These are the slides of the lecture

Pattern Recognition
Winter term 2020/21
Friedrich-Alexander University of Erlangen-Nuremberg.

Erlangen, January 8, 2021
Prof. Dr.-Ing. Andreas Maier

Pattern Recognition (PR)

Prof. Dr.-Ing. Andreas Maier
Pattern Recognition Lab (CS 5), Friedrich-Alexander-Universität Erlangen-Nürnberg
Winter Term 2020/21

Rosenblatt’s Perceptron (1957)

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:

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

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

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

where $\mathscr{M}$ 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 $\mathscr{M}$.

Objective Function (cont.)

- The elements of the sum in the objective function depend on the set of misclassified feature vectors $\mathscr{M}$.
- In each iteration step the cardinality of $\mathscr{M}$ might change.

Objective Function (cont.)

- The elements of the sum in the objective function depend on the set of misclassified feature vectors $\mathscr{M}$.
- In each iteration step the cardinality of $\mathscr{M}$ might change.
- The cardinality of $\mathscr{M}$ is a discrete variable.

Objective Function (cont.)

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

Minimization of Objective Function

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

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

Minimization of Objective Function

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

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

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 $D(\alpha_0, \boldsymbol{\alpha})$:

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

The gradient of the objective function is:

$$\begin{aligned} \frac{\partial}{\partial \alpha_0} D(\alpha_0, \boldsymbol{\alpha}) &= -\sum_{\boldsymbol{x}_i \in \mathscr{M}} y_i \\ \frac{\partial}{\partial \boldsymbol{\alpha}} D(\alpha_0, \boldsymbol{\alpha}) &= -\sum_{\boldsymbol{x}_i \in \mathscr{M}} y_i \cdot \boldsymbol{x}_i \end{aligned}$$

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

Minimization of Objective Function (cont.)

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

initialize $k = 0$, $\alpha_0^{(0)} = 0$ and $\boldsymbol{\alpha}^{(0)} = 0$

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 $\boldsymbol{\alpha}^{(0)} = 0$

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

initialize $k = 0$, $\alpha_0^{(0)} = 0$ and $\boldsymbol{\alpha}^{(0)} = 0$

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 $\boldsymbol{\alpha}$ of the decision boundary is a linear combination of feature vectors.
- The decision boundary thus is:

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

where $\mathscr{E}$ is the list of indices that required an update (indices may appear more than once).

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.

Convergence of Learning Algorithm

Theorem (Convergence Theorem of Rosenblatt and Novikoff)

Assume that for all $i = 1, 2, \ldots, m$

$$y_i(\boldsymbol{x}_i^T \boldsymbol{\alpha}^* + \alpha_0^*) \geq \rho$$

where $\rho > 0$ *and* $\|\boldsymbol{\alpha}^*\| = 1$. *Let* $M = \max_i \|\boldsymbol{x}_i\|_2$.

The perceptron learning algorithm converges to a linear decision boundary after k *iterations, where* k *is bounded by*

$$k \leq \frac{(\alpha_0^{*2} + 1)(1 + M^2)}{\rho^2}.$$

Convergence of Learning Algorithm (cont.)

Let us look at the estimated parameters after k iterations and how the parameters change with iterations:

$$\begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix}^T \begin{pmatrix} \alpha_0^{*} \\ \boldsymbol{\alpha}^{*} \end{pmatrix} =$$

Convergence of Learning Algorithm (cont.)

Let us look at the estimated parameters after k iterations and how the parameters change with iterations:

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

Convergence of Learning Algorithm (cont.)

Let us look at the estimated parameters after k iterations and how the parameters change with iterations:

$$\begin{aligned}
\begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix}^T \begin{pmatrix} \alpha_0^* \\ \boldsymbol{\alpha}^* \end{pmatrix} &= \left(\begin{pmatrix} \alpha_0^{(k-1)} \\ \boldsymbol{\alpha}^{(k-1)} \end{pmatrix} + \begin{pmatrix} y_i \\ y_i \cdot \boldsymbol{x}_i \end{pmatrix} \right)^T \begin{pmatrix} \alpha_0^* \\ \boldsymbol{\alpha}^* \end{pmatrix} \\
&= \begin{pmatrix} \alpha_0^{(k-1)} \\ \boldsymbol{\alpha}^{(k-1)} \end{pmatrix}^T \begin{pmatrix} \alpha_0^* \\ \boldsymbol{\alpha}^* \end{pmatrix} + \begin{pmatrix} y_i \\ y_i \cdot \boldsymbol{x}_i \end{pmatrix}^T \begin{pmatrix} \alpha_0^* \\ \boldsymbol{\alpha}^* \end{pmatrix}
\end{aligned}$$

Convergence of Learning Algorithm (cont.)

Let us look at the estimated parameters after k iterations and how the parameters change with iterations:

$$
\begin{aligned}
\begin{pmatrix}\alpha_0^{(k)}\\ \boldsymbol{\alpha}^{(k)}\end{pmatrix}^T \begin{pmatrix}\alpha_0^{*}\\ \boldsymbol{\alpha}^{*}\end{pmatrix} &= \left(\begin{pmatrix}\alpha_0^{(k-1)}\\ \boldsymbol{\alpha}^{(k-1)}\end{pmatrix} + \begin{pmatrix}y_i\\ y_i\cdot\boldsymbol{x}_i\end{pmatrix}\right)^T \begin{pmatrix}\alpha_0^{*}\\ \boldsymbol{\alpha}^{*}\end{pmatrix}\\
&= \begin{pmatrix}\alpha_0^{(k-1)}\\ \boldsymbol{\alpha}^{(k-1)}\end{pmatrix}^T \begin{pmatrix}\alpha_0^{*}\\ \boldsymbol{\alpha}^{*}\end{pmatrix} + \begin{pmatrix}y_i\\ y_i\cdot\boldsymbol{x}_i\end{pmatrix}^T \begin{pmatrix}\alpha_0^{*}\\ \boldsymbol{\alpha}^{*}\end{pmatrix}\\
&\geq \begin{pmatrix}\alpha_0^{(k-1)}\\ \boldsymbol{\alpha}^{(k-1)}\end{pmatrix}^T \begin{pmatrix}\alpha_0^{*}\\ \boldsymbol{\alpha}^{*}\end{pmatrix} + \rho
\end{aligned}
$$

Convergence of Learning Algorithm (cont.)

Let us look at the estimated parameters after k iterations and how the parameters change with iterations:

$$
\begin{aligned}
\begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix}^T \begin{pmatrix} \alpha_0^{*} \\ \boldsymbol{\alpha}^{*} \end{pmatrix} &= \left(\begin{pmatrix} \alpha_0^{(k-1)} \\ \boldsymbol{\alpha}^{(k-1)} \end{pmatrix} + \begin{pmatrix} y_i \\ y_i \cdot \boldsymbol{x}_i \end{pmatrix} \right)^T \begin{pmatrix} \alpha_0^{*} \\ \boldsymbol{\alpha}^{*} \end{pmatrix} \\
&= \begin{pmatrix} \alpha_0^{(k-1)} \\ \boldsymbol{\alpha}^{(k-1)} \end{pmatrix}^T \begin{pmatrix} \alpha_0^{*} \\ \boldsymbol{\alpha}^{*} \end{pmatrix} + \begin{pmatrix} y_i \\ y_i \cdot \boldsymbol{x}_i \end{pmatrix}^T \begin{pmatrix} \alpha_0^{*} \\ \boldsymbol{\alpha}^{*} \end{pmatrix} \\
&\geq \begin{pmatrix} \alpha_0^{(k-1)} \\ \boldsymbol{\alpha}^{(k-1)} \end{pmatrix}^T \begin{pmatrix} \alpha_0^{*} \\ \boldsymbol{\alpha}^{*} \end{pmatrix} + \rho \\
&\geq k\rho
\end{aligned}
$$

Conclusion: The more iterations (i. e. misclassifications) we have, the more the vectors are aligned.

Convergence of Learning Algorithm (cont.)

Now we apply Cauchy-Schwartz inequality for inner products:

$$\begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix}^T \begin{pmatrix} \alpha_0^* \\ \boldsymbol{\alpha}^* \end{pmatrix} \leq \left\| \begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix} \right\|_2 \cdot \left\| \begin{pmatrix} \alpha_0^* \\ \boldsymbol{\alpha}^* \end{pmatrix} \right\|_2$$

Convergence of Learning Algorithm (cont.)

Now we apply Cauchy-Schwartz inequality for inner products:

$$\begin{aligned}
\begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix}^T \begin{pmatrix} \alpha_0^{*} \\ \boldsymbol{\alpha}^{*} \end{pmatrix} &\leq \left\| \begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix} \right\|_2 \cdot \left\| \begin{pmatrix} \alpha_0^{*} \\ \boldsymbol{\alpha}^{*} \end{pmatrix} \right\|_2 \\
&= \left\| \begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix} \right\|_2 \cdot \sqrt{\alpha_0^{*2} + 1}
\end{aligned}$$

Convergence of Learning Algorithm (cont.)

The norm of the vector estimated in the k-th iteration step is:

$$\left\| \begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix} \right\|_2^2 = \left\| \begin{pmatrix} \alpha_0^{(k-1)} \\ \boldsymbol{\alpha}^{(k-1)} \end{pmatrix} + \begin{pmatrix} y_i \\ y_i \cdot \boldsymbol{x}_i \end{pmatrix} \right\|_2^2$$

Convergence of Learning Algorithm (cont.)

The norm of the vector estimated in the k-th iteration step is:

$$\left\| \begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix} \right\|_2^2 = \left\| \begin{pmatrix} \alpha_0^{(k-1)} \\ \boldsymbol{\alpha}^{(k-1)} \end{pmatrix} + \begin{pmatrix} y_i \\ y_i \cdot \boldsymbol{x}_i \end{pmatrix} \right\|_2^2$$

$$= \begin{pmatrix} \alpha_0^{(k-1)} \\ \boldsymbol{\alpha}^{(k-1)} \end{pmatrix}^T \begin{pmatrix} \alpha_0^{(k-1)} \\ \boldsymbol{\alpha}^{(k-1)} \end{pmatrix} + 2 \begin{pmatrix} \alpha_0^{(k-1)} \\ \boldsymbol{\alpha}^{(k-1)} \end{pmatrix}^T \begin{pmatrix} y_i \\ y_i \cdot \boldsymbol{x}_i \end{pmatrix} + \begin{pmatrix} y_i \\ y_i \cdot \boldsymbol{x}_i \end{pmatrix}^T \begin{pmatrix} y_i \\ y_i \cdot \boldsymbol{x}_i \end{pmatrix}$$

Convergence of Learning Algorithm (cont.)

We only go into iteration step $(k+1)$ if we did a mistake in iteration k.
A misclassification implies:

$$\begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix}^T \begin{pmatrix} y_i \\ y_i \cdot \boldsymbol{x}_i \end{pmatrix} = y_i \cdot (\boldsymbol{x}_i^T \boldsymbol{\alpha}^{(k)} + \alpha_0^{(k)}) \quad < \quad 0$$

Convergence of Learning Algorithm (cont.)

We only go into iteration step $(k+1)$ if we did a mistake in iteration k.
A misclassification implies:

$$\begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix}^T \begin{pmatrix} y_i \\ y_i \cdot \boldsymbol{x}_i \end{pmatrix} = y_i \cdot (\boldsymbol{x}_i^T \boldsymbol{\alpha}^{(k)} + \alpha_0^{(k)}) \quad < \quad 0$$

And thus we get

$$\left\| \begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix} \right\|_2^2 \leq \begin{pmatrix} \alpha_0^{(k-1)} \\ \boldsymbol{\alpha}^{(k-1)} \end{pmatrix}^T \begin{pmatrix} \alpha_0^{(k-1)} \\ \boldsymbol{\alpha}^{(k-1)} \end{pmatrix} + \begin{pmatrix} y_i \\ y_i \cdot \boldsymbol{x}_i \end{pmatrix}^T \begin{pmatrix} y_i \\ y_i \cdot \boldsymbol{x}_i \end{pmatrix} \leq k(1+M^2)$$

Convergence of Learning Algorithm (cont.)

Wrap-up:

$$k\rho \leq \begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix}^T \begin{pmatrix} \alpha_0^* \\ \boldsymbol{\alpha}^* \end{pmatrix} \leq \left\| \begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix} \right\|_2 \cdot \sqrt{\alpha_0^{*2} + 1}$$

Convergence of Learning Algorithm (cont.)

Wrap-up:

$$k\rho \leq \begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix}^T \begin{pmatrix} \alpha_0^{*} \\ \boldsymbol{\alpha}^{*} \end{pmatrix} \leq \left\| \begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix} \right\|_2 \cdot \sqrt{\alpha_0^{*2} + 1}$$

Using Cauchy-Schwartz:

$$k\rho \quad \leq \quad \left\| \begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix} \right\|_2 \cdot \sqrt{\alpha_0^{*2} + 1} \quad \leq \quad \sqrt{k(1 + M^2)(\alpha_0^{*2} + 1)}$$

Convergence of Learning Algorithm (cont.)

Wrap-up:

$$k\rho \leq \begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix}^T \begin{pmatrix} \alpha_0^{*} \\ \boldsymbol{\alpha}^{*} \end{pmatrix} \leq \left\| \begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix} \right\|_2 \cdot \sqrt{\alpha_0^{*2} + 1}$$

Using Cauchy-Schwartz:

$$k\rho \quad \leq \quad \left\| \begin{pmatrix} \alpha_0^{(k)} \\ \boldsymbol{\alpha}^{(k)} \end{pmatrix} \right\|_2 \cdot \sqrt{\alpha_0^{*2} + 1} \quad \leq \quad \sqrt{k(1 + M^2)(\alpha_0^{*2} + 1)}$$

shows:

$$k \quad \leq \quad \frac{(\alpha_0^{*2} + 1)(1 + M^2)}{\rho^2}$$

Lessons Learned

- Objective function changes in each iteration step.
- Optimization problem is discrete.
- Very simple learning rule.
- ⚠ Very important: Number of iterations does not depend on the dimension of the feature vectors.

Pattern Recognition Lab

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Next Time in Pattern Recognition

Further Readings

- Brian D. Ripley:
  Pattern Recognition and Neural Networks,
  Cambridge University Press, Cambridge, 1996.

- T. Hastie, R. Tibshirani, and J. Friedman:
  The Elements of Statistical Learning –
  Data Mining, Inference, and Prediction,
  2nd edition, Springer, New York, 2009.

Comprehensive Questions

- What is Rosenblatt's perceptron?
- What is the objective function for Rosenblatt's perceptron?
- Why is the optimization of the objective function nonlinear?
- When and how does Rosenblatt's perceptron algorithm converge?