Credit Approval Modelling with Neural Networks

Jake Warby - z5259166

---

1 Introduction

Within this paper the credit card application data set will be explored and preprocessed as part of the modelling process. Neural networks will then be used to predict the outcome of credit applications, with the quality of these predictions being used to compare performance of two gradient optimizers, SGD and Adam, as well as the regularization techniques dropout and L2. By the end of this paper predictor/response relationships will be explored, the generalisation improvement of regularization techniques will be quantified and the training capability of different optimizers will be explained.

2 Data

2.1 The Dataset

The credit approval data, provided by Dua and Graff (2019), contains 690 instances of applications and subsequent approval/disprovals. The features are unnamed, consisting of 9 categorical and 6 continuous.

2.2 Pre-processing

To make the data neural network ready, the response vector is one hot encoded and the categorical variables are ordinal/integer encoded. There are 37 instances with missing values (5%). Due to the small proportion of missing instances and a majority of missing values being categorical, these instances were removed from the data set.

During training there will be random 50/50 train/test splits, where the train features will be min-max scaled based on the relative training data. Note that successful applications are labelled '1' and unsuccessful applications are labelled '0'. A sample normalized dataset is saved as data.csv in the attached documentation.

2.3 Exploration

Before modelling it is beneficial to do general data exploration to gain initial insights on the distribution and relationships of the features and response. The distribution of the response is visualised in Figure 1.

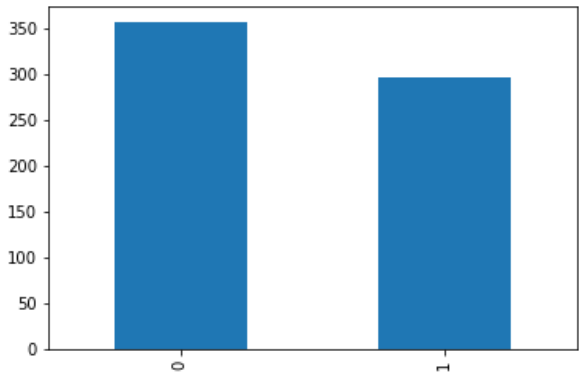

Figure 1: Response Distribution

305 instances are approved (1) and 383 instances are declined (0). This accounts for 44.5% and 55.5% of the data respectively.

The correlation heatmap between the continuous features and response is presented in Figure 2.

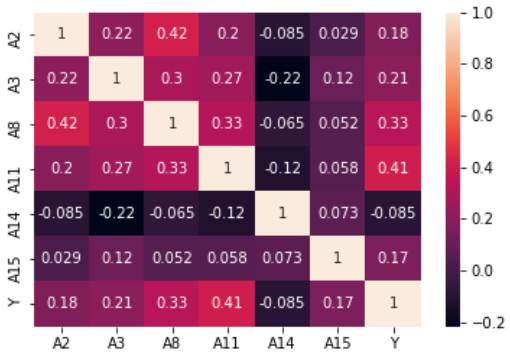

Figure 2: Correlation Heatmap

The 2 numerical features that are most correlated with the response are A11 and A8, while the least correlated was A14. We can also determine there are no extremely collinear predictors.

The distribution of the numerical predictors, grouped by whether the response was approval or not, can be seen in Figure 3. Note that Approval/1 is coloured orange and Declined/0 is coloured blue.

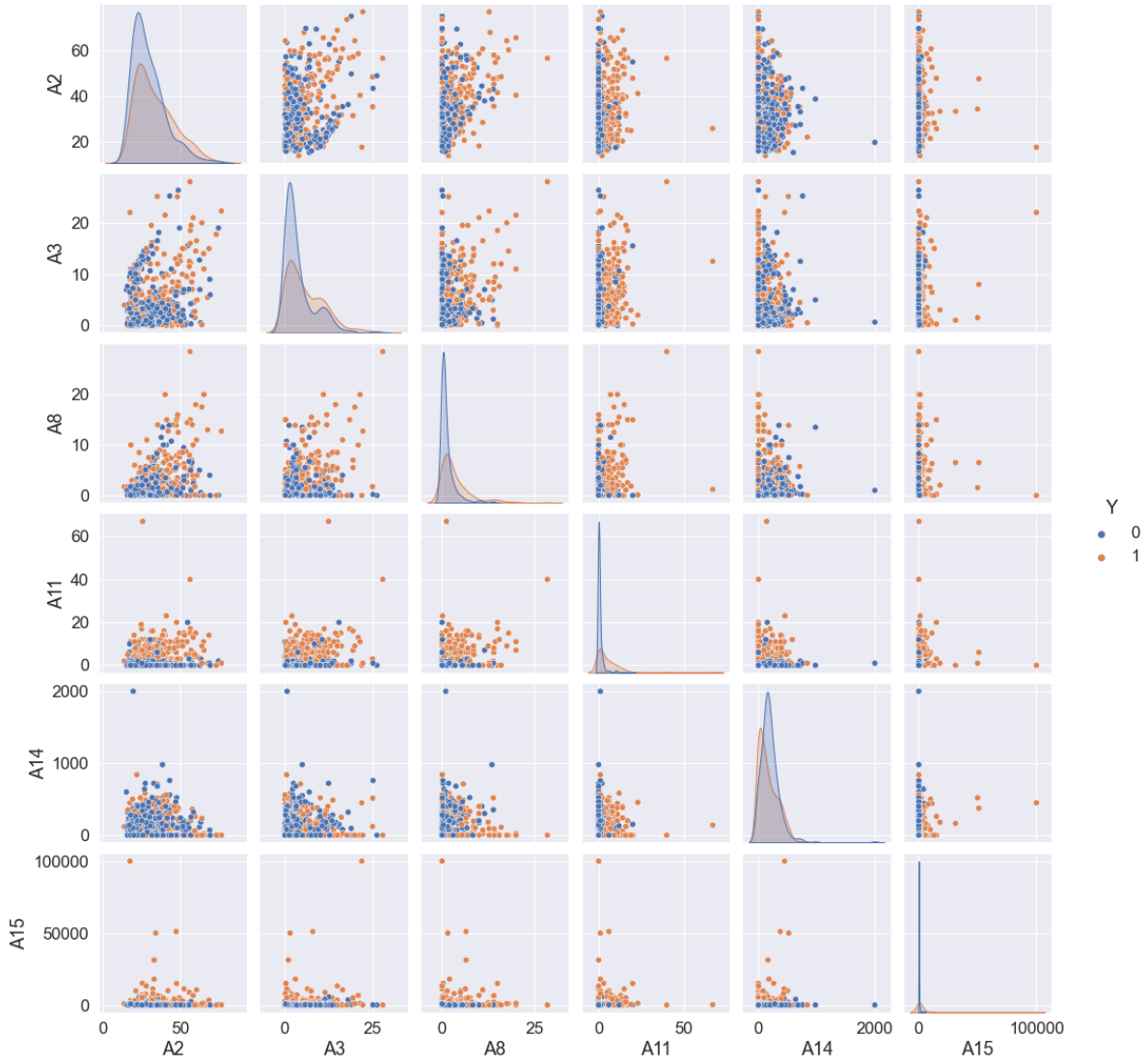

Figure 3: Continuous Variables

This plot accentuates the difference in distribution of the continuous features for the different outcomes. It is particularly obvious for the two most correlated features A11 and A8 which have distinct modes. The scatterplots between predictors also uncover interesting relationships, such as the approvals often being grouped in both dimensions.

The proportion of approval for each level of the categorical variables can be seen in Figure 4. Note that Approval/1 is coloured orange and Declined/0 is coloured blue.

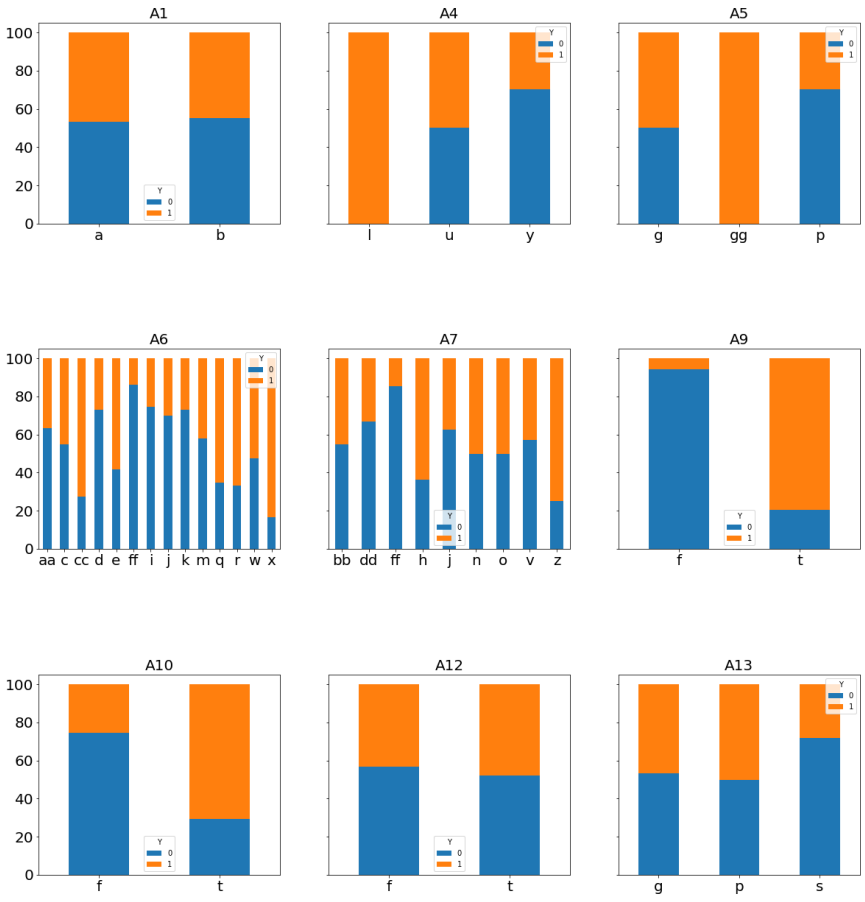

Figure 4: Categorical Variables

These plots show that some categorical features are highly capable of distinguishing between 1/0, such as A4,A5 and A9.

2.3.1 Exploration Summary

Figure 1 shows that the response class distribution is mostly balanced, therefore no sampling changes must be done. Further data exploration has made it obvious that there is a clear relationship between many of the predictors and the response that can be modelled. This can be seen through the high correlation in Figure 2, particularly between A11 and A8 and the response. The pair plot in Figure 3 shows that the distribution for different response classes are distinct for each variable, and a similar thing is seen through the proportionality of the stacked bar charts in Figure 4, particularly for A4,A5 and A9.

3 Model Considerations

3.1 Gradient Methods

One of the important hyperparameter choices for neural networks is the gradient optimizer. Recent advancements have lead to faster and more effective gradient optimizers, which has increased the feasibility and popularity of deep neural networks. In this paper we will explore Stochastic Gradient Descent (SGD) and the new Adaptive moment estimator (Adam), as introduced by Kingma and Ba (2016). Further explanation for these and other contemporary optimizers can be found in Ruder's 2017 paper.

3.1.1 SGD

SGD utilises small batches of randomised training data for computation. For each epoch, the gradient and weight updates are computed with respect to these batches.

$$\theta_{t+1,i} = \theta_{t,i} - \eta \Delta_\theta J(\theta, \text{sample})$$

Where $\theta_{t,i}$ is the ith weight at time t, η is a user defined learning weight and $\Delta_\theta J(\theta, \text{sample})$ is the gradient computed for the minibatch.

3.1.2 Adam

Adam aims to adapt the learning rate by utilising exponentially decaying averages of past gradients.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

Small adjustments are made to correct for bias towards 0 in the initial time steps.

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} * \hat{m}_t$$

Where β_1 (typically 0.9) and β_2 (typically 0.999) are user defined decay rates, g_t is the gradient at t, v_t and m_t are the estimates of the estimates of the first and second moments of the gradients and ϵ is a small scaler used to prevent division by 0.

3.2 Regularization

3.2.1 L2 Weight Decay

A popular method to avoid over fitting in neural networks is weight decay, first proved effective by Krogh and Hertz (1992). A penalty term is added to the normal weight update to help prevent the weights from getting too large as training progresses.
The original weight update:

$$w_i \leftarrow w_i - \eta \frac{\partial E}{\partial w_i}$$

The regularized weight update:

$$w_i \leftarrow w_i - \eta \frac{\partial E}{\partial w_i} - \eta \lambda w_i^2$$

Where λ is a user defined decay constant.

3.2.2 Drop out

Node drop out works to reduce over fitting by randomly dropping nodes, with a set probability, during training. This ensures that the neural network does not become too dependant on each node which in turn helps improve the network's generalisation quality. The effectiveness of this simple strategy was proved by Srivasta et.al (2014).

4 Model Results

4.1 Model Setup

Each experiment will be ran 50 times, with neural networks trained with the SGD and Adam optimizers, for 200 and 25 epochs respectively. These numbers are slightly above the max epochs taken to converge in trial runs (195 and 22 respectively). Therefore, to allow time for convergence, but not too much time for overfitting, they are trained with their respective epoch amounts. A single relu activated layer is used with 50 nodes for training as this was best performing in trial runs. All user defined parameters are defined in Table 1.

Optimizer	Learning Rate	Layers	Neurons	Activation	Epochs
SGD	0.01	1	50	Relu	200
Adam	0.01	1	50	Relu	25

Table 1: Model Parameters

4.2 Experiments

4.2.1 Experiment 1

The first experiment was to compare the performance of the SGD and Adam networks with no regularization. These results can be seen in Table 2. Note these numbers are the average of the 50 runs.

Optimizer	Train Accuracy	Test Accuracy
SGD	0.874	0.865
Adam	0.909	0.855

Table 2: Experiment 1 Results

Table 2 shows greater accuracy on the training sets compared to the test, which is to be expected as the test set is unseen during the training process.

Table 2 also shows that while Adam fits the training set better, the SGD optimizer is able to generalize better with respect to this data. However, Adam was must faster with converging, training for only 25 epochs compared to 200, so for larger neural networks Adam is often preferred, even if the results with SGD could be better (this is dependent on the problem).

4.2.2 Experiment 2

The second experiment was to compare the performance of Adam with no regularization, dropout regularization and l2 regularization. Parameters were chosen as the best from trial runs. The results for regularization and no regularization can be found in Table 3. Note these numbers are the average of the 50 runs.

Optimizer	Regularisation	Reg Parameter	Train Accuracy	Test Accuracy
Adam	L2	0.01	0.875	0.860
Adam	Dropout	0.5	0.894	0.863
Adam	None	N/A	0.909	0.855

Table 3: Experiment 2 Results

Table 3 shows that regularisation techniques, such as dropout and l2, typically lead to an improvement in the generalization ability of a model (measured by test accuracy), which we can assume is a result of less overfitting due to the decrease in train accuracy when compared to no regularization. In regards to this experiment, the dropout regularization method had the best generalization results out of the regularizers, with 0.3% higher test accuracy than L2. However, a choice of either regularizer is an improvement over no regularization for generalization, with 0.8% and 0.5% improvements in test accuracy for dropout and L2 respectively.

The ROC curves for a single experiment run can be seen in Figure 5. For ease of comparison, they are plotted on the same axes in Figure 6, and their relative AUC scores for this experiment run can be found in Table 4.

Figure 5: Seperate ROC

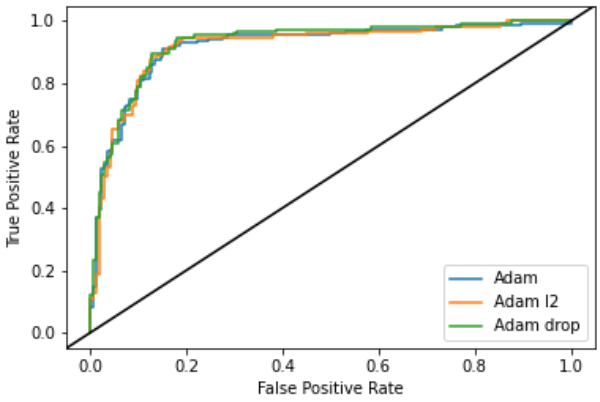

Figure 6: Combined ROC

Regularization	AUC
L2	0.919
Dropout	0.929
None	0.919

Table 4: Regularization AUC Scores

The higher the AUC the better, and in this case Dropout has the highest AUC, as seen in Table 4. This means when considering difference threshold probabilities, the dropout model will be the most accurate on average.

The confusion matrices for a single experiment run can be seen in Figure 7.

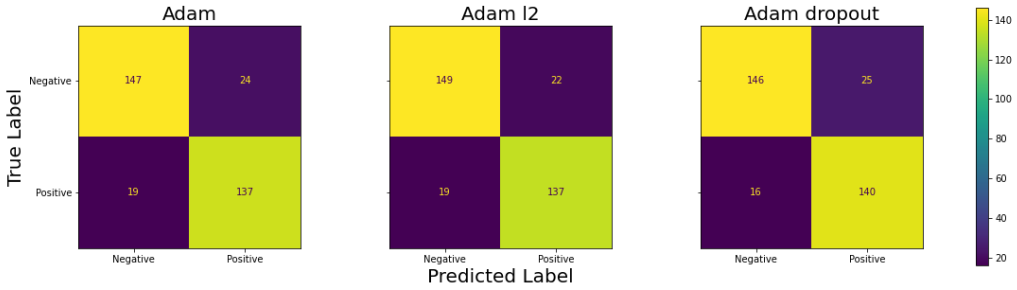

Figure 7: Confusion Matrices

From these matrices in Figure 7 we can determine the strengths and weaknesses of each model. While the accuracy between l2 and dropout are the same in this case, it is clear that they had different classifications. The l2 regularisation model was the most effective in classifying the negative class, while the dropout model was the most effective in classifying the positive class.

5 Limitations and Further Research

While the models used had good results, there is still improvements to be made. This experiment was limited by having only one layer in the neural networks, and sharing the model between different optimizers. These models and relevant hyperparameters were also chosen based off 10 trial runs. To improve this, an extensive grid search could be done over the hyperparameter space to determine the best neural network architecture and hyperparameters for each optimizer and regularization combination individually. This would better emphasise the potential of each optimizer and regularizer. More experiment runs could also lead to more accurate results.

Further research could be done by expanding the range of optimizers and regularisation techniques tested. Testing the impact of the regularisation techniques on each optimizers rather than just Adam would also likely have interesting results. The impact of different activation functions should also be the focus of a future investigation.

6 Conclusion

Clear relationships between the predictors and response were found through data exploration, which were able to be leveraged by neural networks to effectively predict the outcome of credit card applications. The test accuracies ranged from 85.5% (Adam no regularizaiton) to 86.5% (SGD no regularization). In respect to this dataset, the SGD optimizer was found to have the best generalization capability, however it is much slower than Adam when converging. The generalization capability of Adam was boosted through the use of L2 and Dropout regularization techniques, which lead to test accuracy improvement. However the best regularization technique with Adam, dropout

in this case, had a 86.3% average test accuracy, which was still less than the no regularization SGD's 86.5%.

7 References

(1) Dua, D. and Graff, C. (2019). *UCI Machine Learning Repository.* [http://archive.ics.uci.edu/ml]
(2) Kingma, D. P., Ba, J. (2014). *Adam: A Method for Stochastic Optimization.* [https://arxiv.org/abs/1412.6980]
(3) Ruder, S. (2016). *An overview of gradient descent optimization algorithms.* [https://arxiv.org/abs/1609.04747]
(4) Krogh, A. and Hertz, J. A. (1991). *A Simple Weight Decay Can Improve Generalization.*[https://dl.acm.org/doi/10.5555/2986916.2987033]
(5) Srivastava, N. Hinton, G. Krizhevsky, A. Sutskever, I. Salakhutdinov, R. (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting.* [https://jmlr.org/papers/v15/srivastava14a.html]