Neural Networks (2014/15)
Practical Assignment II: Learning a rule

The topic of this assignment is the learning of a linearly separable rule from example data. Hence, we define outputs $S^\mu = \pm 1$ which are defined by a *teacher perceptron*. The resulting data set is guaranteed to be linearly separable, and training by storage is a reasonable strategy in the absence of noise in the data set.

Learning a linearly separable rule

Consider a set of random input vectors as in assigment (I) with similar dimensions N. However, here we consider training labels S^μ which are defined as

$$S^\mu \;=\; \text{sign}(\mathbf{w}^* \cdot \boldsymbol{\xi}^\mu)$$

by a *teacher perceptron*. You can consider a randomly drawn $\mathbf{w}^*$ with $|\mathbf{w}^*|^2 = N$. Note that you could also consider, without loss of generality (why?), $\mathbf{w}^* = (1, 1, \ldots, 1)^\top$. Also, modify your code from assignment (I) so that it ...

a) ... implements the sequential Minover algorithm:
at each time step t, determine all stabilities

$$\kappa^\nu(t) = \frac{\mathbf{w}(t) \cdot \boldsymbol{\xi}^\nu S^\nu}{|\mathbf{w}(t)|} \quad \text{for all examples } \nu$$

and identify the example $\mu(t)$ of minimal stability $\kappa^{\mu(t)} = \min_\nu \{\kappa^\nu(t)\}$.
(In case of a *tie*, it does not matter which example is chosen).
With this example, perform a Hebbian update step

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \frac{1}{N}\, \boldsymbol{\xi}^{\mu(t)}\, S^{\mu(t)}$$

and go to the next time step. In contrast to the Rosenblatt algorithm, the sequence of examples is not fixed and in each step a non-zero update is performed. Note that MinOver should not be stopped when $\{E^\nu > 0\}_{\nu=1}^P$, because the stability will increase further. Run the algorithms until the weight vector does not change anymore over a number of, say, P single training steps according to some reasonable criterion or until $t_{max} = n_{max} P$ single training steps have been performed in total. (Obviously, the total number of single steps should still be proportional to P, although the training is not done in *sweeps* anymore.) The final weight vector for a given set of data should approximate the perceptron of optimal stability $\mathbf{w}_{max}$.

Please include the main piece of code in the report, i.e. the actual realization of the MinOver learning step. Do not include the entire program.

b) ... determines the so-called *learning curve*, i.e. the generalization error

$$\epsilon_g(t) = \frac{1}{\pi} \arccos\left(\frac{\mathbf{w}(t) \cdot \mathbf{w}^*}{|\mathbf{w}(t)|\,|\mathbf{w}^*|}\right)$$

as a function of the size of the training set, i.e. as a function of $\alpha = P/N$. Obtain the result as an average over $n_D \geq 10$ randomized data sets as in (I).

Consider a somewhat larger range of α then in assignment (I), e.g. $\alpha = 0.1, 0.2, \ldots, 5.0, \ldots$. The range and number of different values of α depends, of course, on your patience, on available CPU power, and your implementation. Provide results in terms of a graph for, at least $\alpha = 0.25, 0.5, 0.75, \ldots 3.0$.

Hints:
(1) It is important to make sure that t_{max} is large enough for the stabilities to converge or at least get close to optimal stability.
(2) The division by $\mid \mathbf{w} \mid$ is an important part of the definition of κ^μ. However, if you compare different κ^ν for the <u>same</u> given weight vector, i.e. when identifying the minimum, you can of course drop it. In other words: for one given $\mathbf{w}$, the minimum of the E^ν identifies the relevant example. For the $\kappa(\alpha)$ plot use, of course, the correct κ!

Suggestions for bonus problems:

- Repeat the above experiments for the simpler Rosenblatt Perceptron and compare the learning curves $\epsilon_g(\alpha)$. Can you confirm that maximum stability yields better generalization behavior?
- Repeat n_D MinOver experiments for, say, $\alpha = 1.0$. Determine in each run the perceptron of optimal stability, the example stabilities $\{\kappa^\mu\}$, and the embedding strengths $\{x^\mu\}$. Plot histograms of the observed κ^μ and x^μ values, respectively.
- Consider the learning from noisy examples by replacing the true labels in the data set by

$$S^\mu = \begin{cases} +\text{sign}\left(\mathbf{w}^* \cdot \boldsymbol{\xi}^\mu\right) & \text{with probability } 1-\lambda \\ -\text{sign}\left(\mathbf{w}^* \cdot \boldsymbol{\xi}^\mu\right) & \text{with probability } \lambda\ . \end{cases}$$

  Here $0 < \lambda < 0.5$ controlls the noise level in the training data. Does the student perceptron still approach the correct lin. sep. rule $\mathbf{w}^*$ for $\alpha \to \infty$? Can you observe significant differences between the generalization behavior of the MinOver and Rosenblatt algorithms?