Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 1b

Learning with backpropagation and generalisation in multi-layer perceptrons

Nouhaila AGOUZAL, Abdessamad BADAOUI and Nasr Allah AGHELIAS

September 2023

1 Main objectives and scope of the assignment

Our major goals in the assignment were

- Implementation of a two-layer perceptron for classification of linearly non-separable data
- Utilising the previous two-layer perceptron to perform a function approximation using regression
- Develop a multi-layer perceptron network for chaotic time-series prediction.

2 Methods

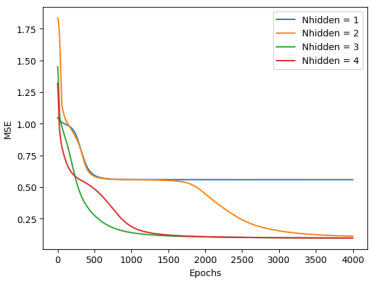
The programming was done in python. We used the numpy library for mathematical calculations and matplotlib.pyplot to generate graphs. To examine more advanced aspects of training multi-layer perceptions, we have used the PyTorch library.

3 Results and discussion

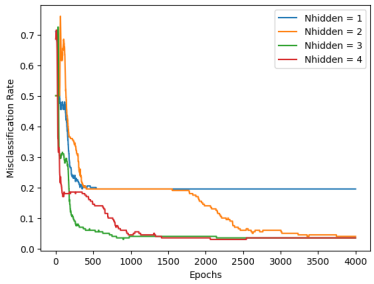
3.1 Classification and regression with a two-layer perceptron

3.1.1 Classification of linearly non-separable data

The number of hidden nodes significantly impacts both MSE and misclassification ratios in training multi-layer perceptrons. With just one node, the model is too simple and ineffective. Two hidden nodes perform better but require more epochs, while using three or four hidden nodes yields good performance with fewer epochs.



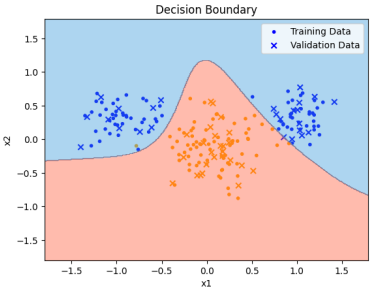
(a) MSE



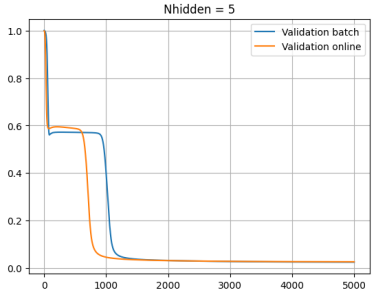
(b) the ratio of misclassifications

Figure 1: the effect of the size of the hidden layer on the performance

The MSE error reveals different error curve behaviors based on training scenarios. In a balanced dataset scenario, both training and validation errors exhibit similar patterns. In biased datasets, validation errors may fluctuate, and in the most challenging scenario, with intentionally biased validation data, the model struggles to generalize, leading to significant divergence between training and validation error curves. We can also notice an increase in validation error because of the over-fitting. An example of the decision boundary is displayed in Figure 2a.



(a) Decision boundary in the first scenario

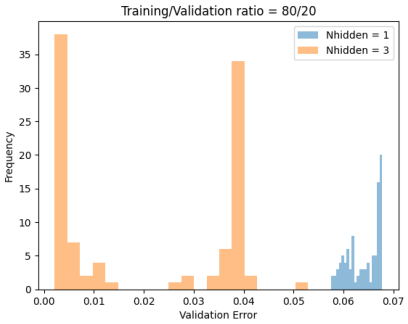


(b) Batch VS Sequential learning with the validation performance

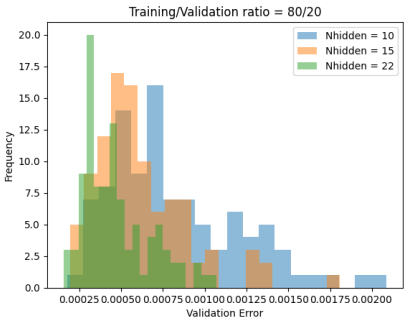
In terms of validation performance, sequential learning converges faster than batch learning with respect to the number of epochs because it updates the model's parameters more frequently.

3.1.2 Function approximation

We used the two layers perceptron defined in the previous section to approximate the bell shaped Gauss function $f(x, y) = e^{-(x^2+y^2)/10} - 0.5$. The input data is a two dimensional grid whereas the targets are the images of couple of points from this grid. We will monitor the performance of our two layer perceptron for different number of nodes in the hidden layer.



(a) Validation Error Histogram for NumUnits 1 and 3



(b) Validation Error Histogram for NumUnits 10, 15 and 22

Figure 3: Validation Error Histogram for different numbers of units

With too few hidden nodes, the model may lack the complexity to adequately capture the intricate features of the Gaussian function. This leads to high bias and limited expressive power, resulting in suboptimal approximation and a validation error that remains relatively high.

Conversely, employing a large number of hidden nodes can enable the model to memorize the training data with a high accuracy. The model becomes overly complex however since the training data perfectly reflects the ground truth the validation error does not seem to be impacted and remains very small. To strike a balance between bias and variance, it's essential to identify the sweet spot in model complexity. Even though the metric we used to identify the best model is the validation error calculated across different number of hidden units, we should be wary of the fact that the more complex models perform better due to the fact that the data does not contain any noise. Thus, we should aim for a slightly less complex model to avoid any issues when faced with new unseen data.

If we take the model of which the number of hidden layers is 22 as our best model and train it with varying the number of training samples, the validation error decreases the smaller the ratio of the training sample from the data set is,

like it is shown in the figure 4

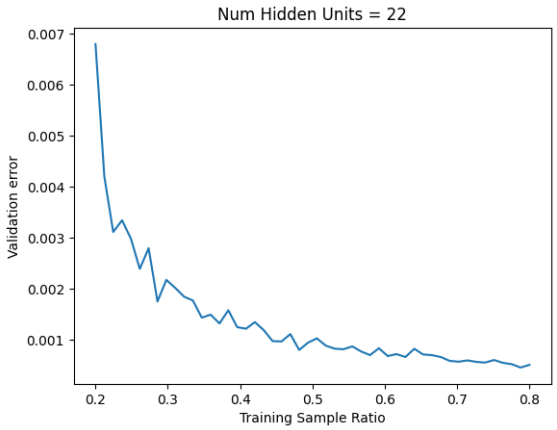


Figure 4: Validation error with respect to the size of the training data

To accelerate convergence without sacrificing generalization, we could use two key strategies. First, learning rate scheduling, which adjusts learning rates during training for faster initial convergence. Second, early stopping, where we halt training based on validation error, preventing unnecessary epochs and optimizing the training process. These combined approaches ensure quicker convergence while maintaining precision and avoiding overfitting.

3.2 Multi-layer perceptron for time series prediction

To approximate this time series, we employed a 3-layer perceptron with early stopping implemented to halt training when the validation error ceases to improve. We utilized the Mean Squared Error (MSE) as the loss function, employed the Adam optimizer, and employed mini-batch learning with a learning rate of 0.01. We conducted a grid search to explore potential values for the number of nodes in the first and second hidden layers, aiming to identify the most and least suitable architectures. To account for the random initialization of weights, we conducted 100 experiments to ensure more reliable and significant results.

3.2.1 Three-layer perceptron for time series prediction - model selection, validation

The conducted grid search resulted in the best model having $nh1, nh2 = (5, 4)$ with a corresponding validation error of 0.071 and test error of 0.175. Conversely, the worst model was $nh1, nh2 = (5, 6)$ with a validation error of 0.089 and test error of 0.179. See Figure 5.

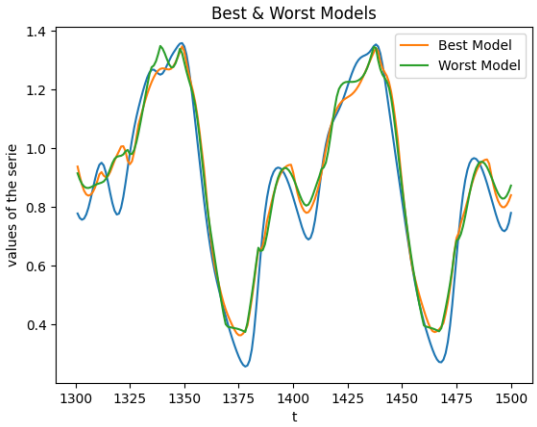
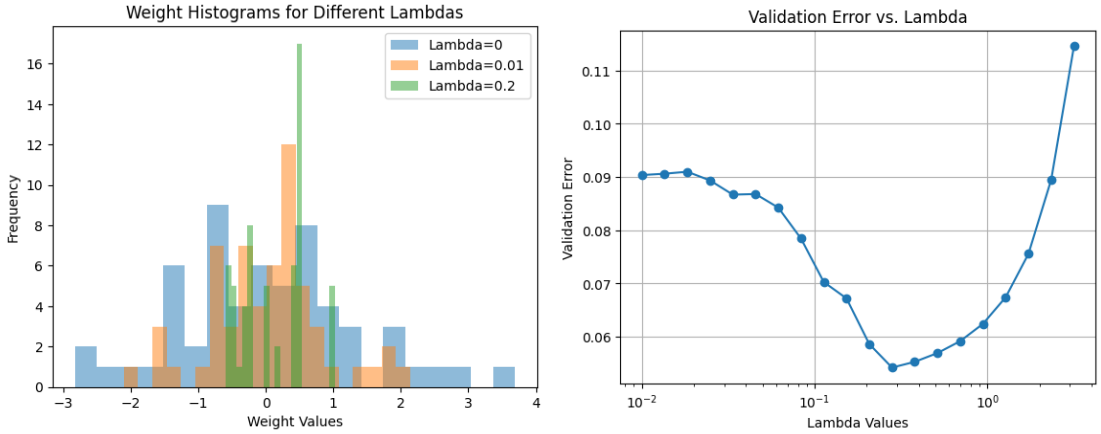


Figure 5: Test predictions along with the known target values.

3.2.2 Three-layer perceptron for noisy time series prediction with penalty regularisation

In the second part of this study, we introduce various levels of noise into the training data. We no longer employ early stopping; instead, we utilize regularization. Initially, all parameters are fixed, and we experiment with different values of lambda to observe its influence on the weight distribution and validation error. The results of these experiments are presented in Figure 6. It is evident that increasing the regularization term causes the weights to converge toward 0, resulting in reduced model complexity.



(a) Distribution of weights with different lambdas

(b) Validation Error vs Regularization

Figure 6: the effect of regularization on the weight distribution and validation error

Based on these findings, we select a regularization term of 0.2. Subsequently, with the number of nodes in the first hidden layer set to 5, we proceed to

determine the appropriate number of nodes in the second hidden layer. Figure 7 illustrates the relationship between the validation error and the number of nodes. Based on this figure, we opt for a value of nh2 equal to 4.

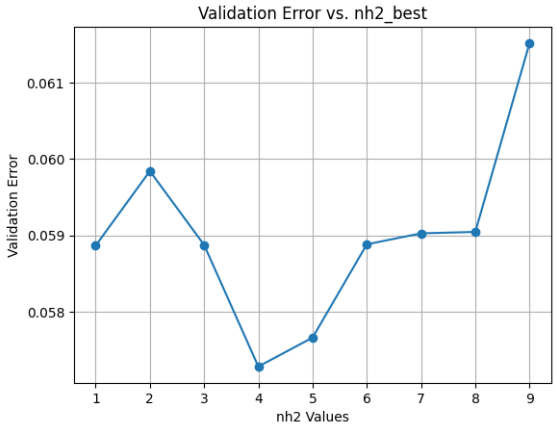
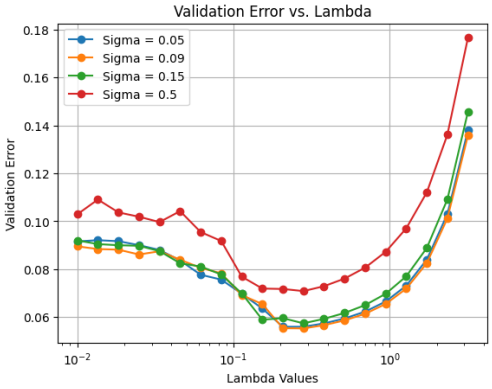
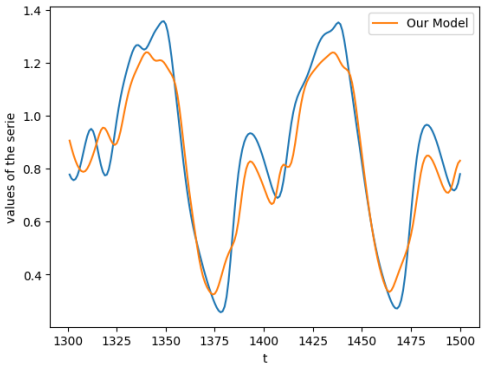


Figure 7: Validation error in relation to the number of nodes in the second hidden layer.

We also examined the interaction between the regularization parameter, λ, and the level of noise in the training data. The main findings are presented in Figure 8a. We observed that as the noise level in the data increases, the plot shifts vertically upwards, while the region where the validation error typically reaches its minimum remains consistent. This shift may be attributed to the increase in the irreducible error, resulting from the higher noise variance in our data. To sum up, The results of our final model, trained on this noisy data with (nh1, nh2) set to (5, 4) and a regularization term of 0.2, are displayed in Figure 8b. The test error for this configuration is 0.173.



(a) Effect of Regularization Parameter λ on Validation Error with Varying Noise Levels



(b) Performance of final model on noisy data ($\sigma = 0.05$) : $(nh1, nh2) = (5, 4)$ with $\lambda = 0.2$.

4 Final remarks

One aspect of the lab that caught our attention was the necessity to manually code a multi-layer perceptron and its learning process initially. This hands-on approach significantly enhanced our understanding of the fundamental mechanisms behind it. Moreover, we delved into the usage of regularization techniques to manage model variance effectively. Through this, we experienced firsthand that overly complex models tend to compromise optimal generalization capabilities.