Pattern Recognition Report - Serie 2b - MLP

JungleSpeed

Optimization of number of neurons in the hidden layer

We first try to optimize the number of neurons in the hidden layer. We fixed :

- epochs = 10
- learning rate = 0.01

For a number of neurons in the range [10, 100] we obtain the following results :

number of neurons	loss value	accuracy
10	0.338	0.902
20	0.311	0,912
30	0.294	0,916
40	0.295	0.918
50	0,280	0,923
60	0.278	0.923
70	0.288	0,922
80	0.285	0.920
90	0.281	0.921
100	0.277	0.922

TABLE 1 – Number of neurons optimization

We obtained a better couple accuracy-loss values with 60 neurons in the hidden layer.

Optimization of learning rate

Then we try to optimize the learning rate parameter. We fixed :

- epochs = 10
- number of neurons = 60

For a learning rate in the range [0.001, 0.1] we obtain the following results :
We obtained a better couple accuracy-loss values with a learning rate of 0.1.

number of neurons	loss value	accuracy
0.001	0.692	0,845
0.005	0.350	0.902
0.01	0.279	0.921
0.05	0.165	0.918
0,1	0.118	0.965

TABLE 2 – Learning rate optimization

Optimization of number of iterations

Finally we optimized the number of iterations. We fixed :

- learning rate = 0.1
- number of neurons = 60

For a number of iterations in the range [10, 100] we obtain the following results :

number of neurons	loss value	accuracy
5	0.164	0.951
10	0.120	0,964
15	0.101	0,971
20	0.092	0.973
30	0,080	0,973
40	0.076	0.977
50	0.082	0,976
75	0.082	0.977
100	0.088	0.976

TABLE 3 – Number of iterations optimization

As we can see, values start to stabilize for 50 epochs. So there is no need to run the model with more than 50 epochs.

Random initialization

According to the previous results, we perform random initialization to refine our parameters and choose the model performing the best. Here our random initialization features :

- epochs = random.choice([35, 40, 45, 50, 55, 60])
- learning rate = random.choice([0.05, 0.075, 0.1])
- number of neurons = random.randint(50, 80)

number of neurons	lr	epochs	loss value	accuracy
55	0.1	50	0.084	0.9752
70	0.05	50	0.081	0.9752
74	0.075	40	0.084	0.9758
40	0.05	56	0.097	0.9711
79	0.1	45	0.079	0.9744
71	0.05	35	0.090	0.9717
56	0.05	35	0.097	0.9714
77	0.05	45	0.082	0.9748
62	0.1	40	0.085	0.9761
56	0.05	55	0.084	0.9741
74	0.075	40	0.084	0.9758
52	0.075	60	0.083	0.9747

TABLE 4 – random initialization

We obtain the following results for 10 steps :

Best results are with the model :

- number of neurons = 62
- learning rate = 0.1
- epochs = 40

Here the plots showing the accuracy and loss on the training and the test set with respect to the training epochs.

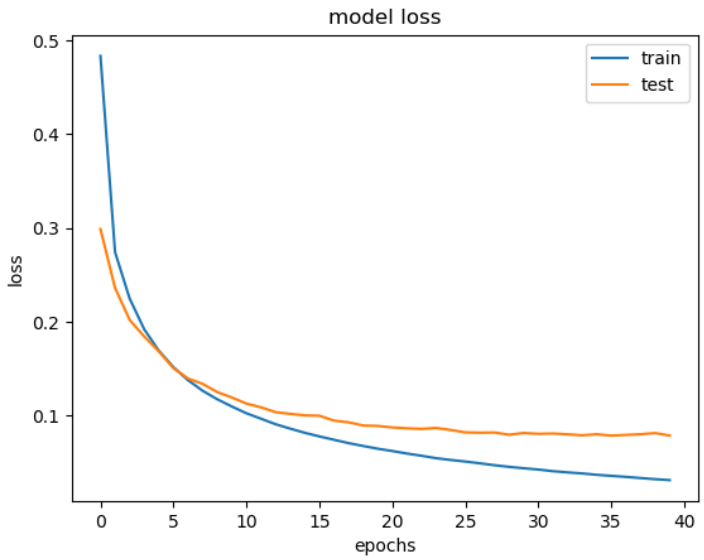

FIGURE 1 – Loss function

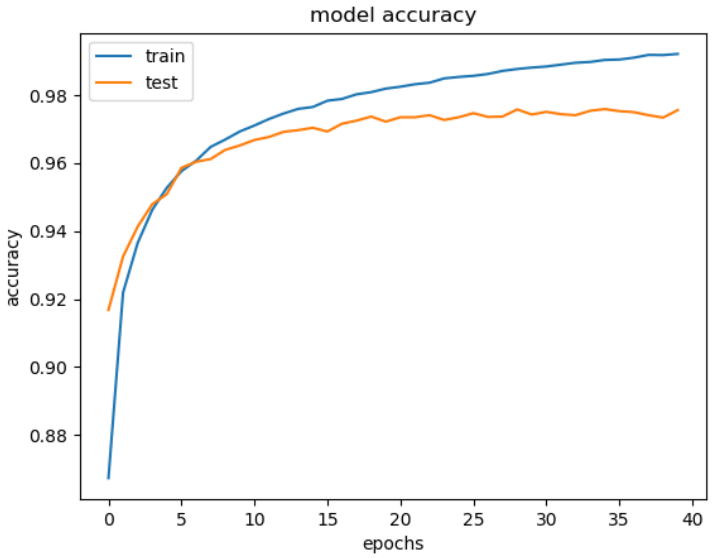

FIGURE 2 – Accuracy function

Notes

For all our models, we used the *Keras* and *TensorFlow* libraries.
We also choose to use SGD as default optimizer. We are aware that we can also change the optimizer to optimize our model.