Neural networks
Statistics, Optimisation, and Learning
Examples sheet 1

1. Deterministic Hopfield model. Write a computer program implementing the Hopfield model with synchronous updating according to

$$S_i \leftarrow \mathrm{sgn}\left(\sum_{j=1}^{N} w_{ij} S_j\right)$$

where S_i is the state of neuron i, N is the number of neurons, and w_{ij} are the synaptic weights given by Hebb's rule for all i and j. Generate and store p random patterns.

1a. Following the calculation on p. 18-22 in Lecture notes, derive an approximation for the one-step error probability P_{Error} valid for large p and N. Take into account that $w_{ii} \neq 0$. (**1p**)

1b. Check the approximation derived in **1a** as follows. Set in your computer program the number of neurons to $N = 200$, and use the following values of p: 1, 20, 40, 60, 80, 100, 120, 140, 160, 180, 200, 220, 240, 260, 280, 300, 320, 340, 360, 380, 400. For each value of p, estimate the one-step error probability P_{Error} using 10^5 independent bits. Plot your numerically determined P_{Error} as a function of p/N. Plot also the corresponding theoretical curve that you derived in **1a**. Discuss your result. (**1p**)

2. Stochastic Hopfield model: phase diagram. Write computer program implementing the Hopfield model (take $w_{ii} = 0$) with asynchronous stochastic updating.

2a. Set the number of neurons to $N = 200$ and store $p = 5$ random patterns, denoted by $\boldsymbol{\zeta}^{(i)}$ $(i = 1, \ldots, p)$. For noise parameter $\beta = 2$ feed the stored pattern $\boldsymbol{\zeta}^{(1)}$ to the network and monitor the corresponding order parameter m_1 as a function of time (one unit of time corresponds to one update). Repeat this experiment 20 times. For each experiment, plot the resulting m_1 as a function of time (make one plot with 20 curves). Describe the results. Discuss: How long time does it take for the network to reach the steady state? Based on m_1 that you obtain in the different experiments, explain whether or not the network performs well. Is your finding consistent with the phase diagram illustrated on page 60 in Lecture notes? Note that the phase diagram on page 60 in Lecture notes is valid in the limit of $N \to \infty$, whereas in your simulation N is finite. Discuss the effect of finite N on the order parameter m_1. (**1p**)

2b. Repeat the task **2a** but now for $p = 40$ (keeping all other parameters the same as in **2a**). Plot m_1 as a function of time for the different experiments made with $p = 40$. Describe the results and compare to the results in **2a**. Is the order parameter closer to unity in this case or in the case analysed in

2a? Why? Explain by referring to the phase diagram illustrated on page 60 in Lecture notes. (**1p**)

3. Back propagation. The task is to write a computer program solving a classification problem using back-propagation. The goal is to achieve a classification error that is as small as possible for a given validation set by training the network on the corresponding training set. The classification error $C_{\rm v}$ for the validation set is defined as

$$C_{\rm v} = \frac{1}{2\,p} \sum_{\mu=1}^{p} |\zeta^{(\mu)} - {\rm sgn}(O^{(\mu)})|$$

where $|x|$ stands for the absolute value of x, p denotes the total number of patterns in the validation set, $\zeta^{(\mu)}$ is the target output for pattern μ, and $O^{(\mu)}$ is the network output for pattern μ. The classification error for the training set is defined similarly. Implement your program on the data set available on the course web page: 'train_data_2017.txt' containing training data, and 'valid_data_2017.txt' containing validation data. Patterns are two-dimensional (first two columns in the data files). Each row corresponds to a different pattern. The third column contains target outputs. To improve the training performance, normalise the input data so that both the first and the second column have zero mean and unit variance. Tasks:

3a. Train the network without hidden layers. Use asynchronous back-propagation with the following parameters: learning rate $\eta = 0.02$, $\beta = 1/2$, weights initialised randomly from the uniform distribution in the interval $[-0.2, 0.2]$, biases initialised randomly from the uniform distribution in the interval $[-1, 1]$. Take activation functions to be $g(b) = \tanh(\beta b)$ (Lecture notes, p. 103). Iterate for 10^6 iterations (one iteration corresponds to feeding one randomly chosen pattern to the network). Perform ten independent training runs. During each training, plot the energy of the training set as well as of the validation set as a function of the training time. Discuss the trends observed. Explain: how is the energy expected to change over time according to the gradient-descent rule? What is the average and the minimum value of the classification error obtained at the end of training for the training set and for the validation set (averaging is over the different experiments made)? Quote also the variance of the classification error at the end of training for the training set and for the validation set? (**1p**)

3b. Now train a network with one hidden layer that has four neurons. As in task **3a**, perform ten independent runs. For each run, plot the energy in the training set as a function of the training time, as well as the energy of the validation set as a function of the training time. What is the average and the minimum value of the classification error obtained at the end of training for the training set and for the validation set (averaging is over the different experiments made)? Quote also the variance of the classification error at the end of training for the training set and for the validation set? Discuss the results in comparison to those obtained in **3a**. Is the performance better? Why? (*Hint: use a graphical illustration of the classification problem to support your answer.*) (**1p**)