Neural Networks Lab 1

Threshold Logical Unit

1 The aim of this lab

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

2 Introduction

In the previous assignment we briefly covered a 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 0 or 1.

Chapters 2, 3 and 4 of the book cover the TLU in detail, so it is a 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.

2.1 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.

3 Lab assignments

For this assignment you can receive 12 points (bonus excluded). By making the bonus assignment you can score an additional 1.0 points. If you receive p points, your grade is given by grade $= \min(p * \frac{9}{12} + 1, 10)$.

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.

3.1 Linear Algebra (2 pt.)

This section tests your linear algebra skills. The basic understanding of vectors, inner products and vector projections is essential to understanding neural networks. This assignment should not require more than 15 minutes. If it does require more, it is a good idea to (re)study some material on vectors. Take another look at section 3.2 of the book for example.

1. Draw a coordinate system on a piece of grid paper. Write 0 at the origin and depict the value +1 on both axes.
2. We consider two vectors [1]: $\boldsymbol{x}, \boldsymbol{y} \in \mathrm{R}^2$

$$\boldsymbol{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \boldsymbol{y} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

[1] Comlumn vectors will be represented using bold font (LaTeX: `$\boldsymbol x$`) $\boldsymbol{x}$. Row vectors will be represented by adding an arrow at the top (LaTeX: `$\vec{ \boldsymbol x}$`): $\vec{\boldsymbol{x}}$.

3. Determine the length of $\boldsymbol{x}$ algebraically, denote this as $\|\boldsymbol{x}\|$ from hereon.

4. Determine the inner product of $\boldsymbol{x}^T\boldsymbol{y} = \boldsymbol{x} \cdot \boldsymbol{y}$ algebraically.

5. Draw both vectors as arrows from the origin in your coordinate system. Use different colors or write the names of the vectors near the arrows. How can you check your answer for the previous question by looking at the geometry?

6. Let $\boldsymbol{w}$ be the weight vector. Draw $\boldsymbol{w}$ in the coordinate plane:

$$\boldsymbol{w} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

7. Draw a dotted line from the right end of $\boldsymbol{x}$ to the line $\boldsymbol{w}$ such that it is perpendicular to $\boldsymbol{w}$. The distance from the origin to the intersection of the dotted line and $\boldsymbol{w}$ is called the *projection* of $\boldsymbol{x}$ on $\boldsymbol{w}$. We write this as $\boldsymbol{x_w}$.

8. Let ϕ be the angle between $\boldsymbol{x}$ and $\boldsymbol{w}$. Show that: $\cos(\phi) = \dfrac{\boldsymbol{x_w}}{\|\boldsymbol{x}\|}$.

9. Given the following relation: $\boldsymbol{x} \cdot \boldsymbol{w} = \|\boldsymbol{x}\|\|\boldsymbol{w}\|\cos(\phi)$, show that $\boldsymbol{x} \cdot \boldsymbol{w} = \boldsymbol{x_w}\|\boldsymbol{w}\|$.

10. Determine $\|\boldsymbol{w}\|$. Suppose that the threshold $\theta = 0.6$. Let $\boldsymbol{a}$ be an arbitrary vector. When does $\boldsymbol{a}$ satisfy $(\boldsymbol{a} \cdot \boldsymbol{w}) - \theta > 0$?

11. The plane is split in two by $\boldsymbol{w}$ and the threshold θ. Draw the separating line.

3.2 Theory questions (2 pt.)

1. What is an action potential and how is it generated in a biologic neuron?

2. What is hyperpolarization?

3. What is a PSP and how is a PSP represented in an artificial neuron?

4. Which feature do the step function and the action potential in a biologic neuron have in common?

3.3 A TLU on paper (1 pt.)

1. Draw a simple artificial neuron with 2 inputs. Which 3 elements can you distinguish other than the input and output?

2. Which parts of the biologic neuron correspond to these 3 elements?

3. If you were to program a class `Neuron` yourself using object oriented programming, which variables would it have and which functions (or methods) would the class contain?

4 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 matrix `examples` with 4 rows and 2 columns. Make sure the matrix contains the four possibilities for two logic inputs.

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

d) Initialize the weights and the threshold randomly between 0 and 1.

e) Compute the weighted sum of the current input and assign it to `summed_input`.

f) Implement the threshold-function (or the step-function). Make sure `output` obtains the correct value.

g) Compute the neuron's error.

h) Compute the delta for the weights and for the new threshold.

i) Update the weights and the threshold.

5 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?

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.

6 XOR-rule (1 pt.)

Change the goal vector such that the TLU learns a XOR-function. Run the program a couple of times. What do you observe? Why does this happen?

7 Bonus questions

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 correct. (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.)

8 What to hand in

You have to hand in the following:

1. Your solutions to the linear algebra questions. You can scan your answers on paper to a pdf.
2. Your code. Follow the guidelines on Nestor!
3. 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.
4. Also take note of any additional guidelines on Nestor.