

# INF3490 Mandatory Assignment 2: Multilayer Perceptron

Paul Wieland

Deadline: Tuesday, October 16th, 2018 23:59:00

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Task . . . . .	2
1.2	Training Data . . . . .	3
<b>2</b>	<b>Implementation</b>	<b>3</b>
2.1	Initialization . . . . .	3
2.1.1	Dimension of the weight matrix . . . . .	3

# 1 Introduction

## 1.1 Task

We will build a Multilayer Perceptron to steer a robotic prosthetic hand. There are 40 inputs of electromyographic signals that we will classify. There are 8 classification values corresponding to a different hand motion:



Figure 1: Possible motions <sup>1</sup>

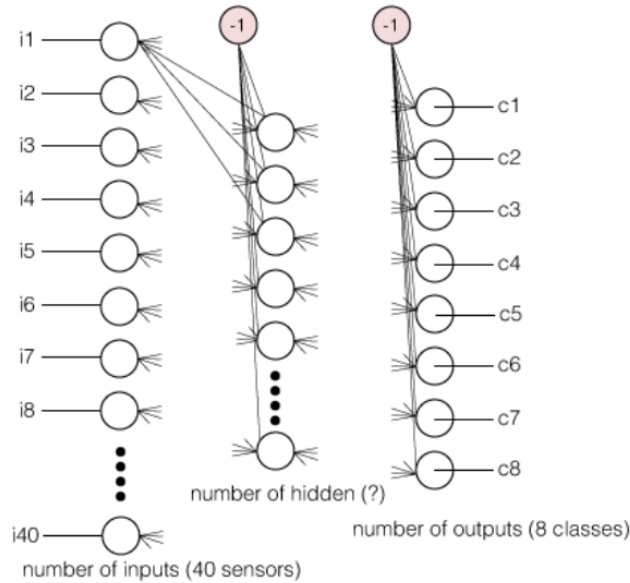


Figure 2: Multilayer Perceptron for our problem <sup>2</sup>

We build a Multilayer Perceptron with 40 entry nodes, that means one node for each input. Then there is one hidden layer with a various number of hidden nodes. For classifying the input, there are 8 output nodes corresponding to the 8 hand motions. We only use one hidden layer to solve this problem.

<sup>1</sup><http://folk.uio.no/kyrrehg/pf/papers/glette-ahs08.pdf>

<sup>2</sup>[https://www.uio.no/studier/emner/matnat/ifi/INF3490/h18/assignments/assignment-2/assignment\\_2.pdf](https://www.uio.no/studier/emner/matnat/ifi/INF3490/h18/assignments/assignment-2/assignment_2.pdf)

## 1.2 Training Data

For each input vector:

$$input = [i_1, i_2, i_3, i_4, \dots, i_{40}], i_n \in \mathbb{R}, n \in [40] \quad (1)$$

we have a target output vector:

$$output = [c_1, c_2, c_3, c_4, \dots, c_8], c_n \in \{0, 1\}, n \in [8], \sum_{n=1}^8 c_n = 1 \quad (2)$$

That means, forwarding the input should result in the given target vector.

## 2 Implementation

The file *mlp.py* contains the class *mlp*. There are 5 functions that i will explain in detail.

### 2.1 Initialization

The function `__init__(self, inputs, targets, nhidden)` has three important parameter that we need to initialize the Multilayer Perceptron.

As the input data is given as a vector, it is a good choice to create two 2D-Array for the two weight layers. As the parameters *inputs* and *targets* have the type `<class 'numpy.ndarray'>`, it is a good idea to work only with numpy arrays.

#### 2.1.1 Dimension of the weight matrix

- `weight_matrix_1`:  
The input vector in (1) has of course a size of 40 ( $len(inputs[n]), n \in [len(inputs) - 1] \cup \{0\}$ ). But we need to add the *bias\_value -1* that can be seen in Figure 2. That means:

$$weight\_matrix\_1 \in \mathbb{R}^{41 \times nhidden} \quad (3)$$

- `weight_matrix_2`:  
There are *nhidden* hidden nodes and 8 ( $len(targets[n]), n \in [len(targets) - 1] \cup \{0\}$ ) exit nodes. So we also need to take into account the *bias\_value -1*. That means:

$$weight\_matrix\_2 \in \mathbb{R}^{(nhidden+1) \times 8} \quad (4)$$

Both, *weight\_matrix\_1* and *weight\_matrix\_2*, will be initialized randomly.