Optical Character Recognition of Handwritten Digits

Oscar Camargo

April 2023

Contents

List of Figures

1 Introduction

Optical character recognition is the process of converting image text into machine-readable text format. This process often involves the use of different algorithms discussed in this course including k-nearest neighbors, support vector machines, and convolutional neural networks. This project will focus on the use of k-nearest neighbors and k-means clustering for the recognition of handwritten digits. For this, different pre-processing and visualization (e.g., t-distributed stochastic neighbor embedding) algorithms will also be used.

2 Pre-processing the MNIST data-set

The Modified National Institute of Standards and Technology (MNIST) database is a database of handwritten digits commonly used for the training and testing of classification algorithms. The MNIST database consists of two different image data-sets—a training data-set with 60,000 images and a testing data-set with 10,000 images—each image data-set has a corresponding label data-set that contains the label of the image (i.e., the digit which is depicted in the image). The width and height of the images from the MNIST database are both equal to 28 pixels. Thus, each image can be represented by a 28×28 matrix A with each entry a being an integer that satisfies $0 \leq a \leq 255$, this is due to the fact that each entry represents a gray-scale RGB code. Subsequently, a common pre-processing step is to scale the image data-sets, this can be done by dividing each entry by 255. Scaling is not necessary but it is preferred to work with scaled entries, since each scaled entry a represents a real number that satisfies $0 \leq a \leq 1$.

In addition, I also decided to use a deskewing algorithm posted in a GitHub Pages site since deskewing the images increases the accuracy of the different classification algorithms used in this project. Moreover, another useful pre-processing step was reshaping each 28×28 image into a vector in $\mathbb{R}^{784}$. This makes the algorithms easier to implement since there is no need to use nested for-loops to traverse the vector. Additionally, picturing a matrix $A_{m \times n}$ as a point in $\mathbb{R}^{m \cdot n}$ is highly important for the implementation and understanding of k-nearest neighbors.

3 K-Nearest Neighbors

K-nearest neighbors is a supervised classification algorithm that uses proximity to make predictions about the grouping of an individual data point. Proximity is calculated using the Euclidean distance—being used here as a measure of dissimilarity. The Euclidean distance between vectors $v, w \in \mathbb{R}^n$ is defined by

$$d(v,w) = \sqrt{\sum_{i=1}^{n} (v_i - w_i)^2} = \|v - w\|_2$$

K-nearest neighbors must use a training data-set for the classification of a point. In this case, training data-set can be defined as the set that contains the data points used for the comparison with the unclassified point. Therefore, the algorithm has to calculate the distance between the point we want to classify and each data point inside the training data-set. After this, the distances are sorted in ascending order and the points from the training data-set that generated the first k distances (i.e., the k training points closest to the unclassified point or k-nearest neighbors) are used for classification. The classification step involves retrieving the labels of the k-nearest neighbors and calculating the frequency that each label appears, the label with the highest frequency would be the prediction of the algorithm. Something to take in consideration is that if two labels have equal frequency and this is the highest frequency value then the algorithm would choose the label that achieve that frequency first. Therefore, this algorithm is deterministic even if a tie occurs after calculating the frequency of the labels.

I had implemented an experiment to test the accuracy of using k-nearest neighbors for the classification of the MNIST test data-set with different values for k (i.e., the number of nearest neighbors). The k-nearest neighbors algorithm had the MNIST training data-set as its training data thus it would be comparing each image of the MNIST test data with each image of the MNIST training dataset for different values of k. Subsequently, this experiment was taking too long and could not be completed. However, we can say that the k-nearest neighbors algorithm with the MNIST training data as its training data-set would have had the highest classification accuracy compared to the other algorithms discussed in this project. This is due to the fact that this is the algorithm—implemented in this project—with the largest training data-set, making it more capable to identify outliers. Thanks to this we can see the trade-off between accuracy and computation time when using k-nearest neighbors.

It is to be noted that computation time can be crucial when developing optical character recognition systems of other types of image text (i.e., not only handwritten digits). For example, some writing systems (e.g., Chinese writing system) contain more than 3,000 characters and the MNIST training data-set contains on average 6,000 images for each type of character thus a training data-set that contained the same number of images per character as MNIST for a writing system with 3,000 characters would contain 18,000,000 images. An optical character recognition system that uses k-nearest neighbors and has such a large training data-set would take too long to run. However, there is something we could do to decrease the size of our training data-set, finding good representatives.

3.1 Mean Images

The mean image $\tilde{a}$ of a class C—C is a set of vectors c in $\mathbb{R}^n$— is a new vector in $\mathbb{R}^n$ with entries $\tilde{a_j}$ defined by

$$\tilde{a_j} = \sum_{i=1}^{|C|} \frac{C_{ij}}{|C|},\ \forall_j (1 \leq j \leq n)$$

In other words, an element $\tilde{a_j}$ is equal to the sum of all entries in position j of all images c divided by the cardinality of C (i.e., number of images in the class). We can use k-nearest neighbors to check the nearest neighbors to each mean image. We can see that the great majority of the nearest neighbors of each mean image are from the class depicted. Therefore, we can say that the mean images of each class of digit are good representatives of their respective class.

Following this, we used the 10 mean images we generated as the training set for our k-nearest neighbors algorithm. This algorithm is much faster since the training data set is significantly smaller and had a classification accuracy of 86.94%. The main reason why the classification accuracy was not higher was that the mean images overlook outliers. We can use clustering algorithms to generate images that are good representatives of not only the mean digits but also of the outliers.

4 K-Means Clustering

K-means clustering is a nondeterministic unsupervised clustering algorithm which partitions the data space into k Voronoi cells—the points that belong to a cluster are the points that lie in the Voronoi cell of their centroid. First, we used scikit-learn method .cluster.KMeans() to partition the MNIST training data into a number of clusters k. Next, we had to implement a method that inferred the class of digit that was being depicted by the centroid. This was done by calculating what was the most frequent class label that appeared in the centroid's corresponding cluster and using that label to classify it. After this, we implemented a k-nearest neighbors algorithm that used the centroids generated by k-means clustering as its training data. Then, we calculated the classification accuracy of this algorithm with the MNIST test data for different values of k. We found that when $k = 10$ the classification accuracy was around 73% but as the number clusters increased the classification accuracy increased, getting as high as $\sim 90\%$ when $k = 50$. Figure 1 shows the centroids generated by k-means clustering when $k = 10$ and the inferred label of each centroid.

Figure 1: Cluster centers generated by k-means with $k = 10$.

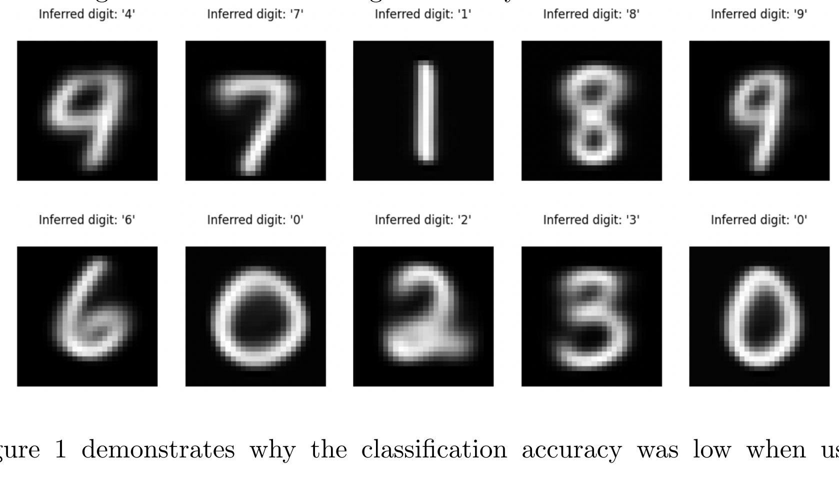

Figure 1 demonstrates why the classification accuracy was low when using

$k = 10$ since it shows that not all classes of digits were represented (i.e., a centroid that represented 5 was missing). It would be impossible for the algorithm to classify correctly the MNIST test images when not all classes of digits were represented in its training data. Next, we tested how increasing the number of nearest neighbors affected the classification accuracy of the algorithm. We found that the classification accuracy decreased as the number of nearest neighbors increased. If the clusters are distributed uniformly (i.e., the number of centroids that depict each class have similar values), we would expect to see the classification accuracy increase until it reaches its peak and then decrease as the number of nearest neighbors increases. Therefore, we suspect that the reason why the classification accuracy decreases as the number of nearest neighbors increases is that the way the clusters are being distributed or generated is not uniform. For this, we will use t-SNE to visualize each cluster and their centroid.

5 Cluster Visualization using t-Distributed Stochastic Neighbor Embedding

t-distributed stochastic neighbor embedding is an unsupervised dimensionality reduction algorithm commonly used for visualization. In this project we used t-SNE to map the first 10,000 images in the MNIST training set $X = \{x_1, x_2, ..., x_{10,000}\} \in \mathbb{R}^{784}$ to $\tilde{X} = \{\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_{10,000}\} \in \mathbb{R}^2$. t-SNE was used after the 50 clusters were generated by k-means thus we knew the labels of the clusters before reducing the dimension of the images, applying k-means clustering after using t-SNE would not have been adequate since t-SNE does not preserve distances and k-means is a distance-based clustering algorithm.

Figure 2: t-SNE visualization of clusters generated by k-means with $k = 50$.

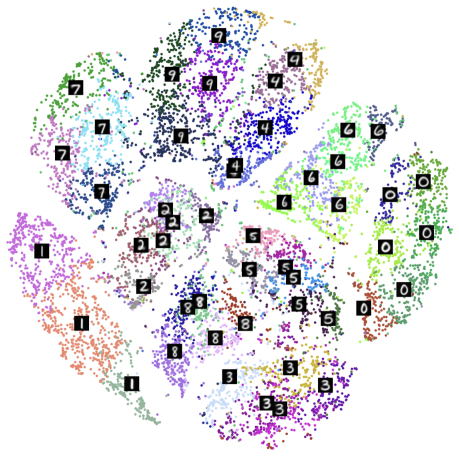

Figure 2 shows the 50 clusters and their centroids after applying t-SNE to the first 10,000 images of the MNIST training set. t-SNE allows us to observe the 10 clusters that represent each class of digit (i.e., the 10 clusters that can be observed when using the labels of the MNIST training set for the distribution of the colors) being divided depending on the way that the 50 clusters were generated (e.g., the cluster that represents the class of digits labeled 1 is divided in 3 clusters while the cluster that represents the class of digits labeled 6 is divided in 6 clusters). Figure 2 also demonstrates why increasing the number of nearest neighbors k reduces the classification accuracy when applying the k-nearest neighbors algorithm that uses the 50 centroids as its training data, the reason is that as you increase k the digits without many clusters representing them get overshadowed. For example, if we try to classify a test image that depicts a digit 1 and use $k = 50$ the algorithm would output 6, 2, 5 or 0 (i.e., the classes of digits with the largest number of centroids depicting them).

6 Optical Character Recognition in Real Life Images

The classification of real life image text using optical character recognition involves the use of different pre-processing algorithms. The reason why is that the training data of our algorithms was the MNIST training set thus before being able to used them for classification we must modify our real life image so that it is similar to the MNIST images. The first step after capturing our real life image is to convert it into a gray-scale image. For this, we used the OpenCV method .cvtColor(). Next, we used adaptive Gaussian thresholding—OpenCV method .adaptiveThreshold()—for image binarization (i.e., converting the image into black-and-white) with a maximum value of 255. Then, we scale it by dividing every pixel by 255. After, this we used OpenCV method .resize() which applied nearest neighbors interpolation to resize the real life image with dimensions 640×480 into a 28×28 image. Finally, the image was deskewed and reshaped into a 784-dimensional vector. Then, we applied three different methods of optical character recognition already discussed in this paper. We used k-nearest neighbors with $k = 11$—an arbitrary positive integer—and with the whole MNIST training set as its training set, k-nearest neighbors with $k = 1$ and with the mean images as its training data, and k-nearest neighbors with $k = 1$ and with the 50 centroids generated by k-means clustering as its training data. The algorithms were tested using different real life images and gave accurate results for most of the images.