Exercise 2-1  The ADALINE learning rule

The adaptive linear element (ADALINE) model uses the least mean square cost function

$$\text{cost} = \frac{1}{2} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2,$$

for $N$ training set elements, where $y_i$ is the actual and $\hat{y}_i$ the computed class label of pattern $i$. In contrast to the simple perceptron, classification is not realized by the signum-function. Instead, it is done directly: $\hat{y} = h$.

(As a reminder: $M$ is the number of input features of patterns $x_i \in \mathbb{R}^M$ and the dimensionality of the weight vector $w \in \mathbb{R}^M$, where $x_0 = 1$ is constant and corresponds to the bias or offset.)

a) Deduce the gradient descent-based learning rule (or: adaption rule) for the ADALINE process (analoguously to the perceptron learning rule).

b) Specify the corresponding sample-based learning rule.

c) What advantages do sample-based learning rules have?

d) Name the most distinctive characteristics between the ADALINE model and the perceptron model.

Exercise 2-2  Regularisation / Overfitting

a) What is overfitting and how does it occur?

b) How can a model be identified as “overfitted”?

c) How can overfitting be avoided?
Exercise 2-3  Basis Functions of Neural Networks

Given a test vector $x_i$, the output of a neural network is defined as

$$f(x_i) = \sum_{h=0}^{M-1} w_h \phi_h(x_i, v_h).$$

The weights of the neurons can be learned by employing the back-propagation rule with sample-based gradient descent. In the lecture neural networks with sigmoid neurons have been introduced, but it is possible to employ different basis functions:

a) Which properties do these basis functions have to fulfill?

b) Can a linear combination $\phi(x_i, v_h) = z_h = \sum_{j=0}^{M} v_{h,j} x_{i,j}$ be suitable for this?

c) Is the number of parameters for $\phi(x_i, v_h)$ limited? Could several different basis functions be used for the same neural network?

Exercise 2-4  Tensorflow Introduction

With the provided Jupyter Notebook, we want to give you a short introduction in Tensorflow and some insights in the basic usage of some common function calls.

Exercise 2-5  Train-Test Split and Regularization in Python

The provided Jupyter Notebook will guide you through the steps of implementing and applying Train-Test Splits, Cross Validation and Regularization in Python. Complete the specified tasks by filling in the missing code. You can further play around with varying parameters or trying out other models provided by sklearn.