INF4490 - Assignment 2

Torstein Gombos,
Tagombos, t.a.b.gombos@fys.uio.no

October 2018

1 Introduction

In this assignment we are to implement a multi-layer perceptron network using iterative methods in python. The algorithm is implemented in the mlp class in the mlp.py script. When the netowrk trains, it checks an early stopping condition to see if the netowrk is overfitting. There is also a method which implements k-fold cross validation.

2 Code instructions

There are two versions of the code to run.

1. Trains a model until earlystopping is activated. Afterwards, a test set is sent through the model. Confusion matrix and resulting accuracy is then printed.
2. Use k-fold cross validation by taking the entire dataset and divide it into k number of equally large groups and train k times while the dataset.

The methods are called in movement.py. To run one of them, comment out the other and run the script:

```
# Trains a net with earlystopping activation
net.earlystopping(train, train_targets, valid, valid_targets, plot=True)
# Check how well the network performed:
net.confusion(test, test_targets)
# Uses k-fold cross validation.
# net.k_fold_cross_vaildation(movements[:-7, 0:40], target[:-7])
```

or

```
# Trains a net with earlystopping activation
# net.earlystopping(train, train_targets, valid, valid_targets, plot=True)
# Check how well the network performed:
# net.confusion(test, test_targets)

# Uses k-fold cross validation.
net.k_fold_cross_vaildation(movements[:-7, 0:40], target[:-7])
```

Parameters are set in the init method in mlp.py, except for the number of hidden nodes which are set in movement.py.

3 How the program is implemented

3.1 Data structure

The init method initializes the weights for the network, which is randomly distributed between $-1/\sqrt{n} < weight < 1/\sqrt{n}$, where n is the number of inputs to a layer. The weights are updated sequentially, since this is easier to implement iterative. Because of this, the input and label data order is shuffled for every epoch. This is done only for training, as it is to decrease chance of overfitting.

3.2 The Network

The network is updated in the train method. The method takes in the training and target data and feed each vector input through the layers. A bias is added to the input and to the hidden layer. Bias is -1. The activation used is the sigmoid activation function. When the net gives out the output, the error is calculated using the Sum of Squares method which averaged after an epoch is finished. Back propagation happens after the Sum of Square error is first computed. It is done by calculating the delta errors of the output layer and the hidden layer using the derivative of the sigmoid function. The weights are then updated using the following formula:

$$\omega_{\zeta k} \leftarrow \omega_{\zeta k} - \eta * \delta(k) * a_{\zeta} \tag{1}$$

3.3 Earlystopping

The method trains the network until the method deems that overfitting is happening. First the methods takes a validation set forward through the network and calculates the error and the accuracy. It then checks the condition if it is overfitting. If it is not, it will start training the network on a training set. Every epoch, it will check the overfit condition with the validation set and continue to train until it is met. When the latest error is higher than the average of the last 20 errors a counter increments. If the counter reaches 50, the method deems that overfitting has begun. The program then proceeds to run a test set through the network to tests its performance. Confusion matrix and resulting accuracy is then printed.

3.4 k-fold Cross Validation

The k-fold cross validation divides the dataset into a k number of equally large groups. One group becomes a testset, another becomes a validationset and the rest becomes the trainingset. If the user uses a k that does not equally divide the dataset, an error is raised which tells the user to use another k. The dataset is 447 elements long, which reduces the number of k's the user may use. It is therefore sliced to use 440 to make it easier to choose a k. 440 should still be sufficient to perform the cross validation. It is then run through the earlystopping method as described previously. When it is finished, the groups are all rotated one slot, and the validationset and testset will change, as well as most of the training set. This is to validate how well the model trains. The validation is done when the validation set has been at least every data group once. Meaning that for k number of groups, it will rotate k number of times.

4 Results

4.1 Testing with different number of nodes in hidden layer

Accuracy 6 nodes: 90.991%							
1	2	3	4	5	6	7	8
13	0	0	0	0	0	0	0
0	15	0	0	0	0	0	0
0	0	17	0	0	0	0	0
0	0	0	9	1	0	0	0
0	0	0	0	15	0	0	0
0	0	0	0	0	11	0	0
0	0	0	0	0	4	7	0
1	0	0	0	0	0	0	14

Accuracy 8 nodes: 89.189%							
1	2	3	4	5	6	7	8
16	0	0	0	0	0	0	1
0	18	0	0	0	0	0	0
0	0	10	0	0	0	0	0
0	0	7	7	1	0	0	0
0	0	0	0	15	0	0	0
0	0	1	0	0	6	0	0
0	0	2	0	0	4	12	0
0	0	0	0	0	0	0	15

Accuracy 12 nodes: 97.297%							
1	2	3	4	5	6	7	8
11	0	0	0	0	0	0	0
0	14	0	0	0	0	0	0
0	0	17	0	0	0	1	0
0	0	0	13	1	0	0	0
0	0	0	0	11	0	0	0
1	0	0	0	0	11	0	0
0	0	0	0	0	4	14	0
0	0	1	0	0	0	0	17

Table 1: Confusion Matrix

- Accuracy for 6 nodes: 90.991%
- Accuracy for 8 nodes: 89.189%
- Accuracy for 12 nodes: 97.297%

The model with 8 nodes had the least accuracy. It often mistook which was class 3. In total it mistook class 3 a total of 9 times. Also class 7 was mistaken for class 6, 4 times.

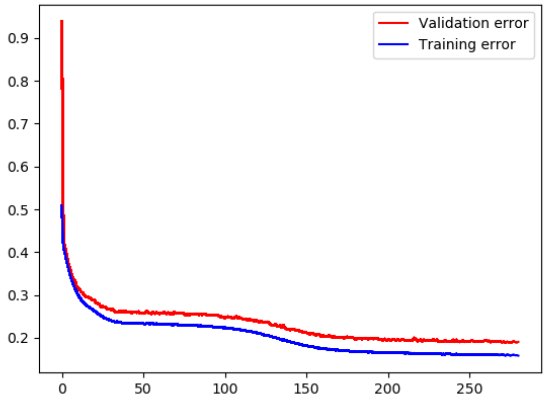

Figure 1: 6 nodes

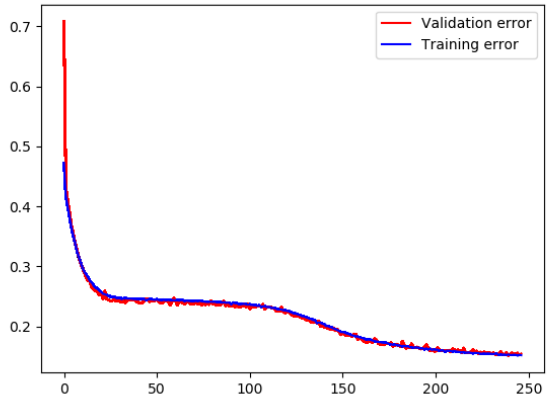

Figure 2: 8 nodes

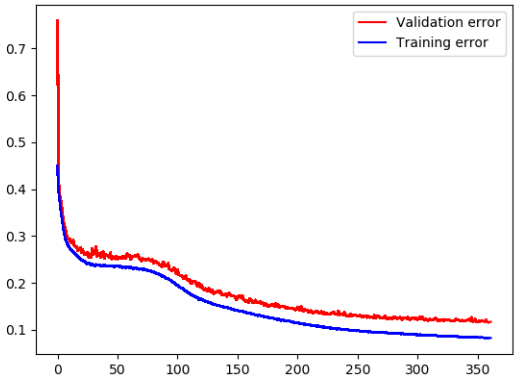

Figure 3: 12 nodes

4.2 Testing k-fold Cross Validation

Accuracy for each fold with k = 8:

1. 89.091%
2. 87.273%
3. 94.545%
4. 96.364%
5. 96.364%
6. 96.364%
7. 94.545%
8. 98.182%

Statistics

- Standard deviation: 0.038
- Mean: 0.941
- Best accuracy: 98.182%