Grade Prediction Using Artificial Neural Networks

|  |  |  |  |
| --- | --- | --- | --- |
| Jan Carlo D. Rabacca  11106344 | Suzette Dela Cruz  11118113 | Marc Janssen Chiu  11114525 | Frank Gomendoza  11106395 |
|  |  |  |  |

***Abstract*—** ***Artificial Neural Networks is modeled after a human’s brain, the ingenuity of neural networks is now being practical because of today’s power computers and now slowly taking over the traditional machine learning approaches [1].***

# Introduction

Artificial Neural Networks or ANN is an approach wherein a machine is trained in order to accomplish tasks that sometimes, we humans normally cannot do as efficiently. Some implementations were applied to automated vehicles or even to social media. A machine can easily recognize a person’s face nowadays, and a machine with more accuracy in face recognition than human has already been developed. An ANN is being developed by google by using YouTube videos as training data wherein the machine has the capability to briefly describe a random picture in a few words.

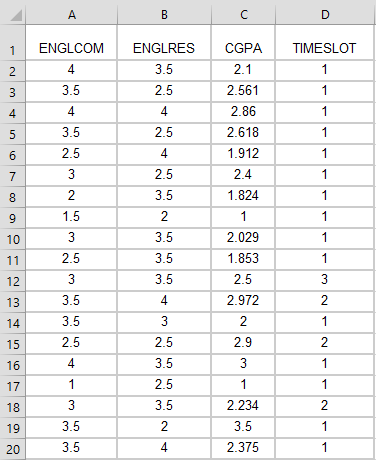
# Implementation

For the project, Artificial Neural Networks will be implemented to train a machine to predict a student’s grade thus, the group chose to base the data from the subjects ENGLRES & ENGLCOM wherein the latter is the pre-requisite subject. The inputs for the ANN are as follows (Refer to Figure 2):

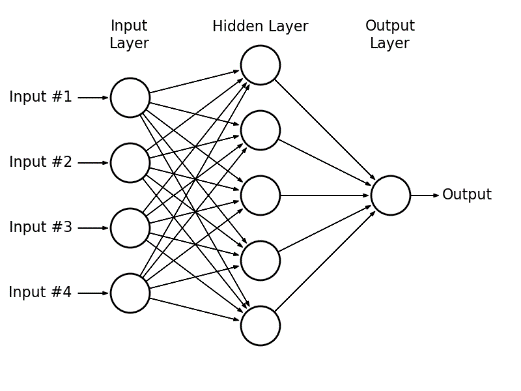
1. CGPA
2. Pre-Requisite Subject Grades
3. Timeslot

For the training data, the group utilized Google’s application of google forms wherein the group asked people to answer a brief survey wherein they are asked to provide the necessary data for the implementation of the ANN. As of date, the group managed to obtain 80 responses where in Figure 1 shows the first 20 responses.

The values for the ENGLCOM & ENGLRES grades are based from De La Salle University – Manila’s grading system where 0 incurs for a failing grade while 4 being the highest and incremented by 0.5. The CGPA is a direct value of the student’s CGPA, CGPA stands for Cumulative Grade Point Average, which is incurred within the whole stay of the student in the university. This is a necessary data to predict the grades of the users. Lastly, the timeslot were segregated into values of 1, 2 and 3 wherein 1 corresponds to a morning schedule, 2 being an afternoon schedule and 3 being an evening schedule which ranges from 6 P.M. to 9 P.M., the group determined that the schedule is a factor that might affect a student’s grade which is the reason why it was included for the necessary inputs



*Figure 1. Sample of Raw Data*

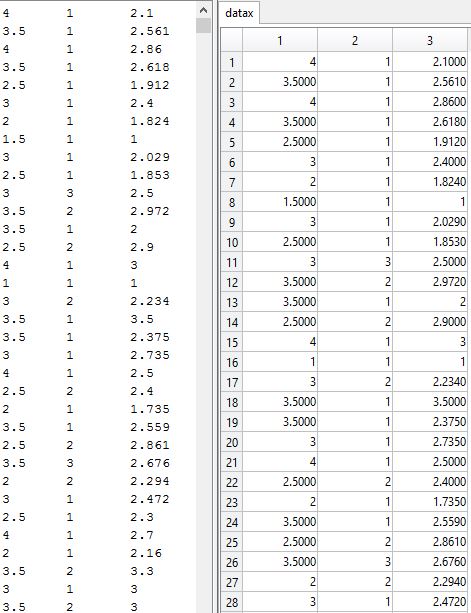


*Figure 2. Artificial Neural Network Diagram*

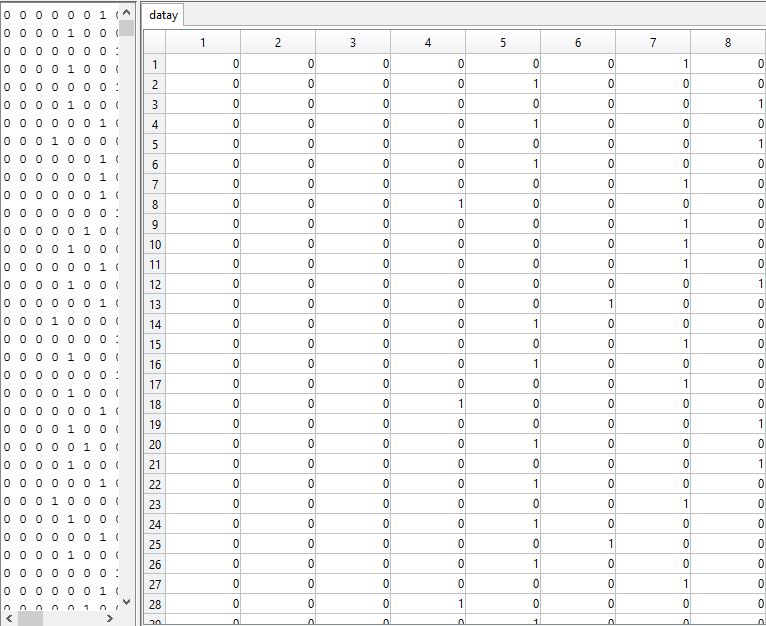
*Pre-Processing of Data*

For the pre-processing of the training data, the group took into consideration the factors that affects the student’s grade which are divided into 3 columns wherein the first column corresponds to the grade in the pre-requisite subject which is ENGLCOM, the second column refers to the timeslot and last column corresponds to the CGPA of the student.

For the target data, the group converted the decimal form of the grade into binary bit representation where for example, 0 1 0 0 0 0 0 0 corresponds to a grade of 1.0 therefore yielding 8 classes of output data.



*Figure 3. Training data*

**

*Figure 4. Target data*

[2] – The data was divided into 3 categories namely, Training, Testing and Validation. In the code, it was setup that 80 percent of the data will be allotted to be the training data, 15 percent of it will be for validation and the remaining small 5 percent will be for testing.



*Figure 5. Data Division*

*Training*

There are many available training algorithms. The compatibility of these algorithms with a set of training data depends on many different factors such as the complexity of the problem, the number of data points in the training set, the number of weights and biases in the network, the error goal and whether the network is being used for discriminant analysis or regression.

|  |  |
| --- | --- |
| **Algorithm** | **MATLAB function** |
| Levenberg-Marquardt | trainlm |
| BFGS Quasi-Newton | trainbfg |
| Resilient Backpropagation | trainrp |
| Scaled Conjugate Gradient | trainscg |
| Conjugate Gradient with Powell/Beale Restarts | traincgb |
| Fletcher-Powell Conjugate Gradient | traincgf |
| Polak-Ribiére Conjugate Gradient | traincgp |
| One Step Secant | trainoss |

*Table 1. Train Functions of MATLAB*

The ***trainlm*** function follows the Levenberg-Marquardt algorithm. The Levenberg-Marquardt algorithm was designed to deal with second-order training speed without the need to obtain the Hessian matrix. This is the fastest method for training medium-sized feedforward neural networks.

The ***trainbfg*** function follows the BFGS quasi-Newton method. This method is an alternative conjugate gradient method for fast optimization.

The ***trainrp*** functions follows the resilient backpropagation algorithm or Rprop. This algorithm eliminates harmful effects from the magnitudes of the partial derivatives such as the so called “squashing” functions or sigmoid transfer functions in the hidden layer.

The ***trainscg*** function follows the scaled conjugate gradient method. The scaled conjugate gradient algorithm is based on conjugate directions, but does not perform a line search at each iteration.

The ***traincgb*** function follows the conjugate gradient backpropagation with Powell-Beale restarts. All conjugate gradient algorithms require the search direction to be periodically reset to the negative of the gradient. There are different reset methods available for the improvement of the efficiency of training. One method was proposed by Powell, based on an earlier version proposed by Beale. This technique restarts if there is very little orthogonality left between the current gradient and the previous gradient.

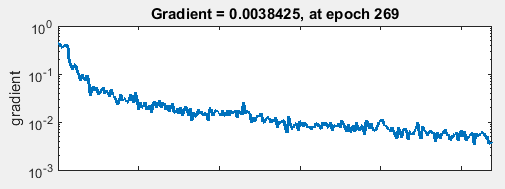
The ***traincgf*** function follows the conjugate gradient backpropagation with Fletcher-Reeves updates. Conjugate gradient algorithms are usually faster than variable learning rate backpropagation algorithms and require only slightly more storage compared to simpler algorithms, so these algorithms are compatible with networks that have a large number of weights.

The ***traincgp***function follows the conjugate gradient backpropagation with Polak-Ribiere updates. This version of the conjugate gradient algorithm was proposed by Polak and Ribiere. The performance of the algorithm is similar to Fletcher-Reeves’ version. The storage requirements for this algorithm are slightly larger than Fletcher-Reeves’.

The ***trainoss*** function follows the ones-step secant method. This algorithm does not store the complete Hessian matrix, and assumes that at each iteration, the previous Hessian was the identity matrix. This means that the new search direction can be calculated without computing for the matrix inverse.

The ***traingdx*** function follows the gradient descent momentum and an adaptive learning rate. This function combines adaptive learning rate with momentum training.

*Cost Function*



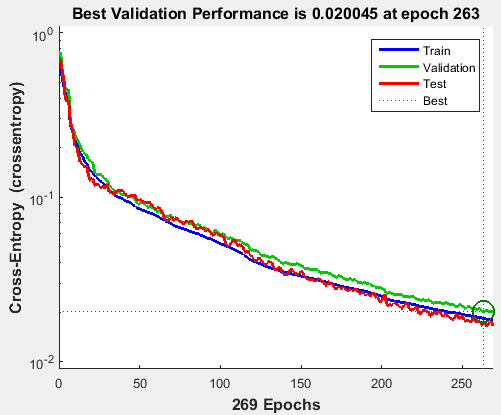
*Figure 5. Cost Function Plot*

*Performance: Cross-Entropy Error vs. Mean Squared Error*

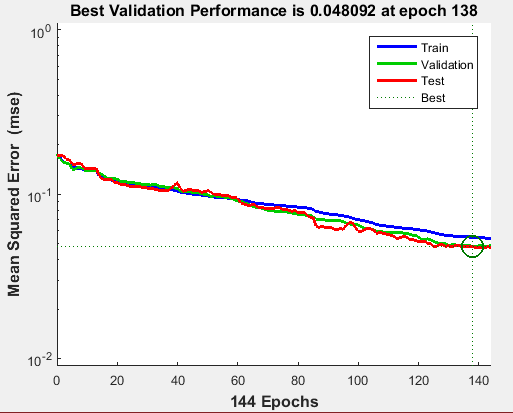
[3] – In performing classification in prediction via neural network, using cross-entropy error usually garners better results as compared to mean square error or MSE. This is mainly due to the fact that the neural network only deals with classification of the data. In this case, predicting a student’s grade from independent data such as their grade in the pre-requisite course, their CGPA and the timeslot of the class. This neural network does not deal with regression wherein the value to be predicted is numeric or any other type of neural network.

MSE is not totally a bad approach, but compared to average cross-entropy or ACE, it can be observed that MSE focuses too much on the incorrect outputs.

Based on Figure 6 and Figure 7, it shows that the performance using cross entropy is better than using mean squared error because MSE affects the computation of the gradient which results in a reduced performance.

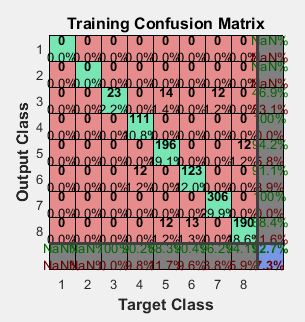


*Figure 6. Cross-Entropy Plot*

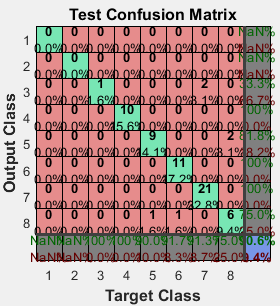


*Figure 7. Mean Square Error Plot*

*Confusion Matrix*



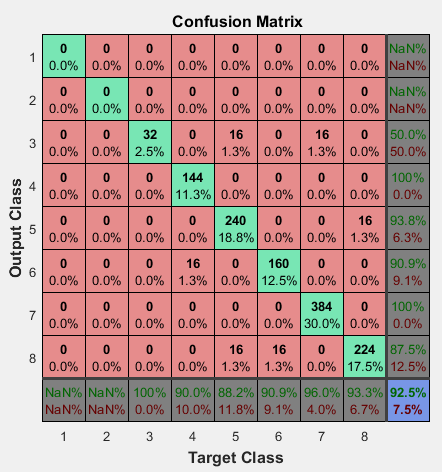
*Figure 8. Confusion Matrix of Training Data*



*Figure 9. Confusion Matrix of Testing Data*

Based from the confusion matrix of the training and testing data, the accuracy of both were above the 80% requirement wherein the training data achieved 92.7% accuracy while the testing data achieved 90.6% accuracy.

*Overall Accuracy*

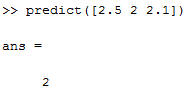


*Figure 10. Accuracy percentage*

Based from Figure 10, the simulation achieved 92.5% accuracy and successfully ran 260+ iterations before terminating it.

# Simulation Results

The group created a testing program titled predict.m, wherein if the function is called, the user would input the pre-requisite grade, the timeslot of the subject and the CGPA. An example of the simulation is shown in Figure 11 wherein the user inputs the required information, and the program outputs the predicted grade of 2.0 based on the inputs.



*Figure 11. Simulation Test*

# References

*[1] Coursera, 'Coursera - Free Online Courses From Top Universities', 2015. [Online]. Available: https://www.coursera.org/course/neuralnets. [Accessed: 24- Nov- 2015].a*

*[2] Mathworks.com, 'Choose a Multilayer Neural Network Training Function - MATLAB & Simulink', 2015. [Online]. Available: http://www.mathworks.com/help/nnet/ug/choose-a-multilayer-neural-network-training-function.html. [Accessed: 03- Dec- 2015].*

*[3] James D. McCaffrey, 'Why You Should Use Cross-Entropy Error Instead Of Classification Error Or Mean Squared Error For Neural Network Classifier Training', 2013. [Online]. Available: https://jamesmccaffrey.wordpress.com/2013/11/05/why-you-should-use-cross-entropy-error-instead-of-classification-error-or-mean-squared-error-for-neural-network-classifier-training/. [Accessed: 03- Dec- 2015].*