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
- Continuous vs. Binary
Basic characteristics(1)

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

- Several Classes $\mathcal{C}_1, \ldots, \mathcal{C}_c$
  \[ y_k(\bar{x}) = (\bar{w}_k)^T \bar{x} + w_{k0} \]
  - Linear discriminant:
  - Distance of the decision boundary of the origin:
    \[ \frac{-(w_{k0} - w_{j0})}{\|\bar{w}_k - \bar{w}_j\|} \]
  - Leads to a set of decision regions, which are connected and convex
Activation functions

- Activation function
  \[ y(\bar{x}) = g((\bar{w})^T \bar{x} + w_0) \]

  - Step (Threshold) function
  \[ y(a) = \Theta(a) = \begin{cases} 
   1 & a > 0 \\
   0 & a \leq 0.
\end{cases} \]

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

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

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

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

• From Bayes Theorem we have:

\[ p(C_1|x) = \frac{p(x|C_1)P(C_1)}{p(x|C_1)P(C_1) + p(x|C_2)P(C_2)} \]

\[ p(C_1|x) = \frac{1}{1 + e^{-a}} \]

\[ a = \ln \left( \frac{p(x|C_1)P(C_1)}{p(x|C_2)P(C_2)} \right) \]

\[ g(a) = \frac{1}{1 + e^{-a}} \]
Logistic Regression

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

\[ a = w^T x + w_0 \]

mit

\[ w = \Sigma^{-1}(\mu_1 - \mu_2) \]

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

• => results: next slide
Logistic Regression

• Outputs of neural networks can be interpreted as posterior probabilities

• Procedure to estimate the weights
Logistic Regression

• Binary Input Vectors
  – Leads to Bernoulli distribution
    \[ p(\tilde{x}|C_k) = \prod_{i=1}^{d} P_{ki}^{x_i}(1 - P_{ki}^{1-x_i}) \]

• \( \Rightarrow \) 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 linearly separable.

• Examples: OR, AND

• Contraexample: XOR, NXOR
Linear Separability

- What fraction of dichotomies is linearly separable?
- Distribute N data points in K dimensions in general position
- Assign the points randomly to Classes $\mathcal{C}_a$ and $\mathcal{C}_b$

$$T(N, K) = \begin{cases} \frac{2^N}{2^\sum_{k=0}^{K-1} (N_k)} & K > N \\ \frac{K}{K!} & K < N \end{cases}$$

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

Solution: generalized linear discriminants
Least-squares techniques

- Sum-of squares error function

\[
E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{C} (y_k((\mathbf{x})^n; \mathbf{w}) - t_k^n)^2
\]

- \( y_k((\mathbf{x})^n) \) : Represents output of unit \( k \)
- \( t_k^n \) : Target value for output of unit \( k \)
- \( N \) : Number of training patterns
- \( C \) : Number of outputs
Pseudo-inverse Solution

- Differentiate of sum-of-squares error function:

\[
\sum_{n=1}^{N} \left( \sum_{j}^{M} w_{kj} \Phi_{j}^{n} - t_{k}^{n} \right) \Phi_{j}^{n} = 0
\]

\[
(\Phi^{T} \Phi) W^{T} = \Phi^{T} I
\]

\[
W^{T} = \Phi^{P} I
\]

- Problems:
  - if non-linear activation-function is used then solution is no longer possible
  - If \( \Phi^{T} \Phi \) is singular, no unique solution \( \Rightarrow \) SVD
  - \( \Rightarrow \) Gradient descent
Gradient Descent

\[ w^{t+1} = w^t - \eta \nabla w^t \]

\[ \nabla w = \left( \frac{\partial E(w)}{\partial w_1}, \frac{\partial E(w)}{\partial w_2}, \ldots, \frac{\partial E(w)}{\partial w_N} \right) \]
Gradient Descent

• For GLN partial differential is:

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

• Leads to delta rule: \( \Delta w_{kj} = -\eta \delta_k^n \Phi_j^n \)

• Gradient Descent for logistic sigmoid

- Derivatives of error function:

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

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:
  \[ y = g\left( \sum_{j=0}^{M} w_j \Phi_j(\tilde{x}) \right) = g((\tilde{w})^T \Phi) \]

• Antisymmetric version of threshold function
  \[ g(a) = \begin{cases} 
  1 & a > 0 \\
  -1 & a \leq 0.
  \end{cases} \]
Perceptron

- The perceptron criterion:
  \[
  E_{\text{perc}}(\vec{w}) = - \sum_{\Phi^i \in M} (\vec{w})^T (\Phi^i t^n)
  \]

- 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 guaranteed to find an solution in a finite number of steps
Perceptron

• Applet for Perceptron learning:
  http://home.cc.umanitoba.ca/~umcorbe9/perceptron.html

• 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 discriminant

● 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
• Online literature: