Using Machine Learning Algorithms to Read Handwritten Digits

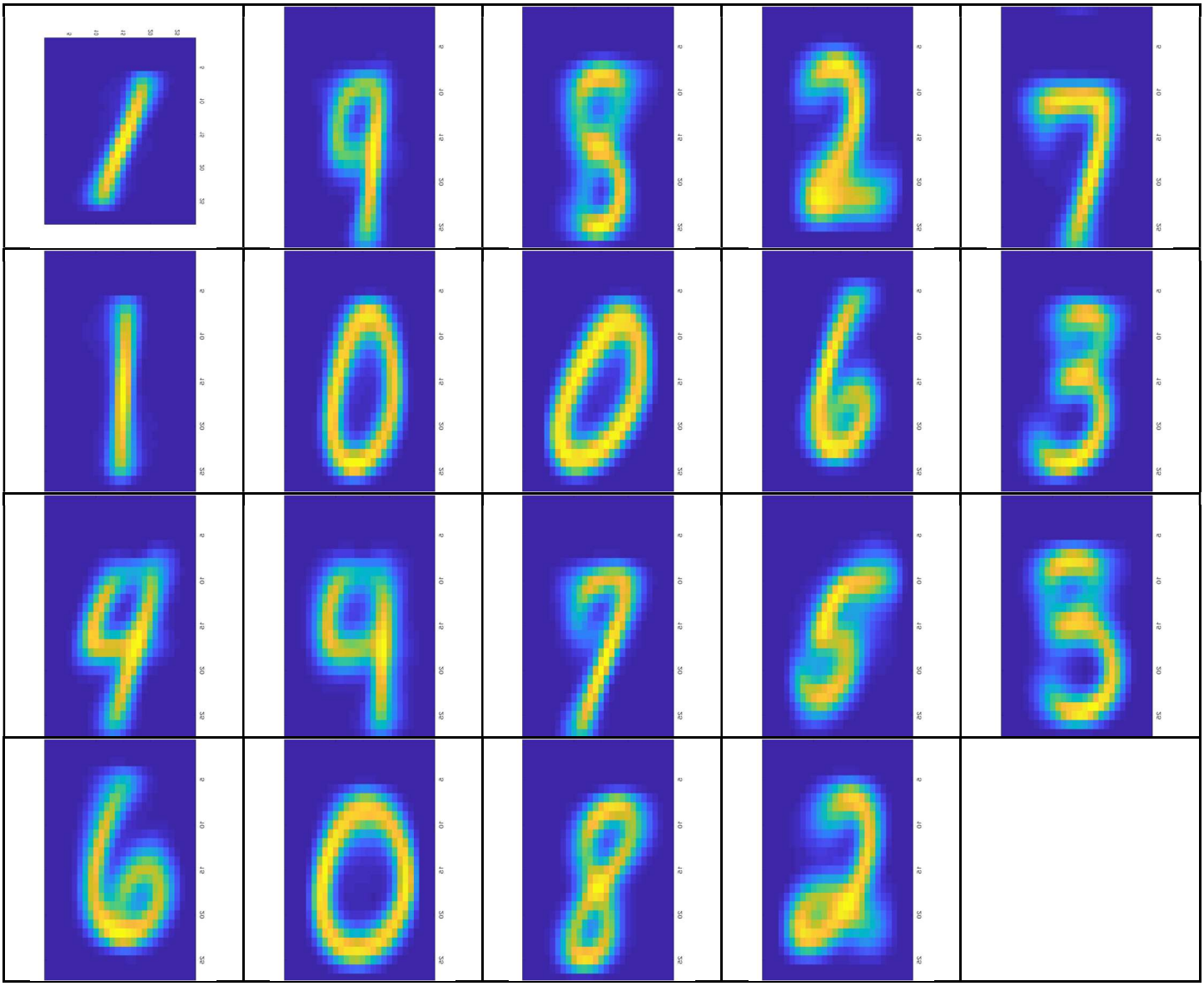
Written by Parker Bibus

Throughout this last semester, I have been learning how to use linear algebra principles and implementing them to create machine learning code that can read handwritten digits. For the handwritten digits for testing, I used the MNIST Dataset. Some of the reasons I used this dataset are all the data is formatted to a single layout, the diversity in the writing style for each number, and the extent of use by others for machine learning testing. With the dataset chosen, I had to learn the linear algebra principles. The linear algebra aspects I learned for analyzing the datasets were the k-means clustering and least squares methods. In order to learn the linear algebra principles, I used the free online book, "Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares". Let's get into the first principle I used, k-means clustering.

The idea behind the k-means clustering algorithm is to take all the data given as a set of data points, and cluster close data points together. To begin, the algorithm chooses random data points as cluster centers. The number of data points should be greater than the number of possible options. After generating these centers, we assign all the data points from our dataset to their closest center to form the clusters. Once the clusters are created, new centers are assigned by averaging all the points in the cluster. This process of finding clusters and assigning new centers is repeated until the centers stay constant meaning the best centers have been found. This entire process is often repeated multiple times as different results are obtained with different starting centers. With the algorithm implemented, I ran some trials and analyzed the results.

To run the trials, I ran my code in Matlab on my personal computer. The hardware I used was an Intel I7-6700T CPU, a Nvidia GTX 1060 6GB Graphics Card, and 16GB of RAM. With this hardware, I achieved an average speed of 190 seconds when doing the whole k-means

process on the 60,000 data point MNIST set. At the end of a run, there was an average of between 15 and 20 centers. One set of centers that I ended up with can be seen below.



As can be seen, some of the numbers have multiple copies. This is due to the threshold for how many items a cluster must have and the unique ways that people write each number. It can also be observed that similar numbers merge together, such as 3 and 8, and 4 and 9. After I finished implementing the k-means algorithm I moved onto implementing the least squares algorithm.

The idea behind the least squares method is to create a linear equation out of the data by giving each coordinate in a vector a specific weight. These weights are then used as coefficients for the linear equation which can then determine if the input is a match to the test data. These multipliers are found by manipulating the matrix containing all the data with both its transpose and inverse in the form of $(Mat^T * Mat)^{-1} * Mat^T * Correct$.In the equation, *Mat* is the matrix,Mat^T is the transpose of the matrix, the power of -1 means the inverse, and$Correct$is a vector specifying whether each column is the number we want or not. All of this comes together to give a vector that contains the multiplier for each data point to find whether the input matches individual number.

After implementing the least squares algorithm in Matlab, I ran multiple tests with different training dataset sizes. As expected, the greater the training dataset size, the more accurate the result. To get an idea of the accuracy, when using 50000 training images, the resulting multipliers have a %86 accuracy rate. The most mistaken numbers from the dataset were the numbers 1 and 9 while 5 and 2 were the most accurate. I also found that this method ran much faster than the k-means algorithm with an average of about 8 seconds to get through 50000 images. A possible way to improve these numbers is to do more error checking or hard-coding information about the dataset before training the algorithm.

Over this semester, I learned a lot about the math behind the k-means clustering and least squares machine learning algorithms. I also implemented these algorithms using Matlab and attained acceptable speeds and accuracy results. Hopefully, I can apply what I have learned in the future when learning about new machine learning algorithms and other types of machine learning.

Resources Used

"Matlab & Simulink." R2017b, MathWorks, 2017.

BOYD, STEPHEN, and LIEVEN VANDENBERGHE. INTRODUCTION TO APPLIED LINEAR ALGEBRA: Vectors, Matrices, and Least Squares. CAMBRIDGE UNIV PRESS, 2018.