**CS5590 APS - Deep Learning Programming**

**ASSIGNMENT 1**

**Introduction**

In class, the linear regression model was utilized in TensorFlow to demonstrate the relationship between fires and theft in a Chicago neighborhood. Linear regression is not the best, most effective model to use for all cases, but this was an important first step in learning about regression in TensorFlow as many of the same ideas apply to other forms of regression modeling.

**Objectives**

The objective of this lab was to illustrate how to use logistic regression modeling in TensorFlow. This report uses the MNIST database and softmax (multinomial logistic) regression to predict what digit each image in the dataset represents.

**Approaches/Methods**

The softmax regression model is known as a multinomial or multi-class logistic regression model which is a generalization of logistic regression used for multi-class classification. This model was chosen for the MNIST dataset because of the way it handles classification and probability. Because the dataset is handwritten images of digits, we know we will only have ten possible options for classifying the data points and that we can assign probability to the options for each digit. Softmax regression returns a list of values between 0 and 1, all adding up to 1, so it is a natural choice and relatively simple as well.

**Workflow**

The first step in the process of creating this program is to import TensorFlow into the Python file and rename as ‘tf’ for ease of use throughout the code.

Then, a two-dimensional (None x 784, where none can be any length) tensor named ‘x’ must be created as a placeholder; this value will represent any number of images from MNIST when TensorFlow runs a computation.

As with linear regression, parameters for weights and biases must also be included, which TensorFlow handles as Variables, aptly named ‘W’ and ‘b’. The initial values for these variables is arbitrary as they are going to be learned through the model, so tensors full of zeros works fine here. The shapes of these tensors are carefully chosen so that they can later interact in the computation manipulations correctly.

The next step is to define and implement the model, which is where the logistic regression element comes into play. The placeholder x is multiplied by Variable W using TensorFlow’s matmul function, then Variable b is added, and finally softmax is applied.

Cross-entropy is utilized next to determine the loss of this model, which essentially analyzes the predicted answers received from the model (y) and the correct answers (y\_) and describes how far off our predictions are from the truth. The placeholder (y\_) is first created for the input of correct answers.

Once the model’s loss has been defined, the training step can occur where TensorFlow is asked to minimize this loss. Here, the gradient descent optimization algorithm is used, in part because of its simple procedure of slightly adjusting each variable in a direction to reduce loss.

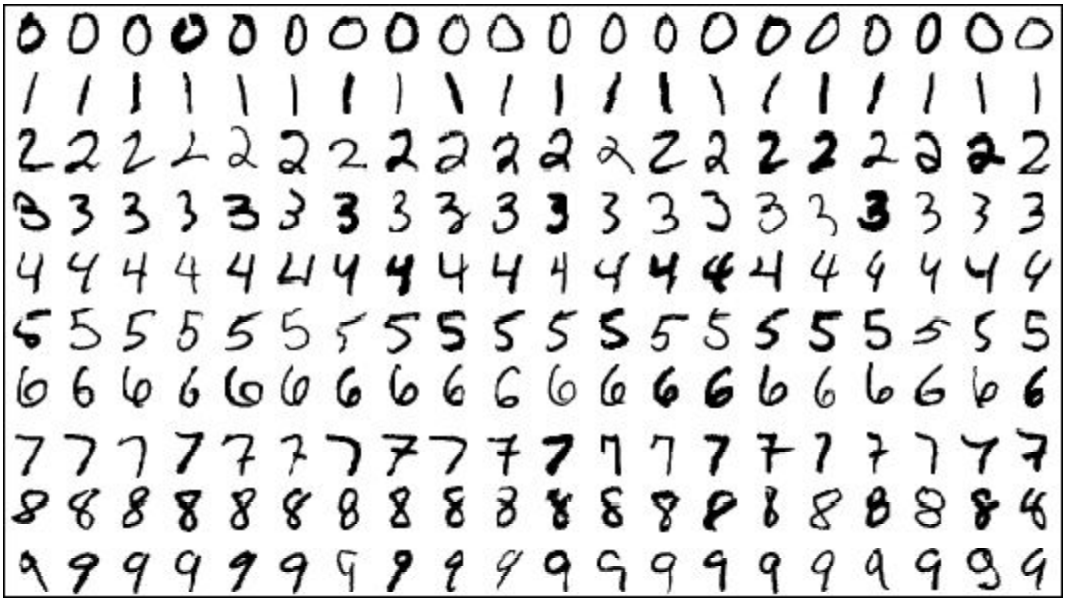
Finally, it is time to create the Interactive Session and launch the model. The first step for this is to open the InteractiveSession, then initialize the variables. The training loop is set to run 1000 times, returning groups of 100 random data points from the training set; it is inside this loop that the placeholders are finally assigned values.

The last step of this program computes the accuracy of the model by comparing the label the model came up with for each digit and the correct label.

**Datasets**

The MNIST (Mixed National Institute of Standards and Technology) dataset ([hosted on Yann LeCun’s website](http://yann.lecun.com/exdb/mnist/)) was used for this report. This dataset was chosen because it is already split into three parts for training data, test data, and validation data, it also works well for demonstrating the softmax (logistic) regression implementation, and it is a very popular database generally used for training image processing systems so there is a lot of documentation regarding the dataset available online.

Below is a snippet of what the images in the MNIST dataset look like:



Every data point in this set has two parts: 1) an image (the handwritten digit seen in the above snippet), and 2) a corresponding label.

Each image is 28 x 28 pixels which is flattened in to a one-dimensional vector with a length of 784. Each entry in this vector represents a pixel intensity between 0 and 1, where 0 is completely white, and 1 is completely black with numbers in between representing various shades of gray.

For each image, a corresponding number between 0 and 9 is used as the image label, indicating the digit drawn in the image. Here, we shape these labels into vectors of length 10, with all 0s but the index for the digit represented, which is a 1.

**Parameters**

The parameters used for this model are Variables, ‘W’ and ‘b’. Because of the modifiability of TensorFlow’s Variables, most machine learning applications use them as their parameters.

**Evaluation & Discussion**

When the final step of computing the model’s accuracy is run, a percentage of around 92 is returned. At first, that might seem like a decently high percentage, but it’s actually quite low and is a result of the use of such a simplistic model.

**Conclusion**

In conclusion, this exercise proved to be extremely helpful and educational, introducing some new ideas, and reinforcing other concepts previously learned.