Lecture 10:

Nonlinear regression models

Outline

- Motivation
- Correlated vs. uncorrelated variables
- Correlation coefficient
- Linear regression
- Nonlinear models (regression trees, RBF networks)

Motivation

Problem: Let us suppose that we know some information about a car (e.g. cylinders, horsepower, weight, acceleration, model etc) and we would like to estimate the fuel consumption (e.g. expressed as miles per gallon)

Example [autoMpg.arff from http://archive.ics.uci.edu/ml/datasets.html]

```
@relation autoMpg
@attribute cylinders { 8, 4, 6, 3, 5} @attribute displacement real
@attribute horsepower real @attribute weight real @attribute acceleration real
@attribute model { 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82}
@attribute origin { 1, 3, 2}
@attribute class real
@data
8,307,130,3504,12,70,1,18
8,350,165,3693,11.5,70,1,15
4,113,95,2372,15,70,3,24
6,198,95,2833,15.5,70,1,22
6,199,97,2774,15.5,70,1,18
```

Motivation

Problem: Let us suppose that we know some information about a car (e.g. cylinders, horsepower, weight, acceleration, model etc) and we would like to estimate the fuel consumption (e.g. expressed as miles per gallon)

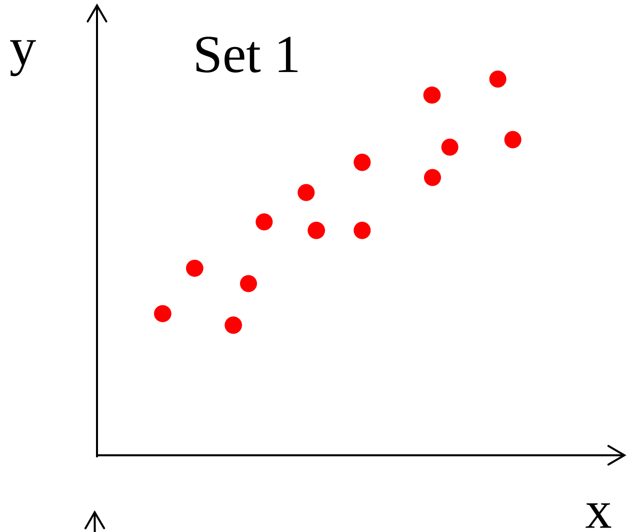
Example [autoMpg.arff from http://archive.ics.uci.edu/ml/datasets.html]

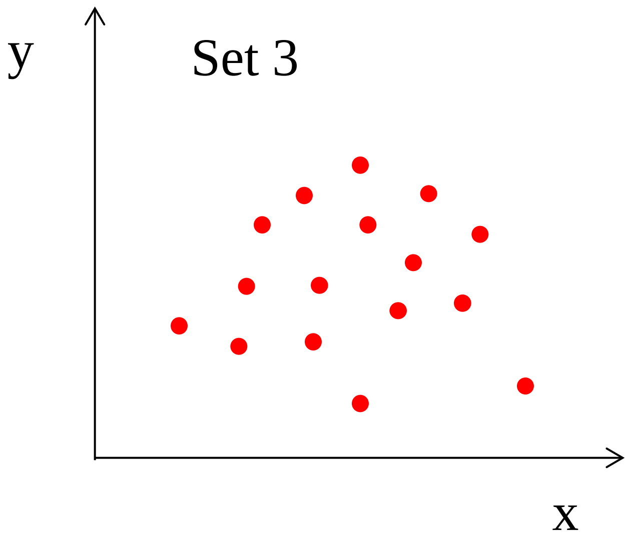
```
@relation autoMpg
@attribute cylinders { 8, 4, 6, 3, 5} @attribute displacement real
@attribute horsepower real @attribute weight real @attribute acceleration real
@attribute model { 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82}
@attribute origin { 1, 3, 2}
@attribute class real
@data
8,307,130,3504,12,70,1,18
8,350,165,3693,11.5,70,1,15
4,113,95,2372,15,70,3,24
6,198,95,2833,15.5,70,1,22
6,199,97,2774,15.5,70,1,18
```

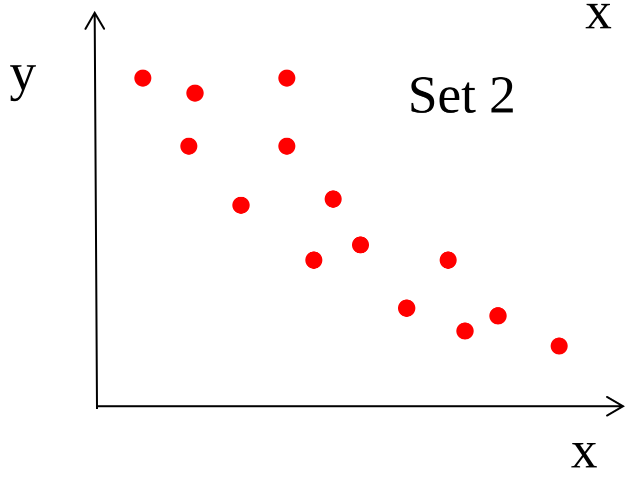
We are looking for a dependence between the fuel consumption (class attribute in the dataset) the car characteristics (first 7 attributes in the dataset)

A simpler example

Some synthetic 2D data



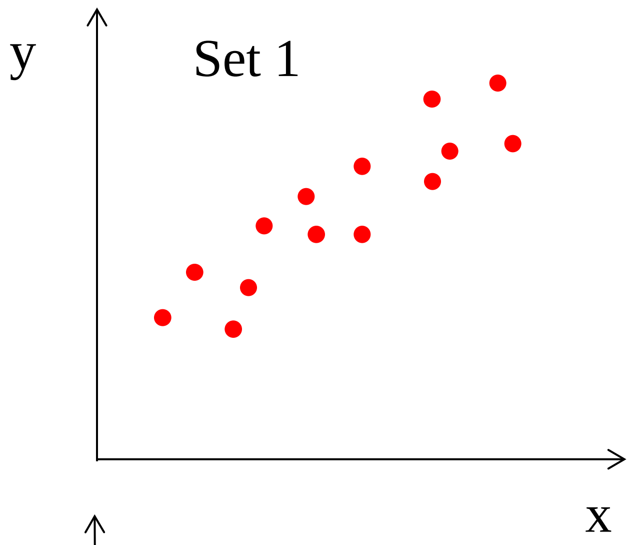


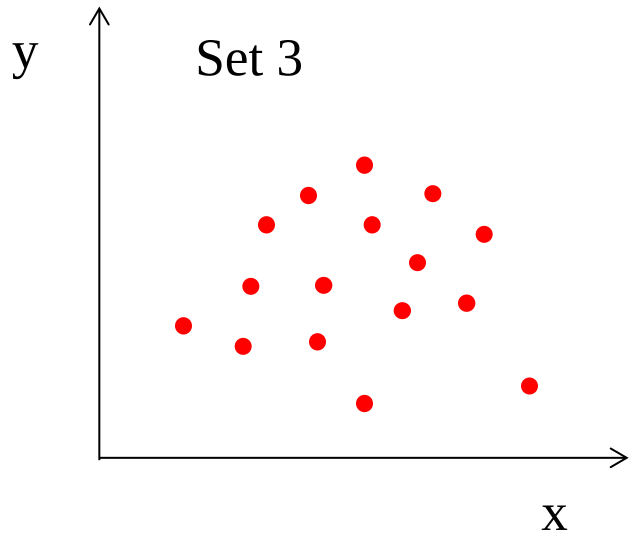


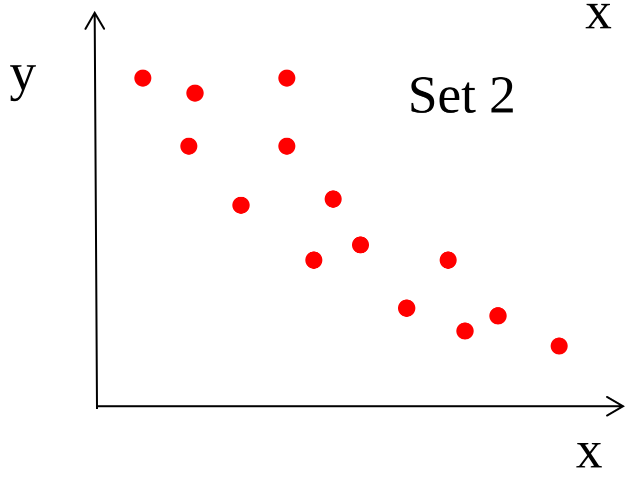
What can we say about the data in each set?

A simpler example

Some synthetic 2D data



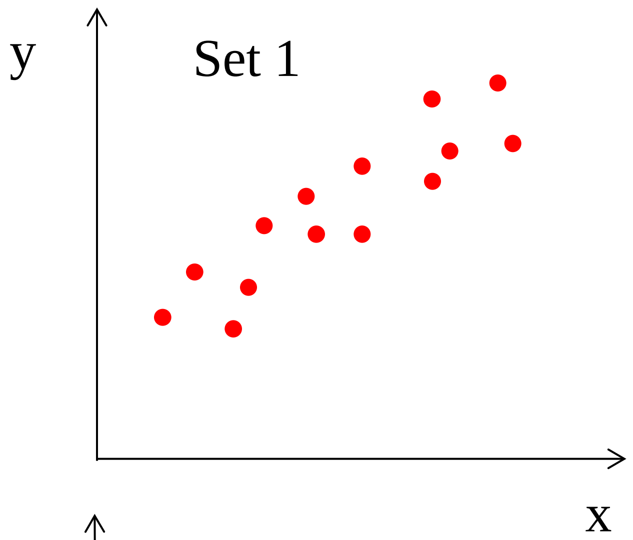


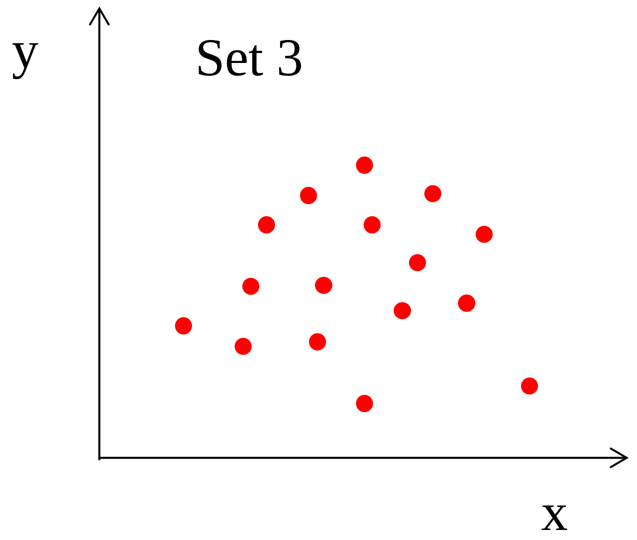


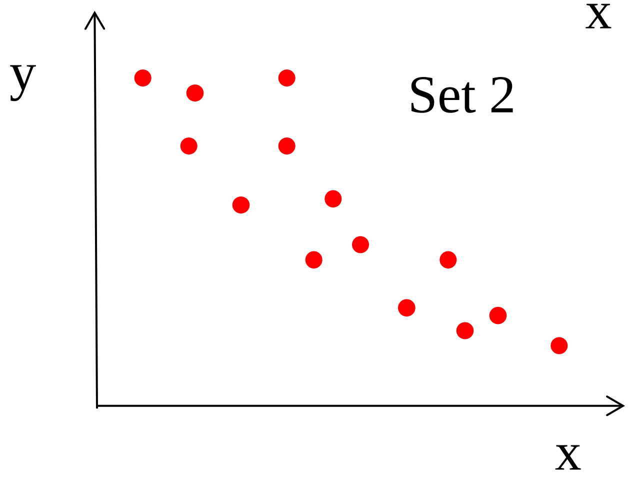
Set 1: the data seem to be “positively correlated” = when x increases y also increases

A simpler example

Some synthetic 2D data



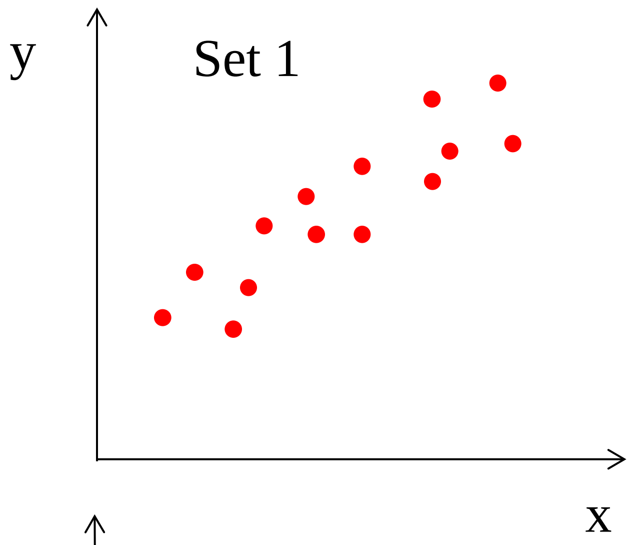


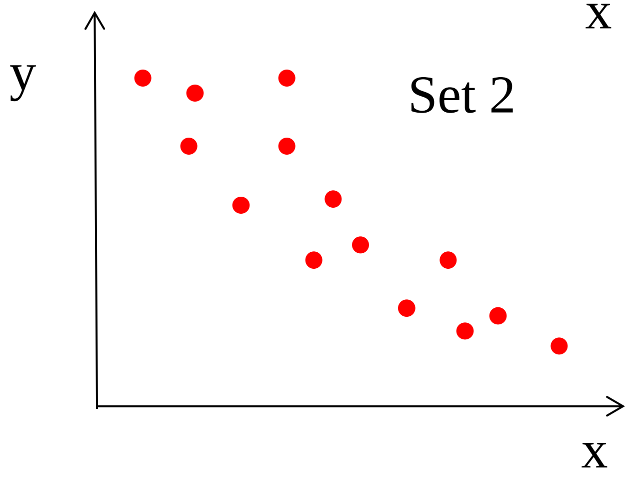


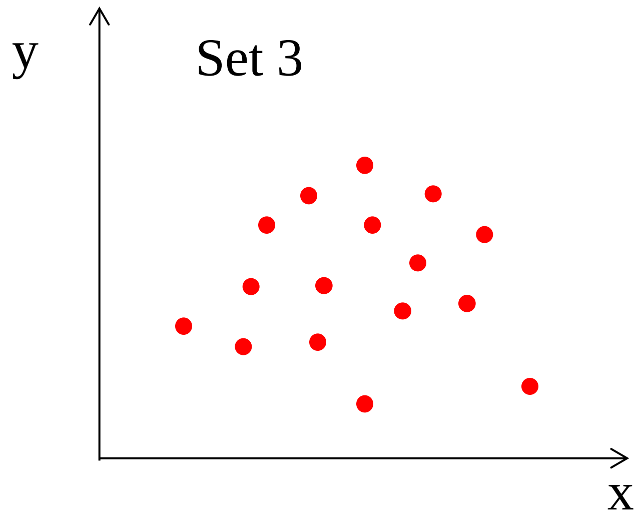
Set 2: the data seem to be “negatively correlated” = when x increases y decreases

A simpler example

Some synthetic 2D data







Set 3: the data does not seem to be correlated (it seems to be just a cloud of points)

Questions:

- How can be measured the degree of correlation?
- What kind of correlation?

Correlation coefficient

How can be measured the degree of correlation?

[reminder – Probability and Statistics]

- For instance, by using the Pearson correlation coefficient – it expresses the degree of linear correlation between two variables

$$R(X,Y) = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - avg(X))(y_i - avg(Y))}{stdev(X)stdev(Y)}$$

$$stdev(X) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - avg(X))^2}$$

$$stdev(Y) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - avg(Y))^2}$$

$$avg(X) = \frac{1}{n}\sum_{i=1}^{n}x_i,\ avg(Y) = \frac{1}{n}\sum_{i=1}^{n}y_i$$

Remark: -1<=R(X,Y)<=1

- R(X,Y) close to 1: positive linear correlation
- R(X,Y) close to -1: negative linear correlation
- R(X,Y) close to 0: no linear correlation (however, X and Y could be nonlinearly correlated)

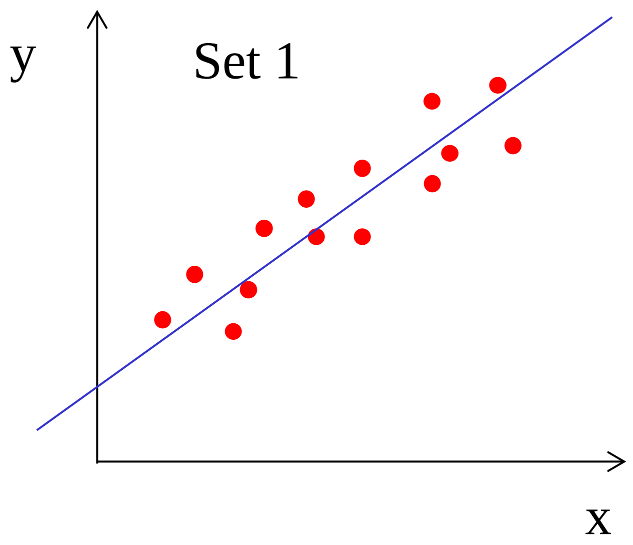
Linear regression

What kind of correlation? [reminder – Probability and Statistics]

Simplest case: Linear dependence between two variables: $Y=w_1X+w_0$

- X= predictor (independent, input, explanatory) variable
- Y= predicted (dependent, response, explained) variable

- Aim of linear regression: estimate the parameters w_1 and w_0 such that the available data for the variables X (i.e. $x_1,x_2,\ldots, x_n$) and Y (i.e. $y_1,y_2,\ldots, y_n$) are well explained by the linear function, i.e. the sum of squared errors is minimized



$$SSE(w_1, w_0) = \sum_{i=1}^{n} (y_i - (w_1 x_i + w_0))^2$$

$$= \sum_{i=1}^{n} (y_i - \overline{w}\,\overline{x_i})^2$$

$$(\overline{w} = (w_1, w_0),\ \overline{x_i} = (x_i, 1)^T)$$

row vector column vector

Simple linear regression

Reminder: some linear algebra

$$w = (w_1, w_0),\ D = \begin{pmatrix} x_1 & x_2 & \dots & x_n \\ 1 & 1 & \dots & 1 \end{pmatrix}^T,\ y = (y_1, y_2, \dots, y_n)^T$$

$$SSE(w) = \left\| y - Dw^T \right\|^2 = (y - Dw^T)^T (y - Dw^T)$$

$$= y^T y - 2wD^T y + wD^T D w^T$$

Finding the vector w which minimizes SSE(w) is equivalent with finding the critical point of SSE, i.e. solving the following equation with respect to w:

$$D^T D w^T = D^T y \Rightarrow w^T = (D^T D)^{-1} D^T y = D^+ y$$

$$D^+ = (D^T D)^{-1} D^T \text{ is called the pseudoinverse of } D$$

Multiple linear regression

Remark: the same approach can be extended in the case when there are d predicting variables (e.g. as in the autoMPG dataset)

$$w = (w_1, w_2, \ldots, w_d, w_0),\ D = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ x_{21} & x_{22} & \ldots & x_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ x_{d1} & x_{d2} & \ldots & x_{dn} \\ 1 & 1 & \ldots & 1 \end{pmatrix}^T,\ y = (y_1, y_2, \ldots, y_n)^T$$

$$SSE(w) = \left\| y - Dw^T \right\|^2 = (y - Dw^T)^T (y - Dw^T)$$

$$= y^T y - 2wD^T y + wD^T D w^T$$

$$D^T D w^T = D^T y \Rightarrow w^T = (D^T D)^{-1} D^T y$$

Linear regression - regularization

Remark: if the matrix D^TD is singular (the inverse cannot be computed) then the objective function (SSE) is modified by adding a so-called regularization term which will modify the matrix of the linear system in such a way that it becomes invertible).

Examples:

- Tikhonov regularization (ridge regression)

$$SSE'(w) = SSE(w) + \lambda \|w\|^2$$

$$w = (D^T D + \lambda I)^{-1} D^T y$$

$$I = (d+1) \times (d+1) \text{ identity matrix}$$

Remarks:

- the parameter of the regularization term (lambda) is usually chosen adaptively based on cross-validation
- the penalty term “discourages” the large values of the weights

Linear regression - regularization

Remark: if the matrix $D^T D$ is singular (the inverse cannot be computed) then the objective function (SSE) is modified by adding a so-called regularization term which will modify the matrix of the linear system in such a way that it becomes invertible).

Examples:

- Lasso regularization

$$SSE'(w) = SSE(w) + \lambda \sum_{i=1}^{d} |w_i|$$

$$\text{(no closed form solution for w)}$$

Remarks:

- In this case the optimization problem is solved by using numerical methods
- Is useful for high dimensional data with many irrelevant features (leading to sparse models)

Generalized linear models

Main idea: instead of $y_i=w_1x_i+w_0$ the output (y_i) is modelled through a random variable with a distribution having a mean $f(w_1x_i+w_0)$

Main elements of a GLM (generalized linear model):

- Mean function: f
- Link function: f^{-1}
- Probability distribution

Mean function	Link function	Distribution
f(u)=u	identity	normal
f(u)=-1/u	inverse	exponential, gamma
f(u)=exp(u)	Log	Poisson
f(u)=1/(1+exp(-u))	Logit	Bernoulli

Generalized linear models

Main idea: instead of $y_i=wx_i$ the output (y_i) is modelled through a random variable with a distribution having a mean $f(wx_i)$

Main elements of a GLM (generalized linear model):

- Mean function: f
- Link function: f^{-1}
- Probability distribution

Mean function	Link function	Distribution
f(u)=u	identity	normal
f(u)=-1/u	inverse	exponential, gamma
f(u)=exp(u)	Log	Poisson
f(u)=1/(1+exp(-u))	Logit	Bernoulli

least squares regression

Generalized linear models

Main idea: instead of $y_i = wx_i$ the output (y_i) is modelled through a random variable with a distribution having a mean $f(wx_i)$

Main elements of a GLM (generalized linear model):

- Mean function: f
- Link function: f^{-1}
- Probability distribution

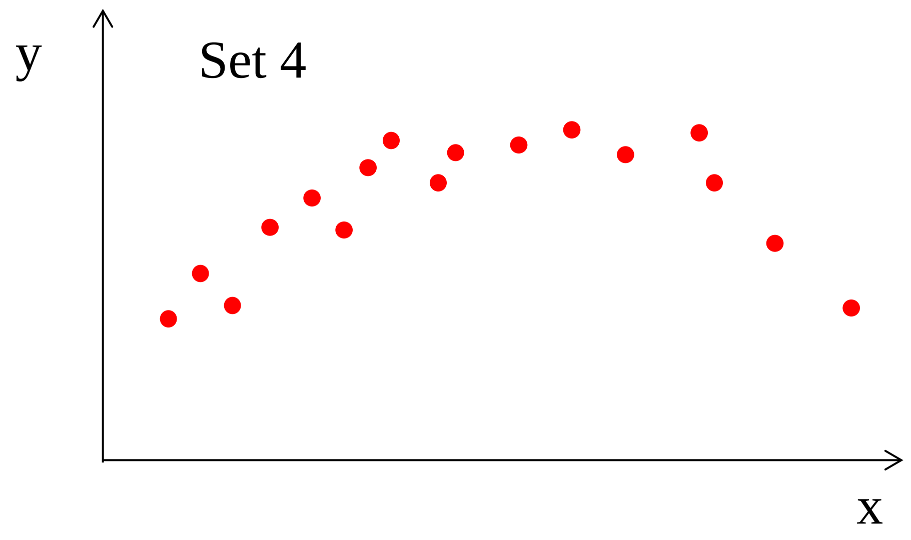
Mean function	Link function	Distribution
f(u)=u	identity	normal
f(u)=-1/u	inverse	exponential, gamma
f(u)=exp(u)	Log	Poisson
f(u)=1/(1+exp(-u))	Logit	Bernoulli

Logistic regression

Nonlinear regression

What about the cases when the dependence between the predicted variable and the predictor(s) is not linear?

Other models are needed



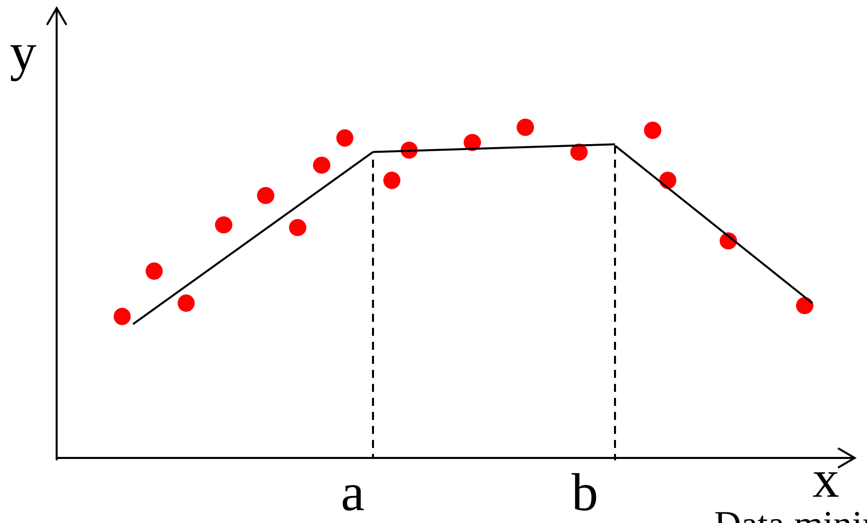
Examples:

- Regression trees
- Nonlinear neural networks

Nonlinear regression

Main idea:

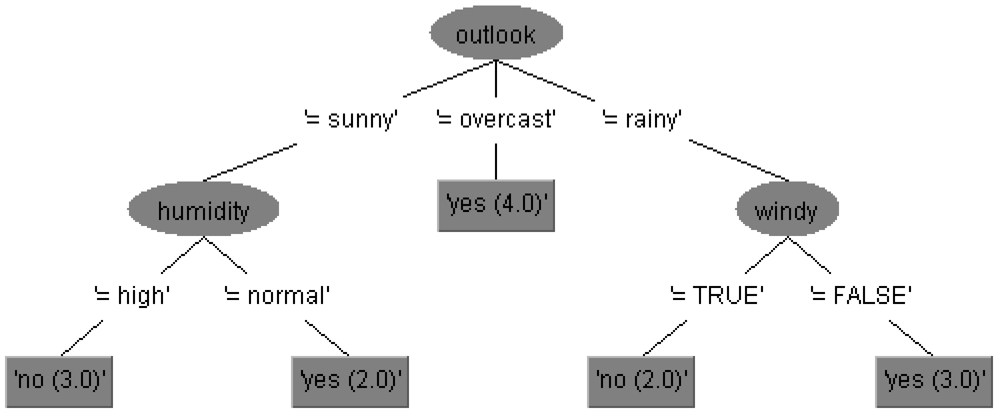
- A nonlinear relationship can be modelled through local linear functions (one linear function per region)
- The regression process would then consist of two steps:
    - Identify the regions by splitting the space of the decision variables
    - Identify a regression model (e.g. a linear one) for each of the identified regions



Regression trees

Reminder:

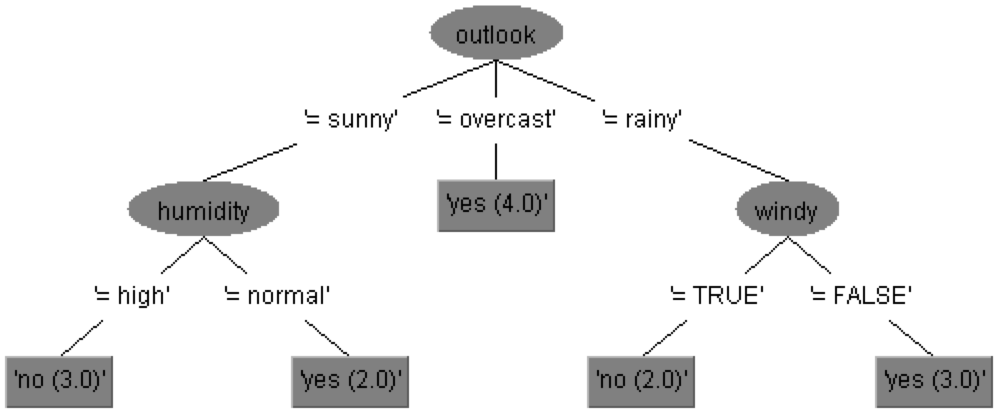
Decision tree = hierarchical structure containing in the internal nodes conditions on the predictor variables and on the leaf nodes information on the predicted variables (e.g. class); if the predicted variable is discrete (categorical/ nominal) then the decision tree is in fact a classification tree



Regression trees

Reminder:

Decision tree = hierarchical structure containing in the internal nodes conditions on the predictor variables and on the leaf nodes information on the predicted variables (e.g. class); if the predicted variable is discrete (categorical/ nominal) then the decision tree is in fact a classification tree



Question:

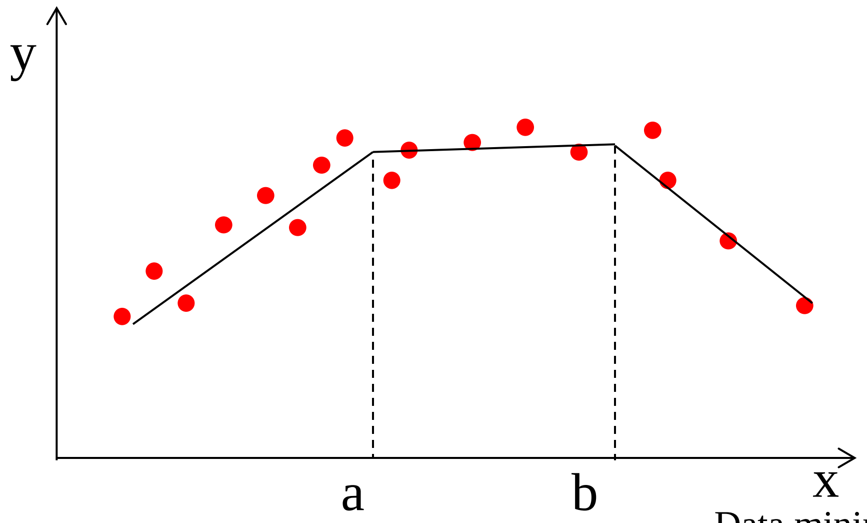
- What about the case when the predicting variable is continuous? (e.g. we would like to obtain not only a yes/no answer to the "weather-play" problem but a value in [0,1] expressing the degree of decision between 0 (no) and 1 (yes)

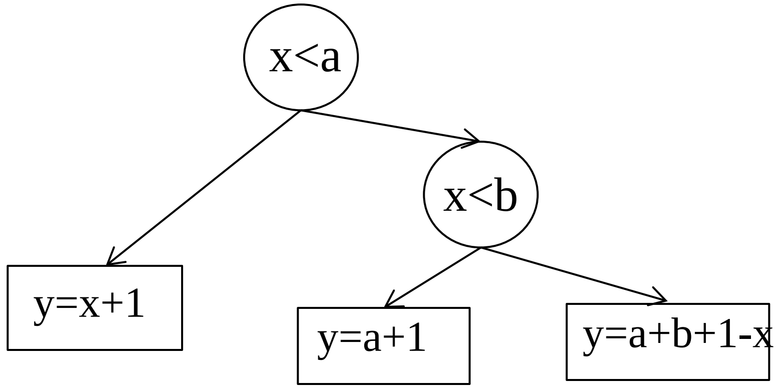
Regression trees

Main idea:

- Use a similar process of splitting the space of the decision (predictor) variables as in the case of trees used for classification
  - In the case of continuous predictor variables the splitting condition is of the one of the types: variable < value or variable > value or variable in [min,max]
- Infer a regression model (e.g. a linear model) for each region identified by the splitting procedure

(Very) simple example -> piecewise linear model:





Nonlinear regression

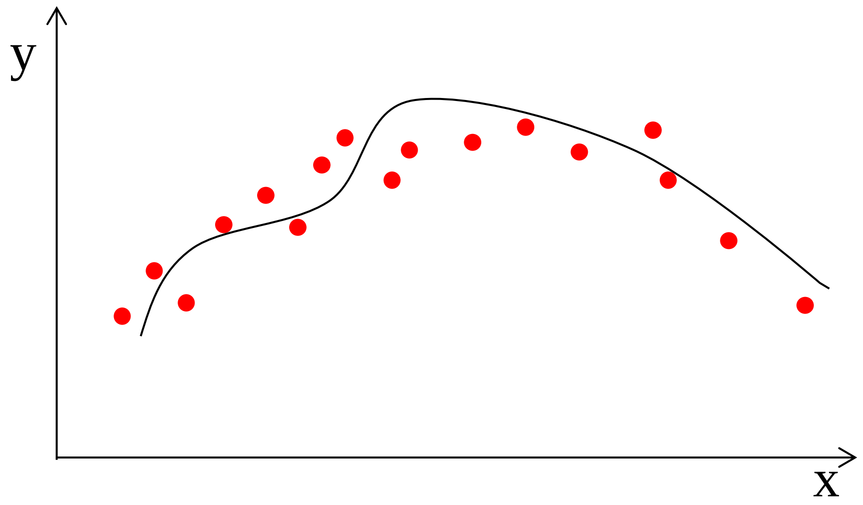
Beyond piecewise linear models:

- Extending basic linear regression by using derived input features:

$$y=w_0+w_1h_1(x)+w_2h_2(x)+\dots+w_mh_m(x)$$

(x can be a vector and h_i a function associating a scalar/vector to another vector)

Particular case 1. Polynomial models: $y= w_0+w_1x+w_2x^2+\dots+w_mx^m$
(x is a scalar)



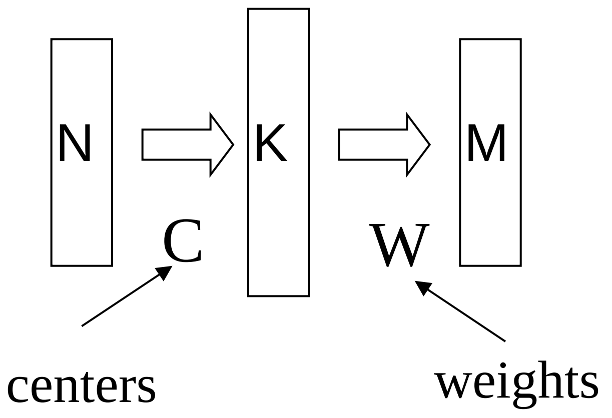
Particular case 2.

Kernel-based models: h_i are functions which can take significant values only for a limited region of the input space.

- when these functions are with radial symmetry (e.g. gaussian functions) then we obtain the so-called RBF networks (a particular case of neural networks)

RBF networks

RBF - “Radial Basis Function”:



Architecture:

– Two levels of functional units
– Aggregation functions:

- Hidden units: distance between the input vector and the corresponding center vector

$$G(X,C^k)=\left\|X-C^k\right\|=\sum_{i=1}^{N}(x_i-c_i^k)^2$$

- Output units: weighted sum

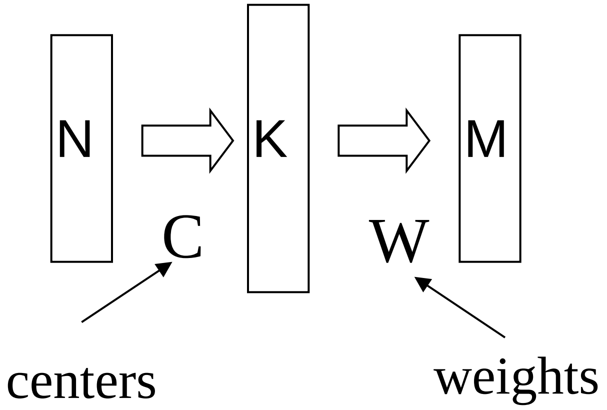
Rmk: hidden units do not have bias values (activation thresholds)

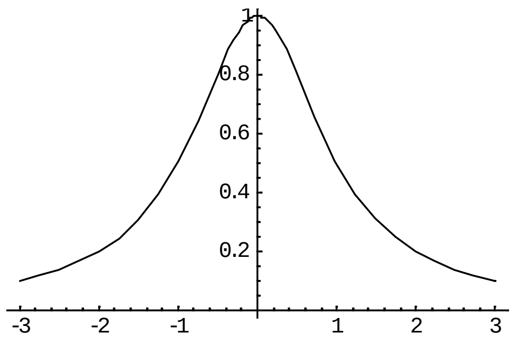
RBF networks

The activation functions for the hidden neurons are functions with radial symmetry

- Hidden units generates a significant output signal only for input vectors which are close enough to the corresponding center vector

The activation functions for the output units are usually linear functions





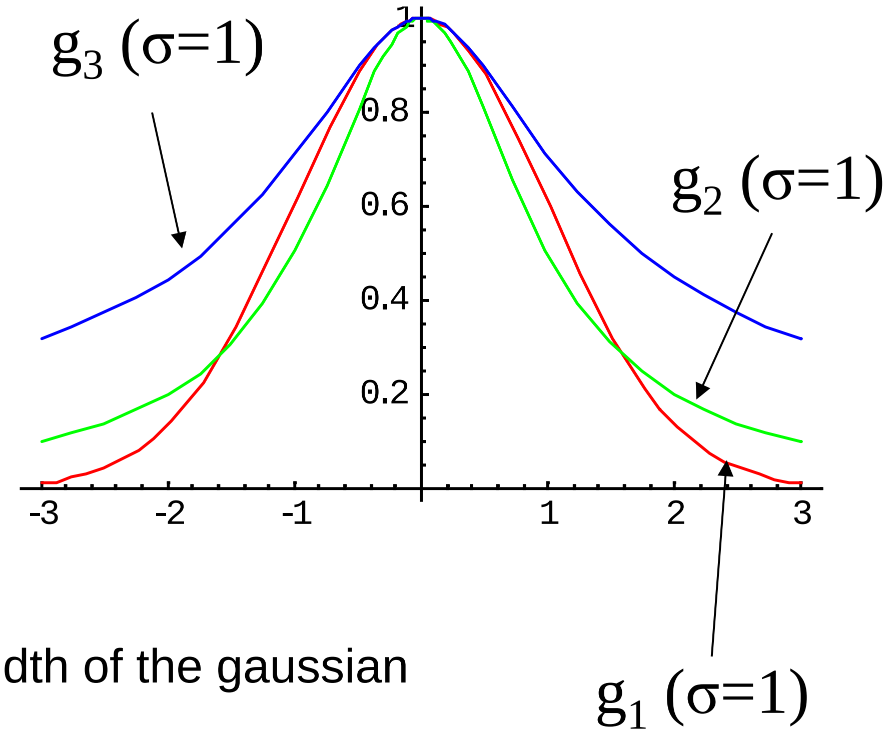
RBF networks

Examples of functions with radial symmetry:

$$g_1(u) = \exp\left(-\frac{u^2}{2\sigma^2}\right)$$

$$g_2(u) = 1/(u^2+\sigma^2)$$

$$g_3(u) = 1/\sqrt{u^2+\sigma^2}$$



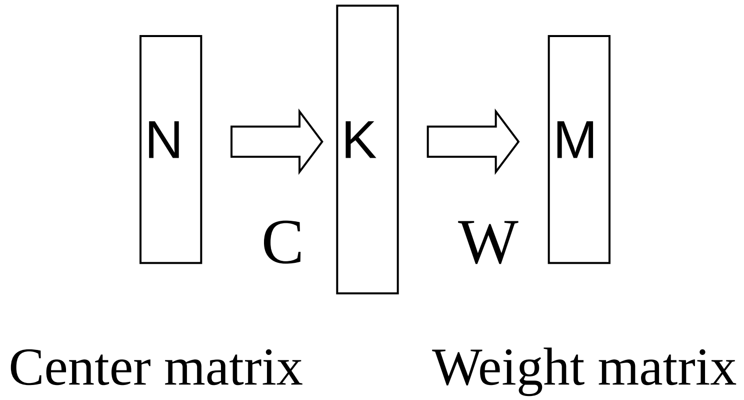
Rmk: the parameter σ controls the width of the gaussian

RBF networks

Computation of the output signal:

$$y_i = \sum_{k=1}^{K} w_{ik}\, g(\|X - C^k\|) - w_{i0},\ i = \overline{1,M}$$

$$y_i = \sum_{k=1}^{K} w_{ik} z_k - w_{i0},\ z_k = g(\|X - C^k\|)$$



The vectors C^k can be interpreted as prototypes;

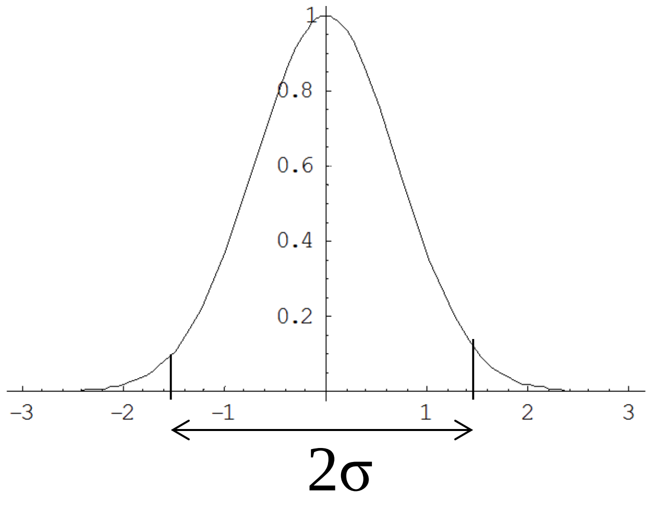
- only input vectors similar to the prototype of the hidden unit “activate” that unit

- the output of the network for a given input vector will be influenced only by the output of the hidden units having centers close enough to the input vector

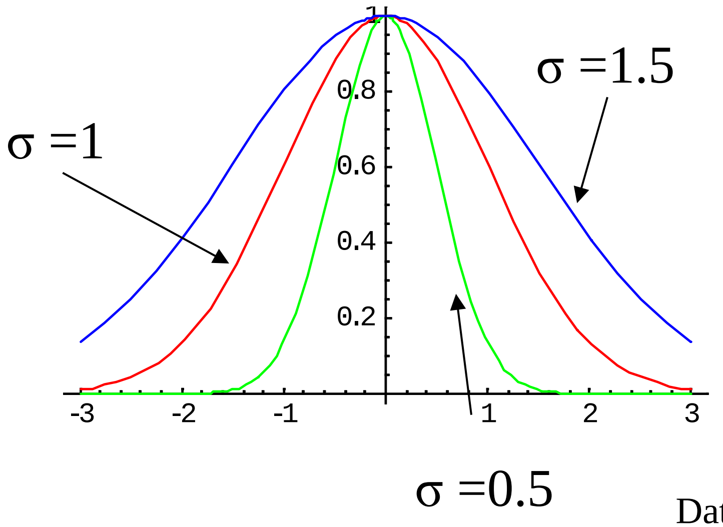
RBF networks

Each hidden unit is “sensitive” to a region in the input space corresponding to a neighborhood of its center. This region is called receptive field

The size of the receptive field depends on the parameter σ



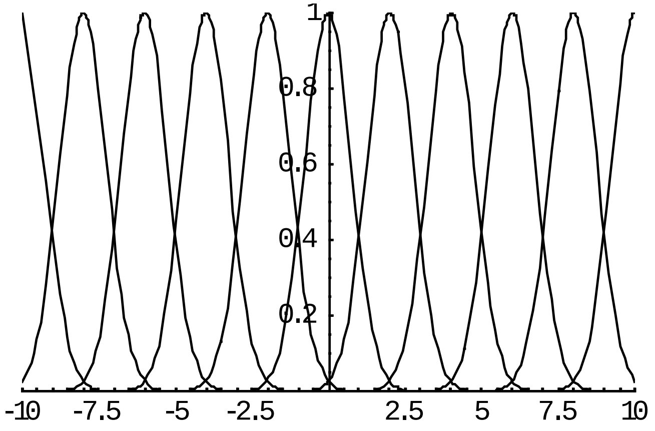
$$g(u) = \exp\left(-\frac{u^2}{2\sigma^2}\right)$$

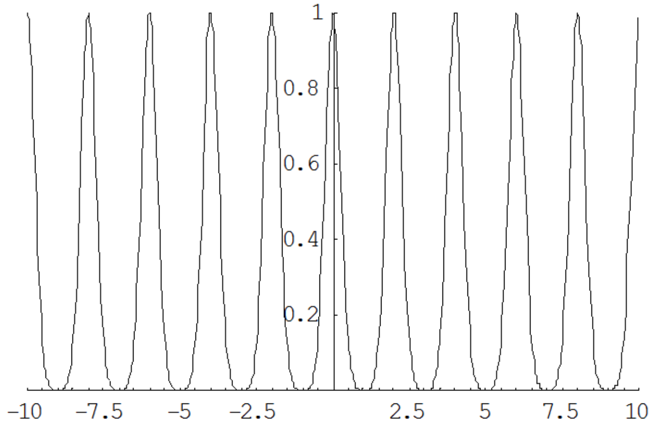


RBF networks

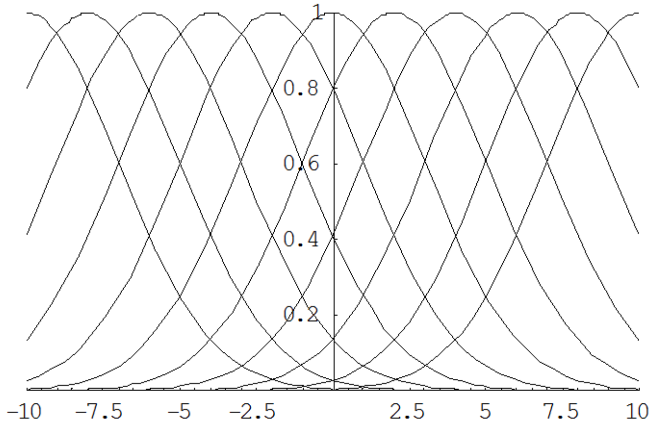
- The receptive fields of all hidden units covers the input space
- A good covering of the input space is essential for the approximation power of the network
- Too small or too large values of the width of the radial basis function lead to inappropriate covering of the input space

appropriate covering





undercovering

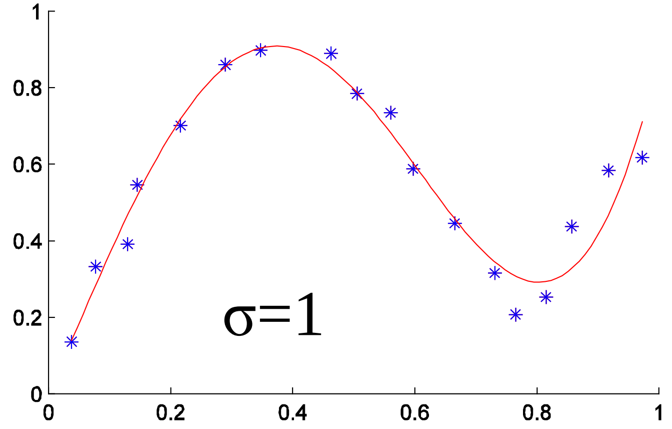


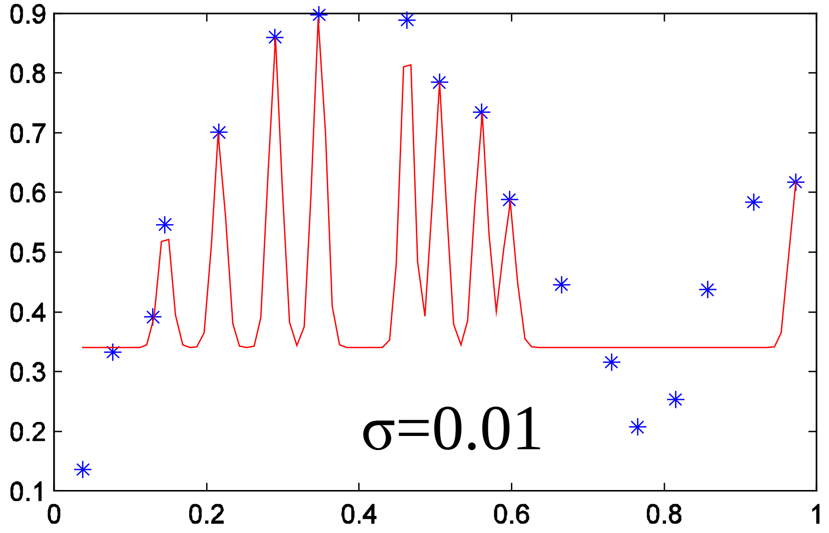
overcovering

RBF networks

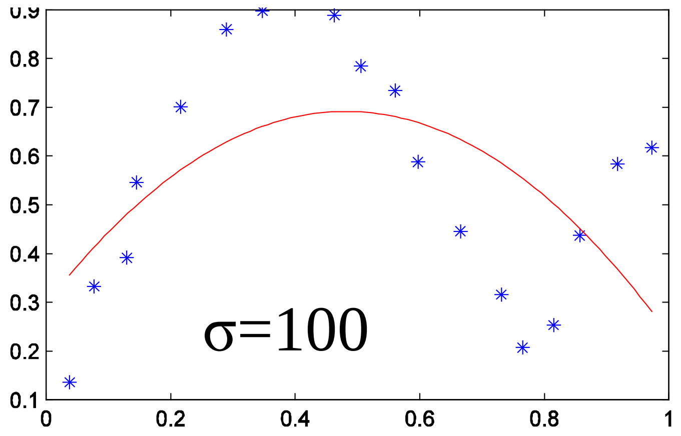
- The receptive fields of all hidden units covers the input space
- A good covering of the input space is essential for the approximation power of the network
- Too small or too large values of the width of the radial basis function lead to inappropriate covering of the input space

appropriate covering





undercovering



overcovering

RBF networks

RBF networks are universal approximators:

a network with N inputs and M outputs can approximate any function defined on R^N, taking values in R^M, as long as there are enough hidden units

The theoretical foundations of RBF networks are:

- Theory of approximation
- Theory of regularization

RBF networks

Adaptive parameters:

- Centers (prototypes) corresponding to hidden units
- Receptive field widths (parameters of the radial symmetry activation functions)
- Weights associated to connections between the hidden and output layers

Learning variants:

- Simultaneous learning of all parameters (similar to BackPropagation)
  - Rmk: same drawbacks as multilayer perceptron's BackPropagation
- Separate learning of parameters: centers, widths, weights

RBF networks

Separate learning :

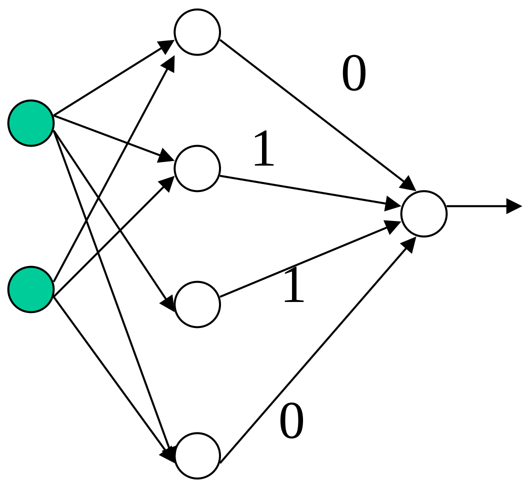
Training set: $\{(x^1,d^1), \dots, (x^L,d^L)\}$

1. Estimating of the centers: simplest variant
   - K=L (nr of centers = nr of examples),
     - $C^k=x^k$ (this corresponds to the case of exact interpolation: see the example for XOR)

RBF networks

Example (particular case) : RBF network to represent XOR

- 2 input units
- 4 hidden units
- 1 output unit



Centers:		Weights:
Hidden unit 1:	(0,0)	w1: 0
Hidden unit 2:	(1,0)	w2: 1
Hidden unit 3:	(0,1)	w3: 1
Hidden unit 4:	(1,1)	w4: 0

Activation function:

g(u)=1 if u=0

g(u)=0 if u<>0

This approach cannot be applied for general approximation problems

RBF networks

Separate learning :

Training set: $\{(x^1,d^1), \ldots, (x^L,d^L)\}$

1. Estimating of the centers

- $K<L$: the centers are established
  - by random selection from the training set
    - simple but not very effective
  - by systematic selection from the training set (Orthogonal Least Squares)
  - by using a clustering method

RBF networks

Orthogonal Least Squares:

- Incremental selection of centers such that the error on the training set is minimized
- The new center is chosen such that it is orthogonal on the space generated by the previously chosen centers (this process is based on the Gram-Schmidt orthogonalization method)
- This approach is related with regularization theory and ridge regression

RBF networks

Clustering:

- Identify K groups in the input data $\{X^1,\ldots,X^L\}$ such that data in a group are sufficiently similar and data in different groups are sufficiently dissimilar
- Each group has a representative (e.g. the mean of data in the group) which can be considered the center
- The algorithms for estimating the representatives of data belong to the class of partitional clustering methods
- Classical algorithm: K-means

RBF networks

Incremental variant:

- Start with a small number of centers, randomly initialized
- Scan the set of input data:
  - If there is a center close enough to the data then this center is slightly adjusted in order to become even closer to the data
  - if the data is dissimilar enough with respect to all centers then a new center is added (the new center will be initialized with the data vector)

RBF networks

Incremental variant:

```
$K := K_0$
$C_i^k := rand(\min, \max), i = 1..N; k = 1..K$
$t := 0$
REPEAT
  FOR $l := 1, L$ DO
      find $k^* \in \{1,\dots,K\}$ such that $d(X^l, C^{k^*}) \leq d(X^l, C^k)$
      IF $d(X^l, C^{k^*}) < \delta$ THEN $C^{k^*} := C^{k^*} + \eta \cdot (X^l - C^{k^*})$
      ELSE $K := K + 1; C^K := X^l$
$t := t + 1$
$\eta := \eta_0 t^{-\alpha}$
UNTIL $t > t_{\max}$ OR $\eta < \varepsilon$
```

δ is a disimilarity threshold
α controls the decrease of the learning rates

RBF networks

2. Estimating the receptive fields widths.

Heuristic rules:

$$\sigma = \frac{d_{\max}}{\sqrt{2K}}, \; d_{\max} = \text{maximal distance between centers}$$

$$\sigma_k = \gamma d(C^k, C^j), C^j = \text{the closest center to } C^k, \gamma \in [0.5,1]$$

$$\sigma_k = \frac{1}{m}\sum_{j=1}^{m} d(C^k, C^j), C^1, \ldots, C^m : \text{the closest } m \text{ centers to } C^j$$

$$\sigma_k = \frac{1}{q_k}\sum_{j=1}^{q_k} d(C^k, X^j), X^1, \ldots, X^{q_k} : \text{input vectors represented by unit k}$$

RBF networks

3. Estimating the weights of connections between hidden and output layers:

- This is equivalent with the problem of training one layer linear network
- Variants:
  - Apply linear algebra tools (pseudo-inverse computation)
  - Apply Widrow-Hoff learning (training based on the gradient method applied to one layer neural networks)

- Initialization:

```
wij(0):=rand(-1,1)  (the weights are randomly initialized in [-1,1]),
k:=0  (iteration counter)
```

- Iterative process

```
REPEAT
    FOR l:=1,L DO
        Compute $y_i(l)$ and $delta_i(l)=d_i(l)-y_i(l)$,  i=1,M
        Adjust the weights: $w_{ij}:=w_{ij}+eta*delta_i(l)*x_j(l)$
    Compute the SSE(W) for the new values of the weights
    k:=k+1
UNTIL SSE(W)<E* OR k>kmax
```

(E*=approximation error, kmax=maximal number of iterations)

RBF vs. BP networks

RBF networks:	BP networks:
• 1 hidden layer	• many hidden layers
• Distance based aggregation function for the hidden units	• Weighted sum as aggregation function for the hidden units
• Activation functions with radial symmetry for hidden units	• Sigmoidal activation functions for hidden neurons
• Linear output units	• Linear/nonlinear output units
• Separate training of adaptive parameters	• Simultaneous training of adaptive parameters
• Similar with local approximation approaches	• Similar with global approximation approaches