Project two, FYS-STK4110

Knut Hauge Engvik

November 10, 2019

Contents

Chapter 1

Introduction

This project aims to compare different approaches to binary classification and regression in machine learning. Specifically a comparison of linear regression and logistic regression versus neural networks.

1.1 The data sets

There were two proposed data sets for classification, the Wisconsin Cancer data(WCD) and the Taiwan banking data(TBD), both of wich presented challenges. The Wisconsin Cancer data was small and consisted of various statistics of the raw data. I could not find the raw data and thus opted to use the much larger Taiwan banking data set.

The Taiwan banking data set consists of a range of predictors. Amount of credit given, marital status, gender, education, age and six months of payment history. The data had several thousand data points containing undefined values. One could remove these data points resulting in a severely reduced data set. One could also choose to remove the predictors where undefined values occur, losing out on any predictive value these traits may contain. One could also opt to leave the data set as is, letting the model try to figure out how to deal with these values.

A priori it would seem natural that the best predictors of a default would be previous failures to pay on time and it would be interesting to see if one would obtain similar results using only these predictors as when using the full data set. Another reason one might want to focus on fewer predictors is that the remaining data set would contain fewer undefined values.

For the comparison of neural networks with the regression methods used in project one, Frankes function was chosen as the data set. This choice was made as a comparison of performance on this function seemed the most straight forward. The training set was made using a 100 by 100 grid and adding noise drawn from a normal distribution of mean zero and standard deviation 0.1. The testing set was made using a 200 by 200 grid such as to create unseen data points.

Chapter 2

Methods

In the initial phase of the project, general purpose scrips for logistic regression and neural networks were developed. The regression script was tested on the Iris data set and generated moons data set from scikit learn. The neural network was tested on the ever popular digit recognition data set, also from scikit learn. While these tests were mostly functional, and not optimized with regards to hyper parameters, both scripts seemed to perform reasonably well. For instance, the neural network consistently achieved accuracies between 96 and 99 percent.

A preliminary rough analysis was performed on the TBD using both logistic regression and neural networks to determine how to handle the challenges posed by this specific data set. Two things in particular stood out. Firstly, while varying what predictors was included, it was found that, for both logistic regression and the neural network, using only columns six through eleven yielded slightly better(higher accuracy) and more stable results. These columns correspond to whether or not the subject had payed on time, and if not, how long the delay was, for a period of six months. Thus, the decision was made to use only these predictors for the remaining analysis. Unfortunately these predictors contained 6561 undefined values leading to a substantial reduction of the data set after cleaning the data.

Secondly, when looking at the accuracy achieved, it was suspiciously close to the percentage of subjects who did not default (77%). Looking at confusion matrices it was noted that while both methods did learn, they tended to learn the rule "Nobody defaults". Intuitively one might expect that skewed categories would present deep local minima in predictor space. Consider if one were to start in a sate of "Nobody defaults". If most subjects do not default, then most changes to the state of the model would tend to lower the accuracy. To combat this, it should be possible to use a skewed cost function where the penalty for a false negative is larger than a false positive, however this was not attempted. Instead two other ideas were tested. An attempt was made using a smaller sample of the data with an equal number of subjects from each category as a training set and testing on the full set. Setting all biases to one, forcing the network to start in an "Everyone defaults" state, was also attempted.

Another problem with skewed data is how to assess model performance. As the simple "Nobody defaults" model achieves an accuracy of 77%, this might not be the best metric. Thus, a balanced accuracy metric, where one takes the average of the percentage of correct classifications for each category, was

implemented.

To find hyper parameters a grid search algorithm was implemented. Following a broad grid search parameters were tweaked manually.

For classification problems the cost function used was cross entropy. This choice was made after reading Nielsens[1] explanation of how this function leads to faster learning when the model is off target. For the fitting of Frankes function it seemed most appropriate to use the squared error as a cost function. In both cases varying degrees of L2 regularization were attempted.

2.1 Results

For the logistic regression the effect of learning rate was explored first. Plots 2.1 and 2.1 show accuracy for various learning rates, without and with regularization respectively. From these plots it seems like the largest effect of the learning rate, especially when not using regularization, is convergence time and the ability to escape local minima. Note that higher learning rates were tested, but they showed erratic behaviour and failure to converge.

To explore the effects of regularization, a similar approach was followed. Using high(1) and low(0.01) learning rates various L2 penalties where tested. The results can be seen in figures 2.1 and 2.1 respectively. There did not seem to be any obvious effect from regularization. Thus, as there was little to be gained in accuracy, the optimal hyper parameters would seem to be a learning rate of one and no regularization, as this converges faster. Using these parameters on a validation set an accuracy of ≈ 0.80 and a balanced accuracy of ≈ 0.64 was achieved. This is inline with the results obtained on the testing set, and indeed on the training set. The highest observed accuracy was ≈ 0.83, but this was an (almost) singular event.

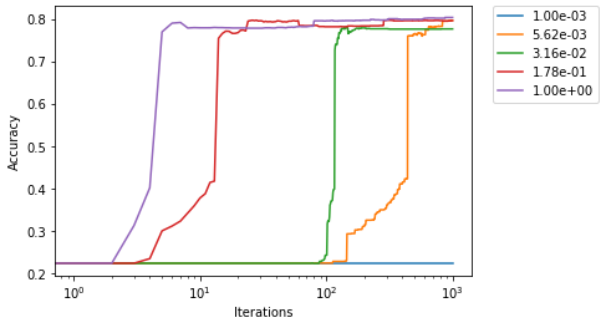

Figure 2.1: This plot shows accuracy for various learning rates as a function of gradient steps using logistic regression. No regularization

The neural net did somewhat better and with optimized hyper parameters could regularly achieve accuracies of ≈ 83 percent. With more parameters to tune it was decided to design a grid search algorithm that, rather than plot results, produced excel tables where one could study performance of various parameters. These files can be found at the github repository[2]. From these tests it seems that none of the attempts to combat the skewedness were successful. When using a balanced training set there was a marked increase in balanced

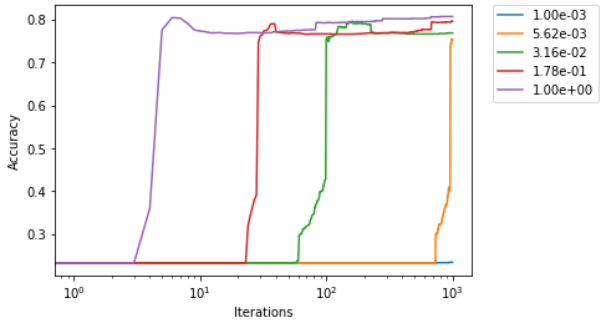

Figure 2.2: This plot shows accuracy for various learning rates as a function of gradient steps using logistic regression. Regularization factor 0.01

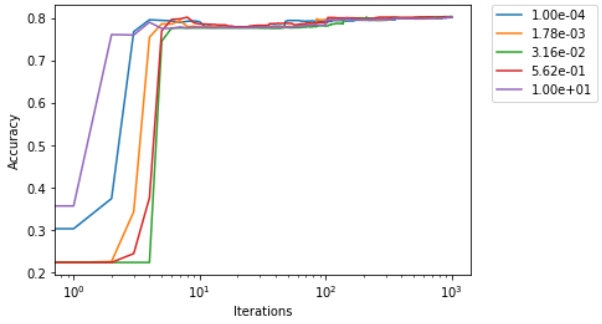

Figure 2.3: This plot shows accuracy for various regularization(L2) as a function of gradient steps using logistic regression. Learning rate 1

accuracy ($\approx +4$ percent), but at the expense of accuracy (≈ -10 percent), when compared to training with the unbalanced training set. Setting the biases to one did not seem to give any noticeable effect.

The best performance observed during grid searches were for learning rate 1 (highest), batch size 5 (smallest), 100 hidden neurons (highest), 50 epochs (highest) and no regularization. Except for the learning rate these were not clear winners and certain combinations seemed to work better in conjunction. For instance few epochs seemed to do better with larger batch sizes. As all the "winning" parameters were extremal, it seemed prudent to do some further testing beyond these values. This did not lead to any improvement. Further increases in learning rate lead to unstable convergence, more neurons or layers lead to increased training times with no increase in accuracy as was also the case for increasing the number of epochs.

Using the optimized hyper parameters a final test was run for both a balanced and an unbalanced training set. The result can be seen in figures 2.1 and 2.1.

For evaluation of performance on Frankes function, the chosen metrics were the coefficient of determination (R2) and the mean squared error (MSE). A grid search was performed[2] and the best parameters seemed to be the same as for the TBD. An attempt was made to tweak the parameters around these values,

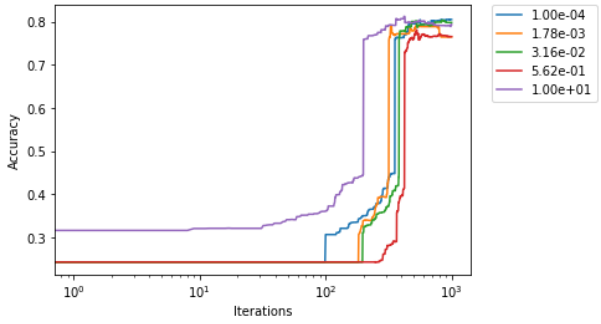

Figure 2.4: his plot shows accuracy for various regularization(L2) as a function of gradient steps using logistic regression. Learning rate 0.01

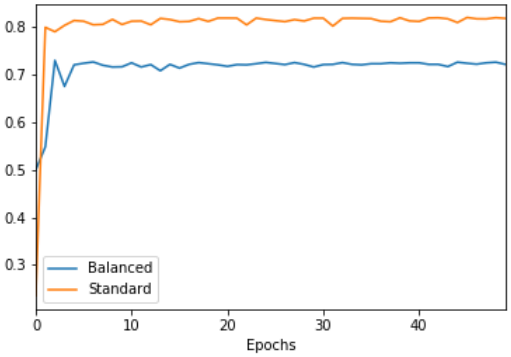

Figure 2.5: This plot shows accuracy as a function of epochs for neural network trained on a balanced and an unprocessed(standard) dataset

but no further improvements could be made, except when increasing number of epochs. Indeed, when training over 500 epochs the model achieved an MSE score of 0.002 and an R2 score of 0.972. Plots 2.1,2.1,2.1 and 2.1 show MSE and R2 scores for the optimized parameters, tested on a finer grid than used for training.

2.2 Evaluation

On the whole the neural network performed as well or better than the logistic regression and linear regression algorithms. After optimization it outperformed the logistic regressor, particularly when looking at the balanced accuracy and seemed about equal on linear regression problems. On the linear regression problem the neural network was presented with what I would imagine is a more difficult task, as it was tested against a much finer grid than the linear regression. It also seemed to be still improving after 500 epochs and could potentialy beat out the linear regression algorithm if trained further. However, it was trained on a larger data set, potentially giving an advantage in that regard. In hindsight it

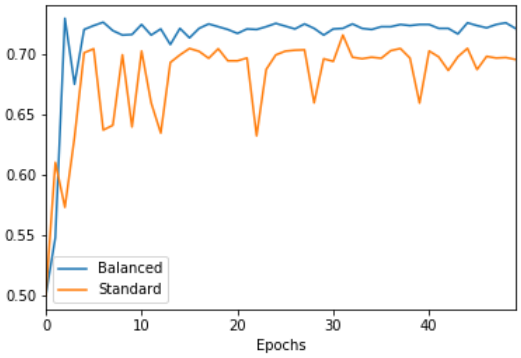

Figure 2.6: This plot shows balanced accuracy as a function of epochs for neural network trained on a balanced and an unprocessed(standard) dataset

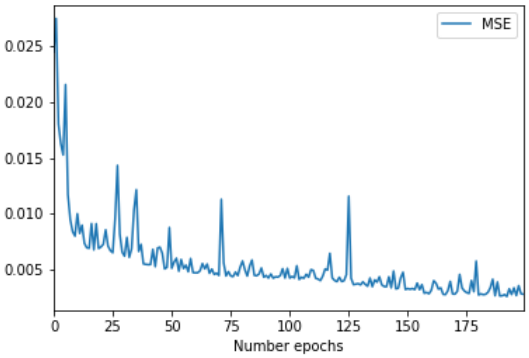

Figure 2.7: This plot shows MSE evolution over 200 epochs using optimized parameters.

would perhaps been better to use the same exact data set for easier comparison. The main strength of the neural network though, seems to be its flexibility. It required very little to adapt it to new challenges.

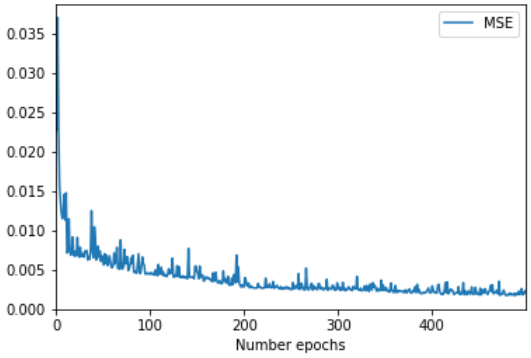

Figure 2.8: This plot shows MSE evolution over 500 epochs using optimized parameters.

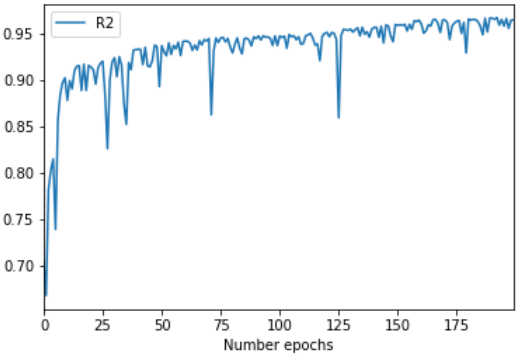

Figure 2.9: This plot shows R2 score evolution over 200 epochs using optimized parameters.

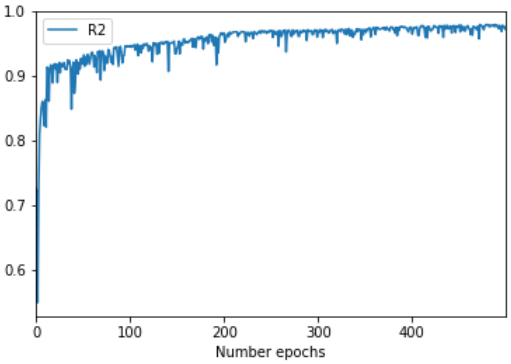

Figure 2.10: This plot shows R2 score evolution over 500 epochs using optimized parameters.

Bibliography

[1] Michael Nielsen *Deep learning and neural networks.* http://neuralnetworksanddeeplearning.com/chap3.html

[2] Knut H. Engvik *Projecttwo.* https://github.com/KnutFys/projecttwo

List of Figures

List of Tables