropicalization maps large probabilities to small weights and vice versa; maximization of probabilities corresponds to the minimization of weights.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Tropicalization

$$\left.\begin{array}{l}\text{semiring } \mathbb{R}_{\geq 0}\\ \text{addition } +\\ \text{multiplication } \cdot\\ \text{data } x\\ \text{large probabilities}\end{array}\right\} \rightarrow \left\{\begin{array}{l}\text{semiring } \mathbb{R}\cup\{\infty\}\\ \text{addition } \oplus \text{ (min)}\\ \text{multiplication } \odot \text{ (add)}\\ \text{data } -\log x\\ \text{small weights}\end{array}\right. \quad (68)$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Viterbi Algorithm

Given integer $n \geq 1$.

- Given belief network (G_n, p) of HMM with joint probability distribution

$$\begin{aligned} p_{X,Y} &= p_{X_1,\ldots,X_n,Y_1,\ldots,Y_n} \\ &= p_{X_1} p_{Y_1|X_1} p_{X_2|X_1} \cdots p_{X_n|X_{n-1}} p_{Y_n|X_n}. \end{aligned} \tag{69}$$

- Suppose the parameters for state transition

$$\theta_{x,x'} = p_{X_{i+1}|X_i}(x'|x), \quad x, x' \in \Sigma, 1 \leq i \leq n-1, \tag{70}$$

and emission

$$\theta'_{x,y} = p_{Y_i|X_i}(y|x), \quad x \in \Sigma, y \in \Sigma', 1 \leq i \leq n, \tag{71}$$

are known.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Viterbi Algorithm

- The probability of output sequence $y = (y_1, \ldots, y_n)$ is given by the marginal distribution

$$\begin{aligned} & p_Y(y) \\ = \; & p_{Y_1,\ldots,Y_n}(y_1, \ldots, y_n) \\ = \; & \sum_{x_1,\ldots,x_n} p_{X_1,\ldots,X_n,Y_1,\ldots,Y_n}(x_1, \ldots, x_n, y_1, \ldots, y_n) \\ = \; & \frac{1}{l} \sum_{x_1 \in \Sigma} \cdots \sum_{x_n \in \Sigma} \theta'_{x_1,y_1} \theta_{x_1,x_2} \theta'_{x_2,y_2} \theta_{x_2,x_3} \cdots \theta_{x_{n-1},x_n} \theta'_{x_n,y_n}. \end{aligned} \tag{72}$$

- Given output sequence $y \in \Sigma'^n$. The objective is to find one (or all) state sequences $x \in \Sigma^n$ with maximum likelihood

$$p_{X|Y}(x|y) = \frac{p_{X,Y}(x,y)}{p_Y(y)}. \tag{73}$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Viterbi Algorithm

- The observed sequence $y \in \Sigma'^n$ is fixed, so the likelihood $p_{X|Y}(x|y)$ is directly proportional to the joint probability $p_{X,Y}(x,y)$ if $p_Y(y) > 0$.
- Suppose $p_Y(y) > 0$. Then the aim is to find a state sequence $\bar{x} \in \Sigma^n$ with the property

$$\begin{aligned} \bar{x} &= \operatorname{argmax}_{x \in \Sigma^n} \{p_{X|Y}(x|y)\} \\ &= \operatorname{argmax}_{x \in \Sigma^n} \{p_{X,Y}(x,y)\}. \end{aligned} \tag{74}$$

  Each optimal state sequence $\bar{x}$ is called an *explanation* of the observed sequence y. The explanations can be found by tropicalization of sum-product decomposition.
- *Tropicalization* means that replacing ordinary sums by tropical sums and ordinary multiplication by tropical multiplication (68).

Soft Computing
K.-H. Zimmermann
Contents
Belief Network
Learning FOM
Probabilistic Inference
Viterbi Algorithm
EM Algorithm
*BW Algorithm
CpG Islands
Using R

Viterbi Algorithm

- Put

$$w(y) = -\log p_Y(y) \quad \text{and} \quad w(x,y) = -\log p_{X,Y}(x,y). \tag{75}$$

- Tropicalization

$$w(y) = \bigoplus_{x \in \Sigma^n} w(x,y) = \min_{x \in \Sigma^n} w(x,y). \tag{76}$$

The explanations $\bar{x}$ are obtained by evaluation in the tropical algebra,

$$\bar{x} \quad = \quad \text{argmin}_{x \in \Sigma^n}\{w(x,y)\}. \tag{77}$$

The value $w(y)$ can be efficiently computed by tropicalizing the sum-product decomposition of the marginal probability $p_Y(y)$.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Viterbi Algorithm

- Replace the transition probabilities by weights,

$$u_{ab} = -\log \theta_{ab} \quad \text{and} \quad v_{ac} = -\log \theta'_{ac} \tag{78}$$

  for all $a, b \in \Sigma$, $c \in \Sigma'$.

- In the sum-product decomposition (63) replace sums by tropical sums and products by tropical products,

$$\begin{aligned} w(y) \quad = \quad & \bigoplus_{x_n} v_{x_n,y_n} \odot \left(\bigoplus_{x_{n-1}} u_{x_{n-1},x_n} \odot v_{x_{n-1},y_{n-1}} \right. \\ & \left. \odot \left(\cdots \left(\bigoplus_{x_1} u_{x_1,x_2} \odot v_{x_1,y_1} \right) \cdots \right) \right). \end{aligned} \tag{79}$$

- For each output sequence $y \in \Sigma'^n$, the tropicalization $w(y)$ of the marginal probability $p_Y(y)$ provides the explanations of y.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Sum-Product Decomposition – Example

Reconsider the occasionally dishonest dealer with $n = 4$:

$$w(y) = \bigoplus_{x_4 \in \Sigma} v_{x_4,y_4} \left(\bigoplus_{x_3 \in \Sigma} u_{x_3,x_4} v_{x_3,y_3} \left(\bigoplus_{x_2 \in \Sigma} u_{x_2,x_3} v_{x_2,y_2} \left(\bigoplus_{x_1 \in \Sigma} u_{x_1,x_2} v_{x_1,y_1} \right) \right) \right)$$

Computation (from inner to outer brackets):

$$\begin{aligned}
M[1,x] &= \bigoplus_{x_1 \in \Sigma} u_{x_1,x} \odot v_{x_1,y_1}, \quad x \in \Sigma, \\
M[2,x] &= \bigoplus_{x_2 \in \Sigma} u_{x_2,x} \odot v_{x_2,y_2} \odot M[1,x_2], \; x \in \Sigma, \\
M[3,x] &= \bigoplus_{x_3 \in \Sigma} u_{x_3,x} \odot v_{x_3,y_3} \odot M[2,x_3], \; x \in \Sigma, \\
w(y) &= \bigoplus_{x_4 \in \Sigma} v_{x_4,y_4} \odot M[3,x_4].
\end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Sum-Product Decomposition – Example (Cont'd)

Reformulation of computation for $n = 4$:

$$
\begin{aligned}
M[1,x] &= \min_{x_1 \in \Sigma} u_{x_1,x} + v_{x_1,y_1}, \quad x \in \Sigma, \\
M[2,x] &= \min_{x_2 \in \Sigma} u_{x_2,x} + v_{x_2,y_2} + M[1,x_2], \; x \in \Sigma, \\
M[3,x] &= \min_{x_3 \in \Sigma} u_{x_3,x} + v_{x_3,y_3} + M[2,x_3], \; x \in \Sigma, \\
w(y) &= \min_{x_4 \in \Sigma} v_{x_4,y_4} + M[3,x_4].
\end{aligned}
$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Viterbi Algorithm

The tropicalized term $w(y)$ can be computed by evaluating iteratively the tropicalized sum-product decomposition using an $(n-1) \times l$ table M:

$$
\begin{aligned}
M[1,x] &:= \bigoplus_{x_1 \in \Sigma} u_{x_1,x} \odot v_{x_1,y_1}, \quad x \in \Sigma, \\
M[k,x] &:= \bigoplus_{x_k \in \Sigma} u_{x_k,x} \odot v_{x_k,y_k} \odot M[k-1,x_k], \\
&\qquad x \in \Sigma,\ 2 \leq k \leq n-1, \\
w(y) &:= \bigoplus_{x_n \in \Sigma} v_{x_n,y_n} \odot M[n-1,x_n].
\end{aligned}
$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Viterbi Algorithm

Rewritten algorithm using the ordinary operations:

$$\begin{aligned} M[1,x] &:= \min_{x_1 \in \Sigma} u_{x_1,x} + v_{x_1,y_1}, \quad x \in \Sigma, \\ M[k,x] &:= \min_{x_k \in \Sigma} u_{x_k,x} + v_{x_k,y_k} + M[k-1,x_k], \\ & \quad x \in \Sigma,\ 2 \leq k \leq n-1, \\ w(y) &:= \min_{x_n \in \Sigma} v_{x_n,y_n} + M[n-1,x_n]. \end{aligned}$$

This is the *Viterbi algorithm* and the computed explanations are the *Viterbi sequences*.

Viterbi Algorithm

- The Viterbi algorithm consists of a *forward algorithm* evaluating the data (table M) and a *backward algorithm* providing the optimal decisions (explanations).
- In the forward algorithm, for each step k, $1 \leq k \leq n$, all states $x \in \Sigma$ which attain the minimum in the minimization step can be recorded.
- This information can be used by the backward algorithm to trace back all optimal decisions (states) from the last (n) to the first position (1) and so provide *all* explanations.
- Time complexity is $O(l^2 n)$ as described by the sum-product decomposition of the output marginals.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Viterbi Algorithm

- Viterbi algorithm follows the paradigm of *dynamic programming*.
- Dynamic programming was developed by Richard Bellman in the 1950s.
- A dynamic programming algorithm consists of forward algorithm (data evaluation) and backward algorithm (optimal decisions).
- Dynamic programming is applicable if the subproblems have the same structure and the subproblems can be recursively nested.
- The Viterbi algorithm is not only optimal for the whole sequence $y \in \Sigma'^n$ but also for each prefix of y.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Viterbi Algorithm

Require: Sequence $y \in \Sigma'^n$, scores (u_{ab}) and (v_{ac})
Ensure: Tropicalized term $w(y)$
  $M \leftarrow \text{matrix}[0..n, 1..l]$
  for $x \leftarrow 1$ to l **do**
    $M[0, x] \leftarrow 0$
  end for
  for $k \leftarrow 1$ to $n-1$ **do**
    for $x \leftarrow 1$ to l **do**
      $M[k, x] \leftarrow \infty$
      for $x' \leftarrow 1$ to l **do**
        $M[k, x] \leftarrow \min\{M[k, x], u_{x',x} + v_{x',y_k} + M[k-1, x']\}$ {Record all x which attain the minimum}
      end for
    end for
  end for
  for $x \leftarrow 1$ to l **do**
    $M[n, x] \leftarrow v_{x,y_n} + M[n-1, x]$
  end for
  $w \leftarrow \infty$
  for $x \leftarrow 1$ to l **do**
    $w \leftarrow \min\{w, M[n, x]\}$ {Record all x which attain the minimum}
  end for
  return $w = w(y)$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

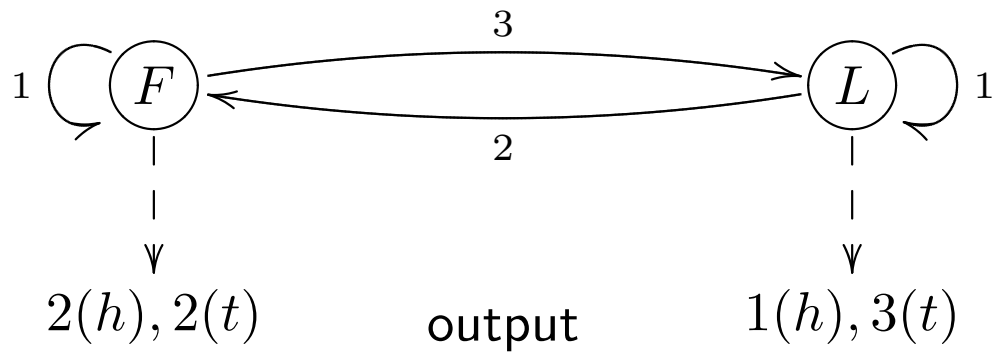
*BW Algorithm

CpG Islands

Using R

Viterbi Algorithm – Example

Reconsider the occasionally dishonest casino with the weights



Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

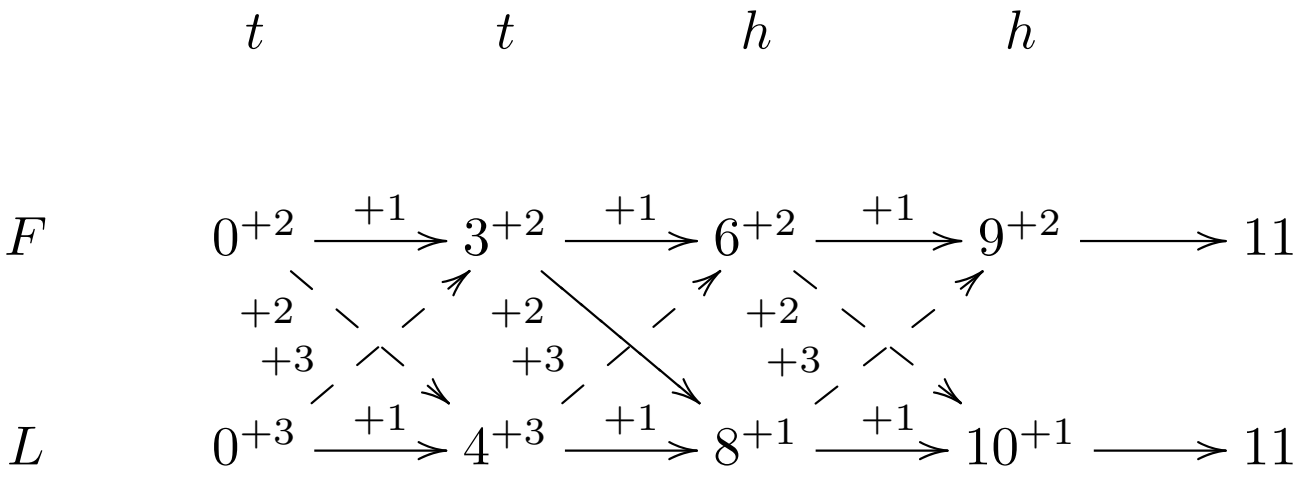
*BW Algorithm

CpG Islands

Using R

Viterbi Algorithm – Example (Cont'd)

Trellis for output sequence $y = tthh$:



- Solid arrows show where the minima are attained.
- The trace back given by the paths of solid arrows provides the explanations: $FFFF$, $FFLL$, and $LLLL$.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

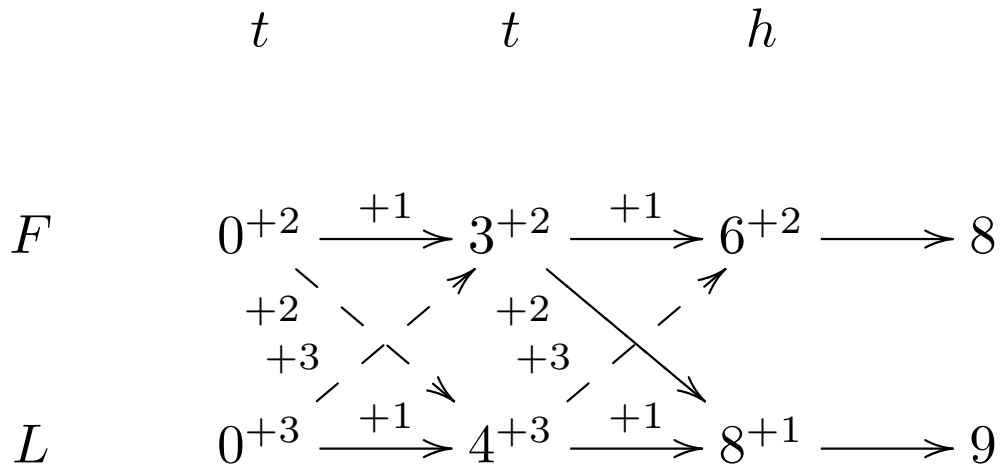
*BW Algorithm

CpG Islands

Using R

Viterbi Algorithm – Example (Cont'd)

Subproblem: trellis for output subsequence $y = tth$:



- Solid arrows show where the minima are attained.
- The trace back given by the paths of solid arrows provides the explanation: FFF.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Learning in Hidden Markov Model

- Objective is to learn or estimate the conditional probabilities.
- Sample data consist only of output sequences.
- Solution by EM algorithm and more efficiently by BW algorithm.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

EM Algorithm for HMM

Require: Hidden Markov model, joint probability function $p_{X,Y|\Theta\times\Theta'}$, parameter space $\Theta \times \Theta' \subseteq \mathbb{R}_{>0}^{l(l-1)} \times \mathbb{R}_{>0}^{l(l'-1)}$, integer $n \geq 1$, observed (output) data $u = (u_y) \in \mathbb{N}^{l'^n}$

Ensure: Maximum likelihood estimate $(\theta^*, \theta'^*) \in \Theta \times \Theta'$

[Init] Threshold $\epsilon > 0$ and parameters $(\theta, \theta') \in \Theta \times \Theta'$

[E-Step] Define matrix $U = (u_{x,y}) \in \mathbb{R}^{l^n \times l'^n}$ with

$$u_{x,y} = \frac{u_y \cdot p_{X,Y|\Theta\times\Theta'}(x, y|\theta, \theta')}{p_{Y|\Theta\times\Theta'}(y|\theta, \theta')}, \quad x \in \Sigma^n,\ y \in \Sigma'^n$$

[M-Step] Compute solution $(\hat{\theta}, \hat{\theta}') \in \Theta \times \Theta'$ of the likelihood function $\ell_{X,Y}$ using data set $U = (u_{x,y})$, linear transformation (55) and equations (57,58)

[Compare] If $\ell_Y(\hat{\theta}, \hat{\theta}') - \ell_Y(\theta, \theta') > \epsilon$, set $\theta \leftarrow \hat{\theta}$ and $\theta' \leftarrow \hat{\theta}'$ and resume with E-step, see (91)

[Output] $\theta^* \leftarrow \hat{\theta}$, $\theta'^* \leftarrow \hat{\theta}$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

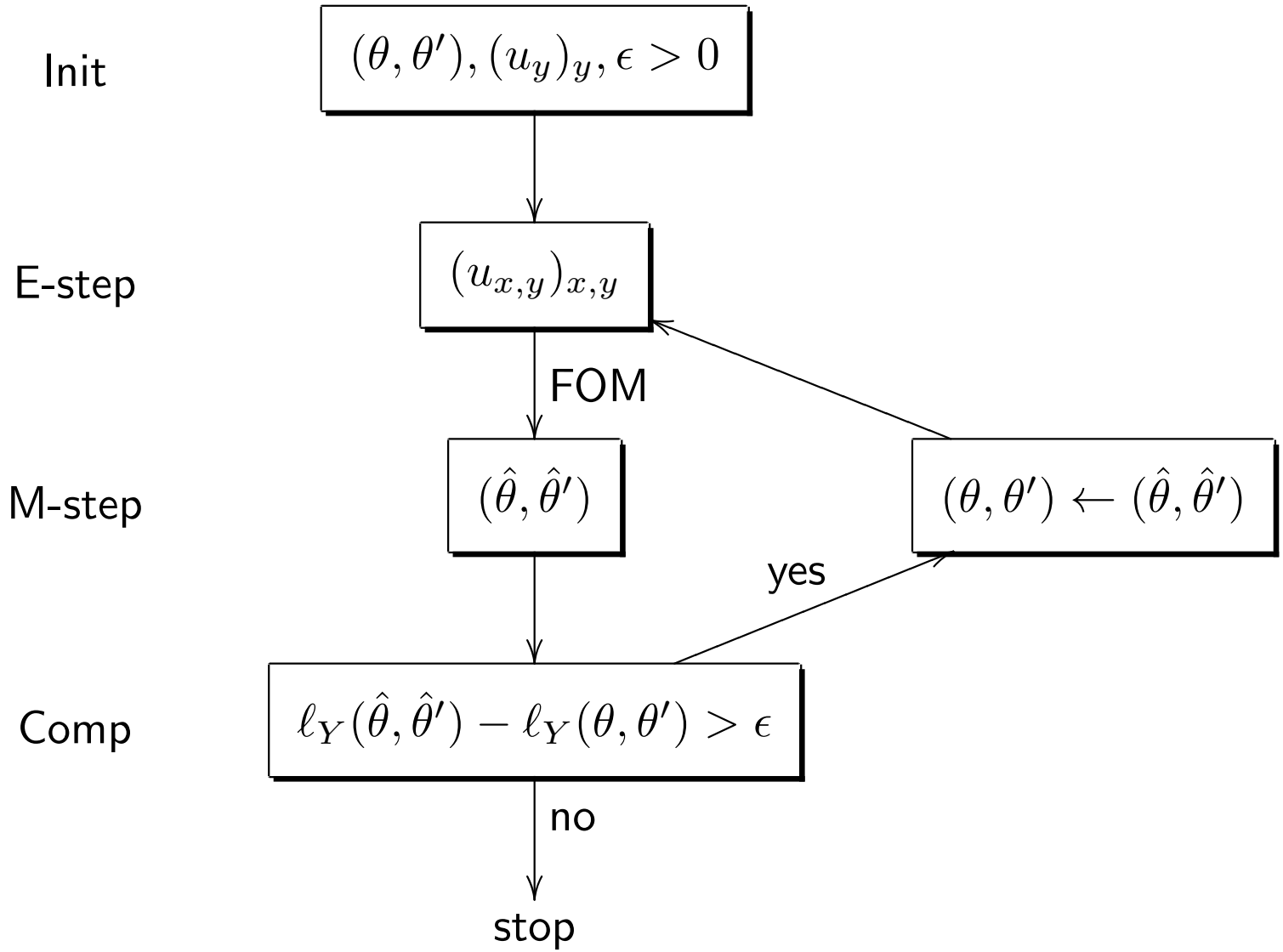
EM Algorithm

*BW Algorithm

CpG Islands

Using R

EM Algorithm for HMM



Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

EM Algorithm for HMM – Example

In view of the occasionally dishonest casino with $n = 4$, given $N = 8$ samples

$$\begin{array}{llll} d_1 = hhhh, & d_2 = hhtt, & d_3 = hhhh, & d_4 = hhtt, \\ d_5 = hhhh, & d_6 = hhtt, & d_7 = tttt, & d_8 = htth. \end{array}$$

Counts

$$u_{hhhh} = 3, \quad u_{hhtt} = 3 \quad u_{htth} = 1, \quad u_{tttt} = 1.$$

Likelihood function

$$\begin{aligned} L_Y(\theta, \theta') &= \prod_{i=1}^{N} p_{Y|\Theta\times\Theta'}(d_i|\theta, \theta') \\ &= p_Y(hhhh)^3 p_Y(hhtt)^3 p_Y(htth) p_Y(tttt). \end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

EM Algorithm for HMM – Example (Cont'd)

Log-likelihood function

$$\ell_Y(\theta, \theta') = 3 \log p_Y(hhhh) + 3 \log p_Y(hhtt) + \log p_Y(htth) + \log p_Y(tttt).$$

Given the parameters (θ, θ'), the estimated values (E-step)

$$u_{x,y} = \frac{u_y \cdot p_{X,Y}(x, y)}{p_Y(y)}, \quad x \in \Sigma^n, \ y \in \Sigma'^n,$$

are computed by calculating directly the joint probabilities

$$p_{X,Y}(x, y) = \frac{1}{2} \theta'_{x_1,y_1} \theta_{x_1,x_2} \theta'_{x_2,y_2} \theta_{x_2,x_3} \theta'_{x_3,y_3} \theta_{x_3,x_4} \theta'_{x_4,y_4}$$

and the marginal probabilities by using the sum-product decomposition (63),

$$p_Y(y) = \sum_x p_{X,Y}(x, y).$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

EM Algorithm for HMM

- In the E-step, the marginal probabilities $p_Y(y|\theta, \theta')$ can be computed efficiently by the sum-product decomposition.
- In the M-step, the maximal estimates $\hat{\theta}$ and $\hat{\theta}'$ can be calculated as in the fully observed model (57, 58).
- The EM algorithm switches back and forth between the fully observed and the hidden model!

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

EM Algorithm (General)

- Given random vectors Y and Z with finite state sets $\mathcal{Y} = \{y^{(1)}, \ldots, y^{(m)}\}$ and $\mathcal{Z} = \{z^{(1)}, \ldots, z^{(n)}\}$, resp.
- Suppose the joint probability distribution depends on the parameters θ of a parameter set Θ. Write

$$f_{ij}(\theta) = p_{Y,Z|\Theta}(y^{(i)}, z^{(j)}), \quad 1 \leq i \leq m, 1 \leq j \leq n. \tag{80}$$

- Marginal distribution with respect to Y,

$$f_i(\theta) = p_{Y|\Theta}(y^{(i)}) = \sum_{j=1}^{n} f_{ij}(\theta), \quad 1 \leq i \leq m. \tag{81}$$

This quantity is often intractable such as in the case when the number of values of Z grows exponentially with the sequence length making the exact calculation of the marginals extremely difficult.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

EM Algorithm (Hidden Case)

- Take collection of N independent samples from state space $\mathcal{Y}$,

$$D = (y^{(i_1)}, \ldots, y^{(i_N)}), \tag{82}$$

where $1 \leq i_1, \ldots, i_N \leq m$.

- This collection gives rise to the data vector

$$u = (u_1, \ldots, u_m) \in \mathbb{N}^m, \tag{83}$$

where u_i is the number of occurrences of the state $y^{(i)}$ in the sequence D, $1 \leq i \leq m$.

- We have

$$\sum_{i=1}^{m} u_i = N. \tag{84}$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

EM Algorithm (Hidden Case)

- Objective is to maximize the likelihood function with respect to the parameters $\theta \in \Theta$ for the observed data,

$$L(\theta) = \prod_{i=1}^{m} f_i(\theta)^{u_i}. \tag{85}$$

- Log-likelihood function:

$$\ell_Y(\theta) = u_1 \cdot \log f_1(\theta) + \cdots + u_m \cdot \log f_m(\theta). \tag{86}$$

The data vector u forms the sufficient statistic of the model.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

EM Algorithm (Fully Observed Case)

- Consider collection of N independent sample pairs from state space $\mathcal{Y} \times \mathcal{Z}$,

$$D = ((y^{(i_1)}, z^{(j_1)}), \ldots, (y^{(i_N)}, z^{(j_N)})) \tag{87}$$

where $1 \leq i_1, \ldots, i_N \leq m$ and $1 \leq j_1, \ldots, j_N \leq n$.

- This collection gives rise to the data vector

$$U = (u_{ij}) \in \mathbb{N}^{m \times n}, \tag{88}$$

where u_{ij} is the number of occurrences of the sequence pair $(y^{(i)}, z^{(j)})$ in the sequence D with $u_i = \sum_{j=1}^{n} u_{ij}$, $1 \leq i \leq m$.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

EM Algorithm (Fully Observed Case)

- The objective is to maximize the likelihood function w.r.t. parameters $\theta \in \Theta$ for the observed sequence pairs,

$$L_{Y,Z}(\theta) = \prod_{i,j} f_{ij}(\theta)^{u_{ij}}. \tag{89}$$

- Log-likelihood function

$$\ell_{Y,Z}(\theta) = u_{11} \cdot \log f_{11}(\theta) + \cdots + u_{mn} \cdot \log f_{mn}(\theta). \tag{90}$$

- Assumption: Optimal parameters attained by maximization of $\ell_{Y,Z}(\theta)$ can be efficiently computed such as for HMM.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

EM Algorithm (General)

Require: Joint probability function $p_{Y,Z|\Theta}$, parameter set $\Theta \subseteq \mathbb{R}^d$, observed data $u = (u_i) \in \mathbb{N}^m$

Ensure: Maximum likelihood estimate $\theta^* \in \Theta$ of the log-likelihood function $\ell_Y(\theta)$

[Init] Threshold $\epsilon > 0$ and parameter $\theta \in \Theta$

[E-Step] Define matrix $U = (u_{ij}) \in \mathbb{R}_{>0}^{m \times n}$ with

$$u_{ij} = \frac{u_i \cdot f_{ij}(\theta)}{f_i(\theta)}$$

[M-Step] Compute solution $\hat{\theta} \in \Theta$ of the likelihood function $\ell_{Y,Z}$ using data set $U = (u_{ij})$

[Compare] If $\ell_Y(\hat{\theta}) - \ell_Y(\theta) > \epsilon$, set $\theta \leftarrow \hat{\theta}$ and resume with E-step, see (91)

[Output] $\theta^* \leftarrow \hat{\theta}$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

EM Algorithm (General)

During each iteration of the EM algorithm, the value of the log-likelihood function increases:

$$\ell_Y(\hat{\theta}) \geq \ell_Y(\theta). \tag{91}$$

If $\ell_Y(\hat{\theta}) = \ell_Y(\theta)$, then $\hat{\theta}$ is a critical point (local maximum) of the log-likelihood function.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

*Proof

We have

$$\begin{aligned}
\ell_Y(\hat{\theta}) - \ell_Y(\theta) &= \sum_{i=1}^{m} u_i \cdot [\log f_i(\hat{\theta}) - \log f_i(\theta)] \\
&= \sum_{i=1}^{m} \sum_{j=1}^{n} u_{ij} \cdot [\log f_{ij}(\hat{\theta}) - \log f_{ij}(\theta)] \\
&\quad + \sum_{i=1}^{m} u_i \cdot \left(\log \left(\frac{f_i(\hat{\theta})}{f_i(\theta)} \right) - \sum_{j=1}^{n} \frac{u_{ij}}{u_i} \cdot \log \left(\frac{f_{ij}(\hat{\theta})}{f_{ij}(\theta)} \right) \right).
\end{aligned}$$

The first term in this expression equals $\ell_{Y,Z}(\hat{\theta}) - \ell_{Y,Z}(\theta)$ and is non-negative due to the M-step.

Proof (Cont'd)

Claim that the second term is also non-negative. Indeed, the parenthesized expression can be written by the E-step as follows,

$$\log\left(\frac{f_i(\hat{\theta})}{f_i(\theta)}\right) - \sum_{j=1}^{n} \frac{u_{ij}}{u_i} \cdot \log\left(\frac{f_{ij}(\hat{\theta})}{f_{ij}(\theta)}\right) = \log\left(\frac{f_i(\hat{\theta})}{f_i(\theta)}\right) + \sum_{j=1}^{n} \frac{f_{ij}(\theta)}{f_i(\theta)} \cdot \log\left(\frac{f_{ij}(\theta)}{f_{ij}(\hat{\theta})}\right).$$

Since $f_i(\theta) = \sum_j f_{ij}(\theta)$ for each $1 \leq i \leq m$, this expression can be rewritten as

$$\sum_{j=1}^{n} \frac{f_{ij}(\theta)}{f_i(\theta)} \cdot \log\left(\frac{f_i(\hat{\theta})}{f_i(\theta)}\right) + \sum_{j=1}^{n} \frac{f_{ij}(\theta)}{f_i(\theta)} \cdot \log\left(\frac{f_{ij}(\theta)}{f_{ij}(\hat{\theta})}\right)$$

and thus amounts to

$$\sum_{j=1}^{n} \frac{f_{ij}(\theta)}{f_i(\theta)} \cdot \log\left(\frac{f_i(\hat{\theta})}{f_{ij}(\hat{\theta})} \cdot \frac{f_{ij}(\theta)}{f_i(\theta)}\right). \tag{92}$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Proof (Cont'd)

Take the non-negative quantities

$$\pi_j = \frac{f_{ij}(\theta)}{f_i(\theta)} \quad \text{and} \quad \sigma_j = \frac{f_{ij}(\hat{\theta})}{f_i(\hat{\theta})}, \qquad 1 \leq j \leq n.$$

Then we have

$$\pi_1 + \ldots + \pi_n = 1 = \sigma_1 + \ldots + \sigma_n.$$

Thus the vectors π and σ are probability distributions. The expression (92) is the Kullback-Leibler divergence from π to σ,

$$D(\pi \| \sigma) = \sum_{j=1}^{n} \pi_j \cdot \log\left(\frac{\pi_j}{\sigma_j}\right).$$

The Kullback-Leibler measure is always nonnegative,

$$D(\pi \| \sigma) \geq 0.$$

It follows that the second expression is also non-negative and the claim follows. Hence, we have $\ell_Y(\hat{\theta}) \geq \ell_Y(\theta)$.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Proof (Cont'd)

Suppose $\ell_Y(\hat{\theta}) = \ell_Y(\theta)$ for some $\theta, \hat{\theta} \in \Theta$. Then the expression in (92) must be zero. But the Kullback-Leibler divergence satisfies $D(\pi \| \sigma) = 0$ if and only if $\pi = \sigma$. Thus we have

$$\frac{f_{ij}(\theta)}{f_i(\theta)} = \frac{f_{ij}(\hat{\theta})}{f_i(\hat{\theta})}, \qquad 1 \leq i \leq m,\ 1 \leq j \leq n. \tag{93}$$

Therefore, we have

$$\begin{aligned}
0 &= \frac{\partial \ell_{Y,Z}(\hat{\theta})}{\partial \theta_k} \\
&= \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{u_{ij}}{f_{ij}(\hat{\theta})} \cdot \frac{\partial f_{ij}(\hat{\theta})}{\partial \theta_k} \\
&= \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{u_i}{f_i(\hat{\theta})} \cdot \frac{\partial f_{ij}}{\partial \theta_k}(\hat{\theta}) \\
&= \sum_{i=1}^{m} \frac{u_i}{f_i(\hat{\theta})} \cdot \left(\frac{\partial}{\partial \theta_k} \sum_{j=1}^{n} f_{ij} \right)(\hat{\theta}) \\
&= \sum_{i=1}^{m} \frac{u_i}{f_i(\hat{\theta})} \cdot \left(\frac{\partial}{\partial \theta_k} f_i \right)(\hat{\theta}) = \frac{\partial \ell_Y(\hat{\theta})}{\partial \theta_k}, \quad 1 \leq k \leq d,
\end{aligned}$$

where in the third equation we used both the E-step and (93). □

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Baum-Welch Algorithm

- Structure of HMM allows a more efficient implementation of the EM algorithm known as the Baum-Welch algorithm.
- Dynamic programming algorithm developed in the late 1960s.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Baum-Welch Algorithm

- Given data vector $u = (u_y) \in \mathbb{N}^{l'^n}$, where u_y is the number of times the output sequence $y \in \Sigma'^n$ is observed.
- The full data vector $U = (u_{x,y}) \in \mathbb{N}^{l^n \times l'^n}$ is not available, where $u_{x,y}$ denotes the number of times the pair $(x, y) \in \Sigma^n \times \Sigma'^n$ is observed.
- The EM algorithm estimates in the E-step the counts of the full data vector by the quantity

$$u_{x,y} = \frac{u_y \cdot p_{X,Y|\Theta\times\Theta'}(x, y|\theta, \theta')}{p_{Y|\Theta\times\Theta'}(y|\theta, \theta')}, \quad x \in \Sigma^n,\ y \in \Sigma'^n. \quad (94)$$

- These counts are used in (57, 58) to provide the sufficient statistic (v, v') of the model, which is used in the M-step to derive updated parameter values θ, θ'.
- In the BW algorithm, the sufficient statistic (v, v') can be directly calculated by dynamic programming.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Baum-Welch Algorithm

Let $y \in \Sigma'^n$.

- The *forward probability*

$$f_{y,s}(i) = p_{Y_1,\ldots,Y_i,X_i}(y_1,\ldots,y_i,s),\ s \in \Sigma,\ 1 \leq i \leq n, \quad (95)$$

is the joint probability that the prefix $y_1 \ldots y_i$ of the observed sequence y having length i ends in state s.

- The *backward probability*

$$b_{y,s}(i) = p_{Y_{i+1},\ldots,Y_n|X_i}(y_{i+1},\ldots,y_n \mid s),\ s \in \Sigma,\ 1 \leq i \leq n, \quad (96)$$

is the conditional probability that the suffix $y_{i+1} \ldots y_n$ of the observed sequence y having length $n - i$ starts in state s.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Baum-Welch Algorithm

- The probability $p_Y(y|\theta,\theta')$ of the emitted sequence y can be calculated by the forward probabilities,

$$p_Y(y|\theta,\theta') = \sum_{r\in\Sigma} f_{y,r}(n). \tag{97}$$

- For each emitted sequence $y \in \Sigma'^n$, consider the $l \times n$ matrices

$$F_y = (f_{y,r}(i)) \quad \text{and} \quad B_y = (b_{y,r}(i)) \tag{98}$$

with $r \in \Sigma$ and $1 \leq i \leq n$, of the forward and backward probabilities, resp.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Baum-Welch Algorithm

- The entries of the matrices F_y and B_y can be efficiently computed in an iterative manner,

$$f_{y,r}(1) = \frac{1}{l}\theta'_{r,y_1}, \quad r \in \Sigma, \tag{99}$$

$$f_{y,r}(i) = \theta'_{r,y_i} \sum_{s\in\Sigma} f_{y,s}(i-1)\cdot\theta_{sr}, \tag{100}$$

$$r \in \Sigma,\ 2 \le i \le n,$$

and

$$b_{y,r}(n) = 1, \quad r \in \Sigma, \tag{101}$$

$$b_{y,r}(i) = \sum_{s\in\Sigma} \theta_{rs}\cdot\theta'_{s,y_{i+1}}\cdot b_{y,s}(i+1), \tag{102}$$

$$r \in \Sigma,\ 1 \le i \le n-1.$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Baum-Welch Algorithm – Example

Reconsider the dishonest casino dealer with output sequence $y = (y_1, \ldots, y_4)$, $n = 4$.

$$
\begin{aligned}
f_{y,r}(1) &= \frac{1}{2}\theta'_{r,y_1}, \quad r \in \Sigma, \\
f_{y,r}(2) &= \theta'_{r,y_2} \sum_{s\in\Sigma} f_{y,s}(1)\theta_{sr} \\
&= \frac{1}{2}\theta'_{r,y_2}\left(\theta'_{F,y_1}\theta_{F,r} + \theta'_{L,y_1}\theta_{L,r}\right), \quad r \in \Sigma, \\
f_{y,r}(3) &= \theta'_{r,y_3} \sum_{s\in\Sigma} f_{y,s}(2)\theta_{sr} \\
&= \frac{1}{2}\theta'_{r,y_3}\left(f_{y,F}(2)\theta_{F,r} + f_{y,L}(2)\theta_{L,r}\right), \quad r \in \Sigma,
\end{aligned}
$$

Baum-Welch Algorithm – Example (Cont'd)

$$\begin{aligned} f_{y,r}(4) &= \theta'_{r,y_4} \sum_{s\in\Sigma} f_{y,s}(3)\theta_{sr} \\ &= \frac{1}{2}\theta'_{r,y_4}\left(f_{y,F}(3)\theta_{F,r} + f_{y,L}(3)\theta_{L,r}\right), \quad r \in \Sigma, \\ &= \sum_{x_1,x_2,x_3\in\Sigma} p_{X,Y}(x_1,x_2,x_3,r,y_1,y_2,y_3,y_4). \end{aligned}$$

and

$$\begin{aligned} p_Y(y_1,\ldots,y_4) &= \sum_{r\in\Sigma} f_{y,r}(4) \\ &= \sum_{x_1,x_2,x_3,x_4\in\Sigma} p_{X,Y}(x_1,x_2,x_3,x_4,y_1,y_2,y_3,y_4). \end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Baum-Welch Algorithm

In view of the sufficient statistic (v, v'), we have for all $r, s \in \Sigma$ and $t \in \Sigma'$,

$$v_{rs} = \sum_{y \in \Sigma'^n} \frac{u_y}{p(y|\theta, \theta')} \sum_{i=1}^{n-1} f_{y,r}(i) \cdot \theta_{r,s} \cdot \theta'_{s,y_{i+1}} \cdot b_{y,s}(i+1) \quad (103)$$

$$v'_{rt} = \sum_{y \in \Sigma'^n} \frac{u_y}{p(y|\theta, \theta')} \sum_{i=1}^{n-1} f_{y,r}(i) \cdot b_{y,r}(i+1) \cdot I_{(y_i=t)}, \quad (104)$$

where I_A denotes the indicator function of statement A; i.e., $I_A = 1$ if A is true and $I_A = 0$ otherwise.

Baum-Welch Algorithm – Proof

For each state sequence $x \in \Sigma^n$,

$$v_{rs} = \sum_{y \in \Sigma'^n} \sum_{x \in \Sigma^n} \sum_{i=1}^{n-1} I_{(x_i x_{i+1} = rs)} \cdot u_{x,y}.$$

Thus in view of the definition of $u_{x,y}$ in the E-step,

$$v_{rs} = \sum_{y \in \Sigma'^n} \frac{u_y}{p(y|\theta, \theta')} \sum_{i=1}^{n-1} \sum_{x \in \Sigma^n} I_{(x_i x_{i+1} = rs)} \cdot p(x, y|\theta, \theta').$$

The innermost term is the sum of all probabilities of pairs (x, y) for an output sequence y and all state sequences x such that $x_i x_{i+1} = rs$; i.e., observing the sequence y and a transition from state r to state s at position i.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Baum-Welch Algorithm – Proof (Cont'd)

Therefore,

$$\begin{aligned}
&\sum_{x \in \Sigma^n} I_{(x_i x_{i+1} = rs)} \cdot p(x, y | \theta, \theta') \\
&= \quad P(Y = y, X_i = r, X_{i+1} = s) \\
&= \quad P(Y_1 = y_1, \ldots, Y_i = y_i, X_i = r) \\
&\quad \cdot P(X_{i+1} = s \mid X_i = r) \\
&\quad \cdot P(Y_{i+1} = y_{i+1} \mid X_{i+1} = s) \\
&\quad \cdot P(Y_{i+2} = y_{i+2}, \ldots, Y_n = y_n \mid X_{i+1} = s) \\
&= \quad f_{y,r}(i) \cdot \theta_{rs} \cdot \theta'_{s y_{i+1}} \cdot b_{y,s}(i+1).
\end{aligned}$$

The second assertion can be similarly proved. □

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Baum-Welch Algorithm

- In the BW algorithm, the $l \times n$ forward and backward probability matrices $(F_y)_y$ and $(B_y)_y$ need to be maintained to calculate the sufficient statistic (v, v') using (103, 104).
- In the EM algorithm, an $l^n \times l'^n$ matrix $U = (u_{x,y})$ needs to be maintained from which the sufficient statistic (v, v') can be established using (94).

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Baum-Welch Algorithm for HMM

Require: Hidden Markov model, joint probability function $p_{X,Y|\Theta\times\Theta'}$, parameter space $\Theta \times \Theta' \subseteq \mathbb{R}_{>0}^{l(l-1)} \times \mathbb{R}_{>0}^{l(l'-1)}$, integer $n \geq 1$, observed data $u = (u_y) \in \mathbb{N}^{l'n}$

Ensure: Maximum likelihood estimate $(\theta^*, \theta'^*) \in \Theta \times \Theta'$

[Init] Threshold $\epsilon > 0$ and parameters $(\theta, \theta') \in \Theta \times \Theta'$

[E-Step] Compute the sufficient statistic (v, v') by dynamic programming in (103, 104)

[M-Step] Compute solution $(\hat{\theta}, \hat{\theta}') \in \Theta \times \Theta'$ of the likelihood function $\ell_{X,Y}$ using (v, v') and (57, 58)

[Compare] If $\ell_Y(\hat{\theta}, \hat{\theta}') - \ell_Y(\theta, \theta') > \epsilon$, set $\theta \leftarrow \hat{\theta}$ and $\theta' \leftarrow \hat{\theta}'$ and resume with E-step, see (91)

[Output] $\theta^* \leftarrow \hat{\theta}$, $\theta'^* \leftarrow \hat{\theta}$

The difference between the EM and BW algorithms lies in the E-step, which is more efficient in the BW algorithm.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

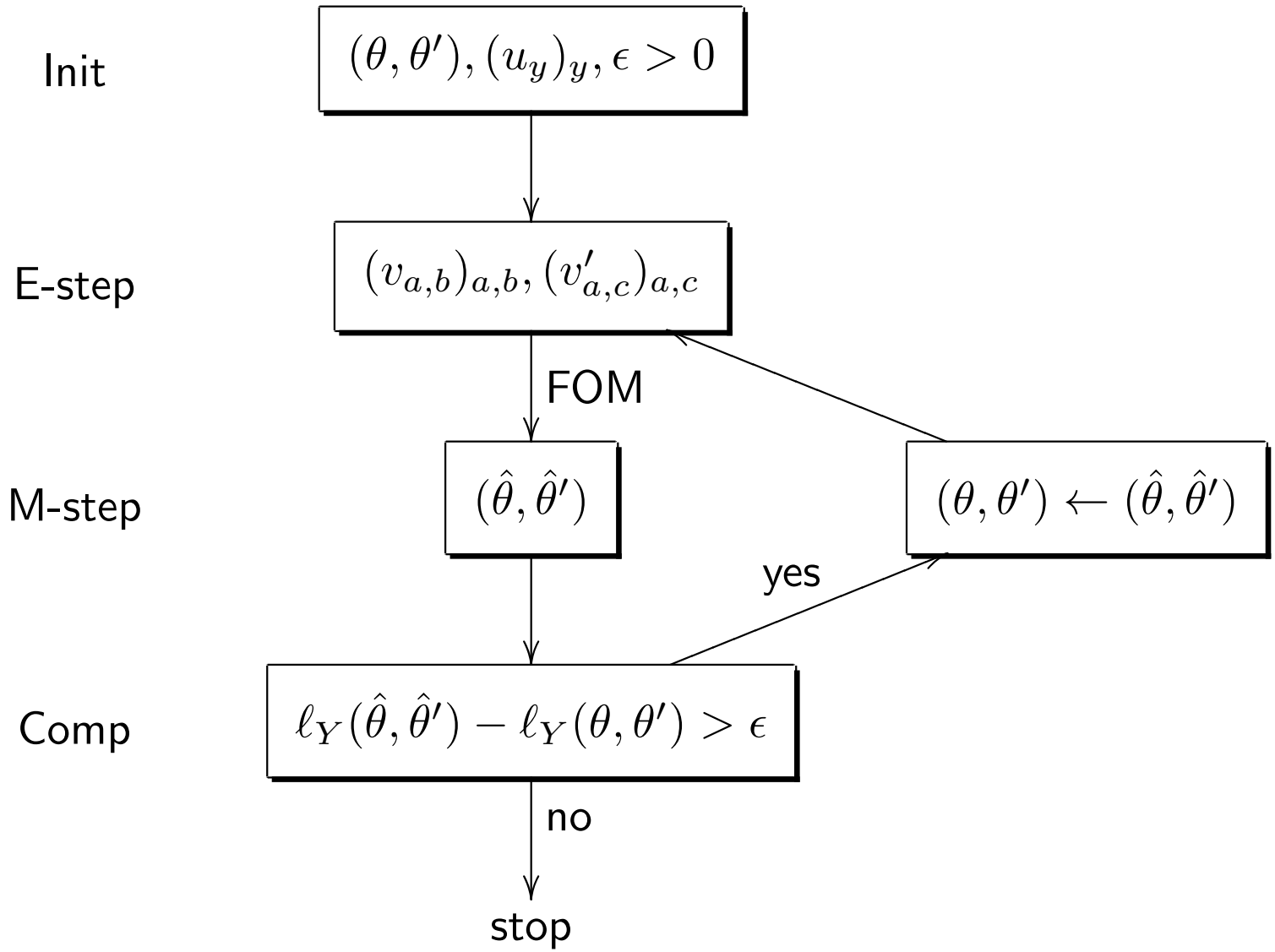
EM Algorithm

*BW Algorithm

CpG Islands

Using R

BW Algorithm for HMM



Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

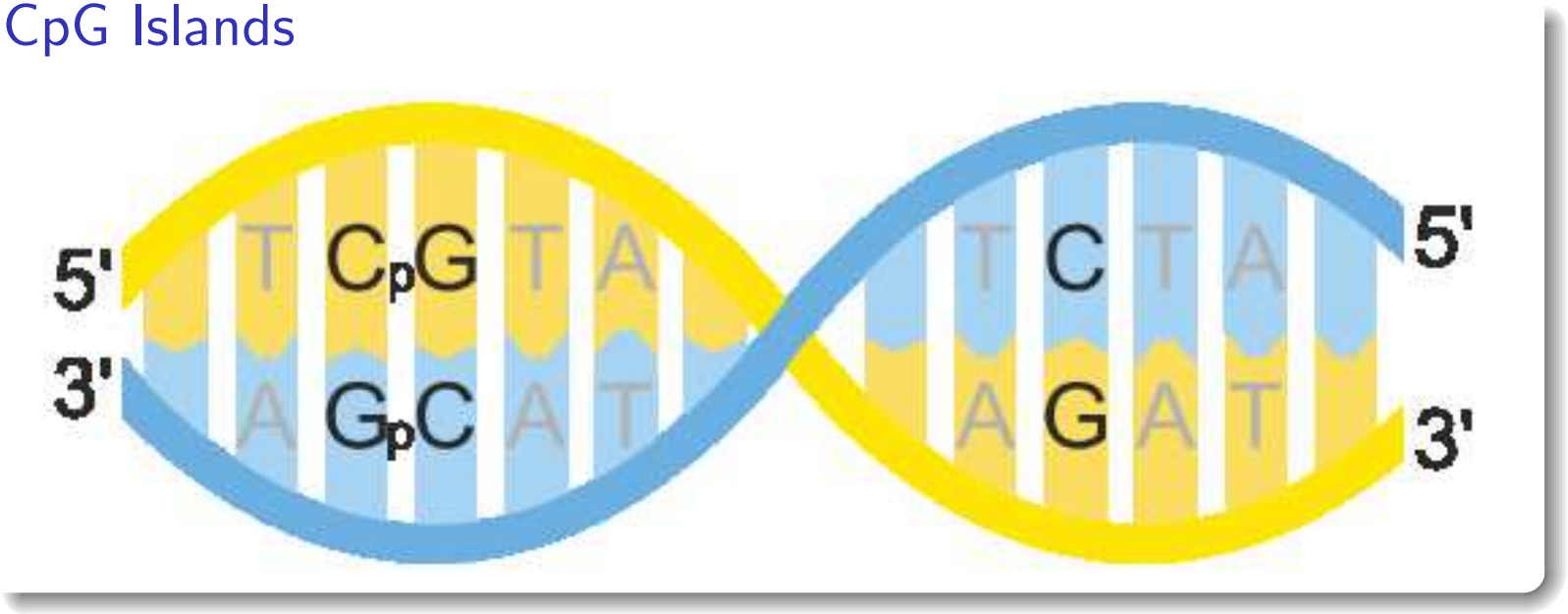
EM Algorithm

*BW Algorithm

CpG Islands

Using R

CpG Islands



Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

CpG Islands

- Region of a genome with higher frequency of CpG sites than in the rest of the genome.
- Formal definition: CpG island is a region with at least 250 bp and a GC percentage of at least 60%.
- CpG is shorthand for "–C–phosphate–G–", i.e., cytosine and guanine separated by one phosphate residue on the *same* single DNA strand (covalent bond).
- Do not mix up with the base pairs C$\cdots$G where C and G lie opposite to each other on complementary strands (hydrogen bond).

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

CpG Islands

- CpG islands located in the promoter region of genes can play important role in gene silencing.
- Cytosines in CpG dinucleotides can be methylated (have methyl group CH_3 attached) to form 5-methylcytosine.
- The presence of multiple methylated CpG sites in CpG islands of promoters causes stable silencing of genes.
- In cancer, loss of expression of genes occurs about 10 times more frequently by hypermethylation of promoter CpG islands than by mutation.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

CpG Islands

In computational biology, two questions about CpG islands arise.

1. Decide whether a short stretch of a genomic DNA strand lies inside of a CpG island.
2. Find the CpG regions of a long stretch of a genomic DNA strand.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Markov Model

- Build two Markov models from a set of human DNA sequences with a total of 48 putative CpG islands.
- The $+$ model corresponds to the regions labelled as CpG islands, the $-$ model is associated with the remaining regions.
- The conditional probabilities for each model,

$$\theta^{+}_{XY} = \frac{c^{+}_{XY}}{\sum_Z c^{+}_{XZ}} \quad \text{and} \quad \theta^{-}_{XY} = \frac{c^{-}_{XY}}{\sum_Z c^{-}_{XZ}}, \tag{105}$$

where c^{+}_{XY} and c^{-}_{XY} are the number of times the nucleotide Y followed the nucleotide X in a CpG island and non-island, respectively.

Markov Model

Conditional probabilities for + model and − model:

+	A	C	G	T
A	0.180	0.274	0.426	0.120
C	0.171	0.368	0.274	0.188
G	0.161	0.339	0.375	0.125
T	0.079	0.355	0.384	0.182

−	A	C	G	T
A	0.300	0.205	0.285	0.210
C	0.322	0.298	0.078	0.302
G	0.248	0.246	0.298	0.208
T	0.177	0.239	0.292	0.292

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Markov Model – Inference (Problem 1)

- Consider DNA sequence $w = w_1 \dots w_n$.
- Calculate in both Markov chains the probabilities of lying inside and outside of a CpG island:

$$p^+(w) = \frac{1}{4}\theta^+_{w_1,w_2}\theta^+_{w_2,w_3}\cdots\theta^+_{w_{n-1},w_n} \tag{106}$$

and

$$p^-(w) = \frac{1}{4}\theta^-_{w_1,w_2}\theta^-_{w_2,w_3}\cdots\theta^-_{w_{n-1},w_n}. \tag{107}$$

- Use log odds ratio for discrimination:

$$S(w) = \log\frac{p^+(w)}{p^-(w)} = \sum_i \log\frac{\theta^+_{w_i,w_{i+1}}}{\theta^-_{w_i,w_{i+1}}}. \tag{108}$$

If the value of $S(w)$ is positive, there is a high chance that the DNA sequence represents a CpG island; otherwise, not.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Hidden Markov Model

Build hidden Markov model for entire DNA sequence that incorporates both Markov models above.

- States are relabeled such that $\mathtt{A}_+$, $\mathtt{C}_+$, $\mathtt{G}_+$, and $\mathtt{T}_+$ determine areas of CpG island and $\mathtt{A}_-$, $\mathtt{C}_-$, $\mathtt{G}_-$, and $\mathtt{T}_-$ provide areas of non-islands.
- For simplicity, assume there is a uniform conditional probability of switching between islands and non-islands.
- Let p^+ and $p^- = 1 - p^+$ denote the probabilities for staying inside and outside of a CpG island, resp.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Hidden Markov Model

Conditional probabilities for HMM:

θ	$\mathtt{A}^+$	$\mathtt{C}^+$	$\mathtt{G}^+$	$\mathtt{T}^+$
$\mathtt{A}^+$	$0.180p^+$	$0.274p^+$	$0.426p^+$	$0.120p^+$
$\mathtt{C}^+$	$0.171p^+$	$0.368p^+$	$0.274p^+$	$0.188p^+$
$\mathtt{G}^+$	$0.161p^+$	$0.339p^+$	$0.375p^+$	$0.125p^+$
$\mathtt{T}^+$	$0.079p^+$	$0.355p^+$	$0.384p^+$	$0.182p^+$
$\mathtt{A}^-$	$\frac{1-p^-}{4}$	$\frac{1-p^-}{4}$	$\frac{1-p^-}{4}$	$\frac{1-p^-}{4}$
$\mathtt{C}^-$	$\frac{1-p^-}{4}$	$\frac{1-p^-}{4}$	$\frac{1-p^-}{4}$	$\frac{1-p^-}{4}$
$\mathtt{G}^-$	$\frac{1-p^-}{4}$	$\frac{1-p^-}{4}$	$\frac{1-p^-}{4}$	$\frac{1-p^-}{4}$
$\mathtt{T}^-$	$\frac{1-p^-}{4}$	$\frac{1-p^-}{4}$	$\frac{1-p^-}{4}$	$\frac{1-p^-}{4}$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Hidden Markov Model

Conditional probabilities for HMM:

θ	$\mathtt{A}^-$	$\mathtt{C}^-$	$\mathtt{G}^-$	$\mathtt{T}^-$
$\mathtt{A}^+$	$\frac{1-p^+}{4}$	$\frac{1-p^+}{4}$	$\frac{1-p^+}{4}$	$\frac{1-p^+}{4}$
$\mathtt{C}^+$	$\frac{1-p^+}{4}$	$\frac{1-p^+}{4}$	$\frac{1-p^+}{4}$	$\frac{1-p^+}{4}$
$\mathtt{G}^+$	$\frac{1-p^+}{4}$	$\frac{1-p^+}{4}$	$\frac{1-p^+}{4}$	$\frac{1-p^+}{4}$
$\mathtt{T}^+$	$\frac{1-p^+}{4}$	$\frac{1-p^+}{4}$	$\frac{1-p^+}{4}$	$\frac{1-p^+}{4}$
$\mathtt{A}^-$	$0.300p^-$	$0.205p^-$	$0.285p^-$	$0.210p^-$
$\mathtt{C}^-$	$0.322p^-$	$0.298p^-$	$0.078p^-$	$0.302p^-$
$\mathtt{G}^-$	$0.248p^-$	$0.246p^-$	$0.298p^-$	$0.208p^-$
$\mathtt{T}^-$	$0.177p^-$	$0.239p^-$	$0.292p^-$	$0.292p^-$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Hidden Markov Model

- The conditional output probabilities are all 0 and 1.
- The states X^+ and X^- output the symbol X with certainty,

$$\theta'_{X^+,Y} = \theta'_{X^-,Y} = \begin{cases} 1 & \text{if } X = Y, \\ 0 & \text{if } X \neq Y. \end{cases}$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Hidden Markov Model

There are three associated problems:

1. Given the model parameters, a DNA sequence (output), and a state sequence. Calculate the joint probability of this pair of sequences.

   For instance, the joint probability that the DNA sequence CGCG is generated by the state sequence $\mathtt{C}_+\mathtt{G}_+\mathtt{C}_+\mathtt{G}_+$ is

$$\begin{aligned} & p(C_+G_+C_+G_+, CGCG) \\ = \quad & \frac{1}{8}\theta'_{C_+,C}\theta_{C_+,G_+}\theta'_{G_+,G}\theta_{G_+,C_+}\theta'_{C_+,C}\theta_{C_+,G_+}\theta'_{G_+,G}. \end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Hidden Markov Model – Inference (Problem 2)

2 Given the model parameters and a DNA sequence (output). Find the maximum likelihood of generating the output sequence.

This problem can be tackled by the Viterbi algorithm which calculates all explanations. When an explanation has a substring consisting only of + states, a CpG island is predicted.

For instance, consider an output sequence and a corresponding (hypothetic) state sequence,

A	C	C	C	A	T	C	G	C	C	G	A	T	T
A^-	C^+	C^+	C^+	A^-	T^-	C^+	G^+	C^+	C^+	G^+	A^-	T^-	T^-

This state sequence suggests that the DNA strand has two CpG islands.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Hidden Markov Model – Learning

3 Given a sample set of DNA sequences (output), find the most likely set of conditional probabilities. This amounts to the estimation of the conditional probabilities of the hidden Markov model given the data set. The EM or BW algorithm can tackle this problem.

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Hidden Markov Model in R

- The R library HMM maintained by Lin Himmelmann (2010) provides procedures for learning and inference in hidden Markov models.
- Initialisation of HMM:

```
> library("HMM")
> hmm = initHMM( c("F","L"), c("h","t"),
          transProbs
           = matrix(c(0.8,0.2,0.2,0.8),2),
          emissionProbs
           = matrix(c(0.5,0.4,0.5,0.6),2))
```

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Hidden Markov Model in R

The `print` command provides a summary of the stats of the defined model.

```
> print(hmm)
$States
[1] "F" "L"

$Symbols
[1]a "h" "t"

$startProbs
  F   L
0.5 0.5

$transProbs
    to
from   F   L
   F 0.8 0.2
   T 0.2 0.8

$emissionProbs
    symbols
states   h   t
     F 0.5 0.5
     T 0.4 0.6
```

Hidden Markov Model in R

- Computation of forward probabilities for a sequence of observations:

```
> observe = c("h","t","t","t")
> logForwardProbabilities = forward( hmm, observe)
> print(logForwardProbabilities)
       index
states         1         2         3         4
     F -1.386294 -2.120264 -2.803460 -3.450556
     L -1.609438 -2.071473 -2.591868 -3.141582
```

- Computation of most probable path for a sequence of observations by Viterbi algorithm:

```
> posterior = viterbi( hmm, observe)
> print(posterior)
[1] "L" "L" "L" "L"
```

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Hidden Markov Model in R

Learning the parameters from observations using BW algorithm or `viterbiTraining`.

```
> hmm = initHMM( c("F","L"), c("h","t"),
         transProbs = matrix(c(0.8,0.2,0.2,0.8),2),
         emissionProbs = matrix(c(0.5,0.5,0.5,0.5),2))
> a = sample(c( rep("h",100), rep("t",200)))
> observe = c(a)
> bw = baumWelch(hmm, observe, 10)
```

Soft Computing

K.-H. Zimmermann

Contents

Belief Network

Learning FOM

Probabilistic Inference

Viterbi Algorithm

EM Algorithm

*BW Algorithm

CpG Islands

Using R

Hidden Markov Model in R

The `print` command provides a summary of the stats of the defined model.

```
> print(bw$hmm)
$States
[1] "F" "L"

$Symbols
[1]a "h" "t"

$startProbs
  F   L
0.5 0.5

$transProbs
    to
from   F   L
   F 0.8 0.2
   T 0.2 0.8

$emissionProbs
    symbols
states         h         t
     F 0.3333333 0.6666667
     L 0.3333333 0.6666667
```

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Part III

Sequence Alignment

Contents

- Representation of pairwise alignment
- *Point-accepted mutation (learning)
- Belief network (PHMM)
- Needleman-Wunsch algorithm
- Parametric sequence alignment
- Polytope propagation algorithm

Knowledge

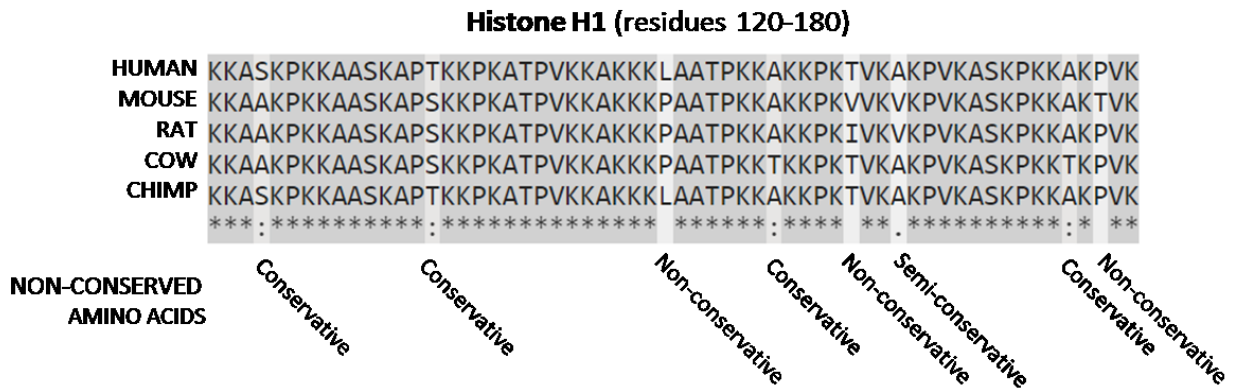
- Representation of pairwise alignment
- Pair-hidden Markov model
- *Point-accepted mutation
- Parametric sequence alignment

Skills

- Needleman-Wunsch algorithm
- Polytope propagation algorithm

Pairwise Alignment

- Pairwise alignment of DNA or AA sequences is a fundamental task in computational biology.
- Goal is to find homologous positions in biological sequences.



Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Representation of Pairwise Alignment

- Two-row representation
- Edit-string representation
- Graph representation

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Representation of Pairwise Alignment

- Given alphabet Σ with l symbols and '$-$' additional symbol, called *blank*.
- $\Sigma' = \Sigma \cup \{'-'\}$ extended alphabet of Σ.
- *DNA alphabet* $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$.
- *Extended* DNA alphabet $\Sigma' = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}, -\}$.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Representation of Pairwise Alignment

- Given two strings $y = y_1 \dots y_m$ and $z = z_1 \dots z_n$ of lengths m and n over alphabet Σ, respectively.
- The *alignment* of y and z is a pair of *aligned strings* (u, v) over Σ' such that
  - u and v have the same length,
  - u and v are copies of y and z with inserted blanks, respectively,
  - alignment (u, v) does not allow blanks at the same position: $\binom{-}{-}$.

Two aligned strings have minimal length $\max\{m, n\}$ and maximal length $m + n$.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Representation of Pairwise Alignment – Example

Given strings $y = \texttt{ACGTAGC}$ and $z = \texttt{ACCGAGACC}$ over DNA alphabet.
Alignment of y and z:

A	C	–	G	–	T	A	–	G	C
A	C	C	G	A	G	A	C	–	C

Alignment of minimal length:

A	C	–	G	–	T	A	G	C
A	C	C	G	A	G	A	C	C

Alignment of maximal length:

A	C	G	T	A	G	C	–	–	–	–	–	–	–	–	–	–
–	–	–	–	–	–	–	A	C	C	G	A	G	A	C	C

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Representation of Pairwise Alignment

- An alignment of (y, z) can be represented by a word x over the *edit alphabet* $\Pi = \{H, I, D\}$.
- The word x is called *edit string*.
- The letters of the edit alphabet stand for *homology* (H), *insertion* (I), and *deletion* (D).
- Letter I stands for insertion (indel) in the first string y.
- Letter D stands for deletion (indel) in the first string y.
- Letter H stands for character change (mutation or mismatch) including the identity change (match).
- Write $\#H$, $\#I$, and $\#D$ for the number of instances of H, I, and D in an edit string, respectively.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Representation of Pairwise Alignment – Example

Given strings $y = \texttt{ACGTAGC}$ and $z = \texttt{ACCGAGACC}$.
An alignment of y and z including the edit string x is

$$\begin{array}{cccccccccccc} x & = & H & H & I & H & I & H & H & I & D & H \\ u & = & \texttt{A} & \texttt{C} & - & \texttt{G} & - & \texttt{T} & \texttt{A} & - & \texttt{G} & \texttt{C} \\ v & = & \texttt{A} & \texttt{C} & \texttt{C} & \texttt{G} & \texttt{A} & \texttt{G} & \texttt{A} & \texttt{C} & - & \texttt{C} \end{array}$$

We have $\#H = 6$, $\#I = 3$, and $\#D = 1$.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Representation of Pairwise Alignment

Let $y \in \Sigma^m$ and $z \in \Sigma^n$. A string x over edit alphabet Π represents alignment of y and z over Σ iff the number of symbols in the edit string x satisfies

$$\#H + \#D = m \quad \text{and} \quad \#H + \#I = n. \tag{109}$$

Proof.

Given an alignment (u, v) of the pair (y, z). Then $\#H + \#D = m$ is the length of the string u and $\#H + \#I = n$ is the length of the string v.
Conversely, let x be an edit string satisfying (109). Reading the string x from left to right reproduces accordingly an alignment of the pair (y, z). □

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Representation of Pairwise Alignment

Let $\mathcal{A}_{m,n}$ be the set of all strings over the edit alphabet Π which satisfy (109).

$$D(m,n) = |\mathcal{A}_{m,n}| \tag{110}$$

is called *Delannoy number*.

Proposition

Let $m, n \geq 0$. The Delannoy number $D(m,n)$ is given by the coefficient of the monomial $X^m Y^n$ in the generating function

$$f(X,Y) = \frac{1}{1 - X - Y - XY}. \tag{111}$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Proof.

Consider the expansion of the generating function

$$\frac{1}{1-X-Y-XY} = \sum_{m=0}^{\infty}\sum_{n=0}^{\infty} a_{m,n} X^m Y^n, \quad a_{m,n} \in \mathbb{Z}. \tag{112}$$

The coefficients are characterized by the linear recurrence

$$a_{m,n} = a_{m-1,n} + a_{m,n-1} + a_{m-1,n-1} \tag{113}$$

for all $m \geq 0$, $n \geq 0$, $m+n \geq 1$, and initial condition

$$a_{0,0} = 1, \quad a_{m,-1} = 0, \quad \text{and} \quad a_{-1,n} = 0 \tag{114}$$

for all $m, n \geq 0$.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Proof (Cont'd).

The same recurrence holds for the cardinality of $\mathcal{A}_{m,n}$, as there are three possibilities for last symbol of an edit string x:

x	$=$	$\ldots$	H			
u	$=$	$\ldots$	y_m			
v	$=$	$\ldots$	z_n			
x	$=$		$\ldots$			D
u	$=$		$\ldots$			y_m
v	$=$	$\ldots$	z_n	$-$	$\ldots$	$-$
x	$=$		$\ldots$			I
u	$=$	$\ldots$	y_m	$-$	$\ldots$	$-$
v	$=$		$\ldots$			z_n

Thus

$$|\mathcal{A}_{m,n}| = |\mathcal{A}_{m-1,n}| + |\mathcal{A}_{m,n-1}| + |\mathcal{A}_{m-1,n-1}| \tag{115}$$

for all $m \geq 0$, $n \geq 0$, $m + n \geq 1$.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Proof (Cont'd).

Moreover, $\mathcal{A}_{0,0}$ has only one element, the empty string, and $\mathcal{A}_{m,n}$ is the empty set if $m < 0$ or $n < 0$.
Thus the coefficients $a_{m,n}$ and the cardinalities $|\mathcal{A}_{m,n}|$ satisfy the same initial condition and the same recurrence. Equality follows. $\square$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Representation of Pairwise Alignment

The first few Delannoy numbers $D(m, n)$:

$m \backslash n$	0	1	2	3	4	5	6	7	8
0	1	1	1	1	1	1	1	1	1
1	1	3	5	7	9	11	13	15	17
2	1	5	13	25	41	61	85	113	145
3	1	7	25	63	129	231	377	575	833
4	1	9	41	129	321	681	1,289	2,241	3,649
5	1	11	61	231	681	1,683	3,653	7,183	13,073
6	1	13	85	377	1,289	3,653	8,989	19,825	40,081
7	1	15	113	575	2,241	7,183	19,825	48,639	108,545
8	1	17	145	833	3,649	13,073	40,081	108,545	265,729
9	1	19	181	1,159	5,641	22,363	75,517	224,143	598,417
10	1	21	221	1,561	8,361	36,365	134,245	433,905	1,256,465

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Representation of Pairwise Alignment

Given strings y and z of lengths m and n over Σ, resp.
The *alignment graph* of strings y and z is the directed graph $G = G_{m,n}$ with node set

$$N(G) = \{0, 1, \ldots, m\} \times \{0, 1, \ldots, n\}$$

and edge set $E(G)$ with

- (east) edge $(i, j) \to (i, j + 1)$ labelled I,
- (south) edge $(i, j) \to (i + 1, j)$ labelled D, and
- (south-east) edge $(i, j) \to (i + 1, j + 1)$ labelled H.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Representation of Pairwise Alignment – Example

Given sequences $y = \texttt{ACG}$ and $z = \texttt{ACC}$ over the DNA alphabet. The edit string $x = HHDI$ provides the alignment

$$\begin{array}{cccc} H & H & D & I \\ \texttt{A} & \texttt{C} & \texttt{G} & - \\ \texttt{A} & \texttt{C} & - & \texttt{C} \end{array}$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

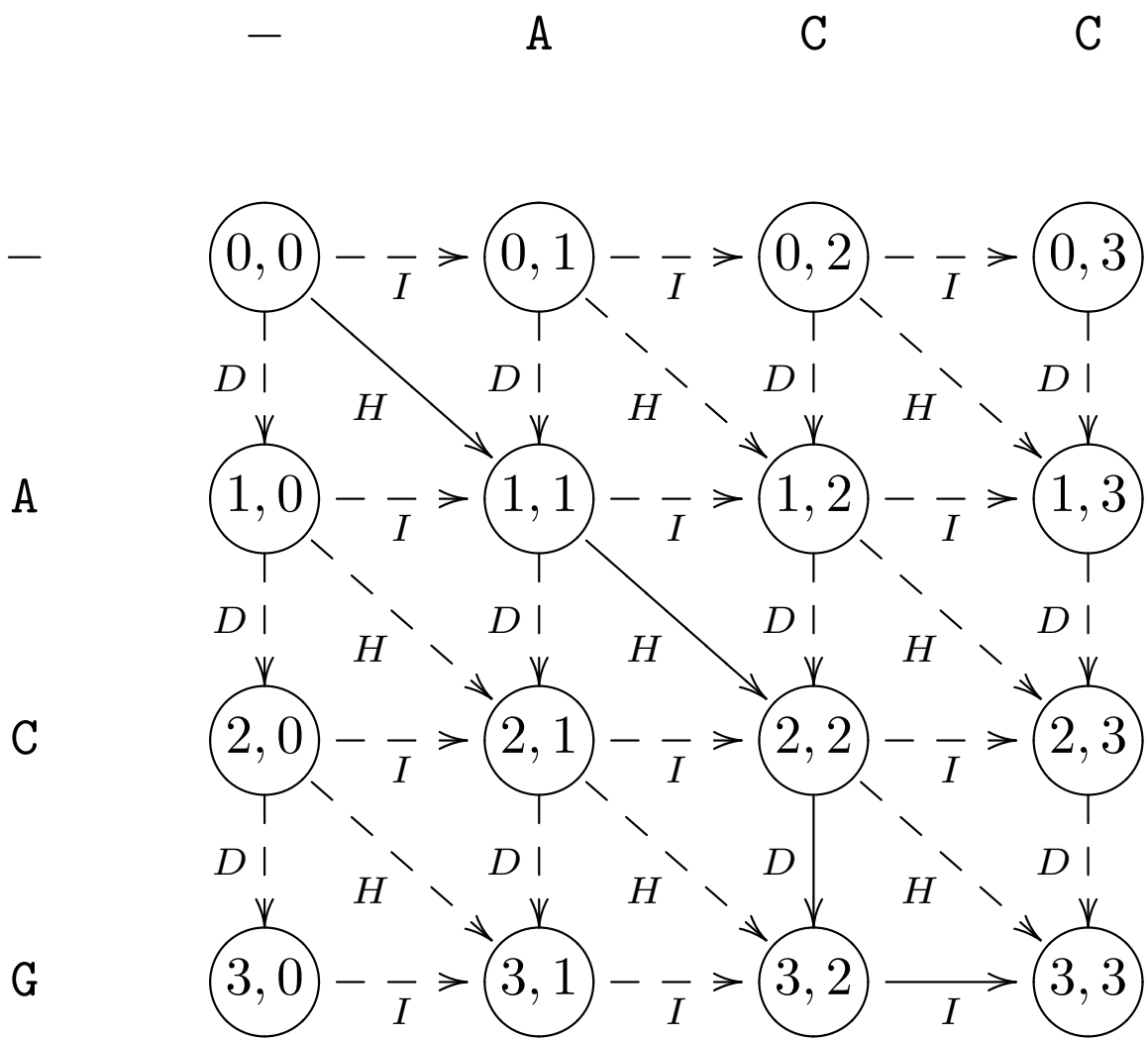
Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Representation of Pairwise Alignment – Example (Cont'd)

The alignment $x = HHDI$ can be traced by the solid path in the alignment graph $G_{3,3}$:



Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Representation of Pairwise Alignment

The set of all alignments $\mathcal{A}_{m,n}$ corresponds one-to-one with the set of all directed paths from node $(0,0)$ to node (m,n) in the alignment graph $G_{m,n}$.

Proof.

- Let x be an edit string of $\mathcal{A}_{m,n}$. Then (109) holds and so x provides a path in the graph $G_{m,n}$ from $(0,0)$ to (m,n).
- Conversely, given a path in the graph $G_{m,n}$ from $(0,0)$ to (m,n). The labeling of the path provides a string x over the edit alphabet that satisfies (109). Thus the string x is an edit string corresponding to an alignment in $\mathcal{A}_{m,n}$.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Representation of Pairwise Alignment

Let $m, n \geq 0$. Then

$$D(m,n) = \sum_{k=0}^{\min\{m,n\}} \binom{m+n-k}{m-k} \binom{n}{k}. \tag{116}$$

Proof.

Let $k \geq 0$. For k homology steps (H), there must be $m-k$ deletion steps (D) and $n-k$ insertion steps (I) in order to travel from node $(0,0)$ to node (m,n). These steps can be performed in any order and thus can be described by all words of length $m+n-k$ with k symbols H, $m-k$ symbols D, and $n-k$ symbols I. The number of these words is given by the multinomial coefficient

$$\binom{m+n-k}{k, m-k, n-k} = \binom{m+n-k}{m-k} \binom{n}{k}.$$

We have $k = 0, \ldots, \max\{m,n\}$. □

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

*PAM

- Proteins and amino acids
- Point-accepted mutations
- PAM matrices

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Proteins and Amino Acids

- Protein: large biomolecule consisting of one or more long chains of amino acid residues.
- Functionality includes catalysis of metabolic reactions, replication of DNA, transport of modules.
- A protein is transcribed and translated from the nucleotide sequence of a gene and then folds into three-dimensional structure. Correct folding is required to provide functionality.
- Proteins are linear polymers consisting of a linear sequence of up to 20 distinct amino acids.
- Amino acids are biomolecules containing an amino ($-\mathtt{NH}_2$) group, a carboxyl ($-\mathtt{COOH}$) group, and a side chain ($\mathtt{R}$ group) which is specific to each amino acid.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

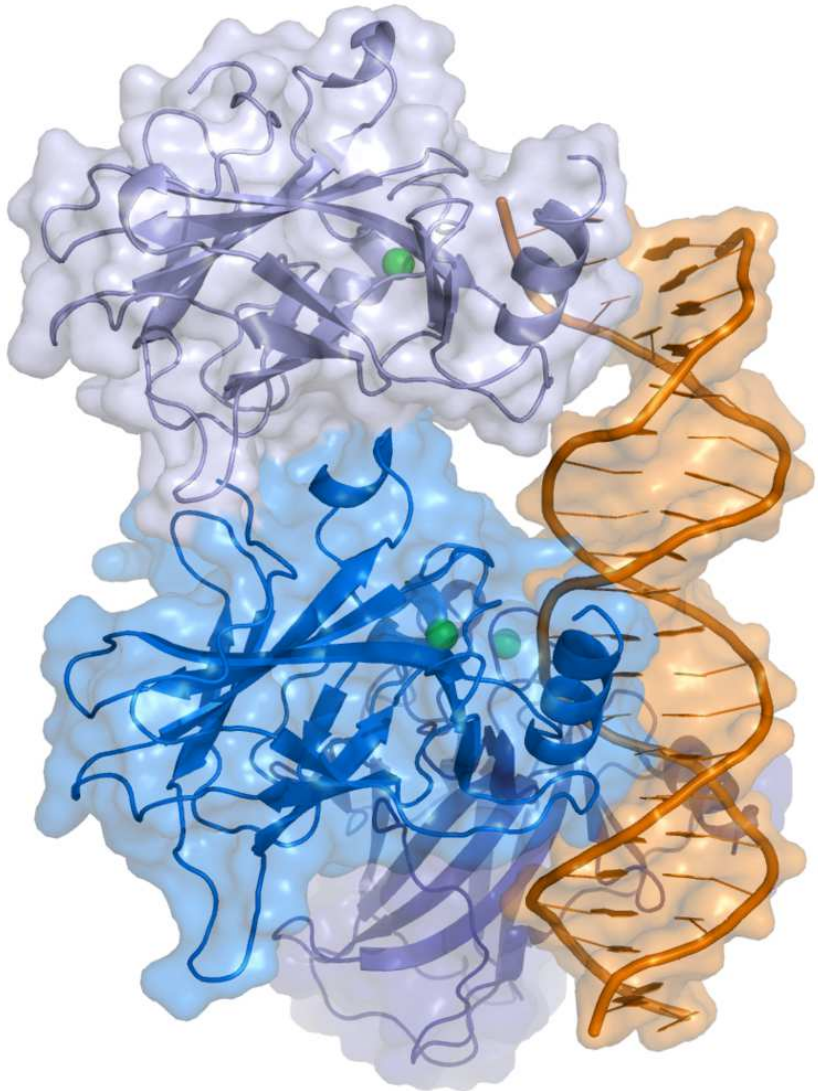
PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Tumor-Suppressor P53



Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Point-Accepted Mutations

- *Point accepted mutation* (PAM) is the substitution of a single amino acid in the AA sequence (primary structure) of a protein by another single amino acid, which is accepted by the process of natural selection.
- This definition does not include all point mutations in the DNA of an organism. E.g., silent mutations which do not significantly change the phenotype of an organism are not accepted mutations. The same holds for lethal mutations or mutations rejected by natural selection in other ways.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Point-Accepted Mutations

- *PAM matrix* is a 20×20 matrix $P = (p_{ij})$ in which the rows and columns are labeled by the 20 standard amino acids.
- Entry p_{ij} provides the conditional probability of amino acid i being substituted by amino acid j through a sequence of one or more point accepted mutations during a given evolutionary interval.
- There are distinct PAM matrices for different lengths of evolutionary intervals.
- The PAM matrices were invented by Margaret Dayhoff (1978) using 1572 observed mutations in the phylogenetic tree of 71 families of closely related proteins. (Alignments of these proteins have $\geq 85\%$ identity resulting supposingly from single mutation event.)

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Point-Accepted Mutations

Dayhoff's procedure for the invention of the PAM matrices.

- $A = (a_{ij})$ is a 20×20 matrix (labeled by AAs) recording the number of mismatches.
- A is symmetric, since samples come from organisms alive today and so the direction of a mutation cannot be determined.
- Diagonal entries of A are neglected.
- N is the total number of amino acids.

Next record mutability and frequency of the occurring amino acids.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Point-Accepted Mutations

- Frequency of amino acid i with n_i occurrences,

$$f_i = \frac{n_i}{N}. \tag{117}$$

- Mutability of amino acid i,

$$m_i = \frac{1}{n_i} \sum_{\substack{j=1 \\ j \neq i}}^{20} a_{ij}. \tag{118}$$

- Relation between frequency and mutability,

$$N f_i = n_i = \frac{1}{m_i} \sum_{\substack{j=1 \\ j \neq i}}^{20} a_{ij}. \tag{119}$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Point-Accepted Mutations

- $M = (m_{ij})$ is a 20×20 mutation matrix (labeled by AAs).
- Entry m_{ij} is the conditional probability that amino acid i mutates into amino acid j.
- The non-diagonal entries are defined as

$$m_{ij} = \mu a_{ij} \frac{m_i}{\sum_{\substack{j=1 \\ j \neq i}}^{20} a_{ij}} = \mu \frac{a_{ij}}{N f_i} = \mu \frac{a_{ij}}{n_i}, \tag{120}$$

  where μ is a constant of proportionality.
- Since the conditional probabilities of amino acid i to mutate must sum up to 1, we have

$$m_{ii} = 1 - \sum_{\substack{j=1 \\ j \neq i}}^{20} m_{ij}. \tag{121}$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Point-Accepted Mutations

- Relation between mutability and mutation,

$$\begin{aligned} m_{ii} &= 1 - \sum_{\substack{j=1 \\ j \neq i}}^{20} m_{ij} \\ &= 1 - \sum_{\substack{j=1 \\ j \neq i}}^{20} \mu \frac{a_{ij}}{n_i} \\ &= 1 - \mu \frac{1}{n_i} \sum_{\substack{j=1 \\ j \neq i}}^{20} a_{ij} \\ &= 1 - \mu m_i. \end{aligned} \tag{122}$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Point-Accepted Mutations

- The symmetry of matrix $A = (a_{ij})$ recording the number of mismatches gives the *relation of detailed balance*:

$$f_i m_{ij} = \frac{\mu}{N} a_{ij} = \frac{\mu}{N} a_{ji} = f_j m_{ji}. \tag{123}$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Point-Accepted Mutations

- Compute the values of mutation matrix $M = (m_{ij})$ for a short time frame.
- Then establish matrices for longer time periods using the assumption that the mutations can be modeled by a homogeneous Markov chain.
- Base unit of time for the PAM matrices is called *PAM unit* (time required for one mutation to emerge per 100 amino acids).
- Constant μ controls the proportion of amino acids left unchanged.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Point-Accepted Mutations

- The mutation matrix for the PAM-1 matrix is established by assuming that 99% of the amino acids in a sequence are conserved. The expression $n_i m_{ii}$ is the number of conserved amino acid units of type i. Thus the total number of conserved amino acids is given by

$$\sum_{i=1}^{20} n_i m_{ii} = \sum_{i=1}^{20} n_i (1 - \mu m_i) = N - \mu N \sum_{i=1}^{20} f_i m_i, \quad (124)$$

  where $n_i = N f_i$ and $\sum_i n_i = N$.

- The value μ needs to be chosen such that 99% identical amino acids are produced after mutation. This gives the equation

$$1 - \mu \sum_{i=1}^{20} f_i m_i = 0.99. \quad (125)$$

  This μ value can be used in the mutation matrix for the PAM-1 matrix.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Point-Accepted Mutations

- Consider the homogeneous Markov chain of protein mutation whose transition matrix is given by the mutation matrix $M_1 = M$ of the PAM-1 matrix. Then for each integral time step $n \geq 1$, the mutation matrix M_n of the PAM-n matrix is defined by the n-th power of the mutation matrix M_1,

$$M_n = M_1^n. \tag{126}$$

Extraploration into the future.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

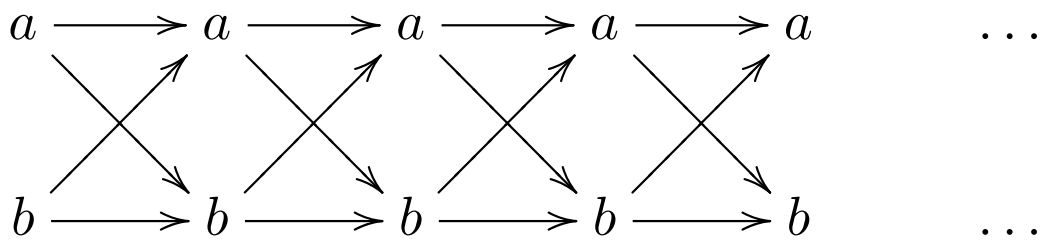
PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Homogeneous Markov Chains



- Initial probability distribution

$$p_0 = \begin{pmatrix} p_a \\ p_b \end{pmatrix}.$$

- Transition matrix of conditional probabilities

$$M = \begin{pmatrix} m_{aa} & m_{ab} \\ m_{ba} & m_{bb} \end{pmatrix}.$$

- Probability distribution after n steps,

$$p_n = M^n p_0, \quad n \geq 0.$$

PAM Matrices

- The PAM-n matrix is constructed from the ratio of the probability of point accepted mutations substituting amino acid j by amino acid i, to the probability of these amino acids to be aligned by chance; that is, the entries of the PAM-n matrix $P_n = (p_{ij}^{(n)})$ are given by

$$p_{ij}^{(n)} = \log \frac{f_i m_{ij}^{(n)}}{f_i f_j} = \log \frac{m_{ij}^{(n)}}{f_j}, \tag{127}$$

  where $M_n = (m_{ij}^{(n)})$ is the corresponding mutation matrix.

- Note that if the *alignment* of two proteins using the PAM-n matrix is considered and the proteins are related, then the evolutionary interval separating them is the time taken for n point accepted mutations to take place per 100 amino acids.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

PAM Matrices

Each PAM matrix P_n is symmetric.

Proof.

Claim that for each integer $k \geq 1$, the non-diagonal entries of the mutation matrix M_n satisfy the relation of detailed balance,

$$f_i m_{ij}^{(k)} = f_j m_{ji}^{(k)}. \tag{128}$$

Indeed, by (123), $f_i m_{ij} = f_j m_{ji}$ for the non-diagonal entries of the mutation matrix $M = M_1$. Suppose the assertion holds for some integer $k \geq 1$.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Proof (Cont'd).

By using the expansion of the product $M^{k+1} = M^k \cdot M = M \cdot M^k$, we obtain

$$\begin{aligned} f_i m_{ij}^{(k+1)} &= f_i \sum_{l=1}^{20} m_{il}^{(k)} m_{lj} = \sum_{l=1}^{20} (f_i m_{il}^{(k)}) m_{lj} \\ &= \sum_{l=1}^{20} (f_l m_{li}^{(k)}) m_{lj} = \sum_{l=1}^{20} m_{li}^{(k)} (f_l m_{lj}) \\ &= \sum_{l=1}^{20} m_{li}^{(k)} (f_j m_{jl}) = f_j \sum_{l=1}^{20} m_{jl} m_{li}^{(k)} \\ &= f_j m_{ji}^{(k+1)}. \end{aligned}$$

Thus the non-diagonal entries of the PAM-k matrix satisfy

$$p_{ij}^{(k)} = \log \frac{f_i m_{ij}^{(k)}}{f_i f_j} = \log \frac{f_j m_{ji}^{(k)}}{f_j f_i} = p_{ji}^{(k)}. \tag{129}$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

PAM Matrices

- The choice of the PAM-n matrix for some integer $n \geq 1$ requires to determine the number of mutations that have occurred per 100 amino acids. The value n is usually not accessible and therefore needs to be estimated.
- However, if two proteins are compared, the number m of mutated amino acids per 100 amino acids can be calculated. These values are approximately related as follows,

$$\frac{m}{100} = 1 - e^{-n/100}. \tag{130}$$

- PAM-250 matrix is the commonly used scoring matrix for protein sequence comparison.
- Another class of scoring matricxes are known as BLOSUM; PAM-250 matrix is comparable with BLOSUM45 matrix.
- BLOSUM looks directly at mutations in motifs of related sequences, while PAM extrapolates evolutionary information making use of closely related sequences.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

PAM Matrices

PAM250 scoring matrix

Cys	12																					
Ser	0	2																				
Thr	-2	1	3																			
Pro	-1	1	0	6																		
Ala	-2	1	1	1	2																	
Gly	-3	1	0	-1	1	5																
Asn	-4	1	0	-1	0	0	2															
Asp	-5	0	0	-1	0	1	2	4														
Glu	-5	0	0	-1	0	0	1	3	4													
Gln	-5	-1	-1	0	0	-1	1	2	2	4												
His	-3	-1	-1	0	-1	-2	2	1	1	3	6											
Arg	-4	0	-1	0	-2	-3	0	-1	-1	1	2	6										
Lys	-5	0	0	-1	-1	-2	1	0	0	1	0	3	5									
Met	-5	-2	-1	-2	-1	-3	-2	-3	-2	-1	-2	0	0	6								
Ile	-2	-1	0	-2	-1	-3	-2	-2	-2	-2	-2	-2	-2	2	5							
Leu	-6	-3	-2	-3	-2	-4	-3	-4	-3	-2	-2	-3	-3	4	2	6						
Val	-2	-1	0	-1	0	-1	-2	-2	-2	-2	-2	-2	-2	2	4	2	4					
Phe	-4	-3	-3	-5	-4	-5	-4	-6	-5	-5	-2	-4	-5	0	1	2	-1	9				
Tyr	0	-3	-3	-5	-3	-5	-2	-4	-4	-4	0	-4	-4	-2	-1	-1	-2	7	10			
Trp	-8	-2	-5	-6	-6	-7	-4	-7	-7	-5	-3	2	-3	-4	-5	-2	-6	0	0	17		
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W		

2

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Pair Hidden Markov Model

Given two sequences y and z over Σ of length m and n, resp.

- Conditional probabilities

$$\theta_{a,b}, \quad a,b \in \Sigma', \quad \text{and} \quad \theta'_{x,x'}, \quad x,x' \in \Pi. \tag{131}$$

- Probability of aligned pair (u,v) of (y,z) with edit string x,

$$p_{X,Y,Z}(x,u,v) = \prod_{i=1}^{|x|} \theta_{u_i,v_i} \cdot \prod_{i=1}^{|x|-1} \theta'_{x_i,x_{i+1}}, \tag{132}$$

where the initial probabilities are uniform (neglected).

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Pair Hidden Markov Model – Example

Pairwise alignment:

$$\begin{array}{cccccccc} x & = & H & H & I & H & D & H \\ u & = & \mathtt{A} & \mathtt{C} & - & \mathtt{G} & \mathtt{G} & \mathtt{T} \\ v & = & \mathtt{A} & \mathtt{G} & \mathtt{C} & \mathtt{G} & - & \mathtt{G} \end{array}$$

Probability of alignment

$$p_{X,Y,Z}(x,u,v) = \theta_{A,A}\theta'_{H,H}\theta_{C,G}\theta'_{H,I}\theta_{-,C}\theta'_{I,H}\theta_{G,G}\theta'_{H,D}\theta_{G,-}\theta'_{D,H}\theta_{T,G}.$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Pair Hidden Markov Model

Given two sequences y and z over Σ of length m and n, resp.

- The probability of the sequences y and z is given by the marginal probability

$$p_{Y,Z}(y,z) = \sum_{x \in \mathcal{A}_{m,n}} p_{X,Y,Z}(x, y^{(x)}, z^{(x)}) \tag{133}$$

  where $(y^{(x)}, z^{(x)})$ is the alignment of the strings (y,z) given by edit string $x \in \mathcal{A}_{m,n}$.

- $\mathcal{A}_{m,n;i}$ denotes the set of all edit strings in $\mathcal{A}_{m,n}$ of length i,

$$p_{Y,Z}(y,z) = \sum_{i=\max\{m,n\}}^{m+n} \sum_{x \in \mathcal{A}_{m,n;i}} p_{X,Y,Z}(x, y^{(x)}, z^{(x)}) \tag{134}$$

- Runtime $O((m+n)D(m,n))$, since $\mathcal{A}_{m,n}$ has $D(m,n)$ elements, x has length $\leq m+n$, and $p_{X,Y,Z}(x, y^{(x)}, z^{(x)})$ has $\leq 2(m+n)$ factors.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Marginal Probabilities

- Computation of marginal probabilites
- Sum-product decomposition of marginal probabilities

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Marginal Probabilities

Given strings y and z of lengths m and n over Σ, resp.

- The probability of y and z is given by the marginal probability

$$p_{Y,Z}(y,z) = \sum_{x \in \mathcal{A}_{m,n}} p_{X,Y,Z}(x, y^{(x)}, z^{(x)}) \tag{135}$$

  where $(y^{(x)}, z^{(x)})$ is the alignment of the strings (y,z) given by edit string $x \in \mathcal{A}_{m,n}$.

- $y_{\leq i}$ denotes the prefix $y_1 \dots y_i$ of y of length i, $1 \leq i \leq m$,
- $z_{\leq j}$ denotes the prefix $z_1 \dots z_j$ of z of length j, $1 \leq j \leq n$.
- $M[i,j,x]$ is the probability of observing the aligned pair of prefixes $y_{\leq i}$ and $z_{\leq j}$ and x is the last symbol in the edit string.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Sum-Product Decomposition

Decomposition of marginal probability $p_{Y,Z}(y,z)$:

$$p_{Y,Z}(y,z) = \sum_{x \in \Pi} M[m,n,x], \tag{136}$$

where

$$M[i,j,I] = \theta_{-,z_j} \cdot \sum_{x \in \Pi} \theta'_{x,I} \cdot M[i,j-1,x], \tag{137}$$

$$M[i,j,D] = \theta_{y_i,-} \cdot \sum_{x \in \Pi} \theta'_{x,D} \cdot M[i-1,j,x], \tag{138}$$

$$M[i,j,H] = \theta_{y_i,z_j} \cdot \sum_{x \in \Pi} \theta'_{x,H} \cdot M[i-1,j-1,x]. \tag{139}$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Sum-Product Decomposition

The alignment of the prefixes $y_{\leq i}$ and $z_{\leq j}$:

$$
\begin{array}{|lcccccc|l|}
\hline
h & = & & \ldots & & & & I \\
y_{\leq i} & = & \ldots & y_i & - & \ldots & & - \\
z_{\leq j} & = & & \ldots & & & & z_j \\
\hline
h & = & & \ldots & & & & D \\
y_{\leq i} & = & & \ldots & & & & y_i \\
z_{\leq j} & = & \ldots & z_j & - & \ldots & & - \\
\hline
\end{array}
$$

$$
\begin{array}{|lcc|l|}
\hline
h & = & \ldots & H \\
y_{\leq i} & = & \ldots & y_i \\
z_{\leq j} & = & \ldots & z_j \\
\hline
\end{array}
$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Sum-Product Decomposition

Initial conditions:

$$
\begin{aligned}
M[0,0,x] &= 1, \quad x \in \Pi, && (140)\\
M[0,j,x] &= 0, \quad x \in \{H,D\},\ 1 \le j \le n, && (141)\\
M[i,0,x] &= 0, \quad x \in \{H,I\},\ 1 \le i \le m, && (142)\\
M[0,j,I] &= M[0,j-1,I] \cdot \theta'_{I,I} \cdot \theta_{-,z_j}, && (143)\\
M[i,0,D] &= M[i-1,0,D] \cdot \theta'_{D,D} \cdot \theta_{y_i,-}. && (144)
\end{aligned}
$$

Runtime $O(mn)$, since the table M has $O(mn)$ entries and each entry is computed in a constant number of steps.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Sum-Product Decomposition - Example (Maple)

For $y = \texttt{ACG}$ and $z = \texttt{ACC}$ we obtain $p_{Y,Z}(y, z)$ as

```
20*tC_^2*t_A*t_C*t_G*tA_
+ 6*tC_^2*t_G*t_C*tAA
+ 3*tC_^2*t_G*t_A*tCA
+ tC_^2*t_A*t_C*tGA
+ 4*tC_*t_G*t_C*tA_*tAC
+ 7*tC_*t_G*tCC*t_A*tA_
+ 3*tC_*t_G*tCC*tAA
+ 9*tC_*tGC*t_A*t_C*tA_
+ 3*tC_*tGC*t_C*tAA
+ 2*t_C*tGC*tA_*tCA
+ t_G*tCC*tA_*tAC
+ tGC*t_C*tA_*tAC
+ 2*tGC*tCC*t_A*tA_
+ tGC*tCC*tAA.
```

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Sum-Product Decomposition - Example (Cont'd)

- The expression $p_{Y,Z}(y,z)$ has 14 terms and each term stands for an alignment.
- The total number of alignments is $D(3,3) = 63$ (equals the sum of coefficients).
- For instance, the term $2 * \_\mathtt{C} * \mathtt{GC} * \mathtt{A}\_ * \mathtt{CA}^a$ stands for the alignments

$$\begin{array}{cccc} \mathtt{A} & \mathtt{C} & \mathtt{G} & - \\ - & \mathtt{A} & \mathtt{C} & \mathtt{C} \end{array} \quad \text{and} \quad \begin{array}{cccc} \mathtt{A} & \mathtt{C} & - & \mathtt{G} \\ - & \mathtt{A} & \mathtt{C} & \mathtt{C} \end{array}$$

[a] Commuting variables.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Needleman-Wunsch (NW) Algorithm

- Tropicalization of sum-product decomposition
- Needleman-Wunsch algorithm (dynamic programming)

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Tropicalization of Sum-Product Decomposition

- Given strings y and z of lengths m and n over Σ, respectively.
- The objective is to find one (or all) edit strings $x \in \mathcal{A}_{m,n}$ which maximize the likelihood

$$p_{X|Y,Z}(x|y,z) = \frac{p_{X,Y,Z}(x,y,z)}{p_{Y,Z}(y,z)}. \tag{145}$$

- Since the observed sequences y and z are fixed, the likelihood $p_{X|Y,Z}(x|y,z)$ is directly proportional to the joint probability $p_{X,Y,Z}(x,y,z)$ provided that $p_{Y,Z}(y,z) > 0$.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Tropicalization of Sum-Product Decomposition

- Let $p_{Y,Z}(y,z) > 0$. The aim is to find the state sequences (edit strings) $\bar{x} \in \mathcal{A}_{m,n}$ with

$$\bar{x} = \operatorname{argmax}_{x \in \mathcal{A}_{m,n}} \{p_{X|Y,Z}(x|y^{(x)}, z^{(x)})\} \quad (146)$$

$$= \operatorname{argmax}_{x \in \mathcal{A}_{m,n}} \{p_{X,Y,Z}(x, y^{(x)}, z^{(x)})\}, \quad (147)$$

  where $(y^{(x)}, z^{(x)})$ is the alignment of the strings (y,z) given by the edit string $x \in \mathcal{A}_{m,n}$.

- Each optimal edit string $\bar{x}$ is called an *explanation* of the observed pair (y,z). The explanations can be found by tropicalization.

- *Tropicalization*: Replace each sum by tropial sum and each product by tropical product.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Tropicalization of Sum-Product Decomposition

- Put $w(y,z) = -\log p_{Y,Z}(y,z)$ and $w(x,y,z) = -\log p_{X,Y,Z}(x,y,z)$.
- Then tropicalization of (133) gives

$$w(y,z) = \bigoplus_{x \in \mathcal{A}_{m,n}} w(x, y^{(x)}, z^{(x)}). \tag{148}$$

- Thus the tropicalization of the marginal distribution $p_{Y,Z}(y,z)$ solves the alignment problem:

$$w(y,z) = \min_{x \in \mathcal{A}_{m,n}} w(x, y^{(x)}, z^{(x)}). \tag{149}$$

The explanations $\bar{x}$ are obtained by evaluation in the tropical algebra,

$$\bar{x} = \operatorname{argmin}_{x \in \mathcal{A}_{m,n}} \{w(x, y^{(x)}, z^{(x)})\}. \tag{150}$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Tropicalization of Sum-Product Decomposition

- Write

$$M[i,j,x] := -\log M[i,j,x] \tag{151}$$

for $1 \leq i \leq m$, $1 \leq j \leq n$, and $x \in \Pi$.

- Put

$$u(a,b) = -\log \theta_{a,b}, \quad a,b \in \Sigma', \tag{152}$$

$$v(x,x') = -\log \theta'_{x,x'} \quad x,x' \in \Pi. \tag{153}$$

- By tropicalization of (136),

$$w(y,z) = \bigoplus_{x \in \Pi} M[m,n,x]. \tag{154}$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Tropicalization of Sum-Product Decomposition

$$M[i,j,I] = u(-,z_j) \odot \bigoplus_{x\in\Pi} v(x,I) \odot M[i,j-1,x], \quad (155)$$

$$M[i,j,D] = u(y_i,-) \odot \bigoplus_{x\in\Pi} v(x,D) \odot M[i-1,j,x], \quad (156)$$

$$M[i,j,H] = u(y_i,z_j) \odot \bigoplus_{x\in\Pi} v(x,H) \odot M[i-1,j-1,x] \quad (157)$$

and

$$M[0,0,x] = 0, \quad x \in \Pi, \quad (158)$$

$$M[0,j,x] = \infty, \quad x \in \{H,D\},\ 1 \le j \le n, \quad (159)$$

$$M[i,0,x] = \infty, \quad x \in \{H,I\},\ 1 \le i \le m, \quad (160)$$

$$M[0,j,I] = M[0,j-1,I] \odot v(I,I) \odot u(-,z_j), \quad (161)$$

$$M[i,0,D] = M[i-1,0,D] \odot v(D,D) \odot v(y_i,-). \quad (162)$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Needleman-Wunsch Algorithm

- The NW algorithm is a special case of this sum-product algorithm neglecting the conditional probabilities corresponding to the edit alphabet.
- Runtime $O(mn)$, since the table M has $O(mn)$ entries and each entry is computed in a constant number of steps.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Needleman-Wunsch Algorithm

Require: Strings $y \in \Sigma^m$, $z \in \Sigma^n$, weights $u(a, b)$ for $a, b \in \Sigma'$
Ensure: Edit string $x \in \mathcal{A}_{m,n}$ with minimal weight $w(y^{(x)}, z^{(x)})$

$M \leftarrow \text{matrix}[0..m, 0..n]$
$M[0, 0] \leftarrow 0$
for $i \leftarrow 1$ to m **do**
  $M[i, 0] \leftarrow M[i-1, 0] + u(y_i, -)$
end for
for $j \leftarrow 1$ to n **do**
  $M[0, j] \leftarrow M[0, j-1] + u(-, z_j)$
end for
for $i \leftarrow 1$ to m **do**
  for $j \leftarrow 1$ to n **do**
    $M[i, j] \leftarrow \min\{M[i-1, j-1] + u(y_i, z_j), M[i-1, j] + u(y_i, -), M[i, j-1] + u(-, z_j)\}$
    Color the edges directed to (i, j) that attain the minimum
  end for
end for
Trace a path P in the backward direction from (m, n) to $(0, 0)$ by following an arbitrary sequence of colored edges.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

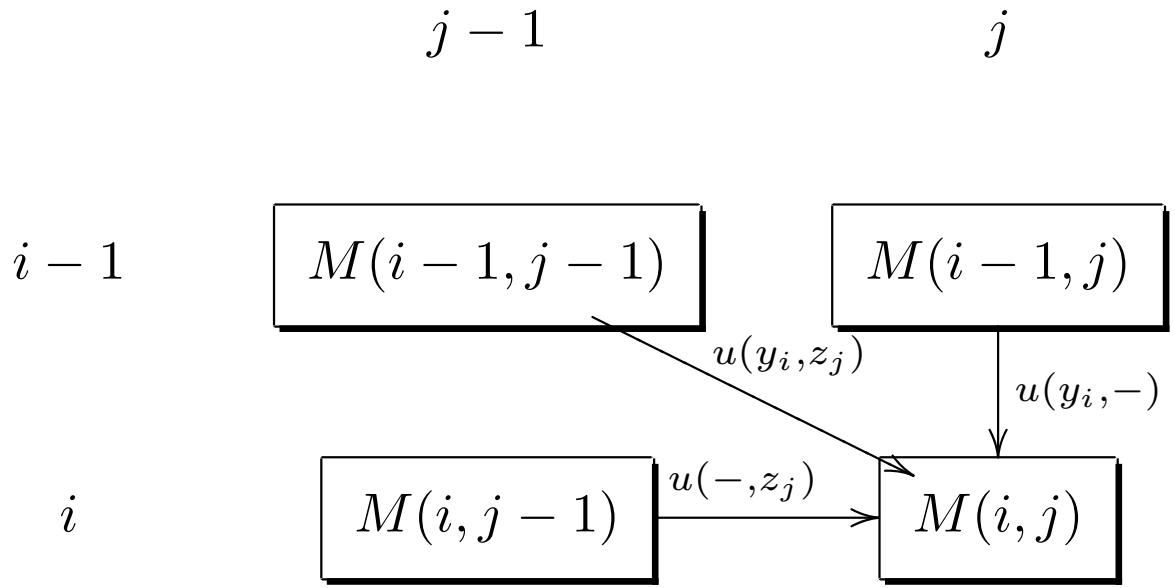
PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Needleman-Wunsch Algorithm



$M[i,j]$ is the minimum of

- $M[i-1,j-1]+u(y_i,z_j)$,
- $M[i-1,j]+u(y_i,-)$,
- $M[i,j-1]+u(-,z_j)$.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Needleman-Wunsch Algorithm – Example

- Given strings $y = \texttt{ACG}$ and $z = \texttt{ACC}$.
- Scores: matches -3, mismatches 1, indels 2.
- Optimal alignment with score -5:

$$\begin{array}{lcccc} x & = & H & H & H \\ y^{(x)} & = & \texttt{A} & \texttt{C} & \texttt{G} \\ z^{(x)} & = & \texttt{A} & \texttt{C} & \texttt{C} \end{array}$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

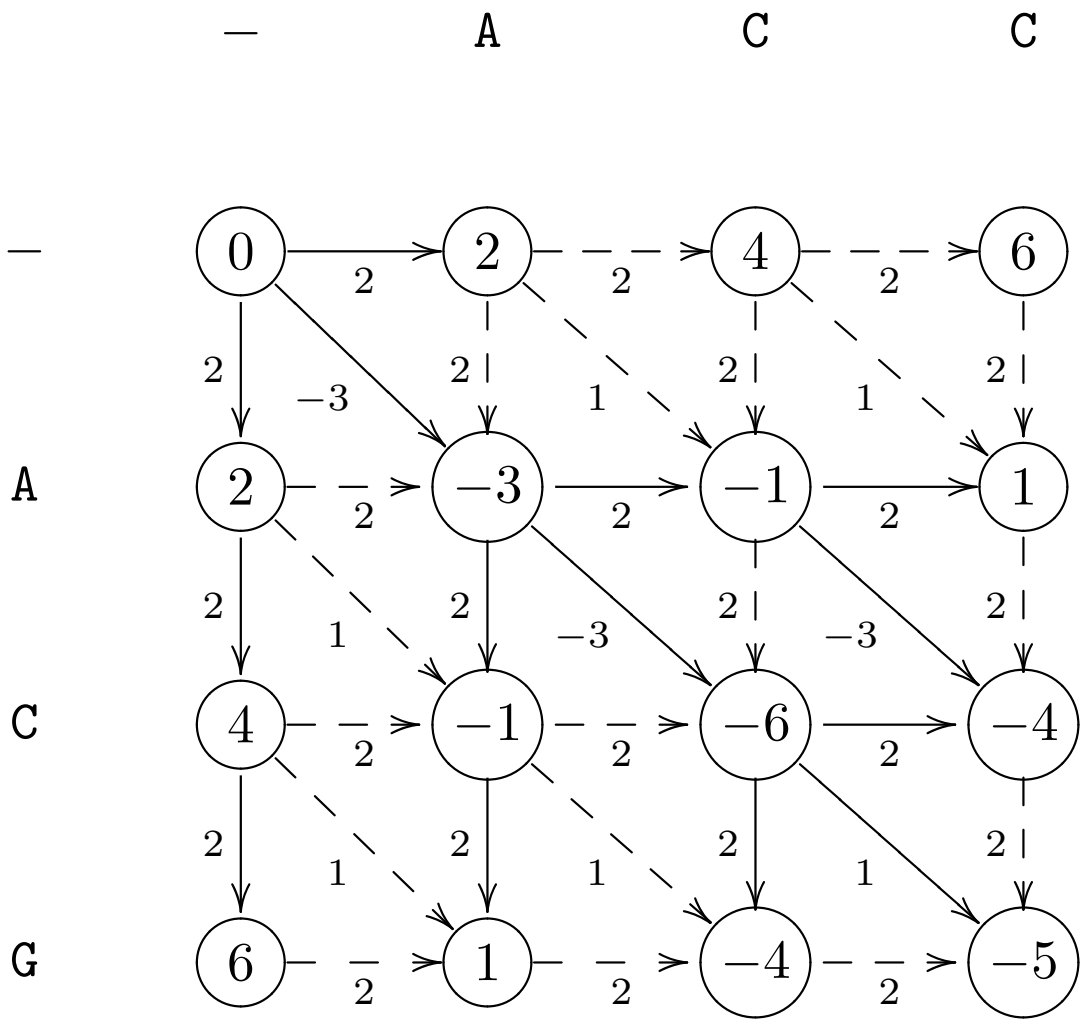
Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Needleman-Wunsch Algorithm – Example (Cont'd)

Alignment graph $G_{3,3}$: The nodes exhibit the values of the table M and the colored edges are indicated by solid lines.



Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Parametric Sequence Alignment

- Polytope algebra
- Newton polytopes
- Polytope propagation algorithm

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Parametric Sequence Alignment

- Given strings y and z of lengths m and n over Σ, respectively.
- Goal is to find all optimal alignments of y and z by varying over a large class of scoring schemes.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

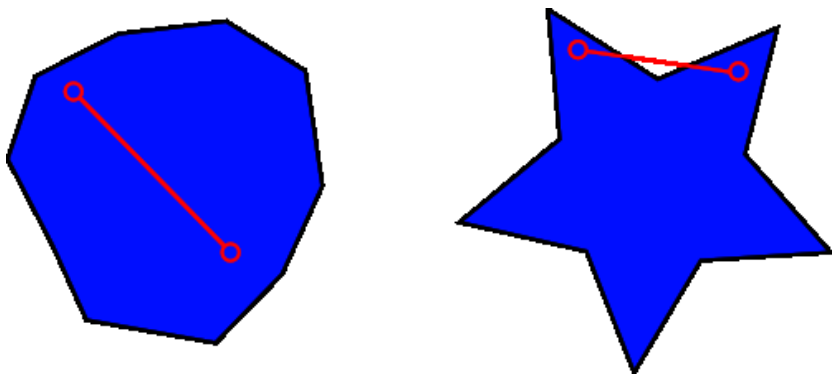
Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytopes

A subset C of $\mathbb{R}^n$ is *convex* if for each pair of distinct points a, b in C, the closed line segment with endpoints a and b is fully contained in C.



Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

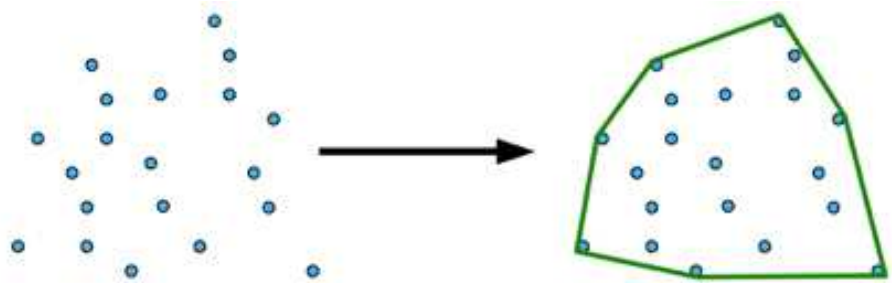
Parametric Alignment

Polytopes

- The *convex hull* of a finite set $S = \{s_1, \ldots, s_m\}$ of points in $\mathbb{R}^n$ is the set of all convex combinations of its points,

$$\operatorname{conv}(S) = \left\{ \sum_{i=1}^{m} \lambda_i s_i \mid \forall i : \lambda_i \geq 0, \sum_{i=1}^{m} \lambda_i = 1 \right\}. \qquad (163)$$

- A *bounded convex polytope* is the convex hull of a finite set of points (V-representation), or the intersection of a finite number of halfspaces (H-representation).



Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Affine Subspaces

- An *affine subspace* of $\mathbb{R}^n$ is a subset A of $\mathbb{R}^n$ such that if $s_1, \ldots, s_m \in A$, $m \geq 0$, then $\lambda_1 s_1 + \ldots + \lambda_m s_m \in A$ whenever $\lambda_1, \ldots, \lambda_m \in \mathbb{R}$ with $\sum_i \lambda_i = 1$.
- Each affine subspace A is the translate of a unique linear subspace U of $\mathbb{R}^n$, i.e.,

$$A = v + U = \{v + u \mid u \in U\} \tag{164}$$

  for some $v \in \mathbb{R}^n$.
- The *dimension* of an affine subspace A of $\mathbb{R}^n$ is the dimension of the unique linear subspace U of $\mathbb{R}^n$ with $A = v + U$, $v \in \mathbb{R}^n$.
- Example: affine lines versus lines through origin (subspaces of dimension 1).

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Affine Subspaces

- The *affine hull* of a set $S \subseteq \mathbb{R}^n$ is the smallest affine subspace containing S,

$$\{\lambda_1 s_1 + \ldots + \lambda_m s_m \mid m \geq 0, s_i \in S, \lambda_i \in \mathbb{R}, \sum_i \lambda_i = 1\} \quad (165)$$

- The *dimension* of a polytope in $\mathbb{R}^n$ is the dimension of its affine hull in $\mathbb{R}^n$, i.e., the smallest affine subspace in $\mathbb{R}^n$ containing the polytope.
- A point has dimension 0, a line segment has dimension 1, a convex polygon has dimension 2, and a convex polyhedron has dimension 3.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

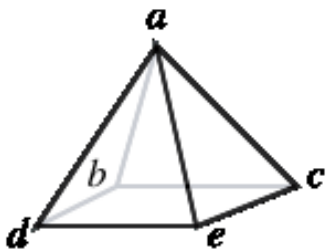
Marginal Probabilities

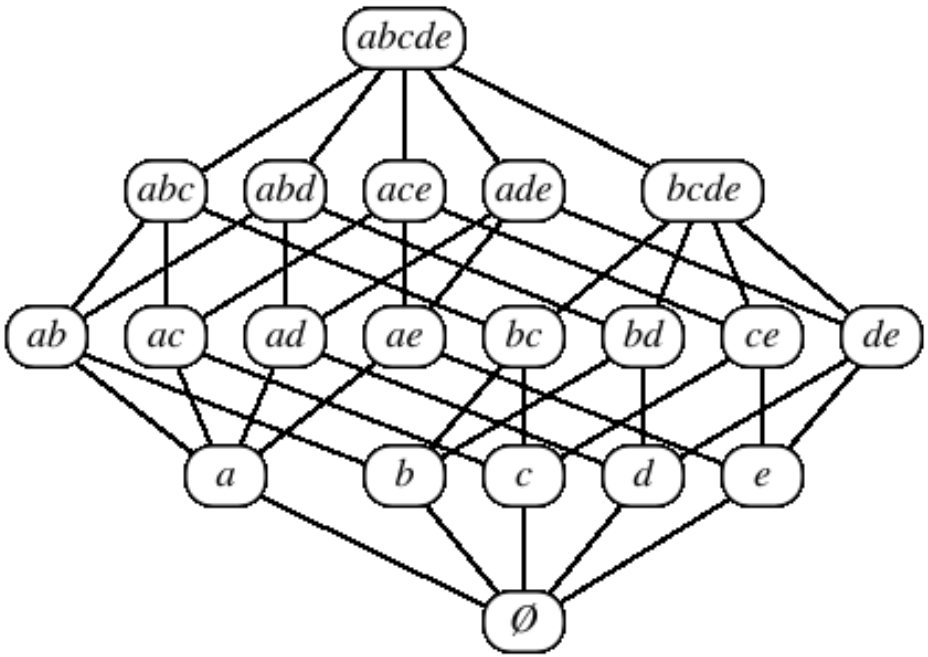
Needleman-Wunsch Algorithm

Parametric Alignment

Polytopes

- A *face* of a convex polytope is a nonempty intersection of the polytope with a halfspace such that all interior points of the polytope lie on one side of the halfspace.
- If the polytope is d-dimensional, its *facets* are its $(d-1)$-dimensional faces, its *vertices* are the 0-dimensional faces, and its *edges* are the 1-dimensional faces.





Cones

- A *cone* in $\mathbb{R}^n$ is a subset of $\mathbb{R}^n$ such that if $s_1, \ldots, s_m \in C$, $m \geq 0$, then $\lambda_1 s_1 + \ldots + \lambda_m s_m \in C$ whenever $\lambda_i \geq 0$, $1 \leq i \leq m$.
- The *positive hull* of a set $S \subseteq \mathbb{R}^n$ is the smallest cone containing S,

$$\{\lambda_1 s_1 + \ldots + \lambda_m s_m \mid m \geq 0, s_i \in S, \lambda_i \geq 0\}. \tag{166}$$

- The positive hull of a set $S \subseteq \mathbb{R}^n$ is convex.
- Examples of cones are the half-lines (rays) in $\mathbb{R}$, the quadrants in $\mathbb{R}^2$, the octants in $\mathbb{R}^3$, and the half-spaces (w.r.t. hyperplanes).

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytopes

A *fan* in $\mathbb{R}^n$ is a family $\mathcal{F} = \{C_1, C_2, \ldots, C_m\}$ of nonempty cones with the following properties:

- Each non-empty face of a cone in $\mathcal{F}$ is also a cone in $\mathcal{F}$.
- The intersection of any two cones in $\mathcal{F}$ is a face of both.

A fan $\mathcal{F}$ in $\mathbb{R}^n$ is *complete* if the union $\bigcup \mathcal{F} = C_1 \cup \ldots \cup C_m$ equals $\mathbb{R}^n$.
A fan $\mathcal{F}$ in $\mathbb{R}^n$ is *pointed* if $\{0\}$ is a cone in $\mathcal{F}$ and therefore a face of each cone in $\mathcal{F}$.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

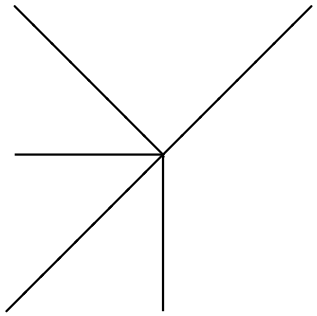
Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytopes – Example

The (pointed) fan in $\mathbb{R}^2$ has $m = 11$ cones, five of which are 2-dimensional, five (half-rays) are 1-dimensional and one is 0-dimensional (intersection point).



Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Linear Programming

Linear programming is a method to minimize a linear function over a convex set.
The canonical form of *linear program* in $\mathbb{R}^n$:

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax \geq b \\ \text{and} & x \geq 0 \end{array} \tag{167}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$ are given and x is the vector of variables.

- Objective function $\mathbb{R}^n \to \mathbb{R} : x \mapsto c^T x = \langle c, x \rangle$ is minimized over convex set $P = \{x \in \mathbb{R}^n \mid Ax \geq b, x \geq 0\}$.
- Minimal vectors $x \in \mathbb{R}^n$ form a face F of P.
- *Normal cone* $N_P(F)$ consists of all vectors $c \in \mathbb{R}^n$ such that the minimum is attained at F.
- *Inverse problem* of linear programming.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

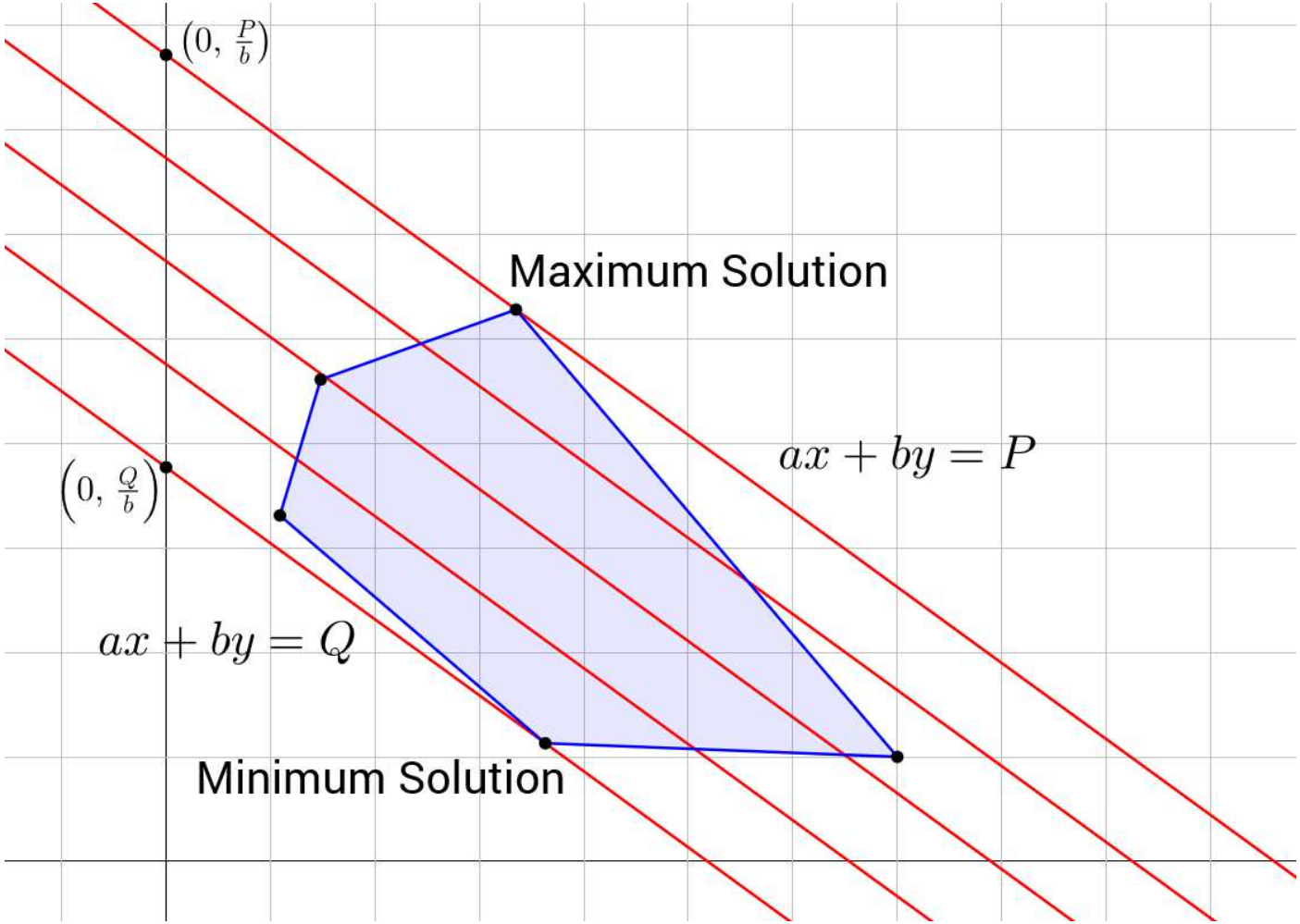
PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Linear Programming - Example



Vector c is perpendicular to the hyperplanes.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytopes

Let P be a polytope in $\mathbb{R}^n$ and let F be a face of P.

The *normal cone* $N_P(F)$ of P at F is defined such that for each $w \in \mathbb{R}^n$,

$$w \in N_P(F) \iff \begin{array}{l} F \text{ is the set of all points in } \mathbb{R}^n \\ \text{at which } P \to \mathbb{R} : x \mapsto \langle x, w \rangle \\ \text{attains the minimum.} \end{array} \tag{168}$$

In particular, if $F = \{v\}$ is a vertex of P, its normal cone $N_P(v)$ consists of all points w such that the linear maps $P \to \mathbb{R} : x \mapsto \langle x, w \rangle$ attain the minimum at the point v.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytopes

Let P be a polytope in $\mathbb{R}^n$ and let F be a face of P. The normal cone $N_P(F)$ is a cone in $\mathbb{R}^n$ with dimension

$$\dim N_P(F) = n - \dim F. \tag{169}$$

Proof.

For each $w \in \mathbb{R}^n$, let $f_w : P \to \mathbb{R} : x \mapsto \langle x, w \rangle$ be the scalar product with fixed w. Let $v, w \in \mathbb{R}^n$ such that the linear mappings f_v and f_w attain the minimum at F, and let $\lambda \geq 0$ and $\mu \geq 0$. Then the linear mapping $f_{\lambda v + \mu w}$ attains the minimum at F and so $\lambda v + \mu w$ also belongs to $N_P(F)$. Hence, $N_P(F)$ is a cone.

Let F be a k-face. Then the face F is determined by $n - k$ linearly independent linear equations and the cone $N_P(F)$ is determined by k linearly independent linear equations. Hence, the dimension formula holds. □

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytopes

The collection of all non-empty normal cones $N_P(F)$, as F runs over all faces of P, is called the *normal fan* of P denoted by $\mathcal{N}(P)$.

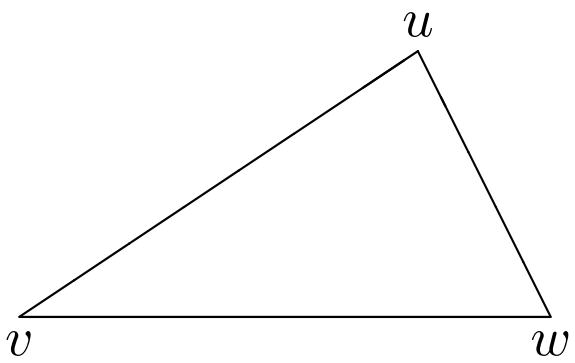
Let P be a polytope of $\mathbb{R}^n$. The normal fan $\mathcal{N}(P)$ is a complete fan of $\mathbb{R}^n$.

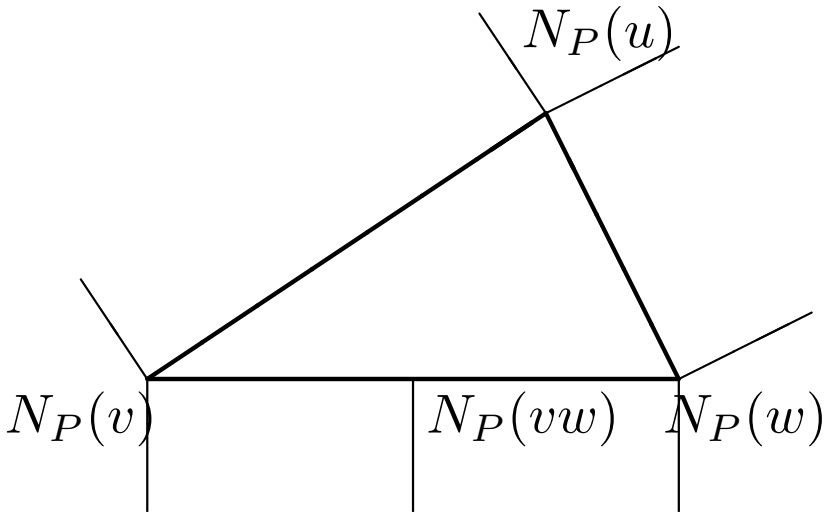
Proof.

Let $w \in \mathbb{R}^n$. If we put $F = P \cap H_{P,w}$ with intersecting hyperplane $H_{P,w}$, then $w \in N_P(F)$. It follows that the normal cones are non-empty and their union is $\mathbb{R}^n$. □

Polytopes – Example

A triangle and its normal cones:





Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

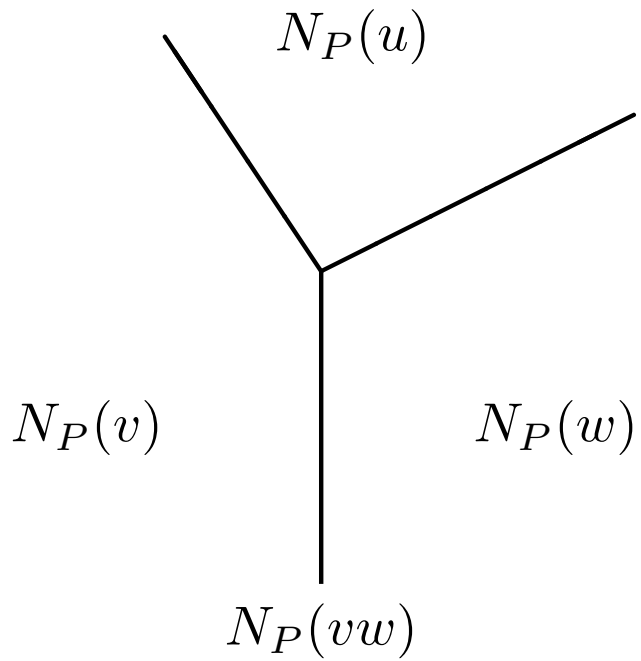
Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytopes – Example (Cont'd)

The normal fan of the above triangle:



Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytope Algebra

Let $n \geq 1$. The *polytope algebra* in $\mathbb{R}^n$ is a triple $(\mathcal{P}_n, \oplus, \odot)$ consisting of the set $\mathcal{P}_n$ of all bounded convex polytopes in $\mathbb{R}^n$ and two binary operations:

- Addition:

$$P \oplus Q = \operatorname{conv}(P \cup Q), \tag{170}$$

- Multiplication (*Minkowski sum*):

$$P \odot Q = \{p + q \mid p \in P, q \in Q\}. \tag{171}$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

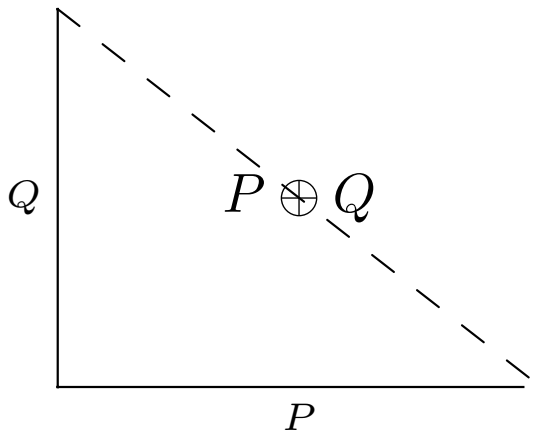
Marginal Probabilities

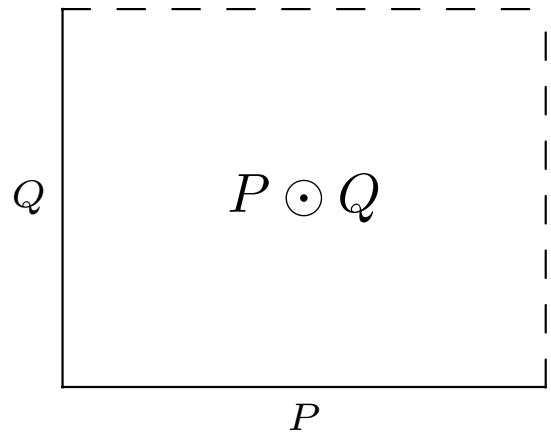
Needleman-Wunsch Algorithm

Parametric Alignment

Polytope Algebra – Example

Two non-collinear line segments P and Q in $\mathbb{R}^2$ and their sum $P \oplus Q$ and product $P \odot Q$:





Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytope Algebra

The polytope algebra $(\mathcal{P}_n, \oplus, \odot)$ on $\mathbb{R}^n$ is an idempotent commutative semiring.

Proof.

- $(\mathcal{P}_n, \oplus)$ is a commutative monoid with identity element $\emptyset$.
- $(\mathcal{P}_n, \odot)$ is a commutative monoid with identity element $\{0\}$.
- A convex set does not change if its convex hull is taken. Thus the addition of polytopes is idempotent.
- The multiplication with the empty set annihilates $\mathcal{P}_n$.
- In view of distributivity, take $p \in P$, $q \in Q$, and $r \in R$. Then for each $0 \leq \lambda \leq 1$,

$$p + (\lambda q + (1 - \lambda)r) = \lambda(p + q) + (1 - \lambda)(p + r).$$

The left-hand side is a point in $P \odot (Q \oplus R)$ and the right-hand side is a point in $(P \odot Q) \oplus (P \odot R)$. □

Polytope Algebra – Example

Consider the polytope algebra $\mathcal{P}_1$ in $\mathbb{R}^1$.

- The elements of $\mathcal{P}_1$ are the line segments

$$[a,b] = \{\lambda a + (1-\lambda)b \mid 0 \leq \lambda \leq 1\}, \quad a,b \in \mathbb{R}. \tag{172}$$

- Sum of two line segments $[a,b]$ and $[c,d]$,

$$[a,b] \oplus [c,d] = [\min\{a,c\}, \max\{b,d\}]. \tag{173}$$

- Product of two line segments $[a,b]$ and $[c,d]$,

$$[a,b] \odot [c,d] = [a+c, b+d]. \tag{174}$$

The polytope algebra in $\mathbb{R}^n$ can be viewed as a natural higher-dimensional generalization of the tropical algebra.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytope algebra

The mapping $f : \mathcal{P}_1 \to \mathbb{R} \cup \{\infty\} : [a, b] \mapsto a$ is an epimorphism from the polytope algebra $\mathcal{P}_1$ onto the tropical algebra.

Proof.

- The mapping is well-defined and we have $f(\emptyset) = \infty$ and $f(\{0\}) = f([0, 0]) = 0$.
- For any two line segments $[a, b]$ and $[c, d]$ in $\mathcal{P}_1$, we have

$$\begin{aligned} f([a, b] \oplus [c, d]) &= f([\min\{a, c\}, \max\{b, d\}]) = \min\{a, c\} \\ &= a \oplus c = f([a, b]) \oplus f([c, d]) \end{aligned}$$

and

$$\begin{aligned} f([a, b] \odot [c, d]) &= f([a + c, b + d]) = a + c \\ &= a \odot c = f([a, b]) \odot f([c, d]). \end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Newton Polytopes

Let $\mathbb{K}[X_1, \ldots, X_n]$ be the polynomial ring in the unknowns $X_1, \ldots, X_n$ over the field $\mathbb{K}$.
Each *polynomial* f in $\mathbb{K}[X_1, \ldots, X_n]$ has the form

$$f = \sum_{\alpha \in \mathbb{N}_0^n} c_\alpha X^\alpha, \tag{175}$$

where the sum is finite and each involved *term* $c_\alpha X^\alpha$ consists of the product of *coefficient* $c_\alpha \in \mathbb{K}^* = \mathbb{K} \setminus \{0\}$ and *monomial* X^α in the variables $X_1, \ldots, X_n$,

$$X^\alpha = X_1^{\alpha_1} \cdots X_n^{\alpha_n}. \tag{176}$$

The *degree* of monomial X^α is $|\alpha| = \alpha_1 + \ldots + \alpha_n$.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Newton Polytopes

The *Newton polytope* of polynomial

$$f = \sum_{\alpha \in \mathbb{N}_0^n} c_\alpha X^\alpha \tag{177}$$

is the convex polytope

$$\mathrm{NP}(f) = \mathrm{conv}(\{\alpha \in \mathbb{N}_0^n \mid c_\alpha \neq 0\}). \tag{178}$$

- The Newton polytope is generated by the exponents of the monomials involved in the polynomial measuring shape or sparsity of the polynomial.
- The actual values of the coefficients c_α do not matter in the definition of the Newton polytope.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Newton Polytopes – Example

Each polynomial in $\mathbb{K}[X, Y]$ of the form

$$f = aXY + bX^2 + cY^3 + d,$$

where $a, b, c, d \in \mathbb{K}^* = \mathbb{K} \setminus \{0\}$, has the Newton polytope

$$P = \text{conv}(\{(1, 1), (2, 0), (0, 3), (0, 0)\}),$$

since

$$XY = X_1^1 X_2^1 = X^{(1,1)}, \quad X^2 = X_1^2 X_2^0 = X^{(2,0)},$$
$$Y^3 = X_1^0 X_2^3 = X^{(0,3)}, \quad 1 = X_1^0 X_2^0 = X^{(0,0)}.$$

This is a triangle with vertices $(2, 0), (0, 3), (0, 0)$ and interior node $(1, 1)$.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Newton Polytopes

Let f and g be polynomials in $\mathbb{K}[X_1, \ldots, X_n]$. Then

$$\mathrm{NP}(f \cdot g) = \mathrm{NP}(f) \odot \mathrm{NP}(g) \tag{179}$$

and

$$\mathrm{NP}(f + g) \subseteq \mathrm{NP}(f) \oplus \mathrm{NP}(g), \tag{180}$$

where equality holds if all coefficients in the polynomials f and g are positive (i.e. terms cannot cancel).

$$(2X^2Y + 5XY) + (3XY^2 - 5XY) = 2X^2 + 3XY^2.$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Newton Polytopes – Example

In $\mathbb{R}[X, Y]$ consider the polynomials

$$f = X^p + 1 \quad \text{and} \quad g = Y^q + 1,$$

where p and q are positive integers. The corresponding Newton polytopes are line segments in $\mathbb{R}^2$ given by

$$\mathrm{NP}(f) = \mathrm{conv}(\{(0,0),(p,0)\}) = \{(x,0) \mid 0 \leq x \leq p\}$$

and

$$\mathrm{NP}(g) = \mathrm{conv}(\{(0,0),(0,q)\}) = \{(0,y) \mid 0 \leq y \leq q\}.$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Newton Polytopes – Example (Cont'd)

The sum $f + g$ has the Newton polytope

$$\begin{aligned} \mathrm{NP}(f+g) &= \mathrm{NP}(X^p + Y^q + 2) \\ &= \mathrm{conv}(\{(p,0),(0,q),(0,0)\}), \end{aligned}$$

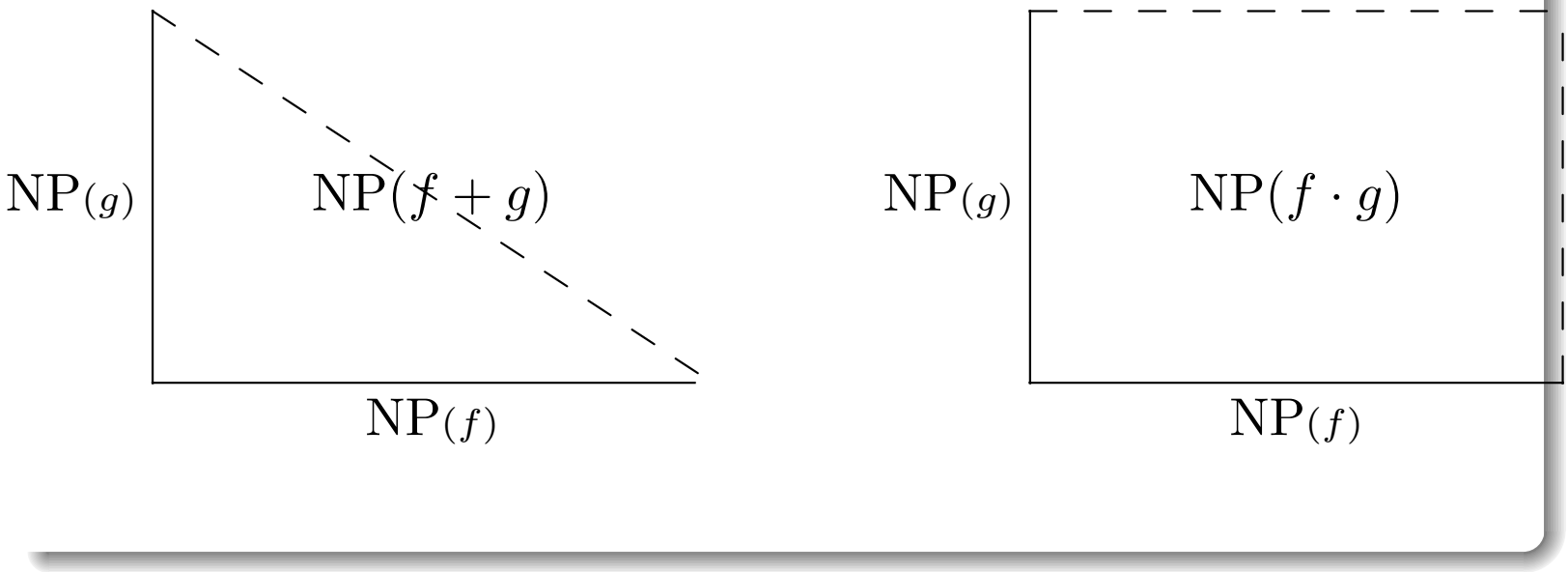
which is a triangle with vertices $(0,0)$, $(p,0)$, and $(0,q)$.

The product $f \cdot g$ has the Newton polytope

$$\begin{aligned} \mathrm{NP}(f \cdot g) &= \mathrm{NP}(X^p Y^q + X^p + Y^q + 1) \\ &= \mathrm{conv}(\{(p,q),(p,0),(0,q),(0,0)\}), \end{aligned}$$

which is a rectangle with vertices $(0,0)$, $(p,0)$, $(0,q)$, and (p,q).

Newton Polytopes – Example (Cont'd)



Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytope Propagation

- Given strings y and z of lengths m and n over alphabet Σ, resp.
- The conditional probabilities

$$\theta_{a,b}, \quad a,b \in \Sigma', \quad \text{and} \quad \theta'_{x,x'}, \quad x,x' \in \Pi, \tag{181}$$

are the unknowns of the polynomial ring

$$\mathbb{R}[\theta,\theta'] = \mathbb{R}[\{\theta_{a,b} \mid a,b \in \Sigma'\}, \{\theta'_{x,x'} \mid x,x' \in \Pi\}]. \tag{182}$$

Note that $(a,b) \neq (-,-)$.

- Each marginal probability $f(y,z)$ is then a polynomial in this ring.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytope Propagation

- Each polynomial $f(y,z)$ is assigned its Newton polytope in $\mathcal{P}_n$, where n denotes the number of unknowns in $\mathbb{R}[\theta,\theta']$; i.e.,

$$n = (|\Sigma'|^2 - 1) \cdot |\Pi|^2. \tag{183}$$

- Newton polytope $\mathrm{NP}(f(y,z))$ of the marginal probability $f(y,z)$ in the polytope algebra $\mathcal{P}_n$ is evaluated by polytopization of (136) instead of tropicalization (154):

$$\mathrm{NP}(f(y,z)) = \bigoplus_{x\in\Pi} \mathcal{P}(m,n,x). \tag{184}$$

- *Polytopization*: Replace each sum by polytope sum and each product by polytope product.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytope Propagation

$$\mathcal{P}(i,j,I) = \mathrm{NP}(\theta_{-,z_j}) \odot \bigoplus_x \mathcal{P}(i,j-1,x) \odot \mathrm{NP}(\theta'_{x,I}), \quad (185)$$

$$\mathcal{P}(i,j,D) = \mathrm{NP}(\theta_{y_i,-}) \odot \bigoplus_x \mathcal{P}(i-1,j,x) \odot \mathrm{NP}(\theta'_{x,D}), \quad (186)$$

$$\mathcal{P}(i,j,H) = \quad (187)$$

$$\mathrm{NP}(\theta_{y_i,z_j}) \odot \bigoplus_x \mathcal{P}(i-1,j-1,x) \odot \mathrm{NP}(\theta'_{x,H}),$$

and

$$\mathcal{P}(0,0,x) = \{0\}, \quad x \in \Pi, \quad (188)$$

$$\mathcal{P}(0,j,x) = \emptyset, \quad x \in \{H,D\},\ 1 \le j \le n, \quad (189)$$

$$\mathcal{P}(i,0,x) = \emptyset, \quad x \in \{H,I\},\ 1 \le i \le m, \quad (190)$$

$$\mathcal{P}(0,j,I) = \mathcal{P}(0,j-1,I) \odot \mathrm{NP}(\theta'_{I,I}) \odot \mathrm{NP}(\theta_{-,z_j}), \quad (191)$$

$$\mathcal{P}(i,0,D) = \mathcal{P}(i-1,0,D) \odot \mathrm{NP}(\theta'_{D,D}) \odot \mathrm{NP}(\theta_{y_i,-}) \quad (192)$$

*Statistical Inference

The normal cones of the polytope $\mathrm{NP}(f(y,z))$ corresponding to the vertices provide the explanations of the marginal probability $f(y,z)$.

Idea

Consider density function with parameters

$$g(\theta) = \sum_{i=1}^{M} \theta_1^{v_{i1}} \cdots \theta_d^{v_{id}}, \tag{193}$$

where $v_i = (v_{i1}, \ldots, v_{id}) \in \mathbb{N}_0^d$, $1 \leq i \leq M$.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Statistical Inference

For *fixed* value of $\theta \in \mathbb{R}^d_{>0}$, the problem is to find a term $\theta_1^{v_{j1}} \cdots \theta_d^{v_{jd}}$, $1 \leq j \leq M$, in the expression $g(\theta)$ with maximum value

$$j = \operatorname{argmax}_i \{\theta_1^{v_{i1}} \cdots \theta_d^{v_{id}}\}. \tag{194}$$

Each such solution is called an *explanation* of the model. Putting $w_i = -\log \theta_i$ and $w = (w_1, \ldots, w_d)$ gives

$$\begin{aligned} -\log(\theta_1^{v_{i1}} \cdots \theta_d^{v_{id}}) &= -[v_{i1} \log(\theta_1) + \cdots + v_{id} \log(\theta_d)] \\ &= \langle v_i, w \rangle. \end{aligned} \tag{195}$$

This amounts to finding a vector v_j that minimizes the linear expression

$$\langle v_j, w \rangle = \sum_{i=1}^{d} w_i v_{ji}, \qquad 1 \leq j \leq M. \tag{196}$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Statistical Inference

This minimization problem is equivalent to the linear programming problem

$$\begin{array}{ll} & \min\langle x, w\rangle. \\ \text{s.t.} & x \in \mathrm{NP}(g) \end{array} \tag{197}$$

Indeed, the Newton polytope $\mathrm{NP}(g)$ of the polynomial g is the convex hull of the points v_i, $1 \leq i \leq M$, and the vertices of this polytope form a subset of these points. But the minimal value of a linear functional $x \mapsto \langle x, w\rangle$ over a polytope is attained at a vertex of the polytope.

Proposition

For a fixed parameter w, the problem of solving the statistical inference problem is equivalent to the linear programming problem of minimizing the linear functional $x \mapsto \langle x, w\rangle$ over the Newton polytope $\mathrm{NP}(g)$, see (197).

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Statistical Inference

For *varying* values of $\theta \in \mathbb{R}^d$, find the set of parameters w for which the vertex v_j gives the explanation.
That is, find all points w such that the linear functional $x \mapsto \langle x, w \rangle$ attains its minimum at the point v_j.
This set is given by $N_{\mathrm{NP}(g)}(v_j)$, the normal cone of the polytope $\mathrm{NP}(g)$ at the vertex v_j.

Proposition

The set of all parameters w for which the vertex v_j provides the explanation equals the normal cone of the polytope $\mathrm{NP}(g)$ at the vertex v_j.

The normal cones associated with the vertices of the Newton polytope $\mathrm{NP}(g)$ are part of the normal fan $\mathcal{N}_{\mathrm{NP}(g)}$ of the Newton polytope $\mathrm{NP}(g)$. The normal fan provides a decomposition of the parameter space into regions, but only the regions corresponding to the vertices of the polytope are relevant for statistical inference.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytope Propagation – Example

- Simplified pair-hidden Markov model for alignment of DNA sequences with two parameters (unknowns) X and Y,

$$\begin{aligned} \theta_{a,a} &= X, \quad a \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}, \\ \theta_{a,b} &= Y, \quad a, b \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}, -\},\ a \neq b,\ (a,b) \neq (-,-), \\ \theta'_{x,x'} &= 1, \quad x, x' \in \{H, I, D\}. \end{aligned}$$

- Given DNA sequences y and z of length m and n, resp. Viewing X and Y as unknowns over $\mathbb{R}$ gives $f(y,z) \in \mathbb{R}[X,Y]$.
- Basic Newton polytopes

$$\mathcal{P}_X = \mathrm{NP}(X) = \{(1,0)\}, \quad X = X_1^1 X_2^0 = X^{(1,0)},$$

and

$$\mathcal{P}_Y = \mathrm{NP}(Y) = \{(0,1)\}, \quad Y = X_1^0 X_2^1 = X^{(0,1)}.$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytope Propagation – Example (Cont'd)

Polytope propagation algorithm for evaluation of marginal probability $f(y,z)$ in $\mathcal{P}_2$:

$$\mathrm{NP}(f(y,z)) = \mathcal{P}(m,n),$$

where

$$\begin{aligned}\mathcal{P}(i,j) &= \big(\mathcal{P}(i-1,j-1) \odot \mathrm{NP}(\theta_{y_i,z_j})\big) \oplus (\mathcal{P}(i-1,j) \odot \mathcal{P}_Y) \\ &\quad \oplus (\mathcal{P}(i,j-1) \odot \mathcal{P}_Y)\end{aligned}$$

and

$$\begin{aligned}\mathcal{P}(0,0) &= \{(0,0)\}, \\ \mathcal{P}(i,0) &= \mathcal{P}(i-1,0) \odot \mathcal{P}_Y, \\ \mathcal{P}(0,j) &= \mathcal{P}(0,j-1) \odot \mathcal{P}_Y.\end{aligned}$$

$\mathcal{P}(i,0)$ and $\mathcal{P}(0,j)$ are translates of polytopes by unit vectors, $\mathcal{P}(i,j)$ is given by the convex hull of three translates of polytopes by unit vectors, $1 \leq i \leq m$ and $1 \leq j \leq n$.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

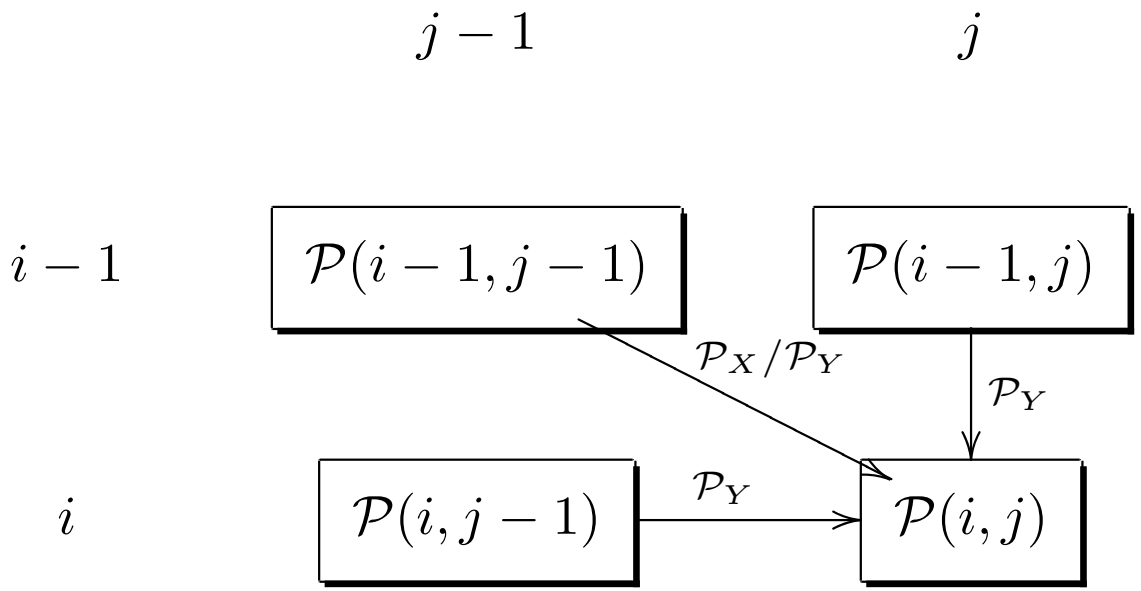
PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytope Propagation – Example (Cont'd)



$$\begin{aligned}\mathcal{P}(i,j) \quad = \quad & \left(\mathcal{P}(i-1,j-1) \odot \mathrm{NP}(\theta_{y_i,z_j})\right) \\ & \oplus \left(\mathcal{P}(i-1,j) \odot \mathcal{P}_Y\right) \\ & \oplus \left(\mathcal{P}(i,j-1) \odot \mathcal{P}_Y\right).\end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytope Propagation – Example (Cont'd)

More concretely,

$$\mathrm{NP}(f(y,z)) = \mathcal{P}(m,n),$$

where

$$\begin{aligned}\mathcal{P}(i,j) \quad = \quad & \mathrm{conv}\,[\mathcal{P}(i,j-1)+\{(0,1)\},\\ & \mathcal{P}(i-1,j-1)+\begin{cases}\{(1,0)\} & \text{if } y_i = z_j,\\ \{(0,1)\} & \text{otherwise,}\end{cases}\\ & \mathcal{P}(i-1,j)+\{(0,1)\}]\,,\end{aligned}$$

and

$$\begin{aligned}\mathcal{P}(0,0) &= \{(0,0)\},\\ \mathcal{P}(i,0) &= \mathcal{P}(i-1,0)+\{(0,1)\},\\ \mathcal{P}(0,j) &= \mathcal{P}(0,j-1)+\{(0,1)\}.\end{aligned}$$

$\ldots+\{(0,1)\}$ shift north; $\ldots+\{(1,0)\}$ shift east.

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

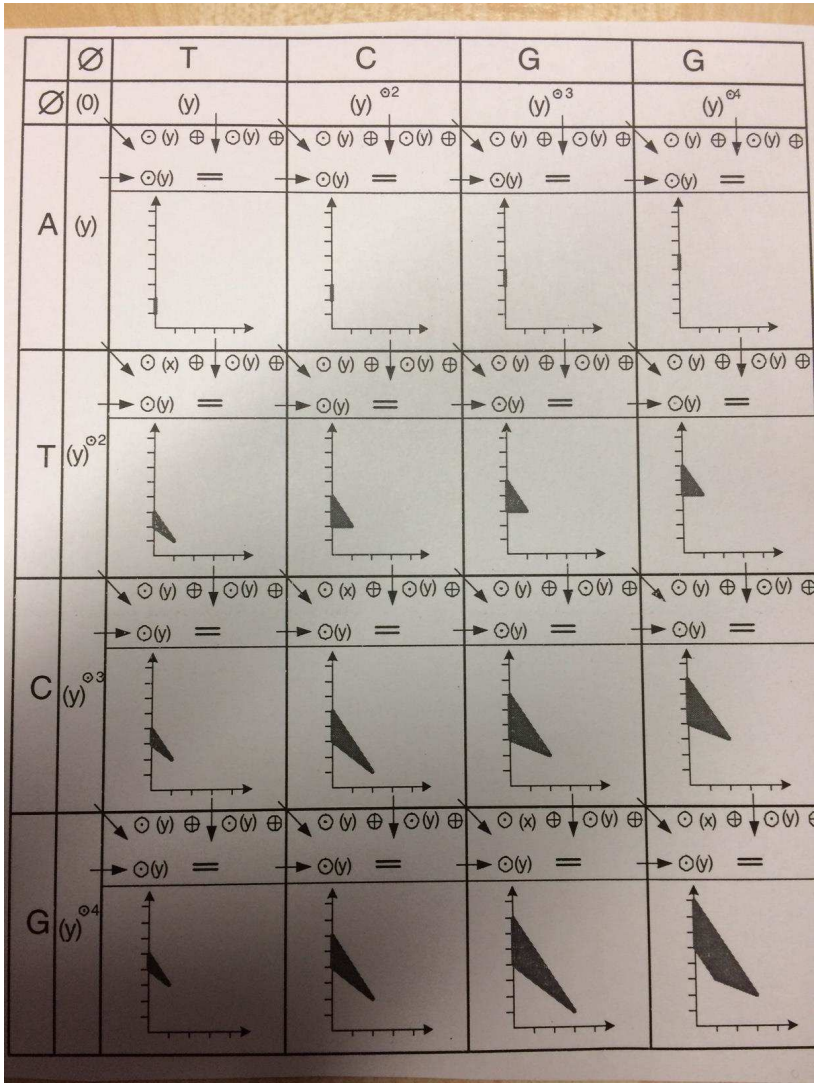
PHMM

Marginal Probabilities

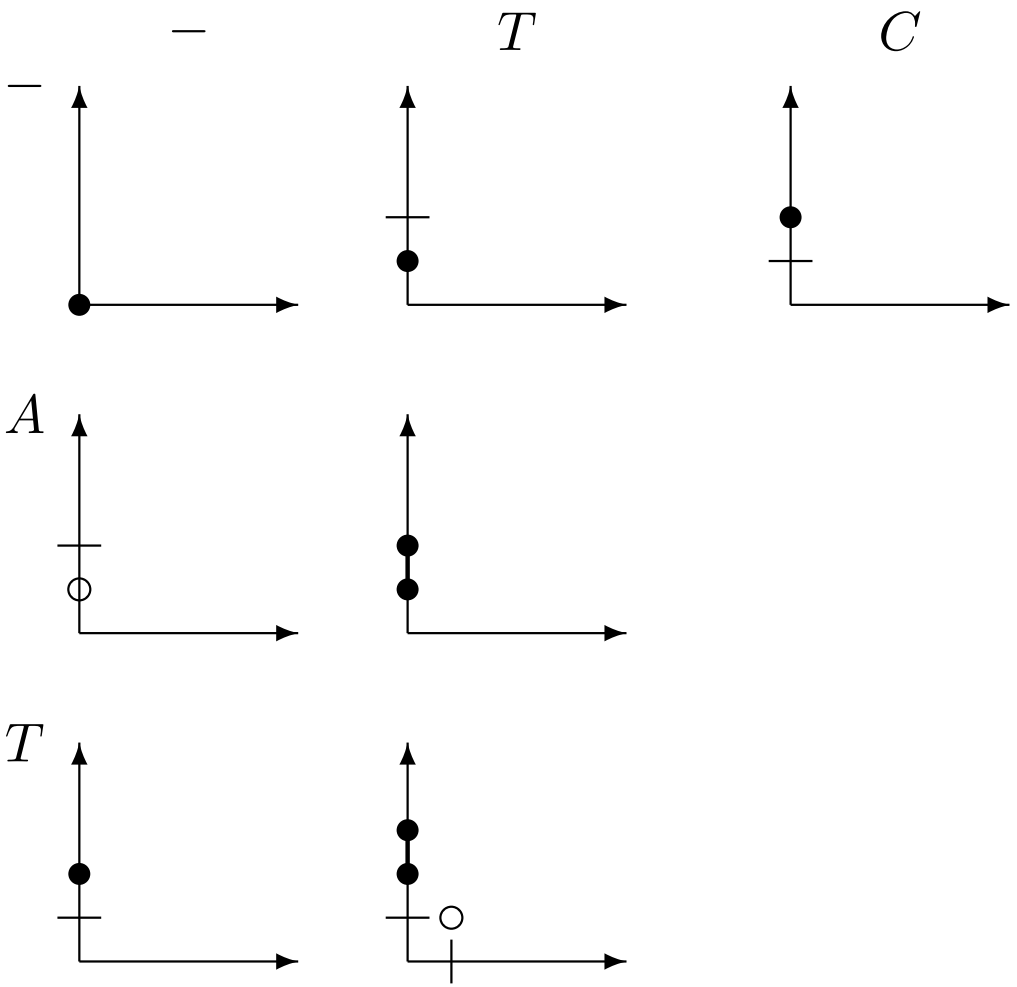
Needleman-Wunsch Algorithm

Parametric Alignment

Polytope Propagation – Example (Cont'd)



Polytope Propagation – Example (Cont'd)



Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

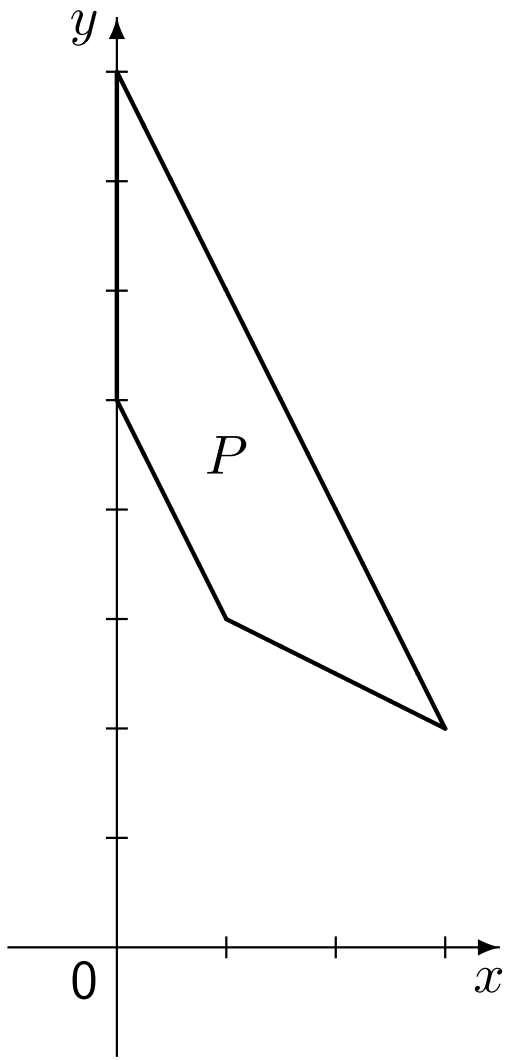
Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytope Propagation – Example (Cont'd)

Resulting polytope $P = \mathrm{NP}(f(\mathtt{ATCG}, \mathtt{TCGG}))$:



Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

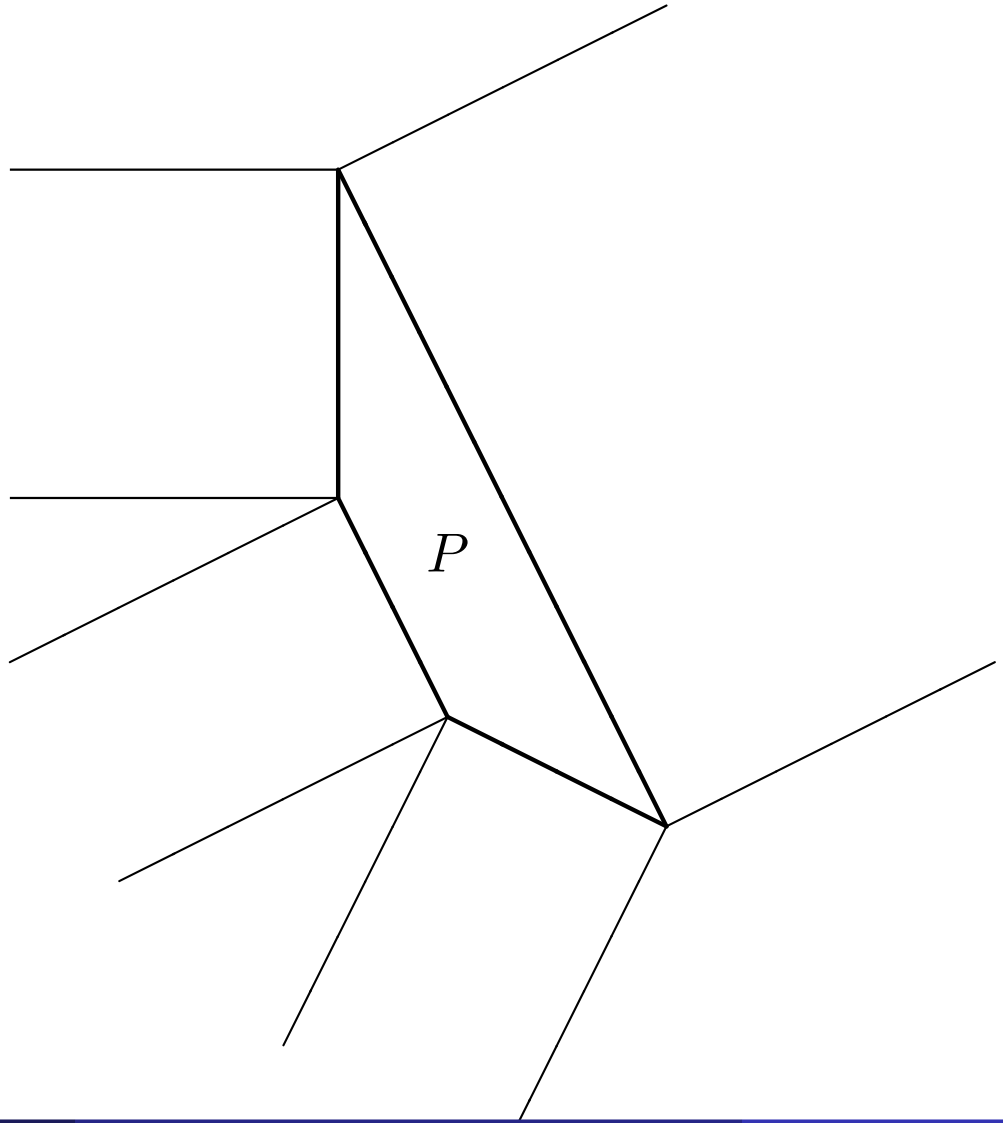
Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytope Propagation – Example (Cont'd)

The normal cones of the polytope $P = \mathrm{NP}(f(\mathtt{ATCG}, \mathtt{TCGG}))$ given by the vertices:



Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

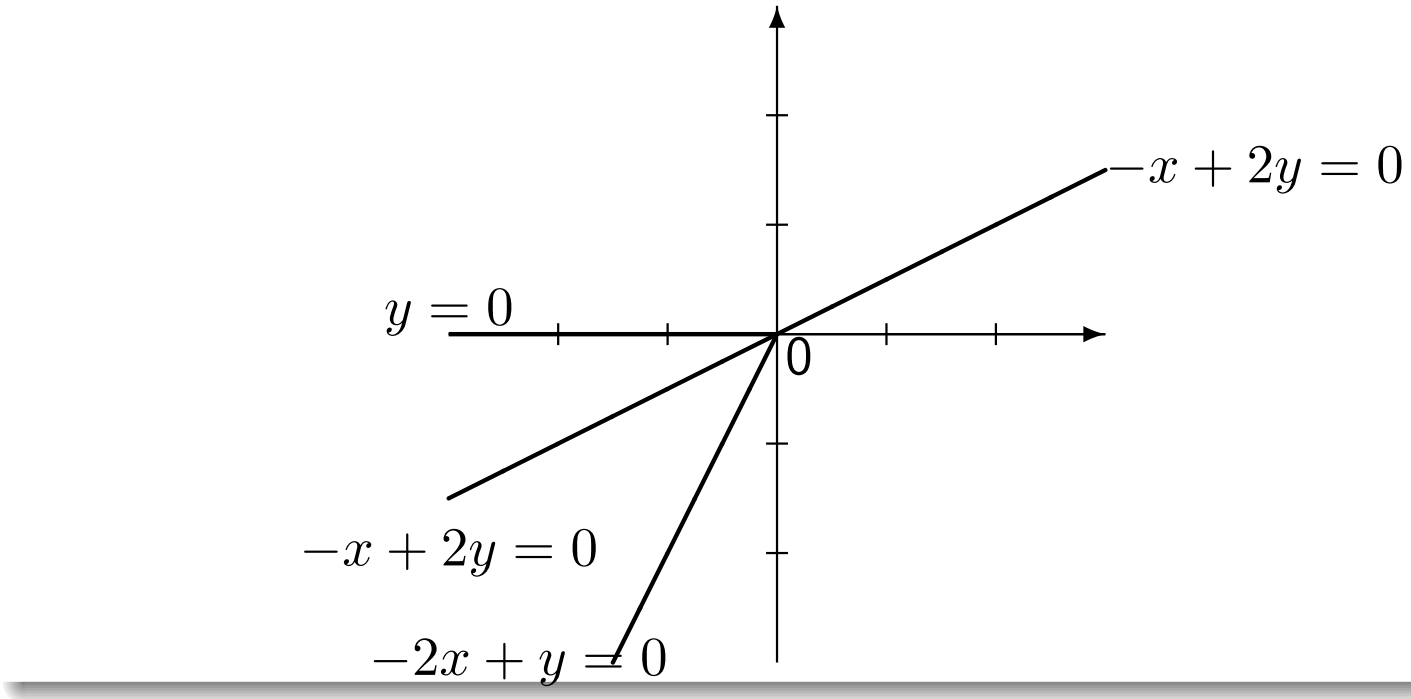
Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytope Propagation – Example (Cont'd)

Normal fan of the polytope $\mathrm{NP}(f(\mathtt{ATCG}, \mathtt{TCGG}))$:



Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Polytope Propagation – Example (Cont'd)

- The normal cones of the vertices yield the optimal sequence alignments.
- For instance, the cone of the vertex $(3, 2)$ is given by the intersection of two halfspaces defined by the inequalities $-X + 2Y < 0$ and $-2X + Y < 0$.
- Take an arbitrary point in this cone, say $(1, 0)$, and calculate an optimal alignment for the associated scoring scheme with $X = 1$ (matches) and $Y = 0$ (mismatches and indels) using the Needleman-Wunsch algorithm.

Polytope Propagation – Example (Cont'd)

The optimal alignments of the sequences ATCG and TCGG:

vertex	normal cone	scoring	alignment
$(0,8)$	$-x+2y>0$	(0,1)	A T C – G – T C G G
	$y>0$		
$(0,5)$	$-x+2y>0$	(-3,-1)	A T C – G – T C G G
	$y<0$		
$(1,3)$	$-x+2y<0$	(-2,-2)	A T C G – – – – – – – – T C G G
	$-2x+y>0$		
$(3,2)$	$-x+2y<0$	(1,0)	A T C G – – – – T C G G
	$-2x+y<0$		

Soft Computing

K.-H. Zimmermann

Contents

Representation

*PAM

PHMM

Marginal Probabilities

Needleman-Wunsch Algorithm

Parametric Alignment

Dynamic Programming

The Needleman-Wunsch and polytope propagation algorithms follow the principle of *dynamic programming* (Bellman, 1955).
A *dynamic programming algorithm* consists of

- *forward* algorithm evaluating the data, and
- *backward* algorithm providing the optimal policies.

A dynamic programming algorithm solves not only the original problem but also each subproblem.
Here the subproblems correspond to the alignments of the prefixes $y_{\leq i}$ and $z_{\leq j}$ of the given strings y and z, resp,

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Part IV

Statistical Inference and Learning

Contents

- Naïve Bayesian classification
- Data fitting
- Clustering and classification
- Approximate Bayesian computation
- Support vector machines

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier

- Training
- Classification
- Example: Iris flower data

M. Mitzenmacher, E. Upfal: Probability and Computing, Cambridge Univ. Press, 2017.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier

- Introduced in the early 1960s.
- Naïve Bayesian classifiers form a family of probabilistic classifiers based on Bayes' rule and a strong independence assumption between the features.
- Easy to implement, but may lead to misleading classification results if the features are not independent.
- Overcomes the *curse of dimensionality* (Richard E. Bellman): When the dimensionality increases, the volume of the sample space increases so fast that the available data become sparse; sparsity is a hindrance when striving for statistical significance.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier

- Given random variables $X_1, \ldots, X_n$ over finite state sets $\mathcal{X}_1, \ldots, \mathcal{X}_n$, resp.
- Random vector $X = (X_1, \ldots, X_n)$ is a feature vector over state space $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$.
- Random variable C over set of possible classifications $\mathcal{C} = \{c_1, \ldots, c_m\}$.
- Suppose there is collection $D = (d_1, \ldots, d_N)$ of N independent samples called *database*.
- Sample d_r is represented by feature vector

$$x(d_r) = (x_{r,1}, \ldots, x_{r,n}) \in \mathcal{X} \tag{198}$$

and classification label $c_r = c(d_r) \in \mathcal{C}$, $1 \leq r \leq N$.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier

Labeled sample set and new sample:

$$\begin{array}{lcccccc|c} x_1 & = & x_{11} & x_{12} & \dots & x_{1n} & c_1 \\ x_2 & = & x_{21} & x_{22} & \dots & x_{2n} & c_2 \\ & \vdots & & & & & \\ x_N & = & x_{N1} & x_{N2} & \dots & x_{Nn} & c_N \\ \hline x^* & = & x_1^* & x_2^* & \dots & x_n^* & c^* \ ? \end{array}$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier – Example

Subject classification of text documents:

- Each sample could be a text document corresponding to a web page or e-mail.
- Feature vector could describe features of the text document such as subject header and text length.
- Classification set could be a collection of labels such as spam or non-spam.
- Given a text document associated to a web page or e-mail, the task is to classify the document according to the features of the document.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier

Given database D.

- Suppose the n features are independent.
- Posterior distribution via Bayes' rule,

$$\begin{aligned} p_{C|X}(c_j \mid x^*) &= \frac{p_{X|C}(x^* \mid c_j) \cdot p_C(c_j)}{p_X(x^*)} \\ &= \frac{\prod_{i=1}^{n} p_{X_i|C}(x_i^* \mid c_j) \cdot p_C(c_j)}{p_X(x^*)}. \end{aligned} \tag{199}$$

Denominator is independent of c_j and thus can be viewed as a normalizing constant.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier

Given database D.

- Empirical marginal distribution of the label c_j,

$$p_C(c_j) = \frac{|\{r \mid c(d_r) = c_j\}|}{|D|}, \tag{200}$$

  where $|D| = N$.

- Empirical conditional distribution of i-th feature value x_i^* given c_j,

$$p_{X_i|C}(x_i^* \mid c_j) = \frac{|\{r \mid x = x(d_r), x_{ri} = x_i^*, c(d_r) = c_j\}|}{|\{r \mid c(d_r) = c_j\}|}. \tag{201}$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier - Training

Require: Database $D = (d_1, \ldots, d_N)$ of samples with feature vectors $x_r = x(d_r) \in \mathcal{X}$ and classification labels $c(d_r) \in \mathcal{C}$

Ensure: Posterior distribution

for each label $c \in \mathcal{C}$ **do**

for each feature $i \in \{1, \ldots, n\}$ **do**

for each feature value $x_i \in \mathcal{X}_i$ **do**

compute $p_{X_i|C}(x_i \mid c) = \frac{|\{r|x_r=x(d_r), x_{ri}=x_i, c(d_r)=c\}|}{|\{r|c(d_r)=c\}|}$

end for

end for

end for

for each label $c \in \mathcal{C}$ **do**

compute $p_C(c) = \frac{|\{r|c(d_r)=c\}|}{|D|}$

end for

Time complexity $O(|\mathcal{C}| \cdot |\mathcal{X}|)$ depends on sizes of $\mathcal{X} = \prod_{i=1}^{n} \mathcal{X}_i$ and $\mathcal{C}$.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier

Given new sample d^* with feature vector

$$x(d_r^*) = x^* = (x_1^*, \ldots, x_n^*). \tag{202}$$

- By (199), calculate for each label $c \in \mathcal{C}$,

$$l(c) = \left(\prod_{i=1}^{n} p_{X_i|C}(x_i^* \mid c) \right) p_C(c) \propto p_{C|X}(c \mid x^*). \tag{203}$$

- *Maximum a posteriori* (MAP) estimation: Classify sample d^* to the class with highest probability $l(c)$.
- *Empirical conditional posterior* distribution for the label c^* given sample x^*,

$$p_{C|X}(c^* \mid x^*) = \frac{\prod_{i=1}^{n} p_{X_i|C}(x_i^* \mid c^*) \cdot p_C(c^*)}{p_X(x^*)}. \tag{204}$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier - Classification

Require: Database $D = (d_1, \ldots, d_N)$ of samples with feature vectors $x_r = x(d_r) \in \mathcal{X}$ and classification labels $c(d_r) \in \mathcal{C}$, new sample d^* with feature vector $x(d^*) = x^* = (x_1^*, \ldots, x_n^*)$

Ensure: Classification label c^* with maximal conditional posterior probability

$l \leftarrow 0$

for each label $c \in \mathcal{C}$ **do**

$l' \leftarrow \prod_{i=1}^{n} p_{X_i|C}(x_i^* \mid c) \cdot p_C(c)$

if $l < l'$ **then**

$l \leftarrow l'$

$c^* \leftarrow c$

end if

end for

return c^*

In practical computations, the products calculated may lead to underflow. Therefore, it is wise to work with logarithms.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier – Example (Cont'd)

- The `R` package `naivebayes` (Michal Majka, 2018) implements the naïve Bayesian classifer.
- The library is loaded by the command

```
> library(naivebayes)
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier – Example (Cont'd)

- The *Iris flower data set* is a multivariate dataset established by the statistician Ronald Fisher (1890-1962) in 1936.
- The data set consists of 150 samples from three Iris species: *Iris setosa*, *Iris versicolor*, and *Iris virginica*.
- Four features have been measured from each sample:
  - length and width of sepals (in cm),
  - length and width of petals (in cm).
- Fisher has developed a linear discriminant model to distinguish the species from each other.
- The aim is to provide a naïve Bayesian classifier with classification labels given by the Iris species.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

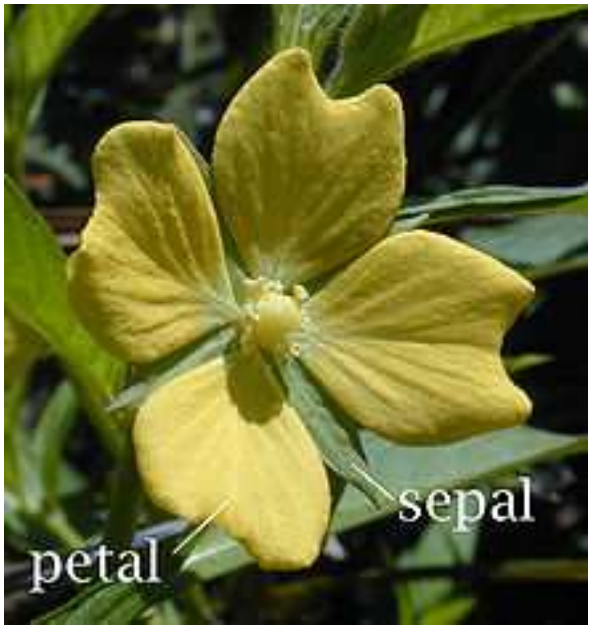
Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier – Example (Cont'd)

Iris flower



Naïve Bayesian Classifier – Example (Cont'd)

Viewing the dataset

```
> iris
```

	Sepal length	Sepal width	Petal length	Petal width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
⋮					
51	7.0	3.2	4.7	1.4	versicolor
52	6.4	3.2	4.4	1.5	versicolor
53	6.9	3.1	4.9	1.5	versicolor
54	5.5	2.3	4.0	1.3	versicolor
55	6.5	2.8	4.6	1.5	versicolor
⋮					
146	6.7	3.0	5.2	2.3	virginica
147	6.3	2.5	5.0	1.9	virginica
148	6.5	3.0	5.2	2.0	virginica
149	6.2	3.4	5.4	2.3	virginica
150	5.9	3.0	5.1	1.8	virginica

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier – Example (Cont'd)

Command `naive_bayes` fits the naïve Bayesian model with a predictor; independence among class labels is assumed.

```
> data(iris)
> nb <- naive_bayes(Species ~ ., data=iris)
> nb
```

```
A priori probabilities
   setosa  versicolor virginica
 0.3333333 0.3333333  0.3333333

Tables:

Sepal.Length      setosa versicolor virginica
          mean 5.0060000  5.9360000 6.5880000
          sd   0.3524897  0.5161711 0.6358796

Sepal.Width       setosa versicolor virginica
          mean 3.4280000  2.7700000 2.9740000
          sd   0.3790644  0.3137983 0.3224966

Petal.Length      setosa versicolor virginica
          mean 1.4620000  4.2600000 5.5520000
          sd   0.1736640  0.4699110 0.5518947

Petal.Width       setosa versicolor virginica
          mean 0.2460000  1.3260000 2.0260000
          sd   0.1053856  0.1977527 0.2746501
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier – Example (Cont'd)

Plotting of the empirical distributions of the individual features with respect to the classification labels:

```
> plot( nb, "Sepal.Length")
> plot( nb, "Sepal.Width")
> plot( nb, "Petal.Length")
> plot( nb, "Petal.Width")
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

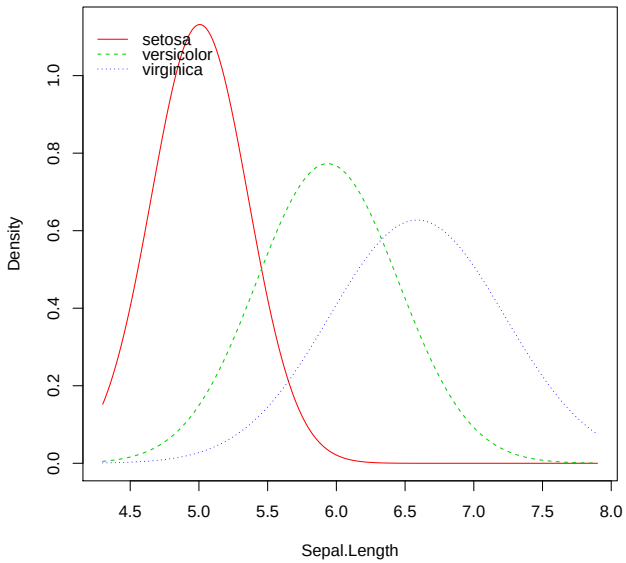
Clustering and Classification

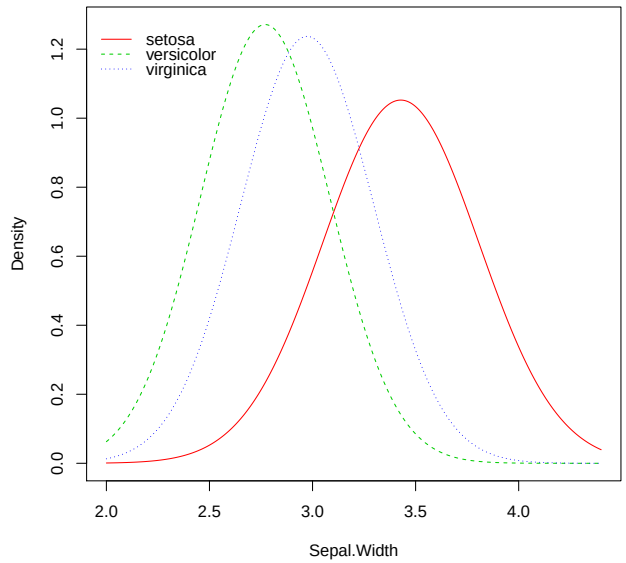
ABC Computation

Support Vector Machines

Naïve Bayesian Classifier – Example (Cont'd)

Individual empirical distributions of features conditioned by class labels:





Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

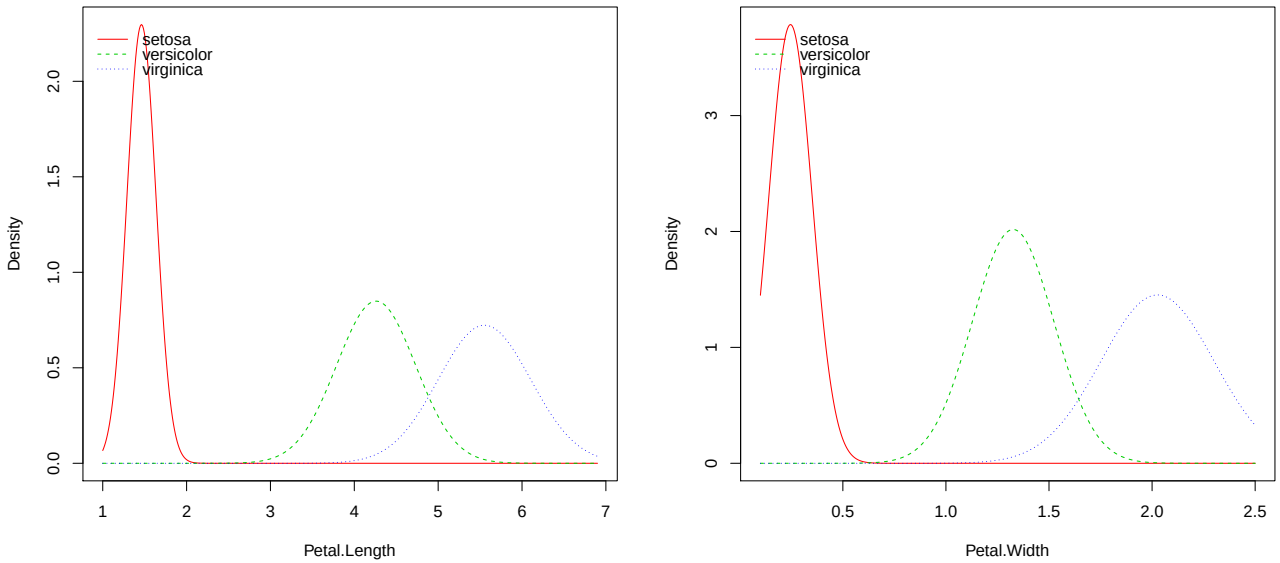
Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier – Example (Cont'd)

Individual empirical distributions of features conditioned by class labels:



Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier – Example (Cont'd)

For testing purposes, the data set is divided into a training set and a test set.

```
> ind_iris <- sample(1:nrow(iris),
                     size=round(0.3*nrow(iris)))
> ind_iris
 [1] 61 112 34 43 82 84 138 126 47 97
[11] 86 19 134 57 119 26 44 111 62 140
[21] 11 128 28 108 106 54 144 4 124 5
[31] 113 72 25 90 150 120 52 139 94 146
[41] 88 49 137 75 96
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier – Example (Cont'd)

The training and test sets make up 70% and 30% of the data set, respectively.

```
> iris_train <- iris[-ind_iris, ]
```

```
> iris_train
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2 setosa
2          4.9         3.0          1.4         0.2 setosa
3          4.7         3.2          1.3         0.2 setosa
6          5.4         3.9          1.7         0.4 setosa
...
149        6.2         3.4          5.4         3.2 virginica
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier – Example (Cont'd)

The training and test sets make up 70% and 30% of the data set, respectively.

```
> iris_test <- iris[ind_iris, ]
```

```
> iris_test
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
61          5.0         2.0          3.5         1.0 versicolor
112         6.4         2.7          5.3         1.9 viriginica
34          5.5         4.2          1.4         0.2 setosa
43          4.4         3.2          1.3         0.2 setosa
...
96          5.7         3.0          4.2         1.2 versicolor
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier – Example (Cont'd)

Command `predict` provides the classification; it returns for each test sample the label with maximal conditional posterior probability.

```
> nb_iris <- naive_bayes(Species ~ ., iris_train)
> predict(nb_iris, iris_test)
```

```
 [1] versicolor virginica  setosa     setosa     versicolor versicolor
 [7] virginica  virginica  setosa     versicolor versicolor setosa
[13] versicolor versicolor virginica  virginica  setosa     virginica
[19] versicolor virginica  setosa     virginica  setosa     virginica
[25] virginica  versicolor virginica  setosa     virginica  setosa
[31] virginica  versicolor setosa     versicolor virginica  versicolor
[37] versicolor virginica  versicolor versicolor virginica versicolor
[43] virginica  versicolor versicolor
Levels. setosa versicolor viriginica
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Naïve Bayesian Classifier – Example (Cont'd)

```
> head(predict(nb-iris, iris_test, type="prob"))

             setosa   versicolor   viriginica
[1,]  3.169305e-41 9.999992e-01 7.682960e-07
[2,] 2.918437e-167 6.036875e-03 9.939631e-01
[3,]  1.000000e+00 8.072905e-22 1.919866e-27
[4,]  1.000000e+00 5.674804e-19 7.638931e-28
[5,]  6.620915e-49 9.999991e-01 8.523488e-07
[6,] 4.548898e-136 7.641880e-01 2.3581120-01
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Least Squares Data Fitting

- Least squares method (C.F. Gauss)
- Data fitting
- Example: Iris flower data

S. Boyd, L. Vandenberghe: Introduction to Applied Linear Algebra, Cambridge Univ. Press, 2018.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Least Squares Method

- Given tall real-valued $m \times n$ matrix A; i.e., $m > n$.
- System of linear equations

$$Ax = b \tag{205}$$

with $b \in \mathbb{R}^m$ is over-determined; m equations with n variables where $m > n$.

- Solution of (205) exists if b is a linear combination of the columns of A; i.e.,

$$\mathrm{rk}(A) = \mathrm{rk}(A|b). \tag{206}$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Least Squares Method

- If there is no solution of (205), a vector x is searched for which the *residual*

$$r = Ax - b \tag{207}$$

is minimized,

$$\|Ax - b\|^2 = \|r\|^2 = r_1^2 + \ldots + r_m^2. \tag{208}$$

- The problem of finding a vector $\hat{x} \in \mathbb{R}^n$ which minimizes (208) is the problem of *least squares* or *regression*.
- Each vector $\hat{x}$ which satisfies

$$\|A\hat{x} - b\|^2 \leq \|Ax - b\|^2 \tag{209}$$

for all $x \in \mathbb{R}^n$ is called *least squares approximate solution* of $Ax = b$.

- If the *optimal residual norm* $\|A\hat{x} - b\|$ is small, then $\hat{x}$ *approximately* solves $Ax = b$.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Least Squares Method

Suppose the matrix A has linearly independent columns. Then the solution of the least squares problem $Ax = b$ is given by

$$\hat{x} = (A^t A)^{-1} A^t b. \tag{210}$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

*Proof.

Suppose the columns of A are linearly independent.

- Claim that the *Gram matrix* A^tA is invertible. Indeed, let $x \in \mathbb{R}^n$ with $(A^tA)x = 0$. Then

$$0 = x^t 0 = x^t(A^tA)x = x^tA^tAx = (Ax)^t(Ax) = \|Ax\|^2$$

  and so by the norm $Ax = 0$. Since the columns of A are linearly independent, it follows that $x = 0$.

- Claim that for each vector $x \in \mathbb{R}^n$ with $x \neq \hat{x}$,

$$\|A\hat{x} - b\| \leq \|Ax - b\|.$$

  Indeed, write

$$\begin{aligned} \|Ax - b\|^2 &= \|(Ax - A\hat{x}) - (A\hat{x} - b)\|^2 \\ &= \|Ax - A\hat{x}\|^2 + \|A\hat{x} - b\|^2 + 2(Ax - A\hat{x})^t(A\hat{x} - b), \end{aligned}$$

  by using the identity

$$\|a + b\|^2 = (a + b)^t(a + b) = \|a\|^2 + \|b\|^2 + 2a^tb.$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Proof (Cont'd)

- Moreover,

$$\begin{aligned}(Ax - A\hat{x})^t(A\hat{x} - b) &= (x - \hat{x})^t A^t(A\hat{x} - b) \\ &= (x - \hat{x})^t(A^t A\hat{x} - A^t b) \\ &= (x - \hat{x})^t 0 = 0,\end{aligned}$$

by using $(A^t A)\hat{x} = A^t b$. Then

$$\|Ax - b\|^2 = \|Ax - A\hat{x}\|^2 + \|A\hat{x} - b\|^2.$$

Since the first term on the right-hand side is nonnegative, we obtain

$$\|Ax - b\|^2 \geq \|A\hat{x} - b\|^2.$$

Thus $\hat{x}$ is a minimizer of $\|Ax - b\|$.

- Claim that $\hat{x}$ is the unique minimizer of $\|Ax - b\|$. Indeed, suppose that $\|Ax - b\|^2 = \|A\hat{x} - b\|^2$. Then it follows that $\|A(x - \hat{x})\|^2 = 0$ which by the norm implies that $A(x - \hat{x}) = 0$. But the columns of A are linearly independent and so we arrive at $x = \hat{x}$, as required. □

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Example (Maple)

Consider the least squares problem $Ax = b$, where

$$A = \begin{pmatrix} 2 & 0 \\ -1 & 1 \\ 0 & 2 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}.$$

This over-determined systems has three linear equations in two variables,

$$2x_1 = 1, \quad -x_1 + x_2 = 0, \quad 2x_2 = -1,$$

and has no solution. The associated least squares problem is

$$\min\{(2x_1 - 1)^2 + (-x_1 + x_2)^2 + (2x_2 + 1)^2\},$$

which has the unique solution

$$\hat{x} = \begin{pmatrix} 1/3 \\ -1/3 \end{pmatrix}.$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Example (Maple) Cont'd

This solution can be verified by the Maple code

```
> with(LinearAlgebra):
> A := Matrix([[2,0],[-1,1],[0,2]]):
> b := Vector([1,0,-1]):
> x := LeastSquares(A, b):
> r := A.x-b:
```

Note that the least square approximate solution $\hat{x}$ does not satisfy $Ax = b$ and the associated residual is

$$\hat{r} = A\hat{x} - b = \left(-\frac{1}{3}, -\frac{2}{3}, \frac{1}{3}\right)^t$$

with optimal residual norm

$$\|\hat{r}\|^2 = \frac{2}{3}.$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Least Squares Solution

- Gram-Schmidt orthogonalization
- Pseudo-inverse
- QR factorization

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Gram-Schmidt

- Given a collection of vectors $a_1, \ldots, a_n \in \mathbb{R}^m$.
- Construct a collection of vectors $q_1, \ldots, q_n \in \mathbb{R}^m$ with the following properties:
  - For each $1 \leq i \leq n$, a_i is a linear combination of $q_1, \ldots, q_i$ and q_i is a linear combination of $a_1, \ldots, a_i$.
  - If the vectors $a_1, \ldots, a_{i-1}$ are linearly independent, but $a_1, \ldots, a_i$ are linearly dependent, the algorithm detects this and stops; i.e., the algorithm finds the first vector a_i which is a linear combination of the previous vectors $a_1, \ldots, a_{i-1}$.
  - If the vectors $a_1, \ldots, a_n$ are linearly independent, the algorithm produces an orthonormal collection of vectors $q_1, \ldots, q_n$.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Gram-Schmidt Algorithm

Require: Vectors $a_1, \ldots, a_n \in \mathbb{R}^m$
Ensure: Orthonormal system of vectors $q_1, \ldots, q_n \in \mathbb{R}^m$ or abortion.
for i from 1 to n **do**
  $\tilde{q}_i \leftarrow a_i - (q_1^t a_i)q_1 - \ldots - (q_{i-1}^t a_i)q_{i-1}$
  if $\tilde{q}_i = 0$ **then**
    return Abortion
  end if
  $q_i \leftarrow \tilde{q}_i / \|\tilde{q}_i\|$
end for

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Gram-Schmidt

Suppose the vectors $a_1, \ldots, a_n \in \mathbb{R}^m$ be linearly independent.

- Let A be the $m \times n$ matrix with columns $a_1, \ldots, a_n$.
- Let Q be the $m \times n$ matrix with orthonormal columns $q_1, \ldots, q_n \in \mathbb{R}^m$.
- Orthonormality gives the matrix equation

$$Q^t Q = I_n \tag{211}$$

in which the i, j entry is the scalar product $q_i^t q_j = \delta_{ij}$, where δ is the Kronecker delta.

- Relation between a_i and q_i in Gram-Schmidt algorithm,

$$a_i = (q_1^t a_i) q_1 + \ldots + (q_{i-1}^t a_i) q_{i-1} + \|\tilde{q}_i\| q_i, \tag{212}$$

$1 \leq i \leq n$.

Gram-Schmidt

- Put

$$a_i = R_{1i}q_1 + \ldots + R_{ii}q_i, \quad 1 \leq i \leq n, \tag{213}$$

where $R_{ij} = q_i^t a_j$ for $i < j$ and $R_{ii} = \|\tilde{q}_i\|$. Define $R_{ij} = 0$ for $i > j$.

- Then (213) becomes in matrix form

$$A = QR, \tag{214}$$

which is the *QR factorization* of A.

- The $m \times n$ matrix Q has orthonormal columns and the $n \times n$ matrix R is upper triangular with positive diagonal entries.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Pseudo Inverse

Suppose the matrix A has linearly independent columns.

- The matrix A^tA is invertible.

  Indeed, let $x \in \mathbb{R}^n$ with $(A^tA)x = 0$. Then

  $$0 = x^t 0 = x^t(A^tA)x = x^tA^tAx = (Ax)^t(Ax) = \|Ax\|^2$$

  and so by the norm, $Ax = 0$. But the columns of A are linearly independent and hence $x = 0$.

- The matrix

  $$A^\dagger = (A^tA)^{-1}A^t \tag{215}$$

  is called the *pseudo-inverse* of A.

- The pseudo-inverse of A is a left inverse of A, since

  $$A^\dagger A = ((A^tA)^{-1}A^t)A = (A^tA)^{-1}(A^tA) = I. \tag{216}$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Pseudo Inverse

- The pseudo-inverse of A can be calculated using the QR factorization of A, since

$$A^t A = (QR)^t (QR) = R^t Q^t Q R = R^t R \tag{217}$$

and so

$$\begin{aligned} A^\dagger &= (A^t A)^{-1} A^t \\ &= (R^t R)^{-1} (QR)^t \\ &= R^{-1} R^{-t} R^t Q^t \\ &= R^{-1} Q^t. \end{aligned} \tag{218}$$

Note that for an invertible matrix A,

$$(A^t)^{-1} = (A^{-1})^t, \tag{219}$$

which is also represented by A^{-t}.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

QR Factorization

Suppose the matrix A has linearly independent columns.

- Take the QR factorization $A = QR$.
- The pseudo-inverse $A^\dagger = R^{-1}Q^t$ of A satisfies by (210),

$$\hat{x} = (A^t A)^{-1} A^t b = A^\dagger b = R^{-1} Q^t b \tag{220}$$

and so

$$R\hat{x} = Q^t b. \tag{221}$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Least Squares Algorithm via QR Factorization

Require: Tall real-valued $m \times n$ matrix A with independent columns and vector $b \in \mathbb{R}^m$

Ensure: Least squares approximate solution $\hat{x}$

Compute QR factorization of A

Compute $Q^t b$

Solve the triangular system of equations $R\hat{x} = Q^t b$ via back substitution

Correctness

Follows from (221).

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Example (Maple)

Consider the matrix

$$A = \begin{pmatrix} 2 & 0 \\ -1 & 1 \\ 0 & 2 \end{pmatrix}.$$

QR factorization of A,

```
> with(LinearAlgebra):
> A := Matrix([[2,0],[-1,1],[0,2]]):
> Q,R := QRDecomposition(A):
```

where

$$Q = \begin{pmatrix} \frac{2}{5}\sqrt{5} & \frac{1}{30}\sqrt{30} \\ -\frac{1}{5}\sqrt{5} & \frac{1}{15}\sqrt{30} \\ 0 & \frac{1}{6}\sqrt{30} \end{pmatrix}, \quad R = \begin{pmatrix} \sqrt{5} & -\frac{1}{5}\sqrt{5} \\ 0 & \frac{2}{5}\sqrt{30} \end{pmatrix}.$$

Example (Maple) Cont'd

Pseudo-inverse of A,

```
> B := MatrixInverse(A);
```

where

$$B = \begin{pmatrix} \frac{5}{12} & -\frac{1}{6} & \frac{1}{12} \\ \frac{1}{12} & \frac{1}{6} & \frac{5}{12} \end{pmatrix}.$$

Approximate solution of the above least squares problem,

$$\hat{x} = Bb = \begin{pmatrix} 1/3 \\ -1/3 \end{pmatrix}.$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Least Squares Data Fitting

- Given sample data

$$x^{(1)}, \ldots, x^{(N)} \in \mathbb{R}^n$$

and

$$y^{(1)}, \ldots, y^{(N)} \in \mathbb{R},$$

where $(x^{(i)}, y^{(i)})$ denotes the ith sample pair.

- Form a *model* of the relationship between the sample pairs

$$\hat{f}(x) \approx y, \tag{222}$$

where $\hat{f} : \mathbb{R}^n \to \mathbb{R}$ is called the *predictor*.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Least Squares Data Fitting

- Suppose the model function has the form

$$\hat{f}(x) = \theta_1 f_1(x) + \ldots + \theta_p f_p(x), \tag{223}$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$ are *basis functions* or *feature mappings* and $\theta_1, \ldots, \theta_p$ are real-valued *model parameters*. This predictor is linear in the parameters.

- When the basis functions are given, the task is to choose the model parameters accordingly.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Least Squares Data Fitting

- *Prediction error* for sample pair $(x^{(i)}, y^{(i)})$,

$$r^{(i)} = y^{(i)} - \hat{f}(x^{(i)}) = y^{(i)} - \hat{y}^{(i)}, \quad 1 \leq i \leq N. \tag{224}$$

- Vector notation

$$\begin{aligned} y &= (y^{(1)}, \ldots, y^{(N)}), && (225)\\ \hat{y} &= (\hat{y}^{(1)}, \ldots, \hat{y}^{(N)}), && (226)\\ r &= (r^{(1)}, \ldots, r^{(N)}). && (227) \end{aligned}$$

Residual can be written as

$$r = y - \hat{y}. \tag{228}$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Least Squares Data Fitting

- A common measure how well the model predicts the observed sample data is the RMS prediction error $\mathrm{rms}(r)$ of the residual r.
- The *root-mean-square* (RMS) value of a vector $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is

$$\mathrm{rms}(x) = \sqrt{\frac{x_1^2 + \ldots + x_n^2}{n}} = \frac{\|x\|}{\sqrt{n}}. \tag{229}$$

- The minimization of $\mathrm{rms}(r)$ w.r.t. the model parameters $\theta_1, \ldots, \theta_p$ given the sample set is a least squares problem.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Least Squares Data Fitting

- Write $\hat{y}^{(i)} = \hat{f}(x^{(i)})$ in terms of the model parameters,

$$\hat{y}^{(i)} = a_{i1}\theta_1 + \ldots + a_{ip}\theta_p, \quad 1 \leq i \leq N, \tag{230}$$

where $A = (a_{ij})$ denotes the $N \times p$ matrix with

$$a_{ij} = \hat{f}_j(x^{(i)}), \quad 1 \leq i \leq N, \quad 1 \leq j \leq p. \tag{231}$$

- The jth column of A is given by the jth basis function $\hat{f}_j$ evaluated at the data $x^{(1)}, \ldots, x^{(N)}$.
- The ith row of A consists of the values of p basis functions at the data point $x^{(i)}$.
- Writing $\theta = (\theta_1, \ldots, \theta_p)^t$, we obtain in matrix form

$$\hat{y} = A\theta, \tag{232}$$

where A is a tall $N \times p$ matrix.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Least Squares Data Fitting

Aim is to find the model parameter vector θ.

- Sum of the squares of the residuals is given by

$$\|r\|^2 = \|y - \hat{y}\|^2 = \|y - A\theta\|^2. \tag{233}$$

- Suppose the columns of the matrix A are linearly independent.
- Then the least squares problem provides the best model parameter vector $\hat{\theta}$,

$$\hat{\theta} = (A^t A)^{-1} A^t y = A^\dagger y. \tag{234}$$

  The solution $\hat{\theta}$ is the *least squares fitting* on the sample set.
- The number $\|y - A\hat{\theta}\|^2$ is the *minimum sum square error*.
- The number $\|y - A\hat{\theta}\|^2/N$ is the *minimum mean square error* (MMSE) of the model.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Polynomial Fit

Simple model of *polynomial fit* ($n = 1$) makes use of the univariate polynomial functions

$$f_1(x) = 1,\ f_2(x) = x, \ldots,\ f_p(x) = x^{p-1}. \tag{235}$$

Model function (223) has then the form

$$\hat{f}(x) = \theta_1 + \theta_2 x + \ldots + \theta_p x^{p-1}. \tag{236}$$

Matrix A in (231) has thus the shape

$$A = \begin{pmatrix} 1 & x^{(1)} & \ldots & (x^{(1)})^{p-1} \\ 1 & x^{(2)} & \ldots & (x^{(2)})^{p-1} \\ \vdots & & & \vdots \\ 1 & x^{(N)} & \ldots & (x^{(N)})^{p-1} \end{pmatrix}. \tag{237}$$

This is a Vandermonde matrix: Columns of A are linearly independent if the numbers $x^{(1)}, \ldots, x^{(N)}$ include at least p distinct values.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Vandermonde Matrix

The real-valued matrix

$$A = \begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^{n-1} \end{pmatrix} \tag{238}$$

has the determinant

$$\det(A) = \prod_{1 \leq i < j \leq n} (x_j - x_i). \tag{239}$$

The matrix A is a Vandermonde matrix.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

*Proof.

- All entries in the ith column have total degree $i-1$.
- By the Leibniz formula, all terms in the determinant have total degree

$$0+1+2+\ldots+(n-1)=\frac{n(n-1)}{2}.$$

- For $i \neq j$ substitute x_i for x_j. The resulting matrix has two equal columns and so zero determinant.
- By the factor theorem, $x_j - x_i$ is a divisor of $\det(A)$.
- By the unique factorization of multivariate polynomials,

$$\det(A) = q \prod_{1 \leq i < j \leq n} (x_j - x_i)$$

where q is a polynomial.

Proof (Cont'd).

- By comparing degrees, the polynomial q is a constant.
- The term corresponding to the diagonal entries of A is

$$1x_2x_3^2 \cdots x_n^{n-1},$$

which is also the monomial by taking the first terms in all the factors in $\prod_{1 \leq i < j \leq n}(x_j - x_i)$. Thus the constant q is 1.

□

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Regression

- The regression model has the shape

$$\hat{y} = x^t \beta + v \in \mathbb{R}, \tag{240}$$

where $x \in \mathbb{R}^n$, $\beta \in \mathbb{R}^n$ is the *weight vector* and $v \in \mathbb{R}$ is the *offset*.

- Write the regression model as

$$\hat{y} = x^t \theta' + \theta_1, \tag{241}$$

where the basis functions are

$$f_1(x) = 1, \quad f_i(x) = x_{i-1}, \quad 2 \leq i \leq n+1, \tag{242}$$

with $v = \theta_1$, $\beta = \theta' = (\theta_2, \ldots, \theta_{n+1})$ and $p = n + 1$.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Regression

- General $N \times p$ matrix A is then of the form

$$A = \left(1 \; X^t\right), \tag{243}$$

where X is the feature matrix with columns $x^{(1)}, \ldots, x^{(N)}$, i.e.,

$$A = \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & x_2^{(N)} & \cdots & x_n^{(N)} \end{pmatrix}. \tag{244}$$

Model is then given by

$$\hat{y} = A\theta. \tag{245}$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

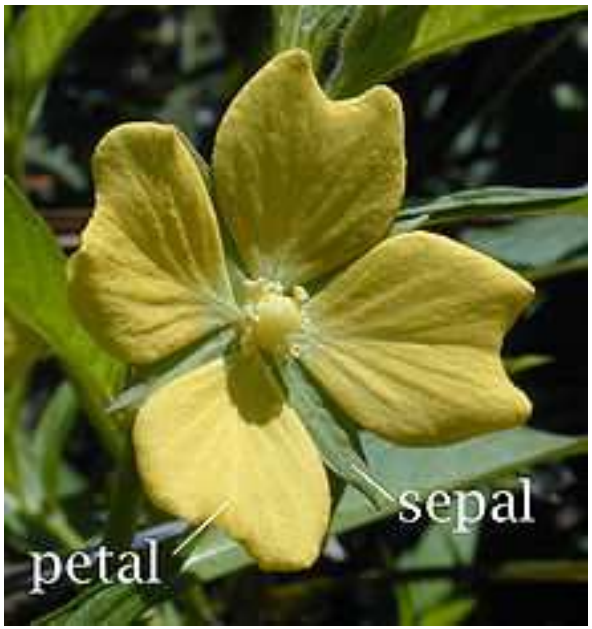
ABC Computation

Support Vector Machines

Iris Dataset

Reconsider the Iris dataset and compare the sepal and petal lengths.

```
> data(iris)
```



Iris Dataset

Linear regression:

```
> x = iris$Sepal.Length
> y = iris$Petal.Length

> model1 = lm(y ~ x)
> coefficients(model1)
(Intercept)           x
  -7.101443    1.858433
```

Linear model (regression line):

$$y = 1.86x - 7.10.$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Iris Dataset (Cont'd)

$\hat{y}$ values:

```
> fitted.values(model1)
       1         2         3
2.3765648 2.0048782 1.6331916 ...
```

Residuals:

```
> residuals(model1)
           1             2             3
-0.976564819 -0.604878224 -0.333181628 ...
```

y values:

```
> y
[1] 1.4 1.4 1.3 ...
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Iris Dataset

Polynomial regression:

```
> new_x = cbind(x, x^2)
> model2 = lm(y ~ new_x)
> coefficients(model2)
 (Intercept)     new_xx       new_x
-17.4467139   5.3921642  -0.2958593
```

Quadratic model (regression parabola):

$$y = -17.45 + 5.40x - 0.30x^2.$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Iris Dataset (Cont'd)

$\hat{y}$ values:

```
> fitted.values(model1)
        1         2         3
2.3580220 1.8713078 1.3609249 ...
```

Residuals:

```
> residuals(model1)
           1            2            3
-0.958021952 -0.471307798 -0.060924896 ...
```

y values:

```
> y
[1] 1.4 1.4 1.3 ...
```

Iris Dataset (Cont'd)

Plotting both models,

```
> ggplot(data=iris)
    + geom_point(aes(x = iris$Sepal.Length,
                     y = iris$Petal.Length))
    + geom_point(aes(x = iris$Sepal.Length,
                     y = model1$fit,color="red"))
    + geom_point(aes(x = iris$Sepal.Length,
                     y = model2$fit,color="blue"))
    + theme(panel.background=element_blank())
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

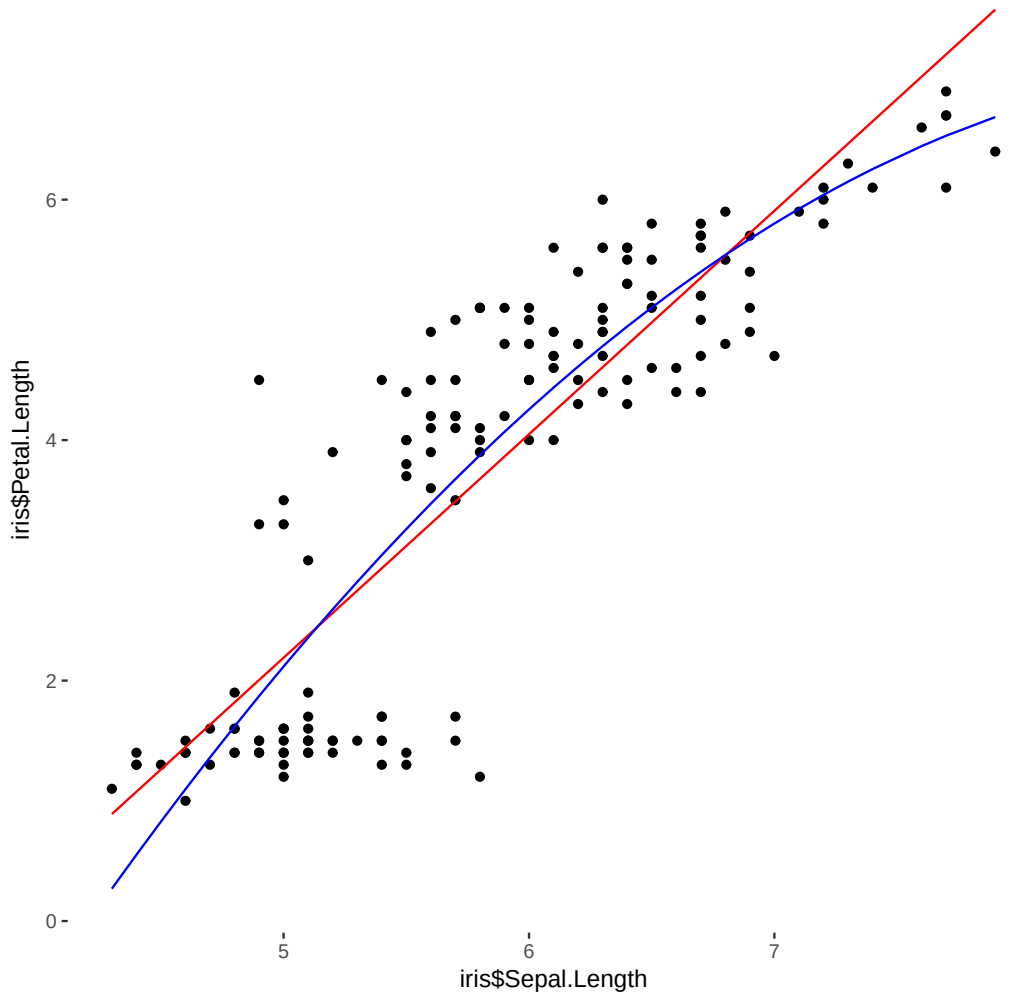
Clustering and Classification

ABC Computation

Support Vector Machines

Iris Dataset (Cont'd)

Linear versus quadratic fit for sepal and petal lengths of Iris data:



Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Clustering and Classification

- Clustering problem
- k-means algorithm
- Boolean classification
- Least squares classification
- Examples: Iris dataset, protein intake, Titanic

S. Boyd, L. Vandenberghe: Introduction to Applied Linear Algebra, Cambridge Univ. Press, 2018.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Clustering

- Fundamental problem in machine learning
- Applications in unsupervised learning
- Famous clustering method called k-means
- Example: protein intake

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Clustering

- Given a collection of samples $x_1, \ldots, x_N$ in the Euclidean space $\mathbb{R}^n$.
- The aim is to partition the samples into k clusters or groups such that the vectors in each cluster are close to each other w.r.t. Euclidean metric.
- The number of clusters k is usually much smaller than the number of samples N; values of k range from a handful to a few hundred and values of N range from a few hundred to billions.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Clustering – Example

- Iris data set is a multivariate data set consisting of 50 samples each from one of three Iris species:

  Iris setosa, Iris versicolor, and Iris virginica.

- Each sample is characterized by four features:
  - Length and width of sepals (in cm).
  - Length and width of petals (in cm).
- Scatterplots (below) compare the features in pairs of the three species.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

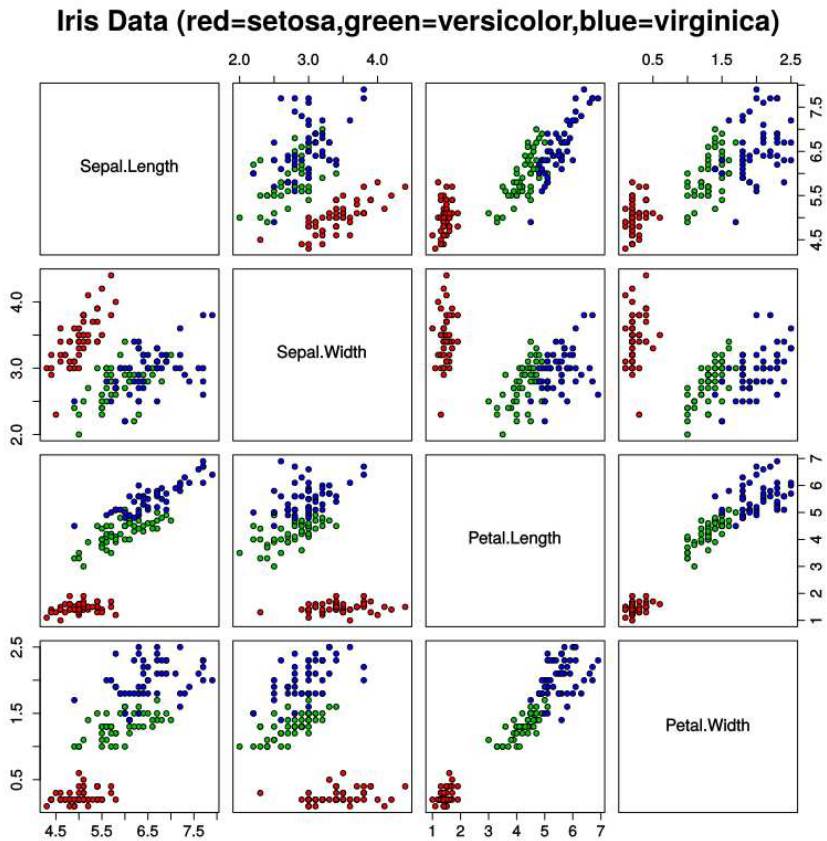
Clustering and Classification

ABC Computation

Support Vector Machines

Clustering – Example

Clustering of the features in pairs of the 150 Iris samples:



Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Clustering

- Given a collection of samples $x_1, \ldots, x_N \in \mathbb{R}^n$.
- A *clustering* of these data into k groups denoted $1, \ldots, k$ is an assignment such that sample x_i is designated to group c_i, $1 \leq i \leq N$.
- The jth group of samples is thus given by

$$G_j = \{x_i \mid c_i = j\}, \quad 1 \leq j \leq k. \tag{246}$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Clustering

- Each group G_j is assigned a representative (center) vector $z_j \in \mathbb{R}^n$, $1 \leq j \leq k$.
- Vector z_j should be close to the vectors in the corresponding group G_j; i.e., the quantities

$$\|x_i - z_{c_i}\|, \quad 1 \leq i \leq N, \tag{247}$$

should be as small as possible; x_i is assigned class c_i and z_{c_i} is center of class.

- Clustering is assigned the overall value

$$J_c = \frac{1}{N} \sum_{i=1}^{N} \|x_i - z_{c_i}\|^2, \tag{248}$$

which is the *mean-square distance* between the samples and the corresponding representatives.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Clustering

- Mean-square distance J_c depends on the cluster assignment c and the choice of the cluster representatives $z_1, \ldots, z_k$.
- Aim is to find a cluster assignment and cluster representatives such that the objective J_c is minimized; such a clustering is *optimal*.
- In practice, finding an optimal clustering is usually intractable.
- k-means algorithm provides generally a good approximation to the optimal clustering; solutions found by the k-means algorithm are usually *suboptimal*.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Clustering – Fixed Representatives I

- First, suppose the group representatives $z_1, \ldots, z_k$ are fixed.
- The task is to find the cluster assignment $c_1, \ldots, c_N$ which minimizes the objective J_c; this problem can be solved in an exact manner!
- The choice of the cluster c_i to which sample x_i belongs is only affected by the ith term of the objective J_c,

$$\frac{1}{N}\|x_i - z_{c_i}\|^2. \tag{249}$$

- This term can be minimized by choosing c_i as the value $j \in \{1, \ldots, k\}$ such that

$$\frac{1}{N}\|x_i - z_j\|^2$$

becomes minimal.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Clustering – Fixed Representatives II

- Sample x_i is thus assigned the *nearest neighbor* among the representatives,

$$\|x_i - z_{c_i}\|^2 = \min_{1 \le j \le k} \|x_i - z_j\|^2, \quad 1 \le i \le N. \tag{250}$$

- Thus when the cluster representatives are fixed, the optimal group assignment can be found by designating each sample to its nearest representative.
- Objective J_c is then given by

$$J_c = \frac{1}{N} \sum_{i=1}^{N} \min_{1 \le j \le k} \|x_i - z_j\|^2. \tag{251}$$

This is the mean of the squared distance between the samples and their closest representative.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Clustering – Optimizing the Group Representatives I

- Second, suppose the group assignment is fixed; i.e., sample x_i is assigned cluster c_i, $1 \leq i \leq N$.
- Task is to choose the group representatives such that the objective J_c is minimized; this problem can be solved in an exact manner.
- Objective J_c is decomposed into k partial sums such that each term is designated to one group,

$$J_c = J_{c,1} + \ldots + J_{c,k}, \tag{252}$$

where

$$J_{c,j} = \frac{1}{N} \sum_{i \in G_j} \|x_i - z_j\|^2, \quad 1 \leq j \leq k, \tag{253}$$

is the partial sum corresponding to the jth group.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Clustering – Optimizing the Group Representatives II

- Choice of group representative z_j only affects the term $J_{c,j}$.
- Each representative z_j can thus be chosen to minimize the corresponding sum $J_{c,j}$.
- Vector z_j can be selected as the *centroid* of the samples in the group G_j,

$$z_j = \frac{1}{|G_j|} \sum_{i \in G_j} x_i, \quad 1 \leq j \leq k, \tag{254}$$

  where $|G_j|$ denotes the cardinality of the group G_j.
- Objective J_c in (252) can thus be minimized by choosing each group representative as the centroid of the samples in the associated group.

Clustering – k-means Algorithm

Require: List of samples $x_1, \ldots, x_N$ in $\mathbb{R}^n$, initial list $z_1, \ldots, z_k$ of group representatives

Ensure: Partitioning of samples into k groups by minimizing J_c

repeat

for i from 1 to N **do**

Assign sample x_i to the group associated with the nearest representative

end for

for j from 1 to k **do**

Set z_j to be the centroid of the samples in group j

end for

until Time elapsed or objective J_c changes only slightly over several iterations

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Clustering – k-means Algorithm

- k-means algorithm (Stuard Lloyd, 1957) iterates between the two subproblems: updating the group assignment and updating the group representatives.
- In each step, the objective J_c gets improved unless the step does not change the assignment.
- Ties can be broken by assigning sample x_i to group z_j with the smallest value of j.
- In the same step, it can happen that one or more groups are empty; such groups should be deleted and then the algorithm would come up with fewer than k groups.
- If the group assignment is the same in two consecutive iterations, the representatives will also not change and then the algorithm should terminate as it will not find further improvements.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Clustering – k-means Algorithm

- Initial setting of the group representatives is purely random.
- k-means algorithm is a heuristic.
- It is thus common to run the algorithm several times with different initial representatives and then to choose the assignment with the smalles objective J_c.
- It is also common to run the k-means algorithm for several values of k and compare the results and particularly the resulting objectives.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Clustering – Example

Consider 25 European countries and their protein intake (in percent) from several food sources:

```
> url = 'http://www.biz.uiowa.edu/faculty/jledolter/
                DataMining/protein.csv'

> food <- read.csv(url)
> food
#     Country RedMeat WhiteMeat Eggs ...
#1    Albania    10.1       1.4  0.5
#2    Austria     8.9      14.0  4.3
#3    Belgium    13.5       9.3  4.1
#4   Bulgaria     7.8       6.0  1.6
#5      Czech     9.7      11.4  2.8
#6    Denmark    10.6      10.8  3.7
#7   EGermany     8.4      11.6  3.7
#8    Finland     9.5       4.9  2.7
#9     France    18.0       9.9  3.3
#10    Greece    10.2       3.0  2.8
```

Clustering – Example (Cont'd)

#11	Hungary	5.3	12.4	2.9
#12	Ireland	13.9	10.0	4.7
#13	Italy	9.0	5.1	2.9
#14	Holland	9.5	13.6	3.6
#15	Norway	9.4	4.7	2.7
#16	Poland	6.9	10.2	2.7
#17	Portugal	6.2	3.7	1.1
#18	Romania	6.2	6.3	1.5
#19	Spain	7.1	3.4	3.1
#20	Sweden	9.9	7.8	3.5
#21	Switz	13.1	10.1	3.1
#22	UK	17.4	5.7	4.7
#23	USSR	9.3	4.6	2.1
#24	WGermany	11.4	12.5	4.1
#25	Yugoslava	4.4	5.0	1.2

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Clustering – Example (Cont'd)

k-means algorithm with $N = 25$ and $k = 3$ applied to the protein intake of white meat and red meat:

```
> set.seed(123456789)
> grpMeat <- kmeans(food[,c("WhiteMeat","RedMeat")],
                    centers=3, nstart=10)
> grpMeat
K-means clustering with 3 clusters of sizes 8,12,5

Cluster means:
  WhiteMeat   RedMeat
1 12.062500  8.837500
2  4.658333  8.258333
3  9.000000 15.180000

Clustering vector
[1] 2 1 3 2 1 1 1 2 3 2 1 3 2 1 2 1 2 2 2 2 3 3 2 1 2
```

Clustering – Example (Cont'd)

Cluster assignments:

```
> o = order(grpMeat$cluster)

> data.frame(food$Country[o],grpMeat$cluster[o])

    food.Country.o. grpMeat.cluster.o
1    Austria                 1
2      Czech                 1
3    Denmark                 1
4   EGermany                 1
5    Hungary                 1
6    Holland                 1
7     Poland                 1
8   WGermany                 1
9    Albania                 2
10   Bulgaria                2
```

Clustering – Example (Cont'd)

11	Finland	2
12	Greece	2
13	Italy	2
14	Norway	2
15	Portugal	2
16	Romania	2
17	Spain	2
18	Sweden	2
19	USSR	2
20	Yugoslavia	2
21	Belgium	3
22	France	3
23	Ireland	3
24	Switz	3
25	UK	3

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

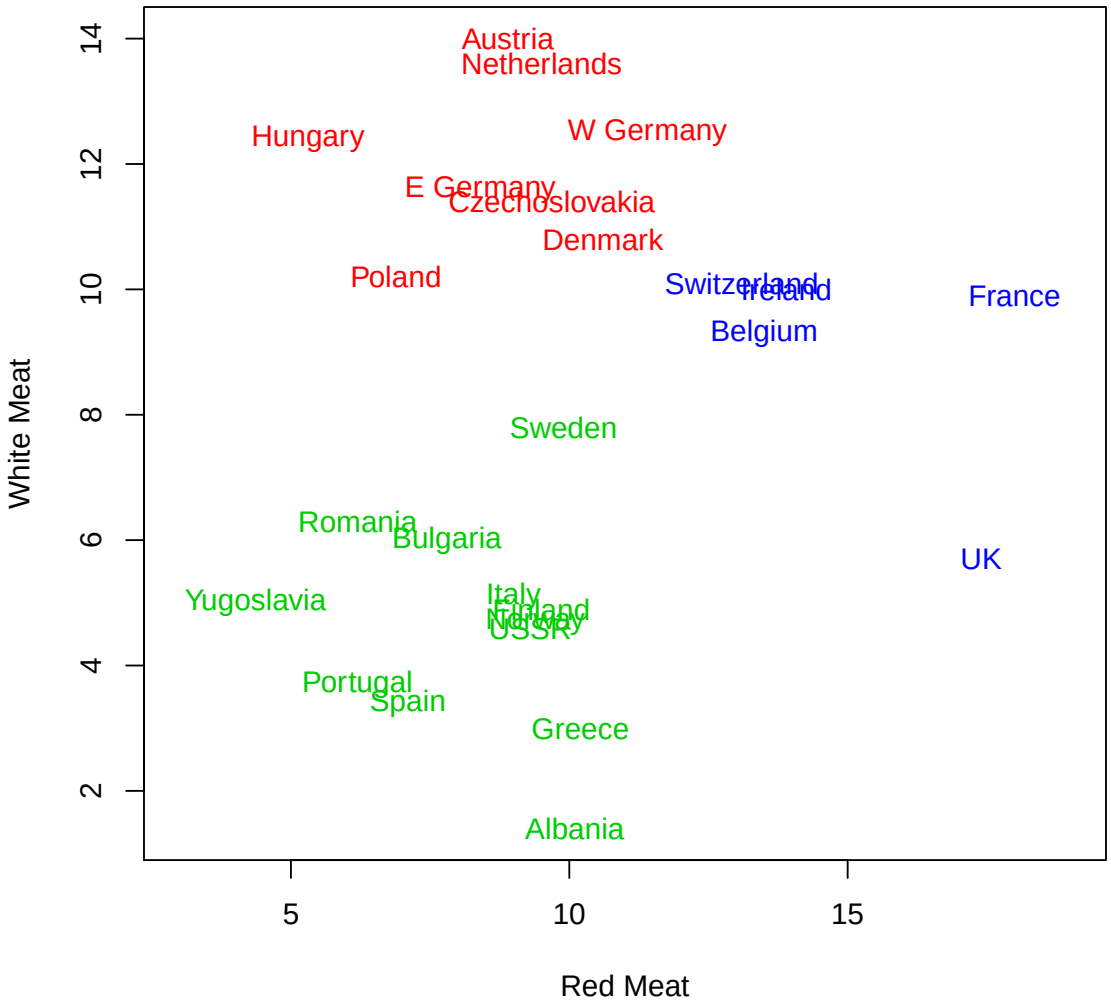
Support Vector Machines

Clustering – Example (Cont'd)

Graphical representation of clustering solution:

```
> plot(food$Red, food$White, type="n", xlim=C(3,19),
      xlab="Red Meat", ylab="White Meat",
      text(x=food$Red, y=foot$White,
      labels=food$Country, col=grpMeat$cluster+1))
```

Clustering – Example (Cont'd)



Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Classification

- Boolean classification
- Least squares classification
- Example: Titanic

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Classification

- The outcome or the dependent variable y takes on a finite number of values (label or categorial).
- In the simplest case, the categorial variable has just two values such as `true` or `false`, or `spam` or `not spam`.
- This is called *binary* or *Boolean classification problem*.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Boolean Classification

- The dependent variable y takes on the values $y = +1$ which means `true` and $y = -1$ which means `false`.
- The model is given by a *classifier* $\hat{f} : \mathbb{R}^n \to \{\pm 1\}$ with

$$\hat{y} = \hat{f}(x). \tag{255}$$

- The classifier will be constructed from the observed sample data.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Boolean Classification

For given data x, y with predicted outcome $\hat{y} = \hat{f}(x)$, there exist four possibilities:

- *True positive*: $y = +1$ and $\hat{y} = +1$.
- *True negative*: $y = -1$ and $\hat{y} = -1$.
- *False positive*: $y = -1$ and $\hat{y} = +1$.
- *False negative*: $y = +1$ and $\hat{y} = -1$.

Note:

- In the first two cases the predicted label is correct.
- In the last two cases the predicted label is incorrect.
- The third case of false positives is referred to as *type I error*.
- The fourth case of false negatives is referred to as *type II error*.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Boolean Classification

Given a collection of samples $x^{(1)}, \ldots, x^{(N)}$, categorials $y^{(1)}, \ldots, y^{(N)}$ and model function $\hat{f}$.

- The four outcome possibilities can be displayed in the form of a *contingency table*:

Outcome	$\hat{y} = +1$	$\hat{y} = -1$	Total
$y = +1$	N_{tp}	N_{fn}	N_p
$y = -1$	N_{fp}	N_{tn}	N_n
All	$N_{tp} + N_{fp}$	$N_{fn} + N_{tn}$	N

(256)

- This gives the *confusion matrix*:

$$\begin{pmatrix} N_{tp} & N_{fn} \\ N_{fp} & N_{tn} \end{pmatrix}. \tag{257}$$

The off-diagonal entries N_{fn} and N_{fp} provide the numbers of the two types of error.

Boolean Classification

These data give rise to several performance measures:

- The *error rate* gives the fraction of the committed errors:

$$\frac{N_{fp} + N_{fn}}{N}. \tag{258}$$

- The *recall rate* or *sensitivity* gives the fraction of data points with $y = +1$ that are correctly guessed $\hat{y} = +1$:

$$\frac{N_{tp}}{N_p}. \tag{259}$$

- The *false alarm rate* gives the fraction of data points with $y = -1$ which are incorrectly guessed $\hat{y} = +1$:

$$\frac{N_{fp}}{N_n}. \tag{260}$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Boolean Classification

- The *specificity* gives the fraction of the data points with $y = -1$ which are correctly guessed $\hat{y} = -1$:

$$\frac{N_{tn}}{N_n}. \tag{261}$$

- The *precision* is the fraction of true predictions which are correct:

$$\frac{N_{tp}}{N_{tp} + N_{fp}}. \tag{262}$$

A decent classifier will have small error rate and false alarm rate, and high recall rate, specificity and precision.

Boolean Classification – Example

Contingency table for spam detection of emails:

Outcome	$\hat{y} = +1$ (spam)	$\hat{y} = -1$ (not spam)	Total
$y = +1$ (spam)	88	15	103
$y = -1$ (not spam)	33	719	752
All	121	734	855

Note:

- Number of sample emails $N = 855$ of which 103 are spam $(y = +1)$ and 752 are not spam $(y = -1)$.
- Classifier provides 88 true positives, 719 true negatives, 33 false positives, and 15 false negatives.
- Error rate $(33 + 15)/855 = 5.61\%$, recall rate $88/103 = 85.44\%$, false alarm rate $33/752 = 4.39\%$, specificity $719/752 = 95.61\%$, precision $88/121 = 72.73\%$.

Least Squares Classification

- Given basis functions $f_1, \ldots, f_p$ and parameters $\theta_1, \ldots, \theta_p$.
- The sum squared error is to be minimized,

$$(y^{(1)} - \tilde{f}(x^{(1)}))^2 + \ldots + (y^{(N)} - \tilde{f}(x^{(N)}))^2, \tag{263}$$

where $\tilde{f}$ is the least squares fit over the sample set,

$$\tilde{f}(x) = \theta_1 f_1(x) + \ldots + \theta_p f_p(x). \tag{264}$$

- The *least squares classifier* $\hat{f}$ is then defined as

$$\hat{f}(x) = \operatorname{sgn}(\tilde{f}(x)), \tag{265}$$

where the *sign function* is used with $\operatorname{sgn}(z) = +1$ if $z \geq 0$ and $\operatorname{sgn}(z) = -1$ if $z < 0$.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Least Squares Classification

- The idea behind the least squares classifier is that the value $\tilde{f}(x)$ is a real number that is ideally close to $+1$ if $y^{(i)} = +1$ and close to -1 if $y^{(i)} = -1$.
- As the outcome must be binary, it is natural to choose the sign function.
- The least squares classifier is often used with a regression model,

$$\tilde{f}(x) = x^t \beta + v, \tag{266}$$

where the classifier has the form

$$\hat{f}(x) = \operatorname{sgn}(x^t \beta + v). \tag{267}$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Classification – Titanic

- The library `caret` provides functions for training and plotting classification and regression models.
- Consider the Titanic data set:

```
> titanicDF <- read.csv('http://math.ucdenver.edu/RTutorial/titanic.txt',sep='\t')
> head(titanicDF)
                                Name PClass   Age    Sex Survived
1        Allen, Miss Elisabeth Walten    1st 29.00 female        1
2         Allison, Miss Helen Loraine    1st  2.00 female        0
3 Allison, Mr Hudson Joshua Creighton    1st 30.00   male        0
4              Allison, Mrs Hudson JC    1st 25.00 female        0
5       Allison, Master Hudson Trevor    1st  0.92   male        1
6                  Anderson, Mr Harry    1st 47.00   male        1
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Classification – Titanic (Cont'd)

Structure of data:

```
> print(str(titanicDF))
'data frame': 1313 obs. of 5 variables:
$ Name     : Factor w/ 1310 levels "Abbing, Mr Antony",...
$ PClass   : Factor w/ 3 levels "1st","2nd","3rd": 1 1 1 1 1 ...
$ Age      : num 29 2 30 25 0.92 47 ...
$ Sex      : Factor w/ 2 levels "female","male": 1 1 2 1 2 2 ...
$ Survived : int 1 0 0 0 1 1 ...
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Classification – Titanic (Cont'd)

The Name variable values are unique, the title (Mr, Mrs, Miss) is extracted:

```
> titanicDF$Title <- ifelse(grepl('Mr',titanicDF$Name),'Mr',
                    ifelse(grepl('Mrs',titanicDF$Name),'Mrs',
                    ifelse(grepl('Miss',titanicDF$Name),'Miss','Nothing')))
> print(str(titanicDF))
'data frame': 1313 obs. of 5 variables:
$ Name     : Factor w/ 1310 levels "Abbing, Mr Antony",...
$ PClass   : Factor w/ 3 levels "1st","2nd","3rd": 1 1 1 1 1 ...
$ Age      : num 29 2 30 25 0.92 47 ...
$ Sex      : Factor w/ 2 levels "female","male": 1 1 2 1 2 2 ...
$ Survived : int 1 0 0 0 1 1 ...
$ Title    : chr "Miss" "Miss" "Mr" "Mrs" ...
```

Classification – Titanic (Cont'd)

The Age variable values have missing data (in the form of NA), they are imputed with the mean value of the available ages.

```
> titanicDF$Age[is.na(titanic$Age)] <- median(titanic$Age, na.rm=T)
```

The outcome variable Survived is placed at the end:

```
> titanicDF <- titanicDF[c('PClass', 'Age', 'Sex', 'Title', 'Survived')]
> print(str(titanicDF))
'data frame': 1313 obs. of 5 variables:
$ PClass   : Factor w/ 3 levels "1st","2nd","3rd": 1 1 1 1 1 ...
$ Age      : num 29 2 30 25 0.92 47 ...
$ Sex      : Factor w/ 2 levels "female","male": 1 1 2 1 2 2 ...
$ Title    : chr "Miss" "Miss" "Mr" "Mrs" ...
$ Survived : int 1 0 0 0 1 1 ...
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Classification – Titanic (Cont'd)

The factor variables need to be considered as most models accept only numerical data, they are dummified.

```
> titanicDF <- as.factor(titanicDF$Title)
> titanicDummy <- dummyVars("~.", data=titanicDF, fullRank=F)
> titanicDF <- as.data.frame(predict(titanicDummy, titanicDF))

> print(names(titanicDF))
[1] "PClass.1st"  "PClass.2nd"    "PClass.3rd"    "Age"
[5] "Sex.female"  "Sex.male"      "Title.Miss"    "Title.Mr"
[9] "Title.Mrs"   "Title.Nothing" "Title.Survived"

> print(str(titanicDF))
'data frame': 1313 obs. of 11 variables:
$ PClass.1st   : num 1 1 1 1 1 1 ...
$ PClass.2st   : num 0 0 0 0 0 0 ...
$ PClass.3rd   : num 0 0 0 0 0 0 ...
$ Age          : num 29 2 30 25 0.92 47 ...
$ Sex.female   : num 1 1 0 1 0 0 ...
$ Sex.male     : num 0 0 1 0 1 1 ...
$ Title.Miss   : num 1 1 0 0 0 0 ...
$ Title.Mr     : num 0 0 1 0 0 1 ...
$ Title.Mrs    : num 0 0 0 1 0 0 ...
$ Title.Nothing: num 0 0 0 0 1 0 ...
$ Survived     : num 1 0 0 0 1 1 ...
```

Classification – Titanic (Cont'd)

Proportions of the outcome variable:

```
> prop.table(table(titanicDF$Survived))
     0      1
0.6573 0.3427
```

Thus 34.27% of the data holds survivors of the cinematic Titanic tradegy.
The outcome variable will be renamed for subsequent use:

```
> outcomeName <- 'Survived'
> predictorsNames <- names(titanicDF)[names(titanicDF) != outcomeName]
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Classification – Titanic (Cont'd)

Library `caret` contains a large number of models for classification and regression; find list of models,

```
> names(getModeInfo())
```

Model `gbm` (Generalized Boosted Model) can be used for both classification and regression:

```
> getModelInfo()gbm$type
[1]  "Regression"  "Classification"
```

`gbm` is required in the classification mode (alter outcome variable to factor)

```
> titanicDF$survived2 <- ifelse(titanicDF$Survived==1, 'yes', 'nope')
> titanicDF$survived2 <- as.factor(titanicDF$Survived2)
> outcomeName <- 'Survived2'
```

Random forests build an ensemble of deep independent trees, whereas `gbm` builds an ensemble of shallow successive trees with each tree learning and improving on the previous.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Classification – Titanic (Cont'd)

For modeling purposes, the data set is divided into a training set and a testing set:

```
> set.seed(1234)
> splitIndex <- createDataPartition(titanicDF[,outcomeName], p=.75, list=FALSE, times=1)
> head(splitIndex)
     Resample1
[1,]         1
[2,]         2
[3,]         3
[4,]         5
[5,]         6
[6,]         7
> trainDF <- titanicDF[splitIndex,]
> nrow(trainDF)
[1] 986
> testDF <- titanicDF[-splitIndex,]
> nrow(testDF)
[1] 327
```

Classification – Titanic (Cont'd)

For the classification of the survivors, the ROC (Receiver Operating Characteristics) metric (curve of FP rate vs. TP rate) is used instead of RMSE.

```
> objModel <- train(trainDF[,predictorsNames], trainDF[,outcomeName], method='gbm',
                            trControl=objControl, metric="ROC",
                            preProc=c("cnenter","scale"))
Iter TrainDeviance ValidDeviance StepSize Improve
   1        1.2373           nan   0.1000  0.0232
   2        1.1982           nan   0.1000  0.0191
   3        1.1647           nan   0.1000  0.0151
...
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Classification – Titanic (Cont'd)

```
> summary(objModel)
                           var    rel.inf
Title.Mr             Title.Mr 24.8207771
PClass.3rd         PClass.3rd 22.5379567
Age                       Age 16.5379567
Sex.female         Sex.female 14.5848495
Sex.male             Sex.male 10.8992552
PClass.1st         PClass.1st  7.9006422
Title.Mrs           Title.Mrs  0.9438202
Title.Nothing Title.Nothing  0.7441969
PClass.2nd         PClass.2nd  0.5650417
Title.Miss          Title.Mrs  0.4893323

> print(objModel)
Stochastic Gradient Boosting
986 samples
 10 predictors
  2 classes: 'nope', 'yes'
Pre-processing: centered(10), scaled(10)
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 657, 658, 657
Resampling results across the tuning parameters
...
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

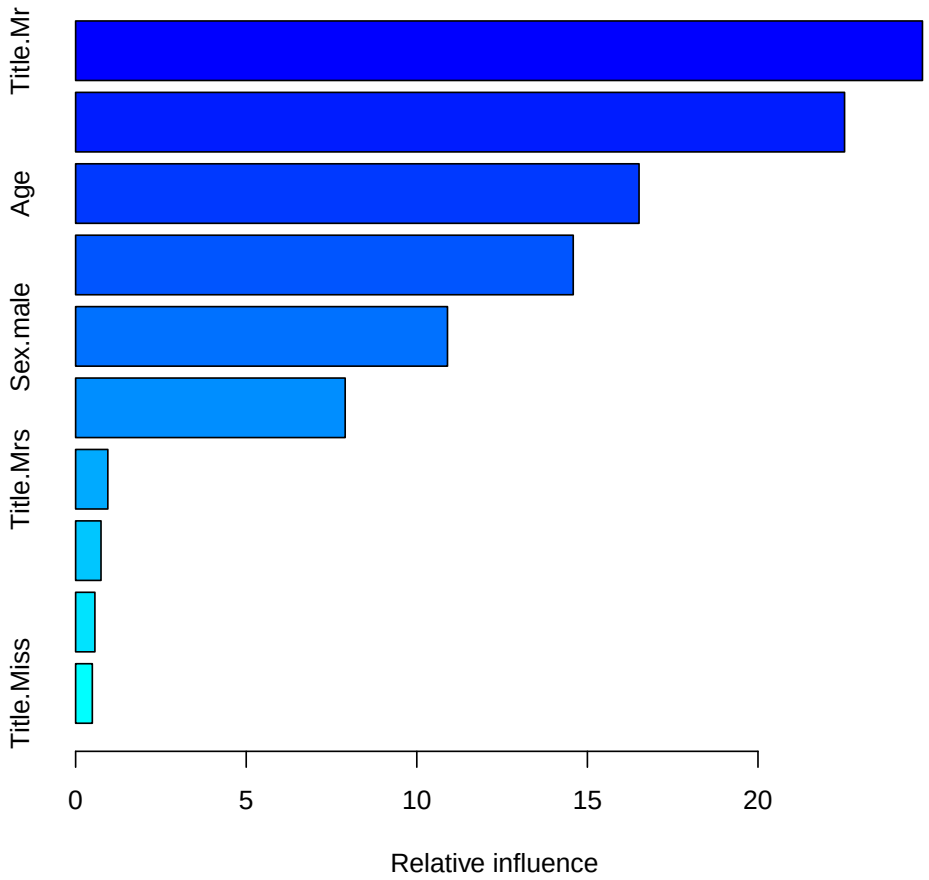
Clustering and Classification

ABC Computation

Support Vector Machines

Classification – Titanic (Cont'd)

The `summary` command shows which variables were most important in classification:



Classification – Titanic (Cont'd)

The prediction function applied to the test set is called with `raw` as type of evaluation (with values `yes` and `nope`).

```
> predictions <- predict(object=objModel, testDF[.predictorsNames], type='raw')
> head(predictions)
[1] yes nope yes nope nope nope
Levels nope yes
> print(postresample(pred=predictions, obs=as.factor(testDF[,outcomeName])))
 Accuracy     Kappa
0.8103976 0.5396458
```

The accuracy score tells that the model is correct 81.35% of the time.

Classification – Titanic (Cont'd)

The prediction function is called with `prob` as type of evaluation which exhibits the survial probabilities of the passengers:

```
> predictions <- predict(object=objModel, testDF[,predictorsNames], type='prob')
> head(predictions)
       nope       yes
1 0.02926335 0.9707366
2 0.87672849 0.1232715
3 0.44050115 0.5594989
4 0.69500967 0.3049903
5 0.69500967 0.3049903
6 0.50727652 0.4927235
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Approximate Bayesian Computation

- Bayes' rule
- ABC rejection algorithm
- Human demographic history (example)

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Bayes' rule

By Bayes' rule (7), the conditional probability $p(\theta \mid D)$ of a parameter θ given data D is related to the probability of D given θ as follows:

$$p(\theta \mid D) = \frac{p(D \mid \theta) \cdot p(\theta)}{p(D)}, \tag{268}$$

where $p(\theta \mid D)$ is the posterior, $p(D \mid \theta)$ the likelihood, $p(\theta)$ the prior and $p(D)$ the evidence.

By eliminating evidence as a normalizing constant, relative posterior plausability is given by

$$p(\theta \mid D) \propto p(D \mid \theta) \cdot p(\theta) \tag{269}$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

ABC

- Likelihood function is central expressing the probability of observed data under a statistical model (see HMM).
- For complex models, an analytical formula for the likelihood might not be available.
- ABC bypasses the computation of the likelihood (David Rubin, 1984).
- Approximate estimation of likelihood/posterior, when likelihood cannot be explicitly determined.

Soft Computing

K.-H. Zimmermann

Contents

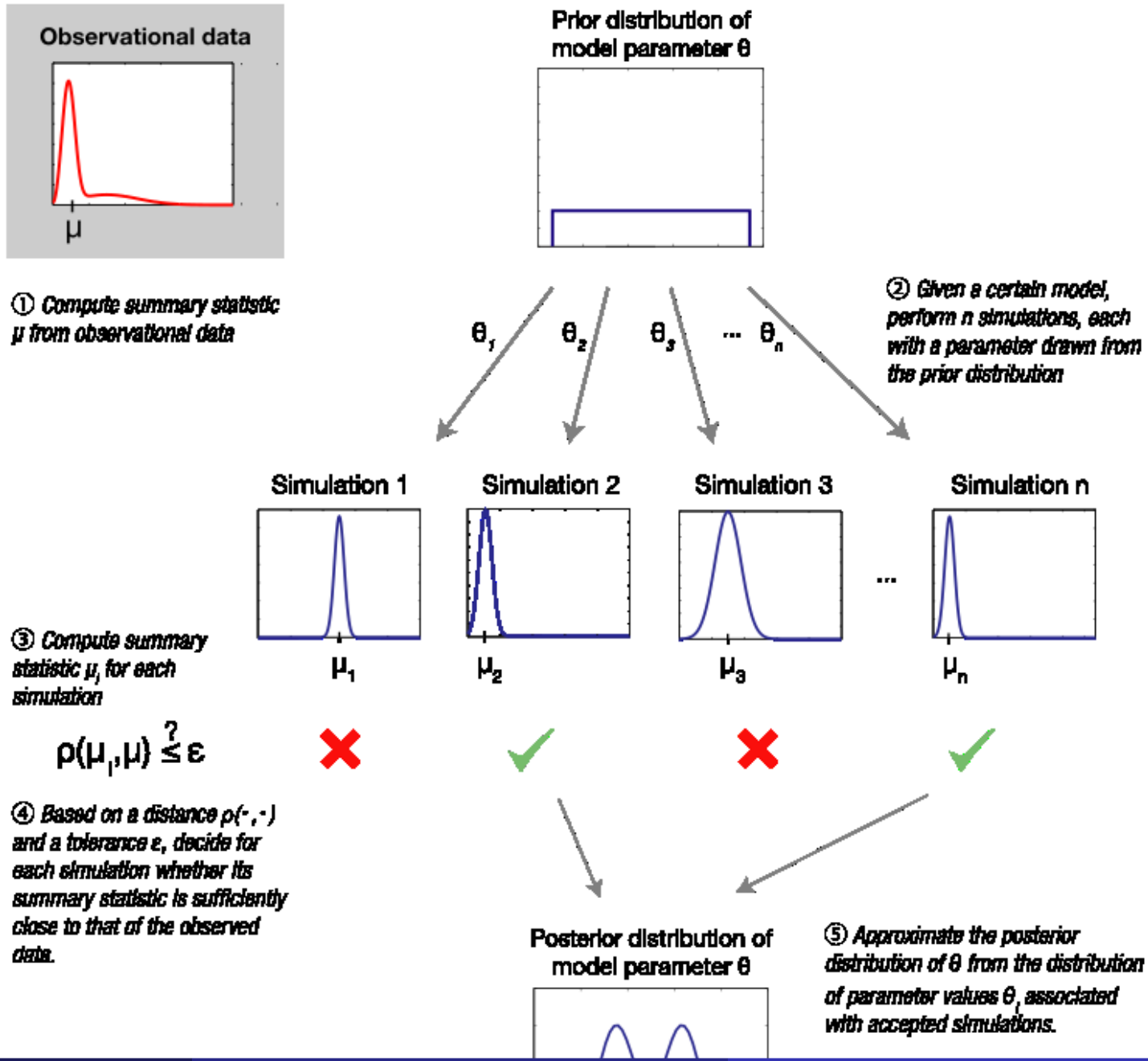
Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines



Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

ABC - Rejection Algorithm

- Sample a set of parameter points $\hat{\theta}$ from the prior distribution.
- Simulate data set $\hat{D}$ under statistical model M specified by $\hat{\theta}$.
- If data $\hat{D}$ are too different from observed data D, the parameter value $\hat{\theta}$ is discarted, i.e., $\hat{D}$ is accepted with tolerance rate $\epsilon > 0$ if

$$\delta(D, \hat{D}) \leq \epsilon, \tag{270}$$

where δ is given by a metric (e.g., Euclidean distance).

Instead of the data, summary statistics (mean, variance) may be used in (270).

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

ABC - Example

Install and load packages:

```
> install.packages("abc")
> install.packages("abc.data")
> library(abc)
> require(abc.data)
> help(abc)                 % online help
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

ABC - Example

Human demographic history:

- Analyze 50 autosomal non-coding regions from Hausa (Africa), Chinese (Asia) and Italian (Europe) population.
- Data: average nucleotide diversity ($\bar{\pi}$), mean and variance of Tajima's D table.

Summary statistics of observed data (Voight et al., 2005):

```
> data(human)
> stat.voight
             pi TajD.m TajD.v
hausa   0.00110  -0.20   0.55
italian 0.00085   0.28   1.19
chinese 0.00079   0.18   1.08
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

ABC - Example

Consider three models of demographic history:

- constant population size,
- exponential growth after a period of constant population size,
- population bottleneck, where after the bottleneck the population recovers to its original size.

50,000 data under each of the three models were simulated using software `ms` and the three summary statistics were calculated in each model.
Exact R code available (`/inst/doc/runms.R`).

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

ABC - Example

Summary statistics under the three demographic models: population bottleneck (bott), constant population size (const), exponential growth (exp):

```
> par(mfcol=c(1,3), mar=c(5,3,4,.5))
> boxplot(stat.3pops.sim[,"pi"]~models,
          main="Mean nucleotide diversity")
> boxplot(stat.3pops.sim[,"TajD.m"]~models,
          main="Mean Tajima'D")
> boxplot(stat.3pops.sim[,"TajD.v"]~models,
          main="Var in Tajima'D")
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

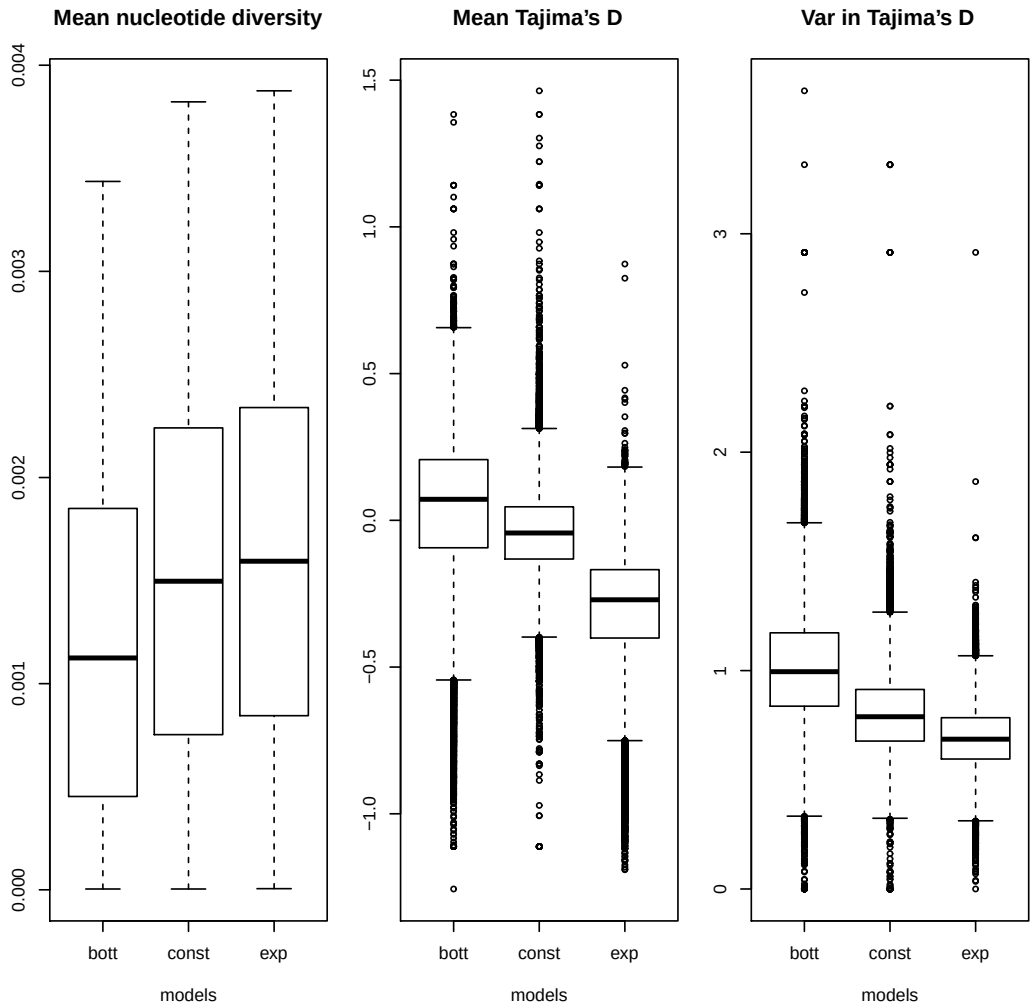
Clustering and Classification

ABC Computation

Support Vector Machines

ABC - Example

Population bottleneck (bott), constant population size (const), exponential growth (exp):



Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

ABC - Example

Posterior probabilities of each demographic model using the rejection method:

```
> modelsel.ha <- postpr(stat.voight["hausa",], models,
          stat.3pops.sim, tol=.05, method("rejection")
> modelsel.it <- postpr(stat.voight["italian",], models,
          stat.3pops.sim, tol=.05, method("rejection")
> modelsel.ch <- postpr(stat.voight["chinese",], models,
          stat.3pops.sim, tol=.05, method("rejection")
```

ABC - Example

Hausa data support best the model of exponential growth:

```
> summary(modelsel.ha)
...
Proportion of accepted simulations (rejection)
  bott  const    exp
0.0199 0.3132 0.6669
...
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

ABC - Example

Italian data support best the model of population bottleneck:

```
> summary(modelsel.it)
...
Proportion of accepted simulations (rejection)
  bott  const    exp
0.8487 0.1502 0.0004
...
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

ABC - Example

Chinese data support best the model of population bottleneck:

```
> summary(modelsel.ch)
...
Proportion of accepted simulations (rejection)
  bott  const    exp
0.6837 0.3159 0.0004
...
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Support Vector Machine (SVM)

- SVM is a supervised learning model which analyzes data for classification.
- Introduced by Vladimir N. Vapnik and Alexey Y. Chervonenskis in 1963.
- Extended by Vapnik et al. to allow nonlinear classification (kernel trick) in 1992.
- Applications in categorization of text, classification of images, and recognition of hand-written characters.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Linear SVM

Consider a linear classifier for a binary classification problem with class labels $y \in \{\pm 1\}$ and features $x \in \mathbb{R}^n$.

- Suppose there is a collection of training samples $D = ((x^{(1)}, y_1), \ldots, (x^{(m)}, y_m))$.
- Find the *maximum-margin hyperplane* that divides the collection of feature vectors $x^{(i)}$ with $y_i = 1$ from the collection of points with $y_i = -1$ such that the distance between the hyperplane and the nearest points from either group is maximized.
- Each *hyperplane* can be written as the set of points x fulfilling

$$w^T x + b = 0, \tag{271}$$

where w is the normal vector to the hyperplane.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Linear SVM

Suppose the sample data are linearly separable.

- It is possible to find two parallel hyperplanes which separate the two classes of samples in such a way that the distance between them is as large as possible.
- The region bounded by these two hyperplanes is called the *margin*, and the *maximum-margin hyperplane* is the hyperplane that lies halfway between them.
- These hyperplanes can be described by the equations

$$w^T x + b \geq 1 \quad \text{and} \quad w^T x + b \leq -1 \tag{272}$$

depending on whether the sample x lies in the class $y = 1$ or $y = -1$, resp.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Linear SVM

- It follows that each sample $(x^{(i)}, y_i)$ satifies

$$y_i \cdot (w^T x^{(i)} + b) \geq 1, \quad 1 \leq i \leq m. \tag{273}$$

- Geometrically, the distance between these two hyperplanes is $2/\|w\|$.
- Thus maximizing this distance means minimizing $\|w\|$ which leads to the convex optimization problem

$$\begin{array}{ll} \min_{w,b} & \|w\|^2 \\ \text{s.t.} & y_i \cdot (w^T x^{(i)} + b) \geq 1, \quad 1 \leq i \leq m, \end{array} \tag{274}$$

which can be tackled by standard convex solvers (objective function is convex: $\|w\|^2 = w_1^2 + \ldots + w_n^2$, constraints are linear).

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Linear SMV – Hard-Margin Classifier

- The parameters w, b which solve problem (274) provide the socalled *hard-margin* linear classifier

$$h(x) = \operatorname{sgn}(w^T x + b), \tag{275}$$

  where sgn denotes the sign function.

- The maximization-margin hyperplane is completely determined by those feature vectors that lie closest to it; these feature vectors are called *support vectors*.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

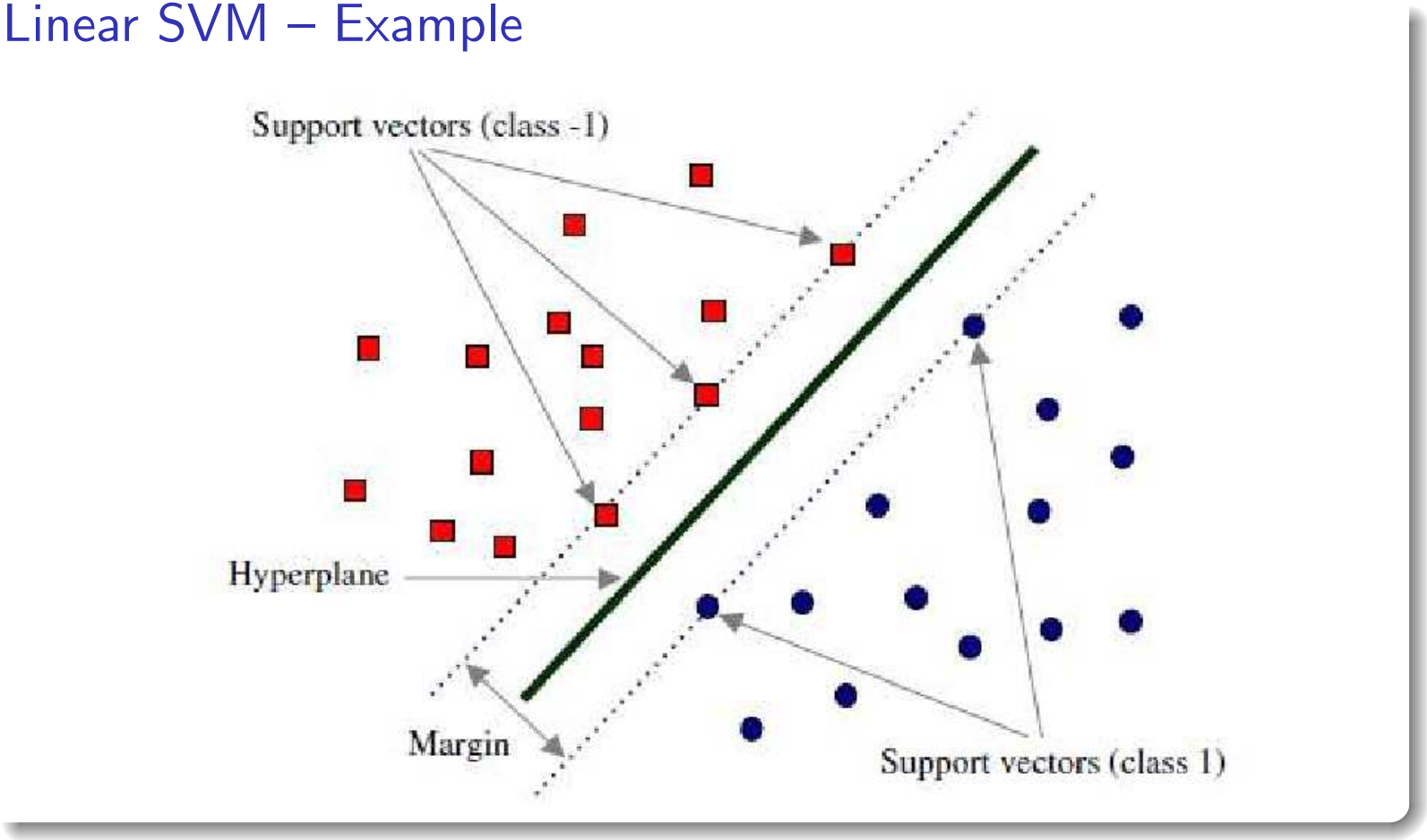
Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Linear SVM – Example



Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Linear SVM – Soft-Margin Classifier

Extend the linear classifier when sample data are not linearly separable.

- Define the *hinge loss* function

$$\max\{0, 1 - y_i \cdot (w^T x^{(i)} - b)\}. \tag{276}$$

- This function is 0 if $y_i \cdot (w^T x^{(i)} + b) \geq 1$, i.e., the sample is correctly classified. Otherwise, the function's value is proportional to the distance from the max-margin hyperplane.
- Problem is to solve the minimization problem

$$\left[\frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i \cdot (w^T x^{(i)} - b)\}\right] + \lambda \|w\|^2 \tag{277}$$

with parameter λ.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

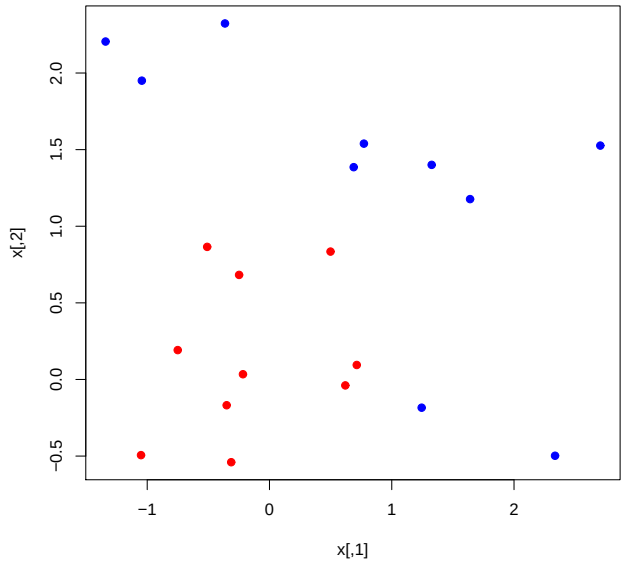
ABC Computation

Support Vector Machines

Linear SVM – Example (R)

Generate 20×2 matrix with normally distributed entries in two classes.

```
> set.seed(1234)
> x <- matrix(rnorm(40), 20, 2)
> y <- rep(c(-1,1), c(10,10))
> x[y==1,] = x[y==1] + 1
> plot(x, col=y+3, pch=19)
```



Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Linear SVM – Example (Cont'd)

```
# load library containing the svm function
> library(e1071)
# data provide a dataframe with $y$ as factor variable
> da <- data.frame(x, y=as.factor(y))
# function call svm takes linear kernel with
# tune-in parameter cost 10 and scale false
> svm-fit <- svm(y ~ ., data=da, kernel="linear",
                 cost=10, scale=FALSE)
> print(svm-fit)

Parameters
SVM-Type:   C-classification
SVM-Kernel: linear
cost:       10
gamma:      0.5

Number of support vectors: 6
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

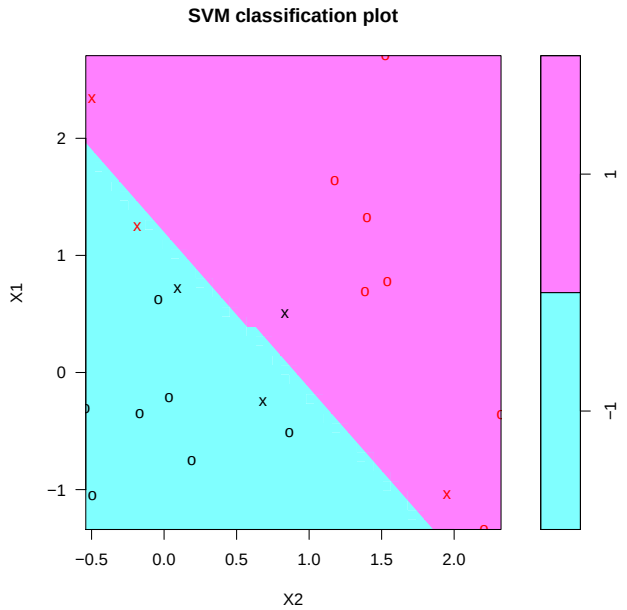
Clustering and Classification

ABC Computation

Support Vector Machines

Linear SVM – Example (Cont'd)

```
> plot(svm-fit, da)
```



Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Linear SVM – Example (Cont'd)

Find the coefficients of the separating hyperplane

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$$

```
> beta <- drop(t(svmfit$coefs)%*%x[svmfit$index,])
> beta
[1] -2.055647 -2.806278
> beta_0 <- svmfit$rho
> beta_0
[1] -2.403917
```

Then

$$-2.403917 - 2.055647 \cdot x_1 - 2.806278 \cdot x_2 = 0.$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Primal Problem

Minimization problem (277) can be rewritten as a constrained optimization problem with differentiable objective function and linear constraints:

$$\begin{array}{lll} \min_{w,b,\lambda,\xi} & \frac{1}{m}\sum_{i=1}^{m}\xi_i + \lambda\|w\|^2 & \\ \text{s.t.} & y_i(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad 1 \leq i \leq m, & (278) \\ \text{s.t.} & \xi_i \geq 0, \quad 1 \leq i \leq m. & \end{array}$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Dual Problem

Lagrangian dual problem

$$\begin{array}{ll} \max_{c,\lambda} & \sum_{i=1}^{m} c_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i c_i (x^{(i)^t} x^{(j)}) y_j c_j \\ \text{s.t.} & \sum_{i=1}^{m} c_i y_i = 0 \\ & 0 \leq c_i \leq \frac{1}{2m\lambda}, \quad 1 \leq i \leq m. \end{array} \tag{279}$$

Quadratic programming problem with quadratic objective function and linear constraints.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Linear Classification

Recovering data w and b:

- Vector w determining the separating hyperplane can be recovered by

$$w = \sum_{i=1}^{m} c_i y_i x^{(i)}. \tag{280}$$

- Offset b of the hyperplane can be recovered by finding a feature $x^{(i)}$ on the hyperplane and solving $y_i(w^T x^{(i)} - b) = 1$, which when multiplied with $y_i^{-1} = y_i$ amounts to

$$b = w^T x^{(i)} - y_i. \tag{281}$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Linear Classification

- New sample x is classified via (280,281) by the function

$$\begin{aligned} h(x) &= \operatorname{sgn}(w^T x - b) \\ &= \operatorname{sgn}\left(\left(\sum_{j=1}^{m} c_j y_j x^t x^{(j)}\right) - b\right). \end{aligned} \tag{282}$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Kernel Trick

- Linear SVM can be generalized for nonlinear classification based on kernel trick.
- Samples $x^{(i)}$ are transformed in some way to obtain data points $\phi(x^{(i)})$ in transformed feature space.
- Corresponding kernel function is defined as

$$k(x^{(i)}, x^{(j)}) = \phi(x^{(i)})^t \phi(x^{(j)}). \tag{283}$$

- Transformation may be nonlinear and the transformed feature space may have higher dimension.
- Classifier will be a hyperplane in the transformed feature space, but it may be nonlinear in the original feature space.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Common Kernel Functions

- Homogeneous polynomial functions

$$k(x, y) = (x^t y)^d, \quad d \geq 1, \tag{284}$$

- Inhomogeneous polynomial functions

$$k(x, y) = (x^t y + 1)^d, \quad d \geq 1, \tag{285}$$

- Gaussian radial basis functions

$$k(x, y) = \exp(-\gamma \|x - y\|^2), \quad \gamma > 0, \tag{286}$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Nonlinear SVM

In the transformed feature space, optimization problem (278) becomes

$$\begin{array}{ll} \max_{c,\lambda} & \sum_{i=1}^m c_i - \frac{1}{2}\sum_{i=1}^m \sum_{j=1}^m y_i c_i (\phi(x^{(i)})^t \phi(x^{(j)}) y_j c_j \\ \text{s.t.} & \sum_{i=1}^m c_i y_i = 0 \\ & 0 \leq c_i \leq \frac{1}{2m\lambda}, \quad 1 \leq i \leq m, \end{array} \tag{287}$$

where the objective function amounts to

$$\sum_{i=1}^m c_i - \frac{1}{2}\sum_{i=1}^m \sum_{j=1}^m y_i c_i k(x^{(i)}, x^{(j)}) y_j c_j. \tag{288}$$

This problem can be tackled by quadratic programming.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Nonlinear Classification

Recovering data w and b:

- Vector w determining the separating hyperplane can be recovered by

$$w = \sum_{i=1}^{m} c_i y_i \phi(x^{(i)}) \tag{289}$$

- Offset b of the hyperplane can be recoverd by finding a transformed feature $\phi(x^{(i)})$ on the hyperplane and solving $y_i(w^T \phi(x^{(i)}) - b) = 1$, which when multiplied with $y_i^{-1} = y_i$ amounts to

$$b = w^T \phi(x^{(i)}) - y_i. \tag{290}$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Nonlinear Classification

New sample x is classified by the function

$$\begin{aligned} h(x) &= \operatorname{sgn}(w^T x - b) \\ &= \operatorname{sgn}\left(\left(\sum_{j=1}^{m} c_j y_j k(x, x^{(j)})\right) - b\right). \end{aligned} \tag{291}$$

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Nonlinear Classification – Example (R)

Data are provided by the file `ESL.mixture.rds`:

```
> load("C:/Users/khz/Downloads/ESL.mixture.rds")
> names(ESL.mixture)
[1] "x" "y" "xnew" "prob" "marginal" "px1" "px2" "means"
> x <- ESL.mixture$x
> x
               [,1]         [,2]
  [1,]  2.526092968  0.321050446
...
[200,] -0.196246334  0.551403559
> y <- ESL.mixture$y
> y
  [1] 0 0 0 0 0 0 0 0 0 0
...
 [91] 0 0 0 0 0 0 0 0 0 0
[101] 1 1 1 1 1 1 1 1 1 1
...
[191] 1 1 1 1 1 1 1 1 1 1
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

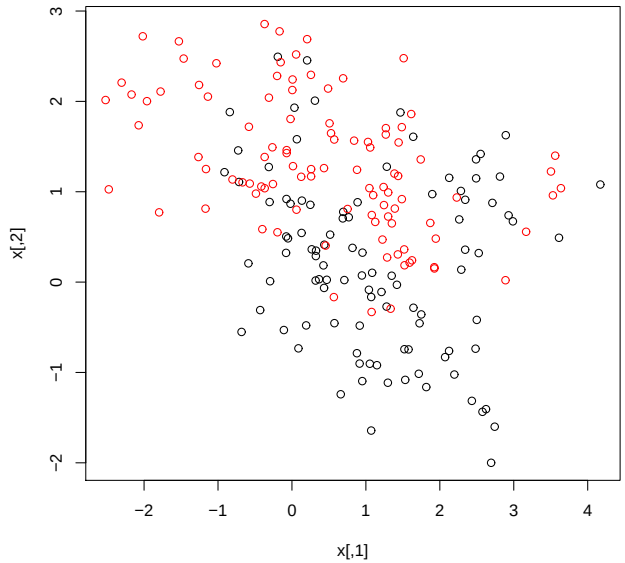
ABC Computation

Support Vector Machines

Nonlinear Classification – Example (Cont'd)

Two data sets for classification:

```
> plot(x, col=y+1)
```



Data overlap to a large extent but a contour between the data sets is partly visible.

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

ABC Computation

Support Vector Machines

Nonlinear Classification – Example (Cont'd)

Data frame with factor y is generated and data are fitted to SVM with radial kernel:

```
> da <- data.frame(y = factor(y), x)
> fit <- svm(factor(y) ~ ., data=da, scale=FALSE,
                 kernel="radial", cost=5)
```

Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

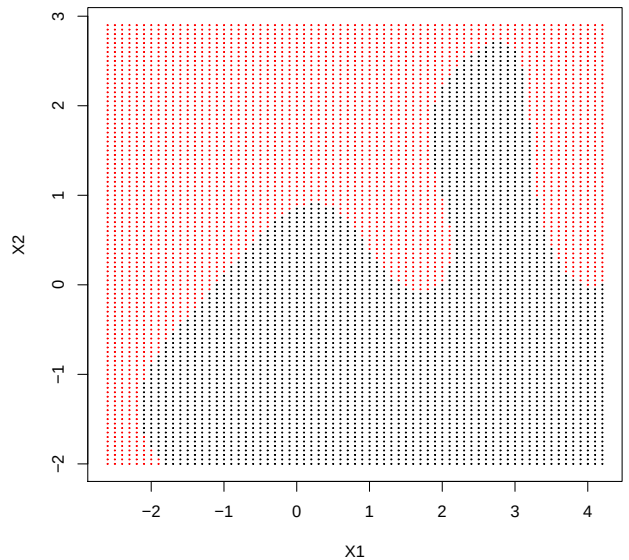
ABC Computation

Support Vector Machines

Nonlinear Classification – Example (Cont'd)

SVM contour plot:

```
> xgrid <- expand.grid(X1=ESL-mixture$px1,
                       X2=ESL-mixture$px2)
> ygrid <- predict(fit, xgrid)
> plot(xgrid, col=as.numeric(ygrid), pch=20, cex=.2)
```



Soft Computing

K.-H. Zimmermann

Contents

Naïve Bayesian Classifier

Data Fitting

Clustering and Classification

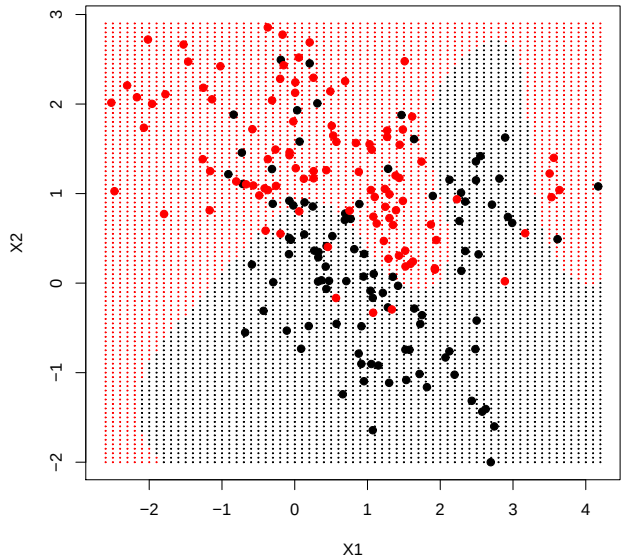
ABC Computation

Support Vector Machines

Nonlinear Classification – Example (Cont'd)

SVM classification plot:

```
> plot(x, col=y+1, pch=19)
```



Part V

BN: Inference and Learning

Contents

- Learning the conditional probabilities
- Inference in belief networks
- Score-based structure learning
- Constraint-based structure learning
- Belief networks in R

Knowledge

- Maximum likelihood estimation in fully observed models
- *EM algorithm for models with hidden data
- *Inference in belief networks
- Network scores for structure learning
- Learning structure using heuristics
- Learning structure using independence tests

Skills

- *Viterbi-like inference algorithm
- Independence tests
- Hill climbing (Heuristic)
- SGS algorithm
- Learning in belief networks using R

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

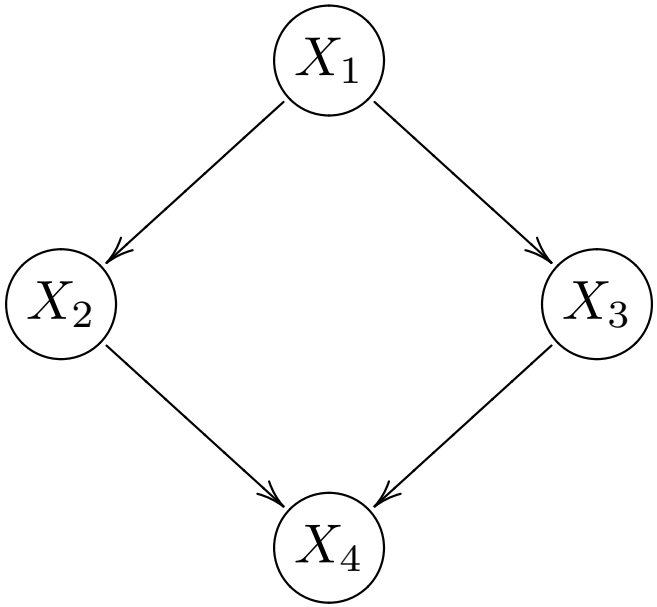
Using R

Learning Probabilities

- Learning in fully observed models
- *EM algorithm for models with hidden data

Learning Probabilities – Example

Belief network with DAG



- Random variables X_1, X_2, X_3, and X_4 defined over state set $\{0, 1\}$.
- Factorization of joint probability distribution

$$p_{X_1,X_2,X_3,X_4} = p_{X_1} \cdot p_{X_2|X_1} \cdot p_{X_3|X_1} \cdot p_{X_4|X_2,X_3}.$$

Learning Probabilities – Example (Cont'd)

Given ten samples

	$d^{(1)}$	$d^{(2)}$	$d^{(3)}$	$d^{(4)}$	$d^{(5)}$	$d^{(6)}$	$d^{(7)}$	$d^{(8)}$	$d^{(9)}$	$d^{(10)}$
X_1	1	1	0	1	0	0	0	0	1	1
X_2	1	1	0	0	1	0	1	1	1	1
X_3	0	0	1	0	0	1	0	0	1	0
X_4	0	1	1	0	0	1	1	0	1	1

Write $d^{(i)} = (x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, x_4^{(i)})$, $1 \leq i \leq 10$.
Likelihood function

$$
\begin{aligned}
L &= \prod_{i=1}^{10} p_{X_1,X_2,X_3,X_4}(d^{(i)}) \\
&= \prod_{i=1}^{10} \Big(p_{X_1}(x_1^{(i)}) \cdot p_{X_2|X_1}(x_2^{(i)} \mid x_1^{(i)}) \\
&\qquad \cdot p_{X_3|X_1}(x_3^{(i)} \mid x_1^{(i)}) \cdot p_{X_4|X_2,X_3}(x_4^{(i)} \mid x_2^{(i)}, x_3^{(i)}) \Big) .
\end{aligned}
$$

Learning Probabilities – Example (Cont'd)

Write

$$\begin{aligned}
\theta_1 &= p_{X_1}(1), & \theta_{4,0,0} &= p_{X_4|X_2,X_3}(1 \mid 0,0),\\
\theta_{2,0} &= p_{X_2|X_1}(1 \mid 0), & \theta_{4,0,1} &= p_{X_4|X_2,X_3}(1 \mid 0,1),\\
\theta_{2,1} &= p_{X_2|X_1}(1 \mid 1), & \theta_{4,1,0} &= p_{X_4|X_2,X_3}(1 \mid 1,0),\\
\theta_{3,0} &= p_{X_3|X_1}(1 \mid 0), & \theta_{4,1,1} &= p_{X_4|X_2,X_3}(1 \mid 1,1),\\
\theta_{3,1} &= p_{X_3|X_1}(1 \mid 1).
\end{aligned}$$

Likelihood function $L = L(\theta)$:

$$\begin{aligned}
L \quad = \quad & \theta_1^5(1-\theta_1)^5\\
& \theta_{2,0}^3(1-\theta_{2,0})^2\theta_{2,1}^4(1-\theta_{2,1})\\
& \theta_{3,0}^2(1-\theta_{3,0})^3\theta_{3,1}(1-\theta_{3,1})^4\\
& (1-\theta_{4,0,0})\theta_{4,0,1}^2\theta_{4,1,0}^3(1-\theta_{4,1,0})^3\theta_{4,1,1}.
\end{aligned}$$

Learning Probabilities – Example (Cont'd)

Maximize likelihood function $L = L(\theta)$ w.r.t. θ.

Maximization splits into maximation of nine separate likelihood functions:

$$
\begin{aligned}
L(\theta_1) &= \theta_1^5(1-\theta_1)^5, & L(\theta_{4,0,0}) &= 1-\theta_{4,0,0},\\
L(\theta_{2,0}) &= \theta_{2,0}^3(1-\theta_{2,0})^2, & L(\theta_{4,0,1}) &= \theta_{4,0,1}^2,\\
L(\theta_{2,1}) &= \theta_{2,1}^4(1-\theta_{2,1}), & L(\theta_{4,1,0}) &= \theta_{4,1,0}^3(1-\theta_{4,1,0})^3,\\
L(\theta_{3,0}) &= \theta_{3,0}^2(1-\theta_{3,0})^3, & L(\theta_{4,1,1}) &= \theta_{4,1,1},\\
L(\theta_{3,1}) &= \theta_{3,1}(1-\theta_{3,1})^4. &&
\end{aligned}
$$

Learning Probabilities – Example (Cont'd)

Maximization gives (see below)

$$\begin{aligned}
\hat{\theta}_1 &= \tfrac{1}{2}, & \hat{\theta}_{4,0,0} &= 0,\\
\hat{\theta}_{2,0} &= \tfrac{3}{5}, & \hat{\theta}_{4,0,1} &= 1,\\
\hat{\theta}_{2,1} &= \tfrac{4}{5}, & \hat{\theta}_{4,1,0} &= \tfrac{1}{2},\\
\hat{\theta}_{3,0} &= \tfrac{2}{5}, & \hat{\theta}_{4,1,1} &= 1,\\
\hat{\theta}_{3,1} &= \tfrac{1}{5}.
\end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

* Statistical Inference

Learning Structure

Using R

Learning Probabilities

- Given DAG $G = (V, E)$ with node set $V = \{1, \ldots, n\}$ corresponding 1-to-1 with random variables $X_1, \ldots, X_n$.
- Random variable X_i has state set $\mathcal{X}_i = \{x_1^{(i)}, \ldots, x_{m_i}^{(i)}\}$, $1 \leq i \leq n$.
- Random vector $X = (X_1, \ldots, X_n)$ has state set $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$.
- Set of instantiations of parent set Π_i of X_i is $\{\pi_1^{(i)}, \ldots, \pi_{l_i}^{(i)}\}$, $1 \leq i \leq n$.

Learning Probabilities

- Parameter set

$$\Theta = \{\theta = (\theta_{ijk}) \mid \theta_{ijk} \geq 0, \sum_{j=1}^{m_i} \theta_{ijk} = 1\} \tag{292}$$

with parameters

$$\theta_{ijk} = p_{X_i|\Pi_i}(x_j^{(i)} \mid \pi_k^{(i)}), \tag{293}$$

$1 \leq i \leq n$, $1 \leq j \leq m_i$, $1 \leq k \leq l_i$.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

* Statistical Inference

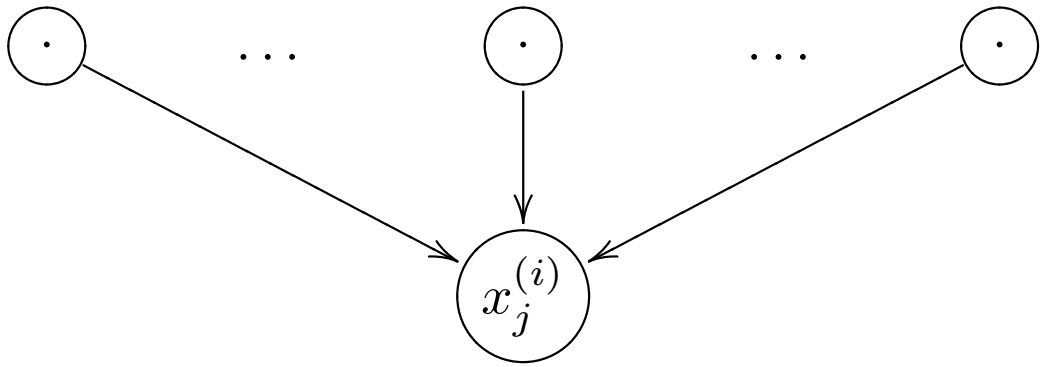
Learning Structure

Using R

Learning Probabilities

Probability $\theta_{ijk} = p_{X_i|\Pi_i}(x_j^{(i)} \mid \pi_k^{(i)})$,

- Parent nodes of i have overall instantiation $\pi_k^{(i)}$



- Node i, $1 \leq i \leq n$.
- Node i is in state $x_j^{(i)}$, $1 \leq j \leq m_i$.
- Parent nodes are in state $\pi_k^{(i)}$, $1 \leq k \leq l_i$.

Learning Probabilities – Example (Cont'd)

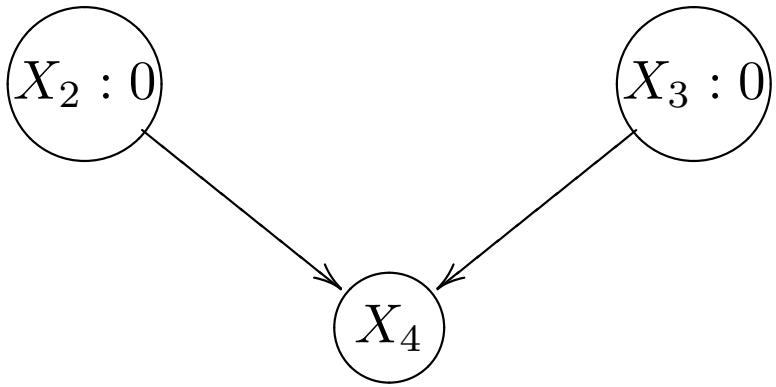
- Node 4 has two states

$$x_1^{(4)} = 0,\ x_2^{(4)} = 1.$$

- Node 4 has four instantiations of parents

$$\pi_1^{(4)} = (0,0),\ \pi_2^{(4)} = (1,0),\ \pi_3^{(4)} = (0,1),\ \pi_4^{(4)} = (1,1).$$

E.g.,



Learning Probabilities

- Given collection of N independent samples (database):

$$D = (d_1, \ldots, d_N). \tag{294}$$

- Likelihood of database:

$$L(\theta) = \prod_{r=1}^{N} p_{X|\Theta}(d_r \mid \theta, G) = \prod_{i=1}^{n} \prod_{j=1}^{m_i} \prod_{k=1}^{l_i} \theta_{ijk}^{n_{ijk}}. \tag{295}$$

where n_{ijk} is the number of times the pair $(x_j^{(i)}, \pi_k^{(i)})$ appears in the sample set.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

* Statistical Inference

Learning Structure

Using R

Learning Probabilities

- Corresponding log-likelihood function:

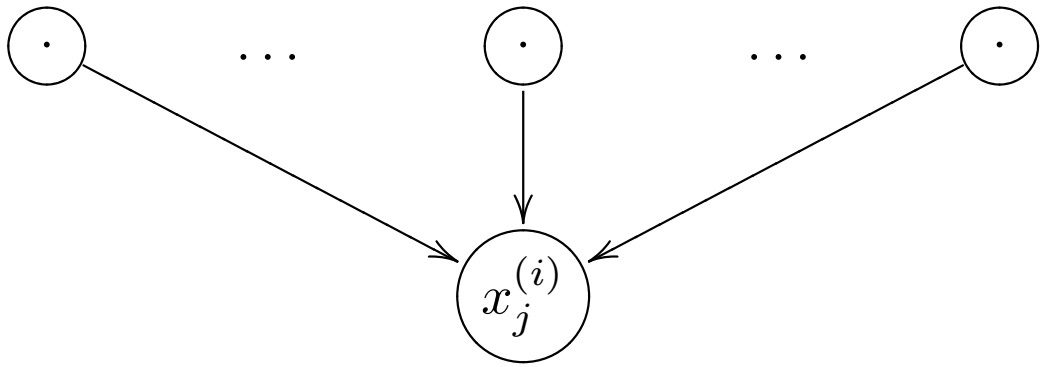
$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} \sum_{j=1}^{m_i} \sum_{k=1}^{l_i} n_{ijk} \log(\theta_{ijk}). \tag{296}$$

- Sufficient statistic of the model: (n_{ijk}).

Learning Probabilities

Multiplicity n_{ijk}:

- Parent nodes of i with overall instantiation $\pi_k^{(i)}$:



- Node i in state $x_j^{(i)}$.
- n_{ijk} is the number of times the pair $(x_j^{(i)}, \pi_k^{(i)})$ is found in the sample set.

Learning Probabilities – Example (Cont'd)

- Node 4 has two states

$$x_1^{(4)} = 0,\ x_2^{(4)} = 1.$$

- Node 4 has four instantiations of parents

$$\pi_1^{(4)} = (0,0),\ \pi_2^{(4)} = (1,0),\ \pi_3^{(4)} = (0,1),\ \pi_4^{(4)} = (1,1).$$

- Multiplicities of samples:

$$\begin{aligned} &n_{4,0,(0,0)} = 1,\ n_{4,1,(0,0)} = 0,\ n_{4,0,(1,0)} = 3,\ n_{4,1,(1,0)} = 3,\\ &n_{4,0,(0,1)} = 0,\ n_{4,1,(0,1)} = 2,\ n_{4,0,(1,1)} = 0,\ n_{4,1,(1,1)} = 1, \end{aligned}$$

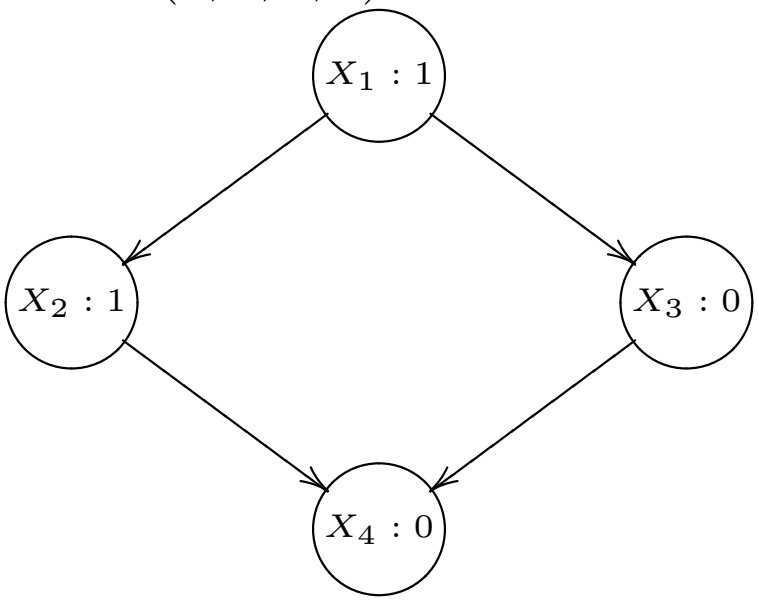
i.e., $n_{4,1,(0,1)} = 2$ corresponds to the samples $d^{(3)}$ and $d^{(6)}$: $X_2 = 0$, $X_3 = 1$, and $X_4 = 1$.

Learning Probabilities – Example (Cont'd)

Write

$$\begin{aligned}
\theta_{1,1,(\epsilon)} &= p_{X_1}(1), & \theta_{4,1,(0,0)} &= p_{X_4|X_2,X_3}(1 \mid 0,0),\\
\theta_{2,1,(0)} &= p_{X_2|X_1}(1 \mid 0), & \theta_{4,1,(0,1)} &= p_{X_4|X_2,X_3}(1 \mid 0,1),\\
\theta_{2,1,(1)} &= p_{X_2|X_1}(1 \mid 1), & \theta_{4,1,(1,0)} &= p_{X_4|X_2,X_3}(1 \mid 1,0),\\
\theta_{3,1,(0)} &= p_{X_3|X_1}(1 \mid 0), & \theta_{4,1,(1,1)} &= p_{X_4|X_2,X_3}(1 \mid 1,1),\\
\theta_{3,1,(1)} &= p_{X_3|X_1}(1 \mid 1).
\end{aligned}$$

Then for sample $d^{(1)} = (1,1,0,0)$,



$$p(d^{(1)}) = \theta_{1,1,(\epsilon)}\theta_{2,1,(1)}(1-\theta_{3,1,(1)})(1-\theta_{4,1,(1,0)}).$$

Learning Probabilities

The maximum likelihood estimate of the likelihood function $L(\theta)$ is

$$\hat{\theta}_{ijk} = \frac{n_{ijk}}{\sum_{j=1}^{m_i} n_{ijk}}, \tag{297}$$

where $1 \leq i \leq n$, $1 \leq j < m_i$, $1 \leq k \leq l_i$.

Proof.

For fixed pair (i, k) with $1 \leq i \leq n$, $1 \leq k \leq l_i$, by (292),

$$\sum_{j=1}^{m_i} \theta_{ijk} = 1. \tag{298}$$

The parameters θ_{ijk} with $1 \leq j \leq m_i$ appear in the log-likelihood function $\ell(\theta)$ as the partial sum

$$\ell_{i,k} = \sum_{j=1}^{m_i} n_{ijk} \log(\theta_{ijk}). \tag{299}$$

Proof (Cont'd)

Using

$$\theta_{im_ik} = 1 - \sum_{j=1}^{m_i-1} \theta_{ijk},$$

we obtain

$$\ell_{i,k} = \sum_{j=1}^{m_i-1} n_{ijk} \log(\theta_{ijk}) + n_{im_ik} \log\left(1 - \sum_{j=1}^{m_i-1} \theta_{ijk}\right),$$

The partial deriviate of $\ell_{i,k}$ w.r.t. θ_{ijk} becomes

$$\frac{\partial \ell_{i,k}}{\partial \theta_{ijk}} = \frac{n_{ijk}}{\theta_{ijk}} - \frac{n_{im_ik}}{1 - \sum_{j=1}^{m_i-1} \theta_{ijk}}, \quad 1 \leq j < m_i - 1.$$

Equating this expression to 0 gives $\hat{\theta}_{ijk}$ as claimed in (297). Thus the vector $\hat{\theta} = (\hat{\theta}_{ijk})$ is a critical point of the likelihood function.

Proof (Cont'd)

The critical point $\hat{\theta}$ is a maximum of the likelihood function (297).

Proof.

In view of the log-likelihood function (297), we obtain

$$\begin{aligned}
\ell(\theta) &= \sum_{i=1}^{n}\sum_{j=1}^{m_i}\sum_{k=1}^{l_i} n_{ijk}\log(\theta_{ijk}) \\
&= \sum_{i=1}^{n}\sum_{k=1}^{l_i}\left(n_{ik}\sum_{j=1}^{m_i}\hat{\theta}_{ijk}\log(\theta_{ijk})\right) \\
&= \sum_{i=1}^{n}\sum_{k=1}^{l_i}\left(n_{ik}\sum_{j=1}^{m_i}\hat{\theta}_{ijk}\log(\hat{\theta}_{ijk}) - n_{ik}\sum_{j=1}^{m_i}\hat{\theta}_{ijk}\log\left(\frac{\hat{\theta}_{ijk}}{\theta_{ijk}}\right)\right) \\
&= \sum_{i=1}^{n}\sum_{k=1}^{l_i}\left(-n_{ik}\left(H(\hat{\theta}_{ik}) + D(\hat{\theta}_{ik}\|\theta_{ik})\right)\right).
\end{aligned}$$

Proof (Cont'd)

$$\ell(\theta) = \sum_{i=1}^{n} \sum_{k=1}^{l_i} \left(-n_{ik} \left(H(\hat{\theta}_{ik}) + D(\hat{\theta}_{ik} \| \theta_{ik}) \right) \right)$$

where $n_{ik} = \sum_{j=1}^{m_i} n_{ijk}$, $\theta_{ik} = (\theta_{i1k}, \ldots, \theta_{im_ik})$, and $\hat{\theta}_{ik} = (\hat{\theta}_{i1k}, \ldots, \hat{\theta}_{im_ik})$. Then

$$\begin{aligned}
\ell(\theta) &= \sum_{i=1}^{n} \sum_{k=1}^{l_i} \left(-n_{ik} \left(H(\hat{\theta}_{ik}) + D(\hat{\theta}_{ik} \| \theta_{ik}) \right) \right) \\
&\leq -\sum_{i=1}^{n} \sum_{k=1}^{l_i} n_{ik} H(\hat{\theta}_{ik}) \\
&= \sum_{i=1}^{n} \sum_{k=1}^{l_i} n_{ik} \sum_{j=1}^{m_i} \hat{\theta}_{ijk} \log \hat{\theta}_{ijk} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{m_i} \sum_{k=1}^{l_i} n_{ijk} \log \hat{\theta}_{ijk} = \ell(\hat{\theta}). \quad \square
\end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

* Statistical Inference

Learning Structure

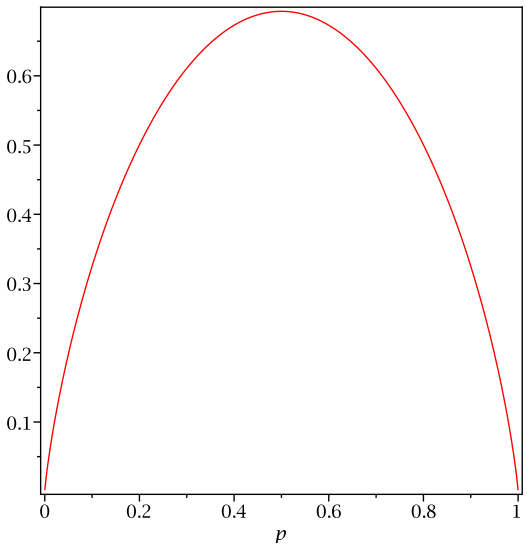
Using R

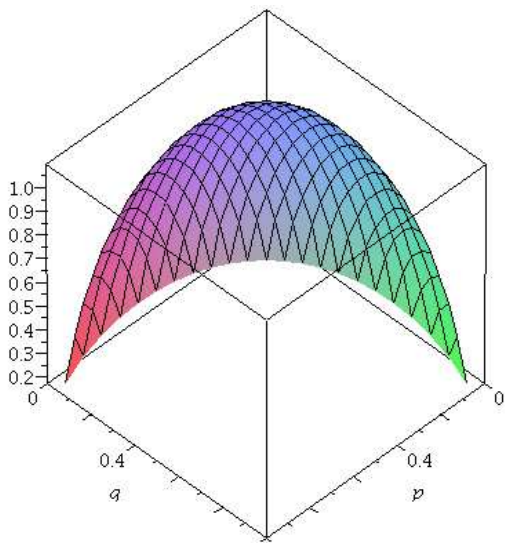
Entropy (C.E. Shannon, 1948)

The *entropy* of probability distribution $\theta = (\theta_1, \ldots, \theta_n)$ is

$$H(\theta) = -\sum_{i=1}^{n} \theta_i \log(\theta_i). \tag{300}$$

Setting $0 \log(0) = 0$ is consistent with limit $\lim_{x \to 0^+} x \log(x) = 0$.
$H(\theta) \geq 0$, since for each number x with $0 < x \leq 1$, $-\log(x) \geq 0$.
Entropy functions $H(p, 1-p)$ and $H(p, q, 1-p-q)$:





Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Using R

Kullback-Leibler Measure (1951)

Given probability distributions $\pi = (\pi_1, \ldots, \pi_n)$ and $\sigma = (\sigma_1, \ldots, \sigma_n)$.
The *Kullback-Leibler divergence* between π and σ is

$$D(\pi \| \sigma) = \sum_{i=1}^{n} \pi_i \log \left(\frac{\pi_i}{\sigma_i} \right). \tag{301}$$

- $D(\pi \| \sigma)$ is the expectation of the logarithmic difference between the probability distributions π and σ, where the expectation is taken w.r.t. the distribution π.
- $D(\pi \| \sigma)$ is defined only if $\sigma_i = 0$ implies $\pi_i = 0$ for all $1 \leq i \leq n$.
- When π_i equals 0, the contribution of the i-th term becomes 0, since $\lim_{x \to 0^+} x \log(x) = 0$.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Using R

Kullback-Leibler Measure (1951)

For probability distributions $\pi = (\pi_1, \ldots, \pi_n)$ and $\sigma = (\sigma_1, \ldots, \sigma_n)$, we have

$$D(\pi \| \sigma) \geq 0. \tag{302}$$

Moreover, $D(\pi \| \sigma) = 0$ iff $\pi = \sigma$.

Proof.

Jensen's inequality states that if X is a random variable and f is a convex function, then $E[f(X)]) \geq f(E[X])$, where E denotes the expected value.
Since the function $f(x) = -\log(x)$ is convex, we obtain

$$D(\pi \| \sigma) = -\sum_{i=1}^{n} \pi_i \log\left(\frac{\sigma_i}{\pi_i}\right) \geq -\log\left(\sum_{i=1}^{n} \pi_i \frac{\sigma_i}{\pi_i}\right) = -\log(1) = 0.$$

Second assertion is shown by induction on the length $n \geq 1$ of the distributions. □

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Using R

*Learning Probabilities

- Belief network (G, p) with DAG $G = (V, E)$, node set $V = \{1, \ldots, k, k+1, \ldots, k+l\}$ corresponds 1-to-1 with random variables $X_1, \ldots, X_k, Y_1, \ldots, Y_l$.
- States of variables $X_1, \ldots, X_k$ are *observable*.
- States of variables $Y_1, \ldots, Y_l$ are *hidden* from observer.
- Random vectors $X = (X_1, \ldots, X_k)$ and $Y = (Y_1, \ldots, Y_l)$ have state sets $\mathcal{X} = \{x^{(1)}, \ldots, x^{(m)}\}$ and $\mathcal{Y} = \{y^{(1)}, \ldots, y^{(n)}\}$, resp.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Using R

*Learning Probabilities

- Joint probability distribution depends on the parameters θ of parameter set Θ,

$$f_{i,j}(\theta) = p_{X,Y|\Theta}(x^{(i)}, y^{(j)}), \tag{303}$$

where $1 \leq i \leq m$, $1 \leq j \leq n$.

- Marginal distribution w.r.t. X is

$$f_i(\theta) = p_{X|\Theta} = \sum_{j=1}^{n} f_{ij}(\theta), \tag{304}$$

where $1 \leq i \leq m$.

*Learning Probabilities

- Given collection of N independent observed samples from state space $\mathcal{X}$,

$$D = (x^{(i_1)}, \ldots, x^{(i_N)}) \tag{305}$$

where $1 \leq i_1, \ldots, i_N \leq m$.

- This collection provides data vector

$$u = (u_1, \ldots, u_m) \in \mathbb{N}^m, \tag{306}$$

where u_i denotes the number of occurrences of the observed state $x^{(i)}$ in database D, $1 \leq i \leq m$.

- We have

$$\sum_{i=1}^{m} u_i = N. \tag{307}$$

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Using R

*Learning Probabilities

- The objective is to maximize the log-likelihood function w.r.t. the parameters $\theta \in \Theta$ for the observed data,

$$\ell_X(\theta) = u_1 \cdot \log f_1(\theta) + \ldots + u_m \cdot f_m(\theta). \tag{308}$$

- The vector u is the sufficient statistic of the model.
- The maximization of this likelihood function is usually intractable.

*Learning Probabilities

- On the other hand, given collection of N independent sample pairs from state set $\mathcal{X} \times \mathcal{Y}$,

$$D = ((x^{(i_1)}, y^{(j_1)}), \ldots, (x^{(i_N)}, y^{(j_N)})) \tag{309}$$

where $1 \leq i_1, \ldots, i_N \leq m$, and $1 \leq j_1, \ldots, j_N \leq n$.

- This collection provides the data vector

$$U = (u_{ij}) \in \mathbb{N}^{m \times n}, \tag{310}$$

where u_{ij} denotes the number of occurrences of the sequence pair $(x^{(i)}, y^{(j)})$ in the database D with $u_i = \sum_{j=1}^{n} u_{ij}$.

*Learning Probabilities

- The objective is to maximize the log-likelihood function w.r.t. the parameters $\theta \in \Theta$ for the observed sequence pairs,

$$\ell_{X,Y}(\theta) = u_{11} \cdot \log f_{11}(\theta) + \ldots + u_{mn} \cdot f_{mn}(\theta). \qquad (311)$$

- The EM algorithm provides an estimate of the parameters.
- The marginal probabilities $f_i(\theta)$ can often be calculated efficiently by an appropriate inference algorithm (Viterbi algorithm for HMM).
- In the M-step, the maximal estimates $\hat{\theta}$ can be computed directly by (297).

*EM Algorithm

Require: Joint probability function $p_{X,Y|\Theta}$, parameter set $\Theta \subseteq \mathbb{R}^d$, observed data $u = (u_i) \in \mathbb{N}^m$

Ensure: Maximum likelihood estimate $\theta^* \in \Theta$ of the log-likelihood function $\ell_X(\theta)$

[Init] Threshold $\epsilon > 0$ and random parameter $\theta \in \Theta$

[E-Step] Define matrix $U = (u_{ij}) \in \mathbb{R}_{>0}^{m \times n}$ with

$$u_{ij} = \frac{u_i \cdot f_{ij}(\theta)}{f_i(\theta)}$$

[M-Step] Compute solution $\hat{\theta} \in \Theta$ of the likelihood function $\ell_{X,Y}$ with data set $U = (u_{ij})$ as in (297)

[Compare] If $\ell_X(\hat{\theta}) - \ell_X(\theta) > \epsilon$, set $\theta \leftarrow \hat{\theta}$ and resume with E-step

[Output] $\theta^* \leftarrow \hat{\theta}$

Soft Computing

K.-H. Zimmermann

Contents

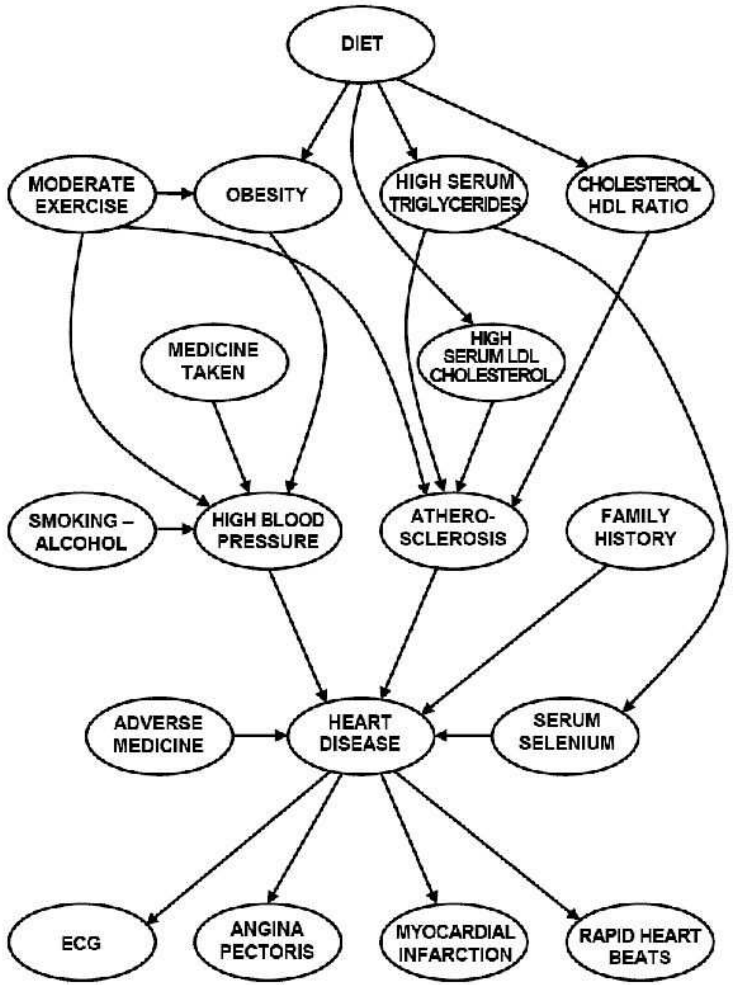
Learning Probabilities

*Statistical Inference

Learning Structure

Using R

*Statistical Inference



Statistical Inference

- Given a belief network with observed and unobserved (hidden) variables.
- Find instantiations of the hidden variables that maximize the likelihood (NP-hard).

Statistical Inference

- Marginalization of joint probability distribution
- Tropicalization of marginal distribution
- Graded belief networks
- Inference in graded belief networks

Statistical Inference

What are belief networks useful for?

- Diagnosis: $P(\text{cause}|\text{symptom})$
- Prediction: $P(\text{symptom}|\text{cause})$
- Classification: $P(\text{class}|\text{data})$

Applications in medicine, bioinformatics, text classification, speech recognition, computer troubleshooting.

Statistical Inference

- Belief network with DAG $G = (V, E)$ and joint probability distribution p.
- Node set $V = \{v_1, \ldots, v_m, v_{m+1}, \ldots, v_{m+n}\}$ with $m, n \geq 1$ is 1-to-1 with random variables $X_1, \ldots, X_m, Y_1, \ldots, Y_n$.
- $X_1, \ldots, X_m$ are observed variables with finite state sets $\mathcal{X}_1, \ldots, \mathcal{X}_m$, resp.
- $Y_1, \ldots, Y_n$ are hidden variables with finite state sets $\mathcal{Y}_1, \ldots, \mathcal{Y}_n$, resp.
- Random vectors $X = (X_1, \ldots, X_m)$ and $Y = (Y_1, \ldots, Y_n)$ have state sets $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_m$ and $\mathcal{Y} = \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_n$, resp.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Using R

Statistical Inference

- Topological sorting of random variables: $X_1 > \ldots > X_m$ and $Y_1 > \ldots > Y_n$.
- X_1 has only parents in the hidden variables and Y_1 has only parents in the observed variables.
- Factorization of global probability distribution

$$p_{X,Y} = \prod_{i=1}^{m} p_{X_i|\Pi(X_i)} \prod_{j=1}^{n} p_{Y_j|\Pi(Y_j)}. \tag{312}$$

- Probability of observed sequence $x \in \mathcal{X}$ is given by the marginal distribution

$$p_X(x) = \sum_{y \in \mathcal{Y}} p_{X,Y}(x, y). \tag{313}$$

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Using R

Statistical Inference

- Each variable Z in the DAG can be assigned a *semi-rank*, written $\rho(Z)$:
  - Each variable Z with empty parent set or parent set in the observed variables has semi-rank 0. Since the graph is a DAG, there is at least one variable with semi-rank 0.
  - Each hidden variable Z whose parents have already assigned ranks is assigned semi-rank

$$\rho(Z) = \max\{\rho(U) \mid U \text{ hidden and parent of } Z\} + 1 \quad (314)$$

  - Each observed variable Z is given the largest semi-rank of its hidden parents,

$$\rho(Z) = \max\{\rho(U) \mid U \text{ hidden and parent of } Z\}. \quad (315)$$

The conditional probability $p_{Z|\Pi(Z)}$ with given value $Z = z$ can be evaluated as soon as the parents are given.

Statistical Inference – Graded Case

- A DAG $G = (V, E)$ is *graded* if it is equipped with a *rank function* ρ from V to $\mathbb{N}_0$.
- A rank function of a DAG must be compatible with the given topological ordering and the rank must be consistent with the covering relation of the ordering.
- In our case, each variable Z with empty parent set or parent set in the observed variables is given the rank $\rho(Z) = 0$.
- Each hidden variable Z is assigned the rank $\rho = \rho(Z) \geq 1$ if all its hidden parents have rank $\rho - 1$.
- Each observed variable Z is assigned the rank $\rho = \rho(Z)$ if all its hidden parents have rank ρ.
- A belief network is *graded* if its underlying DAG is graded.

Statistical Inference

- The rank of a variable exhibits in which of the nested brackets it can be evaluated earliest in the marginal distribution p_X.
- $\rho_{\max}$ = maximal rank of the nodes in DAG.
- $X_1^{(r)}, \ldots, X_{s_r}^{(r)}$ collection of observed random variables with rank r.
- $Y_1^{(r)}, \ldots, Y_{t_r}^{(r)}$ collections of hidden random variables with rank r.
- Then $\sum_{r=0}^{\rho_{\max}} s_r = m$ and $\sum_{r=0}^{\rho_{\max}} t_r = n$.
- State set of the hidden variables with rank r is denoted by

$$D(r) = \mathcal{Y}_1^{(r)} \times \ldots \times \mathcal{Y}_{t_r}^{(r)}. \tag{316}$$

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Using R

Statistical Inference

Marginal distribution of observed data $x \in \mathcal{X}$:

$$
\begin{aligned}
p_X(x) &= \sum_y p_{X,Y}(x,y) \qquad (317)\\
&= \sum_{y \in D(\rho)} \prod_{i=1}^{s_\rho} p_{X_i^{(\rho)}|\Pi(X_i^{(\rho)})} \\
&\quad \cdot \left(\sum_{y \in D(\rho-1)} \prod_{i=1}^{s_{\rho-1}} p_{X_i^{(\rho-1)}|\Pi(X_i^{(\rho-1)})} \prod_{j=1}^{t_\rho} p_{Y_j^{(\rho)}|\Pi(Y_j^{(\rho)})} \right. \\
&\quad \cdots \\
&\quad \cdot \left(\sum_{y \in D(1)} \prod_{i=1}^{s_1} p_{X_i^{(1)}|\Pi(X_i^{(1)})} \prod_{j=1}^{t_2} p_{Y_j^{(2)}|\Pi(Y_j^{(2)})} \right. \\
&\quad \cdot \left. \left(\sum_{y \in D(0)} \prod_{i=1}^{s_0} p_{X_i^{(0)}|\Pi(X_i^{(0)})} \prod_{j=1}^{t_1} p_{Y_j^{(1)}|\Pi(Y_j^{(1)})} \prod_{j=1}^{t_0} p_{Y_j^{(0)}|\Pi(Y_j^{(0)})} \right) \cdots \right)
\end{aligned}
$$

with $\rho = \rho_{\max}$, arguments omitted for readability. Ranking guarantess well-definedness.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

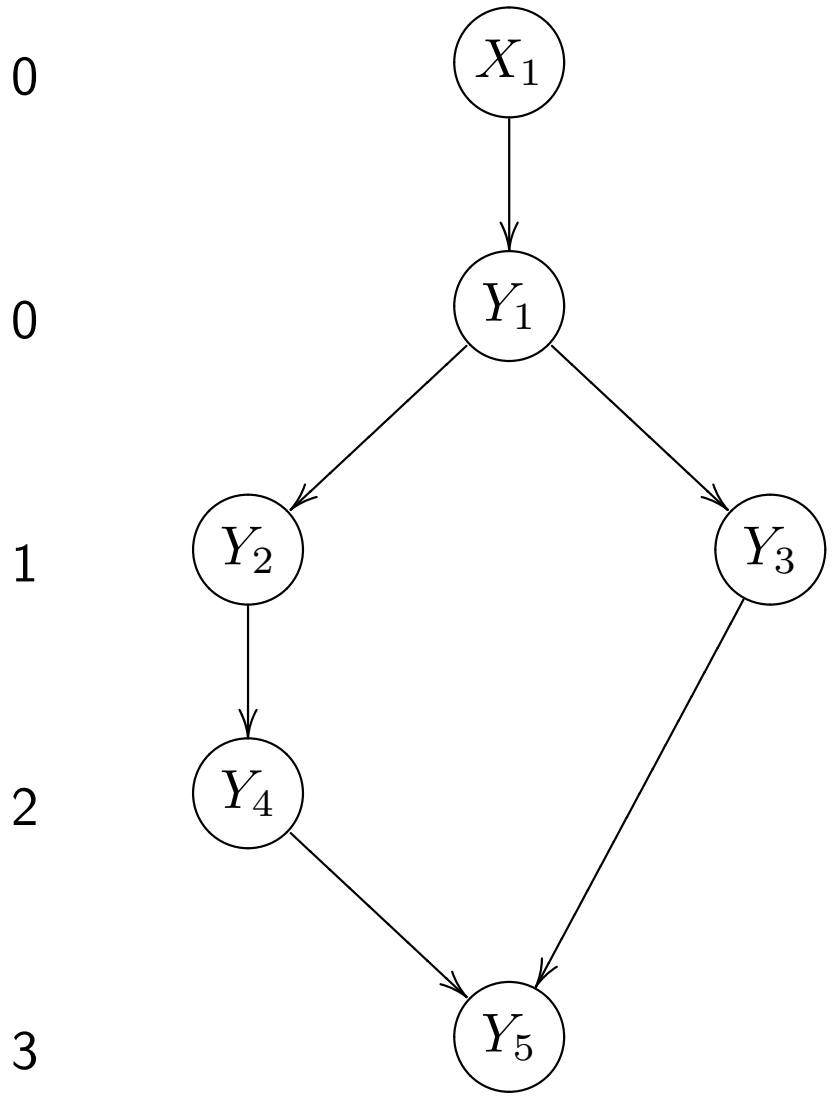
*Statistical Inference

Learning Structure

Using R

Statistical Inference – Example

Non-graded belief network with semi-ranks:



Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

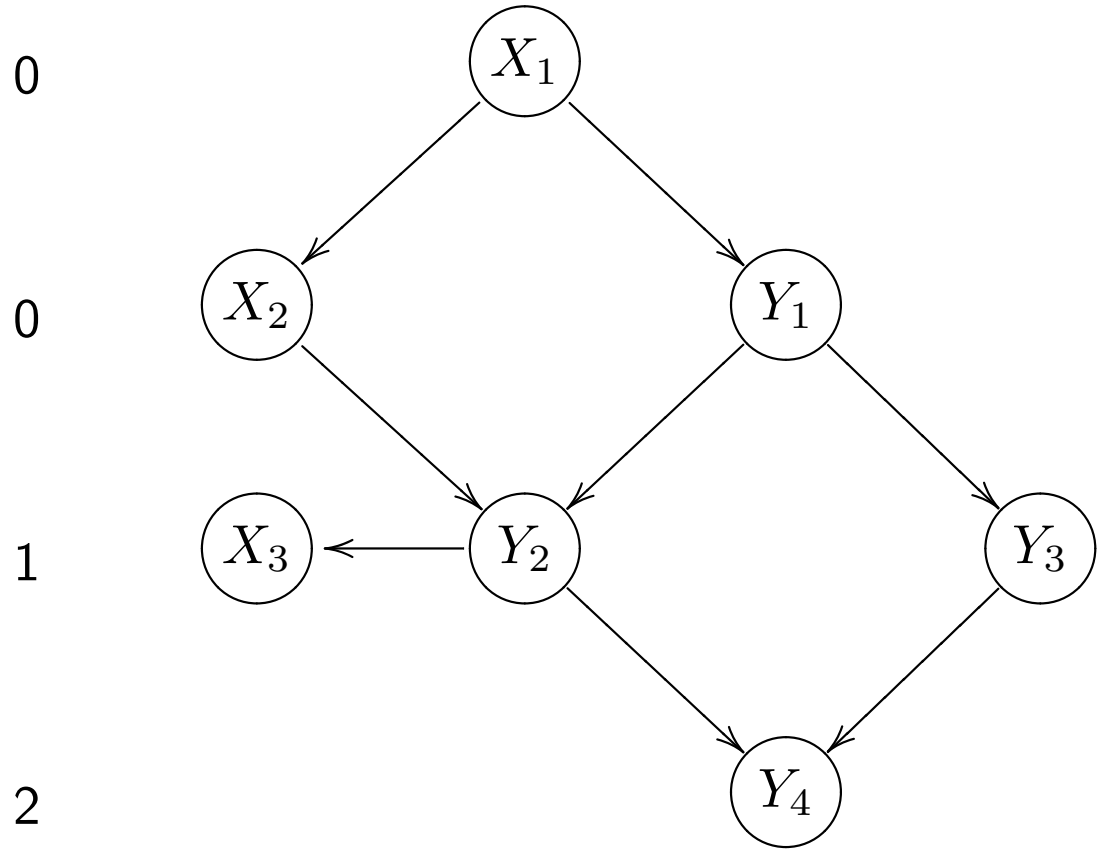
*Statistical Inference

Learning Structure

Using R

Statistical Inference – Example

Graded belief network with ranks:



Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Using R

Statistical Inference

- Given observed sequence $x \in \mathcal{X}$.
- Find one (or all) sequences $y \in \mathcal{Y}$ with maximum likelihood

$$p_{Y|X}(y \mid x) = \frac{p_{X,Y}(x,y)}{p_X(x)}. \tag{318}$$

- Since observed sequence x is fixed, the likelihood $p_{Y|X}(y \mid x)$ is directly proportional to the joint probability $p_{X,Y}(x,y)$ provided that $p_X(x) > 0$.
- Let $p_X(x) > 0$. Find a sequences $\bar{y} \in \mathcal{Y}$ with

$$\begin{aligned} \bar{y} &= \operatorname{argmax}_{y \in \mathcal{Y}} \{p_{Y|X}(y|x)\} \\ &= \operatorname{argmax}_{y \in \mathcal{Y}} \{p_{X,Y}(x,y)\}. \end{aligned} \tag{319}$$

- Each optimal sequence $\bar{y}$ is called an *explanation* of the given sequence x.
- The explanations can be found by tropicalization.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Using R

Statistical Inference

- Tropicalization map

$$\phi : (\mathbb{R}_{\geq 0}, +, \cdot) \to (\mathbb{R} \cup \{\infty\}, \oplus, \odot) : x \mapsto -\log(x). \quad (320)$$

- Put $w_{X,Y}(x,y) = -\log p_{X,Y}(x,y)$ and $w_X(x) = -\log p_X(x)$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.
- Tropicalization gives

$$w_X(x) = \bigoplus_{y \in \mathcal{Y}} w_{X,Y}(x,y). \quad (321)$$

- The explanations $\bar{y}$ can be obtained by evaluation in the tropical semiring,

$$\bar{y} = \text{argmin}_{y \in \mathcal{Y}} \{w_{X,Y}(x,y)\}. \quad (322)$$

- Put $w_{Z|\Pi(Z)} = -\log p_{Z|\Pi(Z)}$ for each random variable Z.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Using R

Statistical Inference

Tropicalization of (317) gives

$$
\begin{aligned}
w_X(x) &= \bigoplus_y w_{X,Y}(x,y) \qquad (323)\\
&= \bigoplus_{y\in D(\rho)} \bigodot_{i=1}^{s_\rho} w_{X_i^{(\rho)}|\Pi(X_i^{(\rho)})}\\
&\quad \cdot \left(\bigoplus_{y\in D(\rho-1)} \bigodot_{i=1}^{s_{\rho-1}} w_{X_i^{(\rho-1)}|\Pi(X_i^{(\rho-1)})} \bigodot_{j=1}^{t_\rho} w_{Y_j^{(\rho)}|\Pi(Y_j^{(\rho)})} \right.\\
&\quad \cdots\\
&\quad \cdot \left(\bigoplus_{y\in D(1)} \bigodot_{i=1}^{s_1} w_{X_i^{(1)}|\Pi(X_i^{(1)})} \bigodot_{j=1}^{t_2} w_{Y_j^{(2)}|\Pi(Y_j^{(2)})} \right.\\
&\quad \cdot \left. \left(\bigoplus_{y\in D(0)} \bigodot_{i=1}^{s_0} w_{X_i^{(0)}|\Pi(X_i^{(0)})} \bigodot_{j=1}^{t_1} w_{Y_j^{(1)}|\Pi(Y_j^{(1)})} \bigodot_{j=1}^{t_0} w_{Y_j^{(0)}|\Pi(Y_j^{(0)})} \right) \cdots \right)
\end{aligned}
$$

where $\rho = \rho_{\max}$, arguments omitted for readability. Ranking guarantess well-definedness.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Using R

Statistical Inference

- Find one (or all) explanations $\overline{y} \in \mathcal{Y}$ of the hidden variables given an observed sequence $x \in \mathcal{X}$.
- Let $x \in \mathcal{X}$. The tropicalization $w_X(x)$ of the marginal probability $p_X(x)$ provides the explanations of the observed sequence x.
- The tropicalization of the marginal probability does not overcome the NP-hardness of the inference problem.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Using R

Statistical Inference – Graded Case

- Inference in a graded belief network has the advantage that in the computation of the r-th expression

$$\bigoplus_{y \in D(r)} \bigodot_{i=1}^{s_r} w_{X_i^{(r)}|\Pi(X_i^{(r)})} \odot \bigodot_{j=1}^{t_{r+1}} w_{Y_j^{(r+1)}|\Pi(Y_j^{(r+1)})}, \tag{324}$$

  the terms $w_{X_i^{(r)}|\Pi(X_i^{(r)})}$ and $w_{Y_j^{(r+1)}|\Pi(Y_j^{(r+1)})}$ depend only on the parent values $y' \in D(r)$ of the hidden variables of rank r, $0 \leq r \leq \rho_{\max} - 1$.

- By gradedness, the hidden parents of hidden variable Y with rank $r = \rho(Y) \geq 1$ have rank $r - 1$
- The inference algorithm is similar to the Viterbi algorithm (below).
- Computation of array element $A[r, y]$ with rank r and $y \in D(r)$ requires only the array elements $A[r-1, y']$ with $y' \in D(r-1)$ of the previous rank.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Using R

Statistical Inference – Graded Case

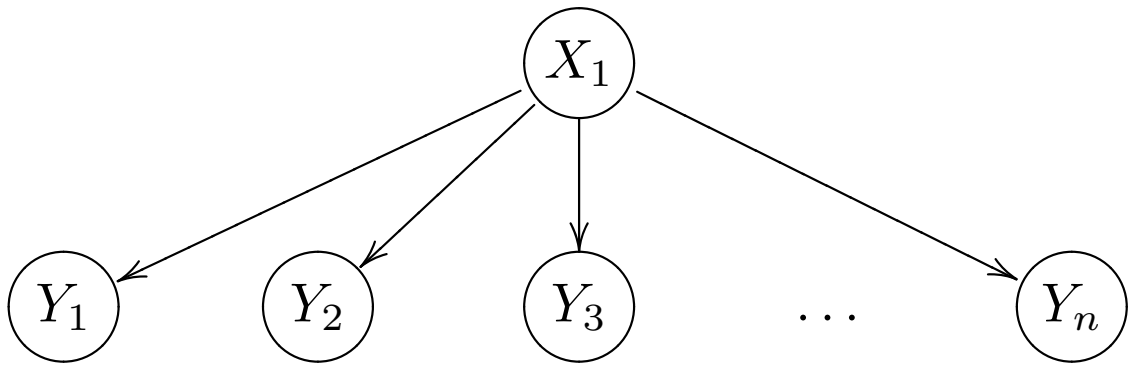
- The complexity of the evaluation of the tropicalized term $w_X(x)$ depends on the underlying DAG.
- The array A has size $\sum_{r=0}^{\rho_{\max}} |D(r)|$ and the computation of array element $A[r, y]$ requires $O(|D(r-1)| \cdot (s_r + t_r))$ steps.
- Suppose all state sets have l elements. Then we have $D(r) = l^{t_r}$ for all $0 \leq r \leq \rho_{\max}$.

Statistical Inference – Graded Case – Best Case

- All hidden random variables have the same rank and common observed ascendants ($\rho_{\max} = 0$).
- In the graded DAG below, $\rho(X_1) = \rho(Y_1) = \ldots = \rho(Y_n) = 0$.
- For each observed value $x_1 \in \mathcal{X}_1$,

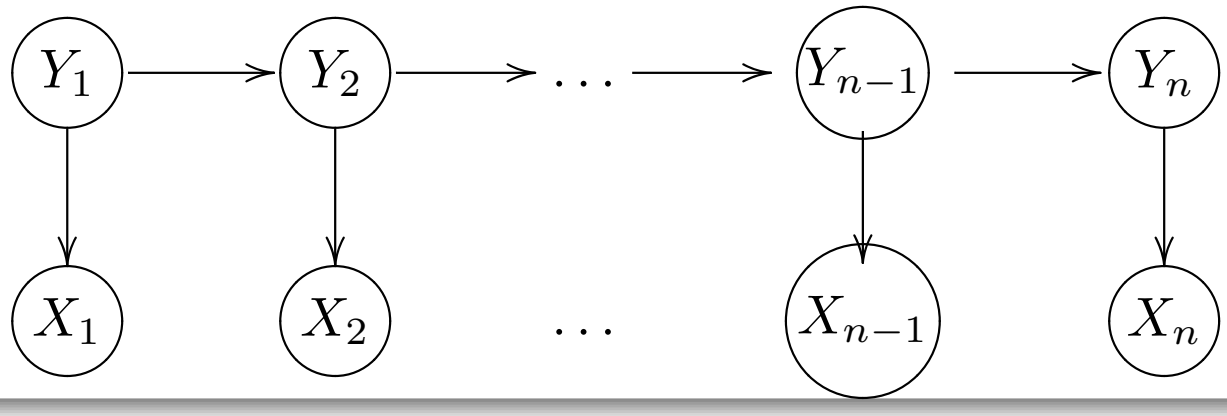
$$\begin{aligned} w_{X_1}(x_1) &= \min_{y_1,\ldots,y_n} \left(w_{Y_1|X_1}(y_1|x_1) + \ldots + w_{Y_n|X_1}(y_n|x_1) \right) \\ &= \min_{y_1} \left(w_{Y_1|X_1}(y_1|x_1) \right) + \ldots + \min_{y_n} \left(w_{Y_n|X_1}(y_n|x_1) \right) \end{aligned}$$

- Fully decoupled minimization has time complexity $O(ln)$.



Statistical Inference – Graded Case - HMM

- In the hidden Markov model (HMM), the hidden random variables form a chain ($\rho_{\max} = n - 1$ and $t_r = 1$ for each $0 \leq r \leq \rho_{\max}$).
- In the graded DAG below, $\rho(X_r) = \rho(Y_r) = r - 1$ for $1 \leq r \leq n$.



Statistical Inference – Graded Case - HMM (Cont'd)

- Viterbi algorithm calculates for each observed sequence $x = x_1 \dots x_n \in \mathcal{X}$ the following,

$$\begin{aligned}
A[0,y] &:= w_{X_1}(x_1) + w_{Y_1}(y), \\
A[1,y] &:= \min_{y_1} \left(w_{Y_2|Y_1}(y|y_1) + w_{X_2|Y_2}(x_2|y) \right. \\
&\quad \left. + A[0,y_1] \right) \\
&\quad \dots \\
A[n-1,y] &:= \min_{y_{n-1}} \left(w_{Y_n|Y_{n-1}}(y|y_{n-1}) + w_{X_n|Y_n}(x_n|y) \right. \\
&\quad \left. + A[n-2,y_{n-1}] \right) \\
w_X(x) &:= \min_{y_n} A[n-1,y_n].
\end{aligned}$$

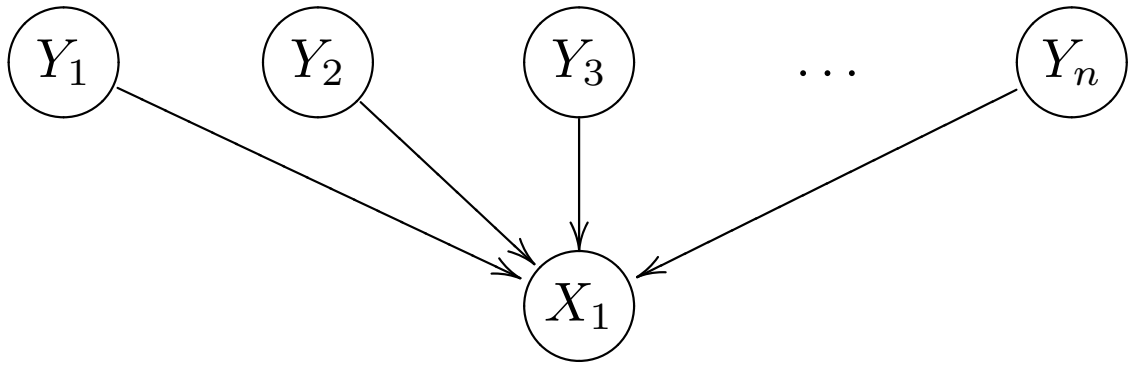
- Runtime $O(l^2 n)$, since the array has size $n \cdot l$ and the computation of each array element requires $O(l)$ steps.

Statistical Inference – Graded Case - Worst Case

- All hidden random variables have the same rank and common observed descendants ($\rho_{\max} = 0$).
- In the graded DAG below, $\rho(Y_1) = \ldots = \rho(Y_n) = \rho(X_1) = 0$.
- For each observed value $x = x_1 \in \mathcal{X}_1$,

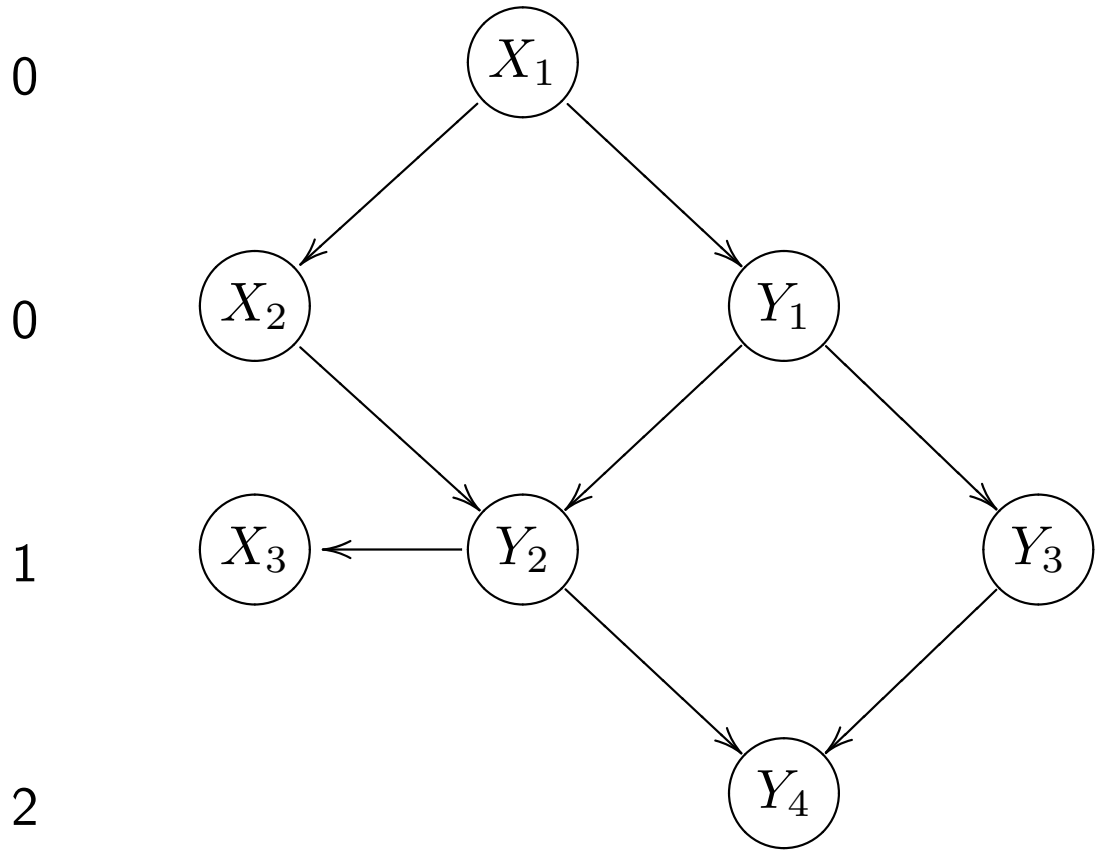
$$w_{X_1}(x_1) = \min_{y_1,\ldots,y_n} \left(w_{X_1|Y_1,\ldots,Y_n}(x_1|y_1,\ldots,y_n) + \sum_{i=1}^{n} w_{Y_i}(y_i) \right).$$

- Fully coupled minimization has time complexity $O(l^n n)$.



Statistical Inference – Example

Reconsider the graded belief network



Ranks: $\rho(X_1) = \rho(X_2) = \rho(Y_1) = 0$, $\rho(X_3) = \rho(Y_2) = \rho(Y_3) = 1$, and $\rho(Y_4) = 2$.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Using R

Statistical Inference – Example (Cont'd)

Suppose hidden variables $Y_1, \ldots, Y_4$ have common state set $\Sigma = \{a, b\}$. Then $D(0) = \Sigma$, $D(1) = \Sigma^2$, and $D(2) = \Sigma$.

$$
\begin{aligned}
A[0,a] &= w_{Y_1|X_1}(a|x_1) + w_{X_1}(x_1) + w_{X_2|X_1}(x_2|x_1),\\
A[0,b] &= w_{Y_1|X_1}(b|x_1) + w_{X_1}(x_1) + w_{X_2|X_1}(x_2|x_1),\\
A[1,aa] &= \min_{y_1 \in D(0)} \left(A[0,y_1] + w_{Y_2|X_2,Y_1}(a|x_2,y_1) + w_{Y_3|Y_1}(a|y_1) + w_{X_3|Y_2}(x_3|a) \right)\\
A[1,ab] &= \min_{y_1 \in D(0)} \left(A[0,y_1] + w_{Y_2|X_2,Y_1}(a|x_2,y_1) + w_{Y_3|Y_1}(b|y_1) + w_{X_3|Y_2}(x_3|a) \right)\\
A[1,ba] &= \min_{y_1 \in D(0)} \left(A[0,y_1] + w_{Y_2|X_2,Y_1}(b|x_2,y_1) + w_{Y_3|Y_1}(a|y_1) + w_{X_3|Y_2}(x_3|b) \right)\\
A[1,bb] &= \min_{y_1 \in D(0)} \left(A[0,y_1] + w_{Y_2|X_2,Y_1}(b|x_2,y_1) + w_{Y_3|Y_1}(b|y_1) + w_{X_3|Y_2}(x_3|b) \right)\\
A[2,a] &= \min_{y_2 y_3 \in D(1)} \left(A[1,y_2 y_3] + w_{Y_4|X_2,Y_3}(a|x_2,y_3) \right),\\
A[2,b] &= \min_{y_2 y_3 \in D(1)} \left(A[1,y_2 y_3] + w_{Y_4|X_2,Y_3}(b|x_2,y_3) \right).
\end{aligned}
$$

and $w_X(x_1,x_2,x_3) = \min_{y_4 \in D(2)} A[2,y_4] = \min\{A[2,a], A[2,b]\}$.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Structure Learning

- Score-based structure learning
- Constraint-based structure learning

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Structure Learning – Assumptions

- *Causal sufficiency:* There are no common unobserved (also known as hidden or latent) variables in the domain that are parents of one or more of the observed random variables in the domain.
- *Markov property:* Each random variable depends only on its parents, i.e., is independent of all its non-decendents given its parents.
- *Faithfulness:* All independence relations valid for the domain are those entailed by the (spatial) Markov property (21):

$$p_{X_1,\ldots,X_n} = \prod_{i=1}^{n} p_{X_i|\Pi_i}.$$

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Structure Learning – Approaches

- *Score-based methods* use heuristics to assign scores to the candidate belief networks in order to find a network with maximal score.
- *Constraint-based methods* learn the network structure by analyzing the probabilistic relations entailed by the Markov property via conditional independence tests and then aim to construct the network that fulfills the associated d-separation statements.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Score-Based Structure Learning

- Likelihood as network score
- Posterior probability as network score
- Hill climbing for structure learning

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Likelihood as Network Score

- Given DAG $G = (V, E)$ with node set $V = \{1, \ldots, n\}$ and edge set (structure) E.
- $\mathcal{E}$ is the set of all possible edge sets each of which providing a DAG with the node set V.
- Random variables $X_1, \ldots, X_n$ correspond 1-to-1 with node set V.
- Random variable X_i has state set $\mathcal{X}_i = \{x_1^{(i)}, \ldots, x_{m_i}^{(i)}\}$, $1 \leq i \leq n$.
- Random vector $X = (X_1, \ldots, X_n)$ has state set $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$.
- Given structure $E \in \mathcal{E}$, the set of instantiations of parent set Π_i of X_i is $\{\pi_1^{(i)}, \ldots, \pi_{l_i}^{(i)}\}$ with $l_i \geq 0$, $1 \leq i \leq n$.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Likelihood as Network Score

- Parameter set

$$\Theta = \{\theta = (\theta_{ijk}) \mid \theta_{ijk} \geq 0, \sum_j \theta_{ijk} = 1\}. \tag{325}$$

- Parameters (conditional probability that $X_i = x_j^{(i)}$ when $\Pi_i = \pi_k^{(i)}$),

$$\theta_{ijk} = p_{X_i|\Pi_i}(x_j^{(i)} \mid \pi_k^{(i)}), \tag{326}$$

where $1 \leq i \leq n$, $1 \leq j \leq m_i$, $1 \leq k \leq l_i$.

- For each pair (i, k) with $1 \leq i \leq n$ and $1 \leq k \leq l_i$, we have

$$\sum_{j=1}^{m_i} \theta_{ijk} = 1. \tag{327}$$

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Likelihood as Network Score

- Given collection of N independent samples (database):

$$D = (d_1, \ldots, d_N). \tag{328}$$

- $d_r = (x^{(1)}_{j_1,r}, \ldots, x^{(n)}_{j_n,r}) \in \mathcal{X}$ denotes the r-th sample.
- Maximum likelihood estimate of sample set D:

$$\hat{p}_{X|\mathcal{E}}(D \mid E) = \prod_{i=1}^{n} \prod_{j=1}^{m_i} \prod_{k=1}^{l_i} \hat{\theta}_{ijk}^{n_{ijk}}, \tag{329}$$

where n_{ijk} is the number of times the pair $(x_j^{(i)}, \pi_k^{(i)})$ appears in the sample set

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Likelihood as Network Score

- Maximum likelihood estimate (297) of the likelihood function $L(\theta, E)$ depending on the structure E,

$$\hat{\theta}_{ijk} = \frac{n_{ijk}}{\sum_{j=1}^{m_i} n_{ijk}}, \tag{330}$$

where $1 \leq i \leq n$, $1 \leq j < m_i$, $1 \leq k \leq l_i$.

- *Akaike information criterion* (AIC) score

$$\log p_{\mathcal{E}|X}(E \mid D) = \log \hat{p}_{X|\mathcal{E}}(D \mid E) - d, \tag{331}$$

where $d = \sum_{i=1}^{n}(m_i - 1)l_i$ by (292) is the number of free parameters in the local distribution functions.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Likelihood as Network Score

- *Bayesian information criterion* (BIC) score

$$\log p_{\mathcal{E}|X}(E \mid D) = \log \hat{p}_{X|\mathcal{E}}(D \mid E) - \frac{d}{2} \log N, \qquad (332)$$

  where $d = \sum_{i=1}^{n} (m_i - 1) l_i$ by (292) is the number of free parameters in the local distribution functions.

- In the language `R`, the AIC and BIC scores can be invoked by the keywords `aic` and `bic`, respectively.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Posterior Probability as Network Score

Let $A(n)$ denote the number of DAGs with n labeled nodes.

Put $A(0) = 1$. Then for each number $n \geq 1$,

$$A(n) = \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} A(n-i). \tag{333}$$

The problem of locating all possible graph structures is NP-hard. Therefore, the enumeration of all structures with n nodes is practically infeasible.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Proof.

Consider DAGs with node set $V = \{1, \ldots, n\}$. Let P_i be the set of DAGs with node set V in which the node i is a leaf; i.e, a node with degree 1. The number of DAGs with this node set which is in none of the P_i's is 0. Thus by the principle of inclusion-exclusion,

$$0 = A(n) - \left(\sum_{\substack{I \subseteq V \\ I \neq \emptyset}} (-1)^{|I|+1} \left| \bigcap_{j \in I} P_j \right| \right). \tag{334}$$

Claim that the cardinality of the set $\bigcap_{j \in I} P_j$ with $|I| = i$ is $2^{i(n-i)} A(n-i)$. Indeed, there are i leaves and these nodes can only be adjacent to the remaining $n - i$ nodes, making a total of $2^{i(n-i)}$ possible configurations of the edges. Moreover, the subgraph induced by the other $n - i$ nodes may be any of the $A(n-i)$ possible DAGs. This proves the claim. This number depends only on the size of the subset I of V. Since there are $\binom{n}{i}$ subsets of V with i elements, the result follows. □

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Posterior Probability as Network Score

- *Cooper-Herskovits likelihood* of graph structure $E \in \mathcal{E}$,

$$p_{X|\mathcal{E}}(D \mid E) = \prod_{i=1}^{n} \prod_{k=1}^{l_i} \frac{(m_i - 1)!}{(n_{ik} + m_i - 1)!} \prod_{j=1}^{m_i} n_{ijk}! \tag{335}$$

where n_{ik} is the number of occurrences of the pairs $(x_j^{(i)}, \pi_k^{(i)})$ with $1 \leq j \leq m_i$; i.e.,

$$n_{ik} = \sum_{j=1}^{m_i} n_{ijk}. \tag{336}$$

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Posterior Probability as Network Score

- By Bayes' rule, the posterior probability of graph structure E given data set D is

$$p_{\mathcal{E}|X}(E \mid D) = \frac{p_{X|\mathcal{E}}(D \mid E) \cdot p_{\mathcal{E}}(E)}{p_X(D)}. \tag{337}$$

- $p_{\mathcal{E}}(E)$ is the prior probability over the state space of the edge sets.
- $p_X(D)$ is the estimate of the database.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Posterior Probability as Network Score

- Score-based algorithms maximize posterior probability called *Bayesian Dirichlet (BD) score*,

$$p_{\mathcal{E}|X}(E \mid D) = \frac{p_{X|\mathcal{E}}(D \mid E) \cdot p_{\mathcal{E}}(E)}{p_X(D)}. \tag{338}$$

- Probability of database (by the law of total probability)

$$p_X(D) = \sum_{i=1}^{A(n)} p_{X|\mathcal{E}}(D \mid E_i) \cdot p_{\mathcal{E}}(E_i). \tag{339}$$

- Maximizing the numerator in (338) is sufficient, since D is fixed and the estimate $p_X(D)$ is independent of graph structure E.
- In the language `R`, the BD score is invoked by the keyword `bde`.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Posterior Probability as Network Score

Estimate $p_{\mathcal{E}}(E)$ using prior information.

- User assigns to each pair of random variables X_i and X_j a probability distribution with three values,

$$P(\{X_i \rightarrow X_j\}) + P(\{X_j \rightarrow X_i\}) + P(\{X_i \cdots X_j\}) = 1 \quad (340)$$

  where $X_i \rightarrow X_j$, $X_j \rightarrow X_i$, and $X_i \cdots X_j$ indicate that there is an edge $X_i \rightarrow X_j$, an edge $X_j \rightarrow X_i$, and no edge $X_i \cdots X_j$, respectively.

- For a pair X_i and X_j for which the user cannot specify a prior, the uniform distribution is assumed,

$$P(\{X_i \rightarrow X_j\}) = P(\{X_j \rightarrow X_i\}) = P(\{X_i \cdots X_j\}) = \frac{1}{3} \quad (341)$$

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Posterior Probability as Network Score

Estimate $p_{\mathcal{E}}(E)$ using prior information – modification in case of partial information.

- Suppose the user assigns to $P(\{X_i \to X_j\})$ the value q, $0 \leq q \leq 1$, but no values to $P(\{X_j \to X_i\})$ and $P(\{X_i \cdots X_j\})$. Then put

$$P(\{X_j \to X_i\}) = P(\{X_i \cdots X_j\}) = (1-q)/2. \tag{342}$$

- For a given structure E, the prior probabilities are obtained by multiplying the appropriate probabilities over all edges in E and normalizing to give $p_{\mathcal{E}}(E)$.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

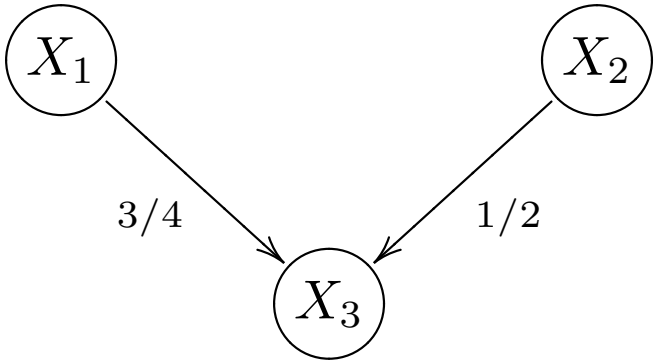
Score-based

Constraint-based

Using R

Calculation of Prior Information – Example

Given random variables X_1, X_2, X_3 with partial prior information



Completion to full prior:

	$P(\{X_i \to X_j\})$	$P(\{X_j \to X_i\})$	$P(\{X_i \cdots X_j\})$
X_1, X_2	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
X_1, X_3	$\frac{3}{4}$	$\frac{1}{8}$	$\frac{1}{8}$
X_2, X_3	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Heuristics in Optimization

- *Heuristic:* method for solving a problem when classical methods fail or are too slow; optimality, completeness, and accuracy are traded for speed.
- *Universe* is the set of all possible solutions of the given problem.
- *Local search* is a heuristic for tackling computationally hard optimization problems.
- Local search algorithms move from one solution to another until a solution suspected to be optimal is found or time has elapsed.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Hill Climbing

- Hill climbing is an iterative local search method.
- For structure learning, the set of all DAGs with n nodes is considered as the universe.
- The neighborhood of a DAG E is given by all DAGs which result from E by adding, deleting or reversing one edge. Beware of cycles!
- The language `R` has a built-in hill-climbing algorithm for structure learning called `hc`.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Hill Climbing

Require: Random variables $X_1, \ldots, X_n$, data set D, score σ, maximum number of iterations $T_{\max}$
Ensure: Edge set of DAG
  $E \leftarrow$ edge set of a DAG with n nodes {Initialization}
  $\sigma_{\max} \leftarrow \sigma(E)$
  $E_{\max} \leftarrow E$
  for t from 1 to $T_{\max}$ **do**
    for each DAG E' in the neighborhood of E **do**
      if $\sigma(E') > \sigma_{\max}$ **then**
        $\sigma_{\max} \leftarrow \sigma(E')$
        $E_{\max} \leftarrow E'$
      end if
    end for
    if $\sigma_{\max} > \sigma(E)$ **then**
      $E \leftarrow E_{\max}$ {Continue with edge set that has max. score in neighborhood}
    else
      return E {Exit if no edge set with better score in neighborhood found}
    end if
  end for
  return E

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Constraint-Based Structure Learning

- Conditional independence tests
- Pearson's chi-squared test
- SGS algorithm

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Conditional Independence Tests

- Mutual information
- Pearson's chi-squared test

Conditional independence tests for discrete data are functions of the conditional probability tables given by the structure of the network through the observed frequences

$$\{n_{ijk} \mid 1 \leq i \leq U, 1 \leq j \leq V, 1 \leq k \leq W\} \tag{343}$$

for random variables X and Y and all configurations of the conditioning variables S.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Mutual Information

- Test statistic

$$\mu(X, Y|S) = \sum_{i=1}^{U} \sum_{j=1}^{V} \sum_{k=1}^{W} \frac{n_{ijk}}{n} \log \frac{n_{ijk} n_{++k}}{n_{i+k} n_{+jk}}, \tag{344}$$

  where $n = n_{+++}$ is the sample size.

- In the language R, both the asymptotic chi-squared test (`mi`) and a Monte Carlo test (`mc-mi`) are available.

Pearson's chi-squared test

- Test statistic (for contingency tables)

$$\chi^2(X, Y|S) = \sum_{i=1}^{U}\sum_{j=1}^{V}\sum_{k=1}^{W}\frac{(n_{ijk} - m_{ijk})^2}{m_{ijk}} \tag{345}$$

with

$$m_{ijk} = \frac{n_{i+k}n_{+jk}}{n_{++k}}. \tag{346}$$

- In the language `R`, both the asymptotic chi-squared test (`x2`) and a Monte Carlo test (`mc-x2`) are available.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Conditional Independence Tests

- Null hypothesis: variables X and Y are statistically independent given S.
- Alternative hypothesis: variables have a relationship where the structure of relationship is not specified.
- Independence test for $S = \emptyset$: $(U-1)(V-1)$ is the number of degrees of freedom.
- Small p-value (typically ≤ 0.05) indicates strong evidence against the null hypothesis, larger values only weak evidence.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Conditional Independence Test in R

Consider the built-in data set `lizards`,

```
> library(bnlearn)
> data(lizards)
```

The data set contains three variables:

- `Species` (species of the lizard) is a two-level factor with levels `Sagrei` and `Distichus`.
- `Height` is a two-level factor with levels `high` (greater than 4.75 feet) and `low` (less or equal to 4.75 feet).
- `Diameter` is a two-level factor with levels `wide` (greater than 4 inches) and `narrow` (less or equal to 4 inches).

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Conditional Independence Test in R (Cont'd)

Visualization of data set

```
> lizards
      Species Diameter Height
1      Sagrei   narrow    low
2      Sagrei   narrow    low
...
100    Sagrei   narrow   high
...
409 Distichus     wide   high
```

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Conditional Independence Test in R (Cont'd)

Test conditional independence of two factors from the third one using the command `ci.test`.

```
>H = lizard$Height
>D = lizard$Diameter
>S = lizard$Species

> ci.test(H, D, S, data=lizards, test="x2") // asymptotic chi-squared test
          Pearson's X^2
data: H ~ D | S
x2 = 2.0174, df = 2, p-value = 0.3647

> ci.test(H, S, D, data=lizards, test="x2") // asymptotic chi-squared test
          Pearson's X^2
data: H ~ S | D
x2 = 11.617, df = 2, p-value = 0.003002

> ci.test(S, D, H, data=lizards, test="x2") // asymptotic chi-squared test
          Pearson's X^2
data: S ~ D | H
x2 = 13.781, df = 2, p-value = 0.001017
```

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

SGS Algorithm

SGS algorithm is named after the inventors Peter Spirtes, Clark Glymour, Richard Scheines (1993):

- The skeleton of the network given as undirected graph with the colletion of random variables as node set is estimated.

  The existence of an edge between two variables X and Y is tested by a number of independence tests. If faithfulness holds and there is an edge $X - Y$, then all independence tests should fail. Otherwise, there is a subset S d-separating them. This determines the undirected connectivity.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

SGS Algorithm (Cont'd)

- The directionality of the edges is determined.

  For this, triples of variables X, Y, and Z are considered such that $X - Z$ and $Y - Z$ are edges but $X - Y$ is not. If $X \not\perp Y \mid S$ for all $S = S' \cup \{Z\}$ with $S' \subseteq V \setminus \{X, Y, Z\}$, then no subset including Z can d-separate X and Y. Thus Z is a converging node and hence the directionality of the edges $X - Z$ and $Y - Z$ is set to $X \to Z$ and $Y \to Z$.

- The directions obtained in the second step are propagated while maintaining acyclicity.

For a belief network $X \to Y \to Z$, the direction of either edge cannot be determined by any set of independence statements, since two other networks with the same undirected structure, namely $X \leftarrow Y \leftarrow Z$ and $X \leftarrow Y \to Z$, belong to the same equivalence class with respect to the conditional independence statements (31).

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Score-based

Constraint-based

Using R

Drawbacks of constraint-based algorithms

- Lack of *robustness* in the sense that small changes of the input like single errors in the independence tests may have large effects on the output of the algorithm given by the structure of a belief network.
- The runtime of this type of algorithms is exponential in the number of variables of the domain, which makes them impractical for larger domains with tens or hundreds of variables.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

* Statistical Inference

Learning Structure

Using R

Bayesian Networks in R

- Learning the graph structure
- Learning the probability distribution

Structure Learning in R

Load library `bnlearn`:

```
> library(bnlearn)
```

Data set `learning.test` consists of 5000 six-tuples over the set $\{a, b, c\}^5 \times \{a, b\}$ described by six discrete variables $A, \ldots, F$, called categorial variables or factors, with two (F) or three levels $(A, \ldots, E)$.

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

* Statistical Inference

Learning Structure

Using R

Structure Learning in R

```
> data(learning.test)
> str(learning.test)
'data.frame':       5000 obs.  of 6 variables:
 $ A: Factor w/ 3 levels "a","b","c": 2 2 1 1 1 3 ...
 $ B: Factor w/ 3 levels "a","b","c": 3 1 1 1 1 3 ...
 $ C: Factor w/ 3 levels "a","b","c": 2 3 1 1 2 1 ...
 $ D: Factor w/ 3 levels "a","b","c": 1 1 1 1 3 3 ...
 $ E: Factor w/ 3 levels "a","b","c": 2 2 1 2 1 3 ...
 $ F: Factor w/ 2 levels "a","b": 2 2 1 2 1 1 1 2 ...
```

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

* Statistical Inference

Learning Structure

Using R

Structure Learning in R

The values of the factor A are

```
> learning.test$A
   [1] b b a a a c c b b b c c c b a a c b b c c c b a b
  [38] c c c c a ...
...
[4996] a c b a a
levels: a b c
```

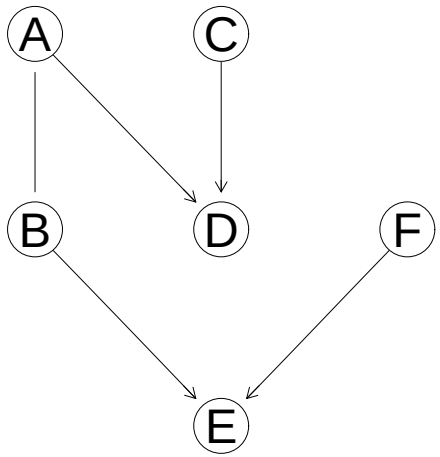
Structure Learning in R

Learning the structure of data set `learning.test` by GS algorithm (variant of SGS):

```
> bn.gs <- gs(learning.test)
> bn.gs
  Bayesian network learned via Constraint-based methods
  model:
    [partially directed graph]
  nodes:                                   6
  arcs:                                    5
    undirected arcs:                       1
    directed arcs:                         4
  average markov blanket size:             2.33
  average neighborhood size:               1.67
  average branching factor:                0.67
  learning algorithm:                      Grow-Shrink
  conditional independence test:           Mutual Information (discrete)
  alpha threshold:                         0.05
  tests used in the learning procedure:    43
  optimized:                               TRUE
```

Structure Learning in R

Resulting partially directed graph:



The undirected edge can be detected by `undirected.arc`

```
> undirected.arc(bn.gs)
     from to
[1,] "A"  "B"
[2,] "B"  "A"
```

Structure Learning in R

Transformation of partially directed graph into DAG using `set.arc` (setting arc direction) or `drop.arc` (dropping arc):

```
> bn.dag <- set.arc(bn.gs, "A", "B")
                              // set arc A -> B
> modelstring(bn.dag)
[1] "[A][C][F][B|A][D|A:C][E|B:F]"
```

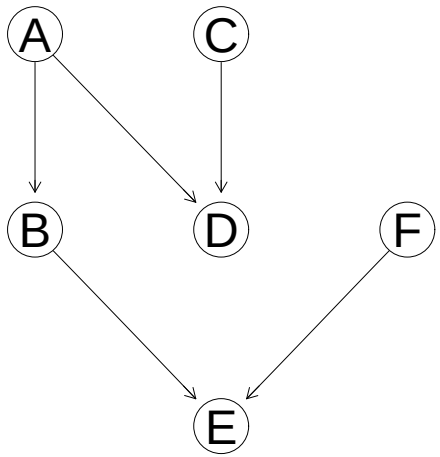
Structure Learning in R

Learning the structure of data set `learning.test` by hill-climbing algorithm:

```
> bn.hc <- hc(learning.test, score="aic")
> bn.hc
  Bayesian network learned via Score-based methods
  model:
    [A][C][F][B|A][D|A:C][E|B:F]
  nodes:                                 6
  arcs:                                  5
    undirected arcs:                     0
    directed arcs:                       5
  average markov blanket size:           2.33
  average neighborhood size:             1.67
  average branching factor:              0.83
  learning algorithm:                    Hill-Climbing
  conditional independence test:         Akaike Information Criterion
  penalization coefficient:              1
  test used in the learning procedure:   40
  optimized:                             TRUE
```

Structure Learning in R

Resulting DAG:



Compare structures by `compare`:

```
> compare(bn.gs, bn.hc)
[1] FALSE
> compare(bn.dag, bn.hc)
[1] TRUE
```

Structure Learning in R

Enter prior information on the domain by `blacklist` and `whitelist`.

- Arcs whitelisted are present, arcs blacklisted are absent.
- Any arc blacklisted and whitelisted is defined to be whitelisted and removed from the blacklist.
- Any arc whitelisted in both directions such as $A \rightarrow B$ and $B \rightarrow A$ is present in the learned structure, but the choice of direction is made by the learning algorithm.
- Any arc blacklisted in both directions such as $A \rightarrow B$ and $B \rightarrow A$ will not be present in the learned structure.
- Any arc whitelisted in one of two possible directions is guaranteed to be present in the learned structure.
- Any arc blacklisted in one of two possible directions will not be present in the learned structure.

Structure Learning in R

Prior information about the relationship between nodes can be whitelisted or blacklisted.

```
> bn.bl <- gs(learning.test, blacklist=c("B", "A"))
> compare(bn.bl, bn.hc)
[1] TRUE
```

Learning the Conditional Probabilities

Consider data set `learning.test` with network structure resulting from GS algorithm

```
> bn.ndag = gs(learning.test)
> bn.dag = set.arc(bn.ndag, from="A", to="B")
```

Maximum likelihood estimate by `bn.fit`

```
> mle.condprob = bn.fit(bn.dag, learning.test)
```

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Using R

Learning the Conditional Probabilities

```
> mle.condprob
   Bayesian network parameters

   Parameters of node A (multinomial distribution)
Conditional probability table:
      a      b      c
0.3336 0.3340 0.3324

   Parameters of node B (multinomial distribution)
Conditional probability table:
  A
B            a          b          c
  a 0.85611511 0.44491018 0.11492178
  b 0.02517986 0.22095808 0.09446450
  c 0.11870504 0.33413174 0.79061372

   Parameters of node C (multinomial distribution)
Conditional probability table:
      a      b      c
0.7434 0.2048 0.0518
```

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Using R

Learning the Conditional Probabilities

```
> mle.condprob
...
   Parameters of node D (multinomial distribution)
Conditional probability table:
, , C = a
  A
D            a          b          c
  a 0.80081301 0.09251810 0.10530547
  b 0.09024390 0.80209171 0.11173633
  c 0.10894309 0.10539019 0.78295820
, , C = b
  A
D            a          b          c
  a 0.18079096 0.88304094 0.24695122
  b 0.13276836 0.07017544 0.49490244
  c 0.68644068 0.04678363 0.25914634
, , C = c
  A
D            a          b          c
  a 0.42857143 0.34117647 0.13333333
  b 0.20238095 0.38823529 0.44444444
  c 0.36904762 0.27058824 0.42222222
```

Soft Computing

K.-H. Zimmermann

Contents

Learning Probabilities

*Statistical Inference

Learning Structure

Using R

Learning the Conditional Probabilities

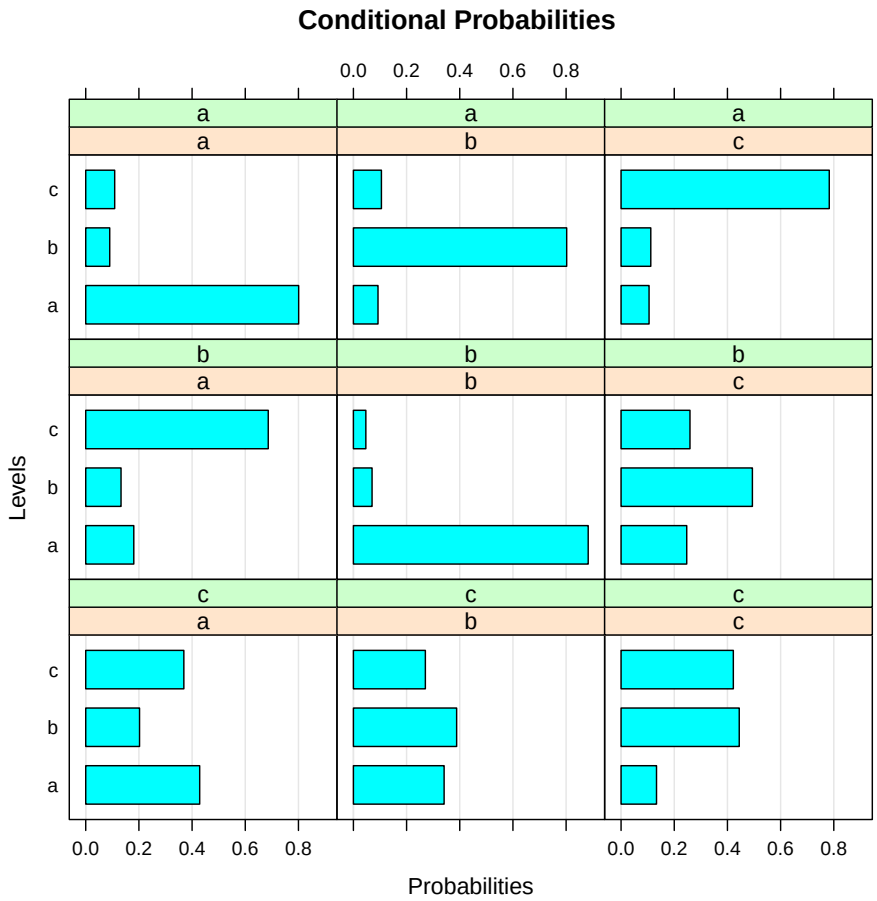
```
> mle.condprob
...
   Parameters of node E (multinomial distribution)
Conditional probability table:
, , F = a
  B
E            a          b          c
  a 0.80524979 0.20588235 0.11837378
  b 0.09737511 0.17973856 0.11448141
  c 0.09737511 0.61437908 0.76614481
, , F = b
  B
E            a          b          c
  a 0.40050804 0.31679389 0.23759542
  b 0.49026249 0.36641221 0.50667939
  c 0.10922947 0.31679389 0.25572519

   Parameters of node F (multinomial distribution)
Conditional probability table:
      a      b
0.5018 0.4982
```

Learning the Conditional Probabilities

Plot of conditional probability $p_{D|A,C}$ as barchart:

```
> bn.fit.barchart(mle.condprob$D)
```



Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Part VI

Artificial Neural Networks

Contents

- Introduction to artificial neural networks
- Multilayer perceptron
- Deep Learning with Keras
- Universal function representation and approximation
- Hopfield network
- Self-organizing maps
- *Spiking neural networks
- Neural networks in R

Pattern recognition uses features from the physiology of the brain – ANN as computational model.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Introduction to Artificial Neural Networks

- Perceptron
- Linear separability
- Perceptron learning algorithm
- Linear programming for perceptron learning

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

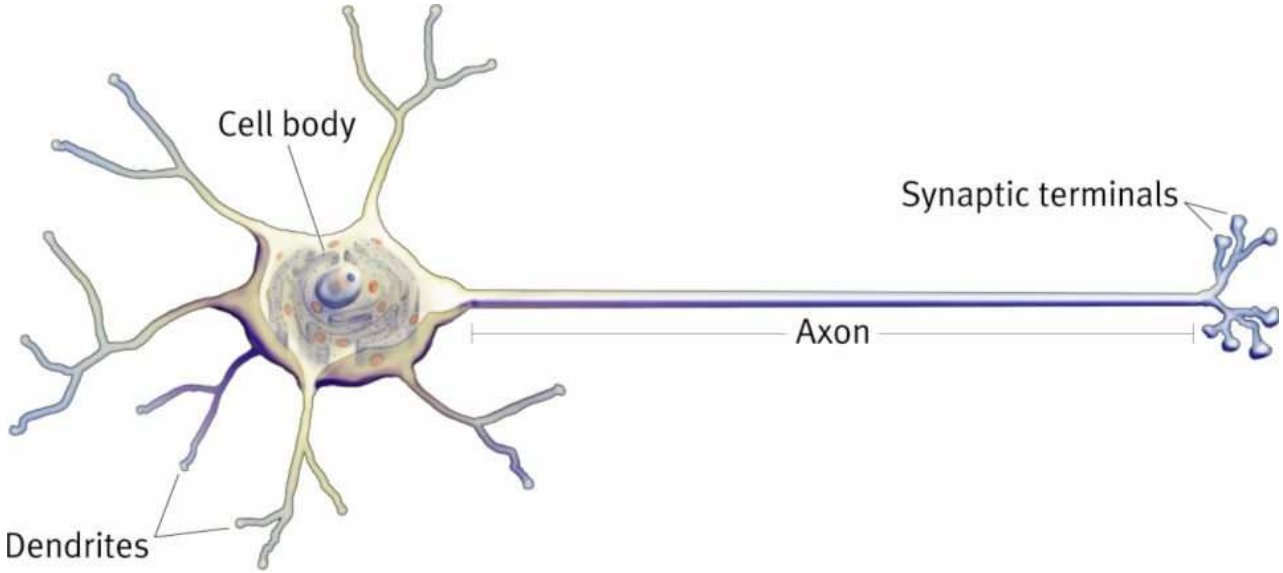
Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Artificial Neural Networks (ANNs)



- Neuron consists of cell body (soma), dendritic tree and single axon.
- Neurons receive signals via dendrites, soma sends signal (if threshold is reached) across the axon to other cells.
- Human brain has about $9 \cdot 10^{10}$ neurons.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Threshold Logic Unit (TLU)

First ANN as computing machine by Warren Mc Culloch and Walter Pitts (1943):

- Activity of neuron is all or nothing operation.
- Exitation of neuron if a certain number of synapses is excited within a period of time.
- Delay within the nervous system is only by synaptic delay.
- Activity of inhibitory synapse prevents excitation of neuron.
- Interconnection network is time invariant.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

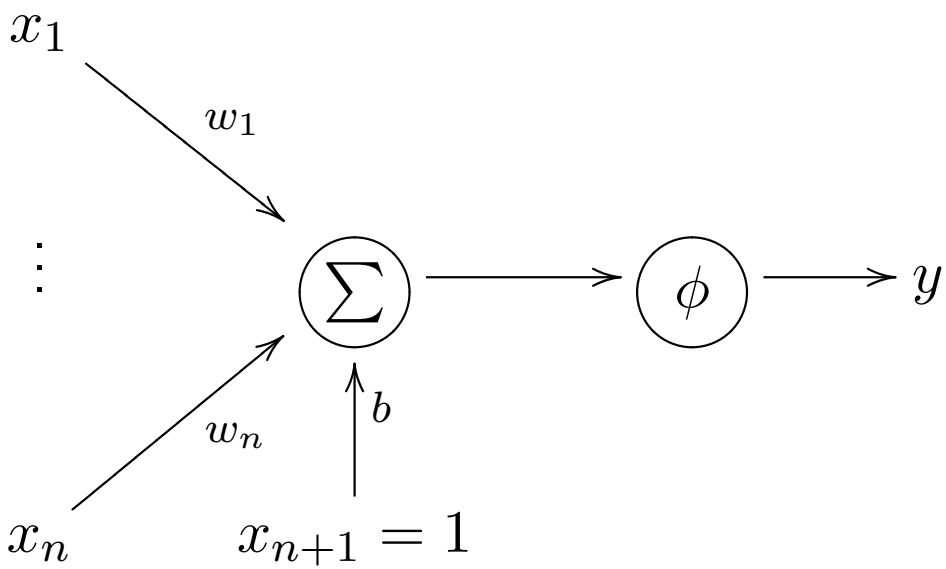
Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Mathematical Model of Artificial Neuron



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Mathematical Model of Artificial Neuron

- Input signals $x_1, \ldots, x_n$, bias $x_{n+1} = 1$
- Weights $w_1, \ldots, w_n$, $w_{n+1} = b$
- Linear combination of the input signals and weights

$$u = \sum_{i=1}^{n} w_i x_i. \tag{347}$$

- Addition of bias (affine linear combination) gives *activation potential* (hyperplane in $\mathbb{R}^n$)

$$h = u + b. \tag{348}$$

- Output of neuron using *activation function* ϕ,

$$y = \phi(h) = \phi\left(\sum_{i=1}^{n} w_i x_i + b\right). \tag{349}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Perceptron

- First algorithmically described ANN by Frank Rosenblatt (1958)
- Threshold function for artificial neuron:

$$y = \phi(h) = \begin{cases} +1 & \text{if } h \geq 0, \\ -1 & \text{otherwise.} \end{cases} \tag{350}$$

- Usage is classification of a collection of input signals $x_1, \ldots, x_n$ into one of two subclasses $\mathcal{A}$ and $\mathcal{B}$ of $\mathbb{R}^n$.
- Decision rule classifies real-valued point $x = (x_1, \ldots, x_n)$ to $\mathcal{A}$ if output $y = 1$, and $\mathcal{B}$ if $y = -1$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Definition

Let A, B be finite subsets of Euclidean space $\mathbb{R}^n$.

- A and B are *linearly separable* if there are real numbers $w_1, \ldots, w_n, w_{n+1}$ such that

$$\sum_{i=1}^{n} w_i a_i \geq w_{n+1} \quad \text{for each } (a_1, \ldots, a_n) \in A \tag{351}$$

and

$$\sum_{i=1}^{n} w_i b_i < w_{n+1} \quad \text{for each } (b_1, \ldots, b_n) \in B. \tag{352}$$

- A and B are *absolutely linearly separable* if they are linearly separable and the condition on the points $(a_1, \ldots, a_n) \in A$ is strengthened to

$$\sum_{i=1}^{n} w_i a_i > w_{n+1}. \tag{353}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

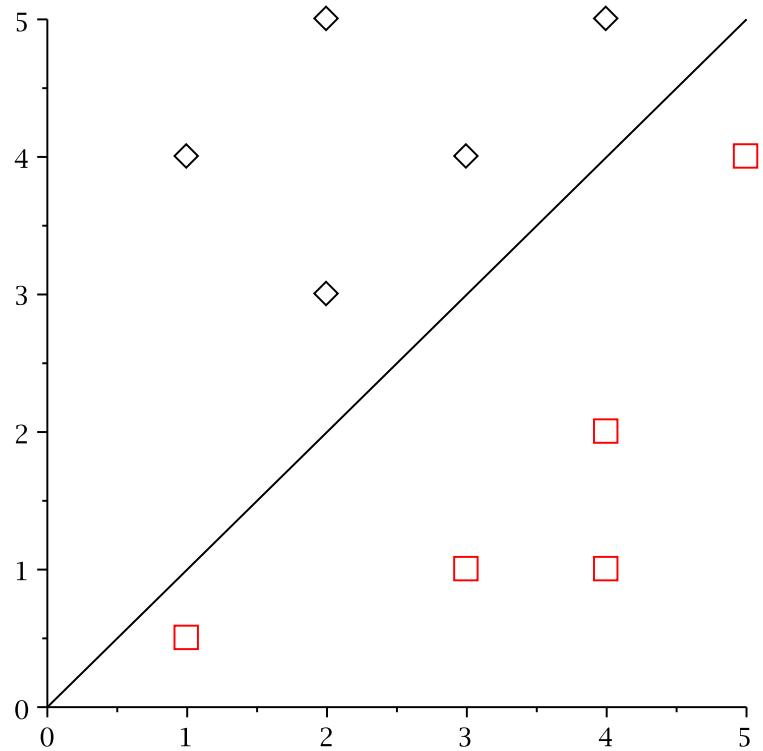
Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example

Two linearly separable sets in $\mathbb{R}^2$:



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Proposition

Two finite subsets A and B of $\mathbb{R}^n$ which are linearly separable are also absolutely linear separable.

Proof.

Suppose A and B are linearly separable. Then there are real numbers $w_1, \ldots, w_n, w_{n+1}$ such that $\sum_{i=1}^n w_i a_i \geq w_{n+1}$ for each point $(a_1, \ldots, a_n) \in A$ and $\sum_{i=1}^n w_i b_i < w_{n+1}$ for each point $(b_1, \ldots, b_n) \in B$. Put

$$\gamma = \max\{\sum_{i=1}^{n} w_i b_i - w_{n+1} \mid (b_1, \ldots, b_n) \in B\}.$$

Then $\gamma < 0$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Proof (cont'd).

Claim that replacing w_{n+1} by $w' = w_{n+1} + \gamma/2$ makes A and B absolutely linear separable.
Indeed, for each point $(a_1, \ldots, a_n) \in A$,

$$\sum_{i=1}^{n} w_i a_i - (w' - \gamma/2) = \sum_{i=1}^{n} w_i a_i - w_{n+1} \geq 0.$$

Thus $\sum_{i=1}^{n} w_i a_i - w' \geq -\gamma/2 > 0$ and hence $\sum_{i=1}^{n} w_i a_i > w'$.
Similarly, for each point $(b_1, \ldots, b_n) \in B$,

$$\sum_{i=1}^{n} w_i b_i - (w' - \gamma/2) = \sum_{i=1}^{n} w_i b_i - w_{n+1} \leq \gamma.$$

Thus $\sum_{i=1}^{n} w_i b_i - w' \leq \gamma/2 < 0$ and hence $\sum_{i=1}^{n} w_i b_i < w'$. □

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Perceptron Learning Algorithm

- Given two linearly separable classes $\mathcal{A}$ and $\mathcal{B}$ in $\mathbb{R}^n$.
- Input: two finite training sets A and B of $\mathcal{A}$ and $\mathcal{B}$, resp.
- Find extended weight vector $w = (w_1, \ldots, w_n, w_{n+1})$ with $w_{n+1} = -\theta$ such that for each extended input $x = (x_1, \ldots, x_n, 1)$,

$$\langle w, x \rangle > 0 \quad \text{if } x \in \mathcal{A} \tag{354}$$

and

$$\langle w, x \rangle < 0 \quad \text{if } x \in \mathcal{B}. \tag{355}$$

Supervised learning from labeled training data!

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Perceptron Learning Algorithm

Require: Finite linearly separable subsets A and B of $\mathbb{R}^n$
Ensure: Weight vector w for linear separability
  Choose randomly a weight vector w_0
  $w \leftarrow w_0$
  repeat
    Select randomly a vector $x \in A \cup B$
    if $x \in A$ and $\langle w, x \rangle \leq 0$ **then**
      $w \leftarrow w + x$
    else if $x \in B$ and $\langle w, x \rangle \geq 0$ **then**
      $w \leftarrow w - x$
    end if
  until All vectors in $A \cup B$ are classified correctly
  return w

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Proposition

Let A and B be finite and linearly separable subsets of $\mathbb{R}^n$. The perceptron learning algorithm with input A, B provides a weight vector w after a finite number of steps which absolutely separates the two sets.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Proof.

- The sets A and B are joined into a single set $C = A \cup B^-$, where $B^- = \{-b \mid b \in B\}$ consists of the negated elements of B.
- The vectors in C can be normalized, since if a weight vector w satisfies $\langle w, x\rangle > 0$ for all $x \in C$, then $\langle w, \eta x\rangle = \eta\langle w, x\rangle > 0$ for all $x \in C$ and any constant $\eta > 0$.
- The weight vector can be normalized by the same argument and can be chosen to absolutely separate the sets A and B. This vector is denoted by w^*.

Suppose after iteration $t+1$ of the percetron learning algorithm the weight vector w_{t+1} has been computed, and that at time t the vector $x \in C$ has been incorrectly classified by the weight vector w_t. Thus we have $w_{t+1} = w_t + x$. Consider the cosine of the angle between the vectors w_{t+1} and w^*,

$$\cos\beta = \frac{\langle w^*, w_{t+1}\rangle}{\|w_{t+1}\|}.$$

We have

$$\langle w^*, w_{t+1}\rangle = \langle w^*, w_t + x\rangle = \langle w^*, w_t\rangle + \langle w^*, x\rangle \geq \langle w^*, w_t\rangle + \epsilon,$$

where $\epsilon = \min\{\langle w^*, c\rangle \mid c \in C\}$. Since the weight vector w^* provides an absolute separation of A and B, we have $\epsilon > 0$. By induction, we obtain

$$\langle w^*, w_{t+1}\rangle \geq \langle w^*, w_0\rangle + (t+1)\epsilon.$$

Moreover,

$$\|w_{t+1}\|^2 = \langle w_t + x, w_t + x\rangle = \|w_t\|^2 + 2\langle w_t, x\rangle + \|x\|^2.$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Proof (cont'd).

But we have corrected w_t using the input x and so $\langle w_t, x\rangle \leq 0$. Thus by hypothesis, we have

$$\|w_{t+1}\|^2 \leq \|w_t\|^2 + \|x\|^2 \leq \|w_t\|^2 + 1.$$

By induction, we obtain

$$\|w_{t+1}\|^2 \leq \|w_0\|^2 + (t+1).$$

Therefore,

$$\cos\beta = \frac{\langle w^*, w_{t+1}\rangle}{\|w_{t+1}\|} \geq \frac{\langle w^*, w_0\rangle + (t+1)\epsilon}{\sqrt{\|w_0\|^2 + (t+1)}}$$

When t becomes large, the term on the right-hand side grows proportionally to $\sqrt{t}$. Since ϵ is positive, the term can become arbitrarily large. But $\cos\beta \leq 1$ and so the parameter t must be bounded from above. Hence, the number of corrections to the weight vector must be finite. □

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

*Accelerated Perceptron Learning Algorithm

Require: Finite linearly separable subsets A and B of $\mathbb{R}^n$
Ensure: Weight vector w for linear separability
  Choose randomly a weight vector w_0
  $w \leftarrow w_0$
  repeat
    Select randomly a vector $x \in A \cup B$
    if $x \in A$ and $\langle w, x \rangle \leq 0$ **then**
      $\delta \leftarrow -\langle w, x \rangle$
      Choose a small number $\epsilon > 0$
      $w \leftarrow w + \frac{\delta+\epsilon}{\|x\|^2} x$
    else if $x \in B$ and $\langle w, x \rangle \geq 0$ **then**
      $\delta \leftarrow -\langle w, x \rangle$
      Choose a small number $\epsilon > 0$
      $w \leftarrow w + \frac{\delta-\epsilon}{\|x\|^2} x$
    end if
  until All vectors in $A \cup B$ are classified correctly
  return w

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Proposition

Let A and B be finite and linearly separable subsets of $\mathbb{R}^n$. The accelerated perceptron learning algorithm with input A, B provides a weight vector w after a finite number of steps which absolutely separates the two sets.

Proof.

Similar to perceptron learning algorithm. □

Corrective Learning

The weight vector is not just reinforced, but completely corrects the error committed.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example

Take a Boolean function $f : \{0,1\}^n \to \{0,1\}$ with arity $n \geq 1$.

- The *fibres* of f are

$$f^{-1}(0) = \{x \in \{0,1\}^n \mid f(x) = 0\}$$

and

$$f^{-1}(1) = \{x \in \{0,1\}^n \mid f(x) = 1\}.$$

Then

$$\{0,1\}^n = f^{-1}(0) \cup f^{-1}(1).$$

- For which Boolean function f are the associated fibres linearly separable?

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example (cont'd)

In view of Boolean AND function, the fibres

$$f^{-1}(0) = \{(0,0),(0,1),(1,0)\} \quad \text{and} \quad f^{-1}(1) = \{(1,1)\}$$

are linearly separable.

- Take training sets $A = \{(0,0,1),(0,1,1),(1,0,1)\}$ and $B = \{(1,1,1)\}$.
- Start with weight vector $w = (0,0,0)$.
- The perceptron learning algorithm outputs (after 8 steps) weight vector $w = (-3,-2,4)$ or normalized vector $(-0.56,-0.37,0.74)$.
- We have

$$\begin{aligned} \langle (0,0,1), w\rangle &= 4, \quad \langle (0,1,1), w\rangle = 2, \\ \langle (1,0,1), w\rangle &= 3, \quad \langle (1,1,1), w\rangle = -1. \end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example (cont'd)

Maple code:

```
> with(LinearAlgebra):
> x1:=Vector([0,0,1]): x2:=Vector([0,1,1]):
> x3:=Vector([1,0,1]): x4:=Vector([1,1,1]):
> w:=Vector([0,0,0]):
> bval:=TRUE: i:=0:
> while (bval = TRUE) do
>   bval:=FALSE: i:=i+1:
>   if DotProduct(x1,w)<=0 then w:=w+x1; bval:=TRUE fi;
>   if DotProduct(x2,w)<=0 then w:=w+x2; bval:=TRUE fi;
>   if DotProduct(x3,w)<=0 then w:=w+x3; bval:=TRUE fi;
>   if DotProduct(x4,w)>=0 then w:=w-x4; bval:=TRUE fi;
> end;
> print(i); print(w);
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

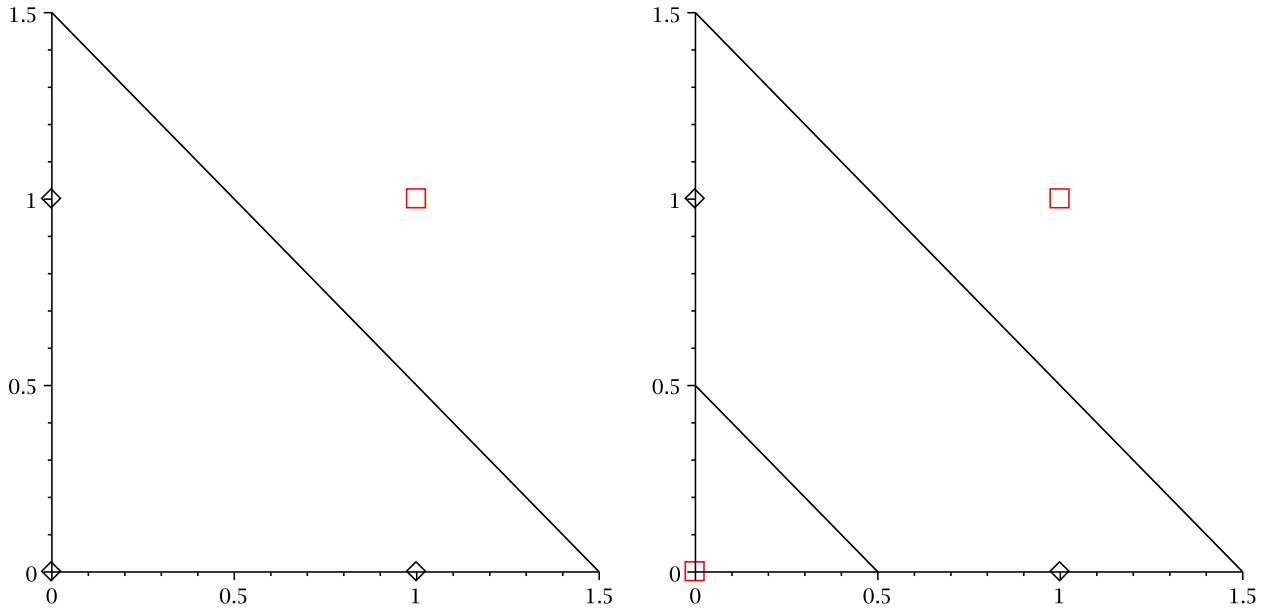
*Spiking Neural Networks

Neural Networks in R

Example (cont'd)

The Boolean XOR function has the fibres

$$f^{-1}(0) = \{(0,0),(1,1)\} \quad \text{and} \quad f^{-1}(1) = \{(0,1),(1,0)\}.$$



AND is linearly separable, but XOR is not!

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Linear Programming for Perceptron Learning

Given training sets $A = \{a_1, \ldots, a_k\}$ and $B = \{b_1, \ldots, b_l\}$ of classes $\mathcal{A}$ and $\mathcal{B}$ in $\mathbb{R}^n$, respectively.
Define the real-valued $(k+l) \times n$ matrix

$$A' = \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ a_{21} & \ldots & a_{2n} \\ \vdots & \ldots & \vdots \\ a_{k1} & \ldots & a_{kn} \\ -b_{11} & \ldots & -b_{1n} \\ -b_{21} & \ldots & -b_{2n} \\ \vdots & \ldots & \vdots \\ -b_{l1} & \ldots & -b_{ln} \end{pmatrix}. \tag{356}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Linear Programming for Perceptron Learning

Take small number $\epsilon > 0$, put $e = (\epsilon, \ldots, \epsilon)^t$.
Consider the linear program

$$\begin{array}{ll} \min & w_1 + \ldots + w_n, \\ \text{s.t.} & A' \cdot w \geq e \end{array} \tag{357}$$

The set of feasible solutions $\{w \in \mathbb{R}^n \mid A' \cdot w \geq e\}$ provides a hyperplane that linearly separates the training sets.

Proposition

The linear program is solvable iff the training sets A and B are absolutely separable.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example

For the Boolean AND function, the linear program is

$$\begin{aligned} \min \quad & w_1 + w_2 + w_3. \\ \text{s.t.} \quad & w_3 \geq \epsilon \\ & w_2 + w_3 \geq \epsilon \\ & w_1 + w_3 \geq \epsilon \\ & -w_1 - w_2 - w_3 \geq \epsilon \end{aligned}$$

Maple code

```
> with(Optimization):
> e := 0.01:
> LPSolve(w1+w2+w3, {w3>=e, w2+w3>=e, w1+w3>=e,
          -w1-w2-w3>=e});
```

gives solution $w_1 = -0.02$, $w_2 = -0.02$, $w_3 = 0.03$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example

For the Boolean XOR function, the linear program is

$$\begin{aligned} \min \quad & w_1 + w_2 + w_3. \\ \text{s.t.} \quad & -w_3 \geq \epsilon \\ & w_2 + w_3 \geq \epsilon \\ & w_1 + w_3 \geq \epsilon \\ & -w_1 - w_2 - w_3 \geq \epsilon \end{aligned}$$

Maple code

```
> with(Optimization):
> e := 0.01:
> LPSolve(w1+w2+w3, {-w3>=e, w2+w3>=e, w1+w3>=e,
          -w1-w2-w3>=e});
```

returns message that no feasible solution exists.

Multilayer Perceptron (MLP)

- Structure of MLP
- Training of MLP
- Backpropagation algorithm
- Backpropagation in R
- MNist

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Multilayer Perceptron (MLP)

- Layered feedforward artificial neural network with input and output.
- Directed graph with sequence of layers of nodes, each layer is fully connected to the successive layer.
- Apart from input nodes, nodes are perceptrons with nonlinear activation function.
- Three or more layers: input layer, one or more hidden layers, and output layer.
- An n layered MLP has input layer, $n-1$ hidden layers, and output layer.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

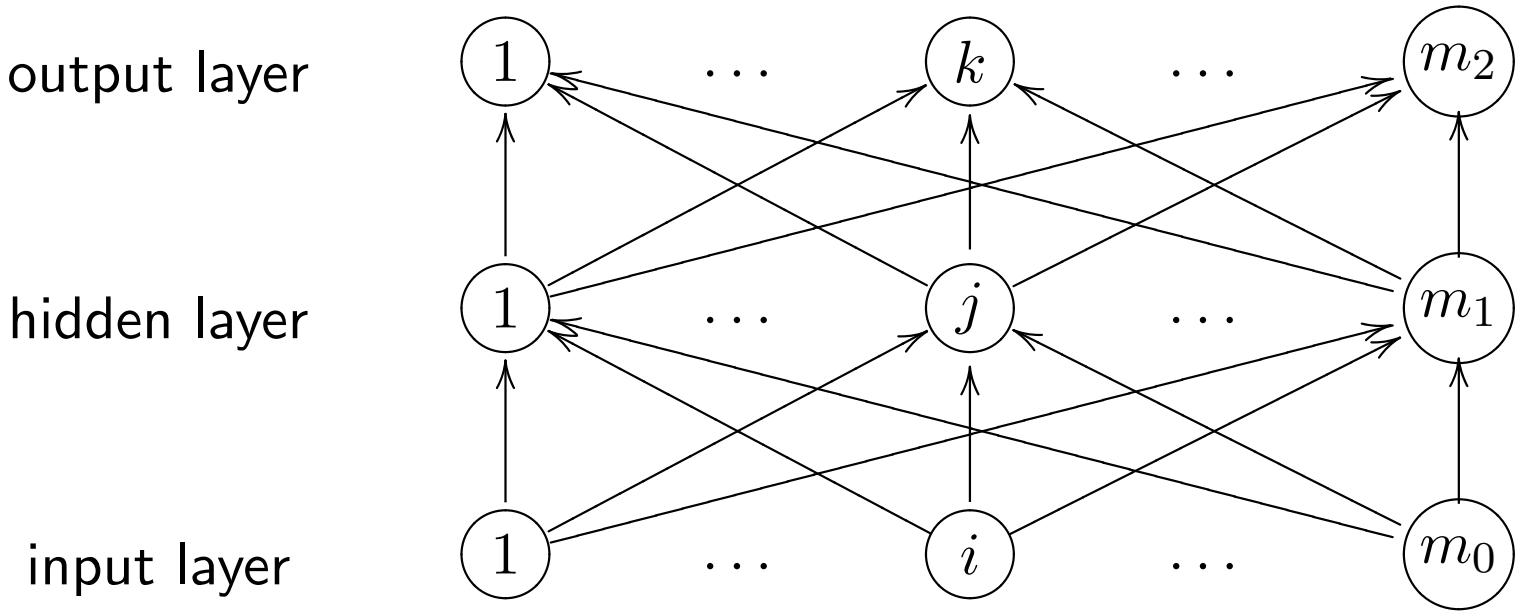
Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example

Two-layered MLP:



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Definition

- MLP has $n \geq 2$ layers with l-th layer having m_l units (neurons), $1 \leq l \leq n$.
- $\text{in}(i) =$ value of i-th input neuron.
- $w_{ij}^{(l)} =$ weight between i-th unit of layer $l-1$ and j-th unit of layer l.
- The j-th unit of layer l computes the value

$$s_j^{(l)} = f^{(l)} \left(\sum_i s_i^{(l-1)} w_{ij}^{(l)} \right). \tag{358}$$

where $f^{(l)}$ denotes an activation function; let's assume a common activation function for all units.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

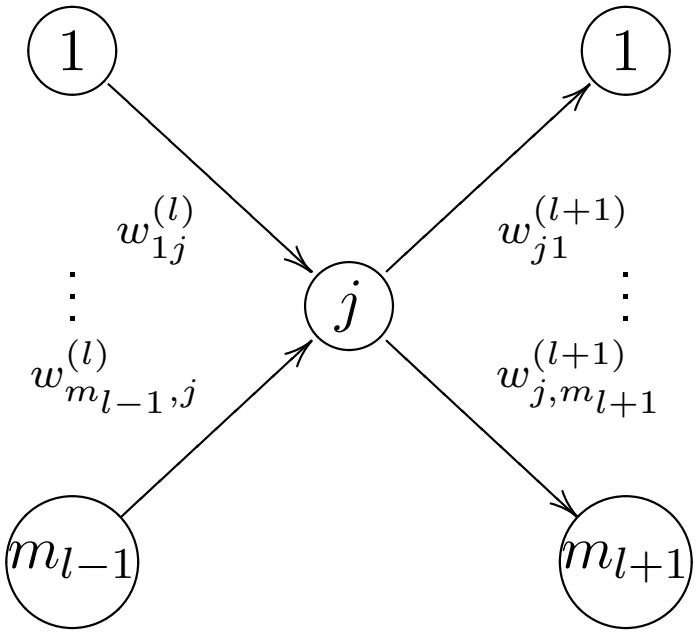
Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Definition

Notation for j-th neuron in l-th layer of MLP:



$$s_j^{(l)} = f^{(l)}\left(\sum_i s_i^{(l-1)} w_{ij}^{(l)}\right).$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Two-Layered MLP

Layered computation:

$$s_k^{(2)} = f^{(2)}\left(\sum_j s_j^{(1)} w_{jk}^{(2)}\right), \tag{359}$$

$$s_j^{(1)} = f^{(1)}\left(\sum_i s_i^{(0)} w_{ij}^{(1)}\right), \tag{360}$$

$$s_i^{(0)} = \mathrm{in}(i). \tag{361}$$

Network output:

$$\begin{aligned} s_k^{(2)} &= f^{(2)}\left(\sum_j s_j^{(1)} w_{jk}^{(2)}\right) \\ &= f^{(2)}\left(\sum_j f^{(1)}\left(\sum_i s_i^{(0)} w_{ij}^{(1)}\right) w_{jk}^{(2)}\right). \end{aligned} \tag{362}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

S-shaped Activation Functions

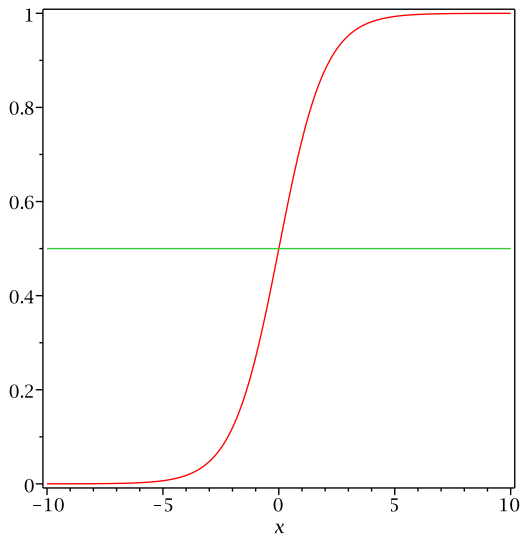
- *Sigmoid function:*

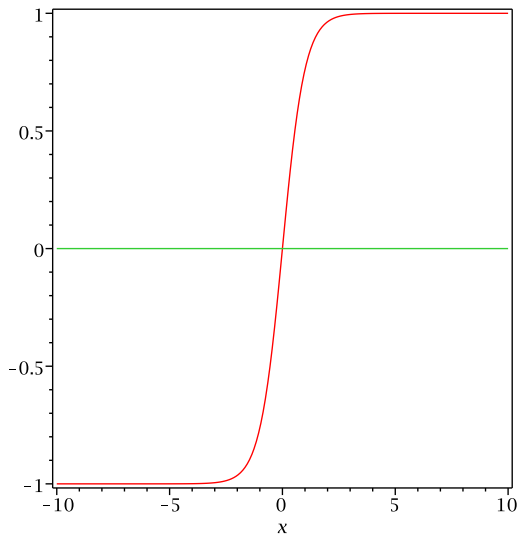
$$\mathrm{sig}(x) = \frac{1}{1+e^{-x}}, \quad x \in \mathbb{R}. \tag{363}$$

Bounded, differentiable, positive derivative at each point.

- *Hyperbolic tangent:*

$$\tanh(x) = 2 \cdot \mathrm{sig}(2x) - 1, \quad x \in \mathbb{R}. \tag{364}$$





Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Training of MLP

1. Take training patterns $x^{(1)}, \ldots, x^{(M)}$ in $\mathbb{R}^m$.
2. Set up MLP with $n \geq 2$ layers.
3. Generate random initial weights from certain range.
4. Select error function E depending on the weights; take error bound $\epsilon > 0$ and learning rate $\eta > 0$.
5. Update weights according to the negative of the gradient of the error function,

$$\Delta w_{ab}^{(l)} = -\eta \frac{\partial E}{\partial w_{ab}^{(l)}}. \tag{365}$$

   One set of updates of all the weights for all training patterns is called an *epoch* of training.
6. Repeat weight update (step 5) until error is $< \epsilon$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Training of MLP II

- $y^{(p)}$ = desired output for training pattern $x^{(p)}$.
- $s_i^{(n,p)}$ = actual output in output layer for training pattern $x^{(p)}$.
- $s_i^{(l,p)} = s_i^{(l)}$ indicates dependence on the p-th training pattern.
- Adjust the network weights in order to minimize the *error function*

$$E = \frac{1}{2} \sum_{p=1}^{M} \sum_{j=1}^{m_n} \left(y_j^{(p)} - s_j^{(n,p)} \right)^2. \tag{366}$$

$E = E(w)$ is a multivariate function of the network weights $w_{ij}^{(l)}$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Training of MLP III

- Method for minimizing the error function is the local method of *gradient descent*.
- Multivariate function $f(x)$ differentiable in neighborhood of point p decreases fastest if travelling from p in direction of negative gradient of f at p:

$$p' = p - \eta \nabla f(p) \tag{367}$$

for some small number $\eta > 0$. Then $f(p) \geq f(p')$.

- Start with initial point $p^{(0)}$ and take sequence of points $(p^{(k)})_{k \geq 0}$ such that

$$p^{(k+1)} = p^{(k)} - \eta \nabla f(p^{(k)}), \quad k \geq 0. \tag{368}$$

Then the sequence of points $(p^{(k)})_{k \geq 0}$ may converge to local minimum,

$$f(p^{(0)}) \geq f(p^{(1)}) \geq f(p^{(2)}) \geq \ldots. \tag{369}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Training of Two-layered MLP

Layered computation ($f^{(2)} =$ identity function):

$$s_k^{(2)} = \sum_j s_j^{(1)} w_{jk}^{(2)} = \sum_j f^{(1)} \left(\sum_i s_i^{(0)} w_{ij}^{(1)} \right) w_{jk}^{(2)}, \quad (370)$$

$$s_j^{(1)} = f^{(1)} \left(\sum_i s_i^{(0)} w_{ij}^{(1)} \right), \quad (371)$$

$$s_i^{(0)} = \text{in}(i). \quad (372)$$

Partial derivative of error function w.r.t. weight $w_{ab}^{(l)}$:

$$\frac{\partial E}{\partial w_{ab}^{(l)}} = -\sum_p \sum_k \left(y_k^{(p)} - s_k^{(2,p)} \right) \cdot \frac{\partial s_k^{(2,p)}}{\partial w_{ab}^{(l)}}. \quad (373)$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Training of Two-layered MLP II

Difference quotient (δ = Kronecker delta):

$$\frac{\partial s_k^{(2)}}{\partial w_{ab}^{(2)}} = \sum_j s_j^{(1)} \frac{\partial w_{jk}^{(2)}}{\partial w_{ab}^{(2)}} = \sum_j s_j^{(1)} \delta_{ja} \delta_{kb} = s_a^{(1)} \delta_{kb} \tag{374}$$

and ($f = f^{(1)}$)

$$\begin{aligned} \frac{\partial s_k^{(2)}}{\partial w_{ab}^{(1)}} &= \sum_j f' \left(\sum_i s_i^{(0)} w_{ij}^{(1)} \right) \cdot \left(\sum_i s_i^{(0)} \frac{\partial w_{ij}^{(1)}}{\partial w_{ab}^{(1)}} \right) \cdot w_{jk}^{(2)} \\ &= f' \left(\sum_i s_i^{(0)} w_{ib}^{(1)} \right) \cdot s_a^{(0)} w_{bk}^{(2)} . \end{aligned} \tag{375}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Training of Two-layered MLP III

Substitute difference quotients back into error function:

$$\begin{aligned}\Delta w_{ab}^{(2)} &= -\eta \frac{\partial E}{\partial w_{ab}^{(2)}} \qquad (376)\\ &= \eta \sum_p \sum_k \left(y_k^{(p)} - s_k^{(2,p)} \right) \cdot s_a^{(1,p)} \delta_{kb}, \text{ by (373,374)},\\ &= \eta \sum_p \left(y_b^{(p)} - s_b^{(2,p)} \right) \cdot s_a^{(1,p)},\end{aligned}$$

$$\begin{aligned}\Delta w_{ab}^{(1)} &= -\eta \frac{\partial E}{\partial w_{ab}^{(1)}} \qquad (377)\\ &= \eta \sum_p \sum_k \left(y_k^{(p)} - s_k^{(2,p)} \right) \cdot f' \left(\sum_i s_i^{(0,p)} w_{ib}^{(1)} \right) \cdot s_a^{(0,p)} w_{bk}^{(2)},\end{aligned}$$

using (373,375).

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Training of Two-layered MLP IV

Take as activation function the sigmoid function

$$f(x) = 1/(1 + e^{-x}). \tag{378}$$

Then

$$f'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = f(x) \cdot (1 - f(x)). \tag{379}$$

Thus by (376) (activiation function = identity function)

$$\Delta w_{ab}^{(2)} = \eta \sum_p \left(y_b^{(p)} - s_b^{(2,p)} \right) \cdot s_a^{(1,p)} \tag{380}$$

and by (371,377)

$$\Delta w_{ab}^{(1)} = \tag{381}$$
$$\eta \sum_p \sum_k \left(y_k^{(p)} - s_k^{(2,p)} \right) f(s_b^{(1,p)}) f\left(1 - s_b^{(1,p)}\right) s_a^{(0,p)} w_{bk}^{(2)}.$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Training of Two-layered MLP V

Define *output error* between desired and actual output:

$$\delta_b^{(2)} = y_b - s_b^{(2)}, \quad 1 \leq b \leq m_2. \tag{382}$$

Then the updating rules have the form by (380)

$$\Delta w_{ab}^{(2)} = \eta \sum_p \delta_b^{(2,p)} \cdot s_a^{(1,p)} \tag{383}$$

and by (381)

$$\Delta w_{ab}^{(1)} = \tag{384}$$
$$\eta \sum_p \left(\sum_k \delta_k^{(2,p)} w_{bk}^{(2)} \right) f(s_b^{(1,p)}) f\left(1 - s_b^{(1,p)}\right) \cdot s_a^{(0,p)}.$$

The output error at unit b is back-propagated from output unit k by the weight $w_{bk}^{(2)}$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

General MLP

Extend update rule to MLPs with $n \geq 2$ layers.

- *Output error* between desired and actual output:

$$\delta_b^{(n)} = y_b - s_b^{(n)}, \quad 1 \leq b \leq m_n. \tag{385}$$

- Error is propagated back successively to the preceeding layers:

$$\delta_b^{(l)} = \left(\sum_k \delta_k^{(l+1)} w_{bk}^{(l+1)} \right) \cdot f' \left(\sum_j s_j^{(l-1)} w_{jb}^{(l)} \right). \tag{386}$$

- Weight updates:

$$\Delta w_{ab}^{(l)} = \eta \sum_p \delta_b^{(l,p)} s_a^{(l-1,p)}. \tag{387}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Backpropagation algorithm

Require: MLP with $n \geq 2$ layers, training patterns $x^{(1)}, \ldots, x^{(M)}$ in $\mathbb{R}^m$ with desired outputs $y^{(1)}, \ldots, y^{(M)}$, error function $E = E(w)$, small numbers $\epsilon > 0$ and $\eta > 0$

Ensure: Trained MLP with total error $E(w) < \epsilon$

Choose random weights $w_{ab}^{(l)}$

Compute output $s^{(n,p)}$ for each pattern p by (358)

Compute error function $E(w)$ by (366)

while $E(w) \geq \epsilon$ **do**

Compute weight updates $\Delta w_{ab}^{(l)}$ by (387)

Compute new weights $w_{ab}^{(l)}$ by (367)

Compute output $s^{(n,p)}$ for each pattern p by (358)

Compute error function $E(w)$ by (366)

end while

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Backpropagation in R

Library `neuralnet` for training of neural networks:

```
> library(neuralnet)
```

Generate MLP for Boolean XOR function:

```
> XOR <- c(0,1,1,0)
> xor.data <- data.frame( expand.grid(c(0,1),c(0,1)),
                XOR)
> print( net.xor <- neuralnet( XOR~Var1+Var2,
                xor.data, hidden=2, rep=5))
> plot( net.xor, rep="best")
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

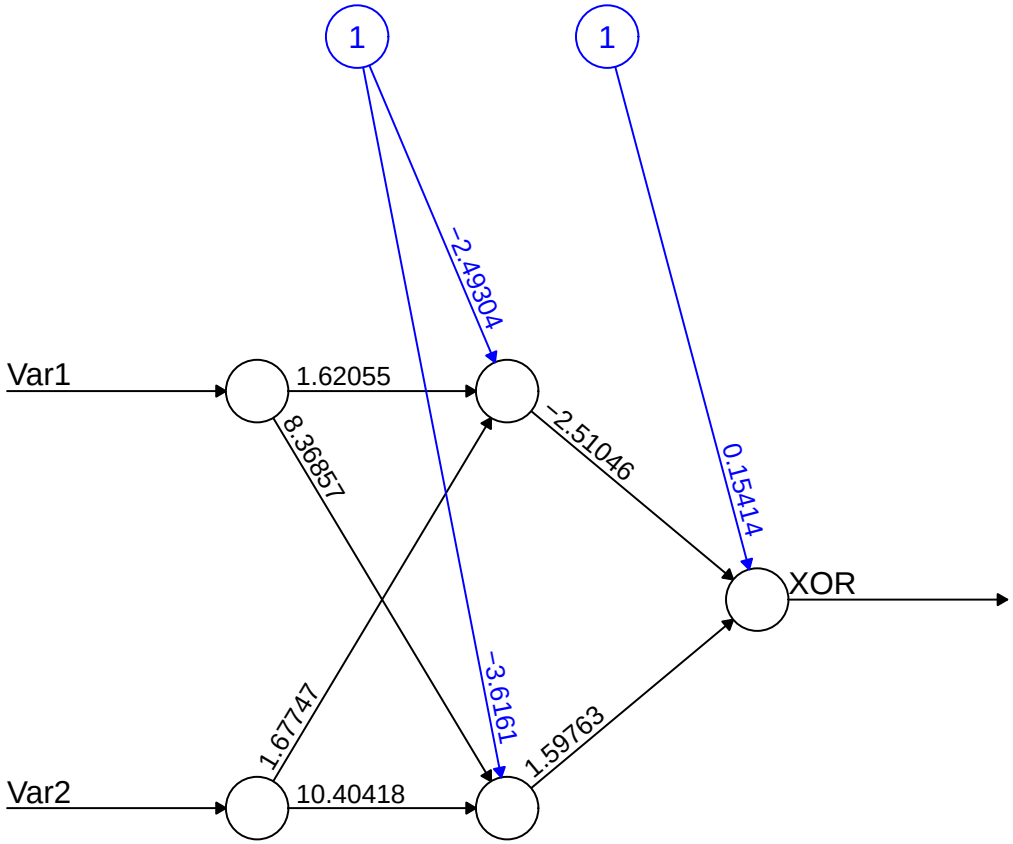
Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Backpropagation in R

MLP for Boolean XOR function:



Error: 0.000354 Steps: 105

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Backpropagation in R

Command `prediction` gives correct responses to all input combinations:

```
> prediction(net.xor)
Data Error:      0;
$rep1
  Var1 Var2             XOR
1    0    0 0.009656447796
2    1    0 0.995312514830
3    0    1 0.977844488036
4    1    1 0.018569238192
$data
  Var1 Var2 XOR
1    0    0   0
2    1    0   1
3    0    1   1
4    1    1   0
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Remarks on MLP

- Two modes of learning: stochastic and batch; compromise is mini-batch.
- Design of appropriated MLP for given problem is key.
  - Under-fitting:
    Too few hidden units lead to high training errors, cannot capture underlying trends in data, poor predictive performance.
  - Over-fitting:
    Too many hidden units result in low training errors, excessively complex model, poor predictive performance.
- Gradient descent with backpropagation may not find global optimum of error function E, only local minimum.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example (Deep Learning) – MNist

- Deep learning (buzzword) refers to artificial neural networks with many processing layers.
- *MNist* is a database of handwritten digits for image processing systems.
- Database has 60,000 training images and 10,000 testing images (50 MB).
- Most widely used benchmark for isolated handwritten digit recognition.
- Black-white images are normalized to $28 \times 28 = 784$ pixels, pixel intensity ranges from 0 (background) to 255 (max foreground).

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

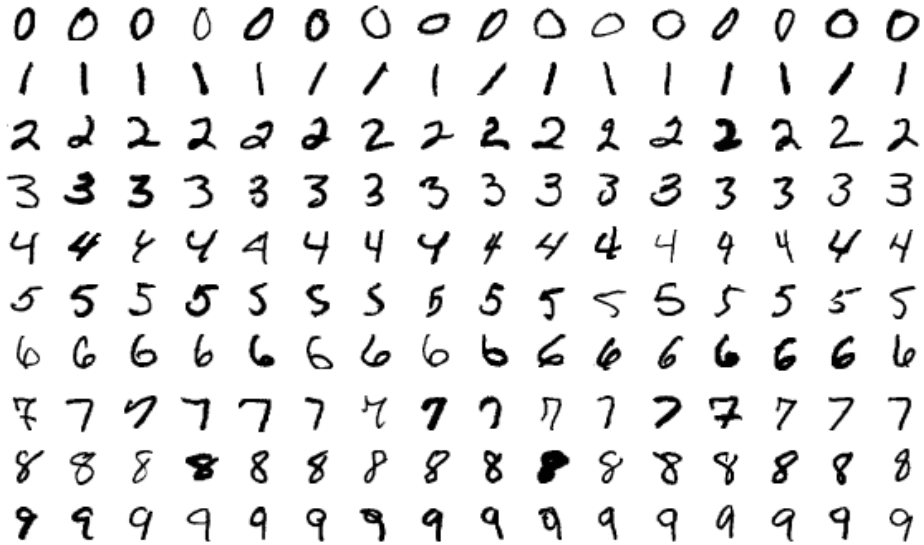
Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example – MNist (cont'd)

Handwritten digits:



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

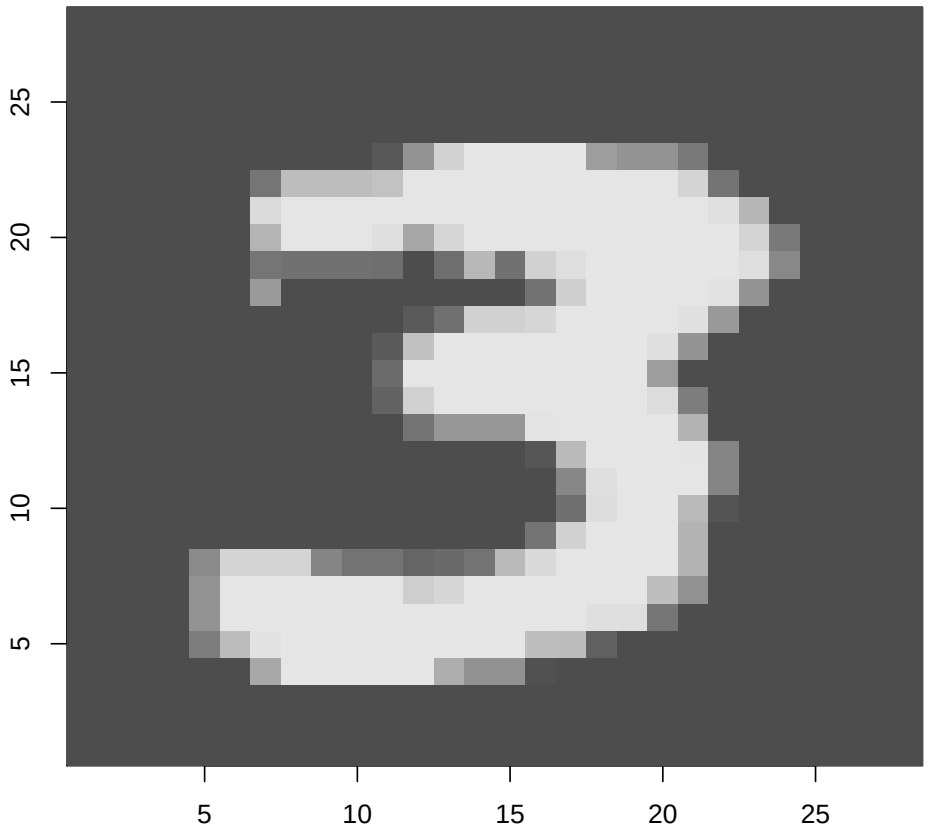
Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Digitalized Handwritten Digit



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example – MNist (cont'd)

- Some machine learning algorithms have attained human-like performance with MNist data.
- Weakest algorithm listed has error rate 12% (linear classifier).
- Today, there are neural network algorithms listed whose error rates range from 0.21% to 0.35% .

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example – MNist (cont'd)

- Deep neural network algorithm by Dan Ciresan et al. (2010) uses backpropagation, error rate 0.35%.
- Trained are 5 MLPs with 2 to 9 hidden layers and varying numbers of hidden units (Table below).
- This gives 1.34 to 12.11 million free parameters in the form of weights.
- Original grey scale images are mapped to real values $\frac{\text{pixelIntensity}}{127.5} - 1.0$ in the interval $[-1.0, 1.0]$ which are then fed into the input layer.
- Weights are initialized with uniform random distribution in $[-0.05, 0.05]$.
- Activation function is scaled hyperbolic tangent $y(h) = A \cdot \tanh(B \cdot h)$ with $A = 1.7159$ and $B = 0.6666$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example – MNist (cont'd)

- For each training epoch, the MNist training set gets slightly deformed by rotation, scaling, and shearing.
- Simulations were carried out on a computer with a Core2 Quad 9450 2.66 GHz processor, 3 GB of RAM, and a GTX280 graphics card.
- This has led to a speedup of the algorithm by a factor of 40.
- The best MLP has error rate of 0.35% which amounts to 35 out of 10,000 test images.
- The misclassified test images are shown in the article and are only partially detectable by a human!

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example – MNist (cont'd)

MLPs for handwritten digit detection:

ID	architecture (neurons per layer)	test error	weights (millions)
1	$1000, 50, 10$	0.49	1.34
2	$1500, 1000, 50, 10$	0.46	3.26
3	$2000, 1500, 1000, 50, 10$	0.41	6.69
4	$2500, 2000, 1500, 1000, 50, 10$	0.35	12.11
5	$9 \times 1000, 10$	0.44	8.86

The output layer has always 10 neurons corresponding to the 10 digits. Decision is made using the principle "winner takes it all".

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Keras

Keras is a high-level neural network API focussing on fast experimentation.

- Same code can run on CPU or GPU.
- User-friendly API for prototyping deep learning models.
- Built-in support for convolutional networks (computer vision) and recurrent networks (sequence processing).
- Capable of running with Tensorflow backend (software library for differential programming).

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

R Interface to Keras

Download package:

```
> install.packages("keras")
```

Load package and tensorflow backend:

```
> library(keras)
> install_keras()
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

MNist Dataset

MNist is already included in the Keras/Tensorflow installation.

```
mnist <- dataset_mnist()
```

Dataset is given by

- Training set:
  - Sample images x: 60,000 28×28 pixel images with grey scale representation (pixels as integers $0 \ldots 255$)
  - Responses y: vector of length 60,000 with corresponding digits between 0 and 9.
- Testing set: same as training set, but with 10,000 images and responses.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

MNist Dataset

Detailled structure of dataset:

```
> str(mnist)

List of 2
$ train: List of 2
.. $ x: int [1:60000, 1:28, 1:28] 0 0 0 0 0 0 0 0 0 0 ..
.. $ y: int [1:60000(1d)] 5 0 4 1 9 2 1 3 1 4
$ test: List of 2
.. $ x: int [1:10000, 1:28, 1:28] 0 0 0 0 0 0 0 0 0 0 ..
.. $ y: int [1:10000(1d)] 7 2 1 0 4 1 4 9 5 9
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

MNist Dataset

Preparation of features (x) and responses (y) for both training and testing.

```
> x_train <- mnist$train$x
> y_train <- mnist$train$y
> x_test <- mnist$test$x
> y_test <- mnist$test$y
```

Check the structure of datasets:

```
> str(x_train)
```

```
int [1:60000, 1:28, 1:28] 0 0 0 0 0 0 0 0 0 0 ...
int [1:60000(1d)] 5 0 4 1 9 2 1 3 1 4
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Plotting of Images

```
> index_image = 28 # arbitrary
> input_matrix <- x_train[index_image, 1:28, 1:28]
# rotate image by 90 degree
> output_matrix <- apply(input_matrix, 2, rev)
> output_matrix <- t(output_matrix)
> image(1:28, 1:28, output_matrix,
        col=gray.colors(256), xlab="", ylab="")
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

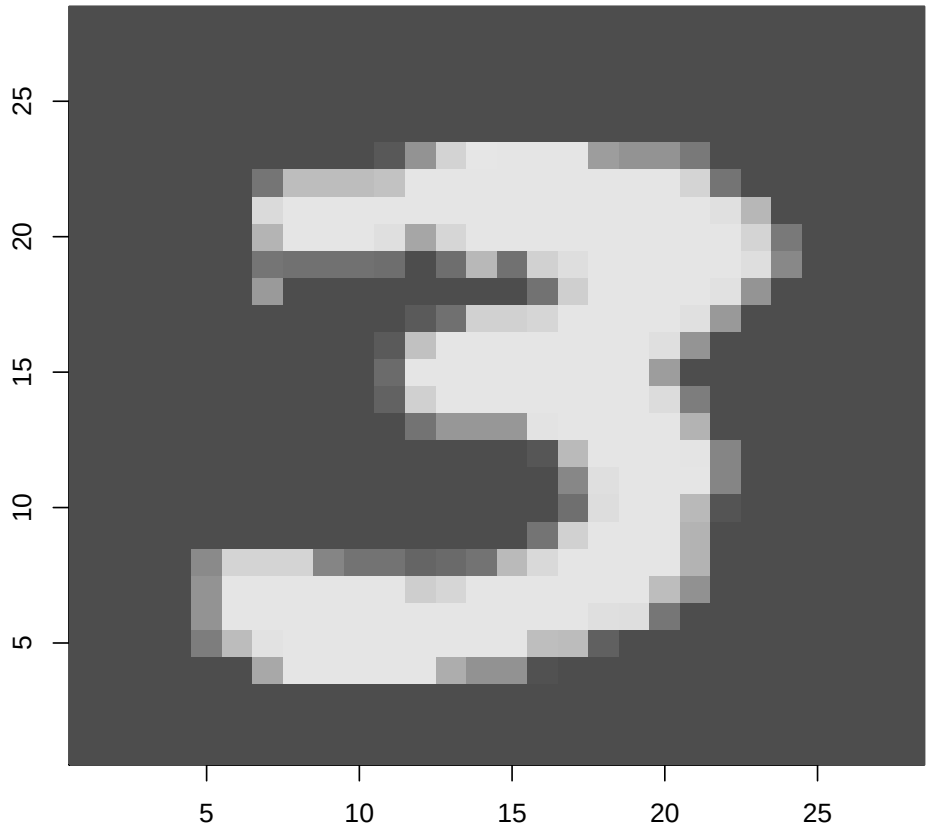
Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Image Plotting

Image number 28:



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Image Representation

Image number 28:

```
> input_matrix

     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] ...
 [1,]    0    0    0    0    0    0    0    0    0     0
 [2,]    0    0    0    0    0    0    0    0    0     0
 [3,]    0    0    0    0    0    0    0    0    0     0
 [4,]    0    0    0    0    0    0    0    0    0     0
 [5,]    0    0    0    0    0    0    0    0    0     0
 [6,]    0    0    0    0    0    0    0    0    0     0
 [7,]    0    0    0    0    0    0   39  158  158   158
 [8,]    0    0    0    0    0    0  226  253  253   253
 [9,]    0    0    0    0    0    0  139  253  253   253
[10,]    0    0    0    0    0    0   39   34   34    34
...
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Parameter Setup

```
> batch_size <- 128
> num_classes <- 10
> epochs <- 10
> img_rows <- 28
> img_cols <- 28
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Data Preprocessing: Adding Channel

In view of CNN, the $m \times n$ input images are $m \times n \times k$ 3D arrays with $k \geq 1$ channels.

```
> x_train <- array_reshape(x_train, c(nrow(x_train),
                img_rows, img_cols, 1))
> x_test <- array_reshape(x_test, c(nrow(x_test),
                img_rows, img_cols, 1))
> input_shape <- c(img_rows, img_cols, 1)

> str(x_train)

num [1:60000, 1:28, 1:28, 1] 0 0 0 0 0 0 0 0 0 0 ...
```

For greyscale images, one channel is sufficient.
For RGB color images, one needs three channels.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Data Preprocessing: Scaling and Categorial Variable

Scaling the input values to be between 0 and 1:

```
> x_train <- x_train / 255
> x_test <- x_test / 255
```

Response variable is converted to categorial:

```
> y_train <- to_categorial(y_train, num_classes)
> y_test <- to_categorial(y_test, num_classes)
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

CNN Model Structure

- Inventor of CNN is Yann LeCun (2016).
- CNN consists of one or more convolutional layers followed by a pooling layer.
- In deep learning, CNN consists of a sequence of such combined layers.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

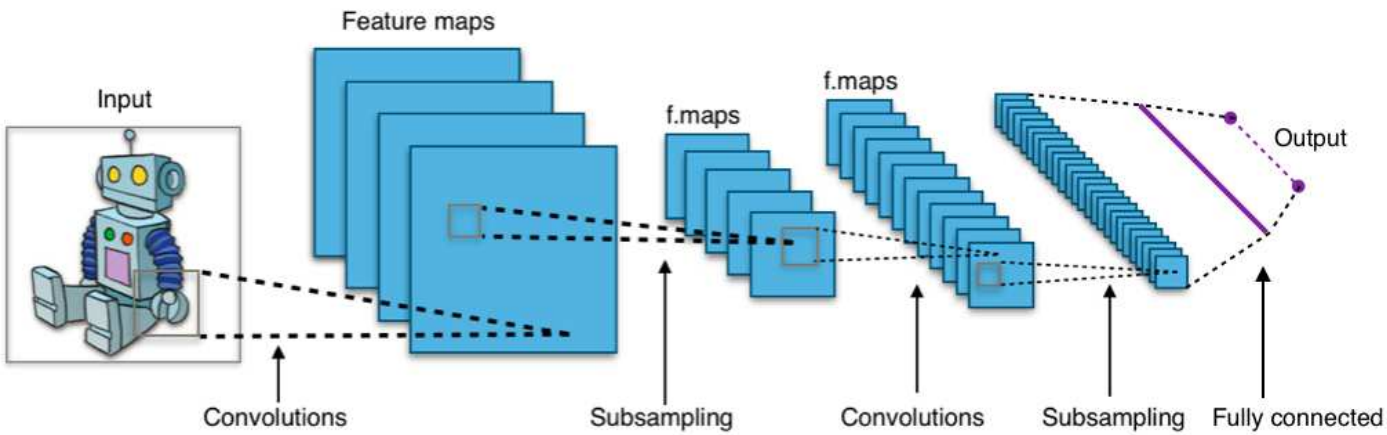
Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

CNN Model Structure



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

CNN Model Structure

- *Convolutional layer* performs discrete convolution via small convolution matrix:

$$g(x,y) = (\omega * f)(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} \omega(i,j) f(x-i, y-j) \quad (388)$$

where $g(x,y)$ is the filtered image, $f(x,y)$ is the original image, and ω is the filter kernel, $(2a+1 \times 2b+1)$ matrix.

- Kernels can cause a wide range of effects, e.g., edge detection:

$$\omega = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

- Activation function of neuron is given by *rectified linear unit* (relu) $f(x) = \max\{0, x\}$ or its differentiable approximation $f(x) = \ln(1 + e^x)$ (for backpropagation).

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

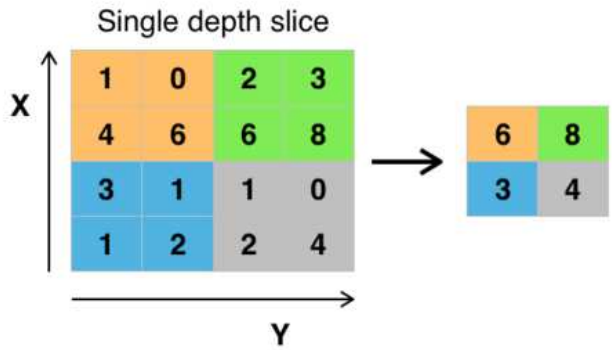
Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

CNN Model Structure

- In the pooling (or subsampling) layer the idea is to remove superfluous information.
- Pooling layer should prevent overfitting, reduce data and speed-up computation.
- Max-pooling is most commonly used:



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

CNN Model Structure

Layers in actual CNN:

- 2D convolutional layers parametrized with kernel_size, number of filters, activation function.
- Pooling layer is applied after each 2D convolutional layer; pool_size of 2×2 halves the size.
- Dropout layer for regularization; e.g., dropout rate of 0.8 refers to (random) deletion of 20% of the neurons preventing overfitting.
- Flatten layer converts 3D tensor to 1D tensor.
- Dense layer (MLP) connects to target response classes.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

CNN Model for MNist

CNN model with two 2D convolutional layers with max-pooling.

```
> cnn_model <- keras_model_sequential() %>%
+   layer_conv_2d(filters=32, kernel_size=c(3,3),
      activation='relu', input_shape=input_shape) %>%
+   layer_max_pooling_2d(pool_size=c(2,2)) %>%
+   layer_conv_2d(filters=64, kernel_size=c(3,3),
      activation='relu') %>%
+   layer_max_pooling_2d(pool_size=c(2,2)) %>%
+   layer_dropout(rate=0.25) %>%
+   layer_flatten() %>%
+   layer_dense(units=128, activation='relu') %>%
+   layer_dropout(rate=0.5) %>%
+   layer_dense(units=num_classes, activation='softmax')
```

softmax = normalized e-function

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

CNN Model for MNist

Compilation of defined CNN model:

```
> cnn_model %>% compile
+   loss = loss_categorial_crossentropy,
+   optimizer = optimizer_adadelta(),
+   metrics = c('accuracy')
)
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

CNN Model for MNist

Training of the model via 10 epochs:

```
cnn_history <- cnn_model %>% fit(
+   x_train, y_train,
+   batch_size = batch_size,
+   epochs = epochs,
+   validation_split = 0.2 )

Train on 48000 samples, validate on 12000 samples
```

This program is run on CPU (approx. 30 min). The training time can be significantly reduced if run on GPU.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

CNN Model for MNist

Training accuracy

```
Epoch 1/10
48000/48000 - accuracy: 0.8861
Epoch 2/10
48000/48000 - accuracy: 0.9639
Epoch 3/10
48000/48000 - accuracy: 0.9723
Epoch 4/10
48000/48000 - accuracy: 0.9783
Epoch 5/10
48000/48000 - accuracy: 0.9803
Epoch 6/10
48000/48000 - accuracy: 0.9822
Epoch 7/10
48000/48000 - accuracy: 0.9842
Epoch 8/10
48000/48000 - accuracy: 0.9850
Epoch 9/10
48000/48000 - accuracy: 0.9870
Epoch 10/10
48000/48000 - accuracy: 0.9875
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

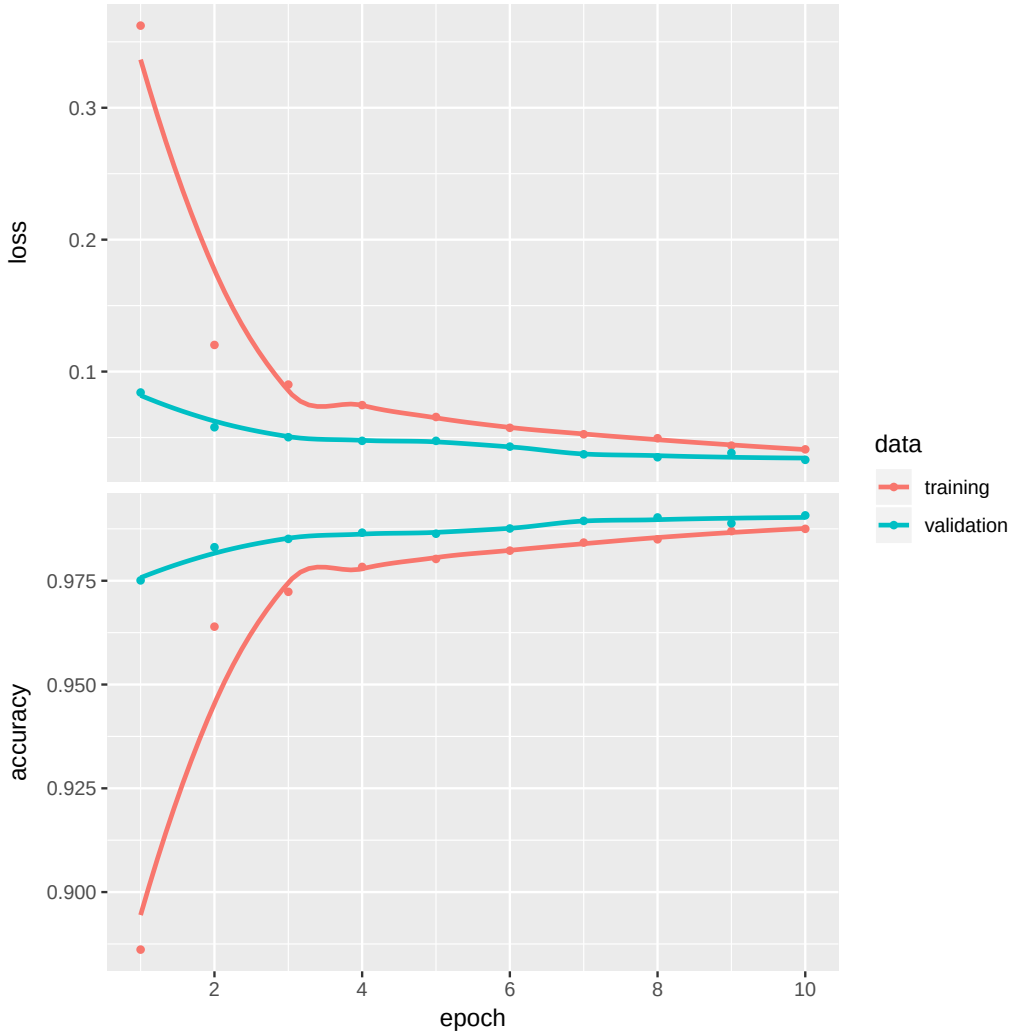
Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

CNN Model for MNist

```
> plot(cnn_history)
```



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

CNN Model for MNist

Model prediction of new images:

```
> cnn_pred <- cnn_model %>%
+   predict_classes(x_test)
> head(cnn_pred, n=10)

[1] 7 2 1 0 4 1 4 9 5 9
```

Check number of misclassified images

```
> sum(cnn_pred != mnist$test$y)

[1] 83
```

Error rate of 0.83% means 83 out of 10.000 test images.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

CNN Model for MNist

Check misclassified images whether human can do better:

```
> missed_image = mnist$test$x[cnn_pred!=mnist$test$y,,]
> missed_digit = mnist$test$y[cnn_pred!=mnist$test$y]
> missed_pred = cnn_pred[cnn_pred!=mnist$test$y]

> index_image = 2 # arbitrary
> input_matrix <- missed_image[index_image, 1:28, 1:28]
> output_matrix <- apply(input_matrix, 2, rev)
> output_matrix <- t(output_matrix)
> image(1:28, 1:28, output_matrix,
    col=gray.colors(256),
    xlab=paste(
     'Image for digit ', missed_digit[index_image],
     ', wrongly predicted as ',missed_pred[index_image]),
    ylab="")
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

CNN Model for MNist

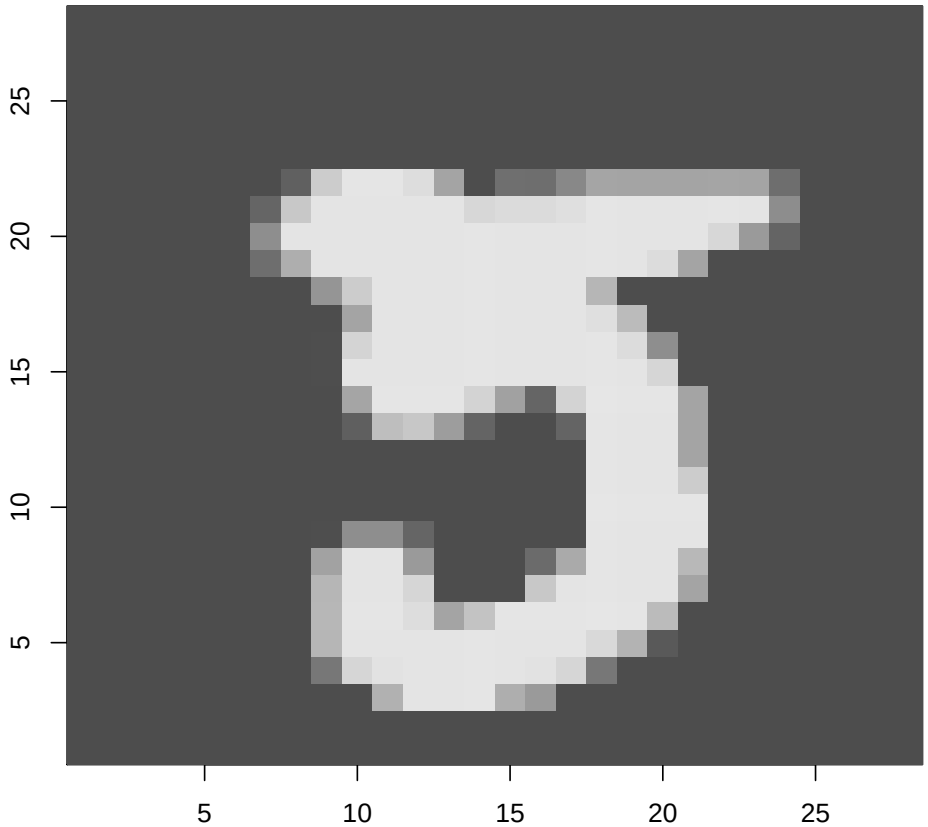


Image for digit 5 , wrongly predicted as 3

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Universal Function Representation and Approximation

Fundamental results about representation and approximation capabilities of feedforward neural networks.

- Kolmogorov network
- *Sprecher's learning algorithm
- *Cybenko's approximation network

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Theorem (Kolmogorow, 1957)

Let $n \geq 2$. Each continuous function $f : [0,1]^n \to \mathbb{R}$ has a representation of the form

$$f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \phi_q \left(\sum_{p=1}^{n} \psi_{p,q}(x_p) \right), \tag{389}$$

where ϕ_q and $\psi_{p,q}$ are monadic continuous functions, $1 \leq q \leq 2n+1$ and $1 \leq p \leq n$.
The inner functions $\psi_{p,q}$ are monotonic increasing and independent of the function f.

Kolmogorov's proof is not constructive; unclear how to choose the involved monadic functions.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

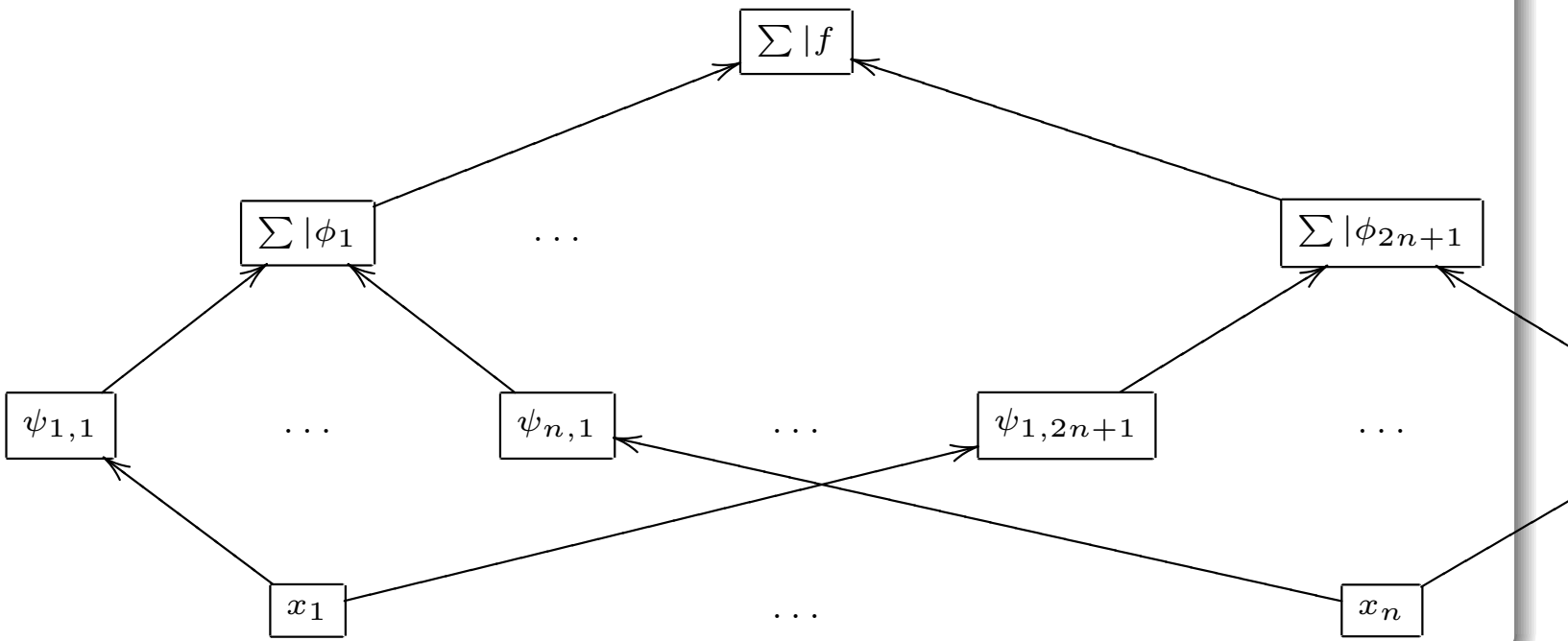
Neural Networks in R

Universal Function Representation and Approximation

Rewrite the representation of f:

$$\begin{aligned} f(x_1, \ldots, x_n) &= \phi_1(y_1) + \phi_2(y_2) + \ldots + \phi_{2n+1}(y_{2n+1}), \\ y_q &= \psi_{1,q}(x_1) + \psi_{2,q}(x_2) + \ldots + \psi_{n,q}(x_n), \quad (390) \\ & 1 \leq q \leq 2n+1. \end{aligned}$$

Layered network for the computation of continuous function:



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example

Dyadic continuous function $f : [0,1]^2 \to \mathbb{R}$ has representation

$$f(x_1, x_2) = \sum_{q=1}^{5} \phi_q(y_q),$$

where the ϕ_q are monadic continuous functions and

$$\begin{aligned} y_1 &= \psi_{1,1}(x_1) + \psi_{2,1}(x_2), \\ &\vdots \\ y_5 &= \psi_{1,5}(x_1) + \psi_{2,5}(x_2), \end{aligned}$$

where the $\psi_{p,q}$ are monadic continuous functions.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example

Consider the dyadic multiplication $f(x_1, x_2) = x_1 \cdot x_2$. Then

$$x_1 \cdot x_2 = \frac{1}{4}(x_1 + x_2)^2 - \frac{1}{4}(x_1 - x_2)^2.$$

Define the functions

$$\phi_1(t) = \frac{1}{4}t^2,\ \phi_2(t) = -\frac{1}{4}t^2,\ \phi_3(t) = \phi_4(t) = \phi_5(t) = 0$$

and

$$\psi_{1,1}(x) = \psi_{2,1}(x) = \psi_{1,2}(x) = x \quad \text{and} \quad \psi_{2,2}(x) = -x.$$

Then

$$\begin{aligned} x_1 \cdot x_2 &= \phi_1(x_1 + x_2) + \phi_2(x_1 + (-x_2)) \\ &= \phi_1\left(\psi_{1,1}(x_1) + \psi_{2,1}(x_2)\right) + \phi_2\left(\psi_{1,2}(x_1) + \psi_{2,2}(x_2)\right). \end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

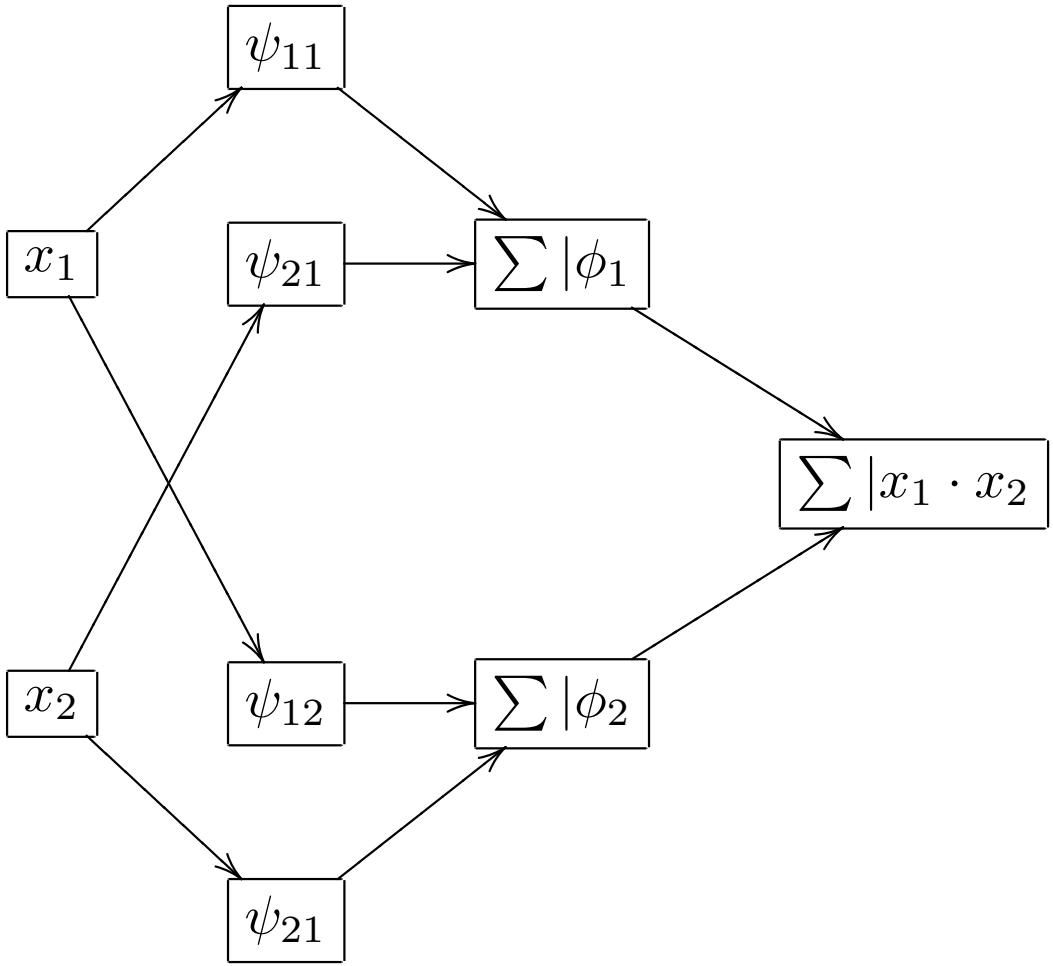
Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example (cont'd)

Layered network structure for dyadic multiplication:



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

*Theorem (Sprecher, 1965)

Let $n \geq 2$. Each continuous function $f : [0,1]^n \to \mathbb{R}$ has a representation of the form

$$f(x_1, \ldots, x_n) = \sum_{q=0}^{2n} \phi_q \left(\sum_{p=1}^{n} \lambda_p \psi(x_p + q\alpha) \right), \tag{391}$$

where ψ and $\phi_0, \ldots, \phi_{2n}$ are continuous functions, and α and $\lambda_1, \ldots, \lambda_n$ are constants.
The inner function ψ is monotonic increasing and independent of the function f.

This representation uses one inner function ψ plus $n+1$ constants $\alpha, \lambda_1, \ldots, \lambda_n$ instead of $(2n+1)n$ inner functions $\psi_{p,q}$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Sprecher's Learning Algorithm

Given dyadic continuous function $f : [0,1]^2 \to \mathbb{R}$.

- Define *inner function* ψ.
- Construct sequence of continuous functions (f_i) which converges uniformly to f.

Sprecher's Inner Function

- Let $k \geq 1$. Consider the *terminating decimals* of length k,

$$d_k = \sum_{r=1}^{k} \frac{i_r}{10^r}, \tag{392}$$

where $0 \leq i_r \leq 9$ for each $1 \leq r \leq k$.

Example: For $k = 3$, we have 0.001, 0.100, and 0.999.

- Define *point function* ψ_1 as

$$\psi_1(d_1) = d_1 \tag{393}$$

for each terminating decimal d_1 of length 1. Extend ψ_1 to continuous staircase function on $[0, 1]$ by connecting consecutive points with line segments; this gives the identity function on $[0, 1]$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Sprecher's Inner Function

Second, define *point function* ψ_2 on terminating decimals of length $k = 2$,

$$\psi_2\left(\frac{i_2}{10^2}\right) = \begin{cases} \frac{i_2}{10^{\beta(2)}} & \text{if } 0 \leq i_2 \leq 8, \\ \frac{1}{2}\left(\frac{8}{10^{\beta(2)}} + \frac{1}{10}\right) & \text{if } i_2 = 9, \end{cases} \tag{394}$$

$$\psi_2\left(\frac{1}{10}\right) = \frac{1}{10} \tag{395}$$

and

$$\psi_2\left(\frac{i_1}{10} + \frac{i_2}{10^2}\right) = \frac{i_1}{10} + \psi_2\left(\frac{i_2}{10^2}\right). \tag{396}$$

Extend ψ_2 to continuous staircase function on $[0, 1]$ by connecting consecutive points with line segments.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

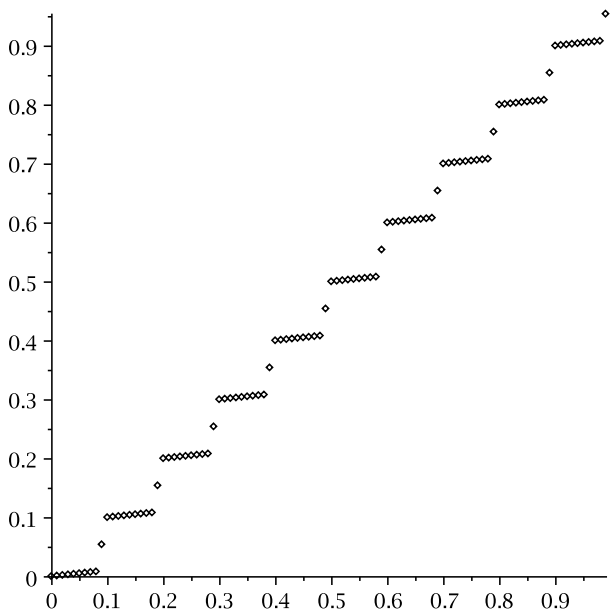
Self-Organizing Networks

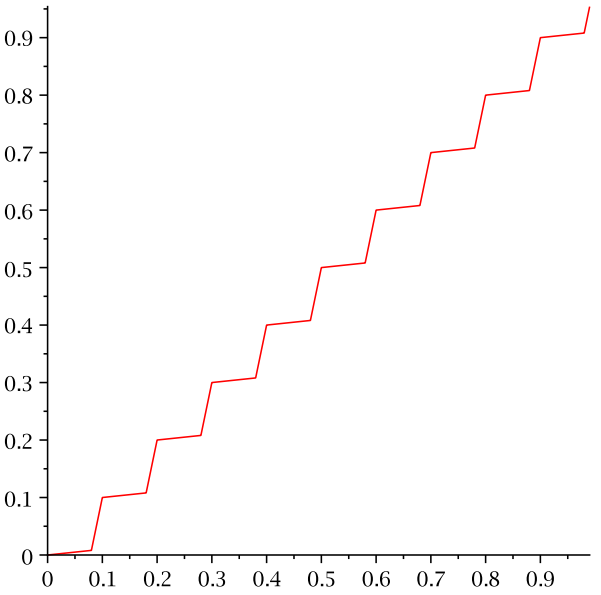
*Spiking Neural Networks

Neural Networks in R

Sprecher's Inner Function

The point function ψ_2 and its extension to a continuous staircase function:





Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Sprecher's Inner Function

- $\mathcal{Q}_k =$ the set of terminating rational numbers of length $k \geq 1$ in $[0, 1]$.
- $\mathcal{Q} = \bigcup_{k \geq 0} \mathcal{Q}_k$ the set of all terminating rational numbers in $[0, 1]$.
- $\mathcal{Q}$ is *dense* in $[0, 1]$; i.e., each interval contained in $[0, 1]$ has an element of $\mathcal{Q}$.

For $k \geq 1$ define the point function ψ_k on $\mathcal{Q}_k$ as

$$\psi_k(d_k) = \tag{397}$$
$$\begin{cases} d_k & k = 1, \\ \psi_k\left(d_k - \frac{i_k}{10^k}\right) + \frac{i_k}{10^{\beta(k)}} & k > 1 \text{ and } 0 \leq i_k \leq 8, \\ \frac{1}{2}\left(\psi_k\left(d_k - \frac{1}{10^k}\right) + \psi_{k-1}\left(d_k + \frac{1}{10^k}\right)\right) & k > 1 \text{ and } i_k = 9. \end{cases}$$

Extend ψ_k to continuous staircase function on $[0, 1]$ by connnecting consecutive points with line segments.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Sprecher's Inner Function

- The set of real numbers is the set of limits of infinite decimal sequences, i.e., each real number $x \in [0,1]$ has a representation to base 10 as

$$x = \sum_{r=1}^{\infty} \frac{i_r}{10^r} = \lim_{k \to \infty} \sum_{r=1}^{k} \frac{i_r}{10^r}. \tag{398}$$

- Define the limit of the sequence of (extended) functions (ψ_k) by extending the dense set $\mathcal{Q}$ to $[0,1]$,

$$\psi(x) = \lim_{k \to \infty} \psi_k \left(\sum_{r=1}^{k} \frac{i_r}{10^r} \right). \tag{399}$$

The (pointwise) limit exists and the limiting function ψ is continuous and monotonic increasing; note that ψ is Lipschitz continuous, but not smooth.

Sprecher's Outer Functions

- For the terminating decimals of length k, put

$$\delta_k = \frac{8}{9 \cdot 10^k} = 8 \sum_{r=k+1}^{\infty} \frac{1}{10^r}. \tag{400}$$

- Consider the intervals of length δ_k,

$$I_k(d_k) = [d_k, d_k + \delta_k]. \tag{401}$$

The inverval $I_k(d_k)$ contains the disjoint union of nine intervals,

$$I_{k+1}(d_k) \dot\cup I_{k+1}(d_k + \frac{1}{10^{k+1}}) \dot\cup \ldots \dot\cup I_{k+1}(d_k + \frac{8}{10^{k+1}}) \tag{402}$$

and the interval $I_{k+1}(d_k + \frac{9}{10^{k+1}})$ lies in the gap between the intervals $I_k(d_k)$ and $I_k(d_k + \frac{1}{10^k})$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Sprecher's Outer Functions

- For each number $k \geq 1$, the intervals $I_k(d_k)$ will be translated by the displacements

$$\frac{q}{9 \cdot 10} = q \sum_{r=2}^{\infty} \frac{1}{10^r}, \quad 0 \leq q \leq 4. \tag{403}$$

This gives a larger family of intervals,

$$I_{q,k} = \left[d_k - \frac{q}{9 \cdot 10}, d_k + \delta_k - \frac{q}{9 \cdot 10} \right]. \tag{404}$$

- We have $I_{0,k}(d_k) = I_k(d_k)$. The geometry of the translated intervals is similar to that of the intervals $I_k(d_k)$.
- Define the Cartesian products

$$S_{q,k}(d_{1k}, d_{2k}) = I_{q,k}(d_{1k}) \times I_{q,k}(d_{2k}), \ 0 \leq q \leq 4, \ k \geq 1 \tag{405}$$

where d_{1k} and d_{2k} are terminating decimals of length k. These squares have diameter $\sqrt{2} \cdot \delta_k$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Sprecher's Outer Functions

Claim that the given continuous function $f : [0,1]^2 \to \mathbb{R}$ can be represented as

$$\begin{aligned} f(x_1, x_2) &= \phi_0(y_0) + \ldots + \phi_4(y_4), \\ y_q &= \psi(x_1 + q\alpha) + \lambda\psi(x_2 + q\alpha), \quad 0 \leq q \leq 4. \end{aligned} \tag{406}$$

Construct a sequence of continuous functions (f_i) which converges uniformly to f.

Supremum norm of real-valued bounded function f on $[0,1]^2$,

$$\|f\| = \sup\{|f(x,y)| \mid (x,y) \in [0,1]^2\}. \tag{407}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Sprecher's Outer Functions

- Let $k = k_1 \geq 1$. Define the functions $\phi_{q,1}$ as

$$\phi_{q,1}(y_q) = \frac{1}{3} f(d_{1,k_1}, d_{2,k_1}), \quad 0 \leq q \leq 4, \tag{408}$$

for each $(x_1, x_2) \in S_{q,k_1}(d_{1,k_1}, d_{2,k_1})$. Extend these functions to continuous staircase functions on $[0, 1]$ by connecting consecutive points with line segments.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Sprecher's Outer Functions

- Define the function f_1 as

$$f_1(x_1, x_2) = \sum_{q=0}^{4} \phi_{q,1}(y_q) = \sum_{q=0}^{4} \phi_{q,1}(\psi(x_1 + q\alpha) + \lambda\psi(x_2 + q\alpha)). \tag{409}$$

Take an error value $\epsilon > 0$ and choose the number k_1 large enough such that

$$|f_1(x_1, x_2) - f_1(x_1', x_2')| \leq \epsilon \|f\| \tag{410}$$

for all pairs $(x_1, x_2), (x_1', x_2')$ in $S_{q,k_1}(d_{1,k_1}, d_{2,k_1})$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Sprecher's Outer Functions

- For each point $(x_1, x_2) \in [0,1]^2$, we have

$$|f(x_1, x_2) - \phi_{q,1}(y_q)| \leq \frac{1}{3}\epsilon\|f\|. \tag{411}$$

It follows that for all $(x_1, x_2) \in [0,1]^2$,

$$|f(x_1, x_2) - f_1(x_1, x_2)| \leq \left(\epsilon + \frac{2}{3}\right)\|f\|. \tag{412}$$

Sprecher's Outer Functions

- By continuing this process, we obtain for each $r \geq 1$,

$$\|\phi_{q,r}\| \leq \frac{1}{3}\left(\epsilon + \frac{2}{3}\right)^{r-1} \|f\|, \quad 0 \leq q \leq 4, \tag{413}$$

and

$$\|f - \sum_{i=1}^{r} f_i\| \leq \left(\epsilon + \frac{2}{3}\right)^{r} \|f\|. \tag{414}$$

- We have $(\epsilon + \frac{2}{3})^r \to 0$ for $r \to \infty$ if $0 < \epsilon < \frac{1}{3}$. With this setting, the function series $(\sum_i f_i)$ and $(\phi_{q,i})$ are convergent uniformly to the functions f and ϕ_q, respectively. It follows that the functions f and ϕ_q, $0 \leq q \leq 4$, are continuous.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Corollary

Each multivariate continuous function can be represented by a network of monadic functions and addition operations called *Kolmogorov network*.

See diagram in manuscript.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

*Approximation Result

- Approximation of continuous function when exact reproducibility is not required and a bounded approximation error is accepted.
- $C(I_n)$ = space of continuous functions on $I_n = [0,1]^n$; supremum norm on $C(I_n)$ is denoted by $\|\cdot\|$.
- $M(I_n)$ = space of finite, signed regular Borel measures on I_n.
- A function $\sigma : \mathbb{R} \to \mathbb{R}$ is *discriminatory* if for some measure $\mu \in M(I_n)$,

$$\int_{I_n} \sigma\left(\sum_{i=1}^{n} x_i y_i + \theta\right) d\mu(x) = 0 \tag{415}$$

for all $y \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$ implies that $\mu = 0$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

A function $\sigma : \mathbb{R} \to \mathbb{R}$ is *sigmoidal* if

$$\lim_{x \to \infty} \sigma(x) = 1 \quad \text{and} \quad \lim_{x \to -\infty} \sigma(x) = -1. \tag{416}$$

Proposition

Let σ be a discriminatory function. The finite sums of the form

$$F(x) = \sum_{j=1}^{m} \alpha_j \sigma \left(\sum_{i=1}^{n} x_i y_{ji} + \theta_j \right), \quad x \in I_n, \tag{417}$$

with parameters $\alpha_j, y_{ji}, \theta_j \in \mathbb{R}$ are dense in $C(I_n)$; that is, for all $f \in C(I_n)$ and $\epsilon > 0$, there is a sum $F(x)$ with

$$|F(x) - f(x)| < \epsilon \quad \text{for all } x \in I_n. \tag{418}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Proof [Sketch]

- Let $L =$ set of functions $F(x)$ as above; L is a linear subspace of $C(I_n)$.
- Claim that closure of L denoted by $\overline{L}$ is dense in $C(I_n)$; $\overline{L}$ closed proper subspace of $C(I_n)$. Suppose $\overline{L}$ is not all of $C(I_n)$;
- Hahn-Banach theorem: $\exists$ bounded linear functional $\ell \neq 0$ on $C(I_n)$ with $\ell(L) = \ell(\overline{L}) = 0$.
- Riesz representation theorem: Functional ℓ has the form

$$\ell(h) = \int_{I_n} h(x) d\mu(x), \quad h \in C(I_n), \tag{419}$$

for some $\mu \in M(I_n)$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Proof [Sketch] cont'd

- $\sigma\left(\sum_{i=1}^{n} x_i y_i + \theta\right) \in \overline{L}$ for all $y \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$. Then

$$\int_{I_n} \sigma\left(\sum_{i=1}^{n} x_i y_i + \theta\right) d\mu(x) = 0 \tag{420}$$

  for all $y \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$.

- But σ is discriminatory and so $\mu = 0$ contradicting the hypothesis that $\ell \neq 0$. Hence, the subspace L is dense in $C(I_n)$.

Lemma

Each continuous sigmoidal function σ is discriminatory.

Theorem (Cybenko, 1989)

Let σ be a continuous sigmoidal function. The finite sums of the shape

$$F(x) = \sum_{j=1}^{m} \alpha_j \sigma \left(\sum_{i=1}^{n} x_i y_{ji} + \theta_j \right), \quad x \in I_n, \tag{421}$$

with parameters $\alpha_j, y_{ji}, \theta_j \in \mathbb{R}$ are dense in $C(I_n)$; that is, for all $f \in C(I_n)$ and $\epsilon > 0$, there is a sum $F(x)$ with

$$|F(x) - f(x)| < \epsilon \quad \text{for all } x \in I_n. \tag{422}$$

Use the Proposition and note that by the Lemma continuous sigmoidal functions are discriminatory.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

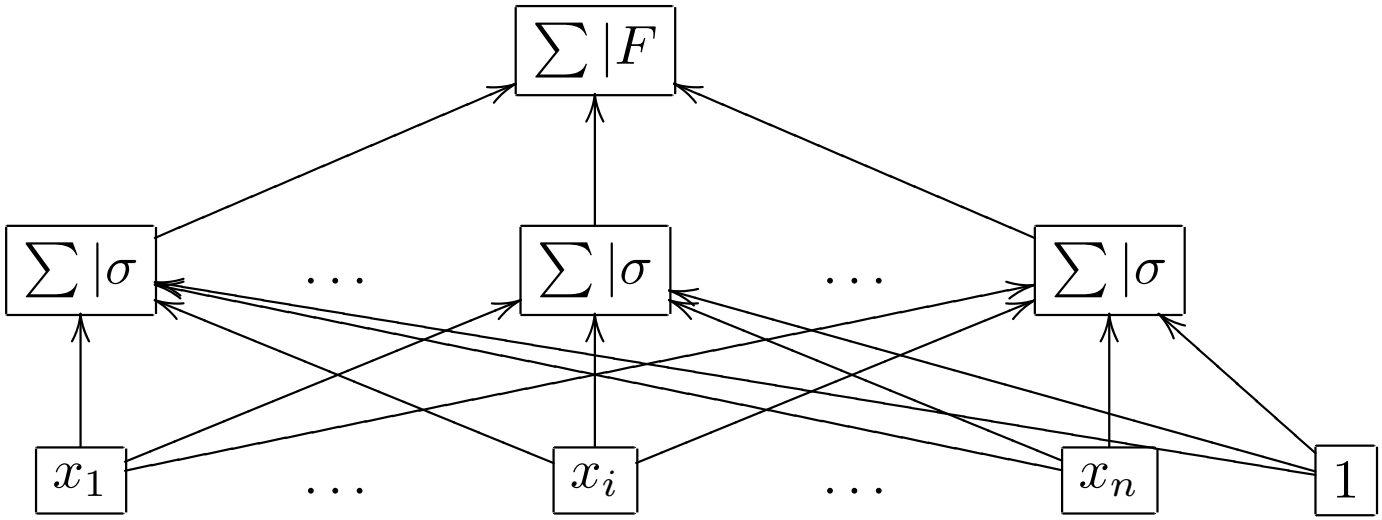
Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Network for the approximation of continuous function f:



MLPs with one hidden layer can approximate continuous functions with arbitrary precision if there are no constraints on the number of units and the size of the weights.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Pattern Classification

- $\mu =$ Lebesgue measure on I_n.
- Partition of I_n into k disjoint measurable subsets $P_1, \ldots, P_k$ of I_n.
- Find decision function $f : I_n \to \{1, \ldots, k\}$ such that

$$f(x) = i \quad \Longleftrightarrow \quad x \in P_i, \quad x \in I_n. \tag{423}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Theorem

Let σ be a continuous sigmoidal function and let f be a decision function for any finite measurable partition of I_n with Lebesgue measure μ. For each $\epsilon > 0$, there is a finite sum of the shape

$$F(x) = \sum_{j=1}^{m} \alpha_j \sigma \left(\sum_{i=1}^{n} x_i y_{ji} + \theta_j \right), \quad x \in I_n, \tag{424}$$

with parameters $\alpha_j, y_{ji}, \theta_j \in \mathbb{R}$ and a subset D of I_n such that $\mu(D) \geq 1 - \epsilon$ and

$$|F(x) - f(x)| < \epsilon \quad \text{for all } x \in D. \tag{425}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Proof.

Lusin's theorem states that each measurable function is a continuous function on nearly all its domain. More precisely, there is a continuous function $h : I_n \to \mathbb{R}$ and a subset D of I_n such that $\mu(D) \geq 1 - \epsilon$ and $h(x) = f(x)$ for all $x \in D$. Thus by Cybenko's result, there is a function F of the required form such that $|F(x) - h(x)| < \epsilon$ for all $x \in I_n$. Hence, $|F(x) - f(x)| = |F(x) - h(x)| < \epsilon$ for all $x \in D$. □

MLPs with one hidden layer can approximate these kind of decision functions with arbitrary precision if there are no constraints on the number of units and the size of the weights.
The total measure of incorrectly classified points can be made arbitrarily small.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Hopfield Network

- Recurrent neural network
- Hebb's learning rule
- Example: letter restoration
- *Ising model

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

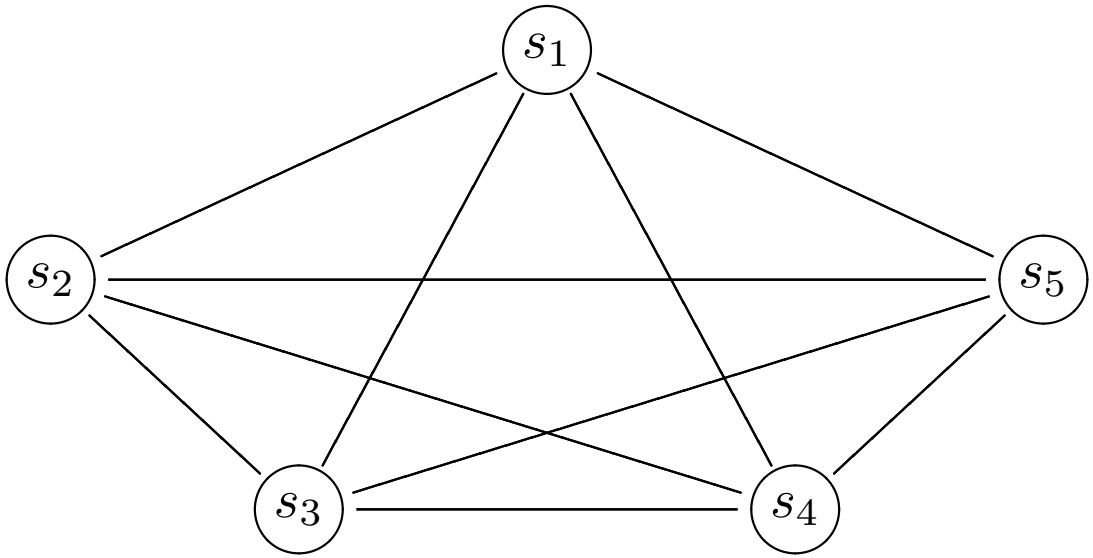
Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Recurrent neural network: neural network structure with bidirectional connections (Hopfield, 1982)



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Structure

- Neuron i has state $s_i \in \{\pm 1\}$; $s_i = 1$ if neuron is excited transmitting a signal, $s_i = -1$ if neuron is at rest.
- *Synaptic efficacy* (weight) w_{ij} between neurons i and j can be positive (excitatory) or negative (inhibitory).
- *Local field* (input to neuron i)

$$h_i = \sum_j w_{ij}(s_j + 1). \tag{426}$$

Input signal from neuron j to neuron i is $2w_{ij}$ if $s_j = 1$ and 0 if $s_j = -1$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Structure

- Neuron i is excited if local field exceeds threshold θ_i at time t:

$$\begin{aligned} s_i(t+\Delta t) &= \operatorname{sgn}\left(\sum_j w_{ij}(s_j(t)+1)-\theta_i\right) \\ &= \operatorname{sgn}\left(h_i(t)-\theta_i\right) \\ &= \begin{cases} +1 & \text{if } h_i(t) \geq \theta_i, \\ -1 & \text{otherwise.} \end{cases} \end{aligned} \tag{427}$$

- Assume threshold θ_i is proportional to synaptic efficacy of input neurons, i.e., $\theta_i = \sum_j w_{ij}$. Then

$$s_i(t+\Delta t) = \operatorname{sgn}\left(\sum_j w_{ij} s_j(t)\right). \tag{428}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Structure

- Given a recurrent neural network with N neurons.
- Study the problem of memorization of patterns.
- Take collection of patterns of excitation $\{\xi^{(1)}, \ldots, \xi^{(p)}\}$, where $\xi^{(\mu)} = (\xi_1^{(\mu)}, \ldots, \xi_N^{(\mu)}) \in \{\pm 1\}^N$ for $1 \leq \mu \leq p$.
- The memorized patterns should be stable fixed points under the *time evolution rule*:

  If

$$s_i(t) = \xi_i^{(\mu)} \tag{429}$$

  for all $1 \leq i \leq N$ and for some $t \geq 0$ and some $1 \leq \mu \leq p$, then

$$s_i(t + \Delta t) = \xi_i^{(\mu)} \tag{430}$$

  for all $1 \leq i \leq N$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Hebb's Rule

- For random patterns, each pattern $\xi^{(\mu)}$ is a stable fixed point as long as p is not too large and we put

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{p} \xi_i^{(\mu)} \xi_j^{(\mu)}, \quad 1 \leq i \neq j \leq N. \tag{431}$$

  Moreover, $w_{ii} = 0$.

- Suppose $s_i(t) = \xi_i^{(\mu)}$ for all $1 \leq i \leq N$ and some $t \geq 0$ and some $1 \leq \mu \leq p$. Then by (428,431),

$$\begin{aligned} s_i(t + \Delta t) &= \operatorname{sgn}\left(\sum_j w_{ij} \xi_j^{(\mu)} \right) \\ &= \operatorname{sgn}\left(\frac{1}{N} \sum_j \sum_\nu \xi_i^{(\nu)} \xi_j^{(\nu)} \xi_j^{(\mu)} \right). \end{aligned} \tag{432}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Hebb's Rule

- *Approximate orthogonality relation* between random patterns holds for large N,

$$\frac{1}{N}\sum_j \xi_j^{(\mu)}\xi_j^{(\nu)} = \delta_{\nu,\mu} + O(\frac{1}{\sqrt{N}}). \tag{433}$$

- Then

$$\begin{aligned}\operatorname{sgn}\left(\sum_j w_{ij}\xi_j^{(\mu)}\right) &= \operatorname{sgn}\left(\frac{1}{N}\sum_j\sum_\nu \xi_i^{(\nu)}\xi_j^{(\nu)}\xi_j^{(\mu)}\right) \\ &= \operatorname{sgn}\left(\sum_\nu \xi_i^{(\nu)}\delta_{\nu,\mu}\right) = \operatorname{sgn}\left(\xi_i^{(\mu)}\right).\end{aligned} \tag{434}$$

Thus $s_i(t+\Delta t) = \xi_i^{(\mu)}$ for all $1 \le i \le N$ and sufficiently large N.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Hebb's Rule

Update of neurons:

- *Asynchronous mode*: only one neuron is updated in each time step usually picked uniformly at randon.
- *Synchronous mode*: all neurons are updated at the same time; in large networks this requires a global clock to maintain synchronization.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example – Letter Memorization

Generating letters C, D, J, M:

```
C <- matrix(c(-1, 1, 1, 1,-1, 1,-1,-1,-1, 1, 1,-1,-1,-1, 1, 1,-1,-1,-1, 1, 1,-1,-1,-1, 1),
  nrow=5, ncol=5)
D <- matrix(c( 1,-1,-1,-1,-1, 1, 1, 1, 1, 1, 1,-1,-1,-1, 1, 1,-1,-1,-1, 1,-1, 1, 1, 1,-1),
  nrow=5, ncol=5)
J <- matrix(c( 1,-1,-1, 1, 1, 1,-1,-1,-1, 1, 1,-1,-1,-1, 1, 1, 1, 1, 1,-1, 1,-1,-1,-1,-1),
  nrow=5, ncol=5)
M <- matrix(c( 1, 1, 1, 1, 1,-1, 1,-1,-1,-1,-1,-1, 1,-1,-1,-1, 1,-1,-1,-1, 1, 1, 1, 1, 1),
  nrow=5, ncol=5)
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example – Letter Memorization (Cont'd)

Print the letters:

```
> show.letter <- function(letter.vector) {
+   letter.vector[letter.vector == 1] <- "*"
+   letter.vector[letter.vector == -1] <- " "
+   col.names(letter.vector) <- rep("",5)
+   row.names(letter.vector) <- rep("",5)
+   print(letter.vector, quote = FALSE)
+ }
> for (i in mget(ls(pattern = "^[A-Z]")))
+   {show.letter(i)}
```

```
##   * * * *
## *
## *
## *
##   * * * *
```

...

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example – Letter Memorization (Cont'd)

Generate distorted letters:

```
> mutate <- function(letter.vector, number.pixel.flips){
+   letter.vector[sample(length(letter.vector),
      number.pixel.flips)] <-
      letter.vector[sample(length(letter.vector),
        number.pixel.flips)]
+   return(letter.vector)
+ }

> mutated.C <- mutate(C,8)
> mutated.D <- mutate(D,8)
> mutated.J <- mutate(J,8)
> mutated.M <- mutate(M,8)
```

Example – Letter Memorization (Cont'd)

```
# Draw the mutated letters
> for (i in mget(ls(pattern = "mutated")))
+   {show.letter(i)}

##     *   *
## * *   *
## * *
## *
##   * * * *

...
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example – Letter Memorization (Cont'd)

Store all 4 letter pixels into a 4×25 weight matrix; each letter has 25 pixels. Hopfield network has 25 computational neurons.

```
> x <- matrix(c(C,D,J,M), nrow=4, byrow=T)
```

Function uses Hebbian learning to restore the mutated letters back to original.

```
> hopfield <- function(current.letter, iteration,
    memory=w){
+   w <- t(x) %*% x
+   diag(w) <- 0
+   for (i in 1:iteration){
+     a <- w %*% as.vector(current.letter)
+     current.letter <- ifelse(a>0, 1, -1)
+   }
+   return(show.letter(matrix(current.letter,
      ncol=5, nrow=5)))
+ }
```

Example – Letter Memorization (Cont'd)

Draw all the iterations (5 per letter) of restoration process

```
> for (i in mget(ls(pattern = "mutated"))) {
+   for (iter in 1:5) {
+     hopfield(current.letter = i, iteration = iter)
+   }
+ }
```

```
##   * * * *
## *
## *
## *
##   * *   *
##
##   * * * *
## *
## *
## *
##   * * * *
```

...

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

*Ising Model

- Problem of memory retrieval can be analyzed by equilibrium statistical mechanics of Ising model.
- *Ising model* developed by physicist Ernst Ising in 1924 is one of the simplest model of interacting many-body systems.
- Statistical mechanics has powerful tools to clarify macroscopic properties of many-body systems using the interaction between microscopic particles.

Ising Model

- Let $d \geq 1$. Take undirected graph given by a d-dimensional lattice Λ
- Nodes are *sites*. Assume lattice Λ has N sites.
- For each site $i \in \Lambda$, there is an *Ising spin* given by a discrete variable s_i with spin values in the set $\{\pm 1\}$.
- A *spin configuration* $s = (s_i)$ is an assignment of spin values to the lattice sites. In view of magnetism, Ising spins are microscopic magnetic moments pointing up ($\uparrow$) or down ($\downarrow$).

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

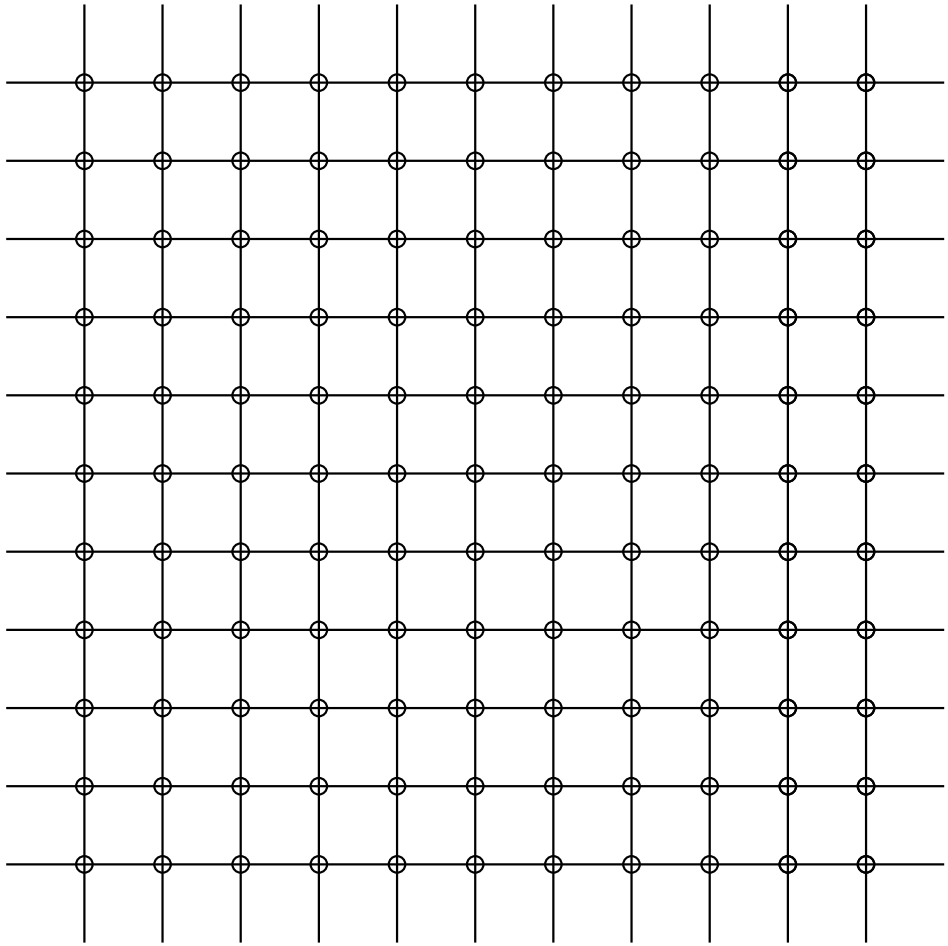
Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Ising Model

- Square 2D lattice



Ising Model

- A *bond* is a pair of adjacent lattice sites (i, j). B denotes the set of bonds.
- *Interaction energy* of bond (i, j) is $-w_{ij}s_is_j$; the interaction is $-w_{ij}$ if $s_i = s_j$ and w_{ij} if $s_i \neq s_j$.
- In view of magnetism, two interacting spins are oriented in the same direction ($\uparrow\uparrow$ or $\downarrow\downarrow$) if $w_{ij} > 0$.
- Positive interaction $w_{ij} > 0$ may lead to macroscopic magnetism (ferromagnetism).

  Negative interaction $w_{ij} < 0$ supports antiparallel states of interacting spins (antiferromagnetism).
- Each site i may have an external magnetic field h_i interacting with it; the term $-h_is_i$ is called *Zeeman energy*. If $h_i > 0$, the spin site i desires to line up in positive direction.

Ising Model

- *Hamiltonian* (total energy) of spin configuration $s = (s_i)$

$$H(s) = -\sum_{(i,j)\in B} w_{ij} s_i s_j - \mu \sum_{i=1}^{N} h_i s_i,$$

where μ denotes the magnetic moment.

- *Boltzmann distribution*: probability of spin configuration $s = (s_i)$

$$P(s) = \frac{e^{-\beta H(s)}}{Z},$$

with inverse temperature $\beta = 1/T$ and Boltzmann factor $e^{-\beta H(s)}$.

Ising Model

- *Partition function* (normalizing constant)

$$Z = \sum_{s} e^{-\beta H(s)} = \sum_{s_1=\pm 1} \cdots \sum_{s_N=\pm 1} e^{-\beta H(s_1,\ldots,s_N)}.$$

The sum over all possible spin configurations is denoted by

$$Z = \mathbf{Tr}\, e^{-\beta H}.$$

Ising Model

- The thermal average of a physical quantity f is computed by using the Boltzmann distribution,

$$\langle f \rangle = \sum_s f(s) P(s).$$

- An order parameter with ferromagnetic interactions is *magnetization*

$$m = \frac{1}{N} \left\langle \sum_{i=1}^{N} s_i \right\rangle = \frac{1}{N} \mathbf{Tr} \left(\left(\sum_{i=1}^{N} s_i \right) P(s) \right).$$

- *Phase transition*: $m \neq 0$ for $T < T_c$ (ferromagnetic phase) and $m = 0$ (paramagnetic phase) for $T > T_c$ for *critical temperature* T_c (transition point).

Ising Model

- Simplification of Ising model: $w = w_{ij}$ for each bond $(i, j) \in B$.
- Hamiltonian

$$H(s) = -w \sum_{(i,j)\in B} s_i s_j - h \sum_i s_i.$$

- Calculation of expectation value $\langle f \rangle$ is difficult, since partition function Z has 2^N terms.
- Approximation by *mean-field theory*.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Ising Model - Meanfield Theory

- Spin variable s_i splits into mean (magnetization) $m = \langle s_i \rangle N$ and deviation (fluctuation) $\delta s_i = s_i - m$.
- Hamiltonian

$$H(s) = -w \sum_{(i,j) \in B} (m + \delta s_i)(m + \delta s_j) - h \sum_i s_i.$$

- If the fluctuation is small, the Hamiltonian becomes

$$H(s) = -wmN_B - wm \sum_{(i,j) \in B} (\delta s_i + \delta s_j) - h \sum_i s_i,$$

where $N_B = |B|$ is the number of bonds.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Ising Model - Meanfield Theory

- *Coordination number* z is the number of sites adjacent to a site. Then $N_B = zN/2$.
- Hamiltonian

$$\begin{aligned} H(s) &= -wm^2 N_B - wmz \sum_i \delta s_i - h \sum_i s_i \\ &= wm^2 N_B - (wmz + h) \sum_i s_i. \end{aligned}$$

  System looks like a collection of non-interacting spins; effects of interaction between spins are shifted into magnetization.

- Partition function

$$Z = \mathbf{Tr} \exp\left(\beta\left(-wm^2 N_B + (wmz + h)\sum_i s_i\right)\right).$$

Ising Model - Meanfield Theory

- Partition function

$$
\begin{aligned}
Z &= \mathbf{Tr}\ \exp\left(\beta\left(-wm^2N_B + (wmz+h)\sum_i s_i\right)\right) \\
&= e^{-\beta w m^2 N_B}\,\mathbf{Tr}\ \exp\left(\beta(wmz+h)\sum_i s_i\right) \\
&= e^{-\beta w m^2 N_B}\,\left(2\cosh\beta(wmz+h)\right)^N .
\end{aligned}
$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Ising Model - Meanfield Theory

Proof (last equality):

$$
\begin{aligned}
Z_N &= \mathbf{Tr}\ \exp\left(\beta(wmz+h)\sum_i s_i\right) \\
&= \sum_{s_1=\pm 1} \cdots \sum_{s_N=\pm 1} e^{\beta(wmz+h)\sum_i s_i} \\
&= \sum_{s_1=\pm 1} \cdots \sum_{s_N=\pm 1} e^{\beta(wmz+h)s_1} \cdots e^{\beta(wmz+h)s_N} \\
&= \sum_{s_1=\pm 1} e^{\beta(wmz+h)s_1} \cdots \sum_{s_N=\pm 1} e^{\beta(wmz+h)s_N} \\
&= \left(\sum_{s=\pm 1} e^{\beta(wmz+h)s}\right)^N = Z_1^N.
\end{aligned}
$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Ising Model - Meanfield Theory

Proof (cont'd):

$$\begin{aligned} Z_1 &= \sum_{s=\pm 1} e^{\beta(wmz+h)s} = e^{\beta(wmz+h)} + e^{-\beta(wmz+h)} \\ &= 2\cosh\beta(wmz+h). \end{aligned}$$

Thus

$$Z_N = \left(2\cosh\beta(wmz+h)\right)^N.$$

□

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Ising Model - Meanfield Theory

Magnetization

$$m = \frac{\textbf{Tr}\ s_i e^{-\beta H}}{Z} = \tanh \beta(wmz + h).$$

Proof:

$$\begin{aligned} Z'_N &= \textbf{Tr}\ s_i e^{-\beta H} \\ &= \sum_{s_1=\pm 1} s_1 \ldots \sum_{s_N=\pm 1} s_N e^{\beta(wmz+h)\sum_i s_i} \\ &= \sum_{s_1=\pm 1} s_1 e^{\beta(wmz+h)s_1} \ldots \sum_{s_N=\pm 1} s_N e^{\beta(wmz+h)s_N} \\ &= \left(\sum_{s=\pm 1} s e^{\beta(wmz+h)s} \right)^N = {Z'}_1^N. \end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

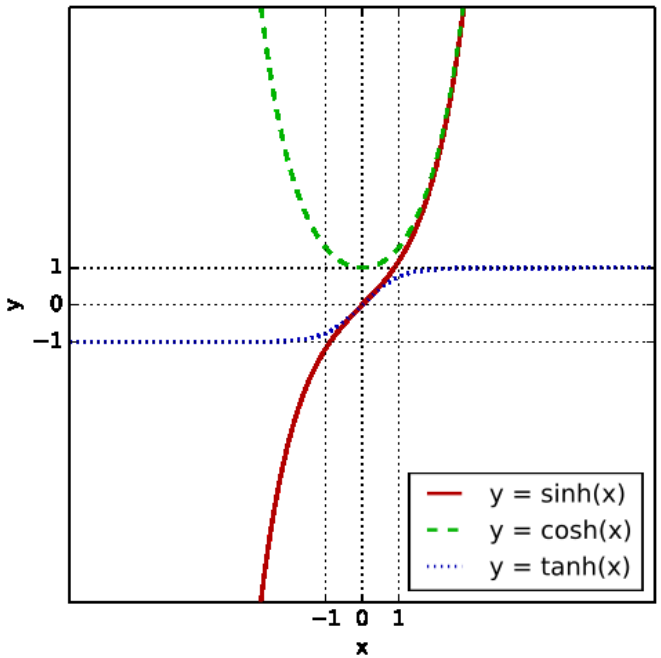
Neural Networks in R

Ising Model - Meanfield Theory

Proof (cont'd):

$$\begin{aligned} Z_1' &= \sum_{s=\pm 1} s e^{\beta(wmz+h)s} = e^{\beta(wmz+h)} - e^{-\beta(wmz+h)} \\ &= 2\sinh\beta(wmz+h) \end{aligned}$$

with $\tanh(x) = \frac{\sinh(x)}{\cosh(x)}$. Hyperbolic functions:



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Ising Model - Meanfield Theory

- *Equation of state* determines the order parameter

$$m = \frac{\textbf{Tr}\ s_i e^{-\beta H}}{Z} = \tanh \beta(wmz + h).$$

- *Spontaneous magnetization* in the absense of external field $h = 0$,

$$m = \tanh \beta(wmz).$$

- Consider the expansion

$$\tanh x = x - \frac{x^3}{3} + \frac{2x^5}{15} - \frac{17x^7}{315} \pm \ldots, \quad |x| < \frac{\pi}{2},$$

- Taking the first term in the expansion of $\tanh \beta wmz$ gives

$$m = \beta wmz$$

which yields the critical temperature $T_c = wz$.

Ising Model - Meanfield Theory

- *Free energy*:

  $$F = -T \log Z = -NT \log \left(2 \cosh \beta(wmz + h)\right) + N_B w m^2.$$

  Expand RHS in powers of m to fourth order,

  $$F = -NT \log 2 + \frac{wzN}{2}(1 - \beta wz)m^2 + \frac{N}{12}(wmz)^4 \beta^3.$$

  The coefficient of m^2 alters the sign at $T_c = wz$.

- The free energy has two minima located at $m \neq 0$ if $T < T_2$ and one minimum at $m = 0$ if $T > T_c$.
- The magnetization in thermal equilibrium is zero if $T > T_c$ and is nonzero if $T < T_c$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Hopfield Model

- The number of embedded patterns p is finite.
- Partition function is described by Hamiltonian,

$$Z = \mathbf{Tr}\ \exp\left(\frac{\beta}{2N}\sum_{\mu}\left(\sum_{i} s_i \xi_i^{(\mu)}\right)^2\right).$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Hopfield Model

- New integration variable m^μ to linearize the square of the exponent,

$$\begin{aligned} Z &= \mathbf{Tr} \int \prod_{\mu=1}^{p} dm^{(\mu)} \exp\left(-\frac{1}{2} N\beta \sum_\mu m_\mu^2 + \beta \sum_\mu m_\mu \sum_i s_i \xi_i^{(\mu)}\right) \\ &= \int \prod_{\mu=1}^{p} dm^{(\mu)} \exp\left(-\frac{1}{2} N\beta \langle m, m\rangle + \sum_i \log(2\cosh \beta \langle m, \xi_i\rangle)\right), \end{aligned}$$

  where $m = (m^{(1)}, \ldots, m^{(p)})^t$, $\xi_i = (\xi_i^{(1)}, \ldots, \xi_i^{(p)})^t$, $1 \le i \le N$, and the multiplicative constant has been deleted as it does not influence the overall physical behavior.

- In the thermodynamic limit $N \to \infty$, the free energy per degree of freedom N is

$$f = -\frac{T}{N} \log \mathbf{Tr}\, e^{-\beta H} = \frac{1}{2}\langle m, m\rangle - \frac{T}{N} \sum_i \log(2\cosh \beta \langle m, \xi_i\rangle).$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Hopfield Model

- Equation of state

$$m = \frac{1}{N} \sum_i \xi_i \tanh \beta \langle m, \xi_i \rangle.$$

- In the thermodynamic limit, the sum over all neurons i becomes equivalent to the average over the random components of the vector $\xi = (\xi^{(1)}, \ldots, \xi^{(p)})$ which corresponds to the configurational average denoted by $[\cdot]$.
- Configurational average of relative free energy

$$\begin{aligned} [f] &= -T[\log Z] = -T \int \prod_{(ij)} w_{ij} P(w_{ij}) \log Z \\ &= \frac{1}{2} \langle m, m \rangle - T[\log(2 \cosh \beta \langle m, \xi \rangle)]. \end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Hopfield Model

- Equation of state

$$m = [\xi \tanh \beta \langle m, \xi \rangle].$$

- Retrieving the first pattern gives $m = m_1$ and $m_2 = \ldots = m_p = 0$. This gives the mean-field solution with $h = 0$,

$$m = [\xi^{(1)} \tanh \beta \langle m, \xi^{(1)} \rangle] = \tanh \beta m.$$

- There are two solution $m \neq 0$ for $T = \beta^{-1} < 1$ (red line); otherwise, there is one solutions (blue line).
- If $T = 0$, the stable state is $m = \pm 1$ and a perfect retrieval of the embedded pattern (or its complement) is achieved.
- The Hopfield model with a finite number of patterns works at low temperature as an *associative memory* which is able to retrieve the appropriate memorized pattern if a noisy variant of the pattern is intially given.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

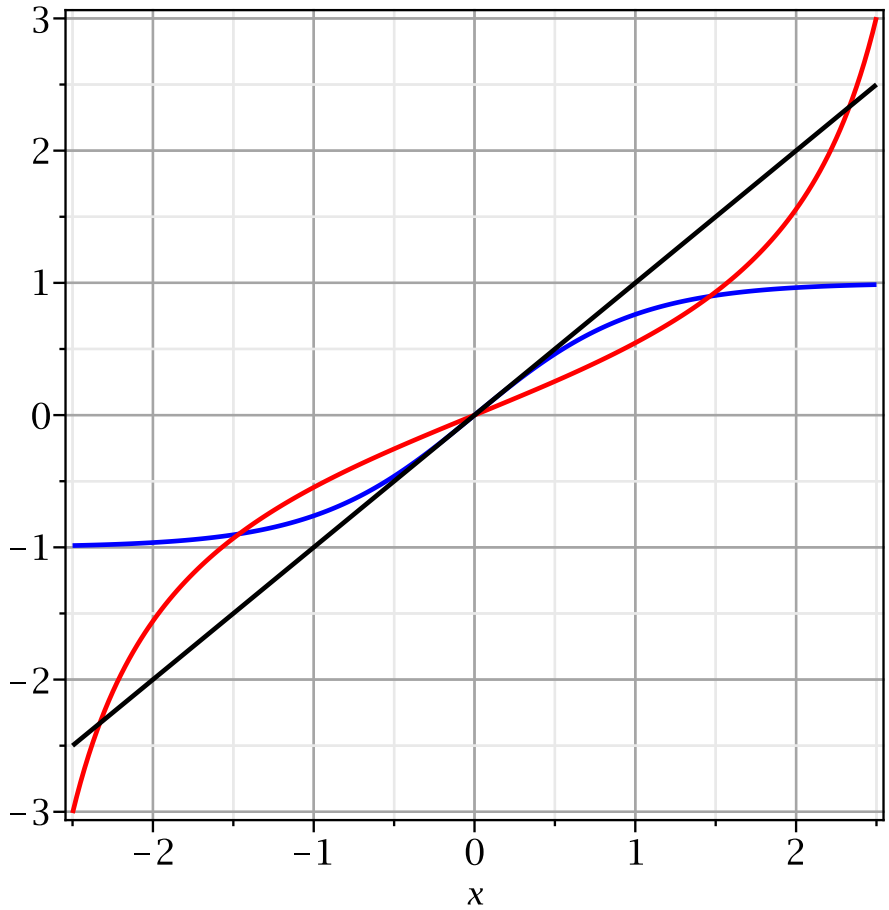
Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Hopfield Model

The equation $x = \tanh \beta x$ has one solution (blue) or two solutions (red).



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Self-Organizing Maps

- Topographic maps
- Kohonen network
- Learning
- Example: SOM for Iris dataset
- *Iterated local search
- *Principal component analysis

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Self-Organizing Maps (SOM)

- SOM provides data compression mapping high-dimensional sample space to low-dimensional feature space.
- SOM is established by *unsupervised learning* (no teacher).
- Assumption: Class membership is inherent in the input patterns defining common features and the network is able to identify some of those features during the training session.
- SOM and k-means are identical if the radius of the neighborhood function in SOM is identical to zero.
- Large self-organizing maps may exhibit *emergent properties*. Emergent properties are phenomena in which larger entities arise through the interactions of smaller or simpler entities such that the larger entities show properties the smaller entities do not show, like macroscopic physical properties or swarming.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Topographic Map

- Ordered projection of sensory surface (retina, skin, eardrum) to structure of central nervous system.
- Construct artificial topographic map by learning via self-organization in neurobiologically inspired manner.
- Nonlinear generalization of principal component analysis (PCA).

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Kohonen Networks

- Form a class of self-organizing maps (Teuvo Kohonen, 1980).
- Kohonen network is an ANN with
  - feed-forward topology,
  - single input layer,
  - single computational layer,
  - neurons arranged in one- or two-dimensional regularly spaced grid.

  Each neuron in input layer is connected to each neuron in computational layer.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

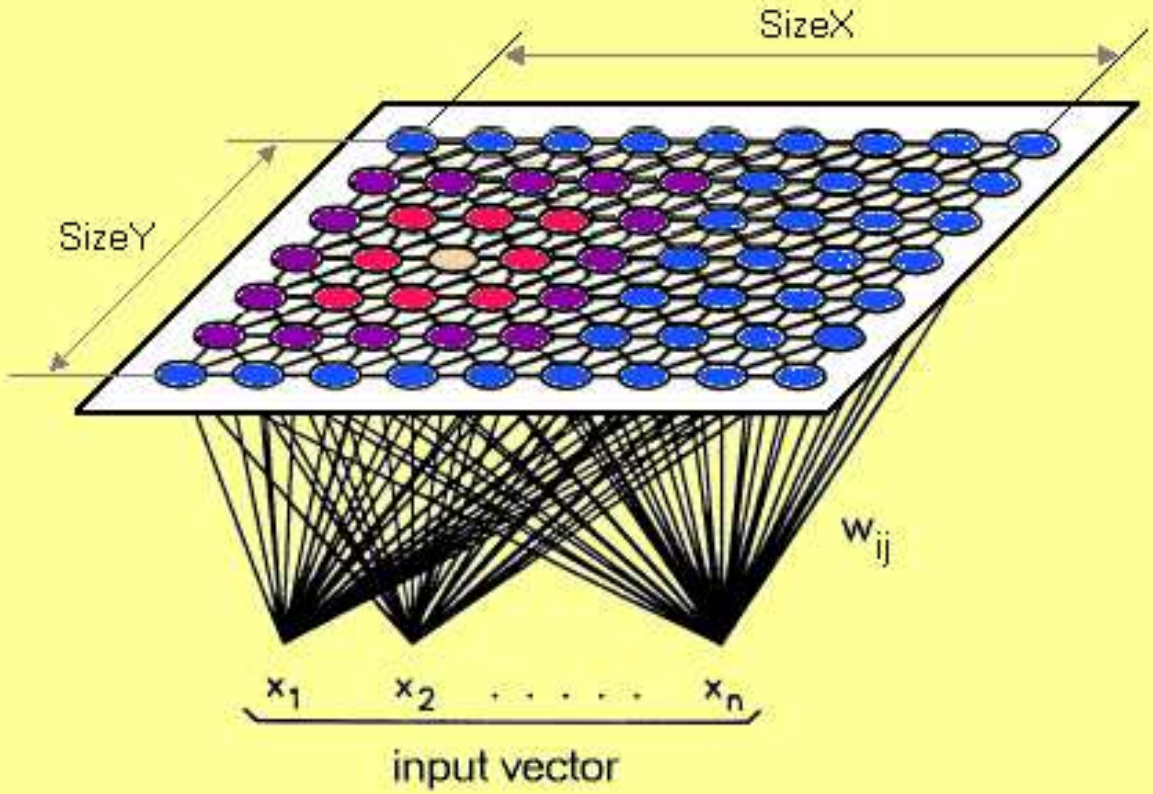
Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Kohonen Network



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Kohonen Network – Learning

Self-organization procedure:

- *Initialization*: The connecting weights are initialized with small values around zero chosen uniformly at random.
- *Competition*: For each input pattern, each neuron computes its value of a discriminant function and the neuron with the smallest value is declared as the winner.
- *Cooperation*: The winning neuron determines the spatial location of a topographic neighborhood of excited neurons.
- *Adaption*: The excited neurons decrease their values of the discriminant function by adjusting their weights such that the response of the winning neuron is enhanced for similar input patterns.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Kohonen Network – Learning

- Input patterns are vectors in $\mathbb{R}^d$.
- Computational layer has N neurons.
- w_{ij} = weight between input neuron j and neuron i in computational layer, $1 \leq i \leq N$, $1 \leq j \leq d$.
- Winning neuron (closest to input pattern x)

$$I(x) = \arg\min\{d_i(x) = \sum_{j=1}^{d}(x_j - w_{ij})^2 \mid 1 \leq i \leq N\}. \quad (435)$$

- Neurons in the topographic neighborhood tend to get excited too; excitation decays with the distance from the winning neuron.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Kohonen Network – Learning

- D_{ik} = distance between neurons i and k in computational layer.
- Topographic neighborhood of winning neuron (Gaussian function)

$$T_{i,I(x)} = \exp\left(-\frac{D_{i,I(x)}^2}{2\sigma^2}\right), \quad 1 \leq i \leq N. \tag{436}$$

- Time adaptive neighborhood decreases with time

$$\sigma(t) = \sigma_0 \exp(-t/\tau_\sigma) \tag{437}$$

for some $\sigma_0, \tau_\sigma > 0$.

- $T_{i,I(x)}$ is maximal for neuron $I(x)$ and decreases monotonously to 0 as the distance from $I(x)$ becomes large.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Kohonen Network – Learning

- Weight update of neurons in the topographic neighborhood of winning neuron,

$$\Delta w_{ij} = \eta(t) \cdot T_{i,I(x)} \cdot (x_j - w_{ij}) \tag{438}$$

- Time adaptive learning rate decreases with time

$$\eta(t) = \eta_0 \exp(-t/\tau_\eta) \tag{439}$$

  for some $\eta_0, \tau_\eta > 0$

- Aim is to move the weight vectors w_i of neurons in the topographic neighborhood of the winning neuron closer to the input pattern.
- Proper selection of parameters $(\sigma_0, \tau_\sigma, \eta_0, \tau_\eta)$ by iterated local search.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Kohonen Network – Learning Algorithm

Require: Kohonen network with input layer of d neurons and computational layer of N neurons, parameter set $(\sigma_0, \tau_\sigma, \eta_0, \tau_\eta)$, training patterns $x^{(1)}, \ldots, x^{(M)}$ in $\mathbb{R}^d$, small number $\epsilon > 0$

Ensure: Trained Kohonen network for feature extraction

Choose random weights w_{ij}

repeat

Draw sample input pattern $x = x^{(l)}$ for some $1 \leq l \leq M$.

Compute the winning neuron $I(x)$ in computational layer

Update the weights by the increments Δw_{ij}

until Magnitudes of all weight increments are smaller than ϵ during the last M steps.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example: SOM for Iris Dataset

The 150-sample Iris dataset is mapped to a 5×5 grid of hexagonal units:

```
# install the kohnen package
> install.packages("kohonen")
# load the kohonen package
> library(kohonen)
# scale the iris data
> iris.sc = scale(iris[,1:4])
# build the grid
> iris.grid = somgrid(xdim=5,ydim=5,topo="hexagonal")
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Example: SOM for Iris Dataset (Cont'd)

Generate the model:

```
> iris.som = som(iris.sc, grid=iris.grid, rlen=100,
                 alpha=(0.05,0.01))
```

The som function has several parameters:

- grid: rectangular or hexagonal group of units (format is returned by somegrid).
- rlen: number of iterations the dataset is presented to the network.
- alpha: learning rate, start with 0.05 and stop at 0.01.

Variable iris.som$codes (codebook) holds the weight vector for each neuron in the grid.
Variable iris.som$changes indicates the size of adaptions to the codebook vectors during training.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

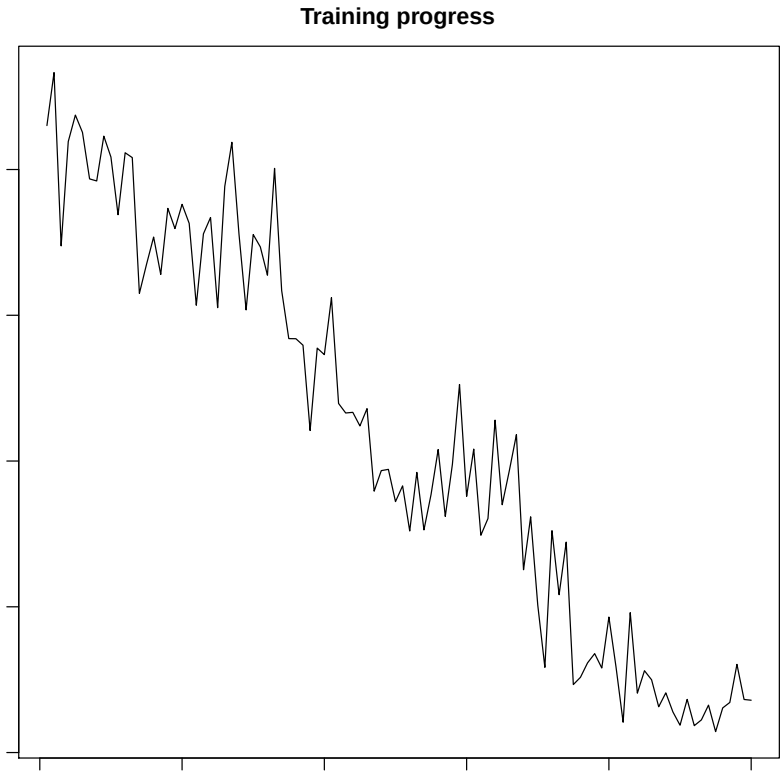
*Spiking Neural Networks

Neural Networks in R

Example: SOM for Iris Dataset (Cont'd)

Training progress (number of iterations vs. mean distance to closest unit):

```
> plot(iris.som, type="changes")
```



Iteration vs. mean distance to closest unit.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

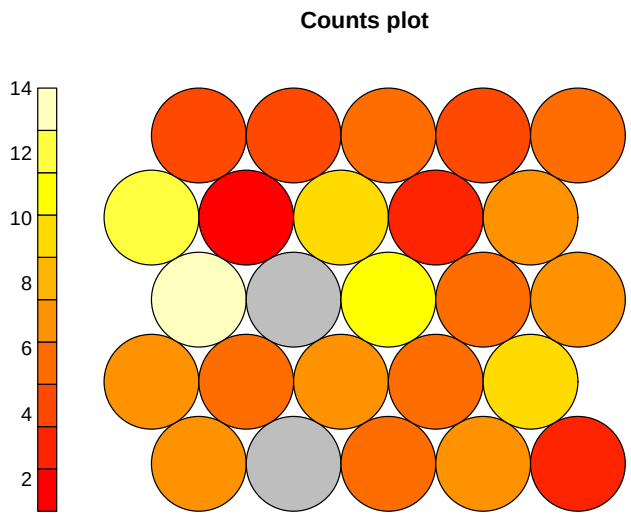
*Spiking Neural Networks

Neural Networks in R

Example: SOM for Iris Dataset (Cont'd)

Node counts (how many samples are mapped to each node on the map):

```
> plot(iris.som, type="count")
```



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

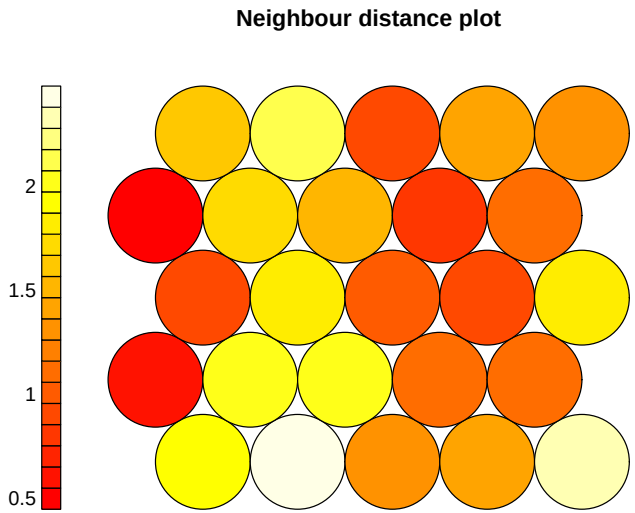
*Spiking Neural Networks

Neural Networks in R

Example: SOM for Iris Dataset (Cont'd)

Neighbor distance (U-matrix, distance between each node and its neighbours on the map):

```
> plot(iris.som, type="dist.neighbours")
```



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

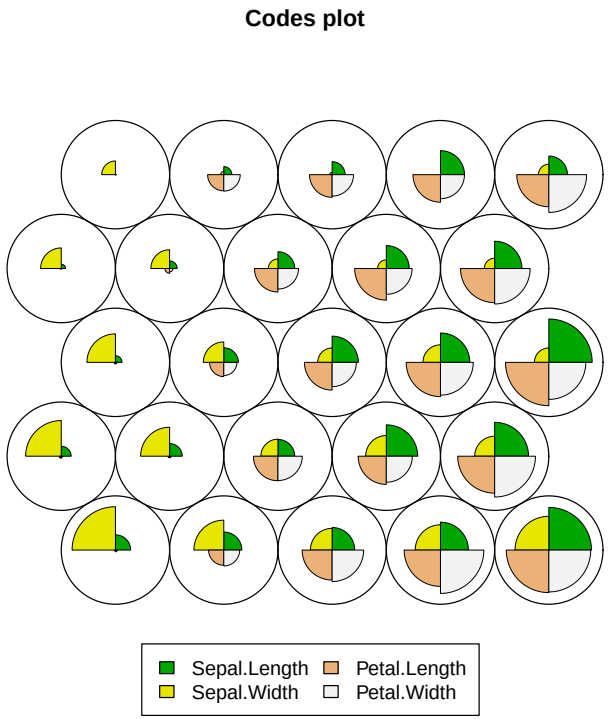
*Spiking Neural Networks

Neural Networks in R

Example: SOM for Iris Dataset (Cont'd)

Codes/Weight vectors (normalized values of the original variables used to generate SOM):

```
> plot(iris.som, type="codes")
```



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

SOM Learning and Clustering

- Intuitive method for segmentation.
- Simple algorithm, easy to explain to non-data scientists.
- New data can be mapped to trained model for prediction.
- Lack of parallelization for very large datasets.
- Difficult to map a collection of variables to 2D plane.
- Requires numerical data.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

*Iterated Local Search (ILS)

- Stochastic search method for tuning the parameters of heuristic algorithm.
- Take fixed parameter set $\theta_b \in \Theta$ as *baseline*.
- The ILS algorithm determines the best $k \geq 1$ parameter sets on training set of the heuristic.
- These parameter sets are compared with the baseline by *Wilcoxon T test*.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Iterated Local Search

Require: Parameter space Θ, heuristic $A = A(\theta)$, integer $K \geq 1$, small number $\epsilon > 0$

Ensure: Best parameter set $\theta^* \in \Theta$ found

Choose initial parameter set $\theta \in \Theta$; $\theta^* \leftarrow \theta$

repeat

Take new parameter set θ' from the neighborhood of θ

if $A(\theta') < A(\theta)$ **then**

$\theta \leftarrow \theta'$

if $A(\theta') < A(\theta^*)$ **then**

$\theta^* \leftarrow \theta'$

end if

else

Draw random number $r \in [0, 1]$ uniformly at random

if $r < \exp(-(A(\theta') - A(\theta)))$ **then**

$\theta \leftarrow \theta'$

end if

end if

until Improvements $|A(\theta') - A(\theta)| < \epsilon$ during last K steps

return θ^*

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Iterated Local Search

Stochastic search method for tuning the parameters of heuristic algorithm.

- Set of N samples (x_i, y_i) with $x_i \neq y_i$, $1 \leq i \leq N$.
- Calculate $|x_i - y_i|$ and $\operatorname{sgn}(y_i - x_i)$ for $1 \leq i \leq N$.
- Rank the pairs beginning with the smallest as 1.
- $r_i =$ rank of pair (x_i, y_i), $1 \leq i \leq N$.
- Here two self-organizing networks A and B with the same input set configured with different parameter sets could be compared by their weight vectors $(w_{ij}^{(A)})$ and $(w_{ij}^{(B)})$.

  Under the null hypothesis, the networks can be discriminated whether they provide significantly different feature maps or not.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Wilcoxon T-Test

Non-parametric statistical hypothesis test for determining if two independent sample sets were selected from populations with the same distribution.

- Null hypothesis H_0: difference between the pairs follows symmetric distribution around zero.
- Calculate test statistic (as sum of signed ranks)

$$T = \sum_{i=1}^{N} \operatorname{sgn}(y_i - x_i) \cdot r_i. \tag{440}$$

- Under the null hypothesis H_0, T has specific distribution with expected value of 0 and variance of $\frac{N(N+1)(2N+2)}{6}$.
- Compare T with critical value from reference table such that null hypothesis is rejected if $|T| > T_{\text{crit},N}$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Iterated Local Search – Example (R)

Consider data set `immer` (with sample size $N = 30$) recording the barley yield in the years 1931 and 1932 from the same field.

```
> library(MASS)    # load the MASS package
> head(immer)
  Loc Var    Y1    Y2
1 UF   M  81.0  80.0
2 UF   S 105.4  82.3
3 UF   V 119.7  80.4
4 UF   T 109.7  87.2
5 UF   P  98.3  84.2
6  W   M 146.6 100.4
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Iterated Local Search – Example (R)

Wilcoxon T-test at 0.05 significance level:

```
> wilcox.test(immer$Y1,immer$Y2,paired=TRUE)

Wilcoxon signed rank test without continuity correction

data: immer$Y1 and immer$Y2
V = 368.5, p-value = 0.005318
alternative hypothesis: true location shift is not equal

warning message:
In wilcox.test.default(immer$Y1,immer$Y2,paired=TRUE):
 cannot compute exact p-value with ties
```

p-value $0.005318 < 0.05$ significance level
$\Rightarrow$ Strong evidence against the null hypothesis (rejected).
$\Rightarrow$ Barley yields from 1931 and 1932 are from different populations.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

*Principal Component Analysis (PCA)

- PCA (multivariate statistical technique) for the analysis of correlated data set.
- PCA (Karl Pearson, 1857-1936) is an application of singular value decomposition (SVD).
- The objective is to extract the important information from a data set in form of a set of orthogonal data.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Principal Component Analysis (PCA)

- Given $A = (a_{ij})$ real-valued $n \times d$ matrix with rows as n (data) points in $\mathbb{R}^d$.
- Find the best k-dimensional subspace of $\mathbb{R}^d$ with respect to the points.
- Best means optimizing the sum of squares of the perpendicular distances of the points to the subspace.
- Euclidean norm of vector v,

$$\|v\| = \sqrt{\sum_i v_i^2}. \tag{441}$$

- Euclidean distance between vectors u and v,

$$d(u,v) = \|u - v\| = \sqrt{\sum_i (u_i - v_i)^2}. \tag{442}$$

Soft Computing
K.-H. Zimmermann
Contents
ANN Technology
Multilayer Perceptron
Deep Learning via Keras
Universality
Hopfield Network
Self-Organizing Networks
*Spiking Neural Networks
Neural Networks in R

Principal Component Analysis (PCA)

- *First singular vector* v_1 of A,

$$v_1 = \arg\max\{\|Av\| \mid v \in \mathbb{R}^d, \|v\| = 1\}. \tag{443}$$

Best-fit line through the origin for the n points in $\mathbb{R}^d$ given by the rows of the matrix A.

- *First singular value* of A,

$$\sigma_1(A) = \|Av_1\|. \tag{444}$$

Let $a_1, \ldots, a_n$ denote the rows of the matrix A. Then

$$\|Av_1\|^2 = \sum_{i=1}^{n} \langle a_i, v_1 \rangle^2, \tag{445}$$

where $|\langle a_i, v_1 \rangle|$ is the length of the projection of a_i onto v_1. σ_1^2 is the sum of the squares of the projections of the given points to the line determined by v_1.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Principal Component Analysis (PCA)

- Continuing this way establishes singular vectors $v_1, v_2, \ldots, v_r$,

  $$\arg\max\{\|Av\| \mid v \in \mathbb{R}^d, \|v\| = 1, v \perp v_1, \ldots, v \perp v_r\} = 0 \quad (446)$$

  The vectors $v_1, \ldots, v_r$ form an orthonormal basis of a subspace V of $\mathbb{R}^d$.

- The assignment

  $$\pi_V(w) = \sum_{i=1}^{r} \langle w, v_i \rangle v_i, \quad w \in \mathbb{R}^d, \quad (447)$$

  gives a linear mapping called *orthogonal projection* from $\mathbb{R}^d$ onto V.

- Any vector $w \in \mathbb{R}^d$ can be uniquely written as $w = v + v'$, where $v \in V$ and $v' \in V^\perp$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Proposition

Let A be a real-valued $n \times d$ matrix with singular vectors $v_1, \ldots, v_r$.

For each $1 \leq k \leq r$, let V_k be the subspace of $\mathbb{R}^d$ spanned by the vectors $v_1, \ldots, v_k$.

Then V_k is the best-fit k-dimensional subspace for A.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Proof.

The assertion is clear for $k = 1$. By induction, let V_{k-1} be a best-fit $k-1$-dimensional subspace for A. Suppose W is a best-fit k-dimensional subspace for A. Choose a basis $\{w_1, \ldots, w_k\}$ of W such that w_k is perpendicular to $v_1, \ldots, v_{k-1}$. Then

$$\begin{aligned} &\|Aw_1\|^2 + \ldots + \|Aw_{k-1}\|^2 + \|Aw_k\|^2 \\ &\quad \leq \|Av_1\|^2 + \ldots + \|Av_{k-1}\|^2 + \|Aw_k\|^2, \end{aligned}$$

since V_{k-1} is optimal. But the vector w_k is perpendicular to $v_1, \ldots, v_{k-1}$ and so by definition of v_k we have $\|Aw_k\|^2 \leq \|Av_k\|^2$. Thus

$$\begin{aligned} &\|Aw_1\|^2 + \ldots + \|Aw_{k-1}\|^2 + \|Aw_k\|^2 \\ &\quad \leq \|Av_1\|^2 + \ldots + \|Av_{k-1}\|^2 + \|Av_k\|^2. \end{aligned}$$

Thus V_k is as least as good as W and hence optimal. □

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Principal Component Analysis (PCA)

- The image of the subspace of singular vectors of A is spanned by the vectors

$$u_i = \frac{1}{\sigma_i(A)} A v_i, \quad 1 \leq i \leq r. \tag{448}$$

- The vectors u_i are normalized, since $\|Av_i\| = \sigma_i(A)$ for $1 \leq i \leq r$.
- The vectors u_i are the *left singular vectors* of A, and the singular vectors v_i of A are the *right singular vectors* of A.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Proposition

Let A be a real-valued $n \times d$ matrix A of rank r.

The left singular vectors $u_1, \ldots, u_r$ of A are mutually orthogonal and we have

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^t. \tag{449}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Proof

The assertion holds for $r = 1$. By induction, suppose the result holds for matrices of rank $r - 1$. Consider the matrix

$$B = A - \sigma_1 u_1 v_1^t.$$

Then $Bv_1 = Av_1 - \sigma_1 u_1 v_1^t v_1 = Av_1 - \sigma_1 \frac{1}{\sigma_1}(Av_1)(v_1^t v_1) = 0$. The first right singular vector w of B is perpendicular to v_1. But for each vector w perpendicular to v_1, we have $Bw = Aw$. Thus the first singular vector of B equals the second singular vector of A. By repeating this argument, B has the right singular vectors $v_2, \ldots, v_r$ and the left singular vectors $u_2, \ldots, u_r$, which are mutually orthogonal. Moreover, by induction,

$$B = \sum_{i=2}^{r} \sigma_i u_i v_i^t$$

and so $A = \sum_{i=1}^{r} \sigma_i u_i v_i^t$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Proof (cont'd)

The vector u_1 is orthogonal to the vectors $u_2, \ldots, u_r$. To see this, assume that $\langle u_1, u_i \rangle \neq 0$ for some $2 \leq i \leq r$; suppose that $\langle u_1, u_i \rangle > 0$. Then for some infinitesimally small number $\epsilon > 0$, the vector

$$A\left(\frac{v_1 + \epsilon v_i}{\|v_1 + \epsilon v_i\|}\right) = \frac{\sigma_1 u_1 + \epsilon \sigma_i u_i}{\sqrt{1+\epsilon^2}}.$$

has length as least as large as its component along u_1, which is given by

$$\begin{aligned}
&\left\langle u_1, \frac{\sigma_1 u_1 + \epsilon \sigma_i u_i}{\sqrt{1+\epsilon^2}} \right\rangle \\
&= (\sigma_1 + \epsilon \sigma_i \langle u_1, u_i \rangle)\left(1 - \frac{\epsilon^2}{2} + O(\epsilon^4)\right) \\
&= \sigma_1 + \epsilon \sigma_i \langle u_1, u_i \rangle - O(\epsilon^2) > \sigma_1.
\end{aligned}$$

This contradicts the hypothesis. □

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Singular Value Decomposition

Let A be a real-valued $n \times d$ matrix with right singular vectors $v_1, \ldots, v_r$, left singular vectors $u_1, \ldots, u_r$, and singular values $\sigma_1, \ldots, \sigma_r$.

Then the matrix A has a decomposition into a sum of rank-one matrices,

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^t. \tag{450}$$

In matrix form, the singular value decomposition is

$$A = UDV^t, \tag{451}$$

where U is the $n \times r$ matrix whose columns contain the left singular vectors of A, D is the $r \times r$ diagonal matrix whose diagonal entries are the singular values of A, and V is the $r \times d$ matrix whose columns consists of the right singular vectors of A.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Singular Value Decomposition – Maple

Maple code

```
> with(LinearAlgebra):
> with(RandomTools):
> n := 4: d := 2:
> A := Matrix(n,d,generator=rand(1..5)):
> U,S,V := SingularValues(A,output=['U','S','Vt']):
> U, S, V;
```

We have $U \cdot D \cdot V = A$, where diagonal matrix D has the entries of S along its main diagonal.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Singular Value Decomposition – Maple (cont'd)

$$A = \begin{pmatrix} 5 & 2 \\ 5 & 2 \\ 3 & 4 \\ 4 & 5 \end{pmatrix}$$

$$U = \begin{pmatrix} -0.48 & -0.52 & -0.43 & -0.56 \\ -0.48 & -0.52 & 0.46 & 0.53 \\ -0.45 & 0.45 & 0.60 & -0.49 \\ -0.58 & 0.51 & -0.49 & 0.40 \end{pmatrix}$$

$$S = \begin{pmatrix} 10.75 \\ 2.90 \\ 0 \\ 0 \end{pmatrix}$$

$$V = \begin{pmatrix} -0.79 & -0.62 \\ -0.62 & 0.79 \end{pmatrix}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Singular Value Decomposition

Let A be a real-valued $n \times d$ matrix with right singular vectors $v_1, \ldots, v_r$, left singular vectors $u_1, \ldots, u_r$, and singular values $\sigma_1, \ldots, \sigma_r$.

- For each $1 \leq k \leq r$, take the truncated sum of A up to the k-th term,

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^t. \tag{452}$$

  The matrix A_k has rank k.

- The *2-norm* of a matrix A is given by

$$\|A\|_2 = \max\{\|Av\| \mid \|v\| = 1\}. \tag{453}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Proposition

We have $\|A - A_k\|_2^2 = \sigma_{k+1}^2$.

Proof.

We have $A - A_k = \sum_{i=k+1}^{r} \sigma_i u_i v_i^t$. Let v be the first singular vector of $A - A_k$. Then v is a linear combination of $v_1, \ldots, v_r$. Write $v = \sum_{i=1}^{r} \alpha_i v_i$. Then

$$\begin{aligned} \|(A - A_k)v\| &= \| \sum_{i=k+1}^{r} \sigma_i u_i v_i^t \sum_{j=1}^{r} \alpha_j v_j \| = \| \sum_{i=k+1}^{r} \alpha_i \sigma_i u_i v_i^t v_i \| \\ &= \| \sum_{i=k+1}^{r} \alpha_i \sigma_i u_i \| = \sqrt{\sum_{i=k+1}^{r} \alpha_i^2 \sigma_i^2}. \end{aligned}$$

A vector v maximizing the last expression subject to $\|v\| = 1$ has $\alpha_{k+1} = 1$ and $\alpha_i = 0$ for $k + 2 \leq i \leq r$. □

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Theorem

Let A be an $n \times d$ matrix.
For each $n \times d$ matrix B of rank $\leq k$,

$$\|A - A_k\|_2 \leq \|A - B\|_2. \tag{454}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Proof

If the matrix A has rank $\leq k$, then $\|A - A_k\|_2 = 0$ and so the result follows.
Assume the matrix A has rank $> k$. Suppose B is an $n \times d$ matrix of rank $\leq k$ such that $\|A - B\|_2 < \sigma_{k+1}$. Then the kernel W of the matrix B has dimension $\geq d - k$. Let V be the subspace of $\mathbb{R}^d$ spanned by the first $k+1$ singular vectors $v_1, \ldots, v_{k+1}$ of A. Since the subspace V has dimension $k+1$, the subspaces V and W cannot trivially intersect. Thus there is a nonzero vector w in W. We may assume that w is a unit vector; i.e., $\|w\| = 1$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Proof (cont'd)

By definition, $\|A-B\|_2^2 \geq \|(A-B)w\|^2$. Since $Bw=0$, we obtain $\|A-B\|_2^2 \geq \|Aw\|^2$. Since w lies in V,

$$\begin{aligned}
\|Aw\|^2 &= \|\sum_{i=1}^{r} \sigma_i u_i v_i^t w\|^2 \\
&= \|\sum_{i=1}^{r} \sigma_i \langle v_i, w\rangle u_i\|^2 \\
&= \sum_{i=1}^{k+1} \sigma_i^2 \langle v_i, w\rangle^2 \\
&\geq \sigma_{k+1}^2 \sum_{i=1}^{k+1} \langle v_i, w\rangle^2 = \sigma_{k+1}^2.
\end{aligned}$$

Thus $\|A-B\|_2^2 \geq \sigma_{k+1}^2$ and hence the result follows. □

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Applications of Singular Value Decomposition

- Data compression
- Spectral decomposition
- Principal component analysis

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Data Compression

- Let $A = (a_{ij})$ be an $n \times n$ matrix, where a_{ij} denotes the intensity of the pixel (i, j).
- Complexity of storing matrix A is $O(n^2)$.
- Take the approximation matrix A_k given by the first k singular values and the first k left and right singular vectors.
- Complexity of storing matrix A_k is $O(kn)$ with $k \leq n$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Spectral Decomposition

- Let A be a real-valued $n \times d$ matrix.
- The $n \times n$ matrix $B = AA^t$ is symmetric and positive definite.
- If $A = \sum_{i=1}^{r} \sigma_i u_i v_i^t$ is the singular value decomposition, then $B = \sum_{i=1}^{r} \sigma_i^2 u_i u_i^t$.

Proof.

We have

$$\begin{aligned} B = AA^t &= \left(\sum_{i=1}^{r} \sigma_i u_i v_i^t\right)\left(\sum_{i=1}^{r} \sigma_i u_i v_i^t\right)^t \\ &= \sum_{i=1}^{r}\sum_{j=1}^{r} \sigma_i \sigma_j u_i v_i^t v_j u_j^t \\ &= \sum_{i=1}^{r} \sigma_i^2 u_i u_j^t. \end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Principle Component Analysis

- Suppose there are customer-product data where n customers can buy d products.
- Suppose $n \times d$ matrix $A = (a_{ij})$ represents customer-product behavior; a_{ij} is the amount (probability) of product j bought by customer i.
- Which basic k factors (like sex, age, income) determine the customer's purchase behavior?
- Consider the best rank k approximation A_k corresponding to the first k singular values.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Iris Flower Data Set

- Multivariate dataset established by Ronald Fisher (1890-1962) in 1936.
- Data set consists of 150 samples from three Iris species: *Iris setosa*, *Iris versicolor*, and *Iris virginica*.
- Four measured features: length and width of sepals and petals (in cm).
- Fisher developed a linear discriminant model to distinguish the species from each other.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

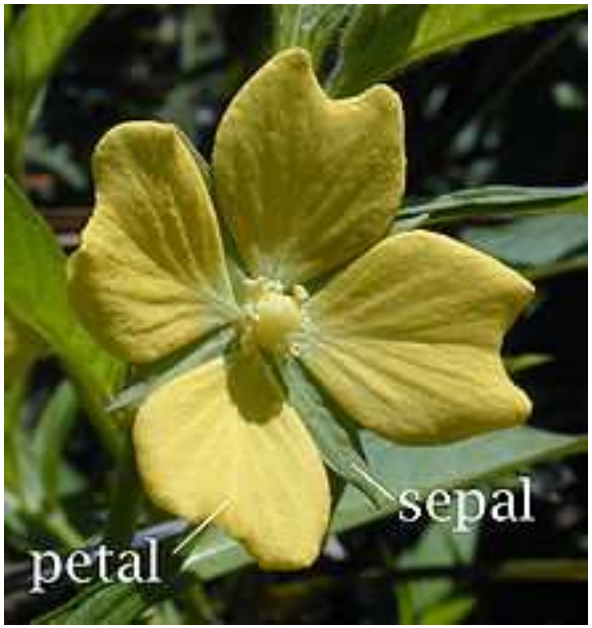
Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Iris Flower Data Set



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Iris Flower Data Set – R

```
> iris
```

	Sepal length	Sepal width	Petal length	Petal width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
⋮					
51	7.0	3.2	4.7	1.4	versicolor
52	6.4	3.2	4.4	1.5	versicolor
53	6.9	3.1	4.9	1.5	versicolor
54	5.5	2.3	4.0	1.3	versicolor
55	6.5	2.8	4.6	1.5	versicolor
⋮					
146	6.7	3.0	5.2	2.3	virginica
147	6.3	2.5	5.0	1.9	virginica
148	6.5	3.0	5.2	2.0	virginica
149	6.2	3.4	5.4	2.3	virginica
150	5.9	3.0	5.1	1.8	virginica

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Iris Flower Data Set – R

```
> irispca <- princomp(iris[-5])

> summary(irispca)
```

	Comp.1	Comp.2	Comp.3	Comp.4
Standard deviation	2.049	0.491	0.279	0.154
Proportion of variance	0.925	0.053	0.017	0.005
Cumulative proportion	0.925	0.978	0.995	1.000

Thus 92.5% of the variation in the dataset is explained by the first component alone, and 97.8% is explained by the first two components.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Iris Flower Data Set – R

```
> irispca$loadings
```

	Comp.1	Comp.2	Comp.3	Comp.4
Sepal length	0.361	-0.657	0.582	0.315
Sepal width		-0.730	-0.598	-0.320
Petal length	0.857	0.173		-0.480
Petal width	0.358		0.017	0.754

Loadings are the eigenvectors scaled by the square root of the corresponding eigenvalues.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Iris Flower Data Set – R

```
> irispca$scores
```

	Comp.1	Comp.2	Comp.3	Comp.4
1	-2.684	-0.319	0.028	0.002
2	-2.714	0.177	0.210	0.099
3	-2.889	0.145	-0.018	0.020
4	-2.745	0.318	-0.031	0.077
5	-2.729	-0.327	0.090	-0.061
⋮				

- The scores given by the base change using the eigenvectors are the coefficients of the loadings providing the observations.
- The analysis suggests that the first two components are sufficient to explain the data.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

*Spiking Neural Networks

- Generations of artificial neural network models
- Spiking neuron model
- *Information coding
- *Learning

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Spiking Neural Networks

- Spiking neural networks (SNNs) are ANN models that mimic more closely neural network structures in the brain.
- SNNs take also the concept of timing into account.
- Neurons do not fire at each propagation cycle but fire only when a membrane potential reaches a specific value (threshold).

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Generations of Neural Network Models

Neural networks can be classifed according to their computational units into three generations.

First generation:

- McCulloch-Pitts neuron (percepton, threshold gate) is the basic computational unit.
- It gives rise to several neural network models with binary encoded output such as multilayer perceptron, Hopfield nets, Boltzmann machines.
- Networks of the first generation are universal for digital computations, i.e., each Boolean function can be calculated by some multilayer perceptron with a single hidden layer.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Generations of Neural Network Models

Neural networks can be classifed according to their computational units into three generations.

Second generation:

- The computational units are equipped with an activation function (sigmoid or hyperbolic tangent) which has a continuous set of possible output values.
- It gives rise to several neural network models with continuous input and output sets such as feedforward and recurrent netral nets.
- Networks of the second generation are univeral for analog computations, i.e., each continuous function with compact domain and range can be approximated well by a network with a single hidden layer.
- Networks of the second generation are trained by supervised learning algorithms based on gradient descent such as backpropagation.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Generations of Neural Network Models

Neural networks can be classifed according to their computational units into three generations.

Third generation:

- In the first two generations, the neurons are modeled by *rate coding* in the form of real number values that represent the activation level.
- The computational units are spiking neurons (integrate and fire) neurons.
- This model has been the result of investigations in neurobiology which indicates that many biological neural systems use the timing of single action potentials (spikes) to encode information.
- The *temporal coding* scheme addresses the temporal relationship between the firing of neurons.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Spiking Neuron Model

- In the spiking neuron model, a neuron v fires whenever its potential P_v reaches a certain threshold Θ_v.
- The potential P_v models the electric membrane potential in a real neuron; P_v is the sum of excitory postsynaptic potentials (EPSPs) and inhibitory postsynaptic potentials (IPSPs).
- The firing of a presynaptic neuron u at time s contributes to the potential P_v of neuron v at time $t \geq s$, written $P_v(t)$, whose amount is given by the term

$$w_{u,v}\epsilon_{u,v}(t-s), \tag{455}$$

where $w_{u,v}$ is a weight value and $\epsilon_{u,v}(t-s)$ is a response function.

- The weight $w_{u,v}$ in the term (455) is a measure of strength (efficacy) of the synapse between neurons u and v.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Spiking Neuron Model

- In view of learning, the weight $w_{u,v}$ can be substituted by a function $w_{u,v}(t)$ depending on time.
- A biological synapse is either excitatory or inhibitory and does not change its state over time.
- The potential P_v is assumed to have the value 0 if postsynaptic potentials are absent and the threshold value Θ_v is positive.
- In a biological neuron, the resting membrane potential is about -70mV and the firing threshold of the rested neuron is about -50mV. The postsynaptic potential (EPSP or IPSP) changes the potential temporarily by a few mV.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Spiking Neuron Model

- When a neuron v fires at time t', it will not fire again for a few msec after time t' independent of the size of the potential.
- This *refractory effect* is modeled by a threshold function $\Theta_v(t - t')$ which is rocket-high for small values of $t - t'$.
- In the deterministic (noise-free) model of spiking neurons it is assumed that a neuron v fires whenever the potential $P_v(t)$ reaches the function $\Theta_v(t - t')$
- It is assumed that $\Theta_v(t - t') = \Theta(0)$ for small values $t - t'$, where $\Theta(0)$ is the resting value;

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Spiking Neuron Model

The *(deterministic) spiking neuron network* (SNN) (W. Maass, 1995) consists of

- finite set V of spiking neurons,
- set $E \subseteq V \times V$ of synapses,
- weight $w_{u,v} \geq 0$ and response function $\epsilon_{u,v} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ for each synapse $(u, v) \in E$,
- threshold function $\Theta_v : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ for each neuron $v \in V$.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Spiking Neuron Model

- Suppose $F_u \subseteq \mathbb{R}_{\geq 0}$ is the set of firing times (spike trains) of neuron u.
- The *potential* at the trigger zone of neuron v at time t is

$$P_v(t) = \sum_{\substack{u \in V \\ (u,v) \in E}} \sum_{\substack{s \in F_u \\ s < t}} w_{u,v} \epsilon_{u,v}(t-s). \tag{456}$$

- A neuron v fires at time t when the potential $P_v(t)$ reaches the threshold $\Theta_v(t-t')$, where t' is the time of the most recent firing of neuron v.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Spiking Neuron Model

- In the spiking neural model, there is a set $V_{\text{in}} \subseteq V$ of input neurons and a set $V_{\text{out}} \subseteq V$ of output neurons.

  For each input neuron $u \in V_{\text{in}}$, the firing times F_u are given from outside of the model. The output of the network is given by the spike trains F_v of the output neurons v.

- In the stochastic (noisy) version of the spiking neural network model, the differences $P_v(t) - \Theta_v(t - t')$ are governed by the probability that the neuron v fires at time t (W. Gerstner, 1995).

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Information Coding

A fundamental coding method is *temporal coding* (Maass, 1997):

- The firing time of a neuron encodes a value in the sense that an early firing of the neuron represents a large value.
- Consider a neuron v which receives excitatory input from m neurons $u_1, \ldots, u_m$ with corresponding weights $w_1, \ldots, w_m$.
- The analog input from presynaptic neuron u_i is mapped by the standard sigmoid function to a value x_i in $[0, 1]$.
- The firing time of neuron u_i is $t_i = T_1 - cx_i$, where $c > 0$ is a constant and T_1 signifies the beginning of a time frame lasting for the period c.
- The neuron v fires at time determined by $T_2 - \sum_i w_i s_i$, where $s_i = cx_i$, $1 \leq i \leq m$, with $T_2 > T_1$ being a constant.

This computation can be performed on the basis of competitive temporal coding without explicit reference to T_1 and T_2.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

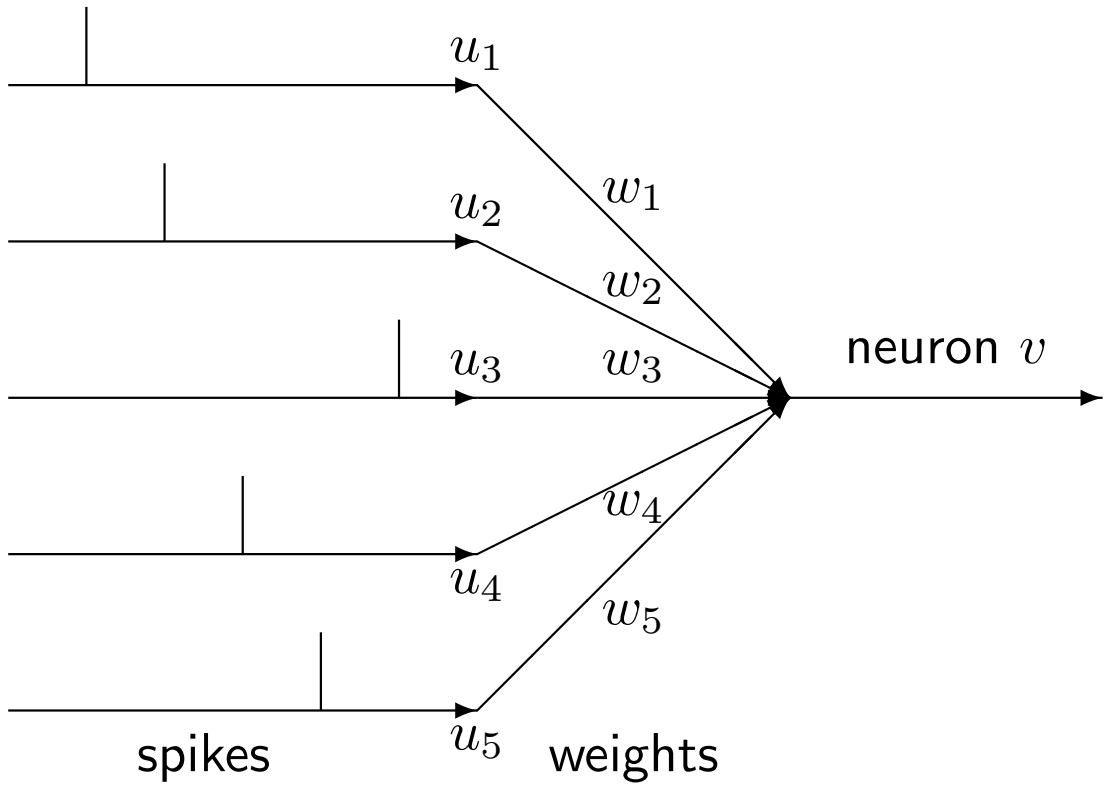
Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Information Coding

Temporal coding:



Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Learning

- The research on spiking neural networks has mainly been focused on unsupervised learning (Ruf, Schmidt, 1998).
- The objective of unsupervised learning is to allow the network to self-organize and eventually learn to discriminate between input patterns which have no explicit identification.
- The most prominent example of an artifical neural network in the second generation trained by unsupervised learning is Kohonen's self-organizing map (SOM).

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Learning

- Given a collection of input vectors $s^{(l)} = (s_{l1}, \ldots, s_{lm})$ of an SNN with m input vectors and n competitive neurons.
- Each competitive neuron v_j receives synaptic feedforward input from each input neuron u_i with w_{ij} and lateral synaptic input from each competitive neuron v_k, $k \neq j$, with weight w'_{kj}.
- The latter weights allow to implement a distance function between the competitive neurons.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Learning

- At each cycle of the learning phase, one input vector $s^{(l)}$ is randomly chosen and presented to the network.
- Each competitive neuron v_j then computes the weighted sum $\sum_i w_{ij} s_{li}$ as described in the section about information coding.
- If the input vector and the weight vector for each neuron are normalized, the weighted sum represents the similarity between the two vectors with respect to the Euclidean distance.
- The earlier the competitive neuron v_j fires, the more similar is its weight vector to the input vector; i.e., the winner among the layer of competitive neurons fires first.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Learning

The *learning rule* is

$$\Delta w_{ij} = \eta \frac{T_o - t_j}{T_o}(s_{li} - w_{ij}), \tag{457}$$

where t_j is the firing time of the jth competitive neuron v_j.

- The rule applies only to neurons that have fired before a certain time T_o.
- The factor $(T_o - t_j)/T_o$ provides the neighborhood function, which is largest for the winner neuron and decreases for neurons which fire later.
- It has been demonstrated that the learning works pretty well for standard one-dimensional input patterns.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Neural Networks in R

R-library `deepnet` provides procedures for deep learning architectures and neural network algorithms (Xiao Rong, 2014). Loading of library:

```
> library(deepnet)
```

A single or multiple hidden layer network can be defined by the command `nn.train`.

Neural Networks in R - Example

Consider two normally distributed data sets with given mean and variance,

```
> var1 <- c(rnorm(50, 0.0, 0.5),rnorm(50, 10.0, 200.0))
> var2 <- c(rnorm(50, 0.1, 0.5),rnorm(50, 9.0, 200.0))
```

Merge these sample sets into 100×2 matrix of input data,

```
> x <- matrix(c(var1, var2), nrow=100, ncol=2) //samples
> x
              [,1]        [,2]
  [1,]  -0.18657632  0.08037026
...
  [50]  -0.30735384  -0.07672294
  [51] -249.07337659 -111.72856325
...
 [100] -0.31854956    37.67063888
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Neural Networks in R - Example (cont'd)

Desired output data is given by vector of length 100,

```
> y <- c( rep(1,50), rep(0,50))   // target values
```

The input data $x[1,] \ldots x[50,]$ and $x[51,] \ldots x[100,]$ have desired output values 1 and 0, respectively.

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Neural Networks in R - Example (cont'd)

Define ANN with two input units, one output unit and 10 hidden units and train the network by 1000 epochs:

```
> nnet <- nn.train(x, y, hidden=c(10), numepochs=1000)
```

Specification of network:

```
> nnet
$input_dim
[1] 2
$output_dim
[1] 1
$hidden
[1] 10
$size
[1] 2 10 1
$activiationfun
[1] "sigm"
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Neural Networks in R - Example (cont'd)

Specification of network (cont'd):

```
$learningrate
[1] 0.8
$W              // weights
$W[[1]]
...
$W[[2]]
...
$B              // biases
$B[[1]]
...
$B[[2]]
...
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Neural Networks in R - Example (cont'd)

Predict new samples by the command `nn.predict`:

```
> test_var1 <- c(rnorm(50, 0.1, 0.51),
                 rnorm(50, 10.1, 200.1))
> test_var2 <- c(rnorm(50, 0.2, 0.51),
                 rnorm(50, 9.1, 200.1))
> test_x <- matrix(test_var1, test_var2),
                 nrow=100, ncol=2)
> y_test <- nn.predict(nnet, test_x)
```

Soft Computing

K.-H. Zimmermann

Contents

ANN Technology

Multilayer Perceptron

Deep Learning via Keras

Universality

Hopfield Network

Self-Organizing Networks

*Spiking Neural Networks

Neural Networks in R

Neural Networks in R - Example (cont'd)

Output is given by the raw values of the classes:

```
> y_test
             [,1]
  [1,] 0.95784997
...
 [50,] 0.95993372
 [51,] 0.07435817
...
[100,] 0.02650158
```

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Part VII

Fuzzy Sets

Contents

- Triangular norms
- Fuzzy sets and fuzzy numbers
- Fuzzy inference and fuzzy control
- *Stochastic fuzzy sets

Knowledge

- Triangular norms
- Fuzzy logic
- Fuzzy sets and numbers

Skills

- Fuzzy control and inference using R

Fuzzy Sets

- Fuzzy sets generalize ordinary set theory.
- Introduced independently by Lofti Zadeh and Dieter Klaua (1965).
- Applications in decision based systems and knowledge engineering.
- No learning needed in fuzzy inference and control.
- Goal is introduction to fuzzy sets and their applications based on triangular norms.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms

Triangular norms were first studied in probabilistic metric spaces (Karl Menger, 1942).

- Triangular norms and conorms
- Duality and continuity
- Important triangular norms

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms

A mapping $T : [0,1]^2 \to [0,1]$ is a *triangular norm* or *t-norm* if for all $x, y, z \in [0,1]$,

- Commutativity:

$$T(x,y) = T(y,x) \tag{458}$$

- Associativity:

$$T(x, T(y,z)) = T(T(x,y), z) \tag{459}$$

- Unit element:

$$T(x,1) = x \tag{460}$$

- Monotonicity:

$$y \leq z \Longrightarrow T(x,y) \leq T(x,z). \tag{461}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms – Example

Four basic t-norms:

- Minimum:

$$T_M(x, y) = \min\{x, y\} \tag{462}$$

- Probabilistic product:

$$T_P(x, y) = x \cdot y \tag{463}$$

- Lukasiewicz t-norm:

$$T_L(x, y) = \max\{x + y - 1, 0\} \tag{464}$$

- Drastic product:

$$T_D(x, y) = \begin{cases} 0 & \text{if } 0 \leq x, y < 1, \\ \min\{x, y\} & \text{otherwise.} \end{cases} \tag{465}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms – Example (cont'd)

The above t-norms are defined in Maple as follows:

```
> M := (x,y) -> min(x,y);
> P := (x,y) -> x*y;
> L := (x,y) -> max(x+y-1,0);
> D := (x,y) -> piecewise(x=1, y, y=1, x, 0);
```

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

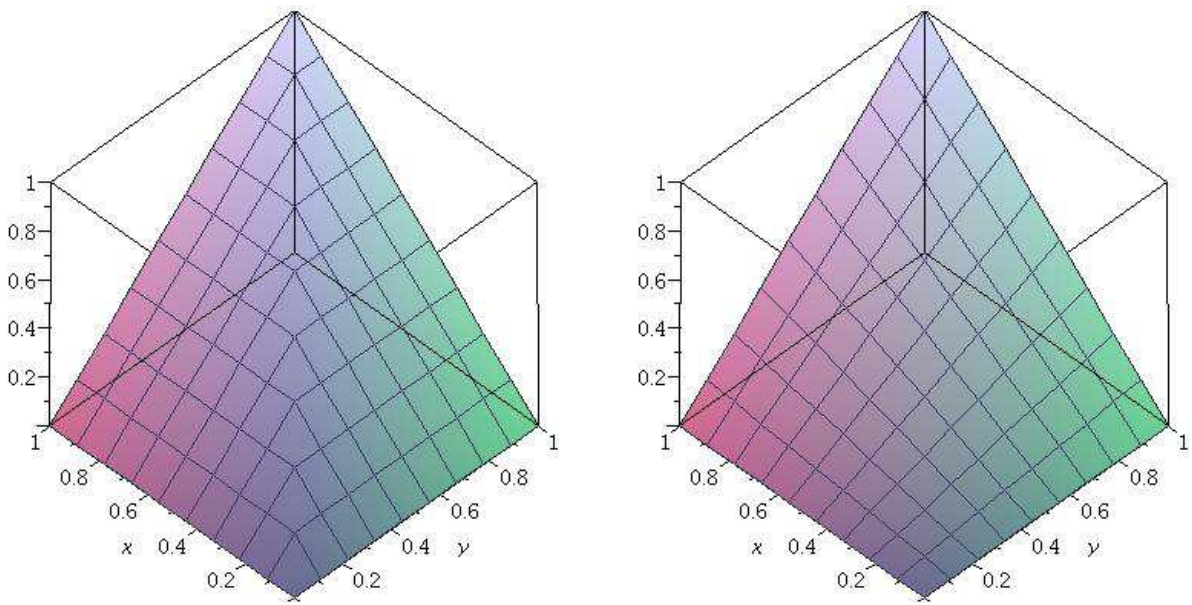
Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms – Example (cont'd)

T-norms T_M and T_P:



Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

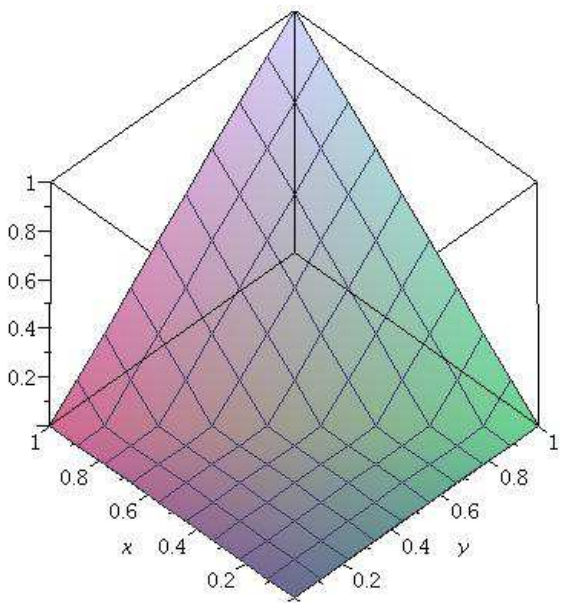
Fuzzy Inference

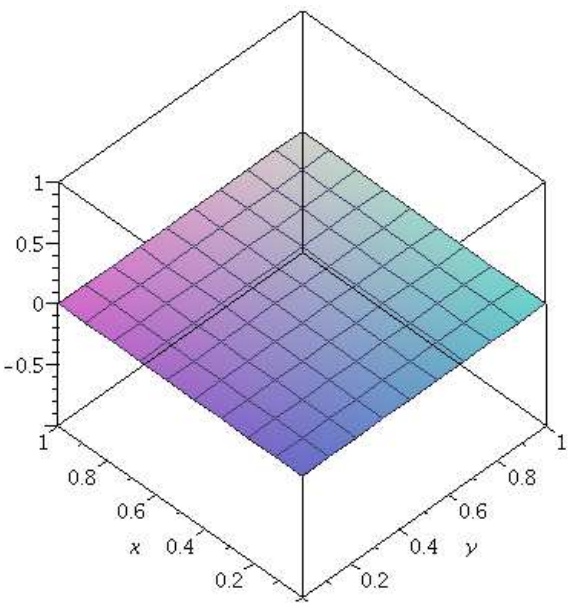
Fuzzy Control

Fuzzy Sets in R

Triangular Norms – Example (cont'd)

T-norms T_L and T_D:





Triangular Norms

Basic properties of t-norm T:

- Boundary conditions:

$$T(0,x) = T(x,0) = 0, \quad x \in [0,1], \tag{466}$$

since for each $x \in [0,1]$, $T(0,x) \leq T(0,1) = 0$, and

$$T(1,x) = T(x,1) = x, \quad x \in [0,1]. \tag{467}$$

- Boundedness:

$$T(x,x) \leq x, \quad x \in [0,1]. \tag{468}$$

- Joint monotonicity:

$$T(x_1,y_1) \leq T(x_2,y_2) \tag{469}$$

if $x_1 \leq x_2$ and $y_1 \leq y_2$ for all $x_1, y_1, x_2, y_2 \in [0,1]$.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms

- A t-norm T_1 is *weaker* than a t-norm T_2, written $T_1 \leq T_2$, if

$$T_1(x, y) \leq T_2(x, y), \quad x, y \in [0, 1]. \tag{470}$$

- Each t-norm T satisfies

$$T_D \leq T \leq T_M. \tag{471}$$

- Ordering of the four basic t-norms:

$$T_D < T_L < T_P < T_M. \tag{472}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms

- The minimum t-norm T_M is the only t-norm T satisfying

$$T(x,x) = x, \quad x \in [0,1]. \tag{473}$$

- The drastic product T_D is the only t-norm T fulfilling

$$T(x,x) = 0, \quad x \in [0,1[. \tag{474}$$

Proof.

- Suppose T is a t-norm with $T(x,x) = x$ for all $x \in [0,1]$. Then for all $x, y \in [0,1]$ with $y \leq x$, monotonicity implies $y = T(y,y) \leq T(x,y) \leq T_M(x,y) = y$. Hence, $T = T_M$.
- Suppose T is a t-norm with $T(x,x) = 0$ for all $x \in [0,1[$. Then for all $x, y \in [0,1[$ with $y \leq x$, monotonicity implies $0 \leq T(x,y) \leq T(x,x) = 0$. Moreover, $T(x,1) = x$ for all $x \in [0,1]$. Hence, $T = T_D$.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms

A mapping $S : [0,1]^2 \to [0,1]$ is a *triangular conorm* or *t-conorm* if for all $x, y, z \in [0,1]$,

- Commutativity:

$$S(x,y) = S(y,x) \tag{475}$$

- Associativity:

$$S(x, S(y,z)) = S(S(x,y), z) \tag{476}$$

- Unit element:

$$S(x,0) = x \tag{477}$$

- Monotonicity:

$$y \leq z \Longrightarrow S(x,y) \leq S(x,z). \tag{478}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms – Example

Four basic t-conorms:

- Maximum:

$$S_M(x, y) = \max\{x, y\} \tag{479}$$

- Probabilistic sum:

$$S_P(x, y) = x + y - x \cdot y \tag{480}$$

- Lukasiewicz t-conorm:

$$S_L(x, y) = \min\{x + y, 1\} \tag{481}$$

- Drastic sum:

$$S_D(x, y) = \begin{cases} 1 & \text{if } 0 < x, y \leq 1, \\ \max\{x, y\} & \text{otherwise.} \end{cases} \tag{482}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms – Example (cont'd)

These t-conorms are defined in Maple as follows:

```
> M := (x,y) -> max(x,y);
> P := (x,y) -> x+y-x*y;
> L := (x,y) -> min(x+y,1);
> D := (x,y) -> piecewise(x=0, y, y=0, x, 1);
```

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

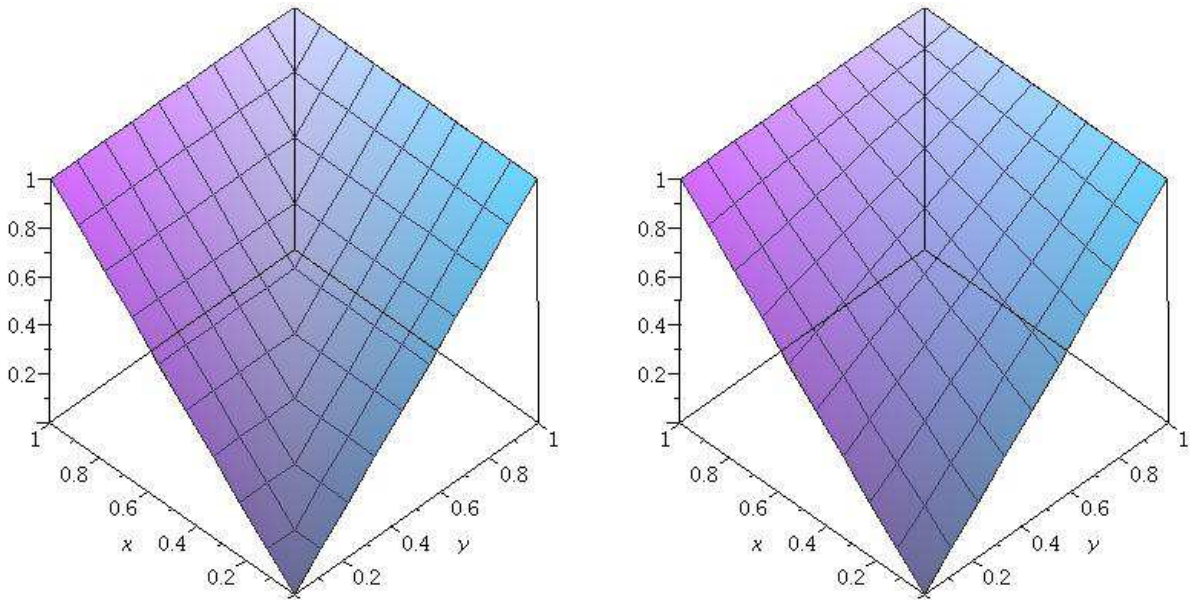
Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms – Example (cont'd)

T-conorms S_M and S_P:



Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

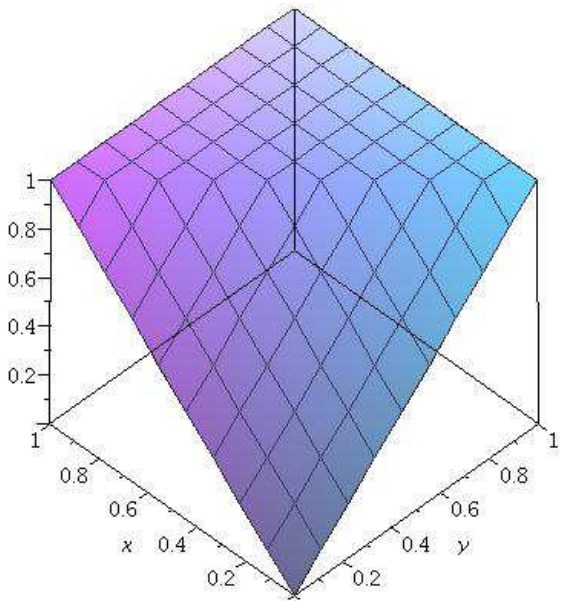
Fuzzy Inference

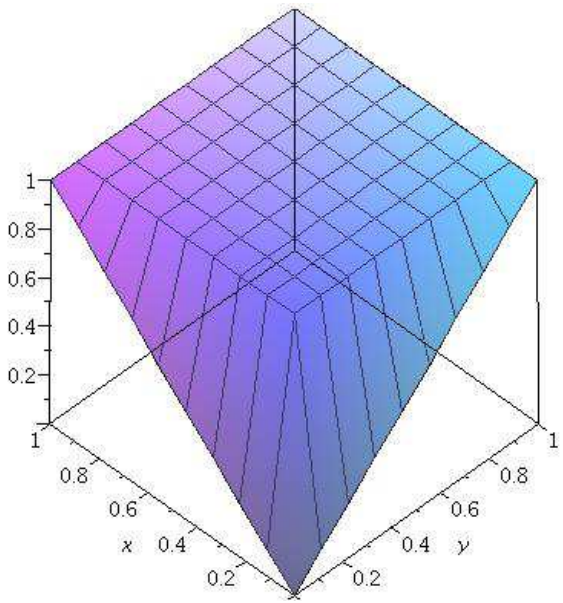
Fuzzy Control

Fuzzy Sets in R

Triangular Norms – Example (cont'd)

T-conorms S_L and S_D:





Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms

Basic properties of t-conorm S:

- Boundary conditions:

$$S(1,x) = S(x,1) = 1, \quad x \in [0,1], \tag{483}$$

since for each $x \in [0,1]$, $S(1,x) \geq S(1,0) = 1$, and

$$S(0,x) = x, \quad x \in [0,1]. \tag{484}$$

- Boundedness:

$$S(x,x) \geq x, \quad x \in [0,1]. \tag{485}$$

- Joint monotonicity:

$$S(x_1,y_1) \leq S(x_2,y_2) \tag{486}$$

if $x_1 \leq x_2$ and $y_1 \leq y_2$ for all $x_1, y_1, x_2, y_2 \in [0,1]$.

Triangular Norms

- For a given t-norm T, the t-conorm S given by

$$S(x, y) = 1 - T(1 - x, 1 - y) \tag{487}$$

  is the *dual t-conorm* of T, written $S = T^*$,

- For a given t-conorm S, the t-norm T given by

$$T(x, y) = 1 - S(1 - x, 1 - y) \tag{488}$$

  is the *dual t-norm* of S, written $T = S^*$.

- For each t-norm T and each t-conorm S, we have

$$(T^*)^* = T \quad \text{and} \quad (S^*)^* = S. \tag{489}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms

- The pairs of t-norms and t-conorms

$$(T_M, S_M), \quad (T_P, S_P), \quad (T_L, S_L), \quad (T_D, S_D) \tag{490}$$

  are mutually dual to each other.

- For t-norms T_1 and T_2,

$$T_1 \leq T_2 \quad \Longrightarrow \quad T_2^* \leq T_1^*. \tag{491}$$

- For t-conorms S_1 and S_2,

$$S_1 \leq S_2 \quad \Longrightarrow \quad S_2^* \leq S_1^*. \tag{492}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms

- Each t-conorm S satisfies

$$S_M \leq S \leq S_D. \tag{493}$$

- The four basic t-conorms have the following ordering,

$$S_M < S_P < S_L < S_D. \tag{494}$$

Triangular Norms

- A function $f : [0,1]^2 \to [0,1]$ is *continuous* if for all convergent sequences $(x_n)_{n\geq 0}$ and $(y_n)_{n\geq 0}$ in $[0,1]$,

$$f\left(\lim_{n\to\infty} x_n, \lim_{n\to\infty} y_n\right) = \lim_{n\to\infty} f(x_n, y_n). \tag{495}$$

- $[0,1]$ is a compact subset of $\mathbb{R}$ and so the continuity of a function $f : [0,1]^2 \to [0,1]$ is equivalent to its uniform continuity.
- A t-norm is continuous iff its dual t-conorm is continuous.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms – Example

- The basic t-norms T_M, T_P, and T_L as well as their dual t-conorms S_M, S_P, and S_L are continuous.
- The drastic product T_D and the drastic sum S_D are not continuous.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Properties of Triangular Norms

- Partially ordered semigroups and duality
- De Morgan triples and Fuzzy logic
- Lattice monoids and residuum operation

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms

Given set $M \neq \emptyset$ and dyadic operation $* : M \times M \to M$.

- $(M, *)$ is *semigroup* if $*$ is associative, i.e., for all $x, y, z \in M$, $x * (y * z) = (x * y) * z$.
- Semigroup $(M, *)$ is *commutative* if $*$ is commutative, i.e., for all $x, y \in M$, $x * y = y * x$.
- $a \in M$ is *annihilator* of semigroup $(M, *)$ if for each $x \in M$, $x * a = a = a * x$.
- $(M, *, e)$ is *monoid* if $(M, *)$ semigroup and $e \in M$ *unit element*, i.e., for all $x \in M$, $x * e = x = e * x$.
- Monoid $(M, *, e)$ is *commutative* if $*$ is commutative.

The unit element of a monoid is uniquely determined, since if e, e' are unit elements, then $e' = ee' = e$.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms

Let $\preceq$ be a partial order on a set M.

- $(M, *, \preceq)$ is *partially ordered semigroup* if
  - $(M, *)$ is semigroup.
  - Operation $*$ is *order preserving*, i.e., for all $x, y, z \in M$,

$$y \preceq z \Longrightarrow x * y \preceq x * z \wedge y * x \preceq z * x. \tag{496}$$

- In particular, if $\preceq$ is linear (or total) order on M, the partially ordered semigroup $(M, *, \preceq)$ is *fully ordered*.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms

- Two semigroups $(M, *)$ and $(N, \circ)$ are *isomorphic* if there is a bijection $\phi : M \to N$ such that for all $x, y \in M$,

$$\phi(x * y) = \phi(x) \circ \phi(y). \tag{497}$$

- Two partially ordered semigroups $(M, *, \preceq)$ and $(N, \circ, \sqsubseteq)$ are *isomorphic* if there is a bijection $\phi : M \to N$ such that for all $x, y \in M$,

$$\phi(x * y) = \phi(x) \circ \phi(y) \tag{498}$$

and ϕ is *order preserving*, i.e.,

$$x \preceq y \Longrightarrow \phi(x) \sqsubseteq \phi(y). \tag{499}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms

- A mapping $T : [0,1]^2 \to [0,1]$ is a t-norm iff

$$([0,1], T, \leq) \tag{500}$$

  is a fully ordered commutative semigroup with unit element 1 and annihilator 0.

- A mapping $S : [0,1]^2 \to [0,1]$ is a t-conorm iff

$$([0,1], S, \leq) \tag{501}$$

  is a fully ordered commutative semigroup with unit element 0 and annihilator 1.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms

Given fully ordered semigroups $(M, *, \preceq)$ and $(N, \circ, \sqsubseteq)$, and strictly decreasing bijection $\phi : M \to N$.

- $(M, *, \preceq)$ is ϕ-*dual* to $(N, \circ, \sqsubseteq)$ if for all $x, y \in M$,

$$\phi(x * y) = \phi(x) \circ \phi(y). \tag{502}$$

- $(M, *, \preceq)$ is ϕ-dual to $(N, \circ, \sqsubseteq)$ iff $(N, \circ, \sqsubseteq)$ is ϕ^{-1}-dual to $(M, *, \preceq)$.
- If the orders are ignored, the semigroups $(M, *)$ and $(N, \circ)$ are simply isomorphic.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms – Construction of New Norms

Given mapping $F : [0,1]^2 \to [0,1]$ such that $([0,1], F, \leq)$ is fully ordered semigroup and $\phi : [0,1] \to [0,1]$ is strictly decreasing bijection.

Define the mapping $F_\phi : [0,1]^2 \to [0,1]$ with

$$F_\phi(x,y) = \phi^{-1}(F(\phi(x), \phi(y))). \tag{503}$$

Then

- $([0,1], F_\phi, \leq)$ is a fully ordered semigroup.
- If F is a t-norm, then F_ϕ is a t-conorm ϕ-dual to F.
- If F is a t-conorm, then F_ϕ is a t-norm ϕ-dual to F.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms – Example

For any strictly decreasing bijection $\phi : [0,1] \to [0,1]$, the minimum norm satisfies

$$(T_M)_\phi = S_M, \quad (S_M)_\phi = T_M, \tag{504}$$

and the drastic norm satisfies

$$(T_D)_\phi = S_D, \quad (S_D)_\phi = T_D. \tag{505}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms – Example

Take the strictly decreasing bijection

$$\phi : [0,1] \to [0,1] : x \mapsto 1 - x. \tag{506}$$

Since $\phi^{-1}(x) = 1 - x$,

$$\begin{aligned}
(T_P)_\phi(x,y) &= \phi^{-1}(T_P(\phi(x),\phi(y))) \\
&= \phi^{-1}((1-x)(1-y)) \\
&= \phi^{-1}(1-(x+y-xy)) \\
&= x+y-xy \\
&= S_P(x,y).
\end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms

Consider duality by extension of Boolean negation.

- A non-increasing function $N : [0,1] \to [0,1]$ is a *negation* if

$$N(0) = 1 \quad \text{and} \quad N(1) = 0. \tag{507}$$

- A negation $N : [0,1] \to [0,1]$ is *strict* if N is continuous and strictly decreasing.
- A strict negation $N : [0,1] \to [0,1]$ is *strong* if N is an *involution*, i.e., $N \circ N = \mathrm{id}_{[0,1]}$ or $N = N^{-1}$.

A strict negation $N : [0,1] \to [0,1]$ is a strictly decreasing bijection.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

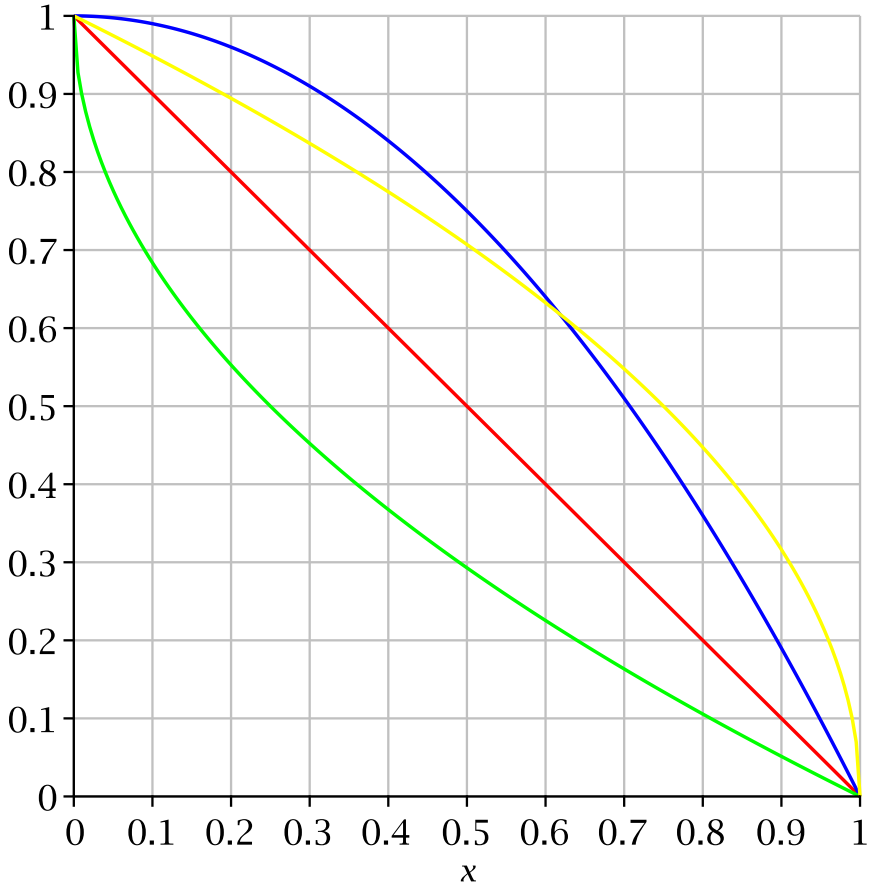
Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms – Negations

```
> plot([1-x, 1-x^2, 1-sqrt(x), sqrt(1-x)], x = 0..1,
    color = ["red", "blue", "green", "yellow"],
    axis = [gridlines=[10, color=grey]]);
```



Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms – Negations

- The most important strong negation is the *standard negation* $N_s : [0,1] \to [0,1]$ defined by

$$N_s(x) = 1 - x. \tag{508}$$

- The negation $N : [0,1] \to [0,1]$ given by

$$N(x) = 1 - x^2 \tag{509}$$

is strict but not strong, since $N^{-1}(x) = \sqrt{1-x}$.

- The negation $N_G : [0,1] \to [0,1]$ given by

$$N_G(x) = \begin{cases} 1 & \text{if } x = 0, \\ 0 & \text{otherwise,} \end{cases} \tag{510}$$

is not strict and thus not strong, called *Gödel negation*.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms – Construction of New Norms

Given t-norm T and strict negation N. The mapping $S : [0,1]^2 \to [0,1]$ given by

$$S(x,y) = N^{-1}(T(N(x), N(y))) \tag{511}$$

is a t-conorm which is N-dual to T; compare with (503).

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms

- (T, S, N) is a *De Morgan triple* if T is a t-norm, S is a t-conorm and N is a strong negation such that for all $x, y \in [0, 1]$,

$$S(x, y) = N^{-1}(T(N(x), N(y))), \tag{512}$$
$$T(x, y) = N^{-1}(S(N(x), N(y))). \tag{513}$$

- A De Morgan triple (T, S, N) satisfies the *law of the excluded middle* if for all $x \in [0, 1]$,

$$T(x, N(x)) = 0 \quad \text{and} \quad S(x, N(x)) = 1. \tag{514}$$

If $T(x, N(x)) = 0$, then

$$\begin{aligned} S(x, N(x)) &= N(T(N(x), N^2(x))) = N(T(N(x), x)) \\ &= N(0) = 1. \end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Triangular Norms - Example

(T_M, S_M, N_s) is a De Morgan triple, since

$$\begin{aligned} T_M(N_s(x), N_s(y)) &= \min\{1-x, 1-y\} = 1 - \max\{x, y\} \\ &= N_s(S_M(x, y)), \end{aligned}$$

but it does not satisfy the law of the excluded middle:

$$\begin{aligned} T_M(x, N_s(x)) &= \min\{x, 1-x\}, \\ S_M(x, N_s(x)) &= \max\{x, 1-x\}. \end{aligned}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

∗Triangular Norms

Let L be a non-empty set.

- A pair $(L, \preceq)$ is a *lattice* if $\preceq$ is a partial ordering on L such that for any two elements $x, y \in L$ it also contains their *join* $x \vee y$, i.e, the supremum of $\{x, y\}$, and their *meet* $x \wedge y$, i.e, the infimum of $\{x, y\}$.
- A lattice $(L, \preceq)$ is *complete* if for each subset B of L, the join $\bigvee B$ and meet $\bigwedge B$ exist and are contained in L.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

∗Triangular Norms

Let $(L, \preceq)$ be a lattice and $(L, *, e)$ be a monoid.

- $(L, *, \preceq)$ is an *l-monoid* if for all $x, y, z \in L$,

$$x * (y \vee z) = (x * y) \vee (x * z) \tag{515}$$

and

$$(x \vee y) * z = (x * z) \vee (y * z). \tag{516}$$

- An l-monoid $(L, *, \preceq)$ is *commutative* if the semigroup $(L, *)$ is commutative.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

$*$Triangular Norms

Let $(L, \preceq)$ be a lattice and $(L, *, e)$ be a monoid.

- A commutative l-monoid $(L, *, \preceq)$ is *residuated* if there is a dyadic operation $\rightarrow_*$ on L, called $*$-*residuum*, such that for all $x, y, z \in L$,

$$x * y \preceq z \quad \Longleftrightarrow \quad x \preceq y \rightarrow_* z. \tag{517}$$

- An l-monoid $(L, *, \preceq)$ is *integral* if there is a greatest element in the lattice $(L, \preceq)$ which coincides with e.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

∗Triangular Norms

- For each mapping $T : [0,1]^2 \to [0,1]$ the following are equivalent:
  - $([0,1], T, \leq)$ is a commutative residuated integral l-monoid.
  - T is a left-continuous t-norm.

  In this case, the T-residuum $\to_T$ has the form

$$x \to_T y = \sup\{z \in [0,1] \mid T(x,z) \leq y\}. \tag{518}$$

- As a consequence, for each left-continuous t-norm T we have $x \to_T y = 1$ iff $x \leq y$, since $T(x,1) = x$ for all x.
- Residual implication is an extension of Boolean implication to $[0,1]$-valued logic.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

∗Triangular Norms – Example

- The minimum t-norm T_M gives rise to the *Gödel implication*

$$x \to_M y = \begin{cases} 1 & \text{if } x \leq y, \\ y & \text{otherwise.} \end{cases} \tag{519}$$

Special case: Boolean implication for Boolean values,

$$0 \to_M 0 = 1,\ 0 \to_M 1 = 1,\ 1 \to_M 0 = 0,\ 1 \to_M 1 = 1. \tag{520}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

∗Triangular Norms – Example

- The product t-norm T_P provides the *Goguen implication*

$$x \to_P y = \begin{cases} 1 & \text{if } x \leq y, \\ \frac{y}{x} & \text{otherwise.} \end{cases} \tag{521}$$

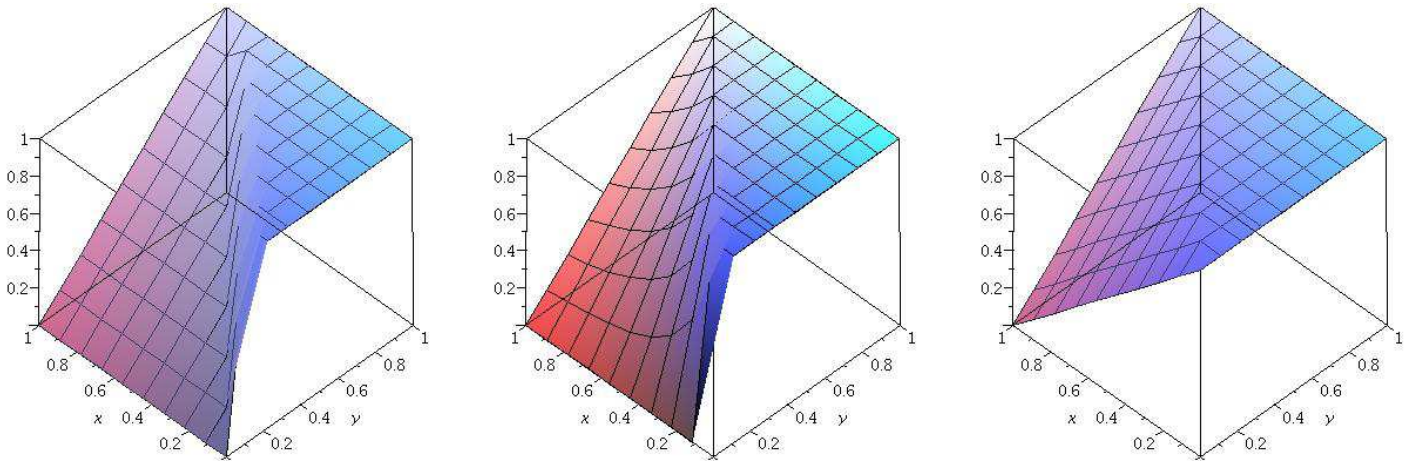
- The Lukasiewicz t-norm T_L yields the *Lukasiewicz implication*

$$x \to_L y = \min\{1 - x + y, 1\}. \tag{522}$$

- The triple $([0,1], T_D, \leq)$ is a commutative integral l-monoid that is not residuated since T_D is not left-continuous.

∗Triangular Norms – Example (cont'd)

Residual implications $\rightarrow_M$, $\rightarrow_P$, and $\rightarrow_L$:



Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Sets

- Crisp sets and characteristic functions
- Fuzzy sets and membership functions
- Kernel, support, and α-cut
- Intersection, union, and negation of fuzzy sets

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Sets

Given a universe (set) X.

- A subset A of X is a *crisp set*.
- A crisp set $A \subseteq X$ can be identified with its *characteristic function* $\chi_A : X \to \{0, 1\}$ defined by

$$\chi_A(x) = \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{otherwise.} \end{cases} \tag{523}$$

- Universe X and empty set $\emptyset$ are identified by the constant functions $\chi_X : x \mapsto 1$ and $\chi_\emptyset : x \mapsto 0$, resp.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Sets

Given a universe X.

- A *fuzzy subset* A of X is given by its *membership function*

$$\mu_A : X \to [0, 1],$$

  where for each $x \in A$ the quantity $\mu_A(x)$ is the *degree of membership* of x in A.

- Let A and B be fuzzy subsets of X.
  - A is a *subset* of B, written $A \subseteq B$, if $\mu_A \leq \mu_B$,
  - A is *equal* to B, written $A = B$, if $\mu_A = \mu_B$.

- A characteristic function is a special membership function; i.e., each crisp subset of X is a special fuzzy subset of X.

The degree of membership $\mu_A(x)$ can be seen as the truth value of the statement "x *is element of* A".

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Sets

Each fuzzy subset A of X can be associated with a number of crisp subsets of X:

- The *kernel* of A is defined by

$$\ker(A) = \{x \in X \mid \mu_A(x) = 1\}. \tag{524}$$

- The *support* of A is given as

$$\text{supp}(A) = \{x \in X \mid \mu_A(x) > 0\}. \tag{525}$$

- For each $\alpha \in [0, 1]$, the α-*cut* of A is

$$[A]_\alpha = \{x \in X \mid \mu_A(x) \geq \alpha\}. \tag{526}$$

A fuzzy subset A of X is *normal* if the kernel of A is non-empty; i.e., $\mu_A(x_0) = 1$ for some $x_0 \in X$.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Sets

Let X be a universe.

- For each fuzzy subset A of X,

$$\ker(A) = [A]_1 \quad \text{and} \quad \operatorname{supp}(A) = \bigcup_{\alpha \in]0,1]} [A]_\alpha. \tag{527}$$

- A fuzzy subset A of X is a crisp set iff

$$\ker(A) = \operatorname{supp}(A) = A. \tag{528}$$

- For the family $([A]_\alpha)_{\alpha \in [0,1]}$ of α-cuts of a fuzzy subset A of X,

$$[A]_0 = X \quad \text{and} \quad [A]_\alpha \subseteq [A]_\beta \quad \text{if } \alpha \geq \beta. \tag{529}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Sets

Given De Morgan triple (T, S, N) and fuzzy subsets A, B of X.

- *Intersection* of A and B:

$$\mu_{A \cap_T B}(x) = T(\mu_A(x), \mu_B(x)) \tag{530}$$

- *Union* of A and B:

$$\mu_{A \cup_S B}(x) = S(\mu_A(x), \mu_B(x)) \tag{531}$$

- *Complement* of A:

$$\mu_{\setminus_N A}(x) = N(\mu_A(x)). \tag{532}$$

De Morgan triple (T, S, N) may not satisfy the law of the excluded middle, i.e., for all $x \in [0, 1]$,

$$\mu_{A \cap_T \setminus_N A}(x) = 0 \text{ and } \mu_{A \cup_S \setminus_N A}(x) = 1. \tag{533}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

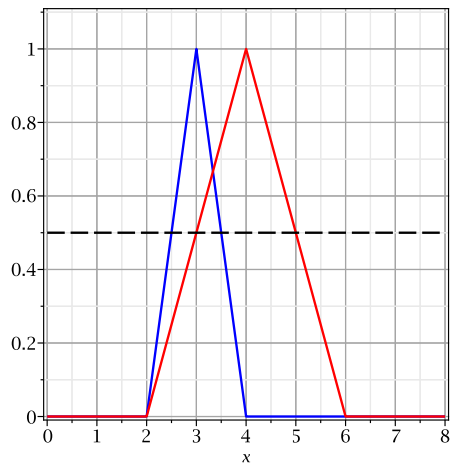
Fuzzy Sets in R

Fuzzy Sets – Example

Given fuzzy sets A and B over $X = \mathbb{R}$.
Consider intersection, union, and negation via the minimum t-norm,

```
> M := x -> min(A(x),B(x));
> P := x -> max(A(x),B(x));
> N := x -> 1-x;
```

Given fuzzy sets A and B over $X = \mathbb{R}$:



Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

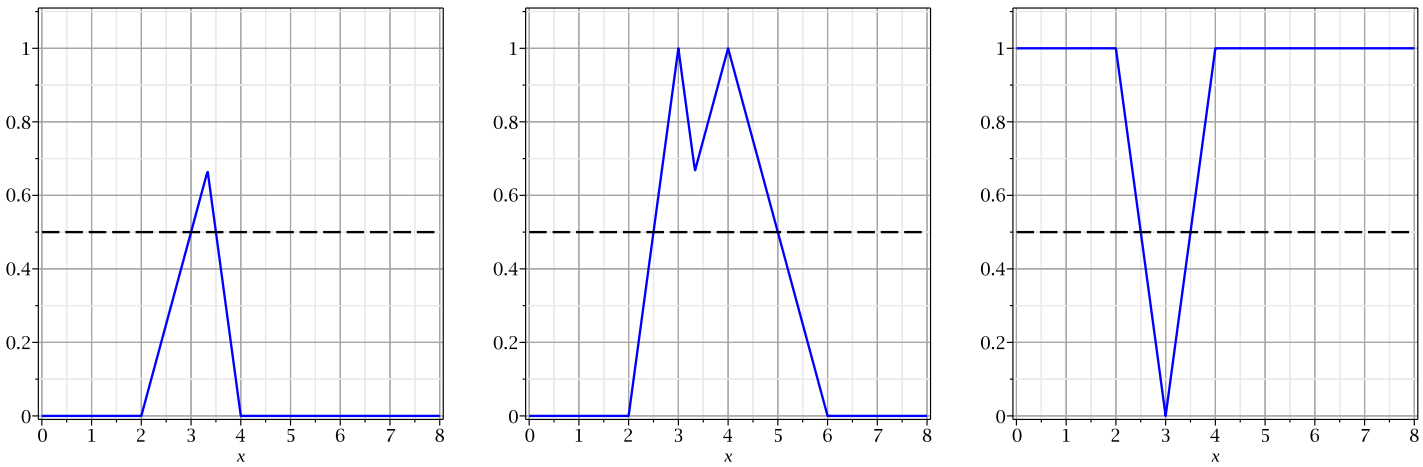
Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Sets – Example (cont'd)

Intersection and union of the above fuzzy sets A and B, and negation of above fuzzy set A:



Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Sets – Example

De Morgan triple (T_L, S_L, N_s) satisfies the law of the excluded middle. Indeed, for all $x \in [0,1]$,

$$T_L(x, N_s(x)) = \max\{x + (1-x) - 1, 0\} = 0$$

and

$$S_L(x, N_s(x)) = \min\{x + (1-x), 1\} = 1.$$

Thus for all $x \in [0,1]$,

$$\mu_{A \cap_{T_L} \setminus_{N_s} A}(x) = T_L(\mu_A(x), N_s(\mu_A(x))) = 0$$

and

$$\mu_{A \cup_{S_L} \setminus_{N_s} A}(x) = S_L(\mu_A(x), N_s(\mu_A(x))) = 1.$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Numbers

- Upper semicontinuous fuzzy numbers
- Gaussian fuzzy numbers
- Fuzzy intervals
- Piecewise linear fuzzy numbers
- Fuzzy reals
- *Zadeh's extension principle

Processes:

- Fuzzification: integer/real $\mapsto$ fuzzy number
- Defuzzification: fuzzy number $\mapsto$ integer/real

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Numbers

- A fuzzy subset A of $X = \mathbb{R}$ is a *fuzzy quantity*.
- A fuzzy quantity A is *convex* if for all $\alpha \in [0, 1]$ the α-cut $[A]_\alpha$ is a convex subset of $\mathbb{R}$, i.e., interval of $\mathbb{R}$.
- A fuzzy quantity A, which is normal, convex and has a bounded kernel, is a *fuzzy number*.
- Fuzzy numbers model linguistic quantities like "*approximately 20 degrees Celcius*".

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

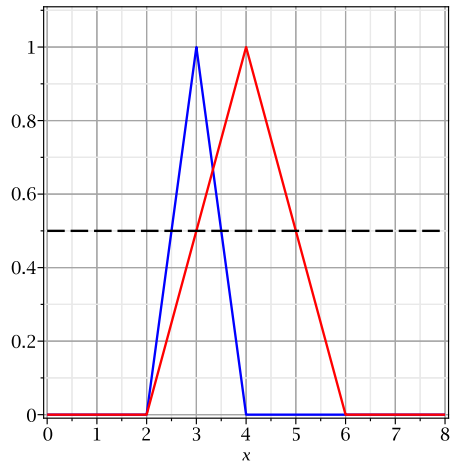
Fuzzy Inference

Fuzzy Control

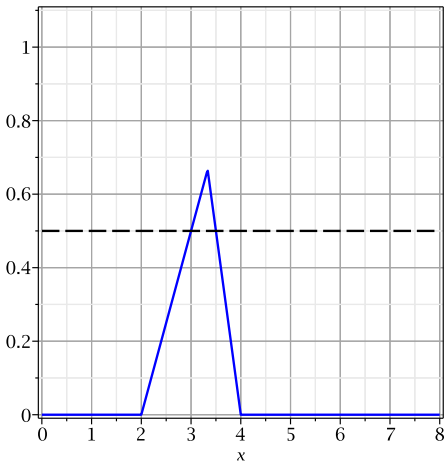
Fuzzy Sets in R

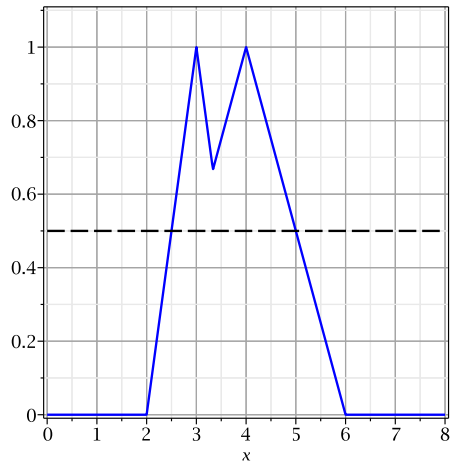
Fuzzy Numbers – Example

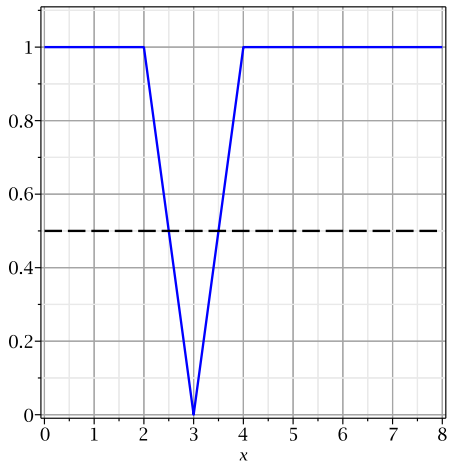
Fuzzy numbers:



No fuzzy numbers:







Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

*Fuzzy Numbers

$A = (l, r, F, G)$ is *upper semicontinuous fuzzy number* if

- l and r are real numbers with $l < r$,
- $F, G :]0, \infty[\to [0, 1]$ are non-increasing left-continuous mappings such that for all $x \in \mathbb{R}$,

$$\mu_A(x) = \begin{cases} F(l - x) & \text{if } x \in]-\infty, l[, \\ 1 & \text{if } x \in [l, r], \\ G(x - r) & \text{if } x \in]r, \infty[. \end{cases} \tag{534}$$

The interval $[l, r]$ is the kernel of A, and F and G determine the left and right shape of A, resp.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Numbers

A *Gaussian fuzzy number* is an upper semicontinuous fuzzy number

$$A = (x_0, x_0, F_\alpha, F_\alpha), \tag{535}$$

where $x_0 \in \mathbb{R}$, $\alpha \in]0, \infty[$, and for all $x \in]0, \infty[$ the shape function $F_\alpha :]0, \infty[\to [0, 1[$ is

$$F_\alpha(x) = e^{-\frac{x^2}{\alpha}}. \tag{536}$$

The corresponding membership function is

$$\mu_A(x) = e^{-\frac{(x-x_0)^2}{\alpha}}. \tag{537}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

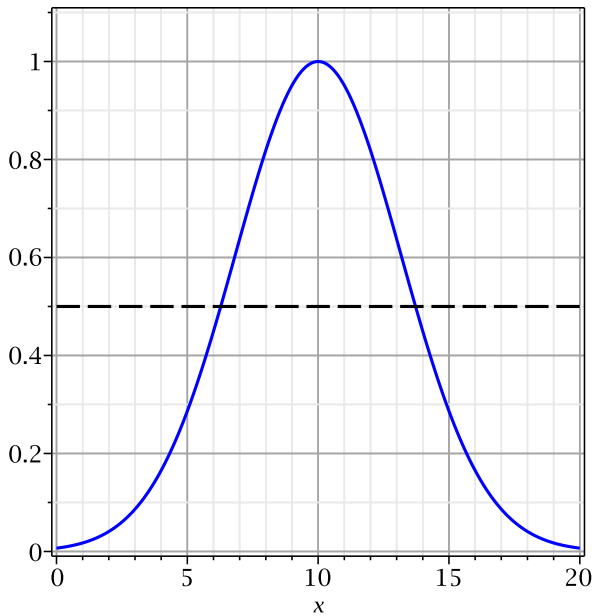
Fuzzy Control

Fuzzy Sets in R

Fuzzy Numbers

Plot of Gaussian fuzzy number A in Maple:

```
> A := exp(-(x-10)^2/20):
> plot([A,0.5],x=0..20,0..1.1, axes=boxed,
   gridlines=true, thickness=[2,2],
   color=[blue,black], font=[helvetica,12],
   linestyle=[solid,dash]);
```



Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

*Fuzzy Numbers

A *fuzzy interval* is a fuzzy number

$$A = (l, r, \alpha, \beta)_{L,R}, \tag{538}$$

where

- $l, r \in \mathbb{R}$,
- $\alpha, \beta \in [0, \infty[$, ($\alpha$ left spread of A and β right spread of A),
- $L, R :]0, 1] \to [0, 1[$ are non-increasing left-continuous mappings with $L(x) > 0$ and $R(x) > 0$ for all $x \in]0, 1[$ such that

$$\mu_A(x) = \begin{cases} L\left(\frac{l-x}{\alpha}\right) & \text{if } x \in]l-\alpha, l[, \\ 1 & \text{if } x \in [l, r], \\ R\left(\frac{x-r}{\beta}\right) & \text{if } x \in]r, r+\beta[, \\ 0 & \text{otherwise.} \end{cases} \tag{539}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

*Fuzzy Numbers

A fuzzy interval A can be defined in Maple as follows,

```
> A := piecewise(l-a<x and x<l, L((l-x)/a),
          l<x and x<r, 1,
          r<x and x<r+b, R((x-r)/b), 0):
```

Define L and R accordingly.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

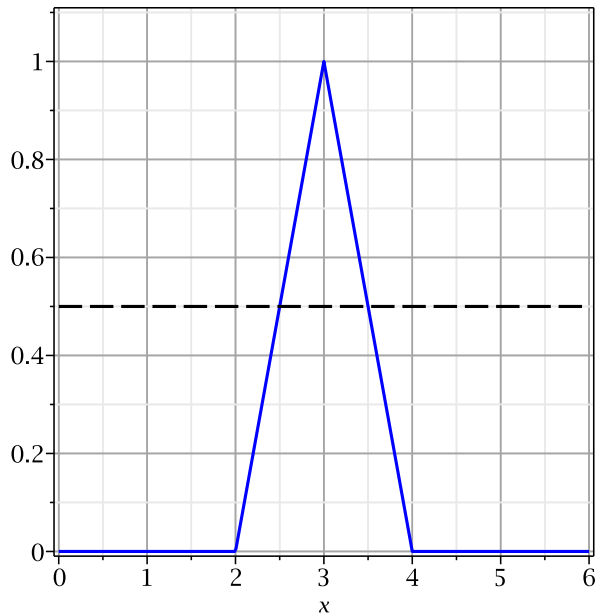
Fuzzy Inference

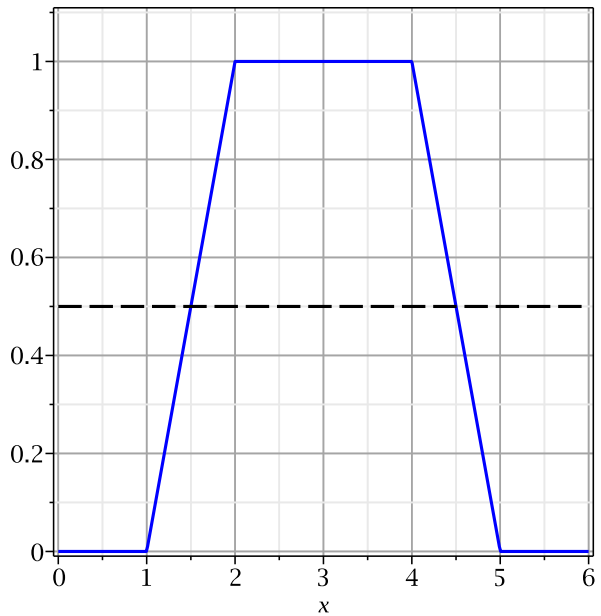
Fuzzy Control

Fuzzy Sets in R

Fuzzy Numbers

- A *piecewise linear fuzzy number* is a fuzzy interval with $L(x) = x$ and $R(x) = 1 - x$ for all $x \in]0, 1]$.
- A piecewise linear fuzzy number $(l, r, \alpha, \beta)_{L,R}$ is *triangular* if $l = r$, and *trapezoidal* if $l < r$.





Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

*Fuzzy Numbers

- A *fuzzy real* is a fuzzy quantity A with a non-decreasing left-continuous membership function.
- A *piecewise linear fuzzy real* is a pair $A = (x_0, x_0 + \epsilon)$ with $x_0 \in \mathbb{R}$ and $\epsilon \in [0, \infty[$ such that

$$\mu_A(x) = \begin{cases} 0 & \text{if } x \in]-\infty, x_0], \\ \frac{x - x_0}{\epsilon} & \text{if } x \in]x_0, x_0 + \epsilon], \\ 1 & \text{if } x \in]x_0 + \epsilon, \infty]. \end{cases} \tag{540}$$

The membership value $\mu_A(x)$ is the truth value of "x *is larger than* x_0".

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

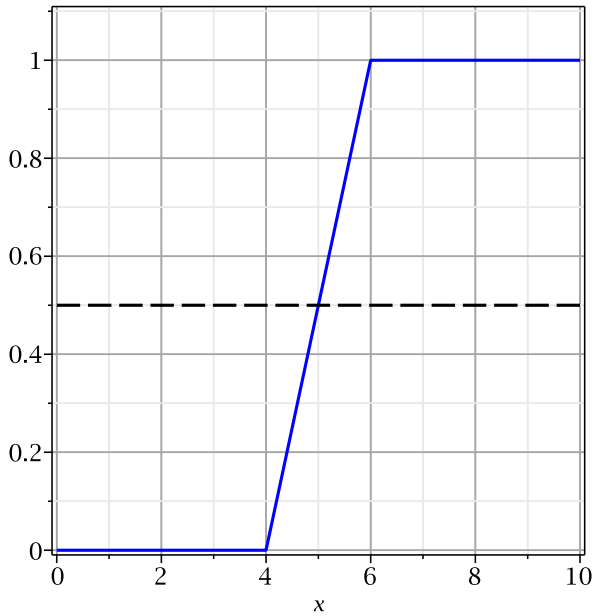
Fuzzy Sets in R

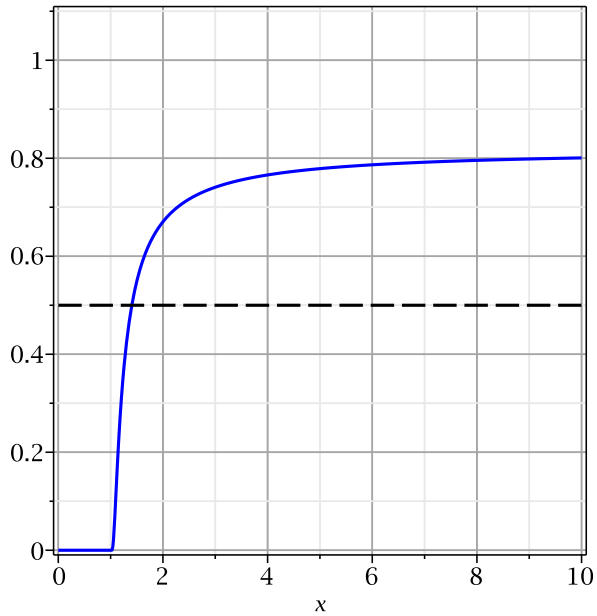
*Fuzzy Numbers

An *infinite fuzzy real* A is given by membership function

$$\mu_A(x) = \begin{cases} 0 & \text{if } x \in]-\infty, x_0], \\ e^{-\frac{x}{x_0 x - x_0^2}} & \text{if } x \in]x_0, \infty[, \end{cases} \qquad x_0 \in]0, \infty[. \quad (541)$$

Piecewise linear fuzzy real and infinite fuzzy real:





Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

*Fuzzy Numbers – Zadeh's extension principle

Given t-norm T and dyadic operation $\cdot : \mathbb{R}^2 \to \mathbb{R}$.

- Define the operation $\odot : \mathcal{F}(\mathbb{R})^2 \to \mathcal{F}(\mathbb{R})$ on the class of fuzzy quantities such that for all $z \in \mathbb{R}$,

$$\mu_{A \odot_T B}(z) = \sup\{T(\mu_A(x), \mu_B(y)) \mid x, y \in \mathbb{R},\ x \cdot y = z\}. \tag{542}$$

- Let A and B be crisp subsets of $\mathbb{R}$. The operation $\odot_T$ is the ordinary complex product

$$A \odot_T B = A \cdot B = \{a \cdot b \mid a \in A, b \in B\} \tag{543}$$

using the characteristic function as membership function.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

*Fuzzy Numbers

- Let T_1 and T_2 be t-norms with $T_1 \leq T_2$. Then for any two fuzzy quantities A and B and any dyadic operation $\cdot : \mathbb{R}^2 \to \mathbb{R}$ we have

$$\mu_{A \odot_{T_1} B} \leq \mu_{A \odot_{T_2} B}. \tag{544}$$

- Let T be a t-norm. Then for any two fuzzy quantities A and B and any dyadic operation $\cdot : \mathbb{R}^2 \to \mathbb{R}$,

$$\mu_{A \odot_{T_D} B} \leq \mu_{A \odot_T B} \leq \mu_{A \odot_{T_M} B}. \tag{545}$$

In the following, we concentrate on the extension $\oplus_T$ of addition operation $+$ on $\mathbb{R}$.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

*Fuzzy Numbers

Let $A = (l_1, r_1, F_1, G_1)$ and $B = (l_2, r_2, F_2, G_2)$ be upper semicontinuous fuzzy numbers.

- Minimum t-norm:

$$\begin{aligned} A \oplus_{T_M} B \quad = \quad & (l_1 + l_2, r_1 + r_2, \\ & ((-F_1)^{(-1)} + (-F_2)^{(-1)})^{(-1)}, \\ & ((-G_1)^{(-1)} + (-G_2)^{(-1)})^{(-1)}), \end{aligned} \tag{546}$$

  where $f^{(-1)}$ is the pseudo-inverse of the monotonous function f,

- Drastic product:

$$\begin{aligned} & A \oplus_{T_D} B = \\ & \quad (l_1 + l_2, r_1 + r_2, \max(F_1, F_2), \max(G_1, G_2)). \end{aligned} \tag{547}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

*Fuzzy Numbers

The *pseudo-inverse* of a monotonous function $f : [a, b] \to [c, d]$ is defined by

$$f^{(-1)}(y) = \sup\{x \in [a, b] \mid (f(x) - y)(f(b) - f(a)) < 0\}. \quad (548)$$

Construct the graph of pseudo-inverse $f^{(-1)}$ of a monotonous function $f : [a, b] \to [c, d]$ as follows:

- Draw vertical line segments at the discontinuities of f.
- Reflect the graph of f at the identity function $y = x$.
- Remove all vertical line segments from the reflected graph except for their lowest points.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

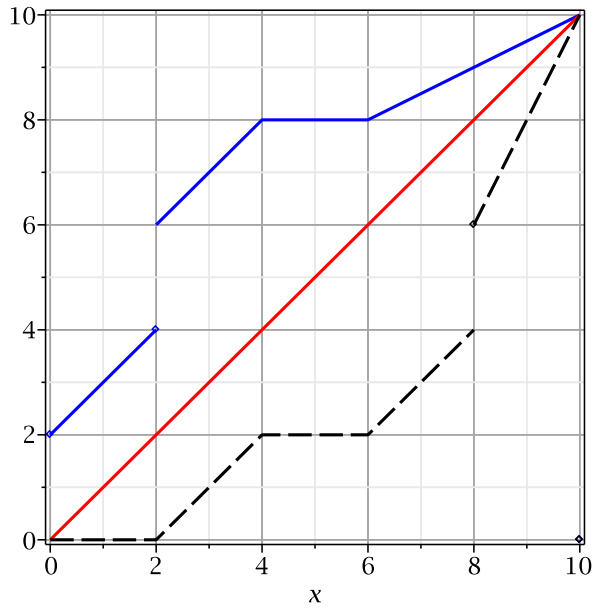
Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

*Fuzzy Numbers

Construction of pseudo-inverse:



Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

*Fuzzy Numbers

Let $A = (l_1, r_1, \alpha_1, \beta_1)_{L,R}$ and $B = (l_2, r_2, \alpha_2, \beta_2)_{L,R}$ be fuzzy intervals with the same shapes L and R.

- Minimum t-norm:

$$A \oplus_{T_M} B = (l_1 + l_2, r_1 + r_2, \alpha_1 + \alpha_2, \beta_1 + \beta_2)_{L,R}. \tag{549}$$

- Drastic product:

$$A \oplus_{T_D} B = (l_1 + l_2, r_1 + r_2, \max(\alpha_1, \alpha_2), \max(\beta_1, \beta_2))_{L,R}. \tag{550}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

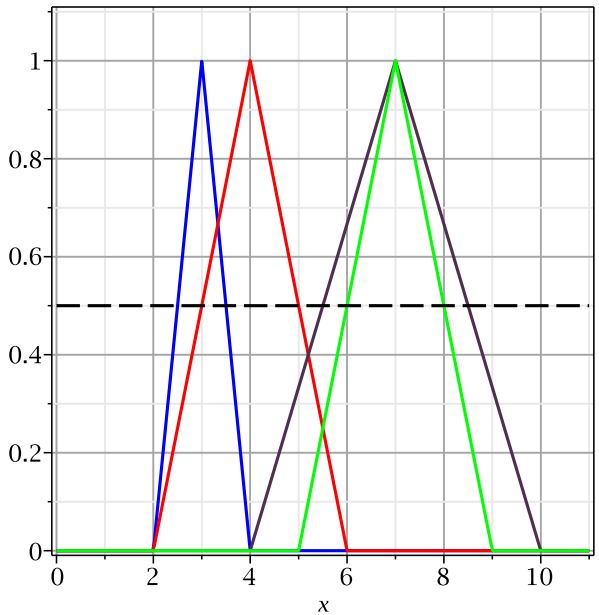
Fuzzy Control

Fuzzy Sets in R

*Fuzzy Numbers

The sum of two triangular fuzzy intervals can be plotted in Maple as follows:

```
> plot([A,B,add_M_AB,add_D_AB,0.5],x=0..11,0..1.1,
   axes=boxed, gridlines=true, thickness=[2,2],
   color=[blue,red,violet,green,black],
   font=[helvetica,12],
   linestyle=[solid,solid,solid,solid,dash]);
```



Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

*Fuzzy Numbers

Let A and B be fuzzy reals.

- Minimum t-norm:

$$A \oplus_{T_M} B = (\mu_A^{(-1)} + \mu_B^{(-1)})^{(-1)} \tag{551}$$

- Drastic product:

$$A \oplus_{T_D} B(x) = \max(\mu(x-a), \mu_B(x-b)), \tag{552}$$

where $a = \inf\{x \in \mathbb{R} \mid \mu_A(x) = 1\}$ and $b = \inf\{x \in \mathbb{R} \mid \mu_B(x) = 1\}$.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

*Fuzzy Numbers

- Let $A = (l_1, r_1, \alpha_1, \beta_1)$ and $B = (l_2, r_2, \alpha_2, \beta_2)$ be piecewise linear fuzzy numbers. Then

$$A \oplus_{T_L} B = (l_1 + l_2, r_1 + r_2, \max(\alpha_1, \alpha_2), \max(\beta_1, \beta_2)) \quad (553)$$

- Let $A = (x_1, \epsilon_1)$ and $B = (x_2, \epsilon_2)$ be piecewise linear fuzzy reals. Then

$$A \oplus_{T_L} B = (x_1 + x_2 + \min(\epsilon_1, \epsilon_2), x_1 + x_2 + \epsilon_1 + \epsilon_2). \quad (554)$$

- Let $A = (x_1, x_1, F_\alpha, F_\alpha)$ and $B = (x_2, x_2, F_\beta, F_\beta)$ be Gaussian fuzzy numbers. Then

$$A \oplus_{T_P} B = (x_1 + x_2, x_1 + x_2, F_{\alpha+\beta}, F_{\alpha+\beta}). \quad (555)$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Inference

Fuzzy inference is the process of finding a mapping from an input fuzzy set to an output fuzzy set. Application is decision making.

- Fuzzy product and fuzzy relation
- Zadeh's compositional rule of inference
- Zadeh's fuzzy relational equation

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Inference

Let X and Y be two universes, $A \subseteq X$ and $B \subseteq Y$ fuzzy subsets, and T be a t-norm.
Define the *product* $A \times_T B$ as the fuzzy subset of the (Cartesian product) universe $X \times Y$ as

$$\mu_{A \times_T B}(x, y) = T(\mu_A(x), \mu_B(y)). \tag{556}$$

The value $\mu_{A \times_T B}(x, y)$ can be interpreted as the truth value of "x *is element of* A *and* y *is element of* B".

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

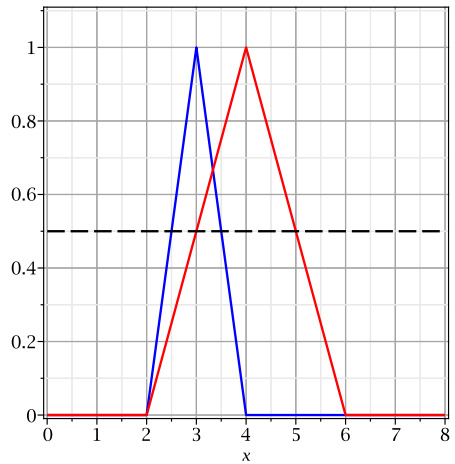
Fuzzy Control

Fuzzy Sets in R

Fuzzy Inference

Given fuzzy subsets A and B of $X = \mathbb{R}$ below.
Consider the t-norms T_M, T_P, and T_L defined in `Maple` as follows,

```
> M := (x,y) -> min(A(x),B(y));
> P := (x,y) -> A(x) * B(y);
> L := (x,y) -> max(A(x)+B(y)-1,0);
```



Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

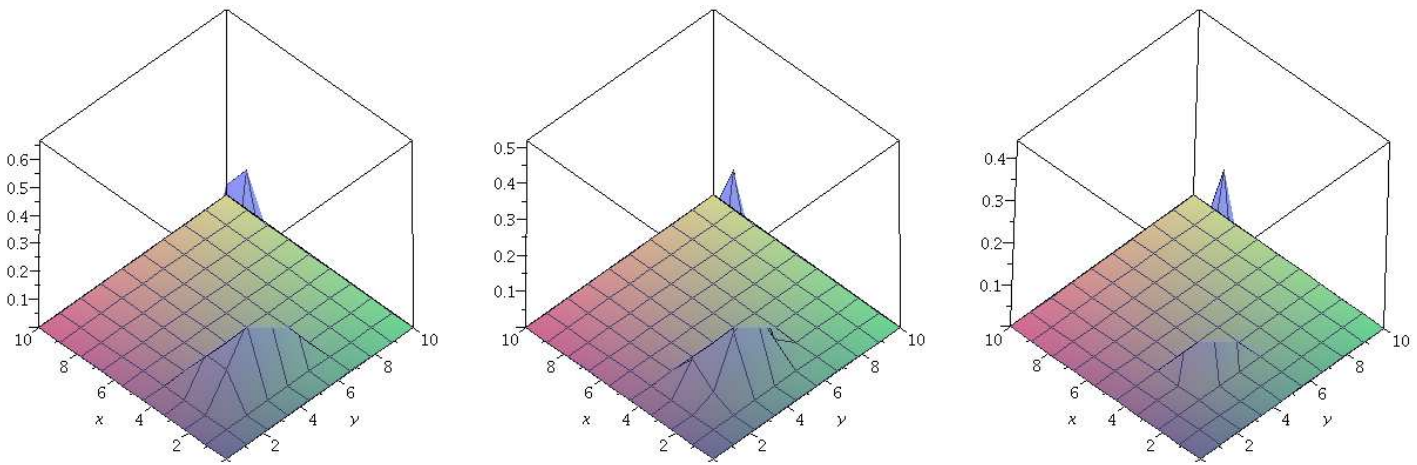
Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Inference

Fuzzy products $A \times_T B$ with t-norms T_M, T_P, and T_L:



Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Inference

- For two crisp sets X and Y, the Cartesian product $X \times Y$ is a crisp set.
- Each crisp subset R of $X \times Y$ is called a *relation* on $X \times Y$.
- More generally, a *fuzzy relation* R on $X \times Y$ is a fuzzy subset of $X \times Y$ given by its membership function $\mu_R : X \times Y \to [0, 1]$.
- The value $\mu_R(x, y)$ can be interpreted as the truth value of "x *and* y *are in relation* R".

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Inference – Zadeh's Compositional Inference Rule

Let T be a t-norm, A be a fuzzy subset of X, and R be a fuzzy relation on $X \times Y$.
The fuzzy subset $A \circ_T R$ of Y is defined by its membership function

$$\mu_{A \circ_T R}(y) = \sup\{T(\mu_A(x), \mu_R(x, y)) \mid x \in X\}. \tag{557}$$

A fuzzy subset A of X is associated to a fuzzy subset $B = A \circ_T R$ of Y given by the membership function $\mu_{A \circ_T R}$.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Inference – Zadeh's Compositional Inference Rule

Given a function $f : X \to Y$ by its graph

$$R = \{(x, f(x)) \mid x \in X\}. \tag{558}$$

If A is a crisp subset of X, then $A \circ_T R$ is a crisp subset of Y given by the image $f(A)$.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Inference – Zadeh's Compositional Inference Rule

Evaluation of Zadeh's rule:

- Cylindric extension of fuzzy set A to fuzzy set $A \times Y$:

$$\mu_{A\times Y}(x,y) = \mu_A(x). \tag{559}$$

- Intersection of fuzzy set $A \times Y$ with fuzzy relation R:

$$\begin{aligned} \mu_{(A\times Y)\cap_T R}(x,y) &= T(\mu_{A\times Y}(x,y), \mu_R(x,y)) \\ &= T(\mu_A(x), \mu_R(x,y)). \end{aligned} \tag{560}$$

- Projection of fuzzy set $(A \times Y) \cap_T R$ onto Y:

$$\begin{aligned} \mu_{A\circ_T R}(y) &= \sup\{\mu_{(A\times Y)\cap_T R}(x,y) \mid x \in X\} \\ &= \sup\{T(\mu_A(x), \mu_R(x,y) \mid x \in X\}. \end{aligned} \tag{561}$$

Fuzzy Inference - Zadeh's Fuzzy Relational Equation

Given fuzzy subset A of X, fuzzy subset B of Y, and t-norm T.

- Find a fuzzy relation R on $X \times Y$ such that

$$A \circ_T R = B. \tag{562}$$

- Fuzzy relation $R_T(A, B)$ on $X \times Y$ given by the membership function provides the fuzzy relation $A \circ_T R = B$ (T left-continuous),

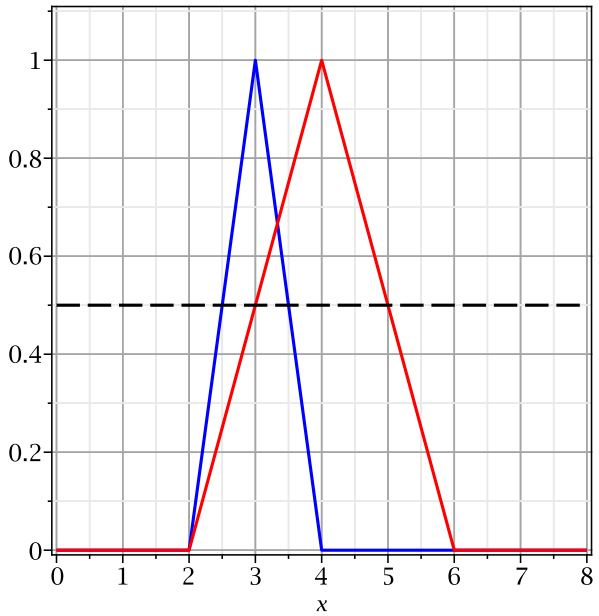
$$\mu_{R_T(A,B)}(x, y) = \mu_A(x) \to_T \mu_B(y). \tag{563}$$

- T-residuum $\to_T$ is given by

$$\begin{aligned} &\mu_A(x) \to_T \mu_B(y) \\ &= \sup\{z \in [0,1] \mid T(\mu_A(x), z) \leq \mu_B(y)\}. \end{aligned} \tag{564}$$

Fuzzy Inference – Example

Consider the fuzzy subsets A and B of $\mathbb{R}$ below.



Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

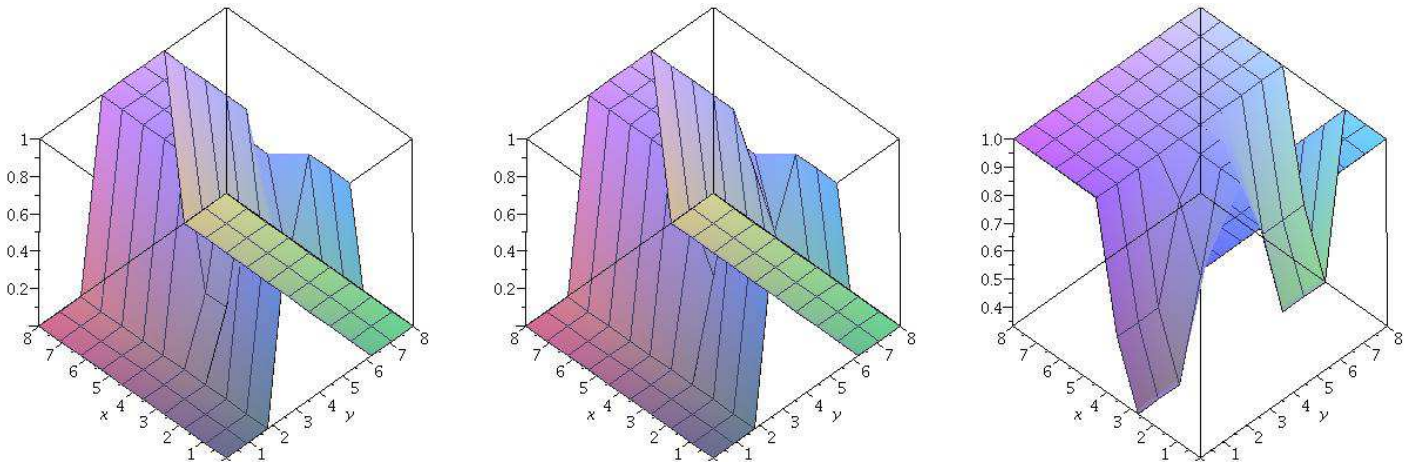
Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Inference – Example (cont'd)

Fuzzy relations $R_T(A, B)$ using t-norms T_M, T_P, and T_L:



Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

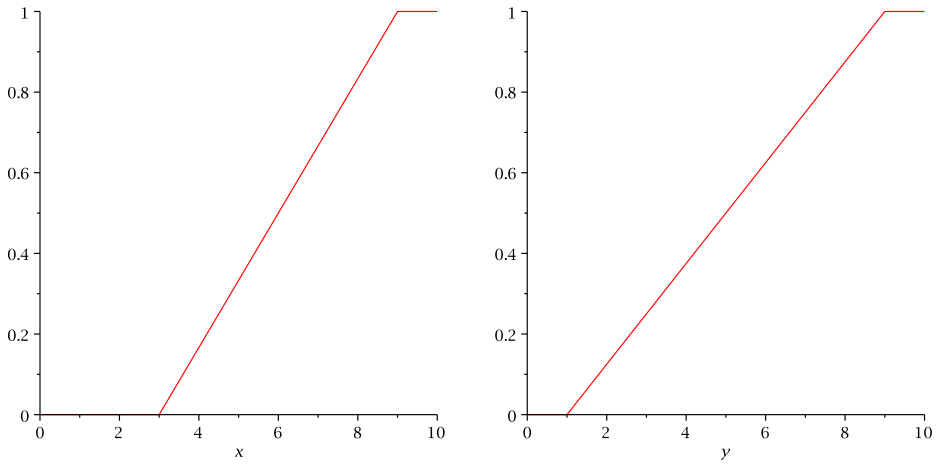
Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

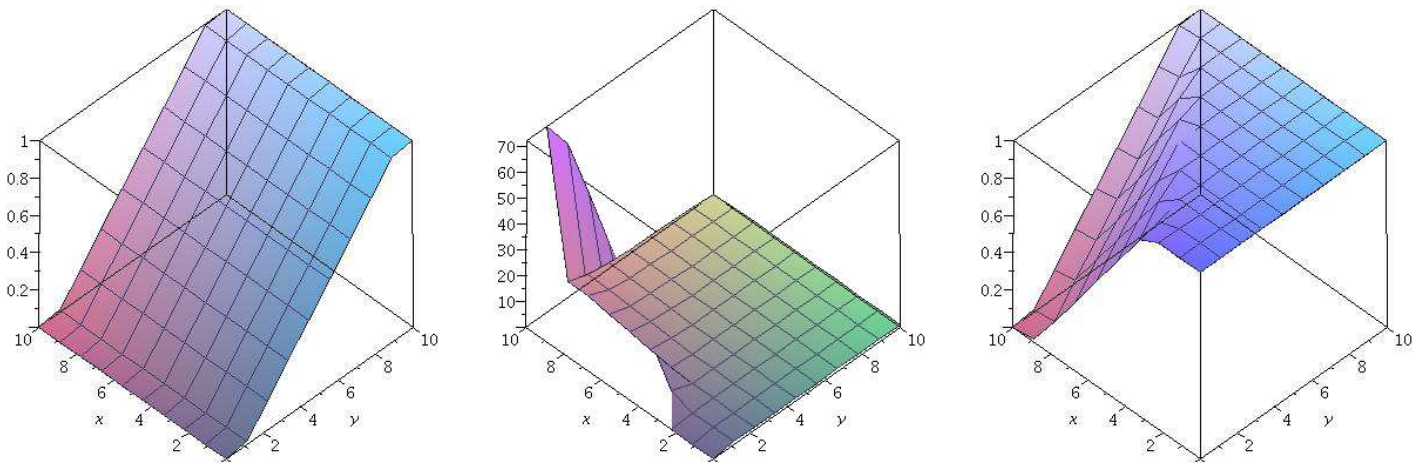
Fuzzy Inference – Example

- Customers of a restaurant rate the quality of service (QoS) and the quality of food (QoF) on a scale of 0 to 10.
- Suppose QoS and QoF are given by fuzzy numbers A and B below, resp.
- The relation between QoS and QoF can be measured by fuzzy relation $R_T(A, B)$ w.r.t. t-norm T.



Fuzzy Inference – Example (cont'd)

Fuzzy relation between QoS and QoF using t-norms T_M, T_P, and T_L:



Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Control

A fuzzy control system is a control system based on fuzzy logic – no learning involved. Application is machine control.

- Mamdani controller
- Takagi-Sugeno controller

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Control – Mamdani Controller

Given input universe X,

- $A_1, \ldots, A_n$ normal fuzzy subsets of X,
- $B_1, \ldots, B_n$ normal fuzzy subsets of $\mathbb{R}$,
- T t-norm.

Consider the *rulebase*

$$\begin{aligned} &\text{IF } x \text{ is } A_1, \text{ THEN } y \text{ is } B_1, \\ &\text{IF } x \text{ is } A_2, \text{ THEN } y \text{ is } B_2, \\ &\vdots \\ &\text{IF } x \text{ is } A_n, \text{ THEN } y \text{ is } B_n. \end{aligned} \tag{565}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Control – Mamdani Controller

- Membership function of fuzzy relation $R \subseteq X \times \mathbb{R}$,

$$\mu_R(x,y) = \max\{T(\mu_{A_1}(x),\mu_{B_1}(y)),\ldots,T(\mu_{A_n}(x),\mu_{B_n}(y))\}. \tag{566}$$

- I/O function $F_M : X \to \mathbb{R}$ of Mamdani controller,

$$F_M(x) = \frac{\int_{\mathbb{R}} \mu_R(x,y)\cdot y dy}{\int_{\mathbb{R}} \mu_R(x,y) dy}, \tag{567}$$

where $\int_{\mathbb{R}} \mu_R(x,y)dy > 0$ for all $x \in X$ (requires measure theoretic assumptions).

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

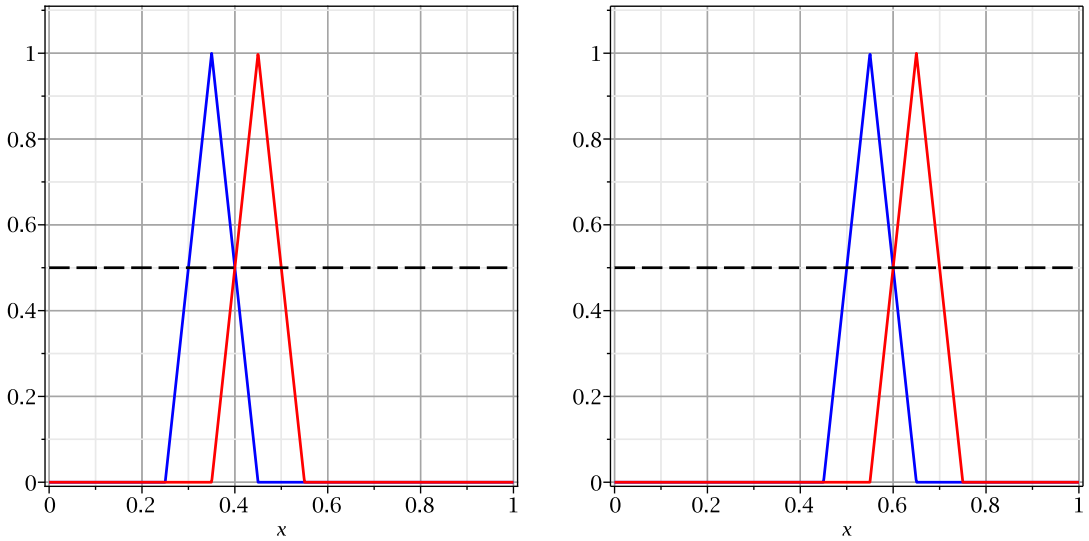
Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Control – Mamdani Controller

Input given by A_1, B_1 and A_2, B_2:



Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Control – Mamdani Controller

Integration via `Maple`:

```
> f:=int(max(A1(x)*B1(y),A2(x)*B2(y)),y=0..1);
```

Output

$$f(x) = 0.10 \begin{cases} 0 & x < 0.45, \\ -4.5 + 10x & x < 0.55, \\ 6.5 - 10x & x \leq 0.65, \\ 0 & 0.65 < x \end{cases}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Control – Mamdani Controller

Integration via `Maple`:

```
> g:=int(max(A1(x)*B1(y)*y,A2(x)*B2(y)*y),y=0..1);
```

Output

$$g(x) = 0.065 \cdot \begin{cases} 0 & x < 0.45, \\ -4.5 + 10x & x < 0.55, \\ 6.5 - 10x & x \leq 0.65, \\ 0 & 0.65 < x \end{cases}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Control – Takagi-Sugeno Controller

Given input universe X,

- $A_1, \ldots, A_n$ normal fuzzy subsets of X with $\sum_{i=1}^{n} \mu_{A_i}(x) > 0$ for all $x \in X$,
- $g_1, \ldots, g_n$ mappings from X to $\mathbb{R}$.

Consider the *rulebase*

$$\begin{aligned} &\text{IF } x \text{ is } A_1, \text{ THEN } y = g_1(x), \\ &\text{IF } x \text{ is } A_2, \text{ THEN } y = g_2(x), \\ &\vdots \\ &\text{IF } x \text{ is } A_n, \text{ THEN } y = g_n(x). \end{aligned} \tag{568}$$

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Control – Takagi-Sugeno Controller

I/O function $F_{TS} : X \to \mathbb{R}$ of Takagi-Sugeno controller,

$$F_{TS}(x) = \frac{\sum_{i=1}^{n} \mu_{A_i}(x) \cdot g_i(x)}{\sum_{i=1}^{n} \mu_{A_i}(x)}. \tag{569}$$

The Takagi-Sugeno controller can be viewed as a special case of the Mamdani controller if the involved functions $g_1, \ldots, g_n$ are constant; i.e., $g_i(x) = y_i$ for all $x \in X$ and $1 \leq i \leq n$.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Control – Basic Tipping Problem

A customer evaluates QoF and QoS and then decides upon the tip using the golden rules:

- If food is rancid and service is poor, tip is cheap.
- If service is good, tip is average.
- If food is delicious and service is excellent, tip is generous.

Cheap tip is 5 %, average tip is 15 %, and generous tip is 25 % of the total bill.

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

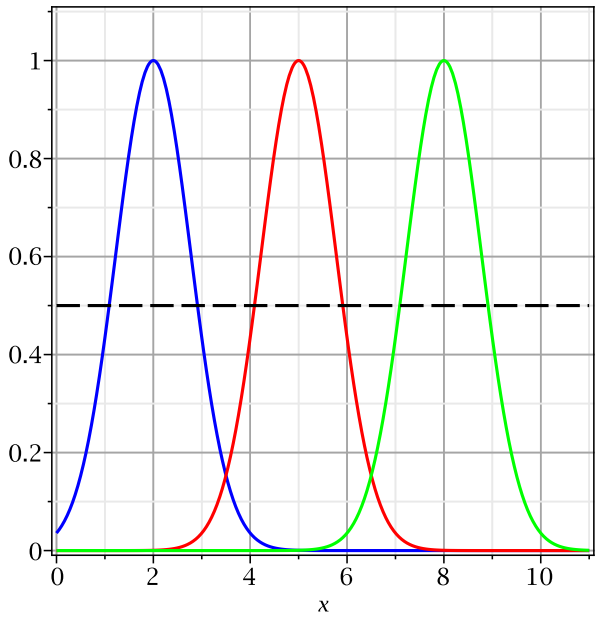
Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Control – Basic Tipping Problem (cont'd)

Customers rate QoS on a scale of 1 to 10.
Fuzzy numbers for QoS: poor (q_1), good (q_2), and excellent (q_3):



Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

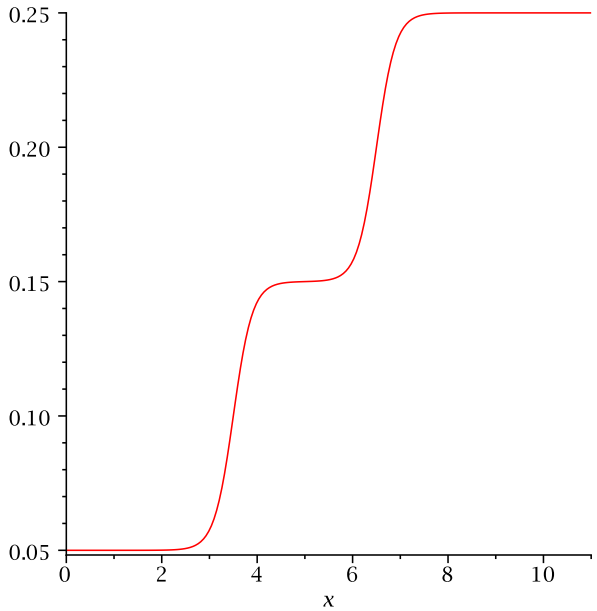
Fuzzy Control

Fuzzy Sets in R

Fuzzy Control – Basic Tipping Problem (cont'd)

Takagi-Sugeno controller:

$$F_{TS}(x) = \frac{q_1(x) \cdot 0.05 + q_2(x) \cdot 0.15 + q_3(x) \cdot 0.25}{q_1(x) + q_2(x) + q_3(x)}, \ 1 \leq x \leq 10.$$



Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Sets in R

The R package `sets` provides data structures and basic operations for ordinary sets and generalizations such as fuzzy sets.

```
> library(sets)
```

Fix a universe to work with,

```
> sets_options("universe", seq(1, 100, 0.5))
```

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Sets in R

Specify Gaussian fuzzy numbers corresponding to temperature (Fahrenheit), humidity (percentage), precipitation (rain), and weather (for output).

```
> variables <- set(
    temp = fuzzy_partition(varnames=c(cold=30,
                  good=70, hot=90), sd=5.0),
    hum = fuzzy_partition(varnames=c(dry=30,
                  good=70, wet=80), sd=3.0),
    prec = fuzzy_partition(varnames=c(no.rain=30,
                  little.rain=60, rain=90), sd=7.5),
    weather = fuzzy_partition(varnames=c(bad=40, ok=65,
                  perfect=80),
                  FUN=fuzzy_cone, radius=10)
                  // function generator
  )
```

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Sets in R

Specification of fuzzy rules:

```
> rules <- set(
    fuzzy_rule(temp %is% good && hum %is% dry
      && prec %is% no.rain, weather %is% perfect),
    fuzzy_rule(temp %is% hot && hum %is% wet
      && prec %is% rain, weather %is% bad),
    fuzzy_rule(temp %is% cold, weather %is% bad),
    fuzzy_rule(temp %is% good || hum %is% good
      || prec %is% little.rain, weather %is% ok),
    fuzzy_rule(temp %is% hot && prec %is% little.rain,
      weather %is% ok),
    fuzzy_rule(temp %is% hot && hum %is% dry
      && prec %is% little.rain, weather %is% ok)
  )
```

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Sets in R

Build a system for fuzzy inference:

```
> model <- fuzzy_system(variables, rules)
```

The printing of the specified model yields a summary of the data.

```
> print(model)
```

```
A fuzzy system consisting of 4 variables and 6 rules.

Variables:
weather (bad, ok, perfect)
hum (dry, good, wet)
temp (cold, good, hot)
prec (no.rain, little.rain, rain)

Rules:
temp %is% good && hum %is% dry && prec %is% no.rain ==> weather %is% perfect
temp %is% hot && hum %is% wet && prec %is% rain ==> weather %is% bad
temp %is% cold ==> weather %is% bad
%temp %is% good || hum %is% good || prec %is% little.rain ==> weather %is% ok
temp %is% hot && prec %is% little.rain ==> weather %is% ok
temp %is% hot && hum %is% dry && prec %is% little.rain ==> weather %is% ok
```

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

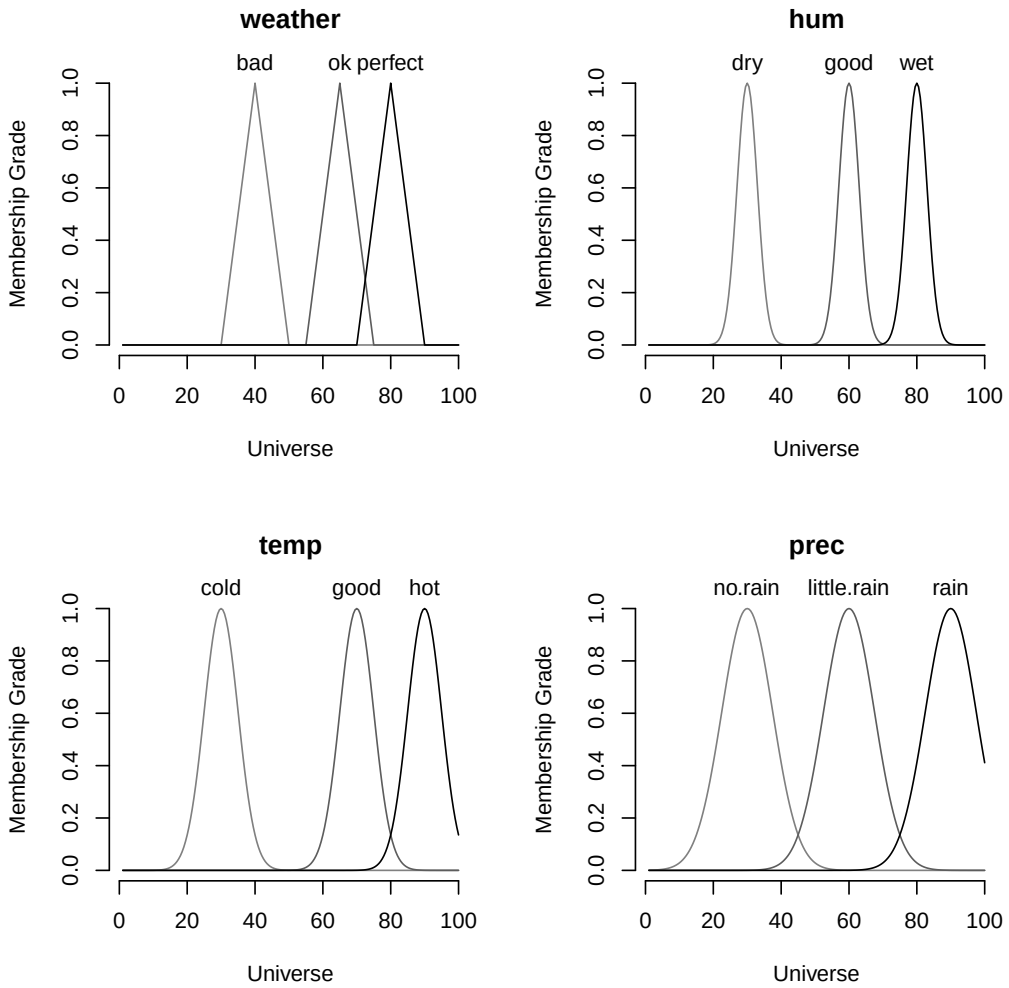
Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Sets in R

Plot of the fuzzy numbers:



Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Sets in R

Fuzzy inference with temperature of 75^oF, zero humidity, and precipitation of 70%:

```
> ex <- fuzzy_inference(model,
          list(temp=75, hum=0, prec=70))
```

Defuzzification transforms the resulting fuzzy number into a number of the universe:

```
> gset_defuzzify(ex, "centroid")
[1] 65
```

Thus according to the inference, the weather is 65 and so `ok` (see `weather` plot).

Fuzzy Sets in R

Reconsider the tipping problem.

```
> sets_options("universe", seq(from=0, to=25, by=0.1))
```

First set up the fuzzy variables.

```
> variables <- set(
    service=fuzzy_partition(
      varnames=c(poor=0, good=5, excellent=10), sd=3),
    food=fuzzy_variable(
      rancid=fuzzy_trapezoid(corners=c(-2,0,2,4)),
      delicious=fuzzy_trapezoid(corners=c(7,9,11,13))),
    tip=fuzzy_partition(
      varnames=c(cheap=5, average=12, generous=20),
    FUN=fuzzy_cone, radius=5))
```

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Sets in R

Next set up the fuzzy rules.

```
> rules <- set(
    fuzzy_rule(service %is% poor
      || food %is% rancid, tip %is% cheap),
    fuzzy_rule(service %is% good, tip %is% average),
    fuzzy_rule(service %is% excellent
      || food %is% delicious, tip %is% generous))
```

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Sets in R

The fuzzy variables and rules are combined into a fuzzy system.

```
> system <- fuzzy_system(variables, rules)
```

The data of the fuzzy system can be printed.

```
> print(system)
```

```
A fuzzy system consisting of 3 variable and 3 rules.

Variables:
food(rancid, delicious)
tip(cheap, average, generous)
service(poor, good, excellent)

Rules:
service %is% poor || food %is% rancid => tip %is% cheap
service %is% good => tip %is% average
service %is% excellent || food %is% delicious
                          => tip %is% generous
```

Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

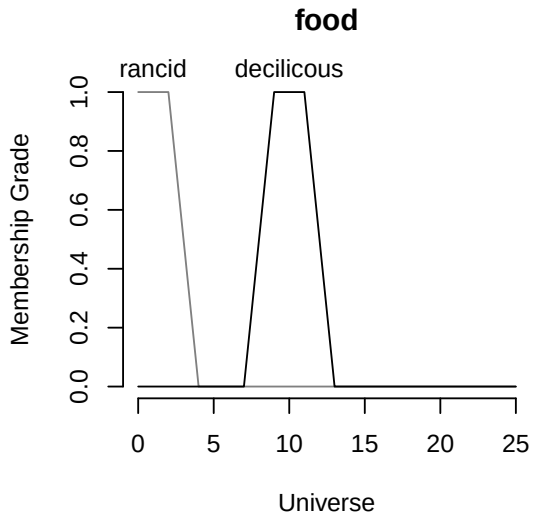
Fuzzy Inference

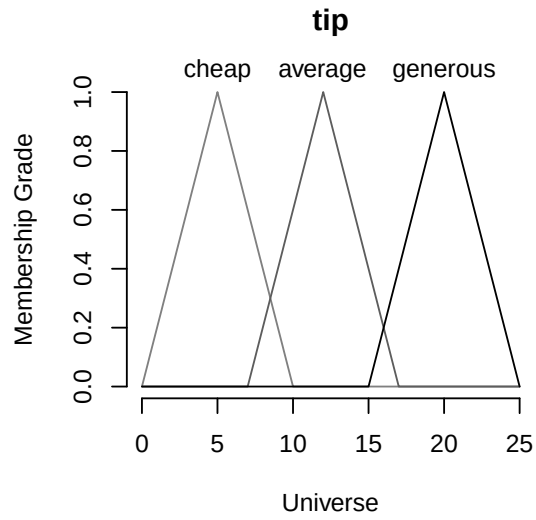
Fuzzy Control

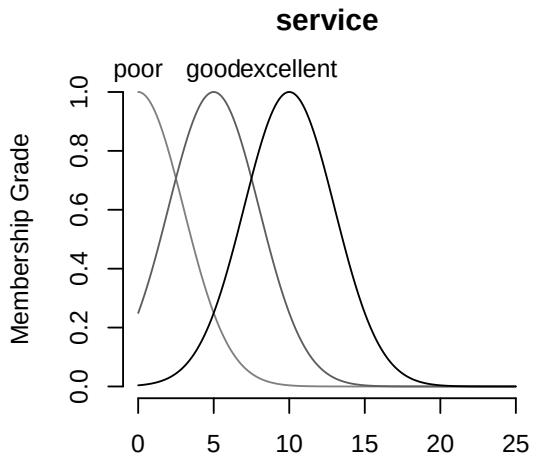
Fuzzy Sets in R

Fuzzy Sets in R

```
> plot(system)
```







Soft Computing

K.-H. Zimmermann

Contents

Triangular Norms

Properties of Triangular Norms

Fuzzy Sets

Fuzzy Numbers

Fuzzy Inference

Fuzzy Control

Fuzzy Sets in R

Fuzzy Sets in R

Fuzzy inference with service 3 and food 8:

```
> fi <- fuzzy_inference(system, list(service=3, food=8))
```

Defuzzification transforms the resulting fuzzy number into a number of the universe:

```
> gset_defuzzify(fi, "centroid")
[1] 9.433852
```

Thus according to the inference, the tip is 9.4 and so `average` (see `foot` plot).