University of Helsinki
Department of Mathematics and Statistics
Neuronetworks February 2014
The Last Exercises

Deadline for submission is **Thu 20.03.2014, 23:59 EECT**. Send the solutions to me via e-mail vadim.kulikov@iki.fi.

Use these either to collect the remaining points to get 12 or to get an extra ECTS by collecting 18 points all together. Only those who have already at least 4 points from the course can participate.

In brackets you see the number of points you can get for each exercise.

Hopfield networks

1 Exercise. (1p) Consider Hopfield networks with possible neuronal states 1 and -1. Compute the weight matrix of a Hopfield network which stores the two vectors $\bar{x}_1 = [1, -1, 1, -1, 1, 1]$ and $\bar{x}_2 = [-1, 1, 1, -1, -1, 1]$ (using Hebbian learning, see Section 13.3.3).

Confirm via a simple calculation that in this network both x_1 and x_2 are stable states.

2 Exercise. (1p) Let $n > 3$. Consider a Hopfield network with n units. Consider a state of this network in which any three units are in state 1 and the others are in state 0. There are $\binom{n}{3}$ such states. Provide the weights and thresholds for this network which make these states the only stable states of the network (along with the proofs that this is the case).

Hint: mimic 13.5.2.

3 Exercise. (1p) Suppose we have a Hopfield network (possible states of units: 1 and -1) with weight matrix

$$\begin{bmatrix} 0 & -0.3 & 0.3 & 0 & 0.3 \\ -0.3 & 0 & -0.3 & 0.3 & 0 \\ 0.3 & -0.3 & 0 & -0.3 & 0 \\ 0 & 0.3 & -0.3 & 0 & -0.3 \\ 0.3 & 0 & 0 & -0.3 & 0 \end{bmatrix}$$

and thresholds 0.2 everywhere. Starting from the state $[1, -1, -1, 1, 1, 1]$ describe an asynchronous flow of the network until it reaches a stable state. One path is enough.

4 Exercise. (1p) For the network of previous exercise draw a flow chart similar to the one given in the book on page 351. Similarly as in the book, ignore the updates that do not change the state of the network.

5 Exercise. (2p) Find out about some application of Hopfield networks in scientific literature and write an essay of around one printed page describing what you've found.

Synchronisation

6 Exercise. (1p) Show that if $f\colon [0,1] \to [0,1]$ with $f(0) = 0$ and $f(1) = 1$ satisfies $f''(x) > 0$ for all $x \in]0,1[$, then there exists a unique $x_0 \in [0,1]$ such that $f(x_0) = 1$.

7 Exercise. (2p) Suppose $f\colon [0,1] \to [0,1]$ is twice differentiable and convex (e.g. $f''(x) > 0$ for all x). Then by the previous exercise there is exactly one $u_0 \in [0,1]$ such that $f'(u_0) = 1$. Show that if $f(u_0) < 1 - u_0$, then the sequence

$$g(x), g(g(x)), \ldots$$

converges either to 0 or to 1 for all $x \in [0,1]$, except possibly one exception, where $g(x) = f(1 - f(1 - x))$.

Hint: Denote $h(x) = 1 - f(1 - x)$ while h is concave ($h'' < 0$) and consider the behaviour of the derivative of g which is $f'(h(x))h'(x)$. Some more hint can be gained from the material on synchronisation on the course web-page. This was done in the last lecture.

8 Exercise. (2p) Generalise the synchronisation analysis given in the lecture (see lecture notes at the course web-page and the previous two exercises) to three or more neurons.

Perceptrons and logic

9 Exercise. (2p) Consider the propositional sentence $s = (p_0 \vee \neg p_1) \wedge (p_0 \vee p_1 \vee \neg p_2)$. Design a neural network of perceptrons which has 3 input units (corresponding to p_0, p_1, p_2) and one output unit. The network has to be such that $\Phi(p_0, p_1, p_2) = 1$ if and only if $s(p_0, p_1, p_2)$ is true where $p_0 \in \{0,1\}$, 0 means false, 1 means true and Φ is the function computed by the network.

10 Exercise. (2p) This is a continuation to the previous exercise, but can be done independently. Describe a rule by which one can construct a neural network of perceptrons for a given propositional sentence s of three variables such that if the function computed by the network Φ, outputs 0 if the sentence is false and outputs 1 if it is true for the corresponding truth distribution of the propositional variables.

11 Exercise. (1p) Given that you have a solution to the previous exercise show that the learning problem is NP-complete. Hint: Use the fact that 3SAT is NP-complete; page 278.

Backpropagation

12 Exercise. (4p) In this exercise you will write a backpropagation algorithm which computes a pseudoinverse of a matrix.

Let $n, m > 0$ and X be an $m \times n$-matrix. The *pseudoinverse* of X is an $n \times m$ matrix W which minimizes the square norms

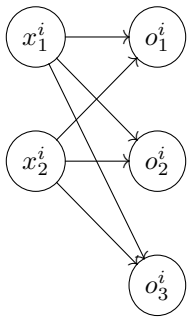
$$\|XW - I_m\|^2 \quad \text{and} \quad \|WX - I_n\|^2$$

where I_k is the $k \times k$-identity matrix for $k > 0$. The pseudoinverse of X always exists and is unique and is denoted by X^+. If X is an invertible $n \times n$ matrix, then $X^+ = X^{-1}$ and if Y is any $n \times k$ matrix, then $W = X^+Y$ minimizes the square norm

$$\|XW - Y\|^2.$$

Thus pseudoinverses are useful when approximate solutions of non-solvable linear equations are needed.

A pseudoinverse can be calculated using a backpropagation algorithm. Let us consider the case when $n = 2$ and $m = 3$. X is a 3×2 matrix and we want to find a 2×3 matrix W such that $W = X^+$. Consider the following network:



The units on the left are input units and they output x_1^i, where $X^i = [x_1^i, x_2^i]$ is the i:th row of X, $i \in \{1, 2, 3\}$. The units on the right are linear, i.e. the output is simply the weighted sum of all the inputs.

If W is the weight matrix of this network then the output of the network is $\bar{o}^i = [o_1^i, o_2^i, o_3^i] = X^iW$. Now our intended output is $[\delta_{i1}, \delta_{i2}, \delta_{i3}]$ where δ_{ij} is the Kronecker delta. because we want the output, XW, to be the identity matrix. The error term is defined by

$$E = \frac{1}{2}\|XW - I\|^2.$$

By definition, $E = \sum_{i=1}^{3}\sum_{j=1}^{2} E_{ij}$ where

$$E_{ij} = \frac{1}{2}|o_j^i - \delta_{ij}|^2$$

By backpropagting we can calculate the derivative of E with respect to w_{ij} (Hint: the answer is $x_j^i(o_j^i - \delta_{ij})$. After we have all the partial derivatives we update the weight w_{ij} by

$$-\gamma\frac{\partial E}{\partial w_{ij}}$$

where γ is the learning constant, so the whole weight matrix is modified in the direction opposite to the gradient of E w.r.t. (w_{ij}).

In short, for each i we update $W := W - \gamma X^T(O - I)$ where $O = [o_j^i]$ is the matrix of outputs, the i:th row being $[o_1^i, o_2^i, o_3^i]$.

Your task is to implement this algorithm which takes a matrix as an input and converges to its pseudoinverse matrix. I expect your algorithm to work for any matrix of dimensions $m \times n$ where m and n are below 10. The learning constant γ and the number of interations usually depend on m and n. In your case the range is $m, n < 10$. and a good value for γ is 0.2 and the number of iterations to be between 100 and 200.

Other

13 Exercise. (2p) If you didn't do Exercise 3.6., you can do it now: find out about an application of Kohonen self-organising maps and write an essay about your findings of length about 1 page prined text.