These are the slides of the lecture

Pattern Recognition
Winter term 2011/12
Friedrich-Alexander University of Erlangen-Nuremberg.

These slides are for your personal usage only in order to prepare for the examination.

Erlangen, April 25, 2018
Dr.-Ing. Stefan Steidl

Pattern Recognition (PR)

Winter Term 2011/12

Stefan Steidl
Computer Science Dept. 5
(Pattern Recognition)

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Motivation

- We want to compute a linear decision boundary.
- We assume that classes are linearly separable.
- Computation of a linear separating hyperplane that minimizes the distance of misclassified feature vectors to the decision boundary.

Objective Function

Assume the following:

- Class numbers are $y = \pm 1$
- The decision boundary is a linear function:

$$y^* = \mathrm{sgn}(\boldsymbol{\alpha}^T \boldsymbol{x} + \alpha_0).$$

Objective Function

Assume the following:

- Class numbers are $y = \pm 1$
- The decision boundary is a linear function:

$$y^* = \operatorname{sgn}(\boldsymbol{\alpha}^T \boldsymbol{x} + \alpha_0).$$

- Parameters α_0 and $\boldsymbol{\alpha}$ are chosen according to the optimization problem

$$\text{minimize} \qquad D(\alpha_0, \boldsymbol{\alpha}) = -\sum_{\boldsymbol{x}_i \in \mathcal{M}} y_i \cdot (\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0)$$

  where $\mathcal{M}$ includes the misclassified feature vectors.

Objective Function (cont.)

- The elements of the sum in the objective function depend on the set of misclassified feature vectors $\mathcal{M}$.
- In each iteration step the cardinality of $\mathcal{M}$ might change.
- The cardinality of $\mathcal{M}$ is a discrete variable.
- Competing variables: continuous parameters of linear decision boundary and the discrete cardinality of $\mathcal{M}$.

Minimization of Objective Function

Remember the objective function $D(\alpha_0, \boldsymbol{\alpha})$:

$$\text{minimize} \qquad D(\alpha_0, \boldsymbol{\alpha}) = -\sum_{\boldsymbol{x}_i \in \mathcal{M}} y_i \cdot (\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0)$$

Minimization of Objective Function

Remember the objective function $D(\alpha_0, \boldsymbol{\alpha})$:

$$\text{minimize} \qquad D(\alpha_0, \boldsymbol{\alpha}) = -\sum_{\boldsymbol{x}_i \in \mathcal{M}} y_i \cdot (\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0)$$

The gradient of the objective function is:

Minimization of Objective Function

Remember the objective function $D(\alpha_0, \boldsymbol{\alpha})$:

$$\text{minimize} \qquad D(\alpha_0, \boldsymbol{\alpha}) = -\sum_{\boldsymbol{x}_i \in \mathcal{M}} y_i \cdot (\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0)$$

The gradient of the objective function is:

$$\frac{\partial}{\partial \alpha_0} D(\alpha_0, \boldsymbol{\alpha}) \quad = \quad -\sum_{\boldsymbol{x}_i \in \mathcal{M}} y_i$$

Minimization of Objective Function

Remember the objective function $D(\alpha_0, \boldsymbol{\alpha})$:

$$\text{minimize} \qquad D(\alpha_0, \boldsymbol{\alpha}) = -\sum_{\boldsymbol{x}_i \in \mathcal{M}} y_i \cdot (\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0)$$

The gradient of the objective function is:

$$\frac{\partial}{\partial \alpha_0} D(\alpha_0, \boldsymbol{\alpha}) = -\sum_{\boldsymbol{x}_i \in \mathcal{M}} y_i$$

$$\frac{\partial}{\partial \boldsymbol{\alpha}} D(\alpha_0, \boldsymbol{\alpha}) = -\sum_{\boldsymbol{x}_i \in \mathcal{M}} y_i \cdot \boldsymbol{x}_i$$

Minimization of Objective Function (cont.)

We want to take an update step right after having visited each misclassified observation. The update rule in the $(k+1)$-st iteration step is:

$$\begin{pmatrix} \alpha_0^{(k+1)} \\ \boldsymbol{\alpha}^{(k+1)} \end{pmatrix} =$$

Minimization of Objective Function (cont.)

We want to take an update step right after having visited each misclassified observation. The update rule in the $(k+1)$-st iteration step is:

$$\begin{pmatrix} \alpha_0^{(k+1)} \\ \boldsymbol{\alpha}^{(k+1)} \end{pmatrix} = \begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix} + \lambda \begin{pmatrix} y_i \\ y_i \cdot \boldsymbol{x}_i \end{pmatrix}$$

Here λ is the learning rate which can be set to 1 without loss of generality.

Minimization of Objective Function (cont.)

Input: training data: $S = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), (\boldsymbol{x}_3, y_3), \ldots, (\boldsymbol{x}_m, y_m)\}$

Minimization of Objective Function (cont.)

Input: training data: $S = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), (\boldsymbol{x}_3, y_3), \dots, (\boldsymbol{x}_m, y_m)\}$
initialize $k = 0$, $\alpha_0^{(0)} = 0$ and $\boldsymbol{\alpha}^{(0)} = \mathbf{0}$
repeat
select pair $(\boldsymbol{x}_i, y_i)$ from training set.

Minimization of Objective Function (cont.)

Input: training data: $S = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), (\boldsymbol{x}_3, y_3), \ldots, (\boldsymbol{x}_m, y_m)\}$
initialize $k = 0$, $\alpha_0^{(0)} = 0$ and $\boldsymbol{\alpha}^{(0)} = \mathbf{0}$
repeat
select pair $(\boldsymbol{x}_i, y_i)$ from training set.
if $y_i \cdot (\boldsymbol{x}_i^T \boldsymbol{\alpha}^{(k)} + \alpha_0^{(k)}) \leq 0$ **then**

$$\begin{pmatrix} \alpha_0^{(k+1)} \\ \boldsymbol{\alpha}^{(k+1)} \end{pmatrix} = \begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix} + \begin{pmatrix} y_i \\ y_i \cdot \boldsymbol{x}_i \end{pmatrix}$$

$k \leftarrow k + 1$
end if

Minimization of Objective Function (cont.)

Input: training data: $S = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), (\boldsymbol{x}_3, y_3), \dots, (\boldsymbol{x}_m, y_m)\}$
initialize $k = 0$, $\alpha_0^{(0)} = 0$ and $\boldsymbol{\alpha}^{(0)} = \boldsymbol{0}$
repeat
select pair $(\boldsymbol{x}_i, y_i)$ from training set.
if $y_i \cdot (\boldsymbol{x}_i^T \boldsymbol{\alpha}^{(k)} + \alpha_0^{(k)}) \leq 0$ **then**

$$\begin{pmatrix} \alpha_0^{(k+1)} \\ \boldsymbol{\alpha}^{(k+1)} \end{pmatrix} = \begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix} + \begin{pmatrix} y_i \\ y_i \cdot \boldsymbol{x}_i \end{pmatrix}$$

$k \leftarrow k + 1$
end if
until $y_i \cdot (\boldsymbol{x}_i^T \boldsymbol{\alpha}^{(k)} + \alpha_0^{(k)}) > 0$ for all i
Output: $\alpha_0^{(k)}$ and $\boldsymbol{\alpha}^{(k)}$

Remarks on Perceptron Learning

- The update rule is extremely simple.
- Nothing happens if we classify all $\boldsymbol{x}_i$ correctly using the given linear decision boundary.
- The parameter α of the decision boundary is a linear combination of feature vectors.

Remarks on Perceptron Learning

- The update rule is extremely simple.
- Nothing happens if we classify all $\boldsymbol{x}_i$ correctly using the given linear decision boundary.
- The parameter α of the decision boundary is a linear combination of feature vectors.
- The decision boundary thus is:

$$F(\boldsymbol{x}) = \left(\sum_{i \in \mathcal{E}} y_i \cdot \boldsymbol{x}_i \right)^T \boldsymbol{x} + \sum_{i \in \mathcal{E}} y_i \quad = \quad \sum_{i \in \mathcal{E}} y_i \cdot \langle \boldsymbol{x}_i, \boldsymbol{x} \rangle + \sum_{i \in \mathcal{E}} y_i$$

where $\mathcal{E}$ is the set of indices that required an update.

Remarks on Perceptron Learning (cont.)

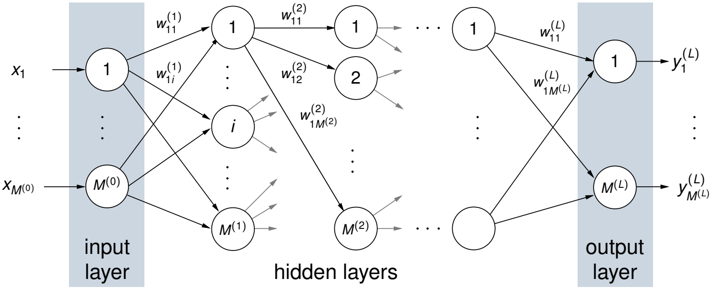
- The final linear decision boundary depends on the initialization, i. e. $\alpha_0^{(0)}$ and $\boldsymbol{\alpha}^{(0)}$.
- The number of iterations can be rather large.
- If data are not linearly separable, the proposed learning algorithm will not converge. The algorithm will end up in hard to detect cycles.

Multi-Layer Perceptrons

- Physiological Motivation
- Topology and Activation Functions
- Backpropagation Algorithm
- Lessons Learned
- Further Readings

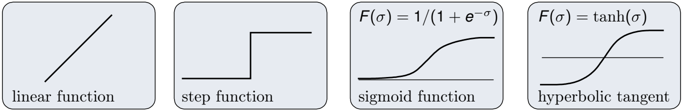
Multi-Layer Perceptrons

Topology



Multi-Layer Perceptrons (cont.)

Activation Functions



$$\text{net}_j^{(l)} = \sum_{i=1}^{M^{(l-1)}} y_i^{(l-1)} w_{ij}^{(l)} - w_{0j}^{(l)}$$

$$y_j^{(l)} = f(\text{net}_j^{(l)})$$

Backpropagation Algorithm

Supervised Learning Algorithm

- Gradient descent to adjust the weights reducing the training error ε:

$$\Delta w_{ij}^{(l)} = -\eta \frac{\partial \varepsilon}{\partial w_{ij}^{(l)}}$$

- Typical error function: mean squared error

$$\varepsilon_{\text{MSE}}(\boldsymbol{w}) = \frac{1}{2} \sum_{k=1}^{M^{(L)}} (t_k - y_k^{(L)})^2$$

Backpropagation Algorithm (cont.)

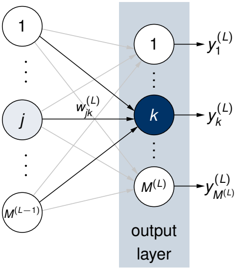
Adjusting the weights $w_{jk}^{(L)}$ of the output layer

$$\frac{\partial \varepsilon_{\text{MSE}}}{\partial w_{jk}^{(L)}} = \frac{\partial \varepsilon_{\text{MSE}}}{\partial \text{net}_k^{(L)}} \cdot \frac{\partial \text{net}_k^{(L)}}{\partial w_{jk}^{(L)}} = -\delta_k^{(L)} \cdot y_j^{(L-1)}$$

The *sensitivity* $\delta_k^{(L)}$:

$$\delta_k^{(L)} = -\frac{\partial \varepsilon_{\text{MSE}}}{\partial \text{net}_k^{(L)}} = -\frac{\partial \varepsilon_{\text{MSE}}}{\partial y_k^{(L)}} \cdot \frac{\partial y_k^{(L)}}{\partial \text{net}_k^{(L)}}$$

$$= (t_k - y_k^{(L)})\, f'(\text{net}_k^{(L)})$$



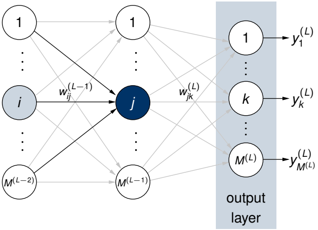
Backpropagation Algorithm (cont.)

Adjusting the weights $w_{jk}^{(L)}$ of the hidden layers

- Desired output values for the hidden layers are not known.
- For the weights $w_{ij}^{(L-1)}$ of the last hidden layer:

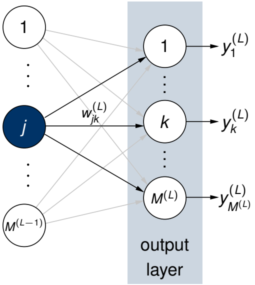
$$\frac{\partial \varepsilon_{\text{MSE}}}{\partial w_{ij}^{(L-1)}} = \frac{\partial \varepsilon_{\text{MSE}}}{\partial y_j^{(L-1)}} \cdot \frac{\partial y_j^{(L-1)}}{\partial \text{net}_j^{(L-1)}} \cdot \frac{\partial \text{net}_j^{(L-1)}}{\partial w_{ij}^{(L-1)}}$$

$$= \frac{\partial \varepsilon_{\text{MSE}}}{\partial y_j^{(L-1)}} \cdot f'(\text{net}_j^{(L-1)}) \cdot y_i^{(L-2)}$$



Backpropagation Algorithm (cont.)

$$
\begin{aligned}
\frac{\partial \varepsilon_{\text{MSE}}}{\partial y_j^{(L-1)}} &= \frac{\partial}{\partial y_j^{(L-1)}} \left[\frac{1}{2} \sum_{k=1}^{M^{(L)}} (t_k - y_k^{(L)})^2 \right] \\
&= -\sum_{k=1}^{M^{(L)}} (t_k - y_k^{(L)}) \frac{\partial y_k^{(L)}}{\partial y_j^{(L-1)}} \\
&= -\sum_{k=1}^{M^{(L)}} (t_k - y_k^{(L)}) \frac{\partial y_k^{(L)}}{\partial \text{net}_k^{(L)}} \cdot \frac{\partial \text{net}_k^{(L)}}{\partial y_j^{(L-1)}} \\
&= -\sum_{k=1}^{M^{(L)}} (t_k - y_k^{(L)}) f'(\text{net}_k^{(L)}) \, w_{jk}^{(L)} \\
&= -\sum_{k=1}^{M^{(L)}} \delta_k^{(L)} w_{jk}^{(L)}
\end{aligned}
$$



Backpropagation Algorithm (cont.)

Sensivity $\delta_j^{(l)}$ for any hidden layer l, $0 < l < L$

$$\delta_j^{(l)} = f'(\mathrm{net}_j^{(l)}) \sum_{k=1}^{M^{(l+1)}} w_{jk}^{(l+1)} \, \delta_k^{(l+1)}$$

Update rule

$$\Delta w_{ij}^{(l)} = \eta \, \delta_j^{(l)} \, y_i^{(l-1)}$$