Grade Prediction
Using Different Methods

Srikanth Polisetty

Research School of Computer Science
Australian National University
Canberra, Australia

u7142680@anu.edu.au

Abstract: In this paper accuracy and loss of different models were compared for grade prediction based on given dataset. Feedforward Neural Network (FFNN), CasPer NN and Feedforward Neural Network with/without Greedy Layer-wise Pretraining (GLP) were used to do the comparison. FFNN is a plan vanilla neural network model with predefined intermediate layers of neurons. CasPer uses Cascade architecture and a variant of RPROP (a backpropagation algorithm) to train the network. FFNN with GLP attacks the problem of training the deep networks in layers. It keeps the weights of the existing hidden layers constant, while adding new hidden layers. FFNN GLP model works on the assumption that it is easy to train a shallow network rather than a deep network. Since the chosen dataset size is small, the FFNN with fixed layers had given best test accuracy.

Keywords: CasCor Neural Network, CasPer Neural Network, Grade prediction, Feedforward model, Greedy Layer-wise Pretraining model

1 Introduction

In this paper, grade prediction based on given set of input features using different models like
- Feedforward Neural Network without adding layers
- CasPer Neural Network and
- Feedforward NN with Greedy Layer-wise Pretraining
- Feedforward NN without Greedy Layer-wise Pretraining

is discussed. Comparison is done for the Loss and Accuracy of results to complete the prediction.

Prediction is a classical machine learning problem which is being used for many purposes. In the days of uncertainty due to pandemic, in some countries the midterm results and assignment results are being used to compute the final exam marks. Also, the prediction can be used to verify the relation between various midterm exams/assignments and the final exam results. This relation helps the faculty to assess the possibility of integrity compromise.

The dataset was chosen from student's marks from University of New South Wales [1]. After doing the necessary preprocessing, this dataset is used for training the above-mentioned models.

Applied the Feedforward model with ADAM optimizer on the preprocessed dataset and calculated the prediction accuracy. In the same way using the CasPer Neural Network and Feedforward NN with and without Greedy Layer-wise Pretraining, the prediction accuracy and the loss were calculated.

2 Methodology

2.1 Data Preparation

1. Header change

Given data set [1] contains column names in 2 different rows also contains column header separator lines and irrelevant data like date, time, page no etc. This data is cleaned up manually and column headers are added to represent the data properly.

2. Data preprocessing

In the chosen dataset, there are many values not filled in (NULL) for some features.

For each student, if we can calculate the percentile position in the class, it can be used to approximately determine the missing data points. For this purpose, for a given student for each of the features where data is present, the percentile value is calculated. Average of all percentile values gives the percentile position of that student in the class. This value is used to estimate the missing data for that student (if any). Note: The rows in which all the entries are NULL are dropped as no data of that student is available to infer other values.

In the original dataset, there are 153 rows and 16 features in the dataset. After dropping the rows where all the entries are null, the dataset has 146 rows and 12 features.

The columns which have more than 10 categories are dropped as the dataset has only 153 rows and the average number of entries corresponding to each class will not be more than 15.3, which is extremely small.

To eliminate the chance of handling different ranges of data, the data in numerical columns is normalized by using the **maximum possible score** of that column.

The categorical columns left are “ES” and “S”. These columns are encoded using the Leave One Out Target Encoding (LOOTE). LOOTE is like target encoding, but unlike target encoding when calculating the encoded value of an entry, that entry’s value is not used. LOOTE is useful especially when dealing with smaller datasets. The formula of LOOTE is as follows:

$Category_{count}$= Number of occurrences of a specific categorical value

$$Category_{mean} = \frac{\text{Sum of target values for a specific categorical value}}{Category_{count}}$$

$$mean = \frac{\text{sum of all target values}}{\text{Count of all target values}}$$

m is a hyper value whose value is initialized to 100

$$Cell\_value = \frac{(Category_{count} * Category_{mean}) + (m * mean)}{Category_{count} + m - 1}$$

To eliminate the chance of handling different ranges of data, the encoded data is normalized by using the **maximum value** of that column.

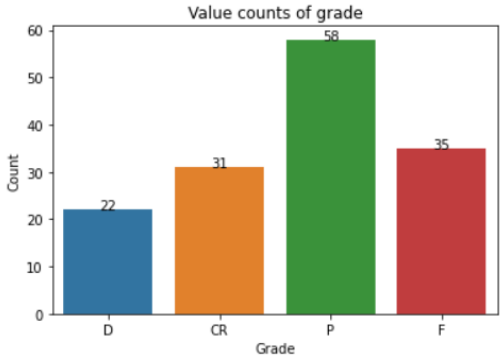

Figure 1. Value counts of grades

3. Outcome column addition

Below criteria is used for assigning the grades corresponding to total marks obtained:

- 75-100%: Distinction (D): Value 1
- 65-74%: Credit (CR): Value 2
- 50-64%: Passed (PS): Value 3
- <50%: Fail (FL): Value 4

A new column is added to the data set to contain these target values.

2.2 Feedforward Neural Network model

Feedforward Neural Network model [4] is trained with 64% of dataset, validated with 16% of dataset and tested with 20% of dataset. Since input layer has 12 features, 12 neurons and 24 neurons were configured in two hidden layers. There are 4 output neurons corresponding to the grades defined above. ReLU activation function is used in the hidden layers and softmax is used for the output layer.

Adam was used as optimizer with learning rate of 0.001 and crossentropy was used as loss function. The neural network was trained with batch size being 16 and epochs 500. Algorithm is taken from [7].
From the results, it was observed that,

- Validation Loss = 0.81 and validation accuracy = 54.17%
- Test Loss = 0.5534 and **Test Accuracy** = **90.0%**

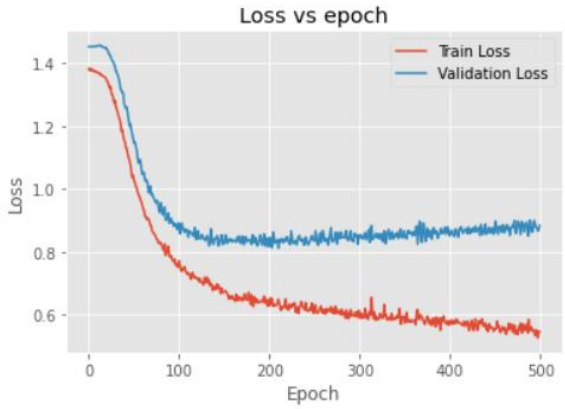

Figure 2. Train Loss and Validation Loss Vs Epochs

2.3 CasPer Neural Network

Cascade-correlation (CasCor) is an architecture and generative, feed-forward, supervised learning algorithm for artificial neural networks. CasCor begins with a minimal network, then automatically trains and adds new hidden units one by one creating a multi-layer structure. In this algorithm, hidden units are added only one at a time and do not change after they have been added. For each new hidden unit, the algorithm tries to maximize the magnitude of the correlation between the new unit's output and the residual error signal of the network. However, the disadvantages of CasCor are, due to smaller size of network early neurons are poor feature detectors and this algorithm leads to too large network in later phases.

CasPer uses Progressive RPROP, to train the network. CasPer does not use a correlation measure or weight freezing but uses RPROP to train the whole network. CasPer results in smaller networks than CasCor and provides better generalization than CasCor. CasPer algorithm maintains 3 regions in the network

- L1: Weights connecting to new neuron
- L2: Weights connecting from new neuron to output neurons
- L3: all remaining weights

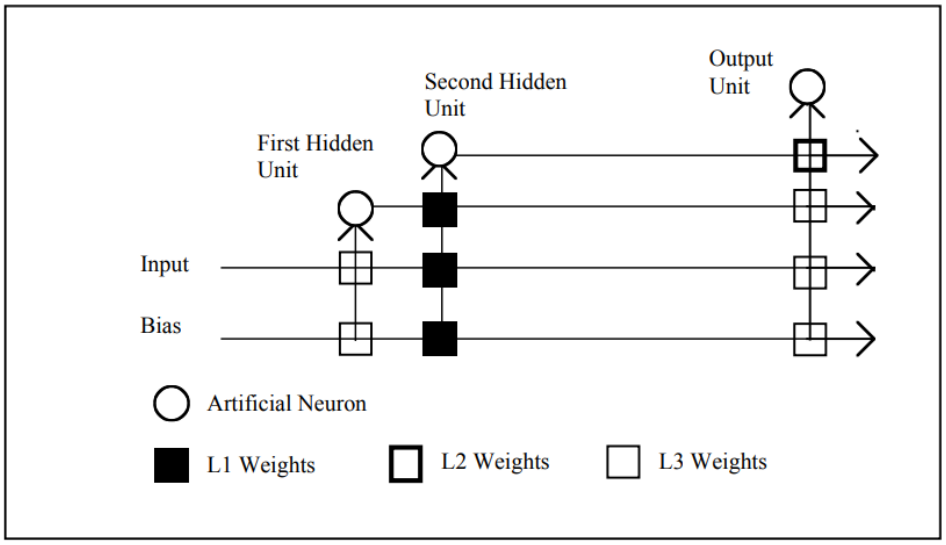

Figure 3. CasPer Algorithm

Unlike CasCor algorithm, CasPer algorithm allows old weights to be changed. CasPer uses weight decay to improve generalization. Weight decay uses Simulated Annealing

$$\delta E/\delta wij=\delta E/\delta wij-D*sign(wij)*wij^2*2^{-T*Hepoch}$$

Where, Hepoch = epochs since addition of last hidden neuron
D = decay parameter

The basic idea of RPROP is that if the error gradient for a given weight had the same sign in two consecutive epochs, we increase its step size, because the weight's optimal value may be far away. If, on the other hand, the sign switched, we decrease the step size.

CasPer algorithm is taken from [8].

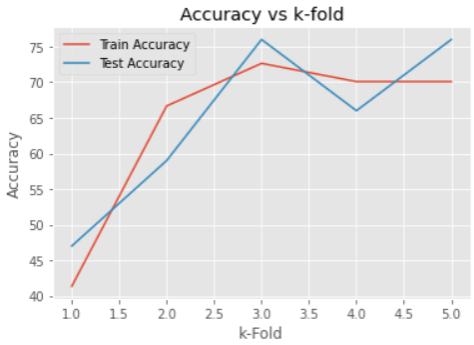

Figure 4. CasPer model's k-Fold Vs Accuracy

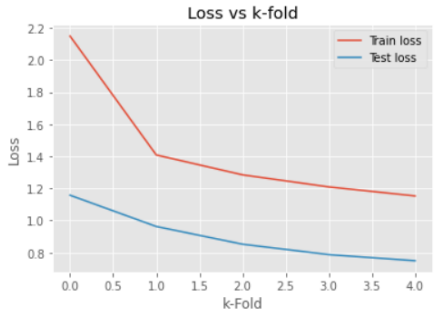

Figure 5. CasPer model's k-Fold Vs Loss

Results after k-Fold cross validation, where k = 5 folds
Average Test Loss = 0.90 and **Test Accuracy = 64.80%**

2.4 Feedforward Neural Network with / without Greedy Layer-wise Pretraining

Below given charts depict the validation loss for given number of layers added. As we can observe, the validation loss is increasing while the layers are being added. Test loss and Test accuracy are calculated with the optimal number of layers found.

1. FFNN with GLP: Test Loss = 3.0755, **Test Accuracy = 26.67%**

```
Validation loss and accuracy when 1 layers are added are 0.8920 and 54.17 respectively
Model saved with 1 layers added
Validation loss and accuracy when 2 layers are added are 0.9766 and 58.33 respectively
Validation loss and accuracy when 3 layers are added are 1.2778 and 50.00 respectively
Validation loss and accuracy when 4 layers are added are 1.4193 and 25.00 respectively
Validation loss and accuracy when 5 layers are added are 1.3150 and 37.50 respectively
Validation loss and accuracy when 6 layers are added are 1.4386 and 12.50 respectively
Validation loss and accuracy when 7 layers are added are 1.3989 and 25.00 respectively
Validation loss and accuracy when 8 layers are added are 1.4431 and 4.17 respectively
Validation loss and accuracy when 9 layers are added are 1.4444 and 4.17 respectively
Validation loss and accuracy when 10 layers are added are 1.4448 and 4.17 respectively
```

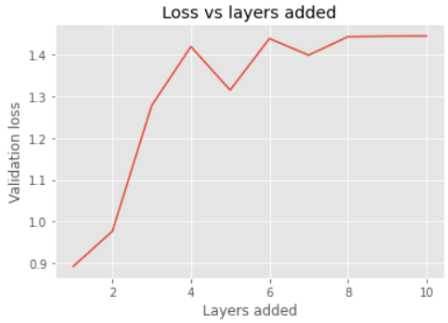

Figure 6. FFNN GLP Validation Loss Vs Layers added

2. FFNN without GLP: Test Loss = 420.6770, **Test Accuracy = 30.0%**

```
Validation loss and accuracy when 1 layers are added are 0.7611 and 66.67 respectively
Model saved with 1 layers added
Validation loss and accuracy when 2 layers are added are 0.7383 and 66.67 respectively
Model saved with 2 layers added
Validation loss and accuracy when 3 layers are added are 0.7795 and 58.33 respectively
Validation loss and accuracy when 4 layers are added are 0.8104 and 62.50 respectively
Validation loss and accuracy when 5 layers are added are 0.9034 and 50.00 respectively
Validation loss and accuracy when 6 layers are added are 1.0114 and 41.67 respectively
Validation loss and accuracy when 7 layers are added are 0.8751 and 62.50 respectively
Validation loss and accuracy when 8 layers are added are 1.0403 and 54.17 respectively
Validation loss and accuracy when 9 layers are added are 1.0461 and 50.00 respectively
Validation loss and accuracy when 10 layers are added are 1.1881 and 37.50 respectively
```

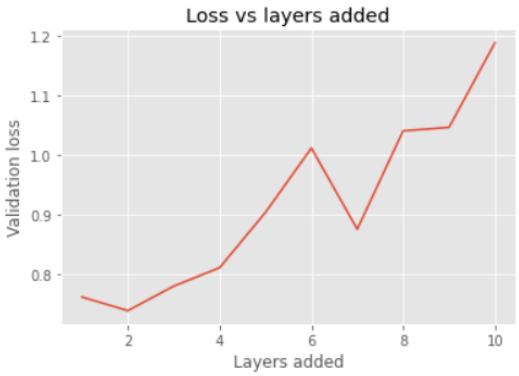

Figure 7. FFNN without GLP Validation Loss Vs Layers added

2.5 Evaluation methods

Results obtained using the Feedforward neural network are compared with that of the Feedforward NN with GLP, Feedforward NN with additional layers (without GLP), and Casper neural network.

Model	Test Accuracy
Feedforward NN with fixed layers	**90.00%**
CasPer NN	64.80%
Feedforward NN with GLP	26.67%
Feedforward NN without GLP	30.00%

Table 1. Model accuracy comparison

3 Results and discussion

3.1 Comparison of various models

The models (except CasPer) were trained on 64% of the dataset and remaining data was used for validation and testing purpose. For comparison purpose same test data was used in all the models. Since dataset size is small, it was observed that the accuracy of the Feedforward NN model is higher than that of other models.

CasPer NN gives the flexibility of adding intermediate hidden layers automatically depending on the need. However, in case of Feedforward NN we need to define the number of hidden layers at the time of defining the model.

4 Conclusion and Extensions

Using deep learning with small datasets usually results in overfitting. Same is proven with the experimental results. To deal with overfitting, we can incorporate the below two strategies into our Feedforward NN model: L2 Regularization and Dropout. When we use these strategies, validation loss will closely match with training loss.

There are two extensions proposed in [3] for CasPer NN. The first is to employ the SARPROP algorithm to train the newly inserted hidden neuron. SARPROP is based on the RPROP algorithm and uses Simulated Annealing to enhance the convergence properties of RPROP. SARPROP has been shown to be successful in escaping local minima, a property which will enable a better search of the error surface by the new hidden neuron. The second extension involves CasPer training a pool of hidden neurons, as is done in CasCor. Each hidden neuron in the pool is trained using SARPROP, as in the first extension.

References

1. Choi, ECY and Gedeon, TD "Comparison of Extracted Rules from Multiple Networks," invited paper, Proceedings IEEE International Conference on Neural Networks, vol. 4, pp. 1812-1815, Perth, 1995.
2. https://wattlecourses.anu.edu.au/pluginfile.php/2558398/mod_resource/content/6/NN9_cascor%2BCasPer.pdf
3. N. K. Treadgold and T. D. Gedeon, "Extending and benchmarking the CasPer algorithm. Advanced Topics in Artificial Intelligence", pp. 398–406. https://doi.org/10.1007/3-540-63797-4_93
4. https://www.freecodecamp.org/news/how-to-build-your-first-neural-network-to-predict-house-prices-with-keras-f8db83049159/
5. https://medium.com/@andreluiz_4916/pytorch-neural-networks-to-predict-matches-results-in-soccer-championships-part-ii-3d02b2ddd538
6. https://machinelearningmastery.com/pytorch-tutorial-develop-deep-learning-models/
7. https://towardsdatascience.com/pytorch-tabular-regression-428e9c9ac93
8. https://github.com/chehao2628/ImprovedCasper/blob/master/CasperNetwork.py
9. N. K. Treadgold and T. D. Gedeon, "A cascade network algorithm employing Progressive RPROP," Biological and Artificial Computation: From Neuroscience to Technology, pp. 733–742, 1997, https://doi.org/10.1007/bfb0032532
10. dmlicht, "How to block calls to print?," Stack Overflow, Dec. 05, 2011. https://stackoverflow.com/questions/8391411/how-to-block-calls-to-print#:~:text=use%20with
11. chehao2628, "chehao2628/ImprovedCasper," *GitHub*, 2021. https://github.com/chehao2628/ImprovedCasper/blob/master/CasperNetwork.py
12. https://www.facebook.com/MachineLearningMastery, "How to Use Greedy Layer-Wise Pretraining in Deep Learning Neural Networks," *Machine Learning Mastery*, Jan. 31, 2019. https://machinelearningmastery.com/greedy-layer-wise-pretraining-tutorial/