A Method of Determining Needy Students Based on Deep Learning

Xiao Wang, Sikai Wu and Quanyin Zhu*

Faculty of Computer & Software Engineering, Huaiyin Institute of Technology, Huaian, China
*Corresponding author’s email: hyitzzy@126.com

Abstract. For a long time, stipend has been send inaccurately due to China’s national conditions. Machine learning has been used to find impoverished students and achieved some results, but it was unsupervised and the accuracy was difficult to measure. In order to ensure the stipend being distributed fairly, a method based on deep learning is presented in this paper. During the experiment, accuracy rate was reached to 96%, through use the data came from school one-card and school library. The results show that this method can effectively improve the accuracy of grants, can save labor cost.

1. Introduction

After decades of development, China has formed a subsidy policy for needy students based on grants, inspirational scholarships, and student loans [1]. However, compared with the developed countries in the West, due to the lack of a sound personal tax system, it is very hard for China to realize the precise subsidy for needy student. At the same time, due to the large population base, it is also impossible to conduct one-by-one visits to the surveys, making it difficult to achieve the precise subsidy for needy students. How to objectively and effectively identify the needy students remains a very difficult task [2]. The widespread use of campus one-card has made it possible to determine needy students scientifically. By using the students' consumption data, library borrowing data and grade data could effectively improve the accuracy of the identification of needy students.

Nowadays, deep learning has been widely used in spam identification, disease judgment, image recognition, voice translation and has achieved remarkable success [3]. If the depth learning algorithm is applied to the identification of needy students, it will be able to improve the accuracy of subsidies for needy students and save workforce survey costs.

1.1. Neural Network Algorithm

The neural network algorithm was invented in 1957, when it was called the Perceptron, which is a simple feedforward neural network, or a binary linear classifier. In the 1980s, the development of neural networks flourished. In the United States, Japan, Europe appeared a wave of neural network research. Neural network algorithms are beginning to be used in areas such as image recognition, voice recognition and home robots. However, due to the limitation of GPU, the development of neural network is rapidly stagnating. In the 1990s, the neural network algorithm development was slow, while machine learning developed rapidly and was widely used. Until the advent of deep learning algorithms, the neural network thrived again, and advanced deep learning algorithms like CNN (Convolutional Neural Network) and LSTM (Long Short-Term Memory) soon appeared [4]. Now our lives are being influenced by deep learning outcomes.
The main structure of neural network algorithm is input layer, hidden layer and output layer. The input layer handles inputting eigenvalues, while the output layer is for outputting predictions or classification results. The hidden layer is where the neural network is located and consists of connected neurons [5]. In depth learning, in order to deepen the neural network, increase the neural network layer, the introduction of the activation function. Activation function through the abandonment of some neurons, to deepen the neural network and improve the accuracy of the prediction or classification purposes. Figure 1 shows the structure of a typical neural network.

![Neural network schematic diagram](image)

**Figure 1.** Neural network schematic diagram

1.2. Cross-entropy Loss Function

The cross-entropy loss function measures the similarity of real and predicted results [6]. Compared with other loss functions, the cross-entropy loss function is used to update the weights between neuronal connections to reduce training errors. Compared with the variance loss function, the cross-entropy loss function overcomes the problem of slow learning speed [7]. Mainly use as the loss function in the output layer whose activation function is Sigmoid or Softmax.

The formula for the cross-entropy loss function is:

\[
H(p, q) = - \sum_x p(x) \log q(x)
\]  

(1)

Where \( p \) is the distribution of real markers and \( q \) is the predicted marker distribution of the trained model. Equation represents the difference of two functions or probability distributions: the larger the difference, the greater the relative entropy. The smaller the difference, the smaller the relative entropy will be. In particular, if the two are the same, the entropy is 0.

1.3. Softmax Function

The Softmax function is used to handle classification or regression problems of three or more categories, and algorithms using Softmax function are supervised learning [8].

\[
\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{k} e^{z_k}}
\]  

(2)

Equation (2) is the Softmax function, Softmax function squashes the outputs of each unit to be between 0 and 1. The output of Softmax function is a set of probabilities, and the total sum of the outputs is equal to 1 [9].
1.4. Sigmoid Function

Sigmoid function is a mathematical function having a characteristic "S"-shaped curve or sigmoid curve. Equation (3) is the definition of Sigmoid function, the picture shows its shape. The range of function values is from 0 to 1 [10]. We use Sigmoid function as activation function of first layer of neural network, to achieve the similar purpose of dataset normalization [11]. Compared with the Tanh function, using the Sigmoid function as the first layer neural network can make greater use of the features, and its result has the characteristics of probability, which is more conducive to the classification calculation of the Softmax function.

\[
S(x) = \frac{e^x}{(e^x + 1)}
\]  

(3)

Figure 2. Sigmoid function curve

1.5. ReLU

ReLU refers to Rectified Linear Unit, Equation (4) is the definition of ReLU. The function value is 0 when \( x < 0 \) and the function value is equal to \( x \) in other cases [12]. Deep learning network for a long time did not reach the accuracy in machine learning, the reason was that the Sigmoid function for a long time to connect neurons as the activation function, until ReLU as activation functions of neural network. Krizhevsky et al. found that the convergence rate of SGD obtained by ReLU was much faster than that of Sigmoid. Compared to Sigmoid, ReLU only needs a threshold to get the activation value, without having to calculate a bunch of complicated operations. Some people say this is because it is linear and non-saturating. The use of the ReLU function makes the depth of the neural network deepen and improve the accuracy, although the ReLU function is very simple [13].

\[
f(x) = \max(0, x)
\]

(4)

1.6. RMSprop Optimizer

RMSprop is a self-adaptive learning rate method proposed by Geoff Hinton, and is an improved version of the SGD optimizer. RMSProp performs better under non-convex conditions, altering the gradient averaging to a moving average of exponential decay to discard distant past histories. The use of RMSprop optimizer makes the neural network converge faster. Empirically, RMSProp has proven effective and useful in deep learning network optimization algorithms. RMSprop optimizer always keeps a moving average over the root mean squared gradients; by which RMSprop divide the current gradient [14]. Equation (5) shows how to update velocity of, \( v_{(w,t)} \) refers to velocity function of weight and times, \( v_{(w)} \) refers to loss of the gradient, and \( \gamma \) refers to a constant which is slight then 1. Equation (6) shows how to update weigh, \( \eta \) refers to learning rate.

\[

\]
\( v(w, t) := \gamma v(w, t-1) + (1-\gamma)(\nabla Q(w))^2 \) \hspace{1cm} (5)

\[ w := w - \frac{\eta}{\sqrt{v(w, t)}} \nabla Q(w) \] \hspace{1cm} (6)

2. Experimental Preparations

2.1. Data Preparations

Experimental data sources came from an algorithm race of Data Castle website (http://www.pkbigdata.com), the race named University Student Financial Aid Funding Forecast. Experimental data includes: students’ one-card consumption data, library borrowing data, student grade data, and the data of financial aid for needy students. In the experiment, the characteristic value was extracted, and the consumption data was extracted by category, and amount and frequency of total consumption, water, meal, supermarket, bathing, laundry, bus, printing, book, school, other, empty consumption was extracted. The library data counted the number of borrowings, and the scores were ranked according to the weighted average. The student financial aid is processed by one-hot code in different amounts. Figure 3 shows some of student features we used in our database table.

![Figure 3. Some of student features](image)

2.2. Experimental Environment

The machine configuration is win10/Ubuntu16.4, CPU is Intel I5, GPU is gtx1050ti, and RAM is 8G.

The experimental language is python3.6, database is mysql5.7.20 the neural network API is keras2.1 and use TensorFlow as its tensor manipulation library.

2.3. Data Training

The activation function used in the first layer of the neural network is the Sigmoid function, and the rest is ReLU active function. The output layer is four neurons, using the Softmax activation function.

Definition 1: Set the student feature matrix given by Equation (7), \( x_i \) has Twenty-six dimension, include water, canteen, supermarket, laundry, medical, public transportation, printing, books, schools, other, empty consumption and frequency, and the score of students, the library borrowing data. \( x_i \) means the \( j \)'th student of the \( i \)'th feature.

\[ STUDENT = \{\{x_1^1, x_1^2, \ldots, x_1^{26}\}, \ldots, \{x_j^1, x_j^2, \ldots, x_j^{26}\}\} \] \hspace{1cm} (7)
Definitin2: Set Weight matrix given by Equation (8), \( w^i \) means the \( k \)th Weight of the \( i \)th neuron. The first layer of neural network uses Sigmoid activation function, so matrix do not contain weight of the first layer so do the bias.

\[
Weight = \{[w^1_1, w^1_2, ..., w^1_n],...,[w^1_1, w^1_2, ..., w^1_n]\} \tag{8}
\]

Definition3: Set bias matrix given by Equation (9), \( b^i \) means the \( k \)th bias of the \( i \)th neuron.

\[
Bias = \{[b^1_1, b^1_2, ..., b^1_n],...,[b^1_1, b^1_2, ..., b^1_n]\} \tag{9}
\]

The Equation (10) shows how to get the result of each \( x^i_j \). And the first layer of neural network result is given by Equation (11). \( S \) means Sigmoid function.

\[
x^i = S(w^i_j x^j_b + b^i_j) \tag{10}
\]

\[
x^i = \{[x^i_1, x^i_2, ..., x^i_n],...,[x^i_1, x^i_2, ..., x^i_n]\} \tag{11}
\]

Definition3: Set real stipend matrix given by Equation (12), \( \hat{y} \) means the \( i \)th student’s real stipend.

\[
r - stipend = \{\hat{y}_1^i, \hat{y}_2^i, ..., \hat{y}_n^i\} \tag{12}
\]

Definition4: Set experimental stipend matrix given by Equation (13), \( y^i \) means the \( i \)th student’s experimental stipend.

\[
e - stipend = \{y^i_1, y^i_2, ..., y^i_n\} \tag{13}
\]

Then we could know the loss is \( H(\hat{y}^i, y^i) \), and the loss function is related to Weight and Bias, so we get the function of function of Weight and Bias. Set the Equation (14) is this function.

\[
F(w,b) = f(\hat{y}^i, y^i) \tag{14}
\]

And we could update Weight and Bias with optimizer function to get lower loss.

The step for model training as follow:

Step 1. The processed student eigenvalues are used as data, set \( STUDENT = \{[x^1_1, x^1_2, ..., x^1_n],..., [x^i_1, x^i_2, ..., x^i_n] \} \) each student has Twenty-six dimension; these dimensions came from student card.

Step 2. First batch of data through the first layer of neural network, student features are transformed into decimals between zero and one, to facilitate the Softmax function to calculate probabilities. Get the layer of neural network result is \( x^i = \{[x^i_1, x^i_2, ..., x^i_n],..., [x^i_1, x^i_2, ..., x^i_n] \} \).

Step 3. First batch of data go through other layers, complete a training, get training result is \( e - stipend = \{y^i_1, y^i_2, ..., y^i_n\} \). Due to the presence of the ReLU function, the neuron will discard the neuron output with a result equals to zero.

Step 4. Calculated from the loss function gets the function of weight and bias, which is \( F(w,b) = f(\hat{y}^i, y^i) \).

Step 5. RMSprop optimizer calculates the direction of weight and bias update weight and bias such that loss decreases.

Step 6. The training is completed; the next training and the above steps are the same.

Step 7. Save model parameter, which will be used in determine needy student.

2.4. Model Training

The first layer of the neural network contains 26 input nodes (feature value extraction number) and 200 output nodes, and the activation function we used is sigmoid function. The hidden layer includes 5 layers of the same neural network. Using ReLU activation function, the input nodes and output nodes
are all 200. The output layer output node number is 4, use Softmax activation function. Besides we set learning rate 0.001 and batch size 200.

We have tried different neural network structures; the number of hidden layers was chosen from 2 to 50. When using 5 hidden layers, we receive the best result. We also tried other structures to determine the batch size and learning rate etc.

3. Experimental Results
Table 1 is the description of Experimental dataset. Dimension refers to number of the features we used, student category is one-hot coded, number of sample refers to each category students’ number, and stipend refers to the number of stipend each category students get.

| Stipend | Dimension | Number of sample | Category name |
|---------|-----------|------------------|---------------|
| 0       | 26        | 350              | Group1        |
| 1000    | 26        | 350              | Group2        |
| 1500    | 26        | 350              | Group3        |
| 2000    | 26        | 350              | Group4        |

In order to ensure the balance of the data, 350 data were randomly selected from four categories of students for training.

Figure 4 shows one result of the experiments, we set the batch size to 200 and the number of epoch to 400. We could see after 400 epochs training, accuracy could reach about 96%, and loss is about 0.1.

| Optimizer | Average accuracy | Average loss | Rating  |
|-----------|------------------|--------------|---------|
| RMSprop   | 96.38%           | 0.1377       | best    |
| Adam      | 92.25%           | 0.9010       | good    |
| Adadelta  | 42.54%           | 1.2758       | bad     |
| SGD       | 55.46%           | 1.0342       | worst   |
Table 2 is the experimental result describe. We did about 40 times comparative experiments. We could see result by using RMSprop optimizer achieves best result, get highest accuracy and lowest loss. In contrast, other optimizers are not as good as RMSprop optimizer, especially Adadelta optimizer and SGD optimizer.

4. Experiments Analysis
The data of 10,885 students were used for feature extraction, which came from 1246588 consumption records, 1013,746 library lending records and 10,885 grade data. There are also 10,885 financial aid data for experiments. The best accuracy of the experiment reached 97.07%. The results of the comparison experiment were better than those of the other 4 experiments. It can provide financial aid for schools and improve the fairness of financial aid distribution.

5. Conclusions
According to the experimental results, we can see that the classification accuracy can reach 96%, and sometimes the results will be better. The worst results in more than a dozen trials were also higher than 90 percent, and the results were satisfactory. This approach requires further research. For example, some schools cannot obtain and experiment with the same type of data and may need to make adjustments as the data is processed.

6. Acknowledgment
The work described in this paper was supported by National Undergraduate Innovation and entrepreneurship training program(201711049006Z). We would like to express our thanks to the anonymous reviewer for their constructive comments.

7. References
[1] Zha, Qiang, and D. Wang. "Case Study: The Chinese Government Scholarship Program—the Brain Development Scheme That Illuminates a Vision Across 30 Years." (2018).
[2] Chua, Catherine S. K. "Education policy borrowing in China: has the West wind overpowered the East wind?" Compare A Journal of Comparative & International Education 45.5(2014):1-19.
[3] Pan, Lu, et al. "The Influence of Training Step on Price Forecasting Based on Back Propagation Neural Network." Fifth International Conference on Multimedia Information NETWORKING and Security IEEE Computer Society, 2013:785 -788.
[4] Courbariaux, Matthieu, and Y. Bengio. "BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1." (2016).
[5] Graves, A, et al. "Hybrid computing using a neural network with dynamic external memory.” Nature 538. 7626 (2016): 471-476.
[6] Hinton, Geoffrey, O. Vinyals, and J. Dean. "Distilling the Knowledge in a Neural Network." Computer Science 14.7 (2015): 38-39.
[7] Gou, Xunjie, Z. Xu, and H. Liao. "Hesitant fuzzy linguistic entropy and cross-entropy measures and alternative queuing method for multiple criteria decision making." Information Sciences s 388–389 (2017): 225 - 246.
[8] KAGAN TUMER, and JOYDEEP GHOSH. "Error Correlation and Error Reduction in Ensemble Classifiers." Connection Science 8.3-4 (2015): 385 - 404.
[9] Campuzano, Francisco, et al. "Generation of human computational models with machine learning." Information Sciences 293.293(2015):97-114.
[10] Donate, Juan Peralta, et al. "Time series forecasting using a weighted cross-validation evolutionary artificial neural network ensemble." Neurocomputing 109.8 (2013): 27-32.
[11] Ito, Yoshihisa. "Approximation Capability of Layered Neural Networks with Sigmoid Units on Two Layers." Neural Computation 6.6 (2014):1233 -1243.
[12] Zagoruyko, Sergey, and N. Komodakis. "Wide Residual Networks." (2016).
[13] Han, Song, et al. "EIE: Efficient Inference Engine on Compressed Deep Neural Network." 44.3(2016):243-254.
[14] Mukkamala, Mahesh Chandra, and M. Hein. "Variants of RMSProp and Adagrad with Logarithmic Regret Bounds." (2017).