AI Final Project: ANN Handwriting Digit Recognition

Hector Zhang – 100947935

Problem Domain

In the field of optical character recognition, Handwriting Digit Recognition is most popular problem to the area of machine learning in AI. For this project, the goal is to complete electronic conversion of images, hand written or printed text by over thousands unique authors, into machine-encoded text, whether from a scanned document, a photo of a document or from subtitle text superimposed on an image.

Depended on the scope of the field, it will be only considered a subset of characters, digits 0 to 9 in this project. This narrowed down domain can be applied to the identification of US postal codes, bank check identification processing, and any task that follows numeric data processing. In order to get formally state the problem domain, AI project will be a handwriting recognition program specifically targeting digits zero to nine, on top of a machine learning algorithm accomplished by artificial neural networks.

Motivation

During the semester, we've learned many techniques to employ artificial intelligent algorithms to search for finite existent (or non-existent) goals within the domain of puzzles and two player games. These types of exercise build rules around the domain and programs that solve a particular narrow task with a high rate of success. I would like to explore something new AI technique which is outside of the realms of definite rule-based game systems and deploy an artificial agent that learn in a way more like the biological, ourselves.

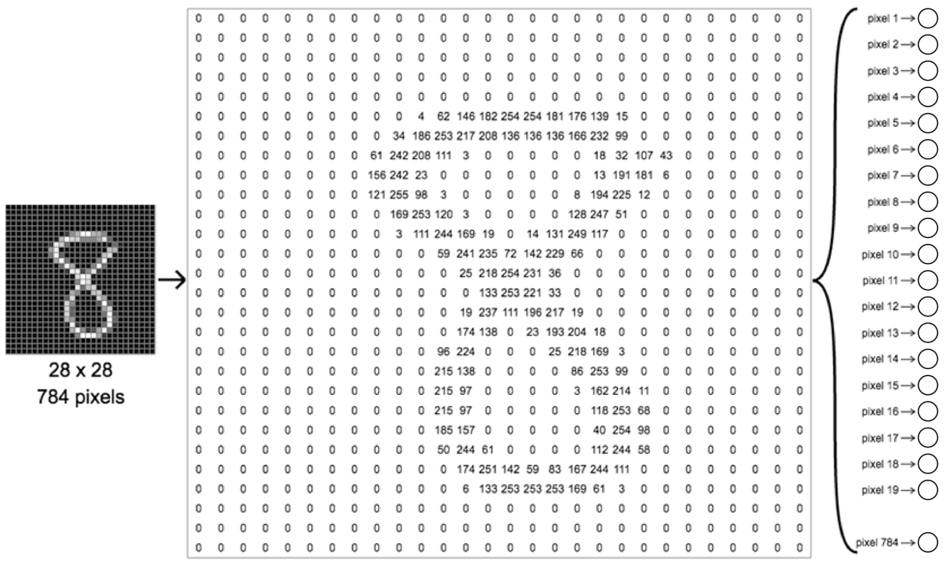
The main motivation is to create a project that can solve my problem without being explicitly programmed. I've had an enjoyable experience with heuristic-driven intelligence who can be only as smart as the programmer themselves. It seems to take up this project to challenge myself in creating an intelligent agent that tunes itself based on experience. This type of agent is known to simplify my problem domain when compared to the performance of ordinary, rule-based programming. I would like to leverage my interest and existing experience in the subject to pursue this topic independently (until the professor teaches it) to learn and experience machine learning and artificial neural networks first hand.

Techniques

Artificial Neural Network is computational model based on the structure and functions of biological neural networks. In this section, it will describe how import the dataset from MNIST, as well as the structure of Feedforward Neural Network and the Backpropagation Algorithm. As mentioned above, this project will consist of building an artificial neural network to learn inputs based on experience. It will also explain the training and testing the data.

Image Pre-processing

The program I implement will mainly focus on identifying $0-9$ from segmented pictures of handwritten digits. In my implementation, the input is imported from MNIST, which is a gray level image, the intensity level of varies from 0 to 255. For each image in MNIST is already normalized to 28×28. The MINST data set contains 60000 training samples and 10000 test samples.



```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 4 62 146 182 254 254 181 176 139 15 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 34 186 253 217 208 136 136 136 166 232 99 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 61 242 208 111 3 0 0 0 0 0 18 32 107 43 0 0 0 0
0 0 0 0 0 0 0 0 0 0 156 242 23 0 0 0 0 0 0 0 13 191 181 6 0 0 0 0
0 0 0 0 0 0 0 0 0 0 121 255 98 3 0 0 0 0 0 8 194 225 12 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 169 253 120 3 0 0 0 0 128 247 51 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 3 111 244 169 19 0 14 131 249 117 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 59 241 235 72 142 229 66 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 218 254 231 36 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 133 253 221 33 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 19 237 111 196 217 19 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 174 138 0 23 193 204 18 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 96 224 0 0 0 25 218 169 3 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 215 138 0 0 0 0 86 253 99 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 215 97 0 0 0 0 3 162 214 11 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 215 97 0 0 0 0 0 118 253 68 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 185 157 0 0 0 0 0 40 254 98 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 50 244 61 0 0 0 0 112 244 58 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 174 251 142 59 83 167 244 111 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 133 253 253 253 169 61 3 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 1 Converting Image to Input Units

The program will convert image into matrix and each pixel in the image is represent as the one input nodes which is showing in *Figure 1*. My neural network will consist of an estimated about 784 input units. In convenience. The program is also implement two options for the image pre-processing: decrease the image from 28×28 to 20×20 and scale the canvas of the image to half. The purpose is giving the options to modify the input units.

Feed-forward Neural Network

Feed forward neural networks were the first type of artificial neural network inverted. They are called feedforward because information only travels forward in the network, first through the input nodes, then through the hidden nodes, finally through the output nodes as showing in *Figure 2*

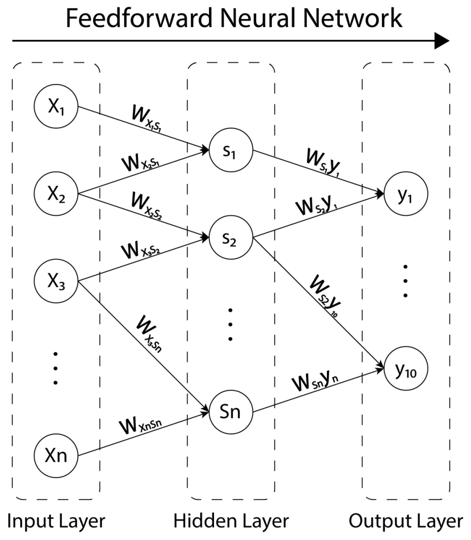


Figure 2 Feedforward Neural Network

Feedforward Neural Networks are primarily used for supervised learning in cases where the data to be learned is neither sequential nor time-dependent. That is, feedforward neural networks compute a function f on fixed size input x such that $f(x) \approx y$ for training pairs (x, y). On the other hand, recurrent neural networks learn sequential data, computing g on variable length input $X_k = \{ x_1, \dots, x_k\}$ such that $g(X_k) \approx y_k$ for training pairs (X_n, Y_n) for all $1 \leq k \leq n$.

Backpropagation

The backpropagation algorithm learns the sample data that I provide. In back propagation has two phases in which is used to learn.

- The first phase propagates the inputs from the first layer (picture pixels) forward into the hidden layer(s) which will be derived for error and will trigger an activation function if the data pass the test (a current setting of weights and bias).
- The second stage will adjust the weights and bias based on error and output its findings into the output layer.

In backpropagation, the parameters of primary interest are $w_{X_iS_j}$, the weight between node X_i in input layer and node S_j in hidden layer and b_{jk}, the bias for node k in output layer in *Figure 3*. There are no connections between nodes in the same layer and layers are fully connected.

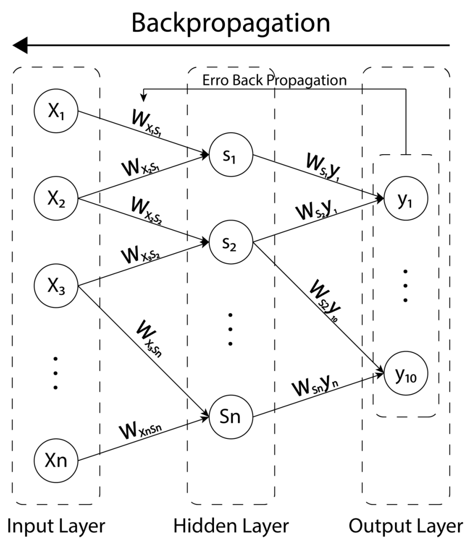


Figure 3 Backpropagation

The error function in backpropagation is the mean squared error

$$E(x,\theta) = \frac{1}{2N}\sum_{i=1}^{N}(\widehat{y_i} - y_i)^2$$

Where y_i is the target value for input-output pair $(\vec{x_i}, y_i)$ and y_i is the computed output of the network on input of the network on input $\vec{x_i}$.

Learning Progress

The output of my program will be the corresponding 0-9 digit contained in input image. The method I use is Artificial Neural Network (ANN). ANN will implicitly learn the corresponding rule between image of handwritten digits and the actual 0-9 identities. To achieve the effect of dimensionality reduction, I make use of multilayer network. The input layer contains the same number of units as the number of pixels in in- put image. In our case it is 28x28=784. Then the hidden layer containing 100 units with sigmoid activation is employed to find a compact representation of input images. Hopefully, this compact representation is easier for the final classification purpose. In the end, the output unit contains 10 units in accordance with 10 different classes. Preferably, I want the output units provide the conditional probability (thus the output of each unit is between 0 and 1, and the outputs of all 10 units will sum to 1) of each class to which each input belongs, and the unit that has the maximum output will determine the class label.

Design Choices

The usual training algorithm introduced by is error back propagation, in which user need to specify at least learning rates apart from computing gradients with respect to weights. Therefore, the program has modification options for choosing learning rate and convergence property.

Sigmoid Function

$$\varphi(z) = \frac{1}{1 + e^{z}}$$

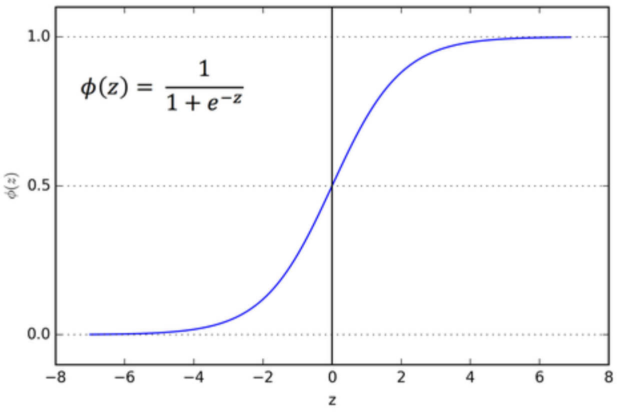


Figure 4 Sigmoid Function

This is known as Logistic Activation Function in *Figure 4*. It takes a real valued number and squashes it into a range between 0 and 1. It is also used in the output layer where our end goal is to predict probability. It convenes large negative numbers to 0 and large positive numbers to 1.

With the help of Sigmoid function, the problem reduces to computing the gradient of the objective function with respect to weights. It is obvious to analyze the result in probability.

Results

The crucial part of this project is training Neural Network. In order to determine the result of the learning, it will test the data using the trained model. Since the goal is typically building Artificial Neural Network.

Training Data

During each training epoch, the 60000 training samples are evenly divided into 100 batches. Weights are updated after processing each batch, resulting in 100 weights updates in every training epoch. In my implementation, I set a fixed number of training epochs–3. Optimal weights are chosen based on best classification performance on training set. At the end of training process, the optimal weights for the neural network will be kept and used to generate class labels for new inputs (test set).

Testing Data

Using the trained model, Artificial Neural Network will test all data and the error of output of prediction is in a proper range as following:

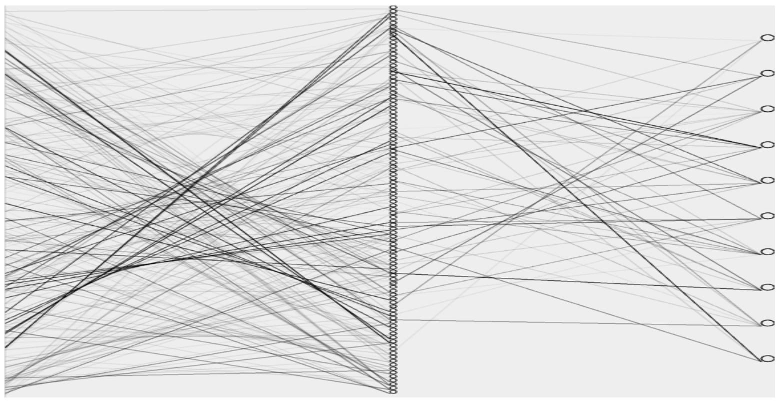


Figure 5 Experiment of Trained Neural network

The setting in *Figure* 5 is simple with the 60000-training size and training epochs of 3. For the input layer, it is consisting 784 input nodes, 100 hidden nodes in the hidden layer and the output layer with 10 output nodes which represent the digit from 0 to 9. This is one of simple experiment of the Trained Model in Artificial Neural Network. The following *Figure 6* is the average error for Error correction Learning. It is testing 10000-testing size and running the testing with the epochs of 18.

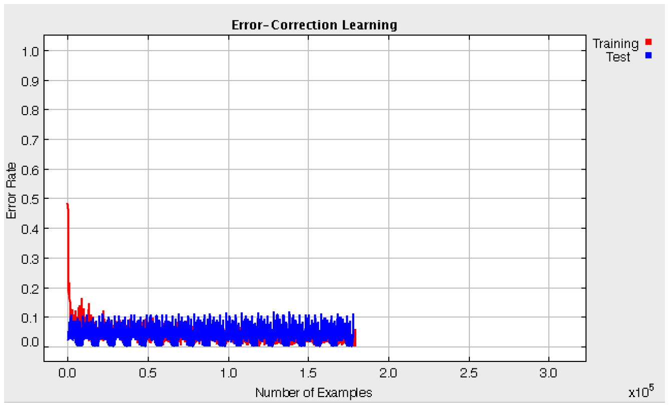


Figure 6 Error Correction Learning

As the result, the accuracy of prediction is about 90%. We can see that, the error quickly decreases on training set, and becomes almost 0.1 after 3 epochs. The error on test set first decreases with that of training set, then fluctuates a little. The smallest error rate 0.05 on test set is achieved at epoch 18.

Enhancement

In conclusion, I implement a simple Artificial Neural Network for human handwritten digits. There still exits something I could improve Neutral Network. In artificial neural network, the Gradient descent with backpropagation is not guaranteed to find the global minimum of the error function, but only a local minimum. And the solution for that is using Stochastic gradient Descent which avoid local minimum in the cost function. It adjusted weights in every batch raw of data.

Another one is about the input vectors for Backpropagation learning which is not required. However, normalization could improve the performance of the learning process. The accuracy of this feed forward neural network can be improved by revealing more information from the inputs. Same to the hidden layers. The number of hidden layers and nodes will be increase the accuracy.

Furthermore, this is first time observed the practical problems using the powerful Artificial Neural Networks, such that designing the architectures of an ANN, choosing appropriate activation functions for each layer, error backpropagation and so on. This experience will definitely be helpful for future research.

References

Aditya Sharma (2017 October 30). *Understanding Activation Functions in Deep Learning*. Retrieve from https://www.learnopencv.com/understanding-activation-functions-in-deep-learning/

Andreas Refsgaard, Francis Tseng & Gene Kogan. Neural networks. Retrieve from https://ml4a.github.io/ml4a/es/neural_networks/

John McGonagle, Kaleab Belete et al. *Artificial Neural Network*. Retrieve from https://brilliant.org/wiki/artificial-neural-network/#training-the-model

John McGonagle, *Feedforward Neural Networks.* Retrieve from https://brilliant.org/wiki/feedforward-neural-networks/

Michael Nielsen (2017 Dec). *Neural Networks and Deep Learning*. Retrieve from http://neuralnetworksanddeeplearning.com

Yann LeCun, Corinna Cortes & Christopher J.C. *THE MNIST DATABASE of handwritten digits.* Retrieve from http://yann.lecun.com/exdb/mnist/

Vikas Gupta(2017 October 9). *Understanding Feedforward Neural Networks*. Retrieve from https://www.learnopencv.com/understanding-feedforward-neural-networks/

Appendix

In order to run the code, you need to

1. Load the project in Eclipse
2. Go to GUI Package
3. Run GUI.java

Running the Artificial Neural Network:

1. Modify the number of hidden nodes in the $[h_1 \quad \cdots \quad h_n]$, where h is represent the number of nodes hidden layers n is represent the number of hidden layer for all $n \geq 1$. Default value is the number of 100 for a single hidden layer.
2. Modify the momentum if needed. Change the value to zero is turning off the momentum. The default value is 0.5.
3. Modify the learning rate if needed. The default value is 0.05.
4. Modify the initial weight if needed. The default value is $[-0.05, 0.05]$.
5. Modify the number of examples if needed. The default value is 60000.
6. Modify the number of runs if needed. The default value is 3.
7. Choose the simulation mode, either train or test. Before test, it should be trained.
8. Select or deselect the choice of decrease the canvas from 28×28 to 20×20.
9. Select or deselect the choice of scale down the image to half size.
10. Click the start button to run the training or testing. Click the stop button if needed.
11. Click the reset button to clean all the data for train and test and all the chat and error plot.

Dependencies:

1. Mnist-tools.jar
2. Plotapplet.jar
3. Plaotmlapplet.jar
4. Pxgraphapplet.jar