Deep Learning

Project Report:

Dasheng Gu
Jingwei Shen
Xinyuan Wang

April 24th 2018
Instructor: Joseph E. Beck

Abstract

The main topic of this project is Deep Learning on Education. Deep Learning is a method of machine learning, it can be supervised, semi-supervised or unsupervised. The problem we are trying to solve is to tweak and improve several neural networks for prediction of student correctness on questions. The approach we adopt to solve the problem is to compare different features, such as drop rate, number of hidden layers, hidden nodes, with/without autoencoder, optimized function, batch size, and if neural network is fully connected so that we can improve AUC (Area under Curve) to get the better prediction results. We had experimented on testing the performance of single output and multiple output network respectively. Based on the data we collected, it turns out that most of the performance of AUC in single output is better than the performance in multiple output network. At the same time, RMSE (Root-mean-squared error) performance on multiple output network is better than the performance on single output network. One obvious aspect of the experiments is that the wheel-spin performance decrease from averaged 0.85 to 0.66, which is a noticeable change when applying multiple output network.

Key words

Deep learning, Neural networks, Autoencoder, Competence, Affects, TensorFlow.

Introduction

Deep learning is one of the hottest topics in computer science. It is an algorithm simulating the biological neural networks structure to discover the relationship between data distributions. Neural networks are mathematical models that consist of multiple layers of nodes. When data is passed through those nodes, each of them is multiplied by a weight and add a bias value. At last, the data is tuned and outputted.

In our experiment, the process of data flowing through the network is represented by student's information going through the network and finally outputting the competence of the student or the effects of the student. From the earlier approach, we have several factors that show the competence of students: wheel spin, retention, and next problem correctness (NPC). Besides, the factors that might influence students' performance are 4 affective states: confusion, boredom, concentration, and frustration.

In this project, one of the accessory library sources we applied is TensorFlow. It is an open source math library published by Google, which is named by its working principles. Tensor means multidimensional array and flow means tensor is computed from one end of the data flow graph to the other one. TensorFlow can be used in aspects like image or sound identification or other deep learning fields. This is used in our first project for classifying pictures of the alphabet. Our team did not construct the network from the basic level. We used Network Builder which is

a wrapped-up package written by Anthony Botelho and Seth Adjei, so we can build the customized network model with only a few lines of codes.

In this project, our goal is to optimize the performance of this network on predicting students' future performance so that we can obtain an overall display of their competence as an output. To achieve the goal, we need to make comparisons between outcomes with modifications on the factors and other values (e.g., the numbers of the hidden layer) to find a more optimal solution to have a closer prediction.

First experiment: Recognizing alphabet graph

To start with, let's talk about another experiment as a warm-up to deep learning. Provided by the online course on Udacity, this experiment asked us to train a network to classify pictures of an alphabet. To build the network, we applied logistic regression model from sklearn, which is a library implementing several machine learning models including logistic regression model for python. We trained the model for several sets. After that, we let this model goes through the test dataset to see how accurately it predicts the correct labels.

Logistic regression

Logistic regression can be considered as a branch of linear regression. It can be used in fields such as machine learning, medical field, and social science for prediction where the output can only take two values. With Logistic Regression function, we can take any real number and turn it into a number between 0 and 1 indicating the probability.

$$f(x) = \frac{1}{1+e^{-x}}$$

Figure 1: Logistic Regression Function

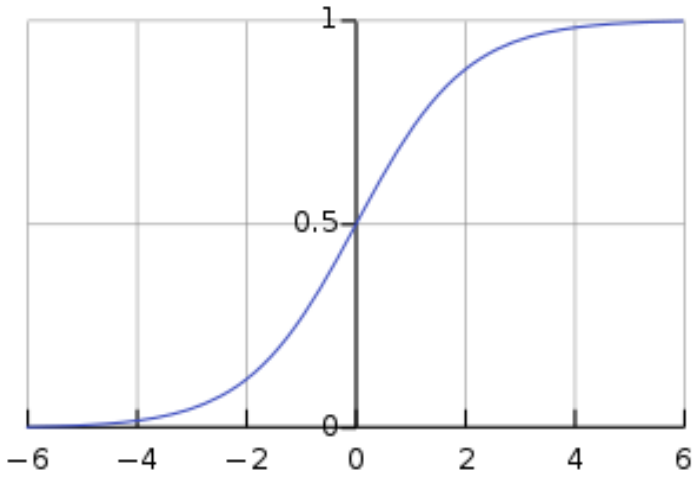


Figure 2: The logistic regression curve

Figure 1 is the formula of logistic regression function, where x stands for any real input. Figure 2 is a graph of the logistic function on the t-interval (-6, 6).

Implementation

The implementation of the model is from Udacity notMNIST tutorial. We followed the 5 following steps: 1) Download the data through the internet; 2) extract, process, 3) store it; 4) train the network; 5) print the result. This sklearn package has already implemented the model of logistic regression, so, we did not need to worry about that.

Conclusion

The output of this experiment only contains the accuracy of the prediction on the test set of alphabet. Our result turned out that the accuracy rate of our neural network is 0.85, which indicates that it successfully classifies the test set of alphabet labels. This experiment is a perfect warm-up for machine learning beginners. By successfully going through this experiment, we understand that machine learning has several basic steps: read inputs, build a model, train network and predict outcomes.

Second experiment: Predicting and optimizing single output network

In this experiment, instead of sklearn, we used TensorFlow to build models and train the network. Since we built the model, we could try different models and test with different parameters. Basically, the process of training the network has four parts, processing inputs, building models, training the network and output predictions.

Processing inputs

The data we used is read from a csv file including results from high school students' survey and test answers. It has more than 500000 rows of data. Each row has student's id, question number, and 92 features. They are independent variables that will be fed into the network. Besides that, each row also has features such as problem correctness, wheel spinning, first action, retention, and affects. In this experiment, we use one of the five variables as output variable at a time. Particularly, we offset problem correctness, so it represents the correctness of the next problem (NPC). Affects represent four columns of data. They are confusion, concentration, boredom, and frustration. A student can only be one of those four affective states.

Before feeding those inputs into the network, we need to feed them into the autoencoder. An autoencoder is a neural network, as well as an unsupervised learning algorithm. By adding autoencoder, which consists of multiple hidden layers between input and output layer, our network is able to reduce noise in raw data. As Figure 3 shows, an autoencoder is a symmetric neural network which has the same numbers of inputs and outputs. It is used to optimize the model by compressing the input and reducing noise. Through our experiment, we found out that the autoencoder can increase the performance of the network by 5% to 10% and reduce the time to train a network by about 50%.

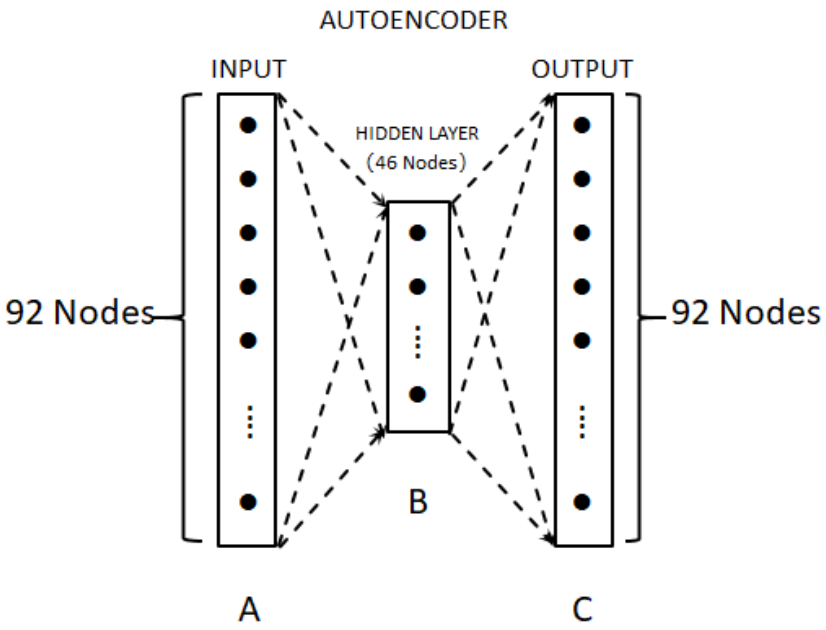


Figure 3: Model of Autoencoder

In this experiment, the autoencoder has one hidden layer with 46 nodes. The 92 features of data are fed into the input and output layer of the network. So, when the autoencoder is trained, the hidden layer, in theory, contains all information the inputs have with fewer nodes. After the autoencoder is trained, we keep A, B, and connections. Drop C and connect the rest with B and C. This pre-training process reduces the size of inputs of the main network resulting in higher efficiency and better performance. It reduces the amount of calculation the main network needs and the noise the original inputs have.

Building and training models

The models we built in this part use network builder written by Anthony Botelho and Seth Adjei, so we only need a few lines of code to construct a fully customized network model. This allows us to focus on improving the performance of the network by matching different models with different parameters.

The neural network is an algorithm simulating the biological neural system. It has nodes (hidden units) and connections corresponding to neurons and axons as shown in Figure 4 below. The nodes are places to temporarily store data (number) and connections representing weights and biases. When the inputs go through the connections as a flow, they are multiplied by the weights and added to the biases, and the sum of those results are added and stored in the next node. When these data flows to the output layer, they are compared with the actual result. Then, the difference or what we call cost entropy is calculated. Then, the network will try to adjust those weights and biases to reduce the cost. When the cost stops improving, the network is well trained.

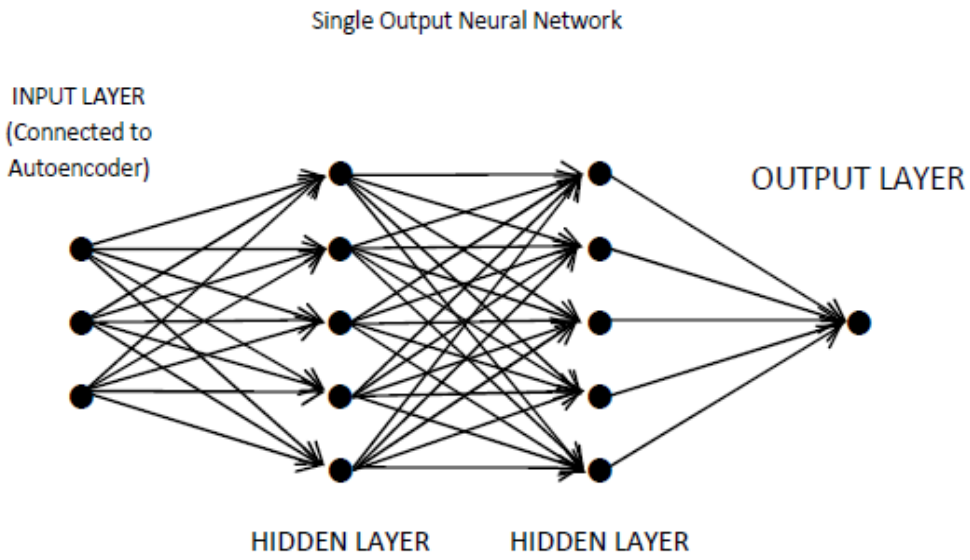


Figure 4: Model of Single Output network

Prediction

After the network is trained, we can test the network by predicting students' competence given features as the input.

Improving performance

To achieve the goal of improving the performance of the network, what we mainly concern about are reducing program running time and increasing the accuracy of the prediction.

Firstly, we test the relationship between layer number with the performance. It turns out that the running time of the 2-layer network is doubled compared to 1-layer network, but the accuracy barely increases 3-layer network took even longer. Thus, we give up finishing running the network.

Second, we try to run the network with different hidden units. We find out that more hidden units increase running time without increasing performance. We test network with 50, 100 and 200 hidden units respectively, so we end up with that 50 hidden units is the optimized network among the three.

Then we compare network with different cost functions. They are adam and adagrad. We did not find any difference in their running time but the performance of adam overwhelm adagrad.

We also test 2 different types of networks, LSTM and RNN networks. They have similar performance but RNN network cost 50% less time to train the network.

At last, the collected data indicates that dropout rate does not affect the performance if the dropout rate is within 0.3 to 0.7. Dropout is a technique that randomly drops nodes and their connected weights and biases for the neural network to reduce overfitting. An overfitted model correspond too closely to the specific data set, so the model is inaccurate when predicting with different data set.

In conclusion, the optimized model we found for this project is RNN network, 1 layer, with adam cost function.

Results

Figure 5 is the table of our collected data under single output version. Figure 6 is the screenshot of the running program outputs.

Lstm					
Output	**AUC**	**AUC SD**	**RMSE**	**RMSE SD**	**Training Time**
npc	0.664	0.013	0.424	0.008	8177.586721
ws	0.853	0.013	0.348	0.013	8173.626886
fa	0.8229033886	0.01498891367	0.1876518106	0.008458737988	8123.676001
rc	0.6702074127	0.09663329907	0.4567821284	0.009594613674	8088.927433
Rnn					
Output	**AUC**	**AUC SD**	**RMSE**	**RMSE SD**	**Training Time**
npc	0.665	0.012	0.423	0.008	5176.49858
ws	0.862397124	0.01246384088	0.3399139531	0.01275966302	5149.610425
fa	0.6752737708	0.09883234338	0.4554491826	0.01026918713	8092.341316
rc	0.8216136221	0.01577902952	0.1877416022	0.008064568051	8100.580911
Multi(4)	lstm				
Output	**AUC**	**AUC SD**	**RMSE**	**RMSE SD**	**Training Time**
4	0.711362	0.0836381	0.4222393172	0.005876949758	5888.147333

Figure 5: Result of Single Output Network

Third experiment: Predicting and optimizing multiple output network with large data set

As it shows in Figure 6, a multiple output network outputs multiple labels at once. In this case, our goal is to find out if it has higher accuracy when the network outputs multiple labels at a time. In addition, we used large datasets to see if we can get a better result. We assume that

there are relationships between those outputs. For instance, the correctness of the problem is related to if the student is confused or concentrated.

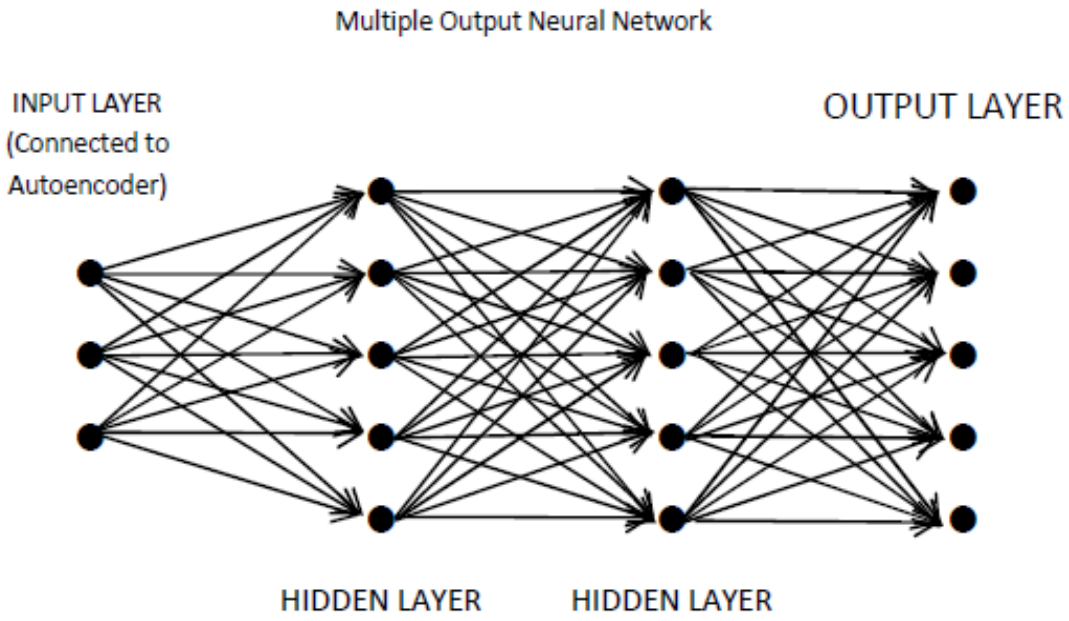


Figure 6: Model of Multiple Output Network

Different from last experiment, the prototype of the code we used was written by a WPI graduate student, Liang Zhang, and then modified by Anthony Botelho. We tested it following the similar pattern we used in the previous experiment.

First, the test results inform us that the running time of network with different hidden units shows that increasing the number of hidden units does not increase as much of running time. What's more, surprisingly, increasing the number of data does increase running time but not proportional to it. Figure 7 below is the chart built upon the data we collected.

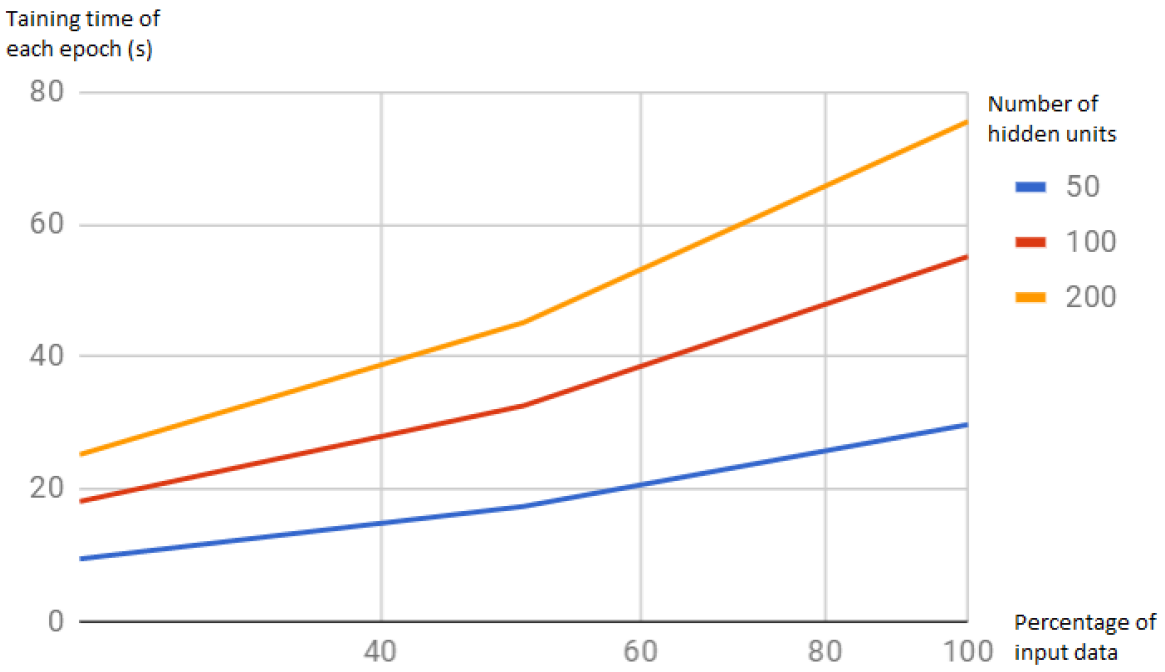


Figure 7: Training time and input data

Then we also implemented the network by modifying Anthony's code. We output all 8 labels mentioned in the second experiment at once, including next problem correctness, first action, retention, wheel spinning, and affects.The running time with larger data set increased and the result is shown in Figure 8 including AUC and RMSE results.

Results:

AUC/RMSE	npc	fa	ws	rc	af	epoch
fold 1	0.667/0.447	0.814/0.230	0.778/0.427	0.478/0.458	0.503/0.190	235
fold 2	0.659/0.452	0.770/0.225	0.115/0.468	0.635/0.420	0.763/0.224	132
fold 3	0.641/0.449	0.778/0.203	0.634/0.434	0.738/0.445	0.617/0.212	128
fold 4	0.657/0.450	0.802/0.224	0.746/0.461	0.565/0.325	0.413/0.204	109
fold 5	0.652/0.444	0.789/0.214	0.916/0.455	0.527/0.391	0.550/0.178	138
avg	0.655/0.448	0.791/219	0.638/0.449	0.589/0.408	0.569/0.202	

Figure 8: Result of Multiple Output with large data set

Conclusion

By implementing deep learning neural network, we can predict students' competence in an efficient way. Dropout rate seems to not affect the performance of the network. However, autoencoder can increase the performance at the same time reduce running time. We can find the optimized parameter for the network to build an efficient and time-saving network. Adam

cost function overwhelms adgrad cost function and a 1-layer RNN network is good enough for our purpose.

However, we have not finished trying a 3-layer network or more, or even mixed type network. It is worth trying to have multiple layer network with different combinations of numbers of hidden units and network type. Besides that, we have not done much on implementing and optimizing multiple output network which has lots of potentials. In our project, our program always stops when reaching a max epoch size. Finding the right cost size to stop can greatly reduce the running time.

Appendix:

Working Log:

Term 1st

(https://docs.google.com/spreadsheets/d/1ZYUxaOZeW_tAefsUuUAC3FgE3xLSD5y6kN6KwcrxeCI/edit?ts=58daf93f#gid=0) Learning Materials schedules

In the first term of the project, we were given materials about deep learning, which provided us with background information and knowledge necessary for the upcoming tasks. These include online deep learning courses and videos from Youtube.

To make an approach to deep learning, the first thing we need to do is to understand how the deep network is built for predicting things we interested about. The network used for deep learning is called neural network. As we learned, deep network is based on neural network which has multiple numbers of hidden layers. From input to output layer, there are weights between each two layers. To get the final output layer, between each layer there are complex calculation done. In learning process, there are concept of overfitting, cross validation, linear regression, logistic regression, and SoftMax regression. Linear, logistic and SoftMax regression are models we may apply when constructing our model for prediction. In the upcoming tasks, we are asked to do experiment on student competence with different models to find out the relationship between the student's competence and factors including frustration, confusion, etc with different models.

First Week:

- Learned basic knowledge of Neural Network (https://www.youtube.com/watch?v=bxe2T-V8XRs Start From here)
- Study deep learning and Neural Network from Udacity
- Learned Overfitting, cross validation, linear regression, logistic regression, SoftMax (https://www.youtube.com/watch?v=VZuKBKd4ck4 Start at Overfitting)
- Read the paper "Modeling student competence: a deep learning approach" and try to have some questions about the theories in the paper：

Second Week:

- Have Udacity deep learning assignment 1 finished which is the nonMNIST from Udacity.
- Watch videos on chapter 6: Autoencoder (Hugo (LaRochelle))
- Reviewed previous videos in the First Week
- Further questions about paper "Modeling student competence: a deep learning approach": questions about pictures in the paper.
  How does LSTM increase the performance?
- Suggested use TensorFlow to develop the neural network and further experiments and we decided to use TensorFlow.

Third Week:

- Went through videos of lecture 7: Dropout, Motivation
- Train basic logistic regression model and the link of data is built and listed below in seventh week.
    - Through the data we collected, we found that as the number of sample increases, the accuracy rises gradually, when the sample numbers reaches a high level, the rate of increase became less obvious.
- Try use autoencoder to reduce the complexity of data and field because we used too much time on setting TensorFlow.
- Reread the paper and have the questions about Long term short term memory.
- Redo all the parts above using different sample numbers

Fourth Week:

- Try training with SoftMax Regression model
- Test using model above with different sample numbers
    - From the collected data, the relationship between sample numbers and accuracy is proportional
- Try adding autoencoder to change the complexity of MNIST data
    - But failed
- Test to find the result under new condition to see the effect of autoencoder on dataset
- Reset the number of sample to see the impact on accuracy

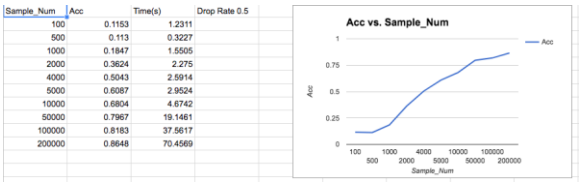
Fifth Week:

- Misunderstood the Neural Network and go through the TensorFlow tutorial
    - We actually built 2 layers of convolutional neural network
- Moved the training module from CPU version of TensorFlow to GPU Version
    - Failed because of out of memory of graphic card
- Trained the MNIST by convolutional neural network with CPU mode
    - With testing using 1,2, and 3 hidden layers, we collected the data and found that:
        - Compared to the result of using 1 hidden layer, the accuracy of using 2 hidden layers has a significant increase
        - Compared to the one using 2 hidden layers, using 3 hidden layers caused the accuracy to drop a little bit
    - Sample number is from 0 to 200000, test sample and validation are all 10000
    - Following the pattern, we found in the previous test, as the sample numbers increase, the accuracy increase in a linear regression
- The goal for next week is to build our own neural network
- Tried to get Autoencoder from Anthony but they still didn’t have the answer yet

Sixth Week:

- Train MNIST using Neural Network
    - We built the Neural Network on our own
- Implement the neural network and test the dropout rate (0.6, 0.5, 0.4)
    - We can hardly find the difference using different dropout rate
    - The sample number is from 0 to 200000
- Need to make some graphs of the accuracy of the module
- Tried to implement Autoencoder
- Fit by using different hidden layer numbers for the test on the result to how number of hidden layer affects the result

Seventh Week:



Sample_Num	Acc	Time(s)	Drop Rate 0.5
100	0.1153	1.2311	
500	0.113	0.3227	
1000	0.1847	1.5505	
2000	0.3624	2.275	
4000	0.5043	2.5914	
5000	0.6087	2.9524	
10000	0.6804	4.6742	
50000	0.7967	19.1461	
100000	0.8183	37.5617	
200000	0.8648	70.4569	

Figure 9: Part of the table built using google docs

- Build tables for test results based on MNIST
    - Logistic Regression (https://docs.google.com/spreadsheets/d/1dEDh4RQtxE4urFXJJl9W96Cm7anc7JiHaChR8PCROg0/edit#gid=0)
    - SoftMax(0 level) (https://docs.google.com/spreadsheets/d/1Nk2-aJf1EinWxK2IeLNsuPqYxeGK-eohLysyBUv5wgs/edit#gid=0)
    - Neural Network with 1 hidden layer (https://docs.google.com/spreadsheets/d/1VzkD4vr7wfSdPwGqJDKlafnyzrwg8kPuWyCbTc9BE0Q/edit#gid=0)
    - Neural Network with 2 hidden layers (https://docs.google.com/spreadsheets/d/1VzkD4vr7wfSdPwGqJDKlafnyzrwg8kPuWyCbTc9BE0Q/edit#gid=0)
    - Neural Network with 3 hidden layers (https://docs.google.com/spreadsheets/d/1FHAj1Q9ioA1tDiPWIp7-p5jkOGTlmD4u1F2aeOHuv8s/edit#gid=0)
- Includes: Using Logistic Regression, applying 0,1,2,3 hidden layers in testing with different dropout rate, here is our conclusion:
    - As sample numbers becomes larger, the accuracy gets higher
    - When the number of hidden layer is 2, the accuracy of outcomes is the highest
    - Dropout rate has tiny influence on the accuracy result
- Finish the progress report for later paper and convenience
- Finished the Autoencoder and test it
- Be prepared for the next term's task
- Extra stuffs in summer

Eighth Week:
try to do this approach by education data
Before number
Now text

Week 5.20-5.26

- Test with different factors in input size, numbers of hidden units, CPU version and GPU version
- Build tables of the result including: time elapse of each epoch, total time on finishing the process, and average time elapse of every epoch

Term 2nd:

Week 1 8/28/2017-9/1/2017

- ●Decide the project direction,
- ●Separate the features of student at 12 years old, train them respectively
- ●Need the data to train for competition
- ●Download the code, modify and train

Week 2 9/1/2017-9/8/2017

- ●Code work
- ●Predict one outcome
- ●Build 4 separate models

Week3: 9/8-9/15

- ●Let the code run successfully
- ●Turn the data given into 3D-array
- ●Need to know the specific features results have
- ●The data structure
- ●array[student] [time step] [feature number] (just for make sure)

Week4: 9/15-9/22

●Built the model for training data

●Try several epics of running with autoencoder

●Failed on training 10 percent of data when input number is 10, 1000, 10000

Week5: 9/22-9/29

●Meet with Anthony, solving bugs

●The code can run at least one batch, but had some problem with offsets

●We decided to delete the one have problem in the dataset when building the model.

Week7: 10/6-10/13

●Experiment the data with different factor including: AUC, hidden layer, output type

●Collect the data and calculate the time cost and average epoch time

Term 3rd:

Week one 10/27-11/3:

- More layers perform worse

Data: ExperimentData.csv

Fri 11/10 2017

- We can apply some change to AUC (e.g. 0.6 vs. 0.7)
- RNN VS LSTM
- ADAGRAD VS ADAM
- Use stepsize as a factor
- Test with alpha {0.0.1, 10^ (-6)}, and keep rate
- Turn off the threshold for Anthony's code

Data: ExperimentData.csv
adam_vs_adagrad.csv
Step_Size.csv

Fri 11/17 2017

- Before last week we kept using ADAGRAD
- Predict each label individually and compare with the Theano Framework (Tensor Flow vs. Theano).
    - 1. Individual models: wheel spin, retention, first action
        - Harder: affects: boredom, frustration, confusion, concentration.
    - 1.5. Run npc on Xiaolu dataset. (The difference on different data set for same model)
    - 2. All above at once (8): multiply all the model at once, probably 20 times longer
- No tune for the parameters and keep going.

Data: adam_vs_adagrad.csv
SingleOutPut

Fri 11/24 2017

- Thanksgiving Break

Fri 12/1 2017

- Get experiment data for single output(npc,rc,fa,ws)

Fri 12/8 2017

- Working on fixing bugs on multiple output with Anthony and Seth.
- Working on One-Hot encoding (Boredom, Frustration, Confusion, Concentration)

References

Bengio, Y.; Courville, A.; Vincent, P. (2013). "Representation Learning: A Review and New Perspectives". *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Schmidhuber, J. (2015). "Deep Learning in Neural Networks: An Overview". *Neural Networks*.

Bengio, Yoshua; LeCun, Yann; Hinton, Geoffrey (2015). "Deep Learning". *Nature*.

David A. Freedman (2009). Statistical Models: Theory and Practice. Cambridge University Press.

Logistic regression. (2018, April 10). Retrieved from https://en.wikipedia.org/wiki/Logistic_regression