Function approximation

VL Algorithmisches Lernen, 12

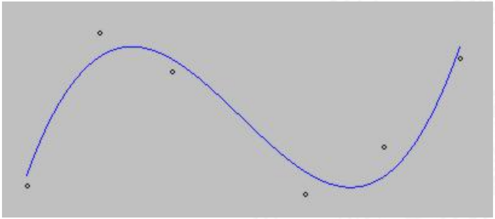
Jianwei Zhang

University of Hamburg
MIN Faculty, Dept. of Informatics
Vogt-Kölln-Str. 30, D-22527 Hamburg
zhang@informatik.uni-hamburg.de

30/06/2009

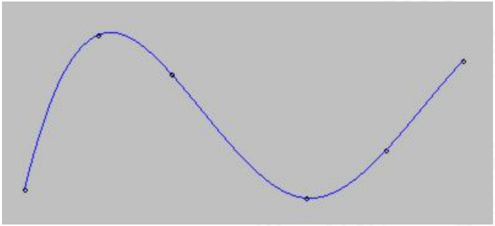
Approximation

Approximation of the relation between $\mathbf{x}$ and y (curve, plane, hyperplane) with a different function, given a limited number of data points $D = \{\mathbf{x}_i, y_i\}_{i=1}^{l}$.



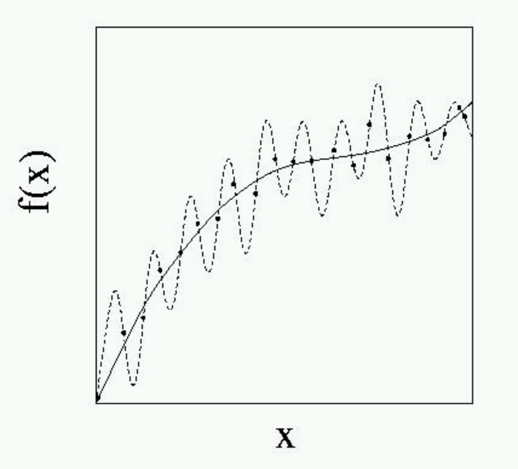
Approximation vs. Interpolation

A special case of approximation is interpolation:
the model exactly matches all data points.



If many data points are given or measurement data is affected by noise, approximation is preferably used.

Approximation without *Overfitting*



Interpolation with Polynomials

Polynomial interpolation:

- Lagrange polynomial,
- Newton polynomial,
- Bernstein polynomial,
- Basis-Splines.

Lagrange interpolation

To match $l+1$ data points (x_i, y_i) $(i = 0, 1, \ldots, l)$ with a polynomial of degree l , the following approach of LAGRANGE can be used:

$$p_l(x) = \sum_{i=0}^{l} y_i L_i(x)$$

The interpolation polynomial in the Lagrange form is defined as follows:

$$L_i(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_l)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_l)}$$

$$= \begin{cases} 1 \text{ if } x = x_i \\ 0 \text{ if } x \neq x_i \end{cases}$$

Newton Interpolation

The Newton basis polynomials of degree l are constructed as follows:

$$p_l(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)(x-x_1) + \cdots + a_l(x-x_0)(x-x_1)\cdots(x-x_{l-1})$$

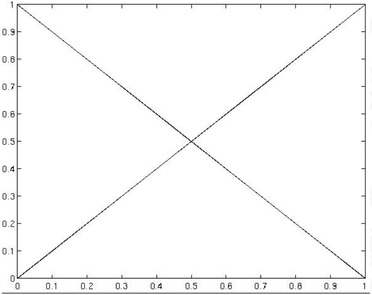
This approach enables us to calculate the coefficients easily.
For $n = 2$ the following system of equations is obtained:

$$\begin{aligned} p_2(x_0) &= a_0 &&= y_0 \\ p_2(x_1) &= a_0 + a_1(x_1 - x_0) &&= y_1 \\ p_2(x_2) &= a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) &&= y_2 \end{aligned}$$

Interpolation with Bernstein polynomials - I

Interpolation of two points with Bernstein polynomials:

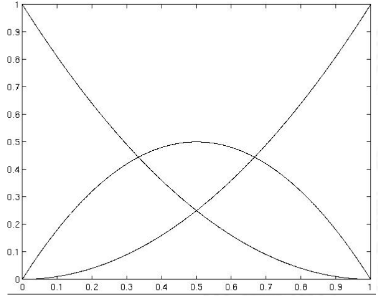
$$\mathbf{y} = \mathbf{x}_0 B_{0,1}(t) + \mathbf{x}_1 B_{1,1}(t) = \mathbf{x}_0(1-t) + \mathbf{x}_1 t$$



Interpolation with Bernstein polynomials - II

Interpolation of three points with Bernstein polynomials:

$$\mathbf{y} = \mathbf{x}_0 B_{0,2}(t) + \mathbf{x}_1 B_{1,2}(t) + \mathbf{x}_2 B_{2,2}(t) = \mathbf{x}_0(1-t)^2 + \mathbf{x}_1 2t(1-t) + \mathbf{x}_2 t^2$$

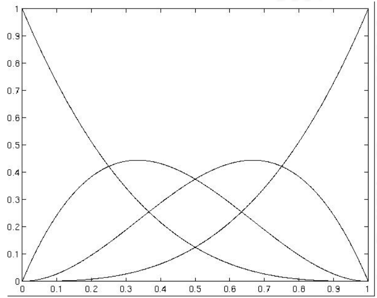


Interpolation with Bernstein polynomials - III

Interpolation of four points with Bernstein polynomials:

$$\mathbf{y} = \mathbf{x}_0 B_{0,3}(t) + \mathbf{x}_1 B_{1,3}(t) + \mathbf{x}_2 B_{2,3}(t) \mathbf{x}_3 B_{3,3}(t)$$

$$= \mathbf{x}_0 (1-t)^3 + \mathbf{x}_1 3t(1-t)^2 + \mathbf{x}_2 3t^2(1-t) + \mathbf{x}_3 t^3$$



Interpolation with Bernstein polynomials - IV

The Bernstein polynomials of degree $k+1$ are defined as follows:

$$B_{i,k}(t) = \binom{k}{i}(1-t)^{k-i}t^i, \quad i = 0, 1, \ldots, k$$

Interpolation with Bernstein polynomials $B_{i,k}$:

$$\mathbf{y} = \mathbf{x}_0 B_{0,k}(t) + \mathbf{x}_1 B_{1,k}(t) + \cdots + \mathbf{x}_k B_{k,k}(t)$$

B-Splines

A normalized B-Splines $N_{i,k}$ of degree k is defined as follows: For $k = 1$,

$$N_{i,k}(t) = \begin{cases} 1 & : \text{ for } t_i \leq t < t_{i+1} \\ 0 & : \text{ else} \end{cases}$$

and for $k > 1$, the recursive definition:

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t)$$

with $i = 0, \ldots, m$.

B-Spline-Curve

A **B-Spline-Curve** of degree k is a composite function built piecewise from **basis B-Splines** resulting in a polynomial of degree $(k-1)$ that is (k-2)-times continuously differentiable (class C^{k-2}) at the borders of the segments.
The Curve is constructed by polynomials, that are defined by the following parameters:

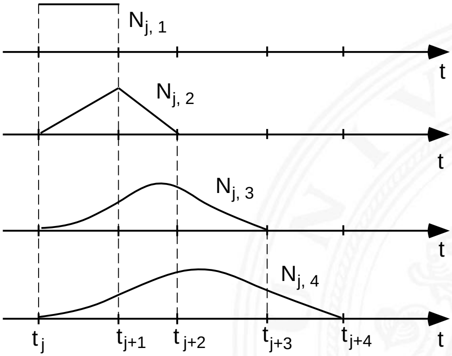
$$\mathbf{t} = (t_0, t_1, t_2, \ldots, t_m, t_{m+1}, \ldots, t_{m+k}),$$

where

- m: depending on the number of data-points
- k: the fixed degree of the B-Spline curve

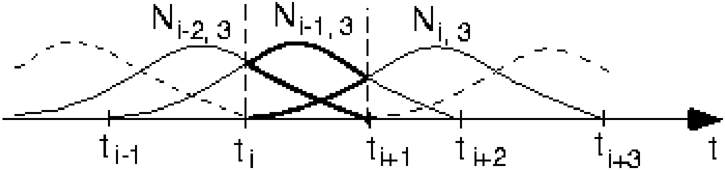
Examples of B-Splines

B-Splines with degree 1, 2, 3 and 4:



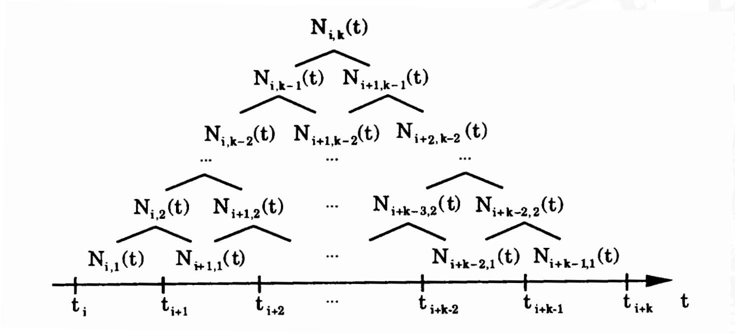
Between the interval of parameters *k* B-Splines are overlapping.

Examples of cubic B-Splines



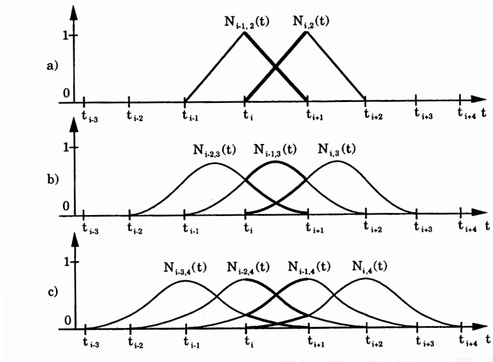
B-Splines of degree k - I

The recursive definition procedure of a B-Spline basis function $N_{i,k}(t)$:

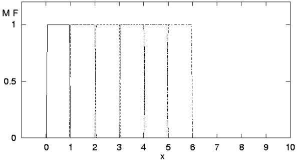


B-Splines of degree k - II

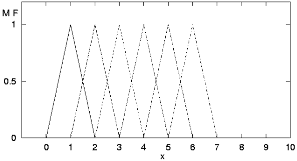
Current segments of B-Spline basis functions of degree 2, 3 and 4 for $t_i \leq t < t_{i+1}$:



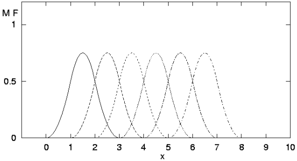
Uniform B-spline of degree 1 to 4



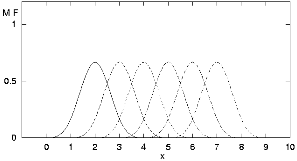
k=1



k=2



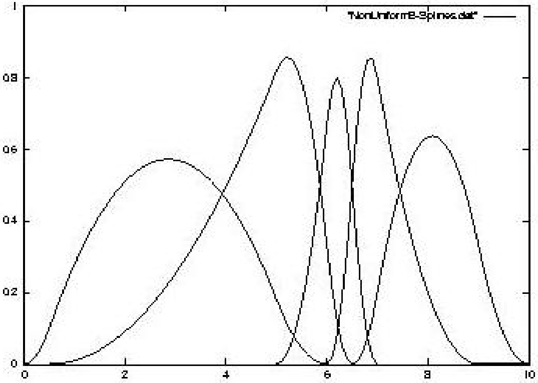
k=3



k=4

Non-uniform B-Splines

Degree 3:



Properties of B-splines

Partition of unity:	$\sum_{i=0}^{k} N_{i,k}(t) = 1.$
Positivity:	$N_{i,k}(t) \geq 0.$
Local support:	$N_{i,k}(t) = 0$ for $t \notin [t_i, t_{i+k}].$
C^{k-2} continuity:	If the knots $\{t_i\}$ different in pairs then $N_{i,k}(t) \in C^{k-2}$, i.e. $N_{i,k}(t)$ is $(k-2)$ times continuously differentiable.

Construction of B-spline curves

A B-spline curve can be constructed blending a number of predefined values (data-points) with B-splines

$$\mathbf{r}(\mathrm{t}) = \sum_{\mathrm{j}=0}^{\mathrm{m}} \mathbf{v}_{\mathrm{j}} \cdot \mathrm{N}_{\mathrm{j,k}}(\mathrm{t})$$

where $\mathbf{v}_j$ are called *control points* (*de Boor-points*).

Let t be a given parameter, then $\mathbf{r}(t)$ is a point of the B-spline curve.

If t varies from t_{k-1} to t_{m+1}, then $\mathbf{r}(\mathrm{t})$ is a (k-2)-times continuously differentiable function (class C^{k-2}).

Calculation of control points from data points

The points $\mathbf{v}_j$ are only identical with the data points if $k = 2$ (interpolation/otherwise approximation). The control points form a convex hull of the interpolation curve. Two methods for the calculation of control points from data points:

Calculation of control points from data points

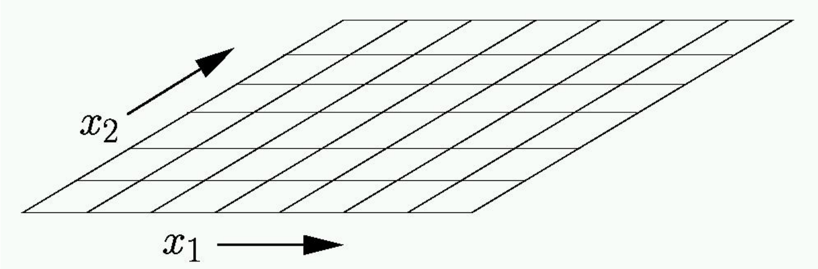
1 Solving the following system of equations (**Böhm84**):

$$\mathbf{q}_j(t) = \sum_{j=0}^{m} \mathbf{v}_j \cdot N_{j,k}(t)$$

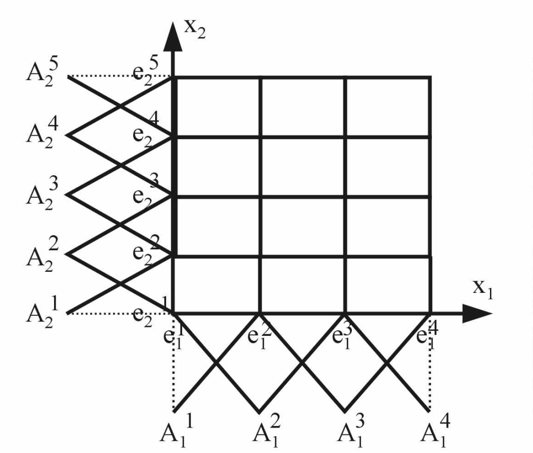
where $\mathbf{q}_j$ are the data-points for interpolation/approximation, $j = 0, \cdots, m$.

2 Learning based on gradient descent(**Zhang98**).

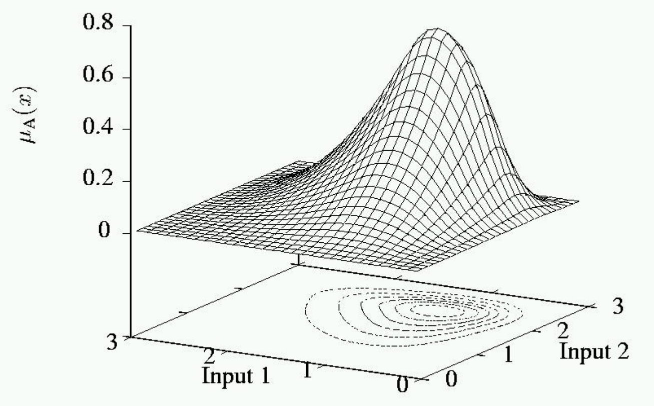
Lattice - I



Lattice - II


x_2
x_1
A_2^5
A_2^4
A_2^3
A_2^2
A_2^1
e_2^5
e_2^4
e_2^3
e_2^2
e_2^1
e_1^1
e_1^2
e_1^3
e_1^4
A_1^1
A_1^2
A_1^3
A_1^4

Tensor 2D-NURBS



Real-world Problems

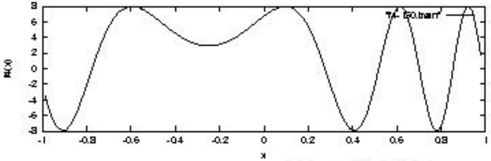
- **modeling**: learning from examples, self-optimized formation, prediction, ...
- **control**: perception-action cycle, state control, Identification of dynamic systems,

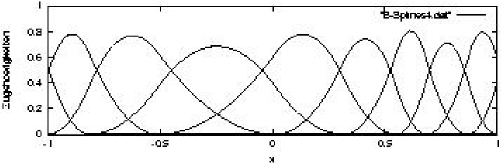
  ...

Function approximation as a benchmark for the choice of a model

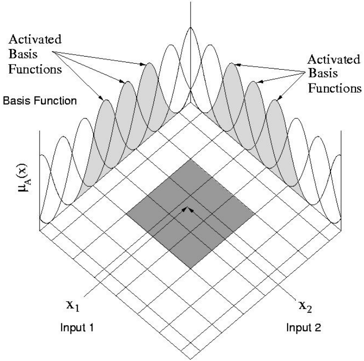
Function approximation - 1D example

An example function $f(x) = 8sin(10x^2 + 5x + 1)$ with $-1 < x < 1$ and the correctly distributed B-Splines:





Lattice



The B-spline model – a two-dimensional illustration.

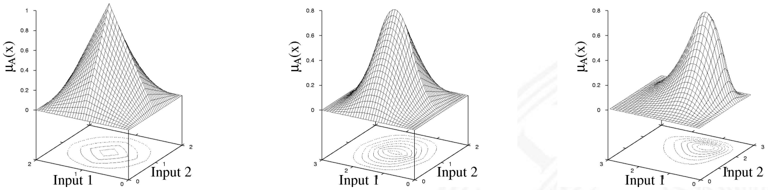
Lattice (cont.)

Every n-dimensional square $(n > 1)$ is covered by the j^{th} multivariate B-spline $N_k^j(x)$. $N_k^j(x)$ is defined by the tensor of n univariate B-splines:

$$N_k^j(x) = \prod_{j=1}^{n} N_{i_j,k_j}^j(x_j) \tag{1}$$

Therefore the shape of each B-spline, and thus the shape of multivariate ones (Figure 2), is implicitly set by their order and their given knot distribution on each input interval.

Lattice (cont.)



(a) Tensor of two, order 2 univariate B-splines.

(b) Tensor of one order 3 and one order 2 univariate B-splines.

(c) Tensor of two univariate B-splines of order 3.

Bivariate B-splines formed by taking the tensor of two univariate B-splines.

General requirements for an approximator

- Universality: Approximation of arbitrary functions
- Generalization: good approximation without *Overfitting*
- Adaptivity: on the basis of new data
- Parallelism: Computing based on biological models
- Interpretability: at least “Grey-box” instead of “Black-box”

Importance of the Interpretability of a Model

Richard P. Feynman: “the way we have to describe nature is generally incomprehensible to us”.

Albert Einstein: “it should be possible to explain the laws of physics to a barmaid”.

Importance of the Interpretability of a Model (cont.)

Important reasons for the symbolic interpretability of an approximator:

- Linguistic modeling is a basis of skill transfer from an expert to a computer or robot .
- Automated learning of a transparent model facilitates the analysis, validation and monitoring in the development cycle of a model or a controller.
- Transparen models provide diverse applications in *Decision-Support Systems*.

B-Spline ANFIS

In a B-Spline ANFIS with n inputs $x_1, x_2, \ldots, x_n$, the rules are used the following form:
$\{Rule(i_1, i_2, \ldots, i_n)$: IF ($x_1$ IS $N^1_{i_1,k_1}$) AND (x_2 IS $N^2_{i_2,k_2}$) AND ...
AND (x_n IS $N^n_{i_n,k_n}$) THEN y IS $Y_{i_1 i_2 \ldots i_n}\}$,
where

- x_j: input j ($j = 1, \ldots, n$),
- k_j: degree of B-spline basis function for x_j,
- $N^j_{i_j,k_j}$: with the i-th linguistic term for the x_j-associated B-spline function,
- $i_j = 0, \ldots, m_j$, partitioning of input j,
- $Y_{i_1 i_2 \ldots i_n}$: control points for $Rule(i_1, i_2, \ldots, i_n)$.
- the "AND"-operator: product

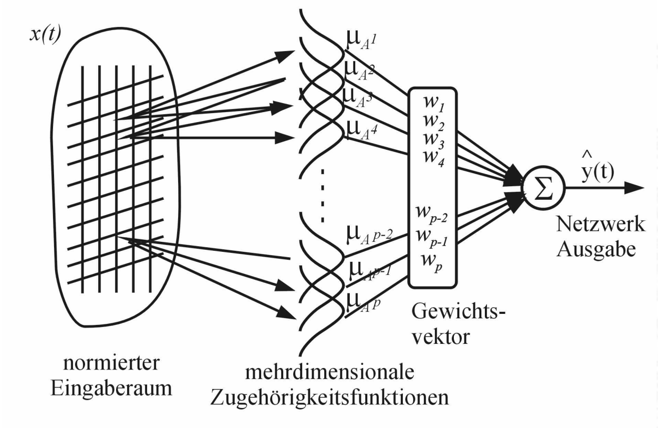
B-Spline ANFIS (cont.)

Then the output y of the MISO control system is:

$$y = \sum_{i_1=1}^{m_1} \dots \sum_{i_n=1}^{m_n} (Y_{i_1,\dots,i_n} \prod_{j=1}^{n} N^{j}_{i_j,k_j}(x_j))$$

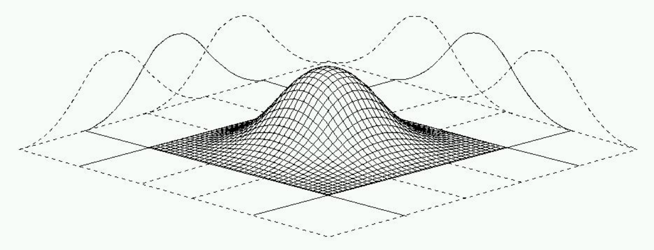
This is a general B-spline model that represents the hyperplane (it NUBS (nonuniform B-spline)).

Architecture of B-Spline ANFIS

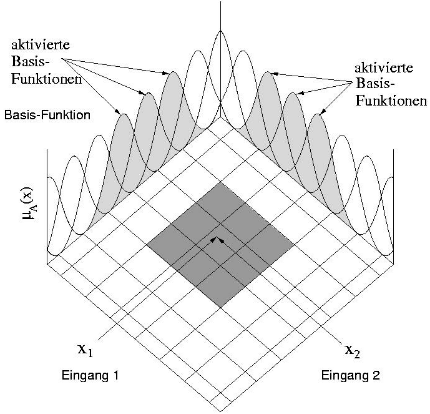


MF(membership function)-Formulation - Tensor

Tensor of 2D-Splines:



The activation of MF by the inputs



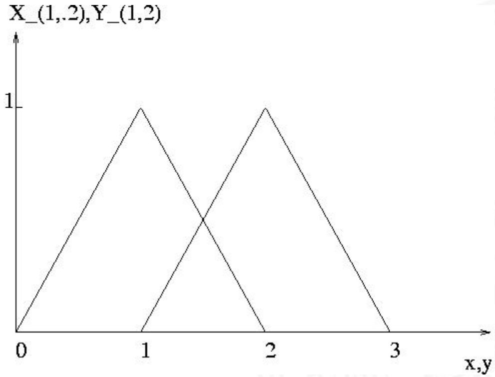
B-Spline ANFIS: example

An example with two input variables (x und y) and one Output z.

The parameters of the THEN-clauses are Z_1, Z_2, Z_3, Z_4.

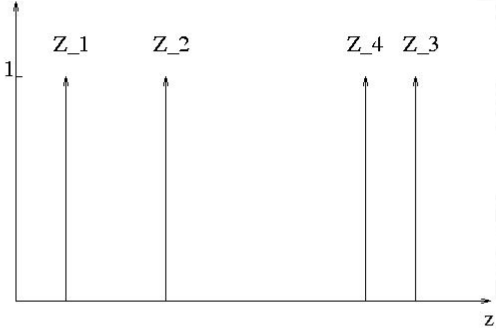
B-Spline ANFIS: example (cont.)

The linguistic terms of inputs (IF-clauses):



B-Spline ANFIS: example (cont.)

The parameters of the THEN-clauses:



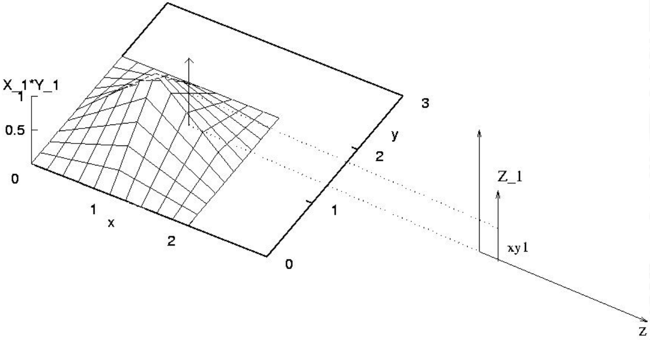
Example: control basis

The sample control basis consists of four rules:

Rule

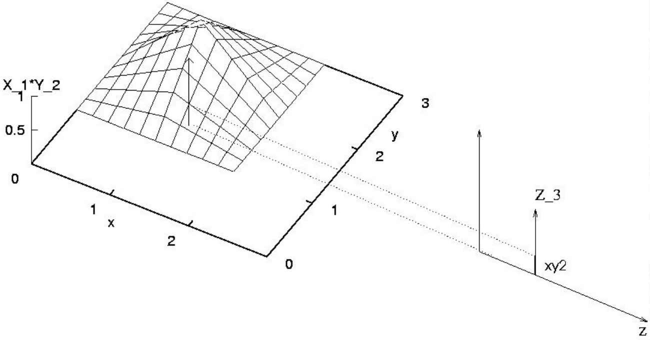
1) IF x is X_1 and y is Y_1 THEN z is Z_1
2) IF x is X_1 and y is Y_2 THEN z is Z_2
3) IF x is X_2 and y is Y_1 THEN z is Z_2
4) IF x is X_2 and y is Y_2 THEN z is Z_4

Illustration of the fuzzy inference



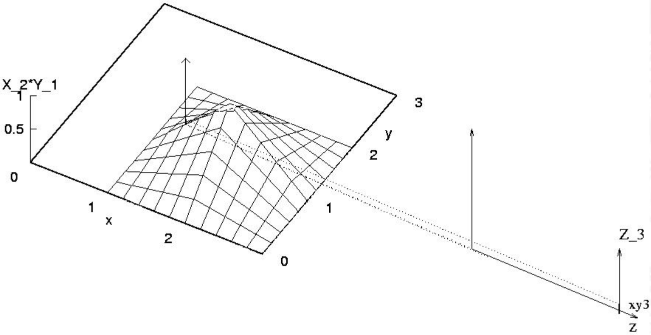
IF (x is X_1) and (y is Y_1) THEN z is Z_1

Illustration of the fuzzy inference (2)



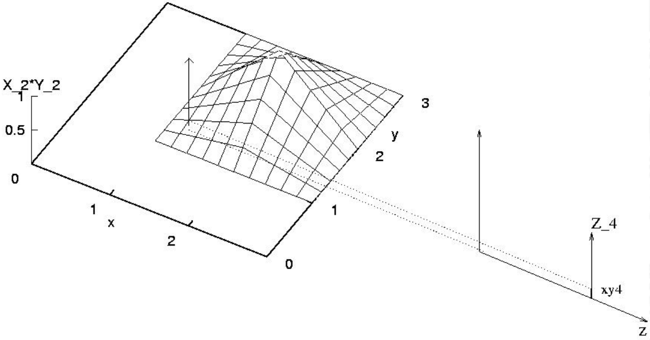
IF (x is X_1) and (y is Y_2) THEN z is Z_2

Illustration of the fuzzy inference (3)



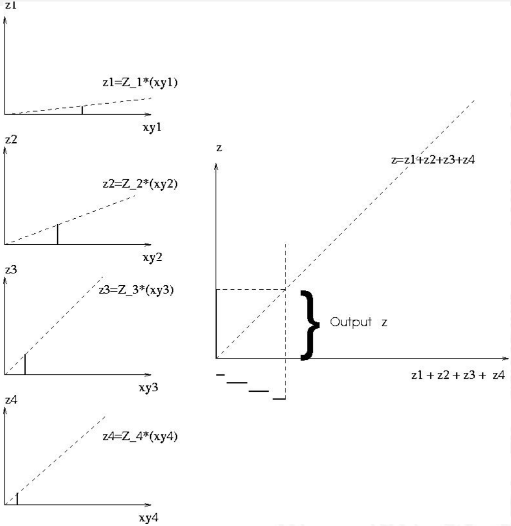
IF (x is X_2) and (y is Y_1) THEN z is Z_3

Illustration of the fuzzy inference (4)



IF (x is X_2) and (y is Y_2) THEN z is Z_4

Illustration of the fuzzy inference (5)



Algorithms for Supervised Learning - I

Let $\{(\mathbf{X}, y_d)\}$ be a set of training data, where

- $\mathbf{X} = (x_1, x_2, \ldots, x_n)$: the vector of input data,
- y_d : the desired output for $\mathbf{X}$.

The LSE is:

$$E = \frac{1}{2}(y_r - y_d)^2, \qquad (2)$$

where y_r is the current real output value during the training cycle. Goal is to find the parameters $Y_{i_1, i_2, \ldots, i_n}$, that minimize the error in (2)

$$E = \frac{1}{2}(y_r - y_d)^2 \equiv \text{ MIN.} \qquad (3)$$

Algorithms for Supervised Learning - II

Each control point $Y_{i_1,\ldots,i_n}$ can be improved with the following gradient descend algorithm:

$$\begin{aligned} \Delta Y_{i_1,\ldots,i_n} &= -\epsilon \frac{\partial E}{\partial Y_{i_1,\ldots,i_n}} && (4) \\ &= \epsilon(y_r - y_d) \prod_{j=1}^{n} N^j_{i_j,k_j}(x_j) && (5) \end{aligned}$$

where $0 < \epsilon \leq 1$.

Algorithms for Supervised Learning - III

The gradient descend algorithm ensures that the learning algorithm converges to the global minimum of the LSE-function, because the second partial derivative of $Y_{(}i_1, l_2,\ hdots, i_n)$ is constant:

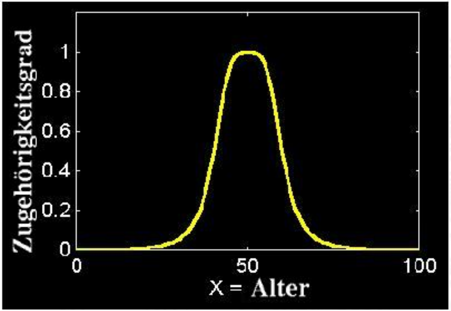
$$\frac{\partial^2 E}{\partial^2 Y_{i_1,\dots,i_n}} = (\prod_{j=1}^{n} N^{j}_{i_j,k_j}(x_j))^2 \geq 0. \tag{6}$$

This means that the LSE-function (ref (error)) is convex $Y_{(}i_1, l_2,\ dots, i_n$ is) and therefore has only one (global) minimum.

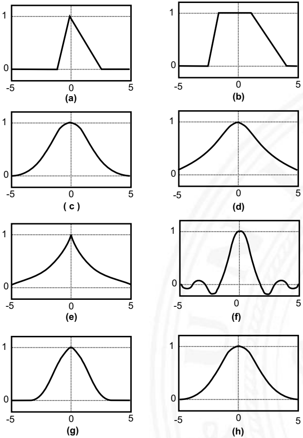
Symbol Transformation of the Core Functions

Positive, convex core functions can be considered as Fuzzy sets, for example:

$$\mu_B(x) = \frac{1}{1 + \left(\frac{x-50}{10}\right)^2}$$



Membership-functions



(a) (b) (c) (d) (e) (f) (g) (h)

Introduction to fuzzy sets

- fuzzy natural-language gradations of terms like “big”, “beautiful”, “strong” ...
- human thought and behavior models using the one-step logic:

Driving: “IF-THEN”-clauses

Car parking: With millimeter accuracy?

Introduction to Fuzzy sets

- Use of fuzzy language instead of numerical description:

  brake 2.52 m before the curve

  → only in machine systems

  brake shortly before the curve

  → in natural language

Definitions

Fuzzy: indistinctive, vague, unclear.

Fuzzy sets / fuzzy logic as a mechanism for

- fuzzy natural-language gradations of terms like “big”, “beautiful”, “strong” ...
- usage of fuzzy language instead of numerical description:.
- abstraction of unnecessary / too complex details.
- human thought and behavior models using the one-step logic.

Characteristic function vs. Membership function

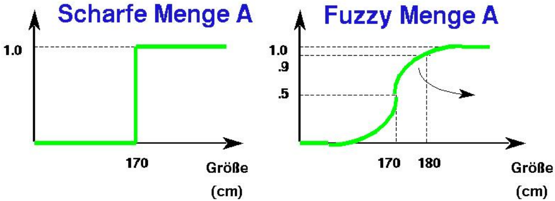
For **Fuzzy-sets** A we used a generalized characteristic function μ_A that assigns a real number from $[0, 1]$ to to each member $x \in X$ — the "degree" of membership of x to the fuzzy set A:

$$\mu_A : X \to [0, 1]$$

μ_A is called membershop-function.

$$A = \{(x, \mu_A(x)|x \in X\}$$

Membership function

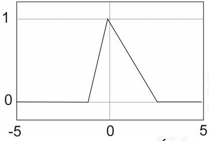


Characteristic of the continuous membership function

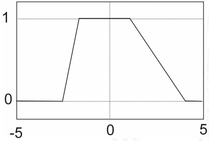
- Positive, convex functions (some important core functions).
- Subjective perception
- no probabilistic functions

Membership function types - I

Triangle: $trimf(x; a, b, c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right)$

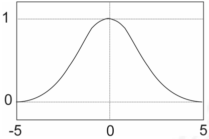


Trapeze: $trapmf(x; a, b, c, d) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right)$

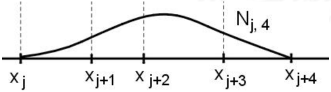


Membership function types - II

Gaussian: $gaussmf(x; c, \sigma) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2}$



B-Splines: $bsplinemf(x, x_i, x_{i+1}, \cdots, x_{i+k})$



Linguistic variables

A numeric variable has numerical values:

$$age = 25$$

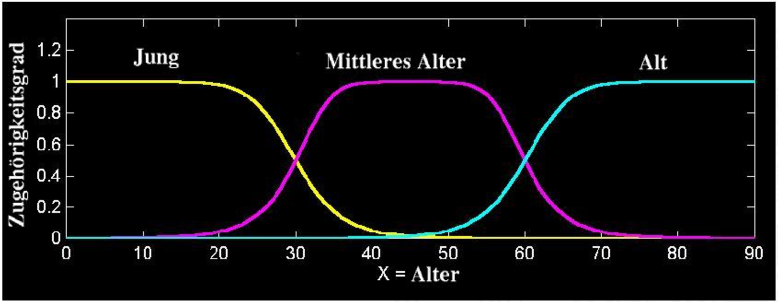
A linguistic variable has linguistic values (terms):

$$age : young$$

A linguistic value is a fuzzy set.

Fuzzy-Partition

Fuzzy partition of the linguistic values “young”, “average” and “old”:



Fuzzy Logic: inference mechanisms

A fuzzy rule is formulated as follows:

"IF A THEN B"

with Fuzzy-sets A, B and the universes X, Y.

One of the most important inference mechanisms is the generalized Modus-Ponens (GMP):

$$\begin{array}{ll} \text{Implication:} & \text{IF } x \text{ is } A \text{ THEN } y \text{ is B} \\ \text{Premise:} & x \text{ is } A' \\ \hline \text{Conclusion:} & y \text{ is } B' \end{array}$$

Fuzzy systems for function approximation

Basic idea:

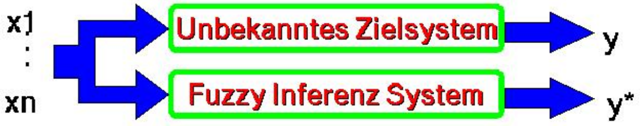
- Description of the desired control behavior through natural language, qualitative rules.
- Quantification of linguistic values by fuzzy sets.
- Evaluation by methods of fuzzy logic or interpolation.

Fuzzy systems for function approximation

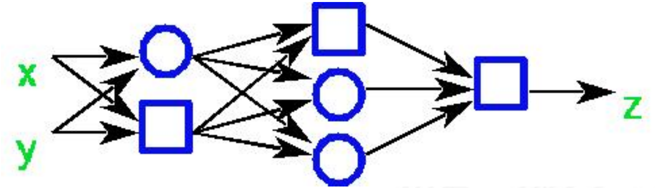
Fuzzy-rules:
„**IF** (a set of conditions is met)
THEN
(a set of consequences can be determined)“

In the premises (Antecedents) of the IF-part: linguistic variables from the domain of process states;

In the conclusions (Consequences) of the THEN-part: linguistic variables from the system domain.



Adaptive networks

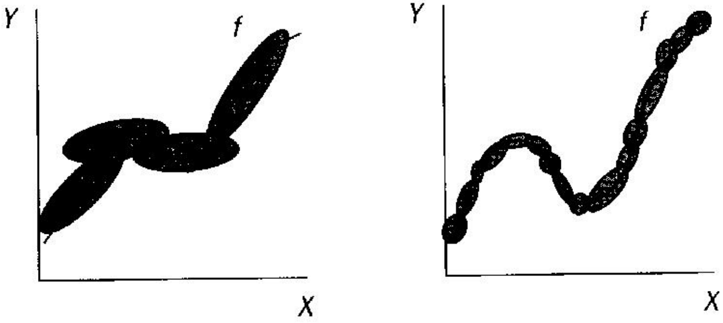


Architecture:

Feedforward networks with different node functions

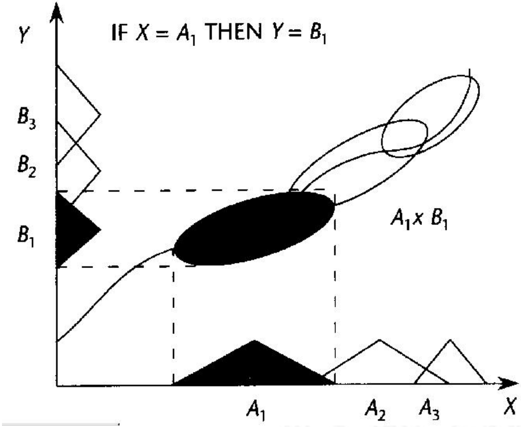
Rule Extraction

The Fuzzy-Patches (Kosko):



Rule Extraction

A Fuzzy-Rule-Patch:



Additive Systeme

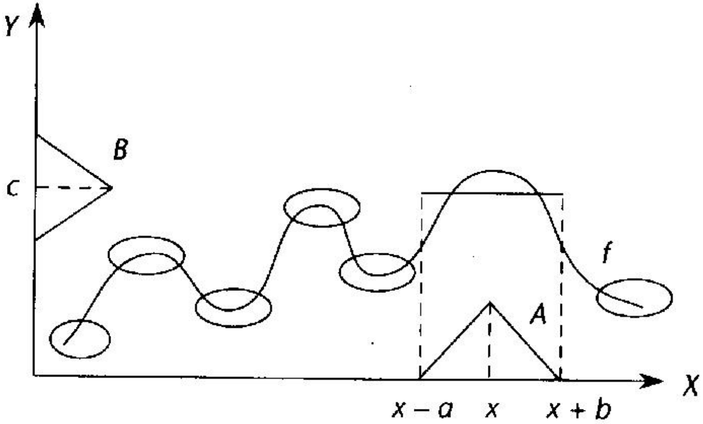
An additive fuzzy controller adds the “THEN”-Parts of the fired rules.

Fuzzy-Approximations-Rule:

An additive Fuzzy controller can approximate any continuous function $f : X \to Y$ if X is compact

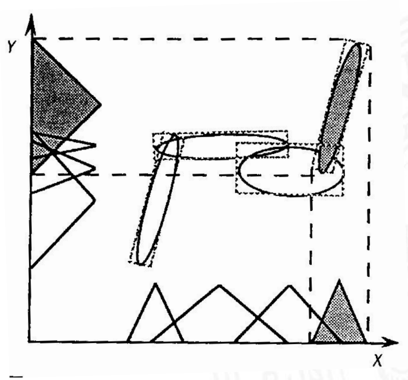
Optimal Fuzzy-Rule-Patches

Optimal fuzzy rule patches cover the extrema of a function:



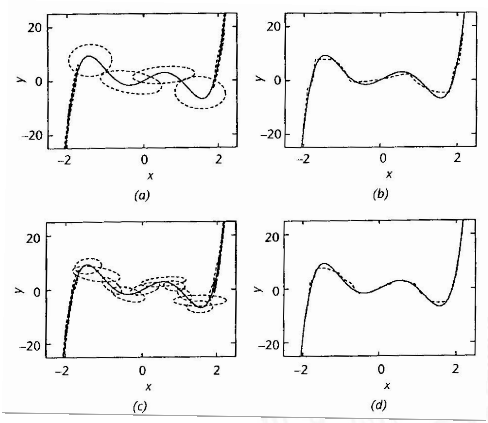
Optimal Fuzzy-Rule-Patches

Projection of the ellipsoids on the input and output axis:



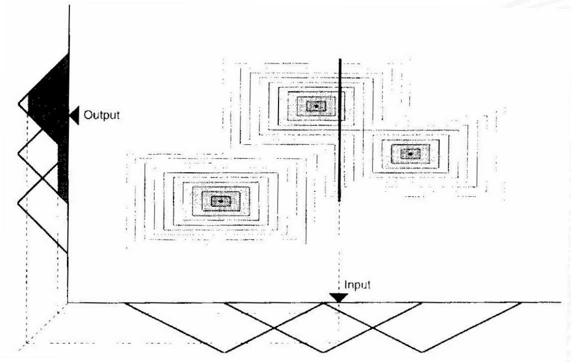
Optimal Fuzzy-Rule-Patches

The size of an ellipsoid depends on the training data.



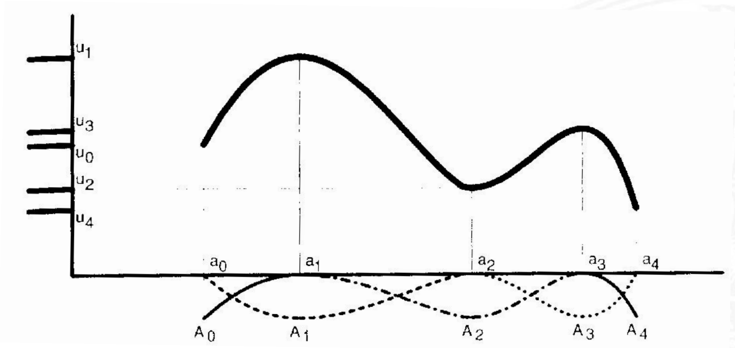
Optimal Fuzzy-Rule-Patches

Visualization of the input-output space:



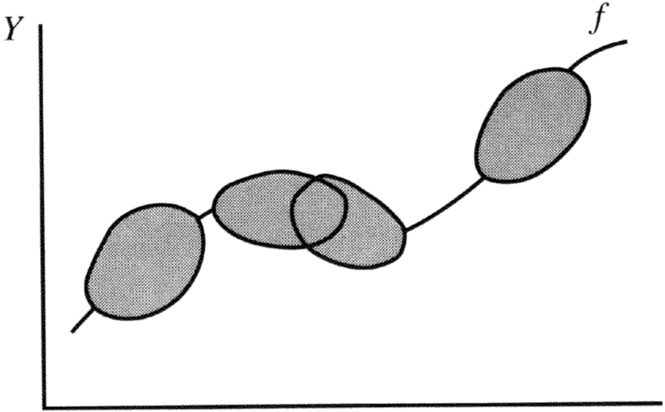
Optimal Fuzzy-Rule-Patches

An example for interpolation:

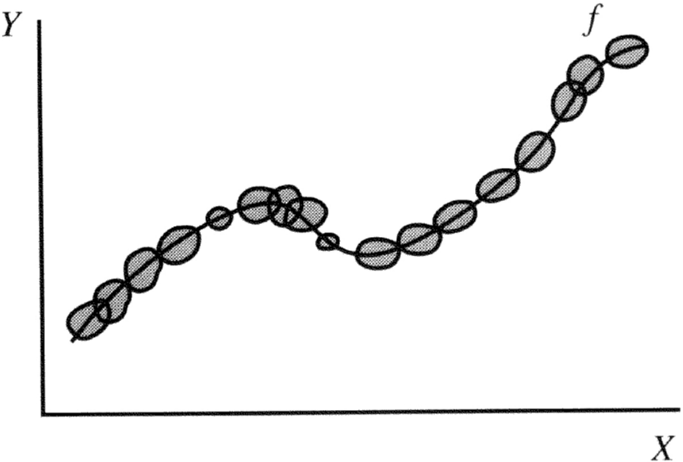


Optimal Fuzzy-Rule-Patches

Data cluster along the function:

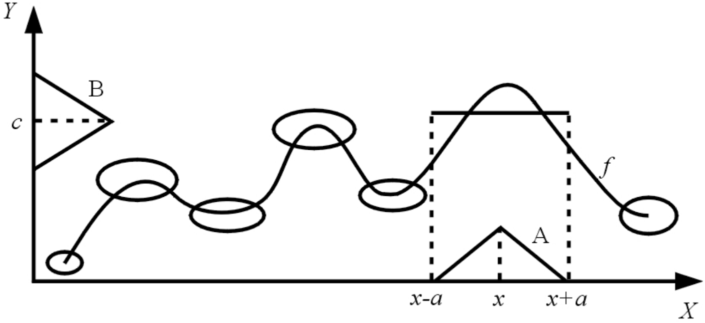


Optimal Fuzzy-Rule-Patches



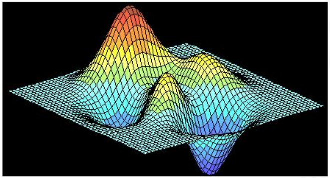
Optimal Fuzzy-Rule-Patches

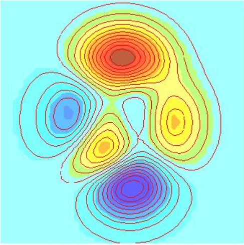
Approximation with Fuzzy-sets using the projections of extremes:



Approximation of a 2D-function

$$
\begin{aligned}
z =& f(x,y)\\
=& 3(1-x)^2 e^{-x^2-(y+1)^2} - 10(\frac{x}{5} - x^3 - y^5)e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2}
\end{aligned}
$$





Approximation of a 2D-function

Derivatives of the function:

$$\begin{aligned}\frac{dz}{dx} =& -6(1-x)e^{-x^2-(y+1)^2} - 6(1-x)^2xe^{-x^2-(y+1)^2}\\ &-10(\frac{1}{5}-3x^2)*e^{-x^2-y^2}+20(\frac{1}{5}x-x^3-y^5)xe^{-x^2-y^2}\\ &-\frac{1}{3}(-2x-2)e^{-(x+1)^2-y^2}\end{aligned}$$

$$\begin{aligned}\frac{dz}{dy} =& 3(1-x)^2(-2y-2)e^{-x^2-(y+1)^2}\\ &+50y^4e^{-x^2-y^2}+20(\frac{1}{5}x-x^3-y^5)ye^{-x^2-y^2}\\ &+\frac{2}{3}ye^{-(x+1)^2-y^2}\end{aligned}$$

Approximation of a 2D-function

$$\begin{aligned}\frac{d\frac{dz}{dx}}{dx} =& 36xe^{-x^2-(y+1)^2} - 18x^2e^{-x^2-(y+1)^2} - 24x^3e^{-x^2-(y+1)^2}\\ &+ 12x^4e^{-x^2-(y+1)^2} + 72xe^{-x^2-y^2} - 148x^3e^{-x^2-y^2}\\ &- 20y^5e^{-x^2-y^2} + 40x^5e^{-x^2-y^2} + 40x^2e^{-x^2-y^2}y^5\\ &- \frac{2}{3}e^{-(x+1)^2-y^2} - \frac{4}{3}e^{-(x+1)^2-y^2}x^2 - \frac{8}{3}e^{-(x+1)^2-y^2}x\end{aligned}$$

Approximation of a 2D-function

$$
\begin{aligned}
\frac{d(\frac{dz}{dy})}{dy} = & -6(1-x)^2 e^{-x^2-y(+1)^2} + 3(1-x)^2(-2y-2)^2 e^{-x^2-(y+1)^2} \\
& + 200y^3 e^{-x^2-y^2} - 200y^5 e^{-x^2-y^2} + 20(\frac{1}{5}x - x^3 - y^5)e^{-x^2-y^2} \\
& - 40(\frac{1}{5}x - x^3 - y^5)y^2 e^{-x^2-y^2} + \frac{2}{3}e^{-(x+1)^2-y^2} \\
& - \frac{4}{3}y^2 e^{-(x+1)^2-y^2}
\end{aligned}
$$

Global Overview of the Statistical Learning Theory - I

Let $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ be a set of data points/examples. We are searching for a function f , which minimizes the following equation:

$$H[f] = \frac{1}{l}\sum_{i=1}^{l} V(y_i, f(\mathbf{x}_i)) + \lambda \|f\|_K^2$$

where $V(\cdot,\cdot)$ is a *loss function* and $\|f\|_K^2$ is a norm in the Hilbert space $\mathcal{H}$, which is defined by a positive kernel K , and λ is the regularization parameter.

The problems in modeling, data regression and pattern classification are each based on a kind of $V(\cdot,\cdot)$.

Global Overview of the Statistical Learning Theory - II

1. $V(y_i, f(\mathbf{x}_i)) = (y_i - f(\mathbf{x}_i))^2$
   ($y_i \in R^1$, V: square error function)
   $\Rightarrow$ *Regularization Networks*, RN
2. $V(y_i, f(\mathbf{x}_i)) = |y_i - f(\mathbf{x}_i)|_\epsilon$
   ($y_i \in R^1$, $|\cdot|_\epsilon$: eine ϵ-unempfindliche Norm)
   $\Rightarrow$ *Support Vector Machines Regression*, SVMR
3. $V(y_i, f(\mathbf{x}_i)) = |1 - y_i f(\mathbf{x}_i)|_+$
   ($y_i \in \{-1, 1\}$, $|x|_+ = x$ für $x \geqq 0$, else $|x|_+ = 0$)
   $\Rightarrow$ *Support Vector Machines Classification*, SVMC

Global Overview of the Statistical Learning Theory - II

1. $V(y_i, f(\mathbf{x}_i)) = (y_i - f(\mathbf{x}_i))^2$
   (y_i: a real number, V: square error function)
   ⇒ *Regularization Networks*, RN
2. $V(y_i, f(\mathbf{x}_i)) = |y_i - f(\mathbf{x}_i)|_\epsilon$
   (y_i: a real number, $|\cdot|_\epsilon$: an ϵ-independent norm)
   ⇒ *Support Vector Machines Regression*, SVMR
3. $V(y_i, f(\mathbf{x}_i)) = |1 - y_i f(\mathbf{x}_i)|_+$
   (y_i: -1 oder 1, $|x|_+ = x$ für $x \geqq 0$, sonst $|x|_+ = 0$)
   ⇒ *Support Vector Machines Classification*, SVMC

For modeling and control tasks, the first definition is most important.

Universelal Function Approximation - I

A control-network can approximate all smooth functions with an arbitrary precision.
The general solution of this problem is:

$$f(x) = \sum_{i=1}^{l} c_i K(x; x_i)$$

where c_i are the coefficients.

Universelal Function Approximation - II

Proposition of the approximation:

For any continuous function Y that is defined on the compact subset R^n and the core function K, there is a function $y^*(x) = \sum_{i=1}^{l} c_i K(x; x_i)$ that fulfills for all x and any ϵ:

$$|Y(x) - y^*(x)| < \epsilon$$

Universal Function Approximation - II

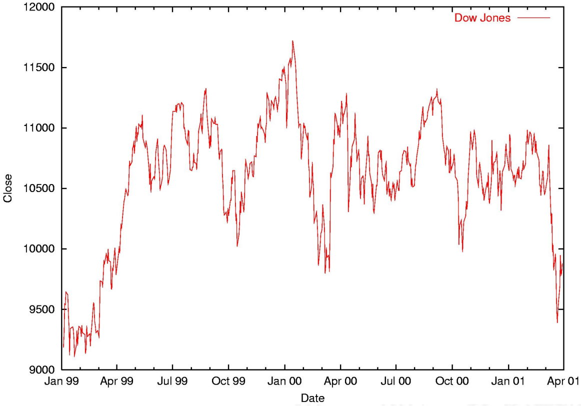
Using different kernel functions leads to different models:

$K(\mathbf{x}-\mathbf{x}_i) = \exp(-\|\mathbf{x}-\mathbf{x}_i\|^2)$	Gaussian RBF
$K(\mathbf{x}-\mathbf{x}_i) = (\|\mathbf{x}-\mathbf{x}_i\|^2 + c^2)^{-\frac{1}{2}}$	Inverse multiquadratic functions
$K(\mathbf{x}-\mathbf{x}_i) = \tanh(\mathbf{x}\cdot\mathbf{x}_i - \theta)$	Multilayer perceptron
$K(\mathbf{x}-\mathbf{x}_i) = (1 + \mathbf{x}\cdot\mathbf{x}_i)^d$	Polynomial of degree d
$K(x-x_i) = B_{2n+1}(x-x_i)$	B-Splines
$K(x-x_i) = \dfrac{\sin(d+\frac{1}{2})(x-x_i)}{\sin\frac{(x-x_i)}{2}}$	Trigonometric polynomial
...	...

Problems

1. "Curse of dimensionality" because of the exponential dependency between the memory requirements and the dimension of input space.
2. *Aliasing* within the feature extraction
3. Not available target data (y).
4. Not available input factors.

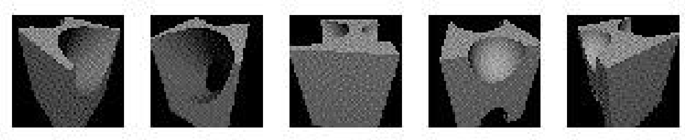
Learning from DJ-Data



Dow-Jones-Index: can the function be modeled?

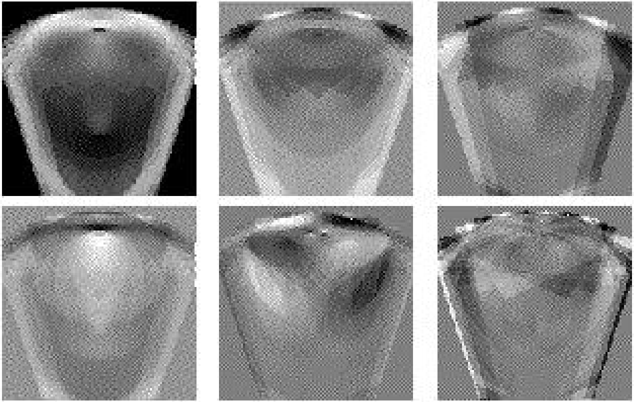
Example: Image Processing in Local Observation scenarios

A sequence of gray scale images of an object is acquired by the movement along a fixed location:

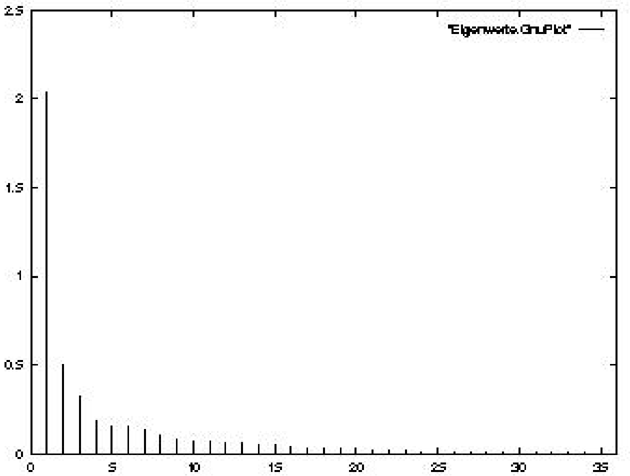


Example: Extraction of Eigenvectors

The first 6 Eigenvectors:



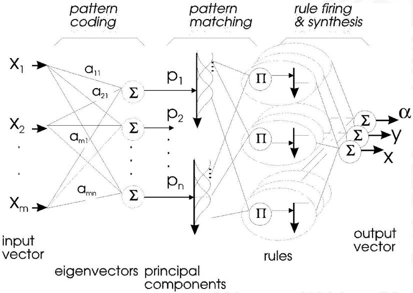
Example: Eigenvectors and Eigenvalues



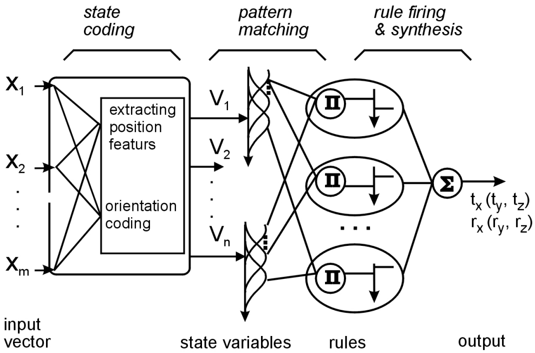
Combination of Dimension Reduction with B-Spline Model

Eigenvectors can be partitioned by linguistic terms.

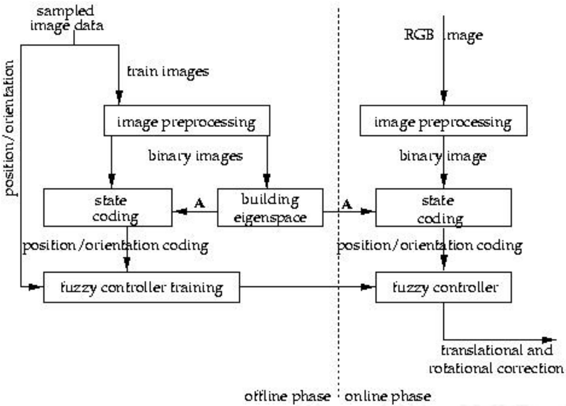
Such a combination of PCA and B-spline model can be considered as a Neuro-Fuzzy model.



The Neuro-Fuzzy Model



The Training and Application Phases


sampled image data
train images
position/orientation
image preprocessing
binary images
state coding
A
building eigenspace
A
position/orientation coding
fuzzy controller training
RGB image
image preprocessing
binary image
state coding
position/orientation coding
fuzzy controller
translational and rotational correction
offline phase
online phase