Assignment #6 (Neural Networks)
John Moderwell
MSDS 422

Summary

The objective of this assignment was to use the MNIST data in Python and find a neural network that optimizes two metrics: time and accuracy. Multiple models were built and tested with various hyperparameter settings. The purpose of this assignment was to make a recommendation about machine learning technologies and optimal parameters in relation to character recognition. Specifically, students used the Python machine learning package TensorFlow to determine how different neural network methods handle classification and optimization tasks on the MNIST dataset.

Research Design

This MNIST dataset was imported using TensorFlow's built-in function found in "tensorflow.examples.tutorials.mnist". This function automatically separates the data into training and validation sets. One hot encoding was used to convert the images into a form that allows classification methods to optimize prediction performance. After ingesting the data, a model was built from code provided in Chapter 10 of Aurelien Geron's book, "Hands-On Machine Learning with Scikit-Learn & TensorFlow". Geron's book used a DNN classifying model with Gradient Descent Optimizer and ReLU activation in its layers. DNN refers to "deep neural network" and is a common estimator used in TensorFlow. Estimators are pre-implemented models that deal with the details of creating computational graphs, initializing variables and training the model. The Gradient Descent Optimizer used in this model is an algorithm that focuses on optimizing metrics specific to neural networks. For the purpose of this assignment, this algorithm attempted to minimize the amount of "loss" in the model. Loss is a calculation of how well the algorithm performs on the training set compared to the validation set.

Loss is not a percentage like accuracy, rather, a sum of the errors made in each set. For this model, loss was defined by using softmax cross entropy between logits and labels. This function measures the probability error of discrete classification tasks where classes are mutually exclusive. For example, a digit can be classified as a 7 or 8 but it cannot be both. Loss was used as metric to compare performance across different models. The two other metrics that were used to evaluate models were accuracy and time. Averages of both loss and accuracy was calculated for training and validation sets in each model. Accuracy is a metric in TensorFlow that computes the frequency in which the predictions match the labels. The Time package in Python was used to calculate wall-clock time. This is a measure of the total time it takes to execute a program in a computer. Using these metrics, subsequent models were created by adjusting the various hyperparameters that defined the model. The adjustable hyperparameters included learning rate, number of epochs, batch size, number of hidden layers and activation functions.

Here is an overview of these hyperparameters:

Learning rate refers to how much the model adjusts to input weights with respect to the loss gradient. For Gradient Descent, this rate is constant. An **epoch** in machine learning is the processing of the entire training set by the learning algorithm. Using more than one epoch is theoretically beneficial for a Gradient Descent algorithm. This is because GD takes an iterative approach to finding the minima of a curve. Therefore, the optimal result must be found multiple times. **Batch size** is the total number of training examples used in a single batch since the entire dataset cannot be passed through a network all at once. Batch size can affect computation and memory efficiencies. **Hidden layers** are layers of nodes whose output is connected to the output of other layers and are not visible in the actual output. Layers perform computations on the weighted inputs using various activation functions. This process results in a net input which is

passed to the next layer or output function. **Activation functions** are used to determine the output of a layer and map the results to a certain range depending on the function. For this assignment, the ReLU (Rectified Linear Unit) activation function was used. It is commonly used for deep neural networks and results in a range from 0 to infinity.

Programming Overview

The entire project was done in Python using the Jupyter Notebook environment. Packages that were used included pandas and numpy for data handling and manipulation. Tensorflow was used to handle the machine learning aspects of the assignment. Other packages included Time for computing program execution time.

Results and Recommendations

Based on the results, it is concluded that Model 2 was the best overall performer in terms of accuracy, loss minimization and computation time. Model 2 had the following hyperparameters: Learning Rate = .5, Epochs = 10, Batch Size = 100, 2 hidden layers and 20 nodes per layer. This model achieved an average accuracy of 99% in the training set and 95.6% in the validation set. It averaged .023 in average training loss and .143 in average validation loss. The wall time was 1 min 47 seconds. Model 8 was also highly accurate and had similar hyperparameters as Model 2 except it had 300 nodes in the first layer and 100 in the second layer instead of 20 in both. It was judged inferior to Model 2 because it took almost 4x more time to compute.

Other Conclusions

A conclusion from these results is that in general, increasing the number of nodes per layer has a positive impact on accuracy. However, as nodes per layer increase so does computation time. Another conclusion is that increasing the number of layers does not

necessarily lead to more accurate results. This may be a result of the data not being very complex, causing the algorithm to overfit the training data and not generalize to new data. It was also apparent that increasing epochs and layers while decreasing learning rate those of Model 2 has a negative effect on the prediction accuracy and also increases time.

Management Question

It can be concluded that while neural networks are promising machine learning methods for character recognition, they require lots of human-made adjustments and fine-tuning. Choosing the optimal network typology and hyperparameters will most likely be the result of performing many experiments or using prior knowledge/experience about the data structure to determine the model.

The DNN classifier and Gradient Descent optimizer are commonly used in neural networks where the objective is classification and optimization. This was the case for this assignment, and they are recommended for similar character recognition tasks.