Machine Learning

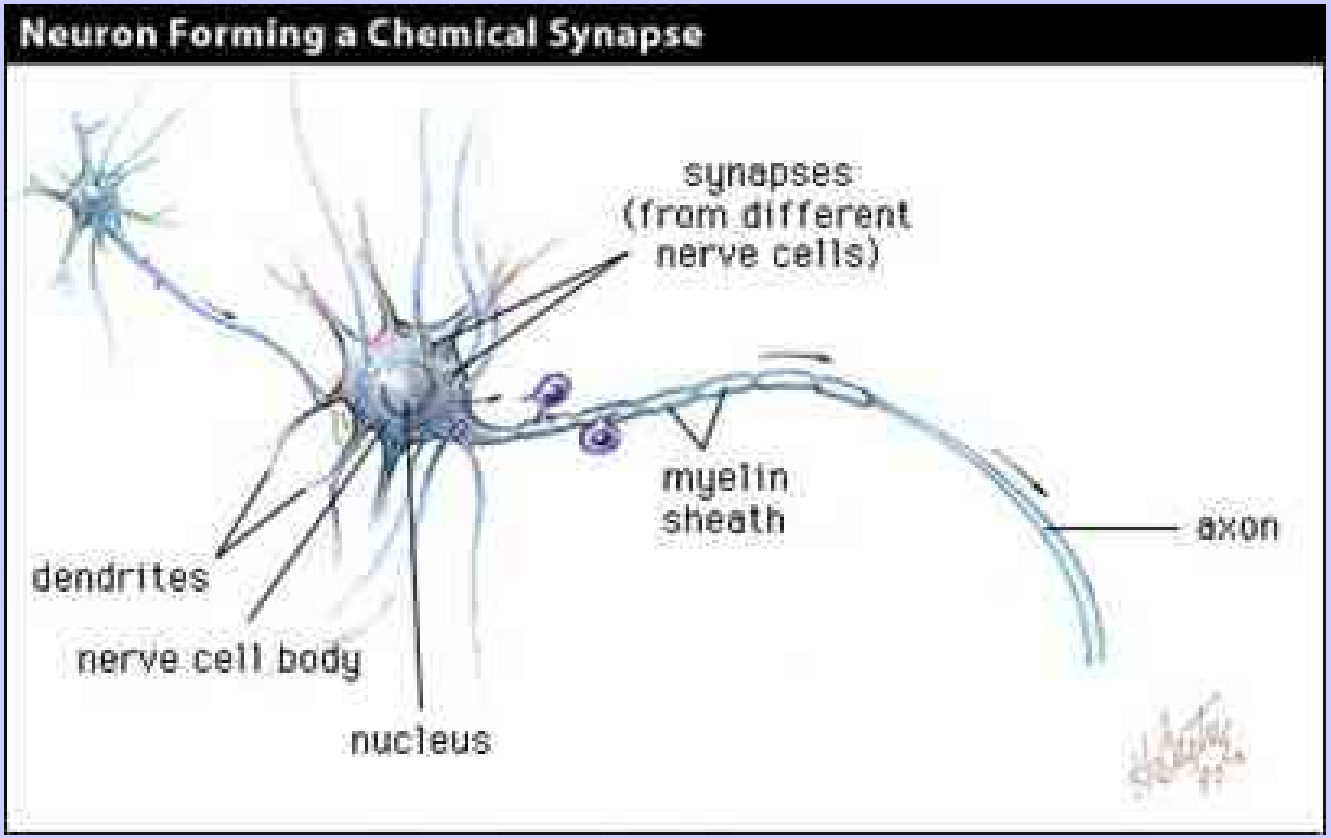
Hauptseminar für Informatiker:
Single-layer neural networks

Referent: Matthias Seidl
Betreuer: Martin Bauer
09.12.2003

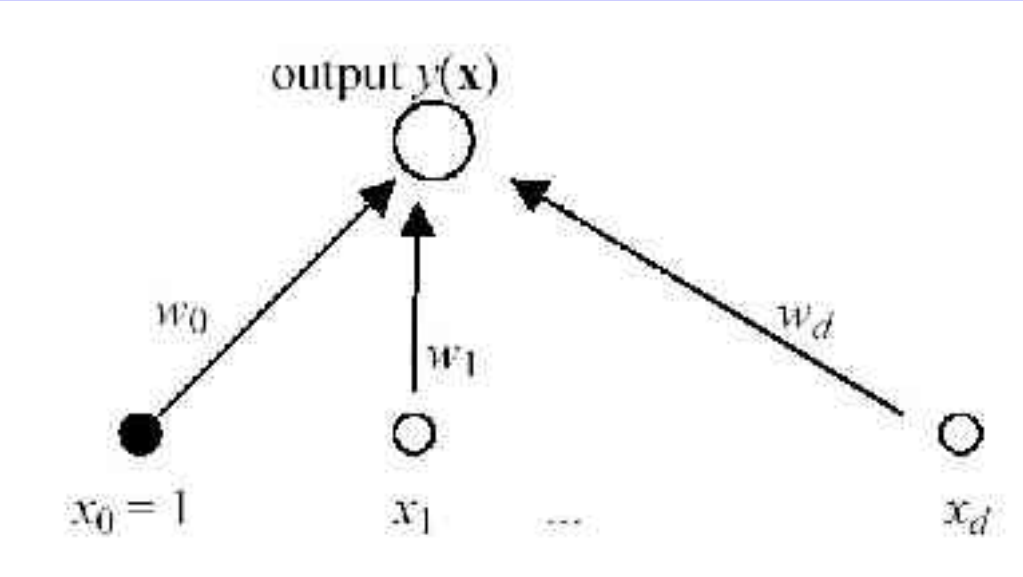
Overview

- Introduction
- Basic characteristics
- Linear separability
- Least-squares techniques
- Perceptron
- Conclusion

The biological neuron



The artificial neuron



- Inputs: $x_1, \ldots, x_d$
- Weights: $w_1, \ldots, w_d$
- Bias: w_0 or threshold: $-w_0$

Applications of neural networks

- Character recognition
- Speech recognition
- Music composition
- Computer Games(e.g. Black & White)
- Forecasting (loan, share etc.)
- Machine control
- Etc ....

Network structures

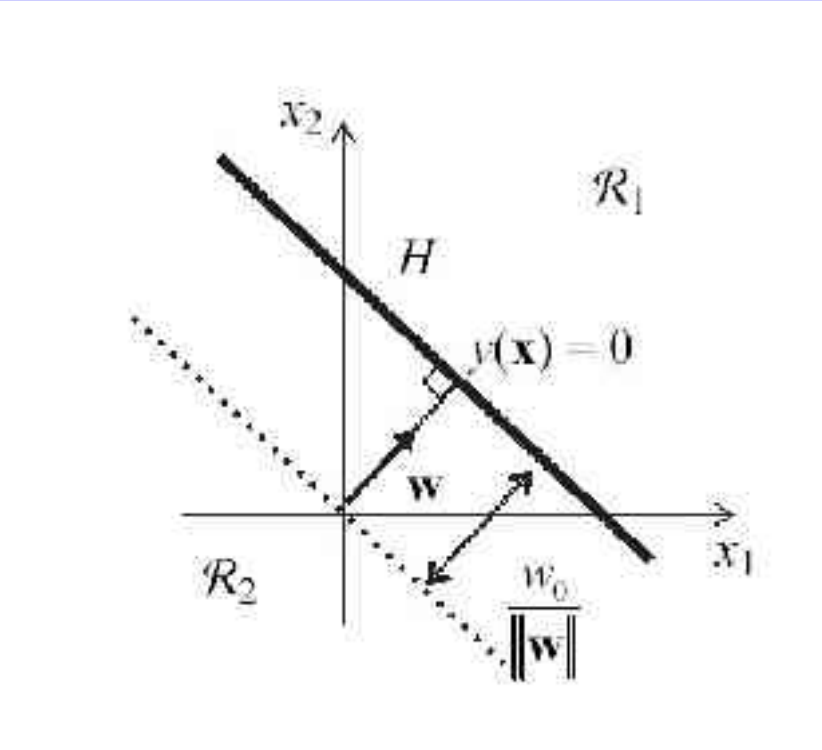
- Feed-forward networks vs Recurrent networks
- Single-layer vs. Multilayer networks
- Supervised vs. Unsupervised
- Continous vs. Binary

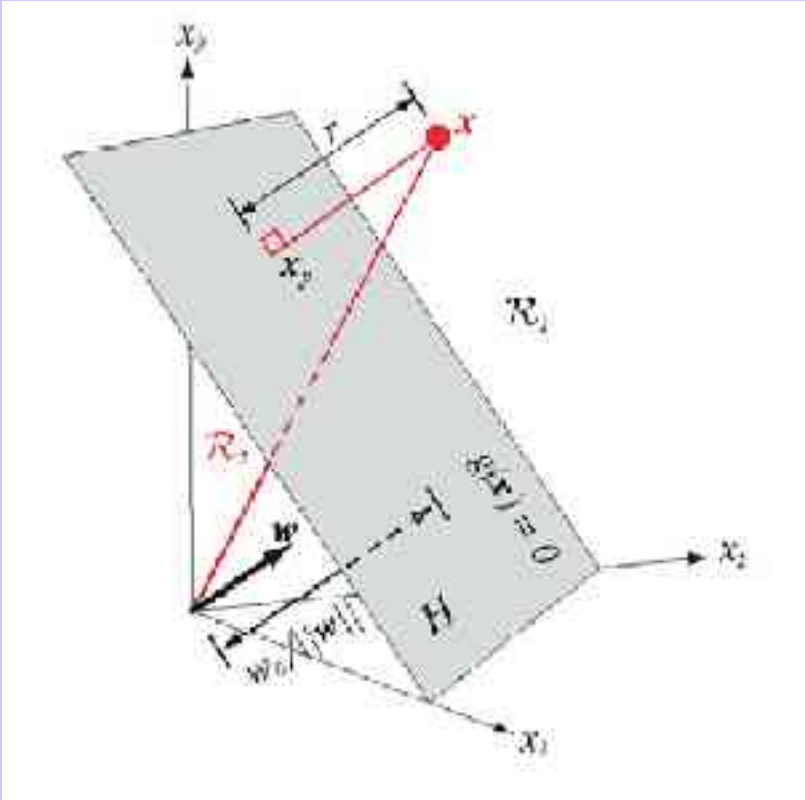
Basic characteristics(1)

- Two Classes: C_1 & C_2
  - Linear discriminant: $y(\vec{x}) = (\vec{w})^T \vec{x} + w_0$
  - Linear dicision boundary: y(x) = 0 corresponds to (d-1)-dimensional hyperplane in d-dimensional x-space
  - W defines orientation of decision boudary
  - Normal distance from the origin to the hyperplane

$$\frac{(\vec{w})^T \cdot \vec{x}}{\|\vec{w}\|} = \frac{-w_0}{\|\vec{w}\|}$$

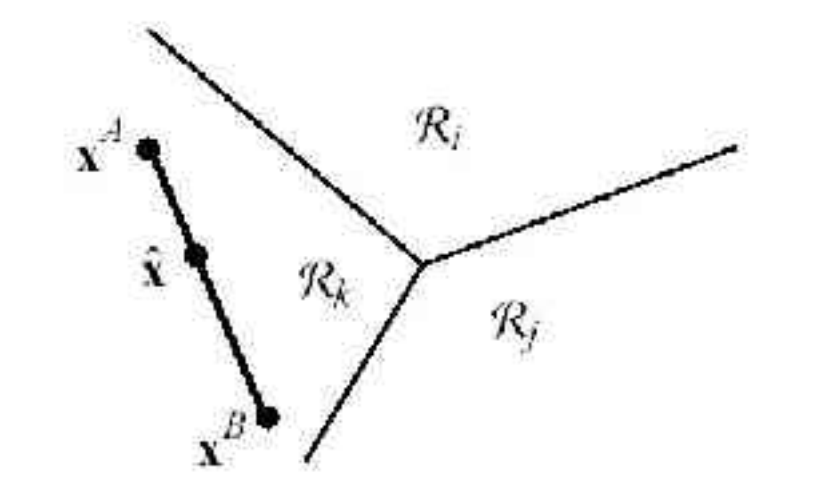
Basic characteristics





Basic Characteristics

- Several Classes: C_1 ,... C_c
  - Linear discriminant: $y_k(\vec{x}) = (\vec{w}_k)^T \vec{x} + w_{k0}$
  - Distance of the decision boundary of the origin: $\frac{-(w_{k0} - w_{j0})}{\|\vec{w}_k - \vec{w}_j\|}$
  - Leads to a set of decision regions, which are connected and convex



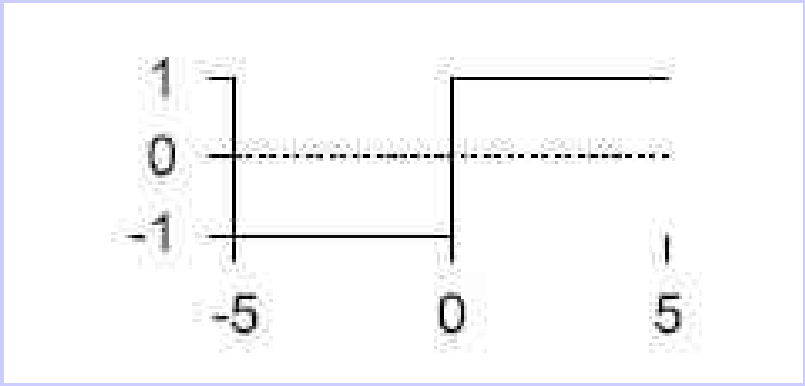
Activation functions

- Activation function

$$y(\vec{x}) = g((\vec{w})^T \vec{x} + w_0)$$

  - Step (Threshold) function

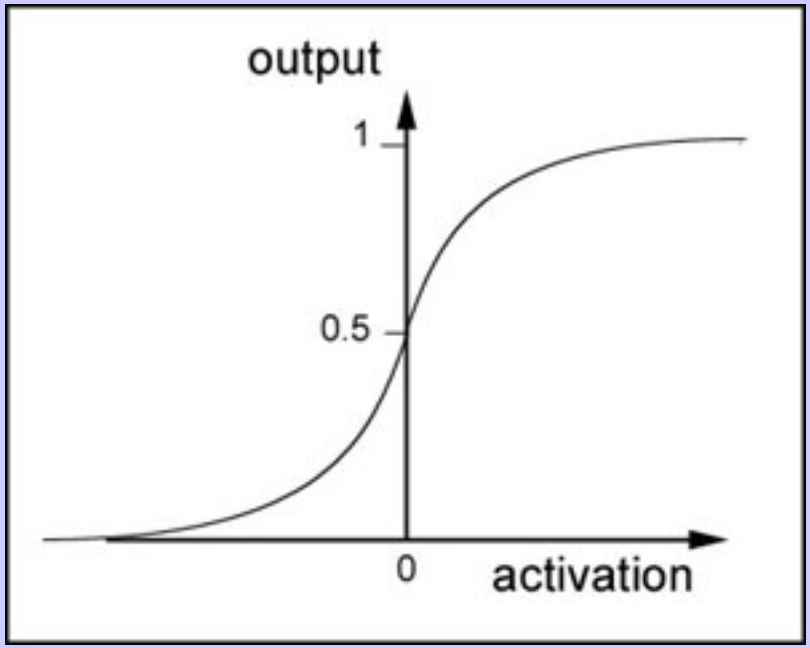
$$y(a) \quad \Theta(a) - \begin{cases} 1 & a > 0 \\ 1 & a \leq 0. \end{cases}$$



  - Linear functions
  - Logistic Sigmoid (=>next slide)

Activation functions

- Logistic sigmoid
  - $y(a) = \frac{1}{1 + e^{-a}}$
  - s-shaped
  - Monotonically increasing
  - Differentiable
  - Maps $(-\infty, \infty)$ auf (0,1)
  - Output of network in a limited range



Logistic Regression

- Motivation for logistic sigmoid: normal distributions with equal covariance matrices

$$p(\mathbf{x}|C_k) = \frac{1}{(2\pi)^{d/2}\|\Sigma\|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_k)^T\Sigma^{-1}(\mathbf{x}-\mu_k)\right)$$

- From Bayes Theorem we have:

$$p(C_1|\mathbf{x}) = \frac{p(\mathbf{x}|C_1)P(C_1)}{p(\mathbf{x}|C_1)P(C_1) + p(\mathbf{x}|C_2)P(C_2)}$$

$$p(C_1|\mathbf{x}) = \frac{1}{1+exp(\quad a)}$$

mit

$$a = \ln \frac{p(\mathbf{x}|C_1)P(C_1)}{p(\mathbf{x}|C_2)P(C_2)}$$

$$g(a) = \frac{1}{1+e^{-a}}$$

Logistic Regression

- After substituting expression for gaussdistribution in expression of Bayes-Theorem we obtain

$$a = \mathbf{w}^T \mathbf{x} + w_0$$

mit $$\mathbf{w} = \Sigma^{-1}(\mu_1 - \mu_2)$$

$$w_0 = -\frac{1}{2}\mu_1^T \Sigma^{-1} \mu_1 + \frac{1}{2}\mu_2^T \Sigma^{-1} \mu_2 + \ln \frac{P(C_1)}{P(C_2)}$$

- => results: next slide

Logistic Regression

- Outputs of neural networks can be interpreteted as posterior probabilities
- Procedure to estimate the weights

Logistic Regression

- Binary Input Vectors
  - Leads to Bernoulli distribution

$$p(\vec{x}|C_k) = \prod_{i=1}^{d} P_{ki}^{x_i}(1-P_{ki}^{1-x_i})$$

- => Outputs of neural Networks can be interpreted as posterior probabilities

Linear Separability

- Definition: If all points of training data is correctly classified by a linear(hyperplanar) decision boundary, then the points are said to be linerarly separable.
- Examples: OR, AND
- Contraexample: XOR, NXOR

Linear Separability

- What fraction of dichtomies is linearly separable?
- Distribute N data points in K dimensions in general position
- Assign the points randomly to Classes C_1 or C_2

$$T(N,K) \begin{cases} 2^N & K > N \\ 2\sum_{k=0}^{K-1} \binom{N-1}{k} & K \leq N \end{cases}$$

- Binary inputs 2^K pattern hence $2^{(2^K)}$ assignments to the two classes. Less than $2^{(2^K)}/K!$ can be implemented by a perceptron and are called treshold logic functions.

Least-squares techniques

- Sum-of sqaures error function

$$E(\vec{w}) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{c} (y_k((\vec{x})^n ; \vec{w}) - t_k^n)^2$$

  - $y_k(\vec{x})^n$:Represents output of unit k
  - t_k^n : target value for output of unit k
  - N : Number of trainig pattern
  - C : Number of outputs

Pseudo-inverse Solution

- Differentiate of sum-of-squares error function:

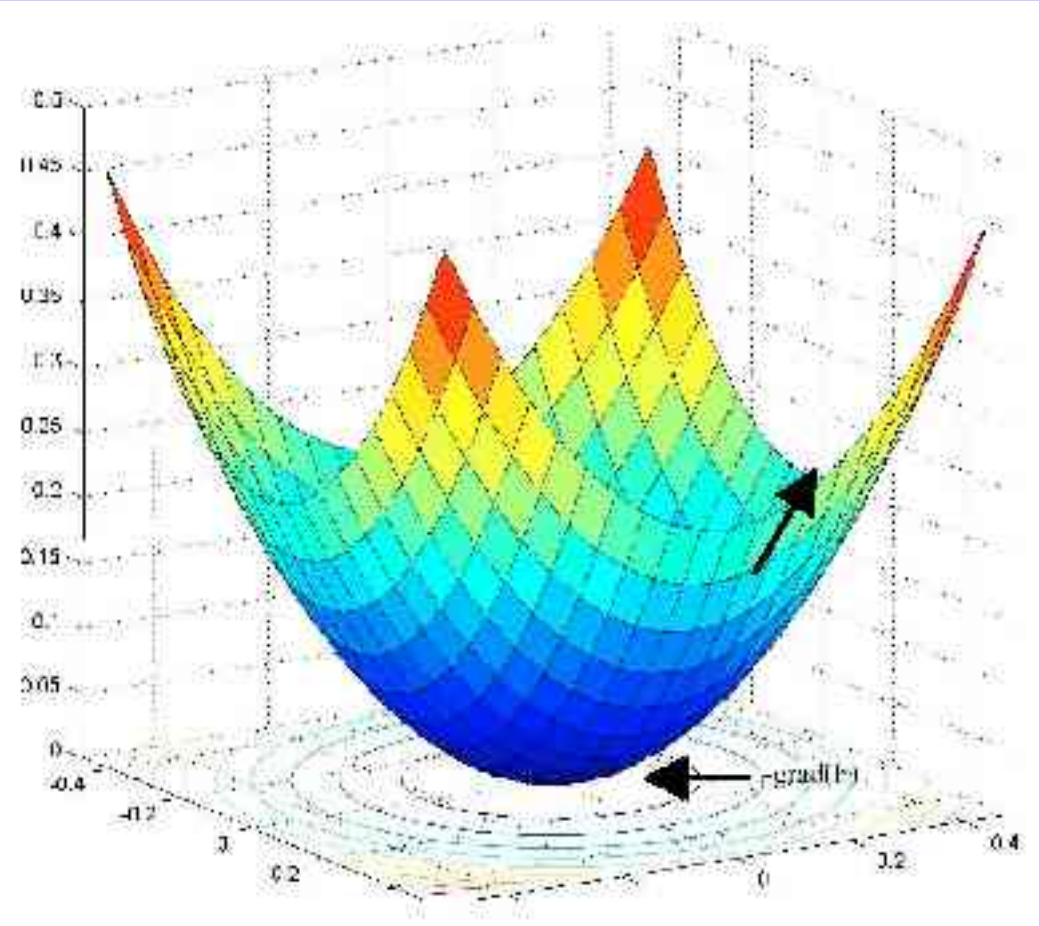
$$\sum_{n=1}^{N} \left(\sum_{j'}^{M} w_{kj'} \Phi_{j'}^{n} - t_k^n \right) \Phi_j^n = 0$$

$$\left(\Phi^T \Phi \right) W^T = \Phi^T T$$

$$W^T = \Phi^P T$$

- Problems:
  - if non-linear activation-function is used then solution is no longer possible
  - If $\Phi^T \Phi$ is singular, no unique solution => SVD => Gradient descent

Gradient Descent



$$\mathbf{w}^{t+1} = \mathbf{w}^{t} - \eta \nabla \mathbf{w}^{t}$$

$$\nabla \mathbf{w} = \left(\frac{\partial E(\mathbf{w})}{\partial w_1}, \frac{\partial E(\mathbf{w})}{\partial w_2}, \ldots, \frac{\partial E(\mathbf{w})}{\partial w_N} \right)$$

Gradient Descent

- For GLN partial differntial is:

$$\frac{(\partial E^n)}{(\partial w_{kj})} = [y_k(\vec{x}^n) - t_k^n]\Phi_j(\vec{x}^n) = \delta_k^n \Phi_j^n$$

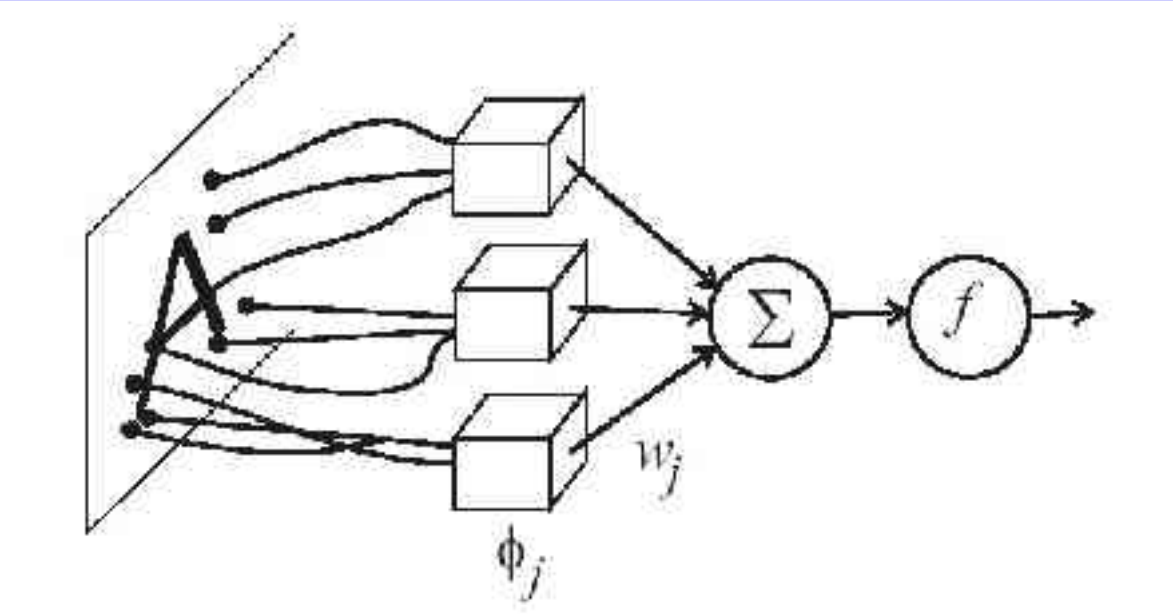
- Leads to delta rule: $\Delta w_{kj} = -\eta \delta_k^n \Phi_j^n$
- Gradient Descent for logistic sigmoid
  - Derivatives of error function: $\delta_k^n = g'(a_k)(y_k(\vec{x}^n) - t_k^n)$ $\frac{(\partial E^n)}{(\partial w_{kj})} = g'(a_k)\delta_k^n \Phi_j^n$ in which:
  - The derivative of logistic sigmoid can easily be expressed in the simple form: $g'(a) = g(a)(1 - g(a))$

Gradient Descent Algorithm

- Initialise weights to random values
- Iterate through a number of epochs. On each epoch do:
  - Run each case through the network, so that the output is produced. Calculate the difference (delta) between the output and the target values. Use this with gradient descent rule to adjust the weights. $w_{kj}^{t+1} = w_{kj}^{t} - \eta \, \delta_k^n \Phi_j^n$

  - When delta-rule becomes almost zero, stop.

Perceptron



Perceptron

- Output of the perceptron:

$$y = g\left(\sum_{j=0}^{M} w_j \Phi_j(\vec{x})\right) = g\left((\vec{w})^T \Phi\right)$$

- Antisymmetric version of treshold function

g(a) $-\Theta(a) - \begin{cases} 1 & a > 0 \\ -1 & a \leq 0. \end{cases}$

Perceptron

- The perceptron criterion: $E^{perc}(\vec{w}) = -\sum_{\Phi^n \in M} (\vec{w})^T (\Phi^n t^n)$

- Perceptron learning: $w_j^{r+1} = w_j^r + \eta \Phi_j^n t^n$

- **Perceptron convergence theorem:**
  For any data set which is linearly separable, the perceptron learning rule is garanteed to find an solution in a finite number of steps

Perceptron

- Applet for Perceptron learning: http://home.cc.umanitoba.ca/~umcorbe9/perceptron.h
- Limitations(Minsky, Pappert)
  - Diameter-limited perceptron

Pros & Cons of single-layer networks

- + simple learning algorithm
- + can solve problems quite readily
- + Insentivity to (moderate) noise or unreliability in data
- + Ability to have more output classes
- - only a small class of problems can be classified correctly (XOR)
- - black box (difficulties in validation the model)

Conclusion

- Single layer neural-networks which form a weighted biased sum of their inputs implement a linear discrimant
- Output of logistic sigmoid network can be interpreted as posterior probabilities
- Can optimize weights using Pseudo-inverse and Gradient descent

Literature

- Christopher M. Bishop Neural Networks for Pattern Recognition" Chapter 3.1.-3.5. , Clarendon Press - Oxford, 1995
- Stuart Russell, Peter Norvig „Artificial Intelligence – A modern approach" Chapter 20.5, Prentice Hall, 2003
- David J.C. MacKay „Information Theory, Inference, and Learning Algorithms" Chapter 38-41, Cambridge University Press
- Online literature:
  - ftp://ftp.sas.com/pub/neural/FAQ.html
  - http://home.cc.umanitoba.ca/~umcorbe9/neuron.html
  - http://www.ai-junkie.com/nnt1.html
  - http://neuralnetworks.ai-depot.com/