MNIST Digit Classifier

Brandon Brodzinski

November 2022

Contents

1 Introduction

In this project, I implemented a deterministic classifier K-Nearest neighbor using the MNIST; handwritten digit database. The goal of this assignment is to create a written digit classifier and predict the correct number based on a given image. KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label in the case of classification.

2 Implementation of KNN

In this section, I will discuss the process of how I implemented this algorithm. The data set supplied already has a split of training set of 60,000 images and a testing set of 10,000 images. To being with, wrote a function to calculate the Euclidean distance; which is the length of line segment between two points represented by the formula d = $\sqrt{(x2–x1)^2 + (y2–y1)^2}$. Following, I made a function to compute the KNN by passing it a query point, training data, and a specified K value. It's important to choose an odd K value so that ties within classification are avoided. After the distances are computed within the function it then calculates which digit it is most like by choosing the class that is the max number and uses that as the prediction. For example, if the K value is 5 and 2 labels are the digit 0, and 3 labels are the digit 8, then 8 would be the prediction since 3 is greater than 2.

2.1 Decision Points

After running my first test, I realized that with a large chunk of data this algorithm performance is hindered. This led me to vectorizing as many things as I could to speed up the process. This was accomplished by avoiding as many for loops as I could and using many built in functions in numpy so that my function could be optimized to the best of its ability. Another key factor I realized was, I was getting an accuracy of 30%

or lower on every run I did. But after augmented research I realized that 0 values weren't being avoided as inputs, which are capable of preventing weight updates. Subsequently, I corrected this by multiplying each pixel by 0.99 / 255 and adding 0.01 to the result of the training/testing set. This amended my accuracy tremendously and increased it to 90% and above on most runs depending on how many training images were utilized.

3 Evaluation

In this section, I discuss the results. In Figure 1, I used a fixed number of 200 test samples and adjusted the number of training samples on each execution. I concluded, the increase in training samples that are used the higher the accuracy you will receive. In Figure 2, I used a fixed number of 500 training samples and altered the number of testing samples on each iteration. I analyzed, as the testing samples increased the lower the accuracy became. A common factor in both graphs is that after 2000 samples the accuracy plateaued respectively. Moving forward in Figure 3, I reported the results in a normalized confusion matrix to show how each digit was either classified correctly or incorrectly. Lastly, in Figure 4 I calculated the likelihood of confusion in a histogram to show the percentage error of each digit that was misclassified.

3.1 Accuracy Graphs

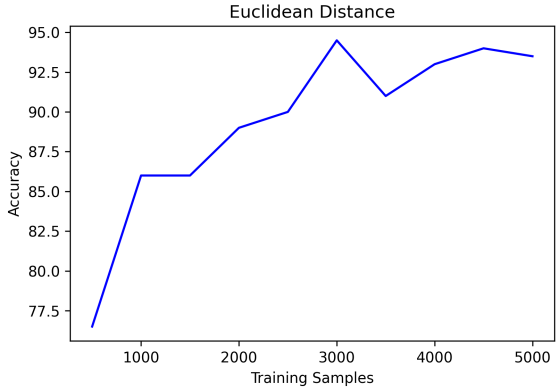

Figure 1: Accuracy as a function of number of training samples, with a fixed number of 200 test samples.

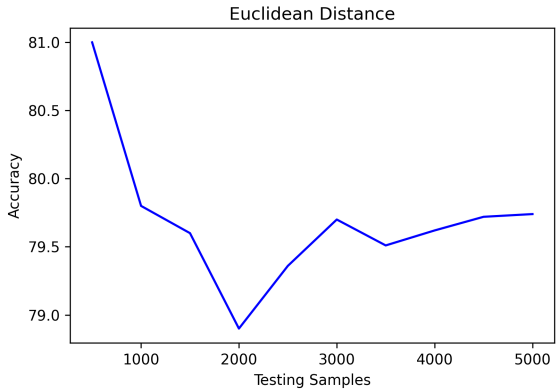

Figure 2: Accuracy as a function of number of testing samples, with a fixed number of 500 training samples.

3.2 Confusion Matrix

Training = 5000 Images, Testing = 500 Images, Accuracy = 94.19%

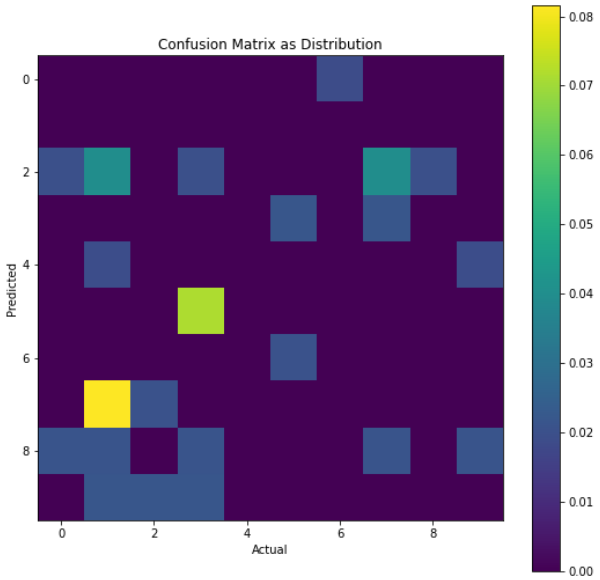

Figure 3: Confusion matrix without main diagonal.

3.3 Likelihood of Confusion

Training = 5000 Images, Testing = 500 Images, Accuracy = 94.19%

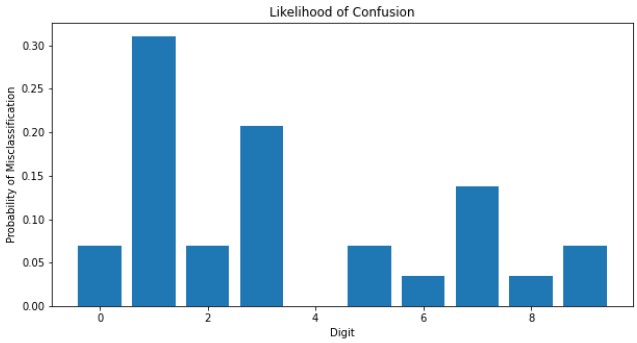

Figure 4: Per digit confusion.

4 What I've learned

In this project, I became more knowledgeable with the certitude that vectorization is extremely important when it comes to implementing demanding algorithms with large data. Additionally, I discerned the pros and cons of the KNN algorithm. The pros being, no training period is required, it can be used for classification or regression, and a variety of distance criteria can be used. The cons are, decelerated performance in large datsets, does not adequately perform with high dimensions, and is sensitive to noisy data. Briefly, this algorithm solves classification problems easily as there are only two parameters, a K value and a distance.

5 Extra Credit

Training = 5000 Images, Testing = 500 Images, Average Accuracy = 94.08%

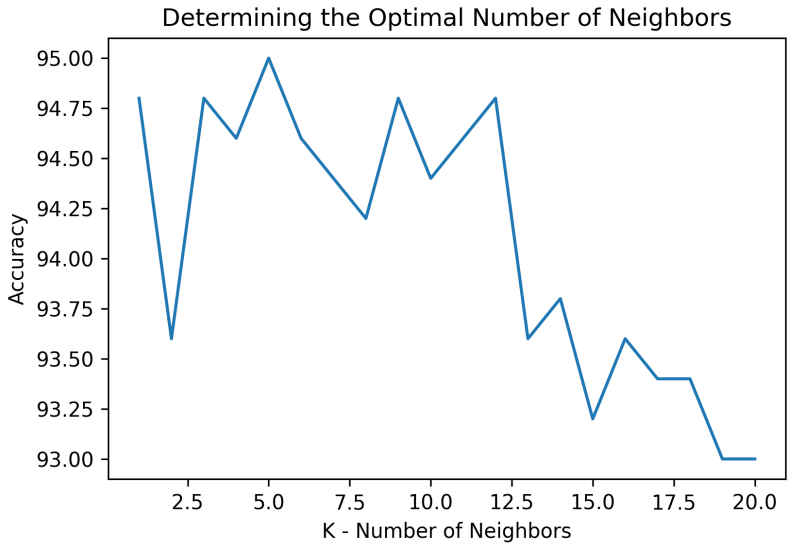

Figure 5: Overall accuracy with respect to K.