Computational Cognitive Neuroscience
Exercise Set 2

Due: 04.02.2016, before class

Guidelines for Submission

- Submissions can be made via email to hande.celikkanat@helsinki.fi before the deadline, or by delivering a hard copy at the beginning of the class.
- In case you are submitting both written exercises and programming exercises, please prepare two separate files respectively for submission.
- Please do submit programming exercises only via email, and as python scripts with .py extension. Also please kindly indicate with a comment which exercise you are attempting before related code.
- Maximum number of points you can get for exercises per week is 10. Maximum number of points you can get in total is 20. One exercise is worth 1 point – unless stated otherwise in the exercise.

1 Rojas Ch. 3

1. Construct a (single-unit) perceptron which implements the NAND logic gate: the input is a binary vector of length 2 and the output is 1 if and only if the input is not equal to [1 1].
2. Construct a network of perceptrons (or a multilayer perceptron) which implements the XOR logic gate: the input is a binary vector of length 2 and the output is 1 if and only if the values of the input vector are unequal.
3. Show that a (single-unit) perceptron cannot detect parity. More precisely, if a perceptron receives a binary vector of length $n \geq 2$, then it is not the case that it outputs 1 if and only if the number of 1's in the input vector is even.
4. Construct a multilayer perceptron with binary inputs of size 8 which detects parity: the (one-dimensional) output is 1 if and only if the number of 1's in the input is even.
5. The following matrix represents an 8×8 image:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

What will it look like if the following detector (perceptron) is applied to each pixel which is not on the edge (internal pixels):

$$\begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}$$

The output should be a 6×6 image/matrix.

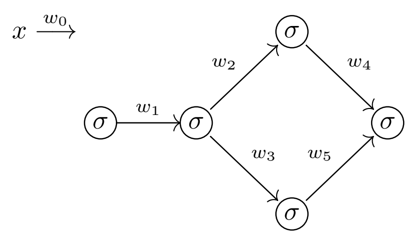
6. Write a program which applies the edge detector from the previous exercise to a given grayscale image.
7. Write a program which detects edges from a given greyscale image.

2 Rojas Ch. 4

1. Show that the sets $\{(0,1),(1,0)\} \subset \mathbb{R}^2$ and $\{(0,0),(1,1)\} \subset \mathbb{R}^2$ are not linearly separable.
2. Show that the sets $\{(x, e^x) \mid x \in \mathbb{R}\}$ and $\{(e^x, x) \mid x \in \mathbb{R}\}$ are linearly separable.
3. Rojas page 83 last four lines: Let A be an $(n+1) \times n$-matrix with last row having only 1 as entriest. Show that $\{\mathbf{w} \mid \mathbf{Aw} > 0\}$ is convex. A set S is convex if for every two points $\mathbf{x}, \mathbf{y} \in S$, the line segment joining $\mathbf{x}$ and $\mathbf{y}$ is a subset of S.
4. Implement the perceptron learning algorithm (page 85) for 2-dimensional inputs. Train it to become AND logic gate.
5. (Continuation of the previous exercise) Generate random sets P and N of vectors so that $P \subset [0,1] \times [0,1]$ and $N \subset [0,1] \times [-1,0]$. Use your learning algorithm to separate them. Try different sizes of P and N: from 2 to 100 and compare how much time does it take for the algorithm to converge.

3 Rojas Ch. 7

1. Consider the following network:



where σ is some differentiable function applied to the weighted sum of the inputs.

(a) What is the function calculated by the network, i.e. write the output of the last neuron in terms of f_i's, w_i's and x.

(b) Denote that function by $\Phi(x, w_0, w_1, w_2, w_3)$. What is the derivative of Φ with respect to w_2?

(c) What is derivative of Φ with respect to x?

2. Consider the network of the previous exercise. Suppose that given input 5 we would like to have output 1. The error is defined by

$$E = \frac{1}{2}|\Phi(5, \bar{w}) - 1|^2.$$

Write down the derivative of E with respect to w_2.

3. (If done well, worth of 2 exercises) Implement a one-layer backpropagation with input n-dimensional and output m-dimensional *without non-linearities*, i.e. the network is equivalent to multiplying the input $\bar{x}$ by the connection matrix $\mathbf{W}$. Use the standard error function $\|\mathbf{x}\mathbf{W} - \mathbf{y}\|^2$.

   This algorithm can be used to compute a pseudoinverse matrix: If the training data is given as a matrix X (a single row is one input) and the number of rows is m and the expected answer for i:th input is a vector with 1 at the i:th coordinate and 0 everywhere else, then the optimal weight matrix is the pseudoinverse of X. Thus your backpropagation algorithm in fact computes the pseudoinverse.

   Hints: Rojas 7.3.3 is useful. For $n, m < 10$ a good learning constant is $\gamma = 0.2$ and number of iterations around 150.