Neural networks
Statistics, Optimisation, and Learning
Examples Sheet 2

1. One-dimensional Kohonen network. Construct a one-dimensional Kohonen network to learn the properties of a geometric distribution of two-dimensional points that are uniform inside the area coloured black in Fig. **??**, and zero outside that area. For the input data, generate 1000 two-dimensional points $(x_1, x_2)^\mathsf{T}$ uniformly distributed inside the black area shown in Fig. **??**. Use 100 output neurons in the one-dimensional Kohonen network. Perform learning without normalising the data. Use the ordering phase and the convergence phase. In the ordering phase use a time-dependent Gaussian width $\sigma(t)$ of the neighbouring function, as well as a time-dependent learning rate $\eta(t)$:

$$\sigma(t) = \sigma_0 \exp(-\frac{t}{\tau_\sigma}) \ , \ \eta(t) = \eta_0 \exp(-\frac{t}{\tau_\sigma}) \ . \tag{1}$$

Parameter values: the total learning time $T_{\rm order}$ in the ordering phase $T_{\rm order} = 10^3$ updates, $\sigma_0 = 100$, $\eta_0 = 0.1$, $\tau_\sigma = 300$. Initialise weights to random numbers uniformly distributed between -1 and 1.

In the convergence phase, the learning rate $\eta_{\rm conv}$, and the Gaussian width $\sigma_{\rm conv}$ of the neighbouring function are kept constant and small. Parameter values: $\sigma_{\rm conv} = 0.9$, $\eta_{\rm conv} = 0.01$, the total learning time $T_{\rm conv}$ in the convergence phase $T_{\rm conv} = 2 \cdot 10^4$ updates.

a. Perform unsupervised learning as explained above. Plot the weight vectors corresponding to the 100 output neurons that you obtain after the ordering phase. In a separate panel of the same figure plot the corresponding weight vectors obtained after the convergence phase. In both cases denote also the desired input pattern. Discuss: does the network learn the properties of the distribution of the input data? What are the differences between the ordering and convergence phase? **(1p)**

b. Repeat task **1a** but now with $\sigma_0 = 5$. Plot the weights obtained after the ordering and after the convergence phase. Compare to the results obtained in **1a**. Discuss. **(1p)**

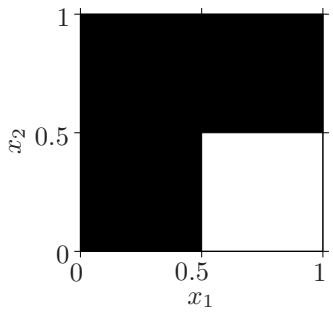

Figure 1: Task 1. The pattern coloured black depicts the area inside which points are uniformly distributed. The properties of this distribution are to be learned by the Kohonen network, as explained in task 1.

2. Unsupervised Oja's learning with one linear unit. Oja's learning rule adjusts weights $\boldsymbol{w} = (w_1, \ldots, w_N)^\mathsf{T}$ according to

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \delta\boldsymbol{w}, \text{ where } \delta\boldsymbol{w} = \eta\zeta(\boldsymbol{\xi} - \zeta\boldsymbol{w}) \ . \tag{2}$$

Here $\eta > 0$ is the learning rate, $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_N)^\mathsf{T}$ is an input pattern chosen uniformly at random out of the p input patterns, and

$$\zeta = \sum_{i=1}^{N} w_i \xi_i \tag{3}$$

is the output of the network corresponding to the input pattern $\boldsymbol{\xi}$. Apply this learning rule to the data set 'data_ex2_task2_2017.txt' that you find on the course web page. The data set consists of 401 input patterns that are two-dimensional (the individual components of each pattern are in the first and the second column in the data file). Tasks:

First, apply Oja's learning rule to this data set without centring its components to zero mean. Second, apply Oja's rule to the data set centred so that both components of the input data have zero mean. In both cases, use the following: $\eta = 0.001$, number of updates $2 * 10^4$. Initialise weights w_i to random numbers uniformly distributed between -1 and 1.

Plot the modulus of the weight vector ($|\boldsymbol{w}|$) as a function of time for the two cases analysed (two panels in the same figure). Make two additional panels in the same figure. In the first (second) panel of these two additional panels, display the data used as an input for learning in the first (second) case, together with the corresponding weight vectors obtained at the end of the corresponding learning. Compare the results obtained in the two cases and discuss. Do the weight vectors converge? If yes, is there a difference between the weight vectors to which the network converges in the two cases? Why? Explain by referring to the three properties of the weight vector to which the network is expected to converge (Lecture notes, p151). In which of the two cases does one expect the network to find the maximal principal-component direction of the input data? Compute the maximal principal-component direction of the input data and compare to the weight vectors learned by your network in the two analysed cases. (**1p**)

3. Unsupervised simple competitive learning combined with supervised simple perceptron.

Download the data set 'data_ex2_task3_2017.txt' from the course home page. It contains 2000 input patterns $\boldsymbol{x} = (x_1, x_2)^\mathsf{T}$ that are two-dimensional. For each input pattern, x_1 and x_2 are given by the entry in the second, and the third column, respectively. The classification (class -1 or 1) of a given pattern is given by the corresponding entry in the first column. Your task is to classify this data set using a combination of simple competitive learning and supervised simple perceptron learning as follows.

- Perform unsupervised simple competitive learning on the given data set (without normalising the input data) using k Gaussian neurons. Values of k are stated below. Use the following learning scheme. In

each iteration, choose randomly one of the input patterns, $\boldsymbol{x}$. Compute the activation $g_j(\boldsymbol{x})$ for each of the network neurons (denoted by $j = 1, \ldots, k$) using

$$g_j(\boldsymbol{x}) = \frac{\exp\left[-\|\boldsymbol{x} - \boldsymbol{w}_j\|^2/2\right]}{\sum_i \exp[-\|\boldsymbol{x} - \boldsymbol{w}_i\|^2/2]} \ . \tag{4}$$

Here $\|\cdots\|$ denotes the Euclidean distance. Determine the winning neuron for the pattern $\boldsymbol{x}$ by finding the neuron i_0 such that $g_{i_0}(\boldsymbol{x}) \geq g_j(\boldsymbol{x})$ for all $j = 1, \ldots, k$. Update the weight vector $\boldsymbol{w}_{i_0}$ assigned to this neuron using

$$\delta \boldsymbol{w}_{i_0} = \eta(\boldsymbol{x} - \boldsymbol{w}_{i_0}) \ . \tag{5}$$

The weights assigned to the neurons other than the winning neuron remain unchanged in this iteration. Parameter values: constant learning rate $\eta = 0.02$, number of updates 10^5. Initialise all weights to random numbers uniformly distributed between -1 and 1.

- After the unsupervised simple competitive learning, perform supervised simple perceptron learning. The inputs to the simple-perceptron network are the outputs $g_j(\boldsymbol{x})$ that you obtain after the competitive learning has been completed. Thus, the number of input units in your perceptron network is equal to the number of Gaussian neurons k. The target output for each pattern is given in the first column of the data set you downloaded. Using the gradient-descent algorithm train the perceptron to recognise the classes in the data set. Use $\tanh(\beta b)$ activation function. Parameter values: $\beta = 1/2$, constant learning rate $\eta = 0.1$, 3000 training steps. Initialise weights and thresholds to random numbers uniformly distributed between -1 and 1.

a. Set the number of Gaussian neurons for the unsupervised simple competitive learning to 4. Perform the algorithm 20 times. Compute the average classification error at the end of learning (the average is over the different runs you made). From the experiments you made, choose one with the smallest classification error. Illustrate the decision boundary by making an image with pixel color determined by the output value of the supervised simple perceptron network. Illustrate the decision boundary together with the input data. Also depict the weights of the Gaussian neurons that you obtained after the unsupervised simple competitive learning. Discuss. **(1p)**

b. Repeat the tasks from **3a** but now using $k = 10$ Gaussian neurons. Determine the average classification error at the end of learning. Compare to the results in **3a**. Among the experiments you made with $k = 10$, choose one with the smallest classification error. Illustrate the decision boundary by making an image with pixel color determined by the output value of the supervised simple perceptron network. Illustrate the decision boundary together with the input data. Also depict the weights of the Gaussian neurons that you obtained after the unsupervised simple competitive learning. Compare to the results in **3a**. Discuss. **(1p)**

c. Compare the learning performance with $k = 1, 2, \ldots, 10$ Gaussian neurons (the results for 4 and 10 neurons are readily available from tasks **3a** and **3b**). For each k, perform 20 independent learning runs, and compute the classification error that you obtain at the end of each learning run. Plot the average classification error obtained at the end of learning as a function of the number of Gaussian neurons. Discuss. **(1p)**