MLP INF3490 oblig2

Øyvind Huuse (oyvinhuu)

October 2016

Contents

Introduction **2**

How to run the code and some explanation **2**

Results **3**
Hidden layer = 6, iterations = 10 4
Hidden layer = 6, iterations = 100 5
Hidden layer = 6, maximum iterations = 1000 6
Hidden layer = 8, iterations = 10 7
Hidden layer = 8, iterations = 100 8
Hidden layer = 8, maximum iterations = 1000 9
Hidden layer = 12, iterations = 10 10
Hidden layer = 12, iterations = 100 11
Hidden layer = 12, maximum iterations = 1000 12

Discussion **13**

Conclusion **13**

Introduction

The goal of this task was to explore supervised learning with multilayer perceptron (MLP) (Fig: 1). We were given a data set consisting of three days of electromygraphic (EMG) signals from a robotic prosthetic hand controller with the object to learn eight hand gestures.

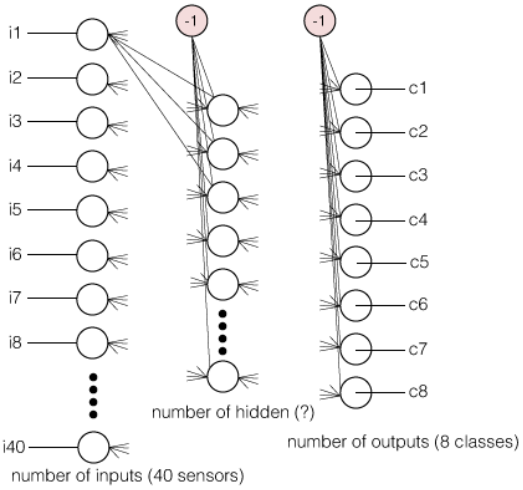

Figure 1: MLP with one hidden layer (figure from inf3490-as2.pdf)

The code is tested with 10, 100 and 1000 (possible) iterations. Additionally it is tested with four different number of hidden layer neurals; 6, 8 and 12.

This paper will report the confusion table for each configuration, try to answer what the minimum number of neurals in the hidden layer is that will result in good classifications by the network. By visual study of the confusion tables this paper will also aim at finding out which classes that are most likely to be mixed up by the network.

How to run the code and some explanation

The code is written in python and based upon the given code handed out with the assignment.

To start the code you write "python movements.py" in the terminal window. The number of iterations the code runs through can be changed in the "mlp.py" file in the input to "def earlystopping" in line 38. The number of hidden layers can be changed in line 37 in the file: "movement.py".

Some changes are made from the default code we received with the assignment.

"def train" runs both the forward and backward step when in training mode.

"def mk_valid" runs the forward step when in validation mode.

"def earlystopping" will try to run "def train" as many times as number of iterations set but will stop if the "def mk_valid", which activates for every 20 iteration, finds that the previous sum square error between the results from running "def mk_valid" and the "validation-target values" starts to increase. This is to avoid over-fitting, when the code starts to adjust to the "noise" in the data and becomes to specialized to the training data-set. If "def earlystopping" exits the iteration sequence or the iteration sequence completes, it will call for "def confusion" which creates a confusion table and the statistics necessary for this assignment (Fig: 2).

The book "Machine Learning - An Algorithmic Percpective" by Stephen Marsland was used as a guide for developing the code and a translation between the variable names in the book to the variable names in the code is presented in the top of "def train" in the file "mlp.py".

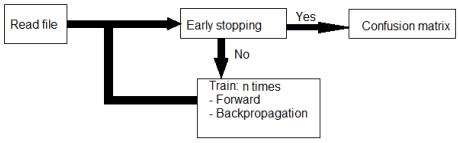

Figure 2: Basic layout of the code (figure from inf3490-as2.pdf)

Results

The results for each set (hidden = 6, 8, 12).

It is worth noticing that running the code, the accuracy varies noticeably between each try whit the same number of hidden neurals and the same number of iterations.

In each following section it is commented what the sum square error was at the last check before it started to increase and at what iteration that occurred at. The validation was done with an interval of 20 runs and started calculating at the first iteration. If the training completed all of its iterations it means that the validation did not stop it, and when that's the case, the previous validation sum square error will not be presented.

Hidden layer = 6, iterations = 10

The training ran through all 10 iterations.

```
Confusion matrix:
"""
Predicted  0.0  1.0  2.0  3.0  4.0  5.0  6.0  7.0  __all__
Actual
0.0         11    0    0    0    0    0    0    2       13
1.0          0   18    0    0    0    0    0    0       18
2.0          0    0   14    0    0    0    0    1       15
3.0          0    0    0   10    0    0    0    3       13
4.0          0    0    0    1   13    0    0    0       14
5.0          0    1    0    0    0    8    2    0       11
6.0          0    0    0    0    0    0   13    0       13
7.0          0    1    0    2    0    0    0   11       14
__all__     11   20   14   13   13    8   15   17      111
"""
```

Class	Accuracy
Overall	0.882882882883
0	0.981982
1	0.981982
2	0.990991
3	0.945946
4	0.990991
5	0.972973
6	0.981982
7	0.918919

Table 1: Accuracy for nhidden = 6, iterations = 10

Hidden layer = 6, iterations = 100

The training ran through all 100 iterations.

```
Confusion matrix:
"""
Predicted   0.0   1.0   2.0   3.0   4.0   5.0   6.0   7.0   __all__
Actual
0.0          10     0     0     0     0     0     0     0        10
1.0           0    11     0     0     0     0     0     0        11
2.0           0     0    13     0     0     0     0     0        13
3.0           0     0     0     8     3     0     0     1        12
4.0           1     0     0     1    14     0     0     0        16
5.0           0     0     0     0     0     7     0     0         7
6.0           0     0     0     0     0     4    20     0        24
7.0           1     0     0     1     0     0     0    16        18
__all__      12    11    13    10    17    11    20    17       111
"""
```

Class	Accuracy
Overall	0.891891891892
0	0.981982
1	1
2	1
3	0.945946
4	0.954955
5	0.963964
6	0.963964
7	0.972973

Table 2: Accuracy for nhidden = 6, iterations = 100

Hidden layer = 6, maximum iterations = 1000

The training ran through 260 iterations and the validation sum square error was 13.23.

```
Confusion matrix:
"""
Predicted   0.0   1.0   2.0   3.0   4.0   5.0   6.0   7.0   __all__
Actual
0.0          14     0     0     0     2     0     0     0        16
1.0           0    12     0     1     0     1     0     0        14
2.0           0     0     8     0     0     0     0     0         8
3.0           0     1     0    13     1     0     0     0        15
4.0           1     0     0     1    10     0     0     0        12
5.0           0     0     0     0     0    13     7     0        20
6.0           0     0     0     0     0     0    10     0        10
7.0           0     0     0     0     0     0     0    16        16
__all__      15    13     8    15    13    14    17    16       111
"""
```

Class	Accuracy
Overall	0.864864864865
0	0.972973
1	0.972973
2	1
3	0.963964
4	0.954955
5	0.927928
6	0.936937
7	1

Table 3: Accuracy for nhidden = 6, iterations = 260

Hidden layer = 8, iterations = 10

The training ran through all 10 iterations.

```
Confusion matrix:
"""
Predicted  0.0  1.0  2.0  3.0  4.0  5.0  6.0  7.0  __all__
Actual
0.0          4    0    0    0    2    0    0    1        7
1.0          0   12    0    0    0    0    0    0       12
2.0          1    0    0    0    0    0   13    5       19
3.0          0    0    0    3    1    0    0    8       12
4.0          0    0    0    1   14    0    0    0       15
5.0          0    3    0    0    2    8    4    0       17
6.0          0    0    0    0    0    1   13    0       14
7.0          0    0    0    0    0    0    0   15       15
__all__      5   15    0    4   19    9   30   29      111
"""
```

Class	Accuracy
Overall	0.621621621622
0	0.963964
1	0.972973
2	0.828829
3	0.90991
4	0.945946
5	0.90991
6	0.837838
7	0.873874

Table 4: Accuracy for nhidden = 8, iterations = 10

Hidden layer = 8, iterations = 100

The training ran through all 100 iterations.

```
Confusion matrix:
"""
Predicted  0.0  1.0  2.0  3.0  4.0  5.0  6.0  7.0  __all__
Actual
0.0         14    0    0    0    0    0    0    0       14
1.0          0   12    0    0    0    0    0    0       12
2.0          0    0   17    1    0    0    0    0       18
3.0          0    0    0   13    0    0    1    1       15
4.0          2    0    0    0    9    0    0    0       11
5.0          0    1    0    0    0    9    1    0       11
6.0          0    0    0    0    0    1   13    0       14
7.0          0    0    0    2    0    0    0   14       16
__all__     16   13   17   16    9   10   15   15      111
"""
```

Class	Accuracy
Overall	0.90990990991
0	0.981982
1	0.990991
2	0.990991
3	0.954955
4	0.981982
5	0.972973
6	0.972973
7	0.972973

Table 5: Accuracy for nhidden = 8, iterations = 100

Hidden layer = 8, maximum iterations = 1000

The training ran through 640 iterations and the validation sum square error was 10.37.

```
Confusion matrix:
"""
Predicted  0.0  1.0  2.0  3.0  4.0  5.0  6.0  7.0  __all__
Actual
0.0          9    0    0    0    0    0    0    0        9
1.0          0   14    0    0    0    0    0    0       14
2.0          0    0   14    0    0    0    0    0       14
3.0          0    0    0   11    0    0    1    1       13
4.0          0    0    0    0   15    0    0    0       15
5.0          0    0    0    0    1   14    0    0       15
6.0          0    0    0    1    0    1   17    0       19
7.0          0    0    0    0    0    0    0   12       12
__all__      9   14   14   12   16   15   18   13      111
"""
```

Class	Accuracy
Overall	0.954954954955
0	1
1	1
2	1
3	0.972973
4	0.990991
5	0.981982
6	0.972973
7	0.990991

Table 6: Accuracy for nhidden = 8, iterations = 640

Hidden layer = 12, iterations = 10

The training ran through all 10 iterations.

```
Confusion matrix:
"""
Predicted  0.0  1.0  2.0  3.0  4.0  5.0  6.0  7.0  __all__
Actual
0.0          8    0    0    0    0    0    0    1        9
1.0          0    0    0    1    1    7    0    5       14
2.0          0    0   13    0    0    0    0    0       13
3.0          0    0    0   14    1    0    0    2       17
4.0          0    0    0    0   14    0    0    0       14
5.0          0    0    0    0    1   15    1    0       17
6.0          0    0    0    0    0    0   13    0       13
7.0          0    0    2    0    0    0    0   12       14
__all__      8    0   15   15   17   22   14   20      111
"""
```

Class	Accuracy
Overall	0.801801801802
0	0.990991
1	0.873874
2	0.981982
3	0.963964
4	0.972973
5	0.918919
6	0.990991
7	0.90991

Table 7: Accuracy for nhidden = 12, iterations = 10

Hidden layer = 12, iterations = 100

The training ran through all 100 iterations.

```
Confusion matrix:
"""
Predicted   0.0   1.0   2.0   3.0   4.0   5.0   6.0   7.0   __all__
Actual
0.0          10     0     0     0     1     0     0     0        11
1.0           0    11     0     0     0     0     0     0        11
2.0           0     0    13     0     0     0     0     0        13
3.0           0     1     0     9     0     0     0     1        11
4.0           0     0     0     1    13     0     0     0        14
5.0           0     0     0     0     0    17     1     0        18
6.0           0     0     0     0     0     0    19     0        19
7.0           0     0     0     0     0     0     0    14        14
__all__      10    12    13    10    14    17    20    15       111
"""
```

Class	Accuracy
Overall	0.954954954955
0	0.990991
1	0.990991
2	1
3	0.972973
4	0.981982
5	0.990991
6	0.990991
7	0.990991

Table 8: Accuracy for nhidden = 12, iterations = 100

Hidden layer = 12, maximum iterations = 1000

The training ran through 240 iterations and the validation sum square error was 16.90.

```
Confusion matrix:
"""
Predicted   0.0   1.0   2.0   3.0   4.0   5.0   6.0   7.0   __all__
Actual
0.0          14     0     0     0     0     0     0     0        14
1.0           0    19     0     0     0     1     0     0        20
2.0           0     0    13     0     0     0     0     1        14
3.0           0     0     0    11     0     0     0     2        13
4.0           0     0     0     1    12     0     0     0        13
5.0           0     0     0     0     0     9     4     0        13
6.0           0     0     0     0     0     1     8     0         9
7.0           0     0     0     0     0     0     0    15        15
__all__      14    19    13    12    12    11    12    18       111
"""
```

Class	Accuracy
Overall	0.90990990991
0	1
1	0.990991
2	0.990991
3	0.972973
4	0.990991
5	0.945946
6	0.954955
7	0.972973

Table 9: Accuracy for nhidden = 12, iterations = 240

Discussion

For 6 neurals in the hidden layer we can see that the overall accuracy is about 88.3% for 10 iterations, 89.2% for 100 iterations and 86.5% for 240 iterations (sum square error in validation = 13.23). This can indicate that with so few neurals in the hidden layer, increasing the number of iterations don't necessarily improve the results.

For 8 neurals in the hidden layer we can see that the overall accuracy is about 62.1% for 10 iterations, 91.0% for 100 iterations and 95.5% for 640 iterations (sum square error in validation = 10.37). This shows a consistent improvement of the result when the number of iterations increase.

For 12 neurals in the hidden layer we can see that the overall accuracy is about 80.2% for 10 iterations, 95.5% for 100 iterations and 90.1% for 240 iterations (sum square error in validation = 16.90). Here we can see that the results improve from 10 to 100 iterations, but then drops between the two results for 240 iterations.

By looking at the confusion tables, it looks like target number 5 often was miss-labeled as number 6 (in 3 simulations). There is also many times number 2 gets miss-labeled as 6, but this occurs all in one simulation.

Conclusion

The results show that the code give poor results for few neurals in the hidden layer and that the best result is obtain for 8 neurals in the hidden layer after 640 iterations and for 12 neurals in the hidden layer after 100 iterations. The results are not consistent. As commented in the start of the results chapter, a great variation in the results was observed between different runs of the code with the same settings. The reason can be that a good combination between iterations and neurals in the hidden layer didn't match up for the pre-set learning rate. It is also possible that the criteria for exiting the learning phase was too strict, leading to a pre-mature testing of the code.

For 100 iterations all three numbers of neurals in the hidden layer resulted in approximately 90% or more in accuracy. For more than 100 iterations the results where more spread for the three numbers of neurals in the hidden layer.

Based on the results from this study, it is not possible to provide a conclusion concerning how many neurals that is needed in the hidden layer to obtain a good classification by the network. Still, it seems that if you have 8 or more neurals and run the code for 100 or more iterations, you will get a good classification with an accuracy above 90%.

Studying the confusion tables visually, it seems that class 5 and 6 where most likely to be mistaken for each other.

Inn all cases, a more valued result could have been obtained by running more tests for the same settings. This could reveal possible trends happening as the number of neurals in the hidden layer and the number of iterations changed.