System and Report: Recognizing Handwritten Digits Using Neural Networks

Devin Efendy

EFENDYD

7851557

University of Manitoba, COMP3190

December 6, 2019

1. MOTIVATION

Machine learning has been used extensively to make a system that learns and classify objects in real life. For instance, autonomous car recognizing pedestrians and traffic signs. This project will help me to learn about a supervised learning technique to solve a simple classification problem.

The purpose of this project is to learn and explore the field of machine learning. The primary source of learning for this project is the Machine Learning by Stanford University, Andrew Ng. from Coursera. In this project I have learned and understood how neural network works and also, how to implement the algorithm to solve a problem. The main goal is to use ML model that uses neural network to recognize a handwritten digit from zero to nine. The model is programmed in Python using libraries for scientific computing such as NumPy.

After being able to properly implement a neural network to recognize the handwritten digit, the ML model will be run multiple times with a set of different values of: regularization parameter (λ), number of iterations for training the model, the number of hidden units, and the size of the training data set. Furthermore, I compared how the ML model behaved on two different data sets of images.

The data collected from the runs are compared and analyzed to see how choosing the data set and the latter values could affect the accuracy/performance of the model.

2. TECHNOLOGY

2.1 Programming Language

For the programming language, I used Python because it has extensive libraries for machine learning and scientific computing and also to learn and get comfortable with the language itself.

2.2 Libraries

There are some libraries that I used for this project: NumPy, SciPy, mlxtend, matplotlib, and Sklearn. These libraries are used to help with basic tasks so I can focus primarily on implementing the neural network algorithm. The tasks include as reading data sets, displaying data,

basic data structure, and matrix operations. In this section I will go through the details explaining why I use these libraries.

NumPy:

NumPy is used for the basic data structure to store the data as 2-dimensional *numpy.ndarray* which we can treat as vector and matrices. Also, NumPy has other useful linear algebra operation such as matrix transpose, multiplication, and summation.

Scipy:

To read data sets of 20x20 pixels images of handwritten digits from course website and to optimize/minimize weight matrices (theta) and cost function.

mlxtend:

To read data sets of 28x28 pixels images of handwritten digits from MNIST database. This is required because the 20x20 and 28x28 images are in different file format.

matplotlib:

To display the images of handwritten digits from both data sets (20x20 pixel and 28x28 pixel).

Sklearn:

This library is used to do feature scaling on training data sets.

3. DATA SETS

3.1 MNIST Database of Handwritten Digits

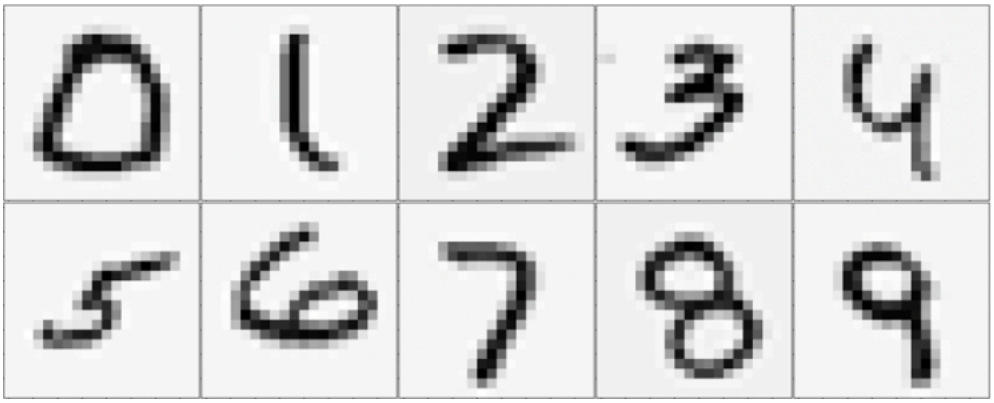
The database consists of two data sets, the first one is the training set which consist of 60,000 examples of handwritten digits, and the second one is the test set of 10,000 examples. Both the training and test set have their own images and the corresponding label for each image, the label indicates the digit that a particular image represent. The size of each image is 28x28 pixel where each pixel represents the greyscale level and the digit is placed at the center of the image.

The 60,000 pattern training set contained example contained from approximately 250 writers and the set of writers for the test set are different (LeCun et al, 1998).

Example images from the MNIST database

3.2 Data Set from Machine Learning by Stanford University, Andrew Ng

This data set has one training set which consist of 5000 example images of handwritten digits and the corresponding label for each image. The size of each image is 20x20 pixel where each pixel represents the pixel intensity. We can see that in the figure below the pixels are more noticeable and the edges are not as smooth compared to the images from MNIST database.



Example images from the course Machine Learning by Stanford, Andrew Ng

4. IMPLEMENTATION DETAILS

In this section I will talk about the implementation details and choices for this ML model. This will be categorized into parts: choosing the number of hidden layers and hidden units. preparing and processing the data sets, implementing the neural networks, and implementing prediction. Any *course* mentioned from this point will refer to the course Machine Learning by Stanford University, Andrew Ng. from Coursera.

4.1 Processing the Data Sets

For each image in the MNIST data set has the size of 28x28 pixel which is 784 pixels in total. Each pixel represents a grey scale value from 0 to 255. This data set will be read using *mlxtend* library that will give us a matrix of size M x N as *numpy.ndarray* where M is the number of images in the set and N is the number of features for each image, where we treat individual pixel as feature. For instance, reading the training data set (train-images-idx3-ubyte) with *mlxtend* library will give us the matrix of size 60,000 x 784 since the training data set has 60,000 examples and each image consists 784 pixels. As for the labels (train-labels-idx1-ubyte) we will have a vector of size 60,000 where each row is a digit that a row in training data set represent.

The same goes for the data set from Coursera but, instead of using *mlxtend* we are going to use *loadmat* from SciPy to load .mat file. In this data set each image has the size of 20x20 pixel and therefore we will have 400 features. By using *loadmat* on the training set (ex4data1.mat) we will have a matrix of size 5,000 x 400 since the training data set has 5,000 examples and each image consists of 400 pixels.

4.2 Neural Network Structure

This ML model has 3 layers: input layer, one hidden layer, and output layer. The number of input units inside the input layer is adjusted to work with image of any size. For example, using the MNIST data set we will have 784 input nodes since we treat each pixel/feature as input and using the data set from Coursera will give us 400 input nodes.

Choosing one hidden layer is for simplicity of the network. As for the number of hidden units, first, we will follow what is given by Andrew Ng, which is using 25 hidden units since we can expect to get a good accuracy (above 90%). Later, when we already successfully implemented the neural network and making the expected prediction with the data set from the course, we will vary the number of hidden units and see how the model performs on the MNIST data set.

Lastly, the number of output units will be 10 since each value will map to the digit from 0 to 9.

4.3 Implementing Neural Network

In this section we will go through the high-level detail on implementing the neural network. This ML model will be based on the course Machine Learning by Stanford University, Andrew Ng. from Coursera. The neural network is built by implementing the mathematical representation of neural network from the week 4 and 5 of the course, as vectorized implementation. Vectorized implementation is very useful since it makes the ML model become more efficient by treating the training sets as matrix, where each row represents an example and each column represent a feature. Hence, we don't need to loop through each example from data set to apply some operation, but instead we can directly apply operation on all examples by using matrix operation on the whole data set. The implementation includes random weight initialization, cost function, sigmoid function, forward propagation, and back propagation.

Since we have 3 layers neural network then we need two matrices of weight called $\theta^{(1)}$ and $\theta^{(2)}$, such that $\theta^{(j)}$ is a matrix of weights controlling function mapping from layer j to layer j+1 (Ng, 2019). Each matrix of weights will include the bias unit. We will initialize $\theta^{(1)}$ and $\theta^{(2)}$ with random initialization using *random_weight_init()* to break the symmetry so that back propagation will work properly (Ng, 2011). Moreover, the method used for implementing the random initialization have been tested such that it can decrease training time of a model (Nguyen and Widrow, 1990, p. 26).

Now, we can start implementing the *cost_function()* which will return a cost function and the partial derivative using forward propagation and back propagation, this function will take regularization into account. But, before that we need to implement *sigmoid()* which will be used in forward propagation and *sigmoid_gradient()* which will be used in back propagation. After that, we will start with implementing the forward propagation to compute our hypothesis value for every example in training data set. After we have the hypothesis value for every example then we can compute the cost function. The next step will be to implement the back-propagation algorithm to compute the partial derivative or the gradient of the cost function that needed to train the neural network.

After implementing *cost_function()* now we can train the neural network by minimizing the cost function using Python's *optimize* library and using a function called *minimize* which will return the trained weight parameters. The reason why I used an optimizer library instead of implementing some optimizer algorithm like gradient descent is because using a function from

library is already tested thoroughly and working properly. Also, another factor is time constraint for this project.

4.4 Training and Test Data Set Sampling

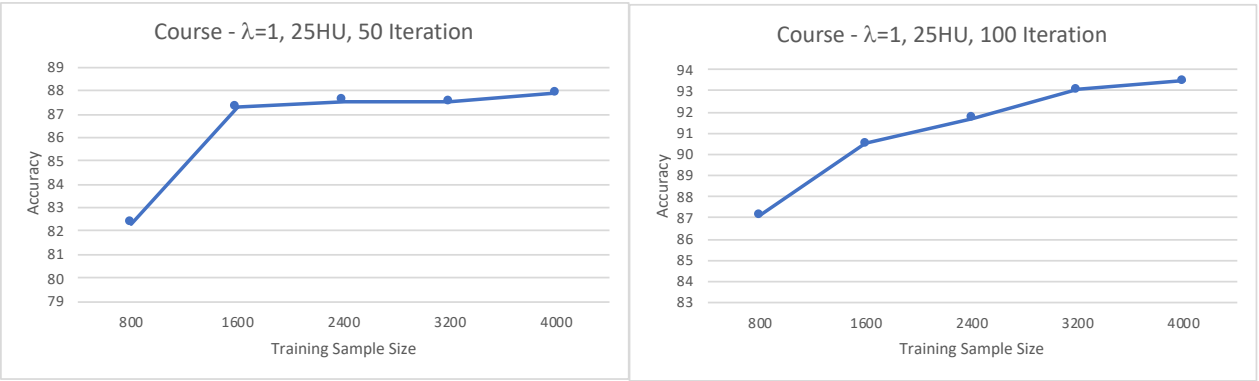
The data set that is used for training the neural network will be disjoint from the data set that is used for testing the accuracy of the ML model. This is to see how the ML model perform / predict the data/examples that it never seen before. The principle is to isolate test data via data sampling to check whether the trained model fits new cases (Zhang et al, 2019, p. 16).

For the MNIST data set, the training (train-images-idx3-ubyte and train-labels-idx1-ubyte) and testing (t10k-images-idx3-ubyte and t10k-labels-idx1-ubyte) data set is already disjoint. But, for the data set from the course we only have one data set. Therefore, we need to separate the data set to training and test set, and the program has made sure that the two sets are disjoint.

5. TESTING MODEL'S PERFORMANCE

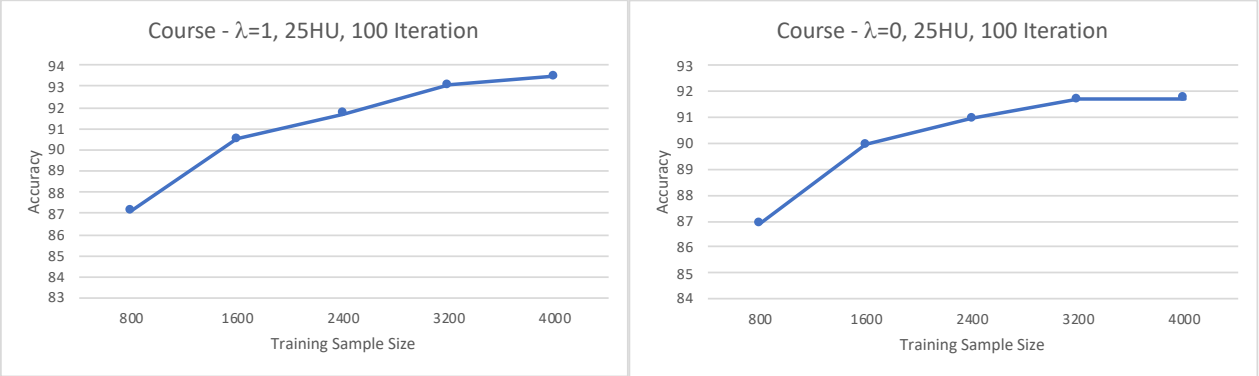
To test a ML model's performance with a particular neural network structure, we will run 10 tests for the model to make prediction against a subset of test data set and take the average accuracy. Since, the data set from the course will be shared for both training and testing then we will select a random sample for the training and testing from the data set. The training set will be 80% from the all the data set and the test set will be 20% of all data set. For MNIST data set, our training set will be 80% of all training set and the test set will be 20% of all test set.

Now, we will try to experiment with a different set of values: regularization (λ), the number of training sample, the number of training iteration (Iteration), and the number of hidden units (HU), and see how these values affect the ML model's performance.



ML models run with different number of training iteration. (left: Model A and right: Model B)

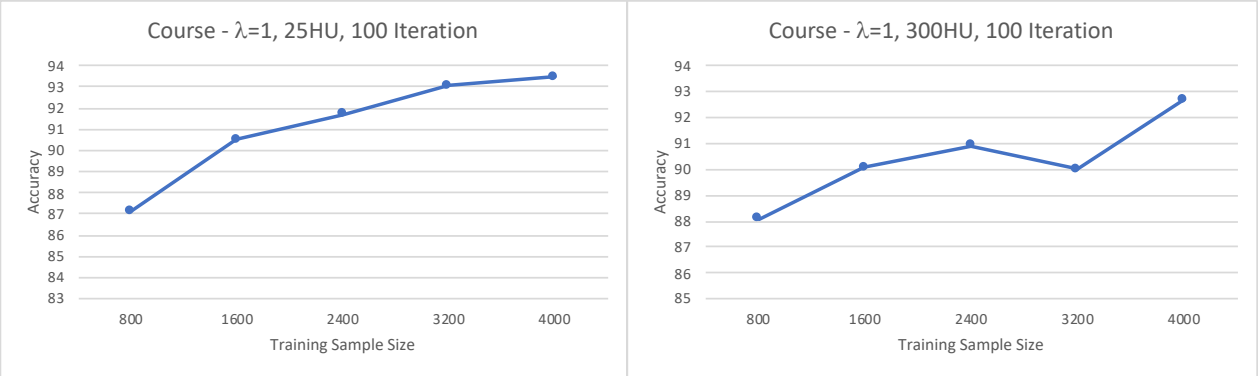
Both of these ML models are tested using the data set from the course and have the regularization parameter set to one and have 25 hidden units. Let model A is the ML model that tested using 50 training iterations and model B is the ML model that tested using 100 training iterations. From the graphs we can see that the number of training sample contributes to a better accuracy of the ML model's prediction. For instance, model B has an increase in every increment of training sample size. Furthermore, the graphs also shown that the number of training iteration also contributes to improving the accuracy of the ML model.



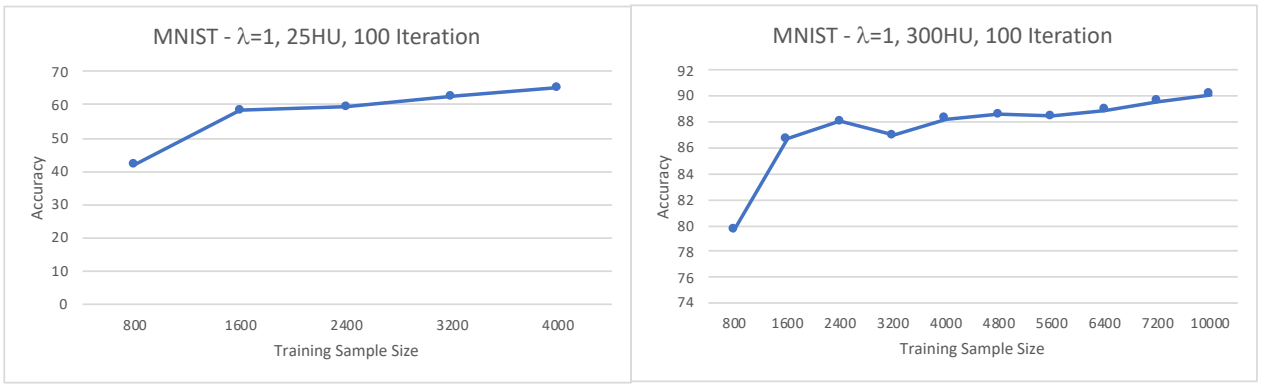
Models run with different regularization parameter (λ) (left: Model A and right: Model B)

The graphs above is showing how the same ML model but with different value of regularization parameter (λ) perform. Regularization helps to address the problem of overfitting, which is the problem of the ML model that fits well the available data but does not generalize well to predict new data (Ng, 2019). When λ is zero then the model will not use any regularization when calculating the cost function. On the other hand, when the $\lambda > 0$ then, the model will use regularization to some degree depending on how big the value of λ is.

Since the training and the test data set is disjoint, both models are tested with the data that they never seen before. Because model A is using regularization, it able to generalize well to predict the new images while model B is not. Hence, model A make a better prediction on the test data than model B.



Models run on course's data set with different number of hidden units (left: Model A_1 with 25 hidden units and right: Model A_2 with 300 hidden units)



Models run on MNIST data set with different number of hidden units (left: Model B_1 with 25 hidden units and right: Model B_2 with 300 hidden units)

Four figures above show us how the number of hidden units can affect the training of neural network. With MNIST data set, model's accuracy improves by a significant amount by increasing the number of hidden units. On the other hand, using the data set from the course we have that increasing the number of hidden units will decrease the performance of the model. To actually determine the size of the neural network we need to perform an experiment on training set by changing the number of hidden units (Panchal and Panchal, 2014, p. 463). The reason why I choose to compare the hidden units to be only 25 and 300 is because the given hidden units for the model course is 25 and

Also, for the mode B_1 that run on MNIST data set, there is a slow increase in accuracy when increasing the size for training sample. Model B_1 is finally achieving the accuracy 90% when feed with 10,000 training samples and 93.39% if feed with 60,000 training samples (this is only tested once because of time and computational power constraint). This is because the size of the

images from MNIST is bigger than the ones from the course. In which leads to more complex and larger features. Such a large number of parameters increases the capacity of the system and therefore requires a larger training data sets (LeCun, et al, 1998, p. 5).

6. CONCLUSION AND DISCUSSION

By doing this project, I have learned machine learning in high-level, knowing how to build a machine learning model that uses neural network by implementing the mathematical representation of neural network algorithm. Furthermore, I also learned about how a different set of values: regularization (λ), the number of training sample, the number of training iteration (Iteration), and the number of hidden units (HU), can affect the performance/accuracy of the model. And these values are going to be different for two sets of problems to achieve their own optimal performance and results. In this case, a good model for recognizing 20x20 pixel images of handwritten digit is going to be different than the model for 28x28 pixel one.

There are also things that could be improved. For example, to use feature scaling for the training data sets which can improve the performance (run time) of the model. Furthermore, to improve the code quality and code documentation of the model.

In my opinion, the bottleneck of this project is the machine or the computational power that is available where I used MacBook Pro (13-inch 2017) with 2.3 GHz Dual-Core Intel Core i5 Processor and 8GB RAM. Because of this, I don't have enough time to do a multiple testing with a huge set of different values. Furthermore, by having sufficient computational power, I could get more accurate data by running more tests for every set of different values. For instance, with a sufficient computational power I could try different values of hidden layer size (the number of hidden units), regularization parameter, and increasing the training sample size for MNIST data set in which I only tested up to using 10,000 training examples, do keep in mind that the test result is the average of 10 runs to get accurate results. For the details, running 10 tests which includes training the network (300 hidden units and 100 training iterations) with 10,000 examples and making the prediction will take my machine almost 3 hours (170 minutes) to run.

This project can be improved if we have sufficient computational power, so that the ML model can be tested many times with variety of sets of different values.

7. REFERENCES

Lecun, Y., Bottou, L., & Bengio, Y. and Haffner, P. (1998). *Gradient-Based Learning Applied to Document Recognition*. Proceedings of the IEEE. 86. 2278 - 2324. 10.1109/5.726791.

Nguyen, D. and Widrow, B. (1990). *Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights*. Proceedings of the International Joint Conference on Neural Networks. 3. 21 - 26 vol.3. 10.1109/IJCNN.1990.137819.

Panchal, F.S. and Panchal, M. (2014). 'International Journal of Computer Science and Mobile Computing', *Review on Methods of Selecting Number of Hidden Nodes in Artificial Neural Network*, 3(11), p455-464.

Ng, A. (2011). *Machine Learning by Stanford University* [Online]. Available at: https://www.coursera.org/learn/machine-learning/home/welcome (Accessed: 6 December 2019).