University of Lübeck
Institute for Neuro- und Bioinformatics

Madany Mamlouk / Martinetz
Lübeck, 3/4 May 2018

Neuroinformatics (CS4405) • SS 2018

FCT 3

Place and Time: **Ü1**: Thursdays, 09:00-10:00, PC Pool 1+2 (building 64, floor)
Ü2: Thursdays, 10:00-11:00, PC Pool 1+2 (building 64, floor)
Ü3: Fridays, 14:00-15:00, PC Pool 1+2 (building 64, floor)
Website: `https://moodle.uni-luebeck.de/`

Perceptron Learning / Maximum Margin Classification

Exercise 3.1
Perceptron learning

Given L training samples $\vec{x}_i \in \mathbb{R}^N$ and its class labels $s_i \in \{1, -1\}$, we want to train a single artificial neuron, i.e. make it to automatically learn its weight vector $\vec{w} \in \mathbb{R}^N$ and its threshold $\theta \in \mathbb{R}$, such that

$$\sigma(\vec{w}^T \vec{x}_i - \theta) = s_i,\ \forall i = 1, \ldots, L \tag{1}$$

holds. The sigmoid function in (1) is defined as

$$\sigma(x) = \begin{cases} 1 & , \text{if } x \geq 0 \\ -1 & , \text{else} \end{cases}$$

In (1) weight vector $\vec{w}$ represents a normal vector of a linear hyperplane and threshold θ represents its (by $\|w\|$ scaled) distance to the origin.

To simplify learning, we apply the *threshold trick*, i.e. we extend the weight vector by an additional component that represents the threshold θ. The input vectors are likewise extended by a component with constant value -1. In this way, for an extended input $\vec{x} \in \mathbb{R}^N \times \{-1\}$ the output of the neuron can be written as

$$y = \sigma(\vec{w}^T \vec{x}) \ . \tag{2}$$

During each learning epoch the given L training samples are presented to the artificial neuron in random order. We use the perceptron learning rule to adapt weight vector $\vec{w}_t$ to $\vec{w}_{t+1}$

$$\vec{w}_{t+1} = \vec{w}_t + \varepsilon(s_i - y_i)\vec{x}_i \tag{3}$$

where

$$y_i = \sigma(\vec{w}_t \vec{x}_i) \ . \tag{4}$$

Here, $\varepsilon \in \mathbb{R}^+$ denotes the learning rate and $\vec{x}_i$ a randomly selected training sample.

Implement the learning rule (3) in MATLAB. On the exercise website you find a number of useful source code files to start with. Furthermore, there are two data files `ueb31.mat` and `ueb32.mat` on the website that contain two-dimensional training sets. Apply your perceptron implementation several times to the example training sets (start with $\varepsilon = 0.01$). Visualize the classification plane during the learning process.

Hints:

- In each iteration t of the perceptron learning process a randomly selected training sample $\vec{x}_i$ with class label s_i is classified according to (4). Then, the weight vector is modified according to (3).

- A single learning epoch might not be sufficient for obtaining a correct classification of all training samples. In this case, further learning epochs should be performed until a correct classification is obtained or a maximum number of learning epochs has been conducted.

Exercise 3.2
The DoubleMinOver learning rule

From the lecture, you know that the DoubleMinOver (DMO) learning rule can be used for maximum margin classification. The DMO algorithm is summarized below. Note in particular that $\vec{w} \in \mathbb{R}^N$ (i.e. the *threshold trick* is not applied) and that an explicit threshold θ is used that is computed after learning has been completed.

for $t = 1$ to $t_{\max}$ **do**

$$\vec{x}^{\min +} = \underset{\vec{x}_i \in X^+}{\arg\min}\, s_i \vec{w}^{\mathrm{T}} \vec{x}_i \qquad (X^+ = \{\vec{x}_i | s_i = 1\})$$

$$\vec{x}^{\min -} = \underset{\vec{x}_i \in X^-}{\arg\min}\, s_i \vec{w}^{\mathrm{T}} \vec{x}_i \qquad (X^- = \{\vec{x}_i | s_i = -1\})$$

$$\vec{w} = \vec{w} + \vec{x}^{\min +} - \vec{x}^{\min -}$$

end for

$$\theta = \frac{\vec{w}^{\mathrm{T}}\left(\vec{x}^{\min +} + \vec{x}^{\min -}\right)}{2}$$

- Implement the DoubleMinOver algorithm in MATLAB (use $t_{\max} = 100$). On the exercise website you will find a template source code file.
- Test your implementation on the 2-dimensional training data sets `ueb31.mat` and `ueb32.mat`.
- Compare your DMO learning results with your perceptron learning results. Run both learning algorithms several times. What differences do you observe in the behaviour of the two algorithms?

If you have any problems or questions, please contact us via mail or moodle.