Neural Networks and Computational Intelligence
Practical Assignment III: Learning by gradient descent

You should either hand in a report for assignment II **or** assignment III. If you hand in assignment II **and** assignment III, report III will be completely disregarded and not graded.

Stochastic gradient descent
The aim of this problem is to get acquainted with gradient descent based training of layered networks and to do some *hands on* experiments. Take the actual assignment as a starting point for further exploration and self-study, see also the suggestions for bonus problems.
We consider a shallow feedforward neural network with real-valued output as our *student network:*

$$\sigma(\boldsymbol{\xi}) = \sum_{k=1}^{K} v_k \tanh[\boldsymbol{w}_k \cdot \boldsymbol{\xi}]$$

where $\boldsymbol{\xi} \in I\!R^N$ represents an input vector and $\boldsymbol{w}_k$ is the N-dim. vector of adaptive weights connecting the input with hidden unit k. The linear hidden-to-output relation is given by a weighted sum of the hidden states.
For the core problem of this assignment, we restrict ourselves to the special case of a so-called *soft committee machine* with fixed second layer weights $v_k = 1$.

a) Stochastic gradient descent
Formulate and implement a stochastic gradient descent procedure w.r.t. the weight vectors $\boldsymbol{w}_k$, $k = 1, 2, \ldots K$, aiming at the minimization of the cost function

$$E = \frac{1}{P} \sum_{\mu=1}^{P} \frac{1}{2} \left(\sigma(\boldsymbol{\xi}^\mu) - \tau(\boldsymbol{\xi}^\mu) \right)^2 = \frac{1}{P} \sum_{\mu=1}^{P} e^\mu \tag{1}$$

for a given data set $I\!D = \{\boldsymbol{\xi}^\mu, \tau(\boldsymbol{\xi}^\mu)\}_{\mu=1}^{P}$ with continuous training labels $\tau(\boldsymbol{\xi}^\mu) \in I\!R$.
For the core problem of the assignment, you can restrict yourself to $K = 2$ and fixed weights $v_k = 1$.
In each learning step, select one of the P examples, say $\boldsymbol{\xi}^\nu$, randomly with equal probability $1/P$ and use the gradient with respect to its contribution $e^\nu = (\sigma(\boldsymbol{\xi}^\nu) - \tau(\boldsymbol{\xi}^\nu))^2/2$, only (stochastic gradient descent):

$$\begin{aligned} \boldsymbol{w}_1 &\leftarrow \boldsymbol{w}_1 - \eta \nabla_1 e^\nu \\ \boldsymbol{w}_2 &\leftarrow \boldsymbol{w}_2 - \eta \nabla_2 e^\nu \end{aligned}$$

where ∇_j denotes the gradient with respect to $\boldsymbol{w}_j$.
Perform $t_{max} \cdot P$ single training steps, where $t \leq t_{max}$ measures the *training time* in units of P single example updates. Initialize the weights as independent random vectors with $|\boldsymbol{w}_1|^2 = 1$ and $|\boldsymbol{w}_2|^2 = 1$. For comparability of results between different reports, use a constant learning rate $\eta = 0.05$. In addition, you may want to explore different values or time dependent η. In any case, hand in the results for constant $\eta = 0.05$.

b) A regression problem
In Brightspace you will find the file `data3.mat` which you can import into Matlab. In addition, the files `xi.csv, tau.csv` and `*.txt` versions are provided for other platforms and languages. The files comprise a 50×5000-dim. array `xi` corresponding to 5000 input vectors (dimension

$N = 50$) and a 5000-dim. vector `tau` corresponding to the target values. Alternatively, the data is provided as *comma separated values* in the files `tau.csv` and `xi.csv`, which can be imported in other programming environments. Note that the data set (or a subset thereof) is not necessarily learnable for the considered student network.
Consider (at least) the first $P = 100$ examples as the training set. In the course of stochastic gradient descent training, measure the cost function E, Eq. (1), and plot it vs. the training time t as defined above.
In addition, evaluate the quantity

$$E_{test} = \frac{1}{Q} \sum_{\rho=P+1}^{P+Q} \frac{1}{2}\Big(\sigma(\boldsymbol{\xi}^\rho) - \tau(\boldsymbol{\xi}^\rho)\Big)^2 \tag{2}$$

which corresponds to the *test* or *generalization error* in terms of the quadratic deviation from the target function for Q test examples. You should set $Q = 100$ or larger.
Plot and compare the evolution of E and E_{test} with the training time t. You should consider large enough training times t_{max} after which the errors seem to decrease no further (apart from random fluctuations). At the end of the training process, display the obtained, final weight vectors, for instance as bar graphs.

Don't include the source code in the report, it should be submitted through Brightspace. Follow the guidelines for reports and hand in **at least** the following:

- A brief introduction of the problem, including the full update equations and their derivation according to the stochastic gradient descent
- The curves $E(t)$ and $E_{test}(t)$ as explained above
- Graphs displaying the two final weight vectors $\mathbf{w}_{1,2}$ at the end of training. Use symbols or bars to represent the weight vector components.
- A discussion/interpretation of your findings

Remarks and hints:

- You can follow the supplementary material (grad-example.pdf) provided with the assignment in Brightspace. Note, however, that the hidden-to-output weights are assumed to be fixed in the core problem.
- You could also consider the concatenated vector of all weights $\underline{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2]$ and consider the gradient with respect to $\underline{W}$, but since $\nabla_{\underline{W}} = [\nabla_1, \nabla_2]$ this is completely equivalent to treating $\mathbf{w}_{1,2}$ individually.
- Compute E and E_{test} after P single randomized steps, not after each individual update. It is recommended to define a function which calculates E and E_{test} with $\boldsymbol{w}_1, \boldsymbol{w}_2$ and the corresponding data set (inputs and labels) as arguments.
- Of course, your results will be more reliable if you repeat the training process over several runs from random initializations and take an average of $E(t)$ and $E_{test}(t)$ over these runs. However, this is not obligatory and depends on your patience and available CPU time.

Possible bonus problems

- consider smaller and larger values of P, e.g. a selection from $P = 20, 50, 200, 500, 1000, 2000$ for the training process. How do the final training and test errors depend on P? Make sure that the Q test examples are never used in the training.
- study systematically the influence of the learning rate η. Potentially consider a time dependent rate as discussed in class.
- can you observe plateau states in the learning curves? If so, display the corresponding weight vectors and compare with the final ones after leaving the plateau.
- consider student networks with $K > 2$ hidden units and/or adaptive hidden-to-output weights v_k, derive the corresponding gradients and include them into the stochastic descent. You could perform additional experiments with the `data3` set or study the real world data set as suggested below.

- **Real world data: Body fat estimation problem**
  A data set relating to a *real world* regression problem is provided and described in Brightspace ("Assignments" folder). Use the data to train a soft committee with $K > 2$, with fixed $v_k = 1/K$ or adaptive v_k as a possible extension. You could split the data randomly into, say, 200 training samples and 52 examples for the estimation of the generalization error.