# Neural Networks 2014/15
# Practical Assignment I: Perceptron training

---

The topic of this assignment is the Rosenblatt perceptron algorithm. We apply it to randomized data and try to observe our theoretical findings (capacity of a hyperplane) in computer experiments. The code (or large parts thereof) will be re-used in the next assignment(s).

---

## 1) Rosenblatt's perceptron algorithm

Write a program (Matlab preferred but not mandatory) which can be used to

- ... generate artificial data sets $I\!D = \{\boldsymbol{\xi}^\mu, S^\mu\}_{\mu=1}^{P}$ where the $\boldsymbol{\xi}^\mu \in I\!R^N$ are vectors of independent random components $\xi_j^\mu$ with mean *zero* and variance *one*. You can use, for instance, Gaussian components $\xi_j^\mu \sim \mathcal{N}(0,1)$ (matlab: `randn`). The labels $S^\mu$ are taken to be independent random numbers $S^\mu = \pm 1$ with equal probability 1/2.

- ... implement sequential perceptron training by repeated presentation of the $P$ examples: At *time step* $t = 1, 2, \ldots$ present example $\mu(t) = 1, 2, \ldots, P, 1, 2, \ldots$.

  Use nested loops where the inner one runs from 1 to $P$ and the outer loop counts the number $n$ of *sweeps* through the data set $I\!D$. Limit the number of *sweeps* to $n \leq n_{max}$, i.e. the total number of single learning steps will be at most $n_{max} \cdot P$.

- ... run the Rosenblatt algorithm for a given data set $I\!D$:

$$\boldsymbol{w}(t+1) = \begin{cases} \boldsymbol{w}(t) + \frac{1}{N}\, \boldsymbol{\xi}^{\mu(t)}\, S^{\mu(t)} & \text{if } E^{\mu(t)} \leq 0 \\ \boldsymbol{w}(t) & \text{else} \end{cases}$$

  where $E^{\mu(t)} = \boldsymbol{w}(t) \cdot \boldsymbol{\xi}^{\mu(t)}\, S^{\mu(t)}$. Initialize the weights as $\boldsymbol{w}(0) = 0$, but make sure that a training step is indeed performed for $E^{\mu(t)} = 0$.

  The training should be performed until either a solution with all $E^\nu > 0$ is found (counted as a *success*) or until the maximum number of *sweeps* $n_{max}$ is reached.

a) Perform the Rosenblatt perceptron training for a number of randomized data sets of the same size. Each training process should be performed until either a solution with all $E^\nu > 0$ is found (counted as a *success*) or until the maximum number of *sweeps* $n_{max}$ is reached.

   For a given value of $P$, use a number $n_D$ of independently generated sets $I\!D$. Determine the fraction $Q_{l.s.}$ of successful runs as a function of $\alpha = P/N$, by repeating the experiment for different values of $P$. The result should resemble the probability $P_{l.s.}(\alpha)$ that was derived in class, see lecture notes.

   **Hint:** A reasonable set of parameters could be $N = 25, P = \alpha N$ with $\alpha = 0.75, 1.0, 1.25, \ldots 3.0$, $n_D \sim 50$, $n_{max} \sim 100$. Your actual choices will of course depend on your implementation, available computing power, and your patience. If (CPU-) time allows, improve the quality of your results by setting $N, n_D$ and $n_{max}$ as large as possible.

b) Observe the behavior of $Q_{l.s.}$ for different system sizes $N$ as well. Does it approach a step function with increasing $N$? To this end, repeat the above experiments for, say, $N = 50$ and $N = 100$.

The report should contain a very brief description of what precisely you did, please do not include actual code. You should submit plots representing your results for parts (a) and (b) together with a brief discussion thereof. Compare your findings with the theoretical result discsuued in class, explain/speculate what causes the differences etc.