# Neural Networks 2013/14, Second Exam (Hertentamen)

The problems are to be solved within 3 hours. **The use of supporting material (books, notes, calculators) is not allowed**. In total, you can achieve a maximum of **9 points**, the grade for the exam will be determined as "1.0 + number of points". Note that you will only obtain a valid grade if your practical reports are sufficient. **Note:** In analogy to the lecture notes, vectors are shown in bold here, as for instance: **a**. For the sake of clarity, you should use *arrows* to denote vectors, for example: $\vec{a}$.

## 1) The perceptron

Consider a data set $I\!D = \{\boldsymbol{\xi}^\mu, S_R^\mu\}_{\mu=1}^P$ with $N$-dimensional input vectors $\boldsymbol{\xi}^\mu \in I\!R^N$ and labels $S_R^\mu = \pm 1$

**a) (1pt)** Define precisely the following statements

a.1) $I\!D$ is homogenously linearly separable

a.2) $I\!D$ is inhomogeneously linearly separable

Also provide graphical illustrations for $N = 2$.

**b) (1pt)** Define and explain the *Rosenblatt* perceptron algorithm for a given set of examples $I\!D$. Be precise, for instance by writing it in a few lines of *pseudocode*. Also use mathematical definitions and equations where necessary, not just words. Suggest at least one possible intialization and one reasonable stopping criterion.

**c) (1pt)** While implementing the *Rosenblatt* algorithm, your partner in the practicals suggests to use a smaller (constant) learning rate. His/her argument: updating $\boldsymbol{w}$ by Hebbian terms of the form $\eta\, \boldsymbol{\xi}^\mu S_R^\mu$ with a small $\eta \ll 1$ should yield better convergence, since large changes of the weight vector are avoided. Do you agree or disagree? Provide precise arguments for your conclusion!

## 2) Learning a linearly separable rule (1pt)

Here we assume that the data set $I\!D = \{\boldsymbol{\xi}^\mu, S_R^\mu\}_{\mu=1}^P$ contains reliable examples for an unknown linearly separable function $S_R(\boldsymbol{\xi}) = \text{sign}(\boldsymbol{w}^* \cdot \boldsymbol{\xi})$ defined by a teacher vector $\boldsymbol{w}^* \in I\!R^N$ with $|\boldsymbol{w}^*| = 1$. Explain the term *version space*, provide a precise mathematical definition and also a graphical illustration. Explain why the perceptron of optimal stability can be expected to yield a student perceptron with good generalization behavior.

## 3) Gradient Descent

a) **(1.5pt)** Consider a feed-forward neural network with ($N$–2–1)–architecture and output

$$\sigma(\boldsymbol{\xi}) = \sum_{j=1}^{2} v_j \tanh[\, \boldsymbol{w}^{(j)} \cdot \boldsymbol{\xi}\,]$$

Here, $\boldsymbol{\xi}$ is an $N$-dim. input vector, $\boldsymbol{w}^{(j)} \in I\!\!R^N$ denotes the adaptive weight vector connecting the $j$-th hidden unit with the inputs. The quantities $v_1, v_2$ are the adaptive hidden-to-output weights.

Consider a single training example, i.e. input $\boldsymbol{\xi}^\mu \in I\!\!R^N$ and label $\tau^\mu \in I\!\!R$. For the quadratic error measure

$$e^\mu = \frac{1}{2}\left[\sigma(\boldsymbol{\xi}^\mu) - \tau^\mu\right]^2$$

derive the partial derivatives of $e^\mu$ with respect to <u>all</u> adaptive weights in the network. Hint (1): $\tanh'(x) = 1 - \tanh^2(x)$.

b) **(1pt)** For a given set of examples $I\!\!D = \{\xi^\mu, \tau^\mu\}_{\mu=1}^{P}$, consider *stochastic gradient descent* with respect to the cost function $E = \sum_{\mu=1}^{P} e^\mu$. Using the results from (a), specify the updates of weight vectors $\boldsymbol{w}^{(j)}$ and single weights $v_j$. You can write the former as a vectorial update for each $\boldsymbol{w}^{(j)}$ and the latter in separate equations. Be precise, include and explain parameter(s) that appear in the update equations.

## 4) Validation and Regularization

a) **(1pt)** Outline the basic idea of $k$-fold cross-validation. How can it be used to estimate the performance of a classifier with respect to novel data?

b) **(1pt)** Explain weight decay as a method of regularization in the context of (batch) gradient based learning. How can it be used to control overfitting effects in feedforward networks of non-linear units? Consider the minimization of a cost function $E(\boldsymbol{w})$ with weight vector $\boldsymbol{w} \in I\!\!R^N$ and gradient $\nabla_w E \in I\!\!R^N$. Provide the generic form of the update equation with weight decay, introduce and explain control parameter(s) if necessary. Re-write the update as a gradient descent for a modified cost function.

c) **(0.5pt)** In the lectures we discussed the so-called *tiling-algorithm* which adds hidden units to a feed-forward neural network until a given set of examples is classified correctly (you do not have to describe the algorithm). Explain in words why this approach might be problematic in terms of the generalization ability of the trained network.