Applying BiDirectional Neural Networks on Mark Prediction

Cheng Yixin

Research School of Computer Science,
The Australian National University, Canberra, Australia,
u6809382@anu.edu.au

Abstract. Neural network models are constantly evolving. As human being's digital footprint grows, students' academic information is stored in a digital way. Based on these, using neural networks to predict student marks is becoming trendy. (Patron, 2018). In this paper, a technique named BiDirectional Neural Networks (BDNN) is introduced for predicting student's final grades which reproduced by this paper from the A.F. Nejad and T.D. Gedeon's work[1]. The first step of implementation is the encoding modification, followed by applying the processed data set on a basic neural network, encoding remodification, applying the new data set on BDNN, model evaluation, and comparison. In the end, the results of the model in this paper is worse than the original paper indicated in terms of prediction accuracy. This paper also found that both two results are worse than the results produced by the basic neural network. Thus, this technique is not very promising on this data set.

Keywords: BDNN, Mark Prediction

1 Introduction

1.1 Background and Motivation

Improving student's academic performance is always the priority and challenge for university. Predicting student's performance can decrease the rate of drop-out, improve the success rate of students and make cost-saving decisions for institutions since it might allow institutions to take pre-emptive measures to help students [**?**,**?**]. Therefore, the prediction of performance is vital to universities and students themselves. As for BDNN, it is similar to electric transmission in the real world. it has been widely applied in various fields including generalization error reduction [**?**,**?**], confidence estimation[**?**,**?**,**?**], speech recognition [**?**,**?**] and so on. This paper will address the implementation of BDNN on Student Final Marks (SFM) classification by using the partial marks of students related to the course and compare the results with the simple neural network.

1.2 Problem Statement and Investigation Outline

Compared with a simple neural network, BDNN enables output layer learning from backward and forwards. By building one-to-one relations between input vectors and output vectors, inputs and outputs neurons can connect hidden layers with two directions. However, whether BDNN is suitable for the classification of marks is to be determined. According to the given paper, the authors implemented BDNN on two data set, that is SFM and Gross National Product (GNP) prediction. During the process, the authors proposed two tasks, that is, context addressable memory and cluster. To achieve these, the authors created 3-layer neural networks (one input, one output, and one hidden layer), introduced the same weight connections between layers, implemented the back-propagation technique in BDNN to get weights adjustment, built a one-to-one relation data set for implementation, etc. Finally, the authors got 74% accuracy on SFM for the first task. As for the last task, the authors trained BDNN as a cluster center finder, achieved 72% on SFM and 62% accuracy on GNP (Tom. 1992). However, the given paper did not compare its performance with the simple neural network. Thus, the objective of this paper is to reproduce BDNN on SFM and validating BDNN by comparing it with the simple neural networks.

1.3 Activities

To achieve the objectives, this paper carries out the following tasks:
Build a simple neural network that classifies the students' final marks. Build a data set contains students' academic information (marks from different assessment in COMP 1111) from the origin data set. Build a simple neural network. Test the data set on the built neural network. Build BDNN that classifies the students' final marks. Build a suitable dataset contains students aggregated academic information which can be used on BDNN. Build BDNN. Test the data set on the built neural network. Compare the results between BDNN and simple neural network

1.4 Outline

Section 2 will introduce the methods this proposed used. Section 3 will

2 Methodology

2.1 Data Collection and Pre-processing

The given data is COMP 1111 students' grades from the University of New South Wales (UNSW)includes 153 students' academic records with 16 columns, that is, Regno (registration number), Case/Prog (course/program), S (semester), ES (enrolment status), tutorial group, lab2, tutorial assessment, lab4, homework1, homework2, lab7, p1, f1, mid, lab10 and final. Note that all of the data is shown in one column in the given data set. Moreover, there is 327 missing value throughout the data set. As a result, the first step is to split and reorganize the organize data set. After this step, a new data set generated, and values are across different columns and rows. In each of the columns, the format of values varies. For example, in the Cre/Prog, ES, Tutgroup columns, the values are in the string. In Regno and final, the values are integers. So next step is to transform data and detect outliers. For the outliers, I replaced them with NaN which means Null in the data set. Furthermore, I normalize the data to [0-1] in float or integer column based on their full mark, for example, a student got 2 marks in lab2 (the full mark is 3) so this value is 0.67. based on this principle, I transformed all numeric data. As for string type value, I created categories to represent them. To be specific, for Turtgroup, a scale from 1 to 10 represents 1-10 tutorials then transform them into 0-1. For Cre/Prog, get the course number then transform them into 0-1. Ford ES, transform it to number from 1-3 respectively representing FS and FL then transform them into 0-1. For grades, a scale from 0-1 represents from 0%-100% respectively. Then I removed the first column since it is meaningless as input. In the final step, transform the final column from 1 to 4 to represent F to D.

As for missing data imputation, first, I tried to keep the NaN values in the data set, however, the result is quite low and unsatisfying. Therefore, I used the Random Forest algorithm to implement because for grades since this method can have a better performance than others in terms of imputation[**?**,**?**,**?**]. To be specific, the order of imputation is from the least column to the most ones first, fill the missing values with 0 in other columns then run the algorithm and do the iteration until each missing value will be handled. Then, I normalize all the input data except the final with the Z-score method.

2.2 Simple Neural Network Construction

To build the simple neural network, I used the first 14 columns as input, which means input size is 14. Since output is 4 (F, P, C, D), so the output size is 4. Other hyper Parameters includes batch size (10), learning rate (0.01), hidden size (50), epochs (70). The optimizer is Adam and I conduct a backpropagation to adjust weights. 80% training data and 20% testing data. Overall, this combination is the most effective and satisfying.

2.3 BDNN Construction

To build the BDNN, I remodified the training data set and testing data set to one-to-one relation (each output vector has only one representative input vector) for fitting this model. To do that, categorizing final mark as 4 classes, as for each type, representing it with an integer. Therefore, getting the mean of 4 classes. After that, creating 3 sub-classes for Fail, 6 sub-classes for Pass, 6 sub-classes for Credit and 3 classes for Distinction respectively. For example, I create sub-classes for fail is F1 = 0.9 * F - 0.1 * P, F2 = 0.8 * F - 0.2 * P and F3 = 0.7 * F - 0.3 * P. Therefore, 22 patterns and 15 columns one-to-one relation data set was created. Then, in terms of hyper parameters, input size = 14, hidden size = 50, num classes = 1 epochs = 150 batch size = 14 learning rate = 0.01 and extra bias (initialized into 0) combined to the model. The optimizer is AdamW and I conduct a backpropagation to adjust weights. 80% training data and 20% testing data. As for training, the first 50 epoch, BDNN was trained as a simple neural network. As for the next 50 epochs, the direction of training is reversed. In the final 50 epochs, the direction is reversed again to complete the whole training.

3 Results and discussion

The results in simple neural network and BDNN vary. In the simple neural network, the testing accuracy is 89% at most with loss 1.4 in testing which means simple neural network successfully classified 88% of student marks in testing. in the BDNN, the testing accuracy is 64% which means BDNN only correctly classified 64% of student marks in testing and corresponding loss is 0.18. Although the BDNN model correctly majority of marks, compared with simple neural network and results on given paper, the result is unsatisfying. Compared with given paper, this result is about 10% less (74% in given paper) [1] which is also unsatisfying. The latent reason why this result happened, I suspect the control of direction reverse is problematic, I did a lot of experiments. For example, I set up a loss threshold, if the over loss in one direction training exceeds the threshold, the direction will be reversed. I set up it to 1.1, then the accuracy is 42%. Then I did another experiment, the condition of direction will not reverse until the over loss is less than the over loss in previous direction of training, the testing accuracy is 33%. Overall, 3 ways have been tried and the first one has the best performance. However, it is unknown that which way the given paper used and what corresponding set-up was. The reason why simple neural network performed well is due to the imputation by Random forest which this algorithm played a key role in the training and proper settings of hyper parameters. According to the result shown, the BDNN is not suitable for this data set.

4 Conclusion and Future Work

A mark prediction based on neural network has been carried out respectively. In the simple neural network, the result is good, and this model can deal with this data set. But BDNN preformed worse than simple neural network on current data set and this paper invalidates the BDNN. As for the future work, there are many improvements, such as softening the sharp edges between subclasses to make them more natural interval, improving the encoding format, applying more advanced technique in training etc. Most importantly, I can make more improvements on the direction reversing because I think this leads to the low accuracy. In terms of extensions, this technique can be extended many data sets such as predicting climate with some import index or classifying problems.

References

1. A. F. Nejad and T. D. Gedeon : Bidirectional neural networks and class prototypes," Proceedings of ICNN'95 - International Conference on Neural Networks. IEEE(1995).

2. Nejad A.F., Gedeon T.D.: Bidirectional Neural Networks reduce generalisation error. IWANN (1995).
3. A. Ragni, Q. Li, M. J. F. Gales and Y. Wang. : Confidence Estimation and Deletion Prediction Using Bidirectional Recurrent Neural Networks. IEEE(1999).
4. A. Graves, N. Jaitly and A. Mohamed. : Hybrid speech recognition with Deep Bidirectional LSTM. IEEE(2013).
5. Smith B.I., Chimedza C., Bührmann J.H. : Random Forest Missing Data Imputation Methods: Implications for Predicting At-Risk Students. ISDA(2019).
6. Patron,R. : Earlyschooldropoutsindevelopingcountries:Anintegerapproach to guide intervention: The case of Uruguay. Journal of Economics (2008).