INF3490/INF4490 Mandatory Assignment 2: Multilayer Perceptron

Eivind Samuelsen, Ole Herman S. Elgesem

October 13, 2016

Rules

This is an individual assignment. You are not allowed to deliver together or copy/share source-code/answers with others. Before you begin the exercise, review the rules at this website:

`http://www.uio.no/english/studies/admin/compulsory-activities/mn-ifi-mandatory.html`

The assignment can be changed/updated after release (to clarify or correct mistakes). It is advised to always use the latest version, available **here**.

Delivering

Deadline: 2016-10-31 23:59:00. All deliveries must be made through **Devilry**.

What to deliver?

Deliver one single zipped folder (.zip, .tgz or .tar.gz) which includes:

- PDF report containing:
  - Your name and username (!)
  - Instructions on how to run your program.
  - Answers to all questions from assignment.
  - *Brief* explanation of what you've done.
- Source code
- Data file (so the program can run *right away*)
- Any files needed for the group teacher to easily run your program on IFI linux machines.

Important: if you weren't able to finish the assignment, use the PDF report to elaborate on what you've tried and what problems you encountered. Students who have made an effort and attempted all parts of the assignment will get a second chance even if they fail initially. This exercise will be graded PASS/FAIL.

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.

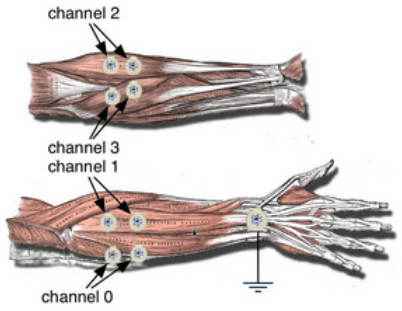

Fig. 1. Sensor placement (muscle anatomy taken from [21]).

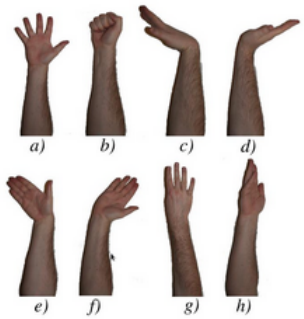

Fig. 2. Motion classes: *a)* open, *b)* close, *c)* flexion, *d)* extension, *e)* ulnar deviation, *f)* radial deviation, *g)* pronation and *h)* supination.

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**(.zip or .tgz) from the **semester page**.

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 everything yourself. Along with the two precode files, you are also given four files with data:

- *data/movements_ day1.dat*
- *data/movements_ day2.dat*
- *data/movements_ day3.dat*
- *data/movements_ day1-3.dat*

As the last file contains data for all days, you don't *have to* use the other files. You can use the *movements_ day1-3.dat* file and split up the data into different sets.

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

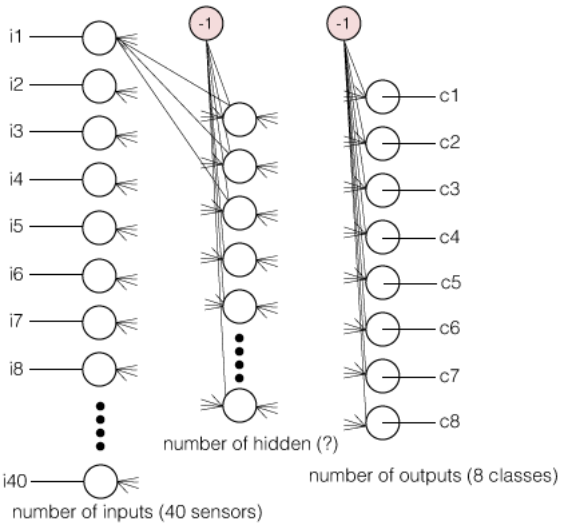

Figure 1: The neural net with one hidden layer

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 trainingdata too well, we stop training to avoid overfitting. See Fig. 4.11 on page 88, S. Marsland.

train trains the network. This means running the backwards phase of the training algorithm to adjust the weights. See section 4.2.1 on page 77-79, 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.

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

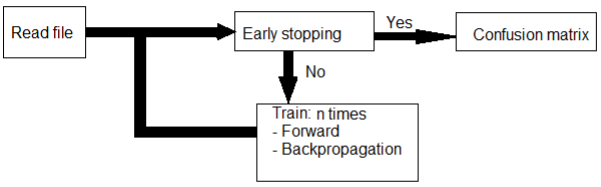

Figure 2: A diagram of how it the algorithm runs

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.

Contact and Github

Corrections of grammar, language, notation or suggestions for improving this material are appreciated. E-mail me at **olehelg@uio.no** or use **GitHub** to submit an issue or create a pull request. The **GitHub repository** contains all source code for assignments, exercises, solutions, examples etc. As many people have been involved with writing and updating the course material, they are not all listed as authors here. For a more complete list of authors and contributors see the **README**.