INF3490 - Assignment 2

Torgeir Hoffmann / twhoffma@uio.no

October 20, 2017

How to run

In code folder:

```
python3 movement.py
```

Additional arguments

May use "--hidden int" for number of hidden nodes, "--mom float" for momentum, "--beta int" for β, "--eta float" for η.

Setup

Maximum of 100 iterations in early stopping by default, not possible to override. 10 sequential training iterations with randomized order. Minimum 0.001 improvement over the last two epochs.

Summarizing Thoughts

There seems to be quite a bit of variation in run time and how well the algorithm performs. However, the general pattern seems to be that the number of hidden nodes are very important.

As can be seen by the confusion table, lower number of hidden nodes compared to output nodes severly limits the efficiency, while 12 hidden mostly get high 70% efficiency, 6 hidden nodes often performs at only 30% efficiency.

This makes sense, since intuitively, one could think that one were compressing the inputs in the hidden layer then expanding it again when moving to the output layer. Intuitively, one can expect that more hidden nodes than input nodes would not yield must improvements over number of hidden nodes between the number of input nodes and output nodes.

In particular, targets with movement "4" (index 3 in array and table) seemed hard to classify for the MLP.

Also, the early stopping seemed volatile in that it could run for a significant variating number of iterations before stopping on same number of hidden nodes. So it looks like number of hidden nodes is not dominant in early stopping.

Confusion Tables

6 hidden nodes (33.33% efficiency)

Efficiency range seems to be 30% - 40%.

target/out	0	1	2	3	4	5	6	7
0	0.00	**0.14**	**0.07**	0.00	0.00	0.00	0.00	**0.79**
1	0.00	**0.94**	0.00	0.00	**0.06**	0.00	0.00	0.00
2	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00
3	0.00	**0.62**	0.00	**0.38**	0.00	0.00	0.00	0.00
4	0.00	**0.09**	0.00	**0.36**	**0.55**	0.00	0.00	0.00
5	0.00	**0.09**	0.00	0.00	**0.91**	0.00	0.00	0.00
6	0.00	**0.05**	0.00	**0.05**	**0.90**	0.00	0.00	0.00
7	0.00	**0.06**	0.00	**0.12**	0.00	0.00	0.00	**0.81**

8 hidden nodes (45.05% efficiency)

Efficiency range seems 40% - 50%.

target/out	0	1	2	3	4	5	6	7
0	**0.11**	0.00	**0.06**	0.00	0.00	0.00	0.00	**0.83**
1	0.00	0.00	**0.92**	0.00	0.00	0.00	**0.08**	0.00
2	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	**0.67**	0.00	**0.28**	0.00	**0.06**	0.00
4	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	**0.67**	**0.33**	0.00
6	0.00	0.00	0.00	0.00	0.00	**0.40**	**0.60**	0.00
7	**0.25**	0.00	**0.25**	0.00	0.00	0.00	0.00	**0.50**

12 hidden nodes (76.58% efficiency)

Efficiency range is mostly 70% and above.

target/out	0	1	2	3	4	5	6	7
0	**0.69**	0.00	0.00	0.00	0.00	**0.19**	0.00	**0.12**
1	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	**0.06**	**0.38**	**0.31**	**0.12**	**0.12**
4	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	**0.38**	**0.62**	0.00
7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**1.00**

Which classes could be mistaken for each other?

Class 4 (index 3) seems hard to place from the 5,6,7,8 for this run in particular. However, this is also a pattern that emerges in many subsequent runs. This does not seem to be the case for the other classes.

This is also often the case for class 5 and 6