Predict Final Total Mark of Students with ANN, RNN and Bi-LSTM

Zheming Han

Research School of Computer Science, Australian National University
E-mail: u6865707@anu.edu.au

Abstract. This research is about predicting the final total scores of students by a variety of assignments and mid-term exam scores. The paper mainly involves data processing of the students score data set, constructing three different neural network models for four classifications, which are Artificial Neural Network (ANN), Recurrent neural network (RNN) and Bidirectional Long Short-Term Memory (Bi-LSTM) and implementing (Principal Component Analysis) PCA, dynamic learning rate, pruning neurons on my ANN model. Finally, the prediction accuracy of my three models are more than 72%. The Bi-LSTM model has the best prediction accuracy, which is larger than 76%.

Keywords: Mark Prediction, Neural network, PCA, Learning rate decay, Prune neurons, RNN, LSTM, Bi-LSTM

1 Introduction

The primary purpose of the paper is to predict the final total scores by constructing a neural network model. The data set used for training and testing the model contains various assignments scores, mid-term exam scores, and final total scores of 153 students. Various assignments, labs and mid-term exam results only account for 40% of the final total mark. The final exam score of 60% of the total score is missing. Students' final total mark may fall into four classes, which are Distinction, Credit, Pass and Fail. The task is to use the grades in the dataset to predict the classification to which the student 's final grade belongs. Based on the research of Dr Choi, E. C. Y and Dr Gedeon, T. D, I build three different neural networks model, and the aim is to the higher prediction accuracy [1].

The reason I chose this dataset is that I am very interested in the classification problem. The size of the dataset is relatively small. In order to obtain stable and great prediction results, a strong ability to adjust parameters is required, which is also one of the capabilities I want to improve. Therefore, for the above reasons, I chose this data set.

I use 3 different neural network models to predict final grades. They are Artificial Neural Network (ANN), Recurrent neural network (RNN) and Bidirectional Long Short-Term Memory (Bi-LSTM).

My ANN model consists of three layers of neurons, which are two hidden layers with relu activation function and one output layer with a softmax activation function. In my ANN model, I use PCA to process the dataset to increase the accuracy and reduce overfitting [2] and use the ReduceLROnPlateau function to speed up model converging [3]. According to Network Reduction Techniques [4], I also implement the pruning neurons in my ANN model to reduce the size of the neural network and improve efficiency.

My RNN model is a kind of feedforward neural network, which usually process sequential input [5]. I entered all the scores of a student as a sequence into the RNN, and I got the prediction results. The RNN model contains one RNN layer and a full connected linear layer, and softmax activation function is used to classify the output of RNN.

My Bi-LSTM model is a kind of more complex RNN model. The input of Bi-LSTM model is the same as RNN model. And my Bi-LSTM model contains three Bi-LSTM layers and one full connected linear layer. I also use the softmax function to classify the output of my model.

My three models all have a high prediction accuracy. The test accuracy of Bi-LSTM model is the highest.

2 Method

To predict the classification of the final total score through various assignments, labs, and mid-term exam results, I processed the data and designed and implement three different neural network models to predict, which are ANN, RNN, Bi-LSTM.

2.1 Process data

In the original dataset, there are some data is '.'. I think '.' is a record that corresponds to the lack of exams or no assignment submission of a student, so I set this '.' to 0 to complete the entire data set.

Nevertheless, there is some samples all by '.' composition. As shown in Fig1, these samples are meaningless, so I removed these samples from the data set. I remove eight samples from my dataset, and there are 145 samples left in the data.

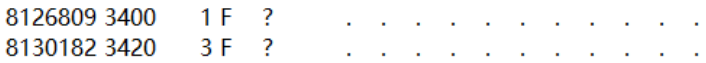

Fig. 1. All empty data

However, the accuracy of the results predicted by my model is very unstable. After investigating, I found that the number of samples in each grade class is quite different. As shown in Table 1, the number of Distinction data is only 22. When I randomly split the data set into a training set and a test set (at a rate of 80%), it is likely that the training set cannot allocate enough Distinction data, so that the Distinction data cannot be fully trained, resulting in low test accuracy.

Table 1. the Number of samples per grade classification

Grade Classification	Distinction	Credict	Pass	Fail
Number	22	31	34	58

In order to solve this problem, I randomly divided the data of each label into the training set and the test set (at a rate of 80%), and then combined the training set and the test set of each label into the final training and test set. In this way, the ratio of each label in the test and training set is fixed, which can ensure that the data of each label is more fully trained. The prediction results of my model are more stable than before.

2.2 Artificial Neural Network (ANN)

ANN is the standard feedforward natural network. In order to predict the classification of students' final grades, I designed my ANN model that can be used for four classifications.

2.2.1 Model Design

The model contains two hidden layers, and the number of neurons is 46 and 50, which is the best result got by cross-validation. The activation function of the two-layer neuron is the relu function. Relu activation function can make the network train faster, increase the nonlinearity of the network, and reduce overfitting. The output layer is composed of 4 neurons, and the activation function is the softmax function, which is very suitable for multi-class classification. The loss function is the Cross-Entropy function.

I use Adam optimizer to optimize my function. Adam is a simple and efficient optimization function and includes a variety of optimization methods, such as momentum and learning rate decay [6]. Adam also contains many adjustable parameters. For instance, 'lr' adjusts the learning rate of the model, and 'weight_decay' adjusts the regularization of the model to reduce the overfitting of the model. I use cross-validation to adjust these parameters and obtain optimal values. These optimal values of parameters are shown under Table 2.

Table 2. The optimal parameters of my ANN model

Names of Parameters	Learning Rate	Weight decay	Epoch Number
Number	0.0088	0.478	500

What's more, I used some techniques to increase test accuracy of my ANN model.

2.2.2 Principal Component Analysis (PCA)

According to a paper of Dr Christian, PCA, a well-known method of compressing data dimensions, can obtain the basic information of the input data, improve the generalization of the model, and reduce the degree of overfitting of the model [2]. I use PCA to process my dataset.

I calculate the percentage of explained variance of each feature by PCA, which could be considered as the importance of each feature. According to Table 2, the first six features accounted for more than 99% of the explained variance. Therefore, I used PCA to compress the dimension of input data to 6.

Table 3. Percentage of explained variance of each feature by PCA

Feature Index	1	2	3	4	5
Percentage	0.57298	0.18692	0.09569	0.07510	0.05563
Feature Index	6	7	8	9	10
Percentage	0.00470	0.00355	0.00243	0.00168	0.00128

When I use PCA in my model, the test accuracy increases. I take 20 different training and test sets to test the prediction accuracy of my model with or without PCA. I average the 20 results to generate the average training and testing accuracy of each epoch and plot the accuracy image. As shown in Figure 2, when my model uses PCA to process the data set, the test accuracy rate is increased from about 62% to about 70%. Moreover, the gap between the accuracy of the test set and the training set reduces, which means that the overfitting of the model reduces. Therefore, PCA can help my model to increase prediction accuracy and reduce overfitting.

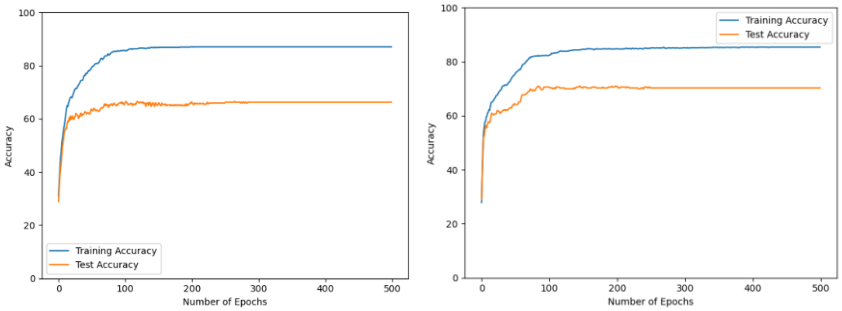

Fig. 2. The figure of the training and test accuracy of my model without PCA (left) and with PCA (right)

We generally need to normalize the data before performing PCA. However, After the experiment, the prediction accuracy of the model reduced a lot after normalizing and PCA, so I did not normalize the dataset and only used PCA in my model.

2.2.3 ReduceLROnPlateau

When testing the model, the accuracy of training and testing is unstable sometimes, which may be caused by the excessive learning rate. In order to stabilize the training and test accuracy, my model used the 'ReduceLROnPlateau' function. The 'ReduceLROnPlateau' function will reduce the learning rate when the model is not improving [3]. ReduceLROnPlateau' help my model converge faster.

2.2.4 Prune Neuros

Based on the research of Dr Gedeon, T. D [4], I attempt to implement the algorithm to prune similar or opposite neurons in my model. In a layer of the neural network, each neuron has an output, which is a vector to represent this neuron. Then calculate the angle between these vectors, and determine whether two neurons are similar or opposite by the angle.

In order to make the model get a better result, after experiment, I set two angles, which are 1 degree and 179 degree. If the angle between two neurons is less than 1 degree, the two are considered to be similar, and the model removes one of the neurons. If the angle between two neurons is larger than 179 degree, the neurons have the opposite effects, and the model removes both neurons. When a neuron is removed because it is similar to other neurons, the model will assign the weight of the neuron to other neurons similar to it. After pruning, the efficiency of the neural network is also improved.

After pruning neurons, my model usually removed 30 to 40 neurons at a time. The test accuracy of my model will reduce by about 10%. However, after continuing to train the model with fewer neurons, the accuracy will rise or even exceed the accuracy of the original model.

2.3 Recurrent Neural Network (RNN)

RNN is a kind of feedforward neural network. RNN usually process sequential data [6]. In a sequence, RNN will train each element in a sequence one by one. Each element will be inputted to RNN with the statement of the last elements. The statement in RNN could be weight and bias. Each input will get output from RNN. If the model takes the last output as the output of the entire model, the RNN model is called many-to-one RNN model. The many-to-one RNN model is shown in Figure 3

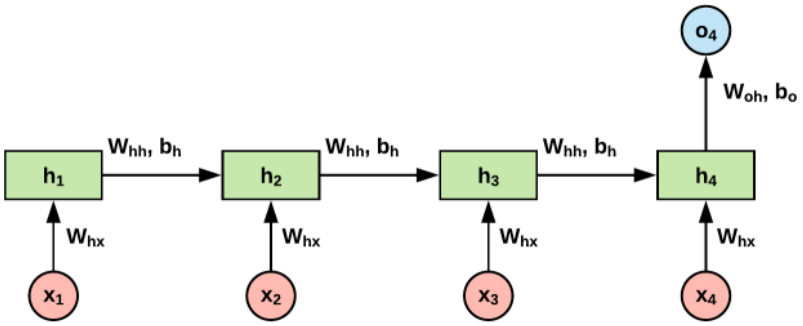

Fig. 3. The structure of a many-to-one RNN Model [7]

My RNN model is a many-to-one RNN model, which contains an RNN layer with 15 hidden neurons. After the RNN layer is a fully connected linear layer containing 4 neurons, which is the output layer. I regard all scores of a student as a sequence, so the dataset has 145 sequences. Input the matrix combined by 145 sequences into my RNN model. Since a student has 10 different scores, and a sequence has 10 elements, then my RNN model will produce 10 outputs. I only select the last output and use the softmax activation function to classify the output. The loss function is a cross-entropy function, and the optimizer is Adam. My RNN model also uses ReduceLROnPlateau to speed up the converge. The optimal parameters are obtained by cross-validation, which is shown in Table 4.

Table 4. The optimal parameters of my RNN model

Names of Parameters	Learning Rate	Weight decay	Epoch Number
Number	0.0088	0.478	500

2.4 Bi-Directional Long Short-Term Memory (Bi-LSTM)

Bi-LSTM is a more complicated RNN model. The Bi-LSTM is related to BI-RNN and LSTM.

BI-RNN is just putting two independent RNNs together [8]. As shown in Figure 4, The two different RNN input the data from two different directions. And the output is combined by the two bi-direction RNN layers. The BI-RNN model in Figure 4 is a many-to-many model. But I only use the last output as the output of my model, which is a many-to-one model.

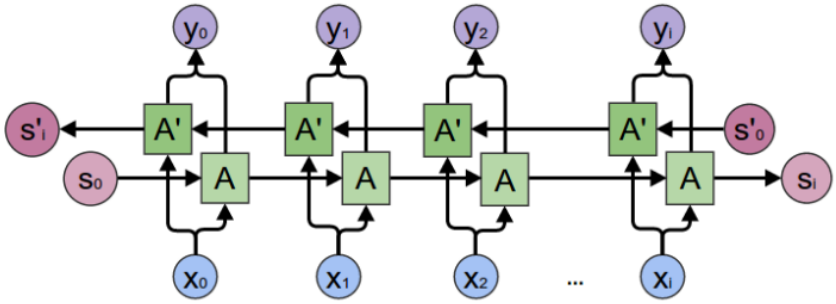

Fig. 4. The structure of a BI-RNN Model [9]

LSTM is a particular RNN model. LSTM could be used to deal with long time sequence and vanishing gradient problem [8]. LSTM layer contains memory cells and corresponding gate units, which help LSTM save the most important memory [10].

Bi-LSTM is also a kind of BI-RNN model, which RNN layer is changed to the LSTM layer.

My Bi-LSTM model contains 3 Bi-LSTM layers. Each Bi-LSTM layer contains 16 hidden neurons. And the final Bi-LSTM model is connected with a fully connected linear layer, which is the output layer with 4 neurons. I only select the last output and use the softmax function to classify it. The loss function is also a cross-entropy function, and the optimizer is also Adam. My Bi-LSTM model also uses ReduceLROnPlateau to speed up the converge. I use cross-validation the get the optimal parameter of my Bi-LSTM model, which is shown in Table 5.

Table 5. The optimal parameters of my Bi-LSTM model

Names of Parameters	Learning Rate	Weight decay	Epoch Number
Number	0.0076	0.0077	200

3 Results and Discussion

I designed and implemented 3 different neural network models and implemented different techniques in my ANN model to increase the accuracy of prediction. Because the test accuracy is unstable, I randomly selected 50 different sets, which is split in 50 different ways. Then the 50 sets of data were inputted into my model to get 50 sets of accuracy and then averaged these 50 sets of accuracy to get the average accuracy. There is a total of 2 groups of data, named Group 1 and Group 2. Each group contains 50 different sets of data. I used Group 1 and Group 2 to test my ANN model with different techniques, and results are shown in Table 6. What's more, I also used Group 1 and Group 2 to test the performance of my 3 different neural models, which is shown in Table 7.

3.1 Results of My ANN Model with Different Techniques

According to the first and second rows of Table 6, when no technique is used, the accuracy of my ANN model is very poor, and the test accuracy is only about 60%.

Based on the third and fourth rows of Table 6, but when I used PCA to compress feature, the training and test accuracy of my ANN model improves, and the test accuracy is more than 71%. And the variance of accurate also reduces, which means that the stability of prediction also improves.

The fifth and sixth rows of Table 6 are the results of adding 'ReduceLROnPlateau' to my ANN model. The accuracy and variance of accuracy do not change significantly.

According to the seventh and eighth rows of Table 6, after pruning neurons, the test and training accuracy of the model sharply decreases, and the variance of the accuracy also doubles. This means that directly removing similar neurons from my ANN model without continuing training will decrease the prediction ability of my ANN model.

The ninth and tenth rows of table 3 are the accuracy generated after I continue to train the ANN model with the neurons removed. After training, the accuracy of my ANN model improves. And the test accuracy is about 1% higher than the original model without the neurons cut off because after some neurons of my ANN model are subtracted, the complexity of the model decreases, and the overfitting of my ANN model decreases. The variance slightly reduces, which means that the stability of the model also slightly improves. After experiments, the time to predict a set of data reduces from 1.995 ms to 0.998 ms, after pruning partial neurons in my ANN model. Thus, pruning neurons could also improve the efficiency of the model.

Based on table 3, the best average test accuracy of my ANN model is 74.83%, which is in the ninth row and belongs to group 1. I use PCA to reduce the dimension of the data, use 'ReduceLROnPlateau' to adjust the learning rate, delete similar neurons after the model training, and retrain the model to get the best prediction accuracy.

Table 6. Accuracy of my ANN model with different techniques

Index	**Test Group**	**Technique**	**Averaged Training Accuracy**	**Averaged Test Accuracy**	**Variance of Test Accuracy**
1	Group 1	None	83.14%	62.32%	64.64
2	Group 2	None	83.96%	61.54%	64.72
3	Group 1	PCA	84.26%	73.74%	58.27
4	Group 2	PCA	84.84%	71.81%	59.44
5	Group 1	PCA, ReduceLROnPlateau	84.12%	73.81%	58.41
6	Group 2	PCA, ReduceLROnPlateau	85.35%	71.69%	58.98
7	Group 1	PCA, ReduceLROnPlateau, Prune Neurons (no continue training)	63.42%	56.38%	159.93
8	Group 2	PCA, ReduceLROnPlateau, Prune Neurons (no continue training)	65.59%	59.35%	127.78

9	**Group 1**	PCA, ReduceLROnPlateau, Prune Neurons (continue training)	84.21%	**74.83**%	**44.95**
10	Group 2	PCA, ReduceLROnPlateau, Prune Neurons (continue training)	84.57%	72.06%	58.77

3.2 Results of My RNN Model

The first and second row of Table 7 is the result of my ANN model with all techniques. The third and fourth rows are the result of my RNN model. For Group 1 data, the averaged test accuracy of my RNN model is similar to my ANN model. For Group 2 data, the averaged test accuracy of my RNN model is 2% higher than my ANN model. Therefore, my RNN model has slightly higher prediction ability than my ANN model. However, the training accuracy of my RNN model is about 90%, which means my RNN model is overfitting. The good prediction performance of RNN also shows that the idea of taking each student's score in the dataset as a sequence and using RNN to train all sequences is feasible.

3.3 Results of My Bi-LSTM Model

The fifth and sixth row of Table 7 is the result of my Bi-LSTM model. Group 1 and Group 2 data all generated a higher average test accuracy than my ANN and RNN model. And my Bi-LSTM model also has the lowest averaged training accuracy, which is close to the averaged test accuracy. So, my Bi-LSTM model is not overfitting. Moreover, my Bi-LSTM model has the smallest variance and is more stable. In summary, my Bi-LSTM model has the best performance in predicting final total mark of students.

Table 7. Accuracy of my ANN, RNN and Bi-LSTM model

Index	Test Group	Model	Averaged Training Accuracy	Averaged Test Accuracy	Variance of Test Accuracy
1	Group 1	ANN	84.21%	74.83%	44.95
2	Group 2	ANN	84.57%	72.06%	58.77
3	Group 1	RNN	88.26%	74.69%	44.23
4	Group 2	RNN	89.38%	73.92%	47.69
5	Group 1	**Bi-LSTM**	80.21%	**76.72%**	33.41
6	Group 2	Bi-LSTM	79.74%	76.18%	35.21

In the original paper [1], the highest test accuracy of the neural network is 62.3%, and the average test accuracy is 53.8%. However, the average test accuracy of my ANN model is more than 72%, the average test accuracy of my RNN model is more than 73%, and the average test accuracy of my Bi-LSTM model is more 76%. Therefore, compared with the original paper, the prediction accuracy of my all models improves significantly, and my all model could predict the final total mark of students better.

4 Conclusion and Future Work

I build an ANN, an RNN, and a Bi-LSTM model to predict the final total score based on the assignments and mid-term exam scores of students. My ANN model uses PCA to improve accuracy and 'ReduceLROnPlateau' to speed up converging and pruning neurons to increase model efficiency and accuracy. The test accuracy of my three models is over 72%. Especially, my Bi-LSTM Model has the highest test accuracy, which is more than 76%. The prediction accuracy is significantly improved compared to the test accuracy in the original paper [1].

There are still many aspects to improve in my models. For example, the accuracy of prediction is still unstable, and my ANN and RNN model still has some overfitting. In the future, a verity of methods could be used to solve these problems and improve the accuracy of my model. For instance, add more features to the dataset, such as the tut-group of students. Implement more techniques such as mini-batch or dropout layer, which could increase the accuracy or reduce the overfitting of my model. Attempt to prune neurons on my RNN and Bi-LSTM model.

Reference

1. Choi, E. C. Y., Gedeon, T. D.: Comparison of extracted rules from multiple networks. In: Proceedings of ICNN'95-International Conference on Neural Networks, Vol. 4, pp. 1812--1815. IEEE (1995)
2. Schittenkopf, C., Deco, G., Brauer, W.: Two strategies to avoid overfitting in feedforward networks. Neural networks. 10(3), 505--516 (1997)
3. Torch. Optim, https://pytorch.org/docs/stable/optim.html
4. Gedeon, T. D.,Harris, D.: Network reduction techniques. In: Proceedings International Conference on Neural Networks Methodologies and Applications, Vol. 1, pp. 119--126 (1991)
5. Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs, http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/
6. Kingma, D. P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
7. Many to One RNN with Variable Sequence Length, http://www.easy-tensorflow.com/tf-tutorials/recurrent-neural-networks/many-to-one-with-variable-sequence-length
8. Graves, A., & Schmidhuber, J.: Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural networks, 18(5-6), 602--610(2005)
9. Bi-LSTM, https://medium.com/@raghavaggarwal0089/bi-lstm-bc3d68da8bd0
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation, 9(8), 1735-1780 (1997)