Neural Networks
Lab assignment 2
The Threshold Logical Unit

General instructions

Print your report and put it in box number 2 near 216 in the Bernoulliborg. Also send the pdf including the code by e-mail to neuralnets19+LAB_2@gmail.com. The assignment is due on May 2^{nd} at 11:00. You should e-mail just one archive file containing at least your code and your report, so the easiest way of doing this is just to archive all of your files at once. Name it as follows:
s1234567_DonaldTrump_s7654321_SantaClaus.zip.
The maximum length of the report for this lab is **5 pages**. This limit of 5 pages does not include the inline code that you submit.

The aim of this lab

In this assignment we will attempt to translate a biologic neuron into an artificial neuron. We will try to understand and implement the perceptron learning rule and look into its application.

1 Introduction

In the previous assignment we briefly covered the biologic neuron. This week we will focus on a simple artificial neuron: the threshold logic unit (TLU). In the TLU the all-or-nothing characteristics of the biologic neuron are represented by a boolean output: 0 or 1.

Chapters 2, 3, and 4 of the book cover the TLU in detail, so it is good practice to study these chapters before doing this assignment. The original paper by Rosenblatt about the perceptron can be found in the folder of this lab assignment on Nestor, which you can study yourself for some interesting background theory. Particularly, the part about the initial use of the perceptron is intriguing.

TLU or Perceptron

The difference between a TLU and a Perceptron is rather obscure. Both concepts tend to be confused with each other. In some texts they are distinguished based on the input that is provided to the unit. The TLU generally receives simple, logical input, whereas a perceptron receives input from specific filters. For this course you can consider both terms equivalent.

Additional Information

For this assignment you can receive 8 points. If you receive p points, your lab grade is given by grade $= p \cdot \frac{10}{8}$, and it will be averaged with the homework grade for this week.

Use the scheduled lab sessions to write and run your code. The theory questions can be answered later. If you think you might need help with the questions about linear algebra, please make them during the lab sessions.

2 Implementing a TLU in Matlab (3 pt.)

We are now going to implement a TLU using two inputs and a single output that can learn to perform logical operations such as an AND or an OR function. Download the file `tlu.m` which provides an unfinished piece of code for a TLU. Chapter 4 of the book covers this step-by-step.

a) First write your name(s) as comments at the top of the file.

b) Define a nullary (0 arguments) function named `examples` (in file examples.m) that returns a matrix with 4 rows and 2 columns. Make sure the matrix contains the four possibilities for two logic inputs.

c) Define a second nullary function `goalAND` that returns a column vector that gives the desired output corresponding to the inputs of `examples`. First we will try to make the TLU learn an AND-function. Think about which output corresponds to each input example.

d) Implement nullary functions `initW` and `initB` that return respectively the initial weight matrix and the initial threshold, both with random values between 0 and 1.

e) Implement ternary function `summedInput` which takes as arguments an input vector (x), the weight matrix (W), and the bias (b), and returns the weighted sum of the current input (x). The bias should be also applied to the weighted sum.

f) Implement the unary threshold-function (or the step-function) and name it `stepA`. Do notice that since the bias is applied in summedInput, the threshold is just 0. Make sure `output` obtains the correct value.

g) Implement the binary function `pError` that given (in this order) a target (t) and an output (o) returns the network's error for this a pattern (test case). Please do notice that this error is the one used in the learning rule.

h) Implement ternary function `deltaW`, that given (in this order) a learning rate, an error (for a specific pattern) and an input vector, returns the delta for the weight matrix.

i) Implement binary function `deltaB`, that given (in this order) a learning rate and an error (for a specific pattern), returns the delta for the bias.

l) Implement binary functions `upW` and `upB` that, given (in this order) the previous values (of weights and bias respectively) and the deltas (of weights and bias respectively) returns the updated weights and threshold (respectively).

m) Finally put everything together in a function `netStep` that given (in order) a learning rate, a current weight matrix, a current bias, an input vector, a target (for that input) will return three values (in order) which are the updated weights, the updated bias, the epoch's error. Plese do notice that you should also set the values of weights, threshold, examples, and goal in the tlu.m file (lines 9, 12, 15, and 16)

3 Experimenting with a TLU (3 pt.)

The changes of the weights and the error over the process of learning are stored and plotted after the number of specified epochs have passed. Run your code multiple times and answer the following questions:

a) The error does not decrease each epoch. Why?

b) Why are we interested in the summed squared error $\sum_{p=0}^{N}(t^{(p)} - y^{(p)})^2$ instead of simply summing the errors $\sum_{p=0}^{N}(t^{(p)} - y^{(p)})$?

c) Why is the number of epochs required to reach an error of 0 not always the same?

d) Increase the learning rate from 0.1 to 0.6. What do you observe? Is a higher learning rate better? Explain you answer.

e) The input is 0 or 1. Change this to 0.1 or 0.9. Is the TLU still capable of learning the AND-function? What happens if the input is set to 0.2 or 0.8? Why?

f) Which important feature of artificial neural networks did we encounter in (e)?

g) Change the vector `goal` such that the TLU learns a NAND-function (NOT-AND). Does this work too? What happens to the weight values? Explain why the threshold has become negative by drawing a geometrical sketch.

4 XOR-rule (1 pt.)

Change the goal vector such that the TLU learns a XOR-function. This should be done by implementing the function `goalXOR` which returns the goal vector for this operator. Run the program a couple of times. What do you observe? Why does this happen?

5 Code extensions (1 pt.)

a) The TLU learns for a fixed number of epochs, which is rather inefficient. Change the code such that the TLU stops learning after all examples are classified correctly. Make sure the plots and the axes of the graphs are displayed correctly. (0.25 pt.)

b) By using matrix-vector multiplication we can replace the inner `for`-loop. This is called batch-learning. We train the system by using all examples at once. Implement this learning method. (0.75 pt.)

What to hand in

You have to hand in the following:

1. Your code. Follow the guidelines on Nestor!
2. A report with your answers to the remaining questions and an explanation of your code. Try to write clear and concise and try to avoid answering in telegram style.
3. Also take note of any additional guidelines on Nestor.