Predicting Secondary School Student Grades using Neural Networks

Allen Huang U6096857

Abstract.
A lot of research has been done on feed forward neural networks to solve problems. One of the main problems encountered when using neural networks are choosing suitable hyperparameters. Genetic Algorithms is one method to select good hyperparameters.

The dataset used is to predict grades of Portuguese secondary school students. It is suggested that neural networks are outperformed by tree-based methods for this dataset. In this paper, we show how neural networks can be used to achieve good results by first applying principal component analysis on the dataset. We show how pruning for distinctiveness and common effects can be used to improve the network by removing redundant neurons. We also explore the effects of using of genetic algorithms to select good hyperparameters for the neural network.

The goal of this research paper is to explore the effectiveness of neural networks and experiment with various techniques to improve the neural network.

Keywords: Neural Networks, Network Pruning, Genetic Algorithm

1 Introduction

In this paper, we use standard feed-forward neural networks. The neural network consists of multiple layers of processing units called neurons, each producing real-valued activations. Information only flows forward and there are no feedback connections where the outputs of the model are fed back into itself. Each neuron in one layer is fully connected to neurons in the next layer. The non-linear logistic sigmoid function is used as the non-linear activation function in the neural network and stochastic gradient descent is used to train the neural network.

2 Method

2.1 Data Pre-processing

Data pre-processing involves the filtering and preparation of data before it is used in the training phase. Good pre-processing often results in more effective and efficient training of the neural network on the data set.

The dataset contains both numerical and categorical variables. Since all input variables of the neural network need to be in a numeric form, categorical variables are converted to a numeric form. Nominal variables are categorical variables with no ordinal relationship. It is not possible to encode them into a single neuron by giving classes numeric labels as then it implies a natural ordered relationship between the classes. Hence, all nominal variables are encoded using the one-hot encoding. A nominal variable with N classes is represented by having N mutually exclusive binary attributes. The resulting dataset has 66 attributes.

It is suggested that neural networks perform badly on the original dataset because it contains too many irrelevant variables. Principal component analysis (PCA) is an algorithm that performs a dimensionality reduction on the dataset **(J.Shlens, 2005)** while preserving as much information as possible. This allows extraction of useful information from the dataset and can provide the neural network with more meaningful input attributes.

Standardization normalises the mean to 0 and the variance to 1. It is applied twice during pre-processing to each of the variables. It is applied once before PCA to make sure all variables have the same variance of 1. This makes sure all features have the same weighting and increases the effectiveness of PCA. Normalization is also applied before training of the neural network (right after applying PCA). By making the input variables have a similar scale, it is less likely for there to be steep slopes in the cost function. This generally results in a faster training time as it is less likely for stochastic gradient descent to get stuck.

2.2 Neural Network Structure

The universal approximation theorem for artificial neural networks states that a two-layered feed forward network with a “squashing” activation function can approximate continuous functions on compact subsets of R^n to any degree of non-zero error given sufficient hidden units.

However, the first hidden layer of a neural network is only able to model convex features. The second layer of a neural network can model both convex and concave features. This means that a three-layered neural network would generally be able to

model a continuous function using a lower total number of hidden units compared to a two-layered neural network. Hence a three-layered neural network structure is used over a two-layered neural network structure.

The input size of the neural network depends on the number of variables after applying PCA. The output of the neural network is a one in hot encoding of 5 mutually exclusive classes. Training of the neural network ends after 1000 epochs and the (S.Arlot, 2000) remaining hyperparameters for the three-layered neural network are chosen using a genetic algorithm.

The performance of the neural network is evaluated using 10 K-fold cross validation. K-fold cross validation is a model validation technique for assessing how well a model generalizes to a data set (S.Arlot, 2000). It averages prediction error over several runs to give a more accurate estimate of model prediction performance at the cost of being more computationally expensive.

2.3 Hyperparameters for Genetic Algorithm

The amount of variance to preserve while applying PCA is chosen as a parameter to be optimised. The parameter determines the trade-off between preserving more information of the original data and having more meaningful input variables for the neural network. This parameter influences the number of input neurons in the neural network.

One of the main problems for neural networks is choosing a suitable number of neurons. Having too few parameters may result in a network having a large bias and being unable to capture the complexity of the data. Having too many parameters may result in a network having a large variance and overfit on the data. We use a genetic algorithm to select the number of hidden units in the first and second hidden layer.

Learning rate is also another important hyperparameter that can be chosen by the genetic algorithm. If the learning rate is too high, the network may have trouble converging. If the learning rate is too low, the network may converge too slowly.

2.3.1 Encoding of Hyperparameters

The variance to preserve when applying PCA is encoded using 5 bits. The 5 bits are a binary representation of a number between 0-31. The variance preserved is taken to be (100 – number) %. Variance preserved is a value between 69% and 100%.

The number of hidden units in the one hidden layer is encoded using 7 bits. The 7 bits are a binary representation of a number between 0-127. The number of neurons in that hidden layer is taken to be (1 + number). The number of hidden units in that hidden layer is a value between 1 and 128.

The learning rate is encoded using 10 bits. For $k \in 1..10$, the k^{th} bit represents $(0.5)^k$. The learning rate is taken to be the sum of these numbers. The learning rate is a value approximately between 0 and 1 with an error of 0.5^{10} (~0.001).

The total length of the DNA is 29 bits.

2.3.2 Hyperparameters that are not chosen

Epoch is a hyperparameter that is fixed at 1000 and is not chosen for the genetic algorithm. This is because learning rate is already chosen and generally, it is possible to create an equal or better performing neural network by lowering the learning rate and increasing the epoch by a sufficient amount. Including both learning rate and epoch would greatly increase time taken for the genetic algorithm to converge while having solutions of similar quality.

The number of layers of the neural network is also a hyperparameter that is not chosen. This is mainly because it is harder to encode this information. One method is having 8 bits in the DNA for each possible layer. The first bit would represent whether to include the layer and the remaining 7 bits would determine the size of that hidden layer. However, this drastically increases the length of the DNA and the time of the genetic algorithm to converge. Another method would be encoding the number of layers into a few bits and making all layers have the same of hidden units. The downside to this is that the genetic algorithm only explores network structures where the network layers have the same number of hidden units.

2.4 Genetic Algorithm

The population size is initialized by generating 30 random DNA sequences. They are randomly generated because this introduces diversity in the population. The DNA are spread out in the search space and are more likely to find better solutions.

The fitness function for a DNA is chosen to be the reciprocal of the cross-validated loss of a network with hyperparameters extracted from that DNA. This is chosen over validation accuracy because accuracy is not as representative of how well the parameters of a neural network are fit to the data. For example, epochs with higher validation losses may have an equal or higher accuracy than the accuracy at the epoch with the minimum validation loss. The reciprocal is taken as then maximising the fitness function is equivalent to minimizing the loss function.

During each successive generation, half of the best individuals from the previous generation are kept and the other half are replaced by newly formed offspring. This is done to help with the convergence of the genetic algorithm. The genetic algorithm terminates at 30 generations.

Offspring are formed using uniform crossover from two parents. The two parents are selected using proportionate selection based on their fitness functions and the offspring are more likely to inherit from DNA with higher fitness. This allows successive generations to improve in fitness performance over time. The offspring are then mutated with a small chance to introduce help increase diversity into the population.

2.5 Network Pruning

Network pruning is a method of regularization that aims to remove irrelevant hidden units. This reduces the number of parameters in the network which improves efficiency and makes the network less likely to overfit. Pruning for distinctiveness and pruning for constant effects are used (D.Harris, 1991).

Hidden units are checked for distinctiveness. If a hidden unit is not distinct, it means that it can be removed because another hidden unit is doing a similar or complementary role. When using the sigmoid activation function, this is done by calculating the angle between outputs of hidden units when they are normalized to 0.5, 0.5. If the angle between two hidden units are less than 15 degrees, this means they have very similar outputs. One of these two hidden units may be pruned and have its weights combined with the other hidden unit. If the angle between two hidden units is larger than 165 degrees, this means they have almost complementary effects and both may be removed without much adjustment of weights.

A hidden unit or groups of hidden units are also pruned if they produce a constant effect. This is because these can be pruned and have their outputs being added to the bias with little change to the functionality of the neural network. A constant effect can be tested by checking the variance of outputs of the hidden neurons.

3 Results and Discussion

3.1 Results for neural network with manually picked hyperparameters

The hyperparameters chosen for the neural network are shown in Table 1 and the cross validated loss and accuracy of the neural network are shown in Fig. 1.

Variance Preserved	95%
#Neurons in Hidden Layer 1	50
#Neurons in Hidden Layer 2	40
Learning Rate	0.1

Table 1. Hyperparamters of Neural Network

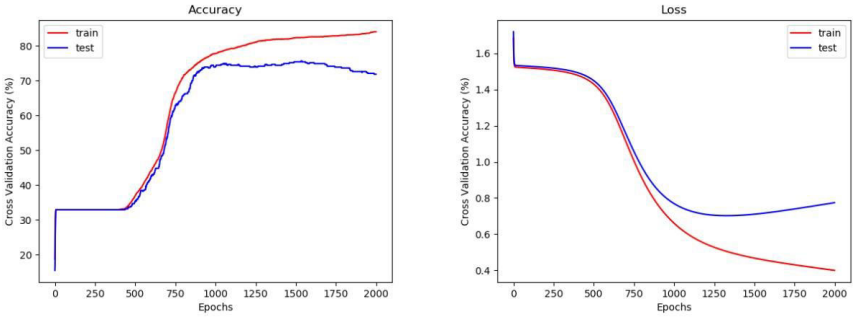

Fig. 1. Cross Validated Accuracy and Loss for neural network

The best cross validated loss is at epoch 1328, with a cross validated accuracy of 74.948%. The paper achieves a 76.1% accuracy using decision trees and a 65.1% accuracy using neural networks. The result we obtained is almost comparable to the decision trees approach in the paper. This is most likely due to PCA giving the neural network more meaningful inputs. However, it is clearly visible that the current network structure is prone to overfitting. This can be seen from the validation loss increases while the training loss decreases.

3.2 Effects of Pruning

The resulting hyperparameters after applying pruning Table 2 and the cross validated loss and accuracy of the pruned neural network are shown in Fig. 3.

Variance Preserved	95%
#Neurons in Hidden Layer 1	44.6
#Neurons in Hidden Layer 2	15.1
Learning Rate	0.1

Table 2. Average Hyperparameters of Pruned Neural Network

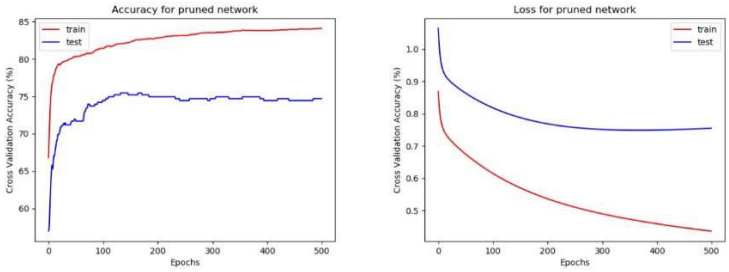

Fig. 2. Cross Validated Accuracy and Loss for neural network after pruning

On average, the pruning algorithm prunes around 5.4 neurons from the first hidden layer and 24.9 neurons from the second hidden layer. Since most of the weights are already trained in the network and the pruning algorithm maintains most of the original functionality of the network, it takes a much shorter time (<100 epochs) to reach an accuracy of over 70% when comapred to the initial training of the unpruned network with randomized weights. The cross validated accuracy at the best cross validated loss is equal to 74.342%.

The pruned network is able to achieve an accuracy similar to that of the unpruned network. The pruned network has less parameters so it is more computationally efficient and also seems to be less prone to overfitting compared to the unpruned network as there is a significantly less increase in validation loss when training loss decreases. Overall the pruned network seems to be an improvement over the unpruned network.

3.3 Results of Genetic Algorithm

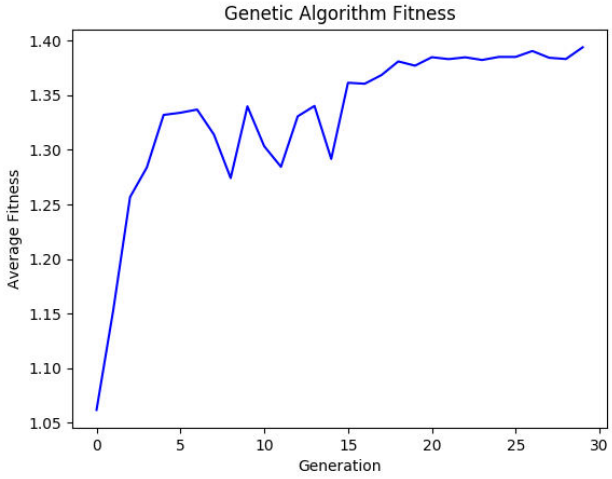

Fig. 3. Average fitness of genetic algorithm

The average fitness each generation as the genetic algorithm is running is shown in Fig. 3. The fitness is shown to gradually increase over time, this shows that the genetic algorithm is improving the population over time. The genetic algorithm converges right under 1.40 which is around a 0.71 validation loss for the neural networks.

Variance Preserved	72%
Neurons in Hidden Layer 1	15
Neurons in Hidden Layer 2	9

Learning Rate	0.27246

Table 3. Hyperparameters for Neural Network of best individual

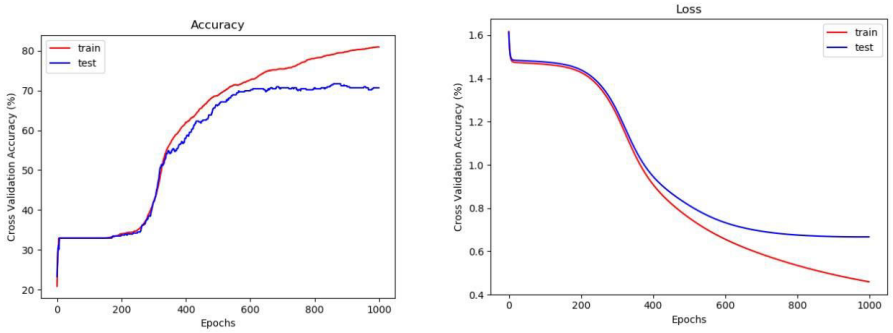

Fig. 4. Genetic Algorithm Cross Validated Accuracy and Loss for best individual

The best neural network obtained from the genetic algorithm after 30 generations achieves an accuracy of 70.41% after 1000 epochs. This appears to be worse than choosing a large number of hidden neurons and pruning the network. Pruning had no effect on this small model found by the genetic algorithm.

The genetic algorithm may have found simple solutions because of the shape of the loss curve. More complex neural networks are more likely to overfit and there is a smaller range of learning rate values where it performs well. Meanwhile, simpler networks have a larger range of learning rate values where it performs since it does not overfit.

One possible reason for the worse performance may have been the time constraint as using cross validated loss in the fitness function is time consuming. Alternatively, a potential change could have been evaluating a single nerual network for some dataset instead of evaluating 10 neural networks for 10-K-fold cross validation. Evaluating a single neural network is much faster and may allow much more generations to be run in the same time however it also provides less information than k-fold cross validaiton and may have trouble converging. Also the results may suggest that more diversity needs to be introduced, especially for learning rate, to further explore the solution space since varying network sizes require vastly different learning rates to perform well.

4 Conclusion and Future Work

We have achieved results on par with tree based methods in the paper using neural networks. This is done by using PCA to remove redundant features. Cross-Validation is used to evaluate the performance of neural networks. We have also shown how

pruning can remove redundant neurons while maintaining a similar performance. We have also explored using genetic algorithms to select good hyperparameters for the neural network.

Potential future works may include experimenting with different hyperparameters for the genetic algorithm. One potential change could include increasing the mutation rate for the bits encoding the learning rate. Other methods such as migration could also be experimented with to increase diversity in the population.

5 References

A.Silva, P. a. (2008). *Using Data Mining to Predict Secondary School Student Performance.* Portugal: In A. Brito and J. Teixeira Eds., Proceedings of 5th FUture BUsiness TEChnologyConference.

D.Harris, T. a. (1991). *Network Reduction Techniques.* San Diego: Proceedings InternationalConference on Neural Networks Methodologies and Applications, AMSE.

J.Shlens. (2005). *A Tutorial on Principal Component Analysis.* San Diego.

S.Arlot, A. (2000). *A survey of cross-validation procedures.* Paris: Statistics Surveys Vol. 4 (2010) 40-79.