MVA: Deep Learning in practice (2022/2023) **TP1**

Deep Learning in practice

Lecturers: *G. CHARPIAT* *(January, 2023)*

Solution by DI PIAZZA Théo

Instructions

- This file is the report of the TP1.

1 Problem 1 - MultiClass

In this exercise, we are interested in the classification problem of the USPS dataset (10 classes, 16x16 pixels for each image, 9298 samples). Below is an example of the samples :

Figure 1: Images from USPS dataset.

The objective is to understand the different architectures of a neural network and the influences of the different hyper-parameters on the performance of the model.

The best model is the one that performed the best on the test sample. It is the model trained with the following configuration : Adam [lr 0.001] on 40 epochs, batch size = 10, MSELoss and 1 CNN (3x3 kernel, 3 channels and 2 for padding) + ReLU as hidden layer.

1.1 Exercice 1 - Impact of the architecture of the model

Impact of the number of linear layers From the neural network with 1 single linear hidden layer with 100 neurons, we propose to see the impact of adding hidden layers on the model performance. We try with 1, 2 and 3 hidden layers with 100 neurons each. When the number of layers increases, the accuracy of the model decreases and the execution time increases.

Impact of the number of neurons From the model with a single linear hidden layer, we modify the number of neurons of this layer to see the impact on the model. We try for 20, 50, 100 and 150 neurons. When the number of neurons increases, the accuracy increases up to 100 neurons and then stagnates, and the execution time also increases.

Impact of the activation function From the model with a single linear layer followed by an activation function, which precedes the last linear output layer, we compare the results obtained using: ReLU, Tanh and Sigmoid. The results seem to be equivalent for ReLU and Tanh but underperform for Sigmoid. Indeed, the Sigmoid activation function is generally used after the final layer of the network in order to obtain values interpretable as probabilities. The execution times are almost equivalent.

Impact of the number of layers in CNN From a new model with convolution layers (3 channels, 3x3 kernel size and padding=2) followed by a ReLU activation function, in hidden layer. We compare the results using 1, 2 and 3 hidden layers. The results are very good and almost equivalent but the performances seem to decrease very slightly with the depth of the hidden layers. Generally, we say "the deeper the better" but it is not always the case, maybe because the classification task is relatively simple.

Impact of the activation function on CNN From the model with a single convolution layer followed by an activation function, which precedes the last linear output layer, we compare the results obtained using: ReLU, Tanh and LeakyReLU. The results are almost equivalent and perform very well. The execution times are almost equivalent.

1.2 Exercice 2 - Impact of the optimizer

For the following exercice, the model with 1 CNN+ReLU will be used.
Impact of batch size For a learning rate fixed at 0.1 we compare different batch sizes (number of samples per batch): 10, 50, 100, 250, 500 and 1000. The more the batch size increases, the more the precision decreases. Indeed, when the batch size is modified, it is also necessary to adjust the learning rate. The execution times are equivalent.

Impact of the learning rate For a batch size fixed to 10, we compare different learning rates : 0.001, 0.01, 0.1, 0.5, 1 and 10. When the learning rate is too low (0.001, 0.01) or too high (10), the model does not seem to learn correctly. For values of 0.1 and 0.5, the results are very good. The execution times are equivalent.

Impact of the learning rate We compare for 5, 10, 20 and 40 epochs. When the number of epochs increases, the accuracy of the model increases and the execution time also increases.

Impact of the optimizer We compare 3 optimizers with the adjusted learning rate: SGD (0.1), Adam (0.001) and RMSprop (0.001) The results are very good and almost equivalent, but the execution times are longer for Adam and RMSprop.

1.3 Exercice 3 - Impact of the loss function

We propose to use the nn.CrossEntropyLoss instead of the MSE. With an Adam optimizer (lr 0.001), batch size of 10 out of 10 epochs, we obtain an accuracy of 94.81% on the validation set. This loss seems well adapted to our problem. Even if in theory, the CrossEntropy loss is more adapted to classification problems, here it is the MSE which allows to obtain the best results.

1.4 Exercice 4 - Prediction on test set

For our best model (Adam [lr 0.001] on 40 epochs, batch size = 10, MSELoss and 1 CNN+ReLU as hidden layer), we obtain a precision of 92.23% on the test sample.

Concerning the details of the experiments, **click here to access the Appendix of problem 1**. Remark : *For each experiment, we change the variable one at a time to estimate its "direct impact" on the model. It is important to note that the impact of a hyper-parameter can be different depending on the value of the other hyper-parameters (e.g. the influence of the learning rate and the batch size on the quality of the learning). This has been taken into account for the final model.*

2 Problem 2 - Regression

In this problem, the goal is to predict, from 1460 samples, the house selling prices from 4 features (OverallQual, YearBuilt, TotalBsmtSF, GrLivArea).

In the first part, the criterion that is used the the MSELoss function. Due to a gradient explosion, the data are normalized. As in the previous section, different configurations are tested. For each configuration, only one variable is modified in order to see its "direct" impact on the performance of the trained model. In a second step, the Gaussian Loss is used, in order to predict a probability of distribution

rather than a quantity, notably in order to manage uncertainties with the variance.

The observations are similar to the previous problem: When the number of neurons or layers increases, the model tends to be more accurate but this has a cost: a risk of overtraining and an increase in computation time. Moreover, we use linear layers rather than convolution layers which are more adapted, here, to tabular data. Finally, the ReLU activation function is retained because it presents better performances than leakyReLU, Tanh and in particular Sigmoid. Concerning the optimizer, it is important to adjust the learning rate to it. For this problem, SGD seems to converge much less quickly to the optimal solution than RMSprop and Adam, despite a learning rate tuning. Finally, increasing the number of epochs allows to decrease the Loss, but we have to be careful to control the learning rate (for example with a learning rate decay) in order not to overdrive the model on the train set, or to diverge.

The selected model is the one with the lowest loss on the test sample (Gaussian Loss : 0.22) with the following configuration : 3 linear layers of 20 neurons followed by ReLU, on 20 epochs with a batch size of 10, and RMSprop optimizer (lr : 0.001).

Concerning the details of the experiments, **click here to access the Appendix of problem 2.** For the sake of simplicity, we only present the results obtained with the Gaussian Loss because the observations are similar. Remark : *For each experiment, we change the variable one at a time to estimate its "direct impact" on the model. It is important to note that the impact of a hyper-parameter can be different depending on the value of the other hyper-parameters (e.g. the influence of the learning rate and the batch size on the quality of the learning). This has been taken into account for the final model.*

Appendix

Problem 1 - Exercice 1

Impact of the number of linear layers			
Number of hidden layers	Train accuracy	Test accuracy	Execution time
1	93.30%	92.40%	7.65s
2	90.10%	90.09%	8.35s
3	57.25%	58.71%	9.13s

Impact of the number of neurons			
Number of neurons	Train accuracy	Test accuracy	Execution time
20	91.37%	90.47%	7.45s
50	92.98%	91.87%	7.45s
100	93.30%	92.40%	7.65s
150	93.15%	92.25%	7.95s

Impact of the activation function			
Activation function	Train accuracy	Test accuracy	Execution time
ReLU	91.37%	90.47%	7.45s
Tanh	93.25%	92.33%	7.67s
Sigmoid	47.22%	48.31%	8.30s

Impact of the number of CNN layers			
Number of hidden layers	Train accuracy	Test accuracy	Execution time
1	96.02%	94.58%	8.92s
2	95.60%	93.49%	12.16s
3	93.15%	93.15%	15.45s

Impact of the activation function after CNN			
Activation function	Train accuracy	Test accuracy	Execution time
ReLU	96.02%	94.58%	8.92s
LeakyReLU	95.73%	94.58%	9.48s
Tanh	95.73%	94.04%	9.37s

Problem 1 - Exercice 2

Impact of the batch size			
Batch size	Train accuracy	Test accuracy	Execution time
10	96.02%	94.58%	8.92s
50	92.82%	92.02%	6.37s
100	89.12%	89.31%	5.63s
250	32.48%	35.17%	5.39s
500	27.25%	28.81%	5.20s
1000	8.10%	7.28%	5.41s

Impact of the learning rate			
Learning rate	Train accuracy	Test accuracy	Execution time
0.001	26.40%	28.51%	9.51s
0.01	71.45%	73.43%	9.16s
0.1	96.02%	94.58%	8.92s
0.5	96.62%	94.73%	9.36s
1	97.22%	94.97%	8.92s
10	16.13%	17.51%	9.32s

Impact of the number of epochs			
Epochs	Train accuracy	Test accuracy	Execution time
5	96.77%	94.89%	5.61s
10	796.62%	94.73%	9.36s
20	98.90%	96.59%	21.11s
40	99.22%	96.90%	42.04ss

Impact of the optimizer			
Optimizer (lr)	Train accuracy	Test accuracy	Execution time
SGD (0.1)	96.62%	94.73%	9.36s
Adam (0.001)	97.27%	95.12%	11.03s
RMSprop (0.001)	96.95%	94.89%	11.04s

Problem 2 - Gaussian Loss

Impact of the number of layers			
Number of layers	Train Loss	Test Loss	Execution time
1	0.20	0.23	2.87s
2	0.21	0.25	2.86s
3	0.25	0.27	3.78s

Impact of the number of neurons			
Number of neurons	Train Loss	Test Loss	Execution time
5	0.91	0.89	4.08s
10	0.21	0.25	4.39s
20	0.16	0.22	4.40s
40	0.25	0.27	4.42s
50	0.24	0.35	4.71s

Impact of the activation function			
Activation function	Train Loss	Test Loss	Execution time
ReLU	0.18	0.26	4.42s
leakyReLU	0.17	0.23	4.43s
Sigmoid	0.40	0.44	4.12s
Tanh	0.26	0.33	5.13s

Impact of the batch size			
Batch size	Train Loss	Test Loss	Execution time
10	0.17	0.23	4.43s
50	0.24	0.27	1.99s
100	0.83	0.80	1.09s
200	1.02	0.97	0.57s
400	1.19	1.17	0.56s

Impact of the optimizer			
Optimizer (lr)	Train Loss	Test Loss	Execution time
RMSprop (0.001)	0.17	0.23	4.43s
Adam (0.001)	0.17	0.23	5.90s
SGD (0.01)	0.48	0.50	3.92s

Impact of the number of epochs			
Number of epochs	Train Loss	Test Loss	Execution time
10	0.26	0.29	3.14s
20	0.17	0.23	5.90s
50	0.18	0.32	12.85s
100	0.07	0.27	28.40s