Deep Learning in Practice

Homework 5

Student :
Abdellah Kaissari
Mohamed Kerroumi
Randa Elmrabet Tarmach

Homework 1

We study in this homework a binary classification problem, performed by a neural network. Each input has two real features, and the output is a binary variable (0 or 1). Training and validation sets contain respectively 4000 and 1000 examples (corresponding respectively to 80% and 20% of the data set).
The data set is presented as in Figure 1:

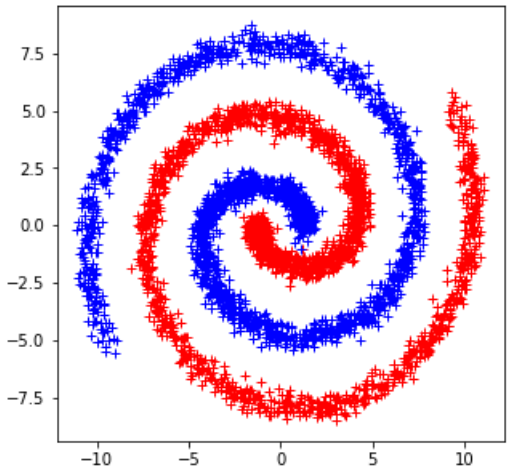


Figure 1: Data set formed of two interlaced spirals

Exercise 1: Impact of the architecture of the model

We have tried 6 different architectures varying the number of linear hidden layers (1, 2, 3, 4, 5 and 10) with 10 neurons and Relu as an activation function. We have trained these models using MSE loss, SGD optimizer with $learning\_rate = 0.01$, and $batch\_size = 10$ for 10 epochs.

- **Impact of the number of layers :** The model with 10 layers predicts randomly the label of inputs. We can improve results for deep neural network by increasing the number of epochs and changing the learning rate. In fact, the model doesn't have enough time to learn the parameters (since in this case we have lot of parameters compared to the neural network with 4 layers) and to converge.
- **Impact of the activation functions :** We have tried different activation functions on the model with 4 hidden layers. When we used Relu, the accuracy is 59.6%, with tanh the accuracy is 57.30% and with sigmoid the accuracy is 50%. As expected, the Relu function gives the best score. using this function, we have at each time only few neurons activated which makes the network sparse and efficient.

- **Impact of the number of neurons for each layer :** with 2 neurons on each layer and using the Relu function we obtained an accuracy of 50%. with 10 neurons we obtained 59.6%, with 150 neurons we obtained 67.1% and with 1000 neurons 71.4%. We can see that adding neurons can increase the accuracy of the model but the training takes more time.

Exercise 2: Impact of the optimizer

We will use here a neural network with 4 hidden layers, 10 neurons en each hidden layer with the Relu as the activation function and the MSE loss.

- **Impact of batch size :** the training with batch_size=10 gives the accuracy= 60.5%, with batch_size=100 the accuracy= 56.4%, with batch_size=400 the accuracy= 50%. The model with batch_size= 10 is the best. In general with a small batch size the model is better but it takes more times for the training. We will take batch_size= 10 in the following experiences.
- **Impact of the learning rate :** the accuracy score with $lr = 0.001$ is 50%, with $lr = 0.01$ is 60.5% , with $lr = 0.1$ is 99.90% with $lr = 1$ is 74.2% and with $lr = 10$ is 50%. As expected, when the learning rate is very small, we stop the training before the model converges and when the learning rate is big the model does not converge. we will use 0.1 and 0.01 as learning rate because it depends also with respect to the number of epochs.
- **Impact of the number of epochs :** with 10 epochs and $lr = 0.1$ we obtained an accuracy of 99.9%, with 20 epochs we obtained an accuracy of 97.5% in this case the model is overfitting the data. with 12 epochs we obtained an accuracy of 100%. it's important to choose the optimal number of epochs to avoid overfitting and to have the best accuracy score in the validation set.
- **Impact of the optimizer :** we have tried two others optimizers, Adam and RMSprop. And both converge faster with $lr = 0.01$ than the SGD optimizer.

Exercise 3: Impact of the loss function

In this case, both the MSE loss and the BCE give 100% of accuracy in the validation set with the same model.

Exercise 4: Prediction on test set

Finally, we have chosen a model with 4 hidden layers, 10 neurons in each hidden layer, Relu as the activation function in each layer, the BCEWithLogitsLoss as loss function, 10 for the batch_size and the number of epochs and ADAM as the optimizer with $lr = 0.01$.
This model gives an accuracy of 100% in both the validation and the test set.

Homework 2 : Multi-classification

In this assignment, we will build a model for multi-classification on the MNIST Dataset, this dataset contains 50000 images for training, 10000 for validation, and 10000 for test. Each image in the dataset is a gray-scale image of digits of shape (28×28), so the neural network we will implement takes as input one of these images, and tries to predict the class of the digit in the image $0, 1, 2, ..., 9$

Exercise 1: Impact of the architecture of the model

We started by evaluating the performance of the initial model, then we introduced 1 Conv Layer, then we added more Conv Layers , we kept the same parameters of the model (SGD as optimizer, MSE as a loss and training for 10 epochs). The performances of the different models are in the table below:

Models	1 Linear Layer	1 Conv Layer	2 Conv Layers	2 Conv 2 Linear Layers
Training loss at epoch 10	0.0208	0.0128	0.0091	0.0099
Validation Accuracy	87.84 %	91.71 %	94.21 %	93.84 %
Test Accuracy	89.02 %	92.39 %	94.80 %	94.66 %

Table 1: Models Comparison

Now we keep the model with 2 Conv Layers and 2 Dense Layers, and we will modify the number of neurons in the first dense layer, and we'll see how the accuracy will vary.

Models	5 Neurons	10 Neurons	20 Neurons
Training loss at epoch 10	0.0370	0.01828	0.0137
Validation Accuracy	76.76 %	88.41 %	91.46%
Test Accuracy	77.50 %	89.29 %	91.84%

Table 2: Models Comparison

From the table above, the more neurons in layers , the better is the accuracy . Hence we kept the model with 30 neurons for the first dense layers. With the **ReLu** activation, as we can see, the validation accuracy is 91.46%. We changed the activation function and use this time **Tanh**, and we got 92.26% in the validation accuracy, so we decided to keep **Tanh** as the activation function.

Exercise 2: Impact of the optimizer

We tried different values for the batch size, the table below presents the accuracy for each value.

Batch Size	10	100	1000
Validation Accuracy	92.26 %	14.90 %	12.36 %

Table 3: Models Comparison

So the batch size has to be small in order that the training goes well. Then we changed the optimizer, we found ADAM optimizer more suitable for the training because the loss decreased constantly unlike SGD for which the loss oscillated a little bit. We found also that 10^{-3} is the appropriate value for the learning rate lr, if we give the lr a bigger value, the loss starts oscillating, and if we give a smaller value, the training needs more epochs to converge. We chose $Num_epochs = 30$, this value is appropriate for the model because it's large enough for the model to converge, and it's not very big, so the model does not overfit the training dataset.

Exercise 3: Impact of the loss function

We trained two models with the same architecture, the only difference between them is the **Loss function**, one uses the **MSE loss** and gives 98.45% of validation accuracy , and the second uses the **Cross Entropy loss** and gives 98.60% of validation accuracy. So we will use for the finale model the Cross Entropy Loss.

Exercise 4: Prediction on test set

Finally, we have chosen a model with 4 hidden layers, 2 Conv Layers and 2 Dense layers, We trained the model for 30 epochs with batch size of 32, we used ADAM as optimizer with $lr = 10^{-3}$ and for the loss, we used the **Cross Entropy Loss**. We tested our model on the test set, and we got an Accuracy of **99%** .

Homework 3 : Multi-Task Problem: Colored MNIST

We study a multi classification problem on colored MNIST data set of 10 digits and 7 color classes, containing 50000 images in training set, 10000 in validation set, and 10000 in test set. Each data point is a colored handwritten digit image of shape 28×28. The goal is to implement a neural network taking an image as an input, able to predict both number and color represented by the digit within the image.

Exercise 1: Impact of the architecture of the model

Keeping the initial training configuration: MSE as a loss function, SGD as an optimizer, *learning rate* $= 10^{-2}$, *batch size* $= 10$ and 10 epochs as training time, we evaluate several models using different combinations of linear and convolutional layers as shown in the table below.

- **Impact of number of layers** : Firstly, we model both tasks with the same network architecture, trained in parallel adding a softmax activation function as the last layer.

Models	1 Linear Layer	1 Conv2D & 1 Linear Layers	2 Conv2D & 1 Linear Layers	2 Conv2D & 2 Linear Layers
Training loss	0.3143	0.3946	0.0016	0.0011
Number Accuracy	38.75 %	24.07 %	97.66 %	98.10 %
Color Accuracy	33.45 %	16.87 %	100.00 %	100.00 %
Validation Accuracy	12.76 %	4.05 %	97.66 %	98.10 %

Table 4: Models Comparison

Afterwards, we model both tasks using one common neural network of 2 convolutional layers and 2 linear layers, sharing weights except for the last one as it generates two different outputs. After 10 epochs, we obtain a loss of 0.0015 on the training set, and an accuracy of 97.77% on the validation set with Number accuracy of 97.79% and Color accuracy of 99.98%.

- **Impact of number of neurons** : Here, using 2 convolutional layers and 2 linear layers, we modify the number of neurons in each layer, and observe the impact on the validation set accuracy. As expected, the more neurons in layers, the better the

Models	Conv2D(3,5) Conv2D(5,5) DenseLayer(5)	Conv2D(3,10) Conv2D(10,10) DenseLayer(10)	Conv2D(3,20) Conv2D(20,20) DenseLayer(20)	Conv2D(3,50) Conv2D(50,50) DenseLayer(50)
Training loss	0.0634	0.0011	0.0012	0.0006
Number Accuracy	52.87 %	98.10 %	97.78 %	98.20%
Color Accuracy	99.95 %	100.00 %	99.99 %	99.99%
Validation Accuracy	52.84 %	98.10 %	97.77%	98.19%

Table 5: Model performance wrt Number of neurons

accuracy is, the slower the training is. Hence we keep the model with 2 Conv2D layers of 10 neurons and 2 Linear layers with 50 neurons in the first dense layer.

- **Impact of activation functions** : With the ReLu activation function, the validation accuracy is 98.10%. We achieved a validation accuracy of 95.98% using Tanh. Hence, we keep ReLu as an activation function in our neural network.

Exercise 2: Impact of the optimizer

- **Impact of batch size** : We train our model with different values of the batch size, the table below presents the accuracy of each value. The larger the batch size is, the

Batch Size	10	64	256	1000	10000
Training loss	0.0011	0.0784	0.0937	0.1436	0.2088
Validation Accuracy	98.10 %	47.68 %	14.37 %	11.62%	3.37%

Table 6: Model performance wrt Batch Size

worse the validation accuracy is. In this case, the convergence is slow, hence we should increase the number of training epochs.

- **Impact of learning rate** : We train our model with different values of the learning rate, the table below presents the accuracy of each value. We observe that both $0,1$

Learning Rate	5	0.5	0.1	0.01	0.001
Training loss	0.4171	0.0032	0.0015	0.0011	0.0865
Validation Accuracy	2.01%	97.45%	98.41%	98.10%	36.28%

Table 7: Model performance wrt Learning Rate

and 0.01 are appropriate values of the learning rate. We choose the value of 0.01 to avoid divergence of the loss. When we set a large learning rate value, the loss diverges or starts oscillating and the smaller the learning rate is, the slower the convergence is.

- **Impact of optimizer and number of epochs** : With a learning rate of 0.01, we have tried SGD, Adam and RMSProp as optimizers and find SGD a suitable one for its fast convergence. We also set the number of training epochs to 20, being large enough to ensure the convergence of our model without over-fitting on the training set.

Exercise 3: Impact of the loss function

We train our model using two different loss functions, MSE and Cross Entropy loss. The one trained with MSE loss gives 97.81% on the validation accuracy, and the one trained with Cross Entropy loss achieves a validation accuracy of 90.59%. Therefore, we use the MSE Loss for the training of our model.

Exercise 4: Prediction on test set

Our final model is composed of 4 hidden layers, a Conv2D(3,10) layer, a Conv2D(10,10), a Dense layer of 50 neurons and 2 final Dense layers (number and color prediction) followed of sigmoid activation function. We add ReLu activation function after each hidden layer. We train our model for 20 epochs using a batch size of 10. We optimize the MSE loss using SGD as an optimizer with a learning rate of 0.01. Our model achieves an accuracy of 98.06% on the test set, with a test accuracy for numbers of 98.09% and a test accuracy for colors of 99.97%.

Homework 4 : Regression Problem

In this assignment, we will build a Regression model for the prediction of wind production, each row in the dataset corresponds to a given time, and contains three input features describing the weather at that time.

Exercise 1: Impact of the architecture of the model

We started by evaluating the performance of a model with 1 layer, then we added more Layers, we used as parameters of the model (Adam as optimizer, MSE as a loss and training for 30 epochs). The performances of the different models are in the table above.

Models	1 Layer	2 Layers	3 Layers
Training MSE after 30 Epochs	4187554.34	2100769.41	1625061.44
Validation MSE	1454633.37	590386.56	387048.71

Table 8: Models Comparison

Then we kept the model with 3 Layers. And we changed the **Tanh** activation function by the **Sigmoid** activation, we got **408622.18** in the validation MSE, so clearly Tanh is the appropriate activation function for the model.
and we will modify the number of neurons in the dense layers, and we'll see how the MSE will vary.

Models	5 Neurons	10 Neurons	20 Neurons
Training MSE after 30 Epochs	5624862.96	4024952.80	**1625061.44**
Validation MSE	1983835.12	1390246.25	**387048.71**

Table 9: Models Comparison

From the table above, the more neurons in layers , the smaller is the MSE . Hence we kept the model with 20 neurons in all the layers. With the **Tanh** activation.

Exercise 2: Impact of the optimizer

We tried different values for the batch size, the table below presents the MSE for each value.

Batch Size	10	100	1000
Validation MSE	**387048.71**	3379446.5	4414091.5

Table 10: Models Comparison

Same remarks as in Exercise 2 ,the batch size has to be small in order that the training

goes well. Then we changed the optimizer, we found ADAM optimizer more suitable for the training . We found also that 0.01 is the appropriate value for the learning rate lr, We chose $Num\_epochs = 50$.

Exercise 3: Impact of the loss function

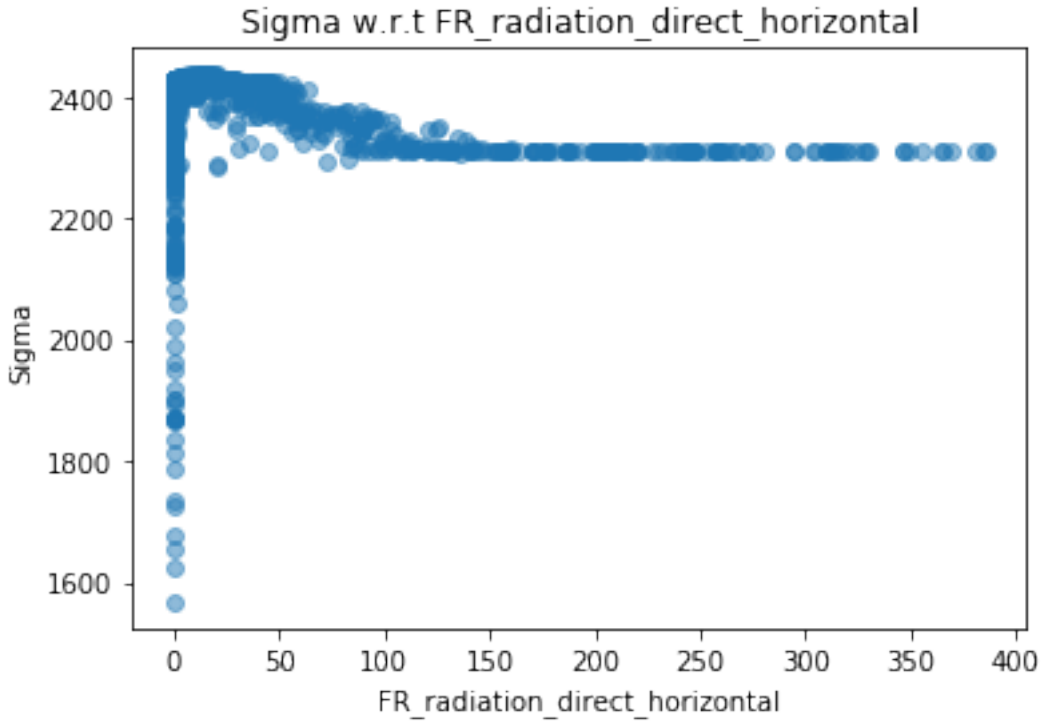
As mentionned in the assignment, we changed the loss function, we used the Gaussian likelihood loss function defined by :

$$\frac{1}{N} \sum_{1}^{N} log(\sigma_i^2) \quad + \quad \frac{1}{\sigma_i^2}(y_i - \mu_i)^2$$

Unlike the MSE loss, this new function decreased during training and after some epochs, became close to 0 (16.33) ,

Exercise 4: Prediction on test set

The finale model we used had 3 Dense layers, and we used the **Gaussian likelihood loss function**.On the training set after 50 epochs, we got 17.23 in the loss, and on the validation set, we got **16.33** for the loss, and on the test set, we got **17.11**.



The plot above represents the variance $\sigma(x)$ w.r.t to FR_radiation_direct_horizontal. Indeed, the uncertainty decreases when the feature FR_radiation_direct_horizontal increases.