INF3490
Mandatory assignment 2

Before you begin the exercise, review the rules at `http://www.mn.uio.no/ifi/english/studies/admin/mandatory-assignments/index.html`.

This exercise will be graded PASS/FAIL.

Implement all the assignements below and produce a report of your work, **including answers to all the questions below**, as a PDF file. Submit the PDF and your code to us at `https://devilry.ifi.uio.no/` by the end of **October 31st**.

Preface

The purpose of this assignment is to give you some experience applying supervised learning with a multilayer perceptron (MLP). In order to control a robotic prosthetic hand, Prosthetic hand controllers (PHCs) reads the electromyographic signals generated by contracting muscles in the under arm. The idea is that we can, through supervised learning, get a PHC to learn which hand movement the user of a robotic prosthetic hand wants to perform.

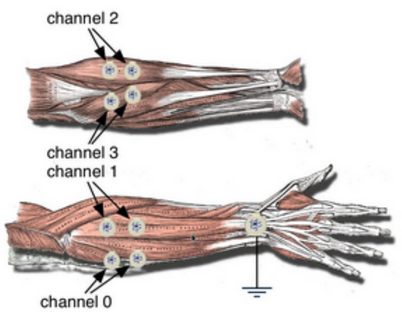

Fig. 1. Sensor placement (muscle anatomy taken from [21]).

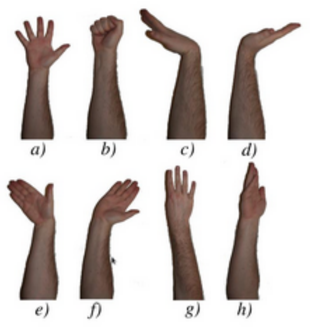

Fig. 2. Motion classes: *a)* open, *b)* close, *c)* flexion, *d)* extension, *e)* ulnar deviation, *f)* radial deviation, *g)* pronation and *h)* supination.

The data

The data you will be classifying are electromyographic (EMG) signals corresponding to various hand motions, collected over three days (one dataset per day).[1] Each row of the data has 40 features (4 sensors, 10 readings each) and one classification value (1-8, corresponding to the 8 hand motions.[2]). You can get the data from `http://www.uio.no/studier/emner/matnat/ifi/INF3490/h14/oblig-2/inf3490-mandatory-assignment-2-data.zip`.

Getting started

You are free to use a programming language of your own choice, but we highly recommend using Python. The precode is only available in Python (the files *movements.py* and *mlp.py* that are bundled with the data). If you choose another language, you will have to implement the everything yourself. Along with the two precode files, you are also given four files with data:

- *movements_day1.dat*
- *movements_day2.dat*
- *movements_day3.dat*
- *movements_day1-3.dat*

The file *movements.py* will read one specified data file, and split it into training, validation and test sets. Then it will create an MLP, run the training on the MLP and call the confusion method of the MLP to print the confusion matrix. A class mlp with four methods is given in the file *mlp.py*:

- *earlystopping*
- *train*
- *forward*
- *confusion*

[1] See section 2 of K. Glette, J. Torresen, Thiemo Gruber, Bernhard Sick, Paul Kaufmann, and Marco Platzner. *Comparing Evolvable Hardware to Conventional Classifiers for Electromyographic Prosthetic Hand Control.* (In Proceedings of the 2008 NASA /ESA Conference on Adaptive Hardware and Systems(AHS-2008), Noordwijk, The Netherlands, IEEE Computer Society, 2008, or `http://folk.uio.no/kyrrehg/pf/papers/glette-ahs08.pdf`) if you are curious about how the data was collected.

[2] Which, for the curious, are: open, close, flexion, extension, ulnar deviation, radial deviation, pronation, and supination

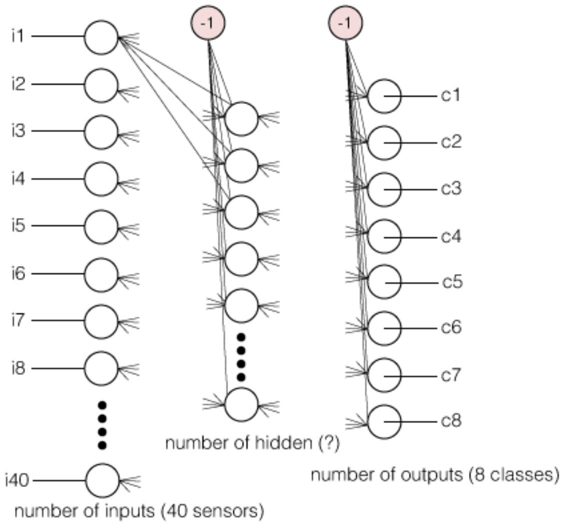

Figure 1: The neural net with one hidden layer

earlystopping is the method that starts the training of the network and keeps track of when to stop the training. It should, in each iteration, call MLPs train method and run the network forward. Before the network learns the training-data too well, we stop training to avoid overfitting. See Fig. 3.14 on page 69, S. Marsland.

train trains the network. This means running the backwards phase of the training algorithm to adjust the weights. See section 3.2.1 on page 54-55, S. Marsland.

forward run the network forward. When running the network forward, a perfect classifier should always have the highest value on the correct output. You will experience that this is not the case the first time(s) the network is run forward, which is why you are adjusting the weights in train.

confusion prints a confusion matrix. Run the network forward with the test set, and fill out the outputs in a matrix. The matrix should be a $c \times c$ matrix (where c is the number of classes), and have the correct classes on one axis and classified classes on the other axis. Print also the percentage of correct classes.

How the algorithm runs

After the input is given to the MLP, we start to train the network. Instead of running the training a fixed number of iterations, we split it up into parts, where we decide whether to stop in between the iterations. This means that we will first check if the error of the network with a validation set in *earlystopping*, and if our stopping criteria is not met, we start to train the network.

The training phase consists of the first running the network forwards, then running it backwards, adjust the weights and repeat for as many iterations as we have chosen (10,100,1000). When training is done we return to *earlystopping* to check if we're done.

Once we're done we print out the confusion table based on how the network classified the data in the test set.

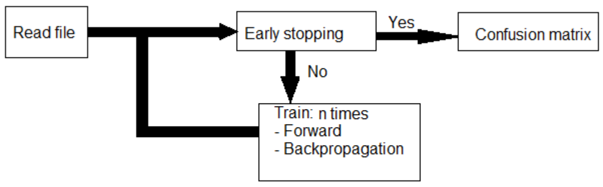

Figure 2: A diagram of how it the algorithm runs

What to do

All students

- Implement the neural net as explained above. Don't use more than one hidden layer (see Fig. 1).
- Run the algorithm on dataset *movements_day1-3.dat*. You should test with at least three different number of hidden nodes (e.g 6, 8, 12). Report your findings and provide the resulting confusion tables with percentages, one for each net with different number of hidden nodes. How many nodes do you find sufficient for the network to classify well?
- By only looking at your reported confusion tables, which classes where likely to me mistaken for each other?

INF4490

- Implement k-fold cross-validation. Choose a suitable k. Report the correct percentages for each fold, along with the average and standard deviation. Please do not include all confusion tables in your report.