AI and AI PROGRAMMING

Assignment : MATLAB Simulation with Neural Network Toolbox

May 8, 2020

Breast Cancer Prediction Project.

By Maxence Boels

Professor Terry Windeatt

1 Contents

2 Executive Summary

This report compares the performances of shallow neural networks for breast cancer classification as either benign or cancerous tumors. The report is structured into 3 experiments, each focusing on a specific task and can be described as follows. First, the complexity of neural networks can be defined by its number of layers and neurons. Thus, the first experiment suggest the optimal number of neurons and training epochs for the neural network. Secondly, combining classifiers predictive abilities as an *Ensemble* of classifiers, and using a majority vote to attribute a class, decreases the classification error. As third experiment, different types of optimizers are tested and described, to understand how they work and their critical role in backpropagation.

3 Introduction

Nine features have been extracted for each mammogram to form a dataset of 699 observations labelled as either cancerous or not cancerous. Thus, it is a binary classification problem with cross-entropy as the loss function and the softmax function as the output layer providing a probability for each class. The default optimizer for backpropagation is the Scaled Conjugate Gradient (SCG) algorithm. In this experiments, no validation sets have been prepared since the experiments are only run once, and the objective is to evaluate the trained models without fine tuning the parameters.

4 Experiment I: Nodes and Epochs

The first experiment focuses on finding the optimal number of *nodes and epochs* for the neural network classifier.

First, the Scaled Conjugate Gradient backpropagation optimizer was selected to update the weights and bias values. Then, no validation set is used, thus equally splitting the dataset into train and test sets. For this experiment, the selected node values were 2, 8 and 32 and epochs 4, 8, 16, 32 and 64 so having 15 combinations.

As depicted on Figure 4.1, the training error converges towards zero for all nodes cases. This is due to the increasing complexity of the model on the training set at each epoch, which ends up learning all the observations and reduces the generalization ability of the neural network. It is necessary to use a validation set to know where to stop the learning process at a certain epoch before the model start overfitting the training data. Based on the plots, it is recommended to select 32 as the ideal number of nodes since it has the lowest error and standard deviation rate.

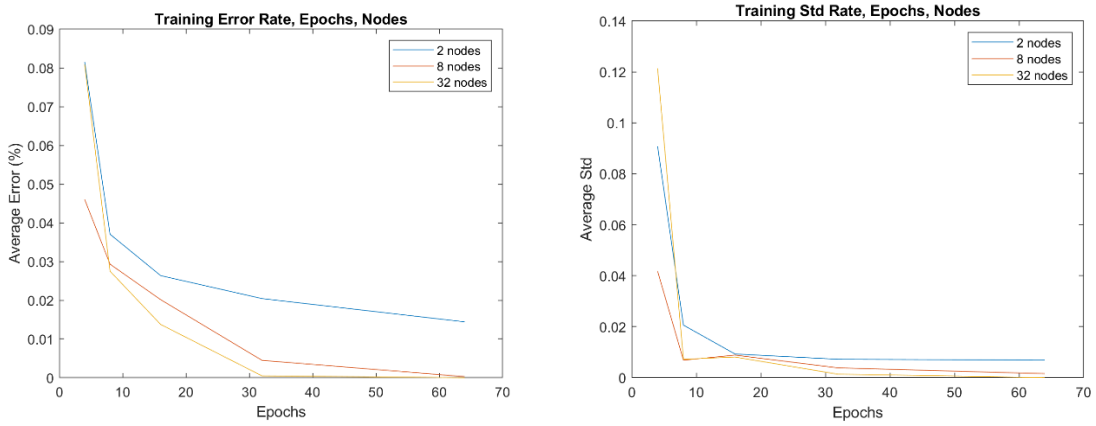


Figure 4.1 - Training Error and Train STD error rate v epochs and nodes

As epochs increase, the test error rate reaches a minimum and then start increasing. This is illustrated in Figure 4.2 as a sign of overfitting of the model on the training data. The testing shows a good performance of 8 and 32 nodes at 8 and 16 epochs, but start overfitting after it. Whereas, networks with 2 nodes perform better with more epochs, as 2 nodes reduces complexity of the model function and thus, start overfitting later.

Regarding the standard deviation, all nodes reach a plateau after 8 epochs which indicate that the error rate is more reliable.

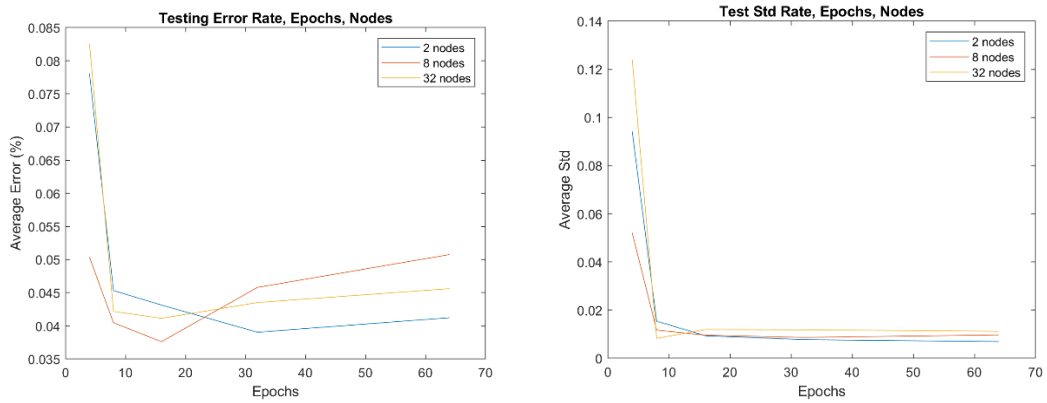


Figure 4.2 – Testing Error and Test STD rate v epochs and nodes

To conclude, the classifiers show a good performance on the training set with almost no misclassification as the number of nodes increases. However, one can observe overfitting on the test set with more epochs during training. In this case, it is recommended to use early stopping parameters.

5 Experiment II: Ensemble of Classifiers

Based on the train and test classification errors in experiment 1, a number of nodes and epochs have been selected to minimize the error.

In this experiment, multiple individual classifiers are combined to form an ensemble with majority vote to classify the tumor as either benign or malignant. Thus, the objective is to verify if ensembles improve the classification error by comparing their results with individual classifiers, and also to find the optimal number of classifiers per ensemble while changing the number of nodes and epochs.

5.1 Finding the best Ensemble Size

Thus, for this experiment, *8 nodes and 16 epochs* have been selected as the optimal combination that gives a trade-off between model complexity, overfitting and generalization ability.

As depicted on Figure 5.1, the training and test errors both improve by increasing the ensemble size with the minimum for training at 21 classifiers per ensemble.

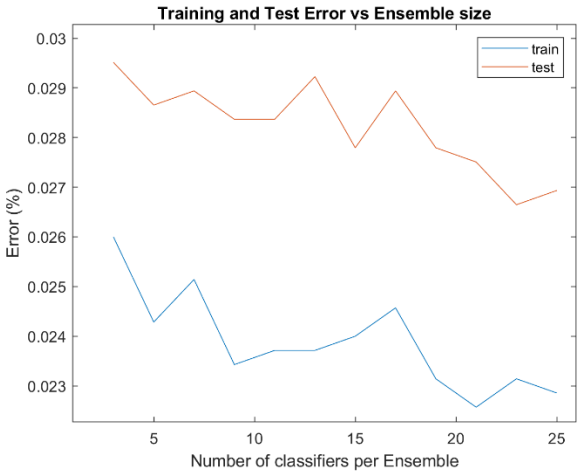
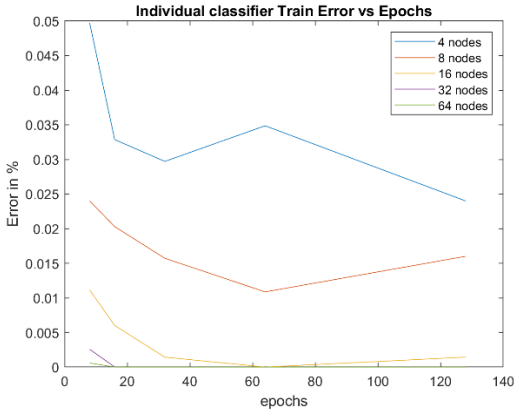


Figure 5.1 - Train and Test Error vs Ensemble size

5.2 Individual Classifiers vs Epochs and Nodes

Let's consider now the average individual classifiers performances on training and test error. Those classifiers are using 'trainscg' optimizers with random shuffle on the training and test sets to have independent classifiers at each run, then taking the average error for 10 independent results. Figure 5.2, depicts a smaller error with increasing nodes during training whereas, fewer nodes perform better during testing. A trade-off between model complexity and overfitting needs to be decided.



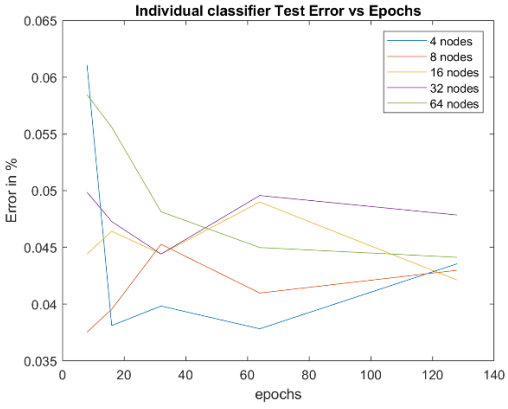


Figure 5.2 - Individual Classifiers train and test errors

5.3 Ensemble size vs Epochs and Nodes

For this final sub-experiment, we seek the best combination of epochs and nodes for a given ensemble of size 23, as it was found to be the near a local optimum in section 5.1.

Figure 5.3, illustrates error rates for an Ensemble of classifiers on the training and test sets with different number of nodes and epochs. One can observe the curbs plummeting after a few epochs and leveling off, for both training and testing classification errors.

As depicted on the graphs, for the training the optimal number of epochs is 21 with 12 nodes. Whereas, for testing the lowest classification error has 22 nodes and around 21 epochs which indicates a local optimum.

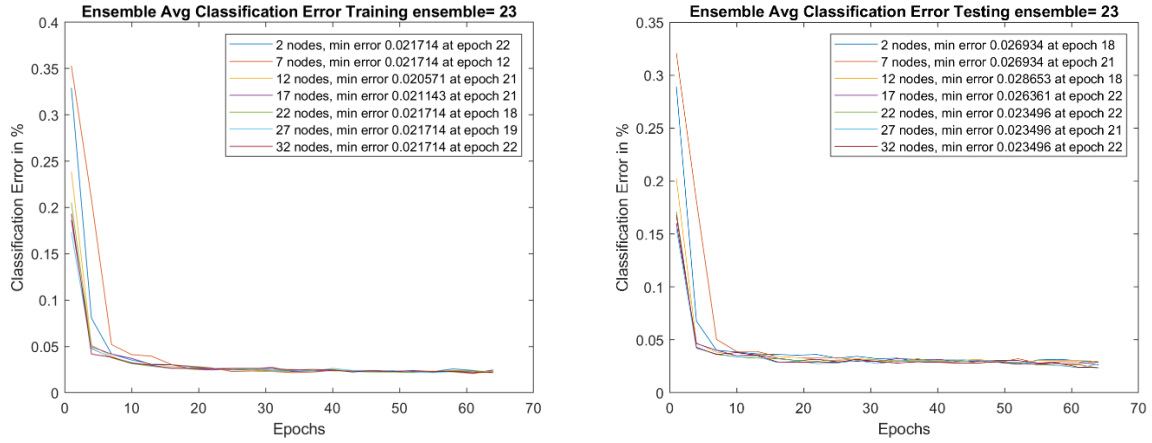


Figure 5.3 - Ensemble Avg Classification Error on training and testing sets

6 Experiment III: Optimization algorithm for backpropagation

The objective of the third experiment is to try different optimizers i.e. 'trainlm' and 'trainrp' which are Levenberg-Marquardt backpropagation and Resilient backpropagation, respectively.

6.1 Levenberg-Marquardt backpropagation algorithm

The *Levenberg-Marquardt backpropagation algorithm* is a weight and bias update function, proven to be fast although it requires a lot of memory. This technique relies on a Mean or Sum of Squared Error function as it uses the Jacobian for calculations.

This algorithm can be described as first, the M-L algorithm is derived from the Newton's method that was designed for minimizing functions that are sums of squares of nonlinear functions [1]. Thus, the Hessian can be computed with the first order derivatives of the weights as the gradient of the Hessian matrix. Secondly, in the L-M algorithm, the error function is minimized, while the step size is kept small to ensure linearity with λ as parameter controlling the step size [2]. The key step is the calculation of the Jacobian matrix. In a nutshell, the *derivatives of the squared errors* are computed as in standard backpropagation.

6.2 Resilient backpropagation algorithm

The next optimization technique selected for this third experiment is the *Resilient backpropagation algorithm*.

'The Resilient Propagation weight update rule was initially introduced as a possible solution to the "vanishing gradient" problem: as the depth and complexity of neural networks increase, the gradient propagated backwards by the standard Stochastic Gradient Descent (SGD) backpropagation becomes increasingly smaller, leading to negligible weight updates, which slow down the training considerably' [3]. For example, the gradient near zero in hidden layers caused by the sigmoid function for very large or small values results in the "vanishing gradient" problem i.e. the magnitude of the slope has no effect on the weight update.

Therefore, the Resilient Propagation fixes this problem by introducing a fixed weight update value δ_{ij} *increased or decreased by a* multiplicative factor *η+* or *η-* to skip flat regions of the gradient. This is performed by "backtracking" the change in sign of the gradient for the 2 last iterations [3].

6.3 Comparing Gradient Descent and L-M optimizers

As depicted on Figure 6.1, the *Levenberg-Marquardt (LM) backpropagation* optimizer performs significantly better than the *Gradient Descent* backpropagation algorithm for all Ensemble sizes. Moreover, the L-M optimizer has a smaller variance of classification error which makes it more stable (on this data set).

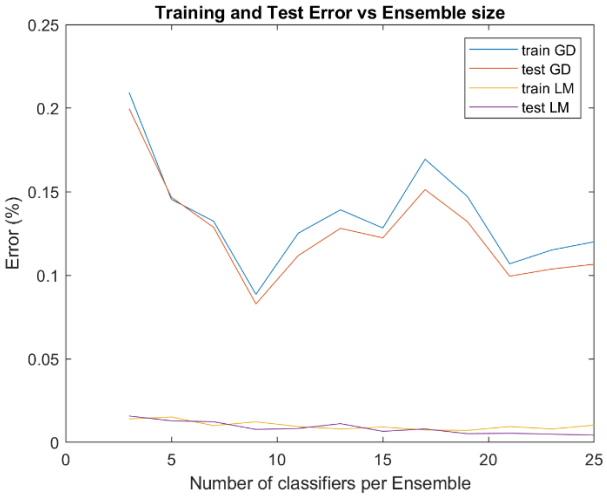


Figure 6.1 - Optimizers Train and Test Errors

In a nutshell, there is no 'one fits all' technique, but instead many different types of optimizers adapted for certain optimization problems. Those backpropagation algorithms are often improved for better performances at state-of-the-art conferences with new papers submissions. Thus, it is advised to stay aware of the many possibilities when choosing optimizers for neural networks and test them according to their pros and cons.

7 Conclusions

To conclude, this report explores 3 experiments with the objectives of having a better understanding of how neural networks work, by fine tuning their hyperparameters and combine their predictive abilities. As a result, those experiments raises a few important observations to remember. First, the number of *nodes and epochs* increases the complexity of the model and need to be restrained to avoid overfitting on the training set which leads to poor generalization performance on the test set. Secondly, by combining neural networks predictions and selecting the class with a majority vote, *ensembles* are able to reduce the classification error, and increasing the ensemble size drops the error until a plateau is reached. The third and final experiment highlights the role of *optimization algorithms* during the backpropagation and their effect on performance. It is suggested to try different optimizers since they all have strengths and weaknesses depending on the fitness landscape, thus it can be considered as a parameter to fine tune. As future work, it should interesting to compare the results with medical imaging i.e. mammograms while using transfer learning on small datasets.

8 References

[1] Y. X. Y. X. J. Z. F.-Y. W. Chen Lv, "Levenberg-Marquardt Backpropagation Training of Multilayer Neural Networks for State Estimation of A Safety Critical Cyber-Physical System," in *IEEE Transactions on Industrial Informatics*, 2017.

[2] D. A. M. K. S.Sapna, "Backpropagation Learning Algorithm Based on Levenberg Marquardt Algorithm," in *The Fourth International Workshop on Computer Networks & Communications*, 2012.

[3] G. D. M. Alan Mosca, "Adapting Resilient Propagation for Deep Learning," London, 2015.