**Institute for Computer Science VI, Autonomous Intelligent Systems, University of Bonn**

**Dr. N. Goerke**
**Friedrich-Hirzebruch-Allee 8, 53115 Bonn, Tel: +49 228 73-4167**
**E-Mail: goerke (at) ais.uni-bonn.de**
**http://www.ais.uni-bonn.de/WS2223/4204_L_NN.html**

# Exercises for module Technical Neural Networks (MA-INF 4204), WS22/23

# Assignments Sheet 5, due: Monday 21.11.2022

14.11.2022

| Group | Name | 27 | 28 | 29 | 30 | 31 | 32 | $\sum$ Sheet 5 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| | | | | | | | | |

## Assignment 27 (4 Points)

Compare the two neural network paradigms Multi-Layer-Perzeptron (MLP) and Radial-Basis-Function Network (RBF).
Please name explicitly **all** aspects that are common, and **all** the aspects that are different, write a brief statement explaining each difference.
Try to discuss or argue, what kind of applications are better suited for MLPs, and which are better for RBF-networks.

## Assignment 28 (2 Points)

Explain the three philosophies (including variants of them) to set, or adjust the centres $\mathbf{C}_k$ and sizes $s_k$ of RBF-networks.

## Assignment 29 (3 Points)

A N-K-M RBF Network (with fixed centers, and fixed width) has the task to implement a mapping from the $N$-dimensional input space, to an $M$-dimensional output space. A total of $P > K$ trainingspatterns $({}^p\mathbf{X}, {}^p\hat{\mathbf{Y}}({}^p\mathbf{X}))$ is available. To determine the weights $\mathbf{G}$ between RBF-layer and output layer the *Moore-Penrose-Pseudoinverse* shall be used.
Determine, as precise as possible, the total number of multiplications with respect to $P$, $N$, $K$ and $M$ that are necessary to calculate all weights $g_{k,m}$.
Remark: the $\mathcal{O}$-notation, with only the largest exponent, is not specific enough.

## Assignment 30 (2 Points)

Construct a 2-K-1 RBF-network that implements the Boolean function XOR. Please draw the network, label all parameters, and give the values of the centers $\mathbf{C}_k$, the width of the transfer functions $\sigma_k$ and the weights $g_{k,m}$ between RBF-layer and output neuron.

# Assignment 31 (2 Points)

Propose an extension of an RBF network approach from radial symmetry of the RBF-neuron characteristics, to a characteristic that have different extensions in the different axes of the input space (going from a radial to an elliptical characteristic).
Please describe what part of the RBF approach is to be changed.
Include formulas into your explanation.

# Assignment 32 (4 Points)

Calculate the partial derivatives of the global error $F$ of a N-K-M- RBF-Network with respect to the centers $\mathbf{C}_k = (c_{1k}, \ldots, c_{Nk})$ and with respect to the sizes $s_k$.

The RBF neurons have the Euclidean distance, and Gaussian bell function with size $s_k$ as transferfunction.

$$F = \frac{1}{2} \sum_{p=1}^{P} \sum_{m=1}^{M} \left( {}^{p}\hat{y}_m - {}^{p}y_m \right)^2 \qquad z_k({}^{p}\mathbf{X}) = exp\left( -\frac{\|{}^{p}\mathbf{X} - \mathbf{C}_k\|^2}{2s_k^2} \right)$$

# Programming assignment PA-D (10 Points, Due date: Mon 28.11.2022 )

Implement (in Python, on your own) a Radial-Basis-Function network (RBF-network), a method to adjust the positions $\mathbf{C}_k$ and widths $s_k$ of the centers, a method to adjust the weights $w_{km}$ and a program that demonstrates the functionality of your RBF network and the implemented methods.

**Radial-Basis-Function-Net, RBF-Net:**
The RBF-network shall have $N$ inputs, $K$ RBF-neurons, and $M$ output neurons. Take the Gaussian function (bell-curve) as radial-transferfunction for the RBF-neurons.

The weights $w_{km}$ between RBF-neurons and output neurons shall be initialized by random values from the interval of $-0.5$ to $+0.5$. The starting value(s) (SEED) for the pseudo-random generator(s) shall be set by you.

**Method to adjust the RBF-centers and RBF-widths:**
Implement a way to adjust the RBF-centers $\mathbf{C}_k$ and the RBF-widths $s_k$. Choose one of the methods that have been described in the lecture and tell which one you have chosen.

**Method to adjust the weights $W$ using gradient descent:**
Implement a gradient descent based (single-step or cumulative, your choice) method to train the weights $w_{km}$ between the RBF-neurons and the output neurons.
The learning rate $\eta$ shall be identical for all output neurons.
Generate your own training data, and/or use data from PA-A, PA-B, and PA-C. Read in the training data from `training_data.txt`, the test data from `test_data.txt`, and create a `learning_curve.txt` to monitor the learning progress.

**EXTRA: only for experienced students:**
Implement the method to directly calculate the weights $w_{km}$ between the RBF-neurons and the output neurons using the Moore-Penrose-Pseudo-Inverse; (you can use libraries for that). Compare the results with the gradient descent based results. Take data sets with $P = K$, one with $P$ slightly larger than $K$ and one with $P >> K$.
Discuss the obtained results w.r.t. achieved accuracy and the time to get the results.