Neural Networks and Computational Intelligence
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 learning in version space 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 $|\mathbf{w}^*|^2 = N$. Also, modify your code from assignment (I) so that it ...

- ... implements the sequential Minover algorithm:
  at each time step t, determine the stabilities

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

  and identify the example $\mu(t)$ that has currently the minimal stability $\kappa^{\mu(t)} = \min_\nu \{\kappa^\nu(t)\}$. Of course, if several stabilities are negative, the one with the largest absolute value is the minimum! 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 stop when $\{E^\nu > 0\}_{\nu=1}^{P}$, because the stability will increase further. Run the algorithm until the weight vector does not change anymore over a number of P single training steps according to some reasonable criterion or until $t_{max} = n_{max} \cdot P$ single training steps have been performed in total. The final weight vector $\mathbf{w}(t_{max})$ with stability $\kappa(t_{max})$ for a given set of data should approximate the perceptron of optimal stability $\mathbf{w}_{max}$. Initialize the weight vector as $\text{w}(t) = 0$ in each training process.

  Please include the main piece of code in the report, i.e. the actual realization of the Minover learning step. In addition submit the full code by e-mail.

- ... determine the generalization error (at the end of the training process)

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

  By repeating the training for different P, determine the so-called *learning curve*, i.e. $\epsilon_g(t_{max})$ as a function of $\alpha = P/N$. Obtain the result as an average over $n_D \geq 10$ randomized data sets per value of P.

Consider a somewhat larger range of α than 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, available CPU power, and efficiency of your implementation. Provide results (a graph) for at least the 12 equidistant values $\alpha = 0.25, 0.5, 0.75, \ldots 3.0$.

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 $|\mathbf{w}|$ is an important part of the definition of κ^μ. However, if you compare different κ^ν for the same given weight vector, i.e. when identifying the minimum, you can of course drop it. In other words: for a given $\mathbf{w}$, the minimum of the E^ν identifies the relevant example.

Suggestions for bonus problems:

- Note that you can also set the teacher to $\mathbf{w}^* = (1, 1, \ldots, 1)^\top$ without loss of generality. Argue why this is not a restriction or *special case* in the context of the practical.
- Determine $\kappa(t_{max})$ as a function of α for random labels. Compare with the results for labels given by a teacher perceptron.
- Implement the AdaTron algorithm (in terms of embedding strengths) and compare the convergence behavior / speed with the Minover algorithm.
- 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?
- Consider the learning from noisy examples by replacing the true labels in the data set by

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

  Here $0 < \lambda < 0.5$ controls 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?
- For non-separable data (e.g. with noisy labels as suggested above) implement the "large margin with error" version of the AdaTron algorithm and perform a similar set of experiments.