UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Laboratory Exercise 2: Neural Networks

Computational Intelligence

João Valério

Eirik Grytøyr

Date: 28/10/2022

Index

1. INTRODUCTION

The main goal of the present work is to analyse the performance of different MLP configurations to classify images from the "CalTech 101 Silhouettes" Data Set. The data set is constituted of 8671 images, each with 28x28 pixels, corresponding to 784 input variables and 101 outputs.

It is necessary to perform preprocessing, in order to obtain the proper data set to train the Neural Network. Since the different values are already normalized, it is only necessary to encode the labels through a one-hot encoding scheme and to transpose the input data, in order to match the output.

The model applied is a Feed-Forward Neural Network with a fixed number of 2 layers in its design, one hidden and one output layer. The report of the performance is based on the mean accuracy measure, with 10 executions per configuration.

Finally, the project is divided into 3 parts. First, includes the study of how the complexity of the network, the distribution of the data and the selection of the activation functions affect the outcomes. The second aspect compares distinct training functions and the last examines the effect of different parameters on accuracy.

2. CONFIGURATIONS

In order to test distinct configurations, the following modifications were performed along the executions:

- **Units:** vary between 50, 200 and 500 units;
- **Layer Functions:** logsig/logsig with mean squared error and logsig/softmax with cross-entropy;
- **Data Split:** 80/10/10, 40/20/40 and 10/10/80 (training/validation/test data).

In the first analysis, it is considered the gradient descent with momentum and adaptive learning rate backpropagation (traingdx) as the training function. Originally, it is necessary to select the proper initial learning rate, momentum and the number of epochs. The first two are established with 0.01 and 0.8, which are common values utilised in Neural Networks. The quantity of epochs is set to 300. This is bigger than the epoch in which traingdx converges in the most exigent test (500 units, 80/10/10 data split and Logsig/Softmax as functions). The accuracy values obtained in the various arrangements are presented in table 1, Appendix A. Finally, it is important to denote that each possibility was executed 10 times, but the values still have a considerable error margin. A test by running the training 20 times, showed a difference from the mean value of up to +- 14%

a. Units

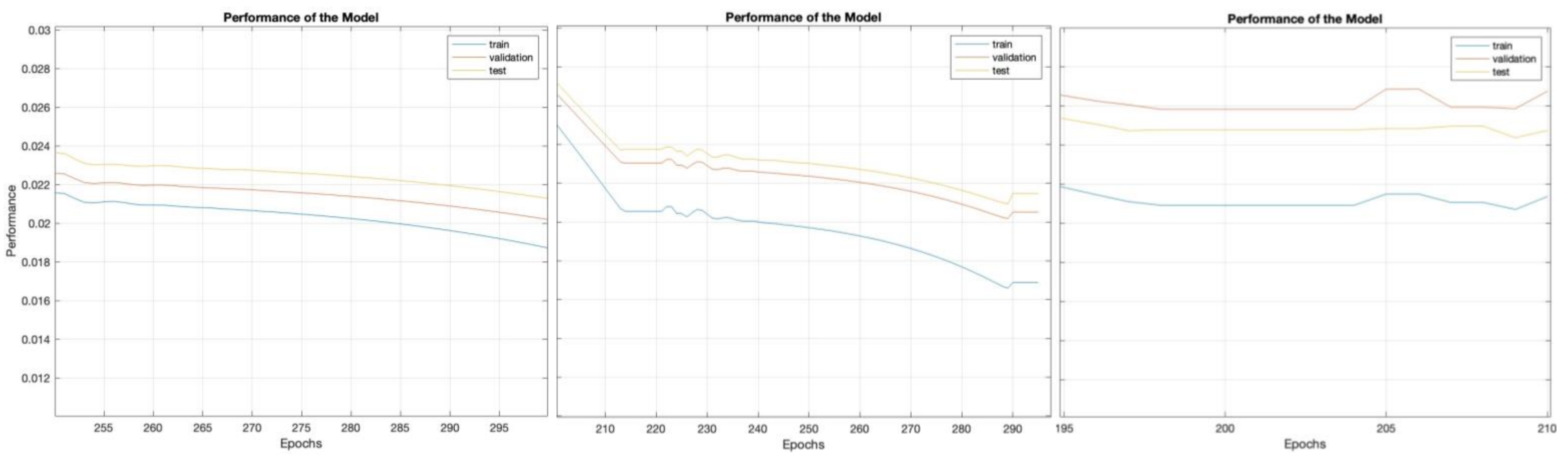
The number of units affects directly the complexity of the Neural Network, such that more units imply a higher degree of complexity. According to that, this parameter is directly responsible for the quality of fitting (underfitting, normal fitting and overfitting) of the model to the data.

By considering the values acquired in table 1, it is perceptible that raising the number of units, for the same quantities of data, implies an increment in all the training accuracies. This pattern suggests that the model conceived is learning more about the training data, but not necessarily useful information. So, this proposes two potential scenarios: a model that describes more satisfactorily the data, or a model that is overfitting by learning non-general information not relevant to describe the data set.

The first scenario is visible in Logsig/Softmax with 80/10/10 split. The number of units increments from 50 to 200 and the validation and testing accuracy follow the same tendency of growth from the training, where the acceptable discrepancy is roughly 5%. However, when the quantity is increased to 500, it is reported different tendencies between the training and the validation/test, since the first increases, while the latter declines, reaching a disparity of 8% (potential overfitting).

Plot 1 illustrates the differences between the training and validation/testing in the local minima, to demonstrate that when the model improves, the difference is roughly maintained, while in an overfitting situation grows. The plost correspond to 50, 200 and 500 units, from left to right, in Logsig/Softmax with 80/10/10 split.

Plot 1 – Difference between training and validation/testing accuracy.



Furthermore, the number of units in which a model overfits the data depends on the transfer functions selected in each layer. For example, in the pair Logsig/Logsig for the splitting 80/10/10 is not registered clear overfitting along the distinct units, since the 3 accuracies increase proportionally (the difference remains at 2%). However, this pattern emerges, because this model has severe difficulties in learning the data.

In general, the best balance between underfitting and overfitting in traingdx is found in 200 units, with 80/10/10 split and logsig/softmax .

b. Layer Activation Functions

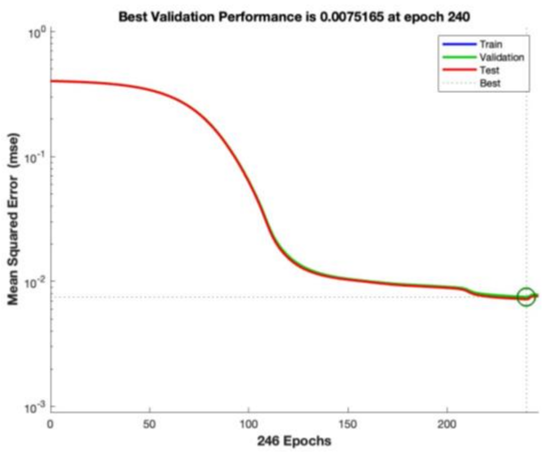
In the Neural Network, the hidden layer is defined by the activation function Logistic Sigmoid (Logsig) in both examples. So, the distinction is present in the output layer by using Logistic Sigmoid (Logsig) or Softmax.

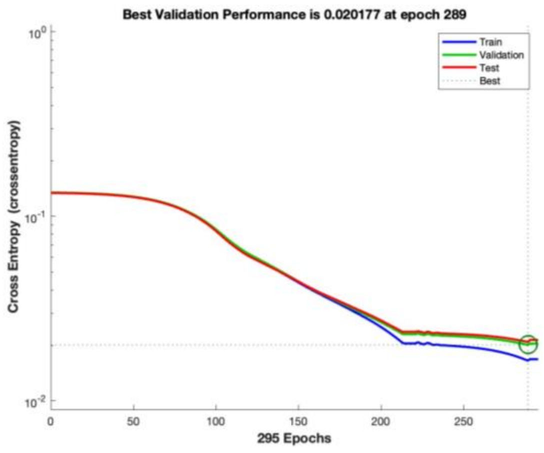
The Logistic Sigmoid converts any number between 0 and 1, by transforming the input linear function into a non-linear. So, according to this description, it is adequate for the hidden layer. However, as the decision boundary in the conversion is linear, meaning that the outcome is a discrete binary value, it is a limited method for the output layer. Usually, it is used to perform the classification of 2 labels.

On the other hand, softmax is an extension of the previous. Instead of individual binary classification, it achieves the output based on probability, between 0 and 1, accordingly to the sum of the perceptrons in the layer. This produces a good function for multi-class classification problems.

Since the present problem is a multi-class classification, the pair Logsig/Softmax outperforms undoubtedly Logsig/Logsig, even though both converge to a local minimum under 300 epochs. Plot 2 exposes the discrepancy in the convergence rate for 200 units and 80/10/10, denoting the extreme difficulties for the training function to learn the data with Logsig/Logsig.

Plot 2 – Difference between Logsig/Logsig (left) and Logsig/Softmax (right).



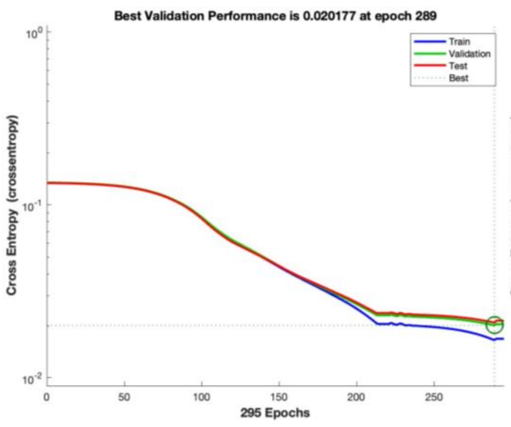


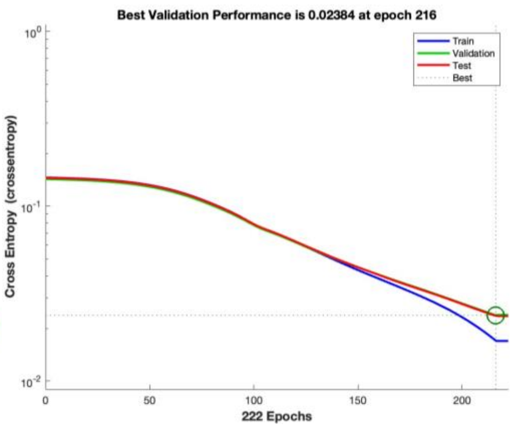
The difficulty to learn the data reflects that with the pair Logsig/Logsig, it is achieved an underfitted model, in which the local minimum found is not as optimal as in Logsig/Softmax.

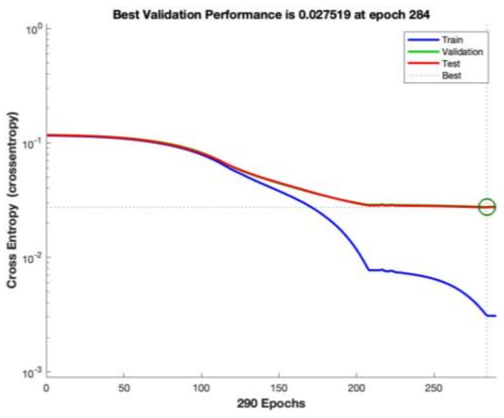
c. Data Split

Relatively to the configuration of the data splitting, the reasoning is similar to point a. With the decrease in the size of the training data, the information to learn is less, so, for the same number of units, it is expected that the training accuracy increases. Consequentially, in the validation/test set the performance of the model might decrease since it is not able to learn all the general relevant information of the data. Plot 4 shows the different types of splitting (80/10/10, 40/20/40, 10/10/80, from left to right) for 200 units, with Logsig/Softmax.

Plot 3 – Influence of the decrement of the training data.







Furthermore, when the variance is high, for example from 80/10/10 to 10/10/80, it is noticeable the overfitting of the model, with better accuracy for the training data and worse accuracy in the test/validation, mainly in Logsig/Softmax. Particularly, for the pair Logsig/Logsig the overfitting is harder to spot, which indicates, once more, the difficulty of the latter configuration to learn the right classifications.

According to the aforementioned, the most satisfactory splitting is 80/10/10, since it provides the majority of the data to the model to learn and then test it in a smaller set.

Overall, the best configurations from points a., b. and c. are 200 units, with Logsig/Softmax as activations functions and a split of 80/10/10 in the data. However, the results are not satisfactory.

3. COMPARING TRAINING FUNCTIONS

In order to widen the scope of the study, it is pretended to compare various learning functions, in which the only fixed parameters are the hidden (Logsig) and output (Softmax) layers, and the 80/10/10 split of the data. The values in table 2 (Appendix A) expose the accuracy values of the following backpropagation algorithms: **traingdm**, Gradient Descent with momentum, **traingdx**, gradient descent with momentum and adaptive learning rate, and **traincgf**, conjugates gradient backpropagation with Fletcher-Reeves updates. The number of maximum epochs was adjusted to 1000 since traingdm needs a lot of steps to converge. However, this value only allows for reaching a more trustworthy value, that in fact does not correspond to the local minimum, which would be over 10000 epochs.

The values in table 2 demonstrate that overall the best model is gradient descent with momentum and adaptive learning rate, reaching an accuracy of 50.5%, without overfitting. Furthermore, depending on the model, the number of units in which the overfit occurs is different. This variation is due to the dissimilar initialization of the weights performed by each training function, leading to a distinct local minimum.

Additionally, it is interesting that traincgf is able to reach the best values in the validation/test phase, even though it appears to overfit. This means that when the model learns the important information of the data plus useless details (overfitting), will not always perform poorly in the validation/test data. However, the invalidity of the model remains. As a last note, between traingdm and traingdx, it is noticeable the impact of the learning rate, since the latter achieves a better local minimum, meaning better accuracy results, at a faster rate.

4. OPTIMIZATION

According to topic 4 the best model tested is traingdx. So at this point, the mode will be more optimized according to the momentum, learning rate and the number of units. The considerations for each parameter are presented in table 3 (Appendix A).

For the values considered, the best combination tested between momentum and learning rate is, respectively, 0.9 and 0.01. This can lead to increments in the accuracy in the validation/test phase of on average 2%. Nonetheless, it is not feasible to infer a preciser answer (including the number of units), since the variation is substantial.

5. CONCLUSION

All the goals initially proposed were achieved with success and will be explained during the conclusion.

Firstly, it was perceptible that increasing drastically the number of units or the high decrement in the training data might lead to poor results, sometimes denoted by overfitting. Furthermore, depending on the activation functions the accuracies contrast significantly since its inner characteristics need to be taken into account along with the problem.

Additionally, the different analyses and optimal parameters might differ depending on the training function. To the optimal model, it is possible to tune the parameters in order to improve the learning process.

Concluding, the results indicate that the models accomplish poor outcomes. Some motivations might be the lack of flexibility of the Neural Network with just 2 layers and the reduced size of the data for the number of classifications required.

6. APPENDIX A

Table 1 – Performance of the training function traingdx.

			Performance of the Training Function: traingdx					
			Logsig/Logsig - MSE			Logsig/Softmax - Cross-Entropy		
			80/10/10	40/20/40	10/10/80	80/10/10	40/20/40	10/10/80
Units	50	Train	0.214	0.215	0.241	0.491	0.535	0.74
		Val.	0.219	0.212	0.236	0.443	0.448	0.413
		Test	0.223	0.207	0.235	0.446	0.447	0.389
	200	Train	0.287	0.263	0.348	0.505	0.581	0.802
		Val.	0.290	0.254	0.281	0.470	0.459	0.400
		Test	0.273	0.258	0.281	0.477	0.458	0.396
	500	Train	0.293	0.306	0.345	0.533	0.687	0.836
		Val.	0.291	0.296	0.289	0.461	0.405	0.387
		Test	0.293	0.297	0.284	0.454	0.408	0.378

Table 2 – Performance of various training functions.

			Performance of the Training Functions Logsig/Softmax - Cross-Entropy - 80/10/10		
			Traingdm	Traingdx	Traincgf
Units	50	Train	0.012	0.491	0.578
		Val.	0.007	0.443	0.508
		Test	0.008	0.446	0.497

	200	**Train**	0.006	0.505	0.812
		Val.	0.006	0.47	0.609
		Test	0.003	0.477	0.601
	500	**Train**	0.192	0.533	0.869
		Val.	0.196	0.461	0.603
		Test	0.200	0.454	0.606

Table 3 – Optimization of the training function traingdx.

			Performance of the Training Function: traingdx			
			Learning Rate 0.1		**Learning Rate 0.01**	
			Mom. 0.8	**Mom. 0.9**	**Mom. 0.8**	**Mom. 0.9**
Units	**100**	**Train**	0.604	0.596	0.567	0.615
		Val.	0.516	0.511	0.501	0.525
		Test	0.508	0.515	0.477	0.542
	150	**Train**	0.525	0.568	0.579	0.602
		Val.	0.463	0.499	0.505	0.515
		Test	0.461	0.500	0.510	0.520
	200	**Train**	0.581	0.565	0.505	0.585
		Val.	0.505	0.503	0.470	0.505
		Test	0.498	0.493	0.477	0.511

	300	**Train**	0.574	0.553	0.583	0.56
		Val.	0.507	0.495	0.506	0.479
		Test	0.492	0.473	0.500	0.493
Average of Val.+Test			**0.494**	**0.497**	**0.493**	**0.511**