Handwritten Digit Classification with KNN Method

Buse Gul ATLI
buse.atli@aalto.fi

Abstract

The goal of this project is to optimally classify hand written digits by using k-nearest neighbor algorithm with different initializations and a simple data. This data is composed of either one stroke or two strokes and consists all digits from 0 to 9. In this algorithm, minimization of the number of data to represent each stroke is aimed. Normalization and centralization methods are implemented and the distance between strokes are calculated by Dynamic Time Warping algorithm. To train the data, k-fold method is used and resulting accuracy and f1 scores are compared for different setups.

1. Introduction

Recently, there has been diverse online applications regarding handwritten digit classification such as recognizing zip codes, digits and characters online. Several machine learning techniques, feature detection and extraction algorithms are implemented to tackle the problems is handwritten digit classification including contrast, noise, rotation, scale and other image changes. MINST and many other sample datasets are constructed to test the accuracy of new algorithms. However, in this report, we will attempt to solve a simpler database. In this dataset, there will be no effect of image noise and we have access to the location of data in the representation of the digit.

2. Data set

We have two different sized handwritten digit data set stored within Matlab files. The training data file has been constructed by two or four writers. Digits are stored in different matrices according to the stroke number. Strokes represent a one effort to write a line without lifting the pen. Each

stroke is represented by 50 data points representing the data point. Therefore, one-stroke digits have 50 points, whereas two-stroke digits are defined by 100 data points. The size of the input matrices is given as N*2*n. N represents the number of the observed digits and 2 indicates the x or y component of the data point. Additionally, lower case n also describes the number of data points. The labels of the samples are given in a separate 1*N vector [1]. The example database for the handwritten digits can be examined in Figure 1.

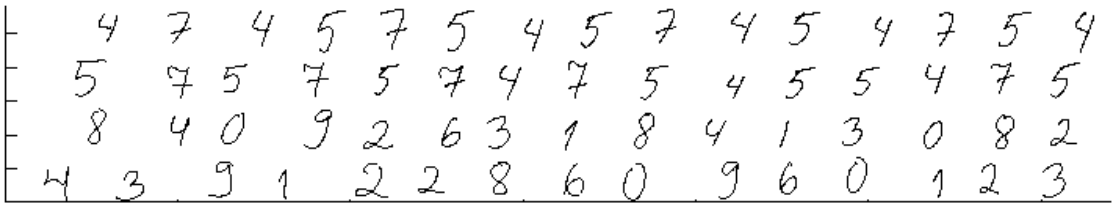

Figure 1 Sample Data For Handwritten Digits

3. Methods

The algorithm consists of three main different blocks, named as preprocessing, dimension reduction in data points and recognition, respectively. There is no appendage feature selection step since the data points are compared directly. Moreover, since there is no separate training/test data, k fold cross validation is used to partition the data between training and test sets. The Flow diagram looks like Figure 2.

Figure 2 Flow Diagram of the Algorithm

1.1 PREPROCESSING

1.1.1 Centralization

Training digits have been collected without looking a specific equivalent writing style. The basic elements could change the writing styles. For instance, a pen or stylus to write with could differ a there could be different level sensitive surfaces resulting distinctive outcomes. Consequently, different styles lead to various aspect ratios, number of strokes, percent of pixels above horizontal or vertical half point, average distance from image center or any other example properties [2]. When the data is investigated, a centralization considered necessary before running any matching algorithm. The centralization function basically moves the center of the bounding box drawn around the digit into to the origin of the coordinate system. (Figure 3)

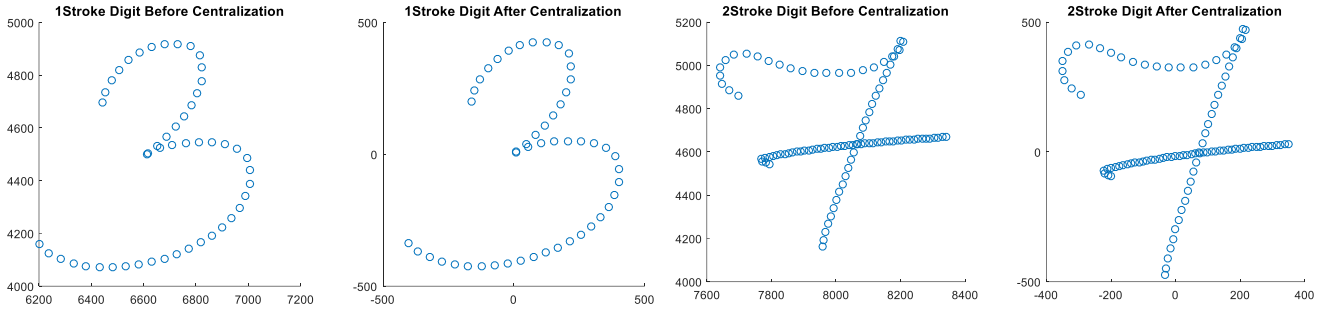

Figure 3 Centralization of Digits

1.1.2 Normalization

Secondly, digits are normalized with respect to a same aspect ratio. The normalization function rescales height and aspect ratio of each bounding box of the data input digit to same value to gain a better comparison. (Figure 4)

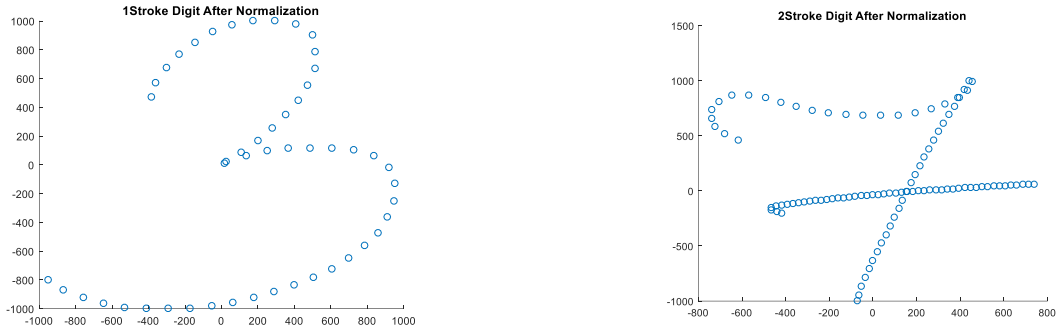

Figure 4 Normalization of Digits

1.1.3 Data Point Reduction

In order to decrease the time to classify a test data, we need to consider reducing the data points per each stroke. For this, decimation method is used. This method reduces the data points by keeping every nth data point and disregarding the others. We know that from the training data, each stroke is represented by 50 data points with x and y coordinates. The algorithm reduced these points to just 2 points with a minimum loss in the accuracy but gained time and fast response.

1.2 K NEAREST NEIGHBOURS

Nearest neighbor method is one of the most intuitive and straight-forward non-parametric algorithm for classification and regression learning. The main idea behind this non parametric approach is simply closer items behaves similarly. The drawback of the nearest neighbor rule is its susceptibility to inherent noise in data. Nevertheless, in our case we have a zero noise digit; so our feature data can obtain good predictions with k nearest neighbor. The classification is done by calculating a cost function and labeling the test data with the majority of k training data having the least error.

Let us define distance between two features a and b, and the cost function is calculated as $d = |a - b|$For all samples from training set x, we can order the distances as

$d_1(x) \leq d_2(x) \leq d_2(x) \ldots\ldots \leq d_N(x)$ where $d_1(x)$ is the nearest sample, $d_2(x)$ is the next to nearest and continue with that order. Than the majority label for the k closest indexes is chosen to predict the label of test sample [3]. In this project, cost function is calculated by dynamic time warping distance.

1.2.1 Dynamic Time Warping Distance

Dynamic time warping (DTW) is a time series alignment algorithm to measure similarity between two sequences even with different temporal frequencies. DTW aims make sequences similar by compressing or stretching them. After the stretching, the distance can be computed by summing the distances of individual aligned elements. This approach mostly used in speech recognition. For instance, the score between the sound signals 'now' and 'noow' will be low and the duration of the o sound have no effect on that [4].

Let's try to compare two different time series with test $X = (x_1, x_2, \ldots\ldots x_N)$ and reference $Y = (y_1, y_2, \ldots\ldots y_M)$. The path remapping the indices of X and Y is called as the warping curve $\varphi(k)$ when $k = 1 \ldots\ldots T$ and written like below:

$$\varphi(k) = (\varphi_x, \varphi_y)$$

$$\varphi_x(k) \in \{1 \ldots\ldots N\} \; and \; \varphi_y(k) \in \{1 \ldots\ldots M\}$$

We can also write a nonnegative dissimilarity function $d(i,j) = f(x_i, y_j) \geq 0$ between the pairs as an input to the DTW algorithm. The common choice is the Euclidean distance. After having the warping functions, we can compute the average accumulated distortion between X and Y:

$$d_\varphi(X,Y) = \sum_{k=1}^{T} d(\varphi_x, \varphi_y) * m_\varphi(k) \,/ M_\varphi$$

$m_\varphi(k)$: per step weighting coefficient

M_φ : normalization constant

The best alignment path can be found by $min_\varphi d_\varphi(X,Y)$. We can say that we should pick the deformation of the time axes X and Y to make them as close as possible [4]. This distance has been widely used in clustering and classification techniques such as k-NN methods.

4. Experiments

Experiments are done with both of the two dataset for comparison. In the result section, tables are constructed with one dataset with 300 total samples Since, there is no clear separation between training and test sets, cross validation is used to measure the mean accuracy and f1 scores.

1.3 K-FOLD CROSS VALIDATION

For large datasets, we can randomly divide it into K parts containing test and training set by repeated use of the same data split differently. We can divide data input data X to K parts and selecting one part as the validation set and other K-1 parts as test set to generate pairs. K is usually selected as 10 or 30. The decision criteria for selecting k is that if K increases the validation set becomes smaller but at time same time increased training size could lead robust estimators [3]. In this project, K is defined both 10 and 30 and results are collected.

1.4 METRICS

Accuracy was obtained as the initial indicator of performance and defined the percentage of correctly classified test items in our experimental setting. For 10-fold, average accuracy was chosen as a baseline. Moreover, mean F1-score for multiple class problems was provided to enable better comparison of the performances.

$$precision = 2 * \frac{(TP)}{(TP + FP)}$$

$$recall = 2 * \frac{(TP)}{(TP + FN)}$$

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

5. Results

Firstly, the optimized number of reduced data points was aimed to find. Although the increased number of data points brings more accurate predictions, the time cost is quite high. The best option is found by reducing data points to two in each stroke with a slight loss in performance. After deciding this criteria, performance metrics are collected for both 10fold and 30fold cross validation (Table 2 and Table 3). When the accuracy and F1 scores are compared, it can be seen that there is no significant difference between. However, the time complexity is better in 30-fold selection. Finally, confusion matrices for different k's (3,5 and 7) are calculated by looking all the predictions and when k = 5 is selected as optimum number. In this matrix, true classes are columns and predicted classes are rows. Furthermore, 1's represent digit 0 and 10's represent digit 9. (Figure 5)

# of data points (x,y)	Elapsed time (sec)
10	5.7577
5	1.614
2	0.21338

Table 1 Time Performance (30-fold,3 Nearest Neighbor)

k	Accuracy	Mean F1 score	Elapsed time(sec)
1	88.667 %	0.9075	0.52536
3	89 %	0.92544	0.52583
5	89.667 %	0.92411	0.52941
7	88.667 %	0.92825	0.5284

Table 2 Performance (10-fold, 2 data for each stroke)

k	Accuracy	Mean F1 score	Elapsed time(sec)
1	88.667 %	0.93689	0.1948
3	88333 %	0.9357	0.19534
5	89.667 %	0.94506	0.19736
7	89 %	0.94302	0.19829

Table 3 Performance (30-fold, 2 data for each stroke)

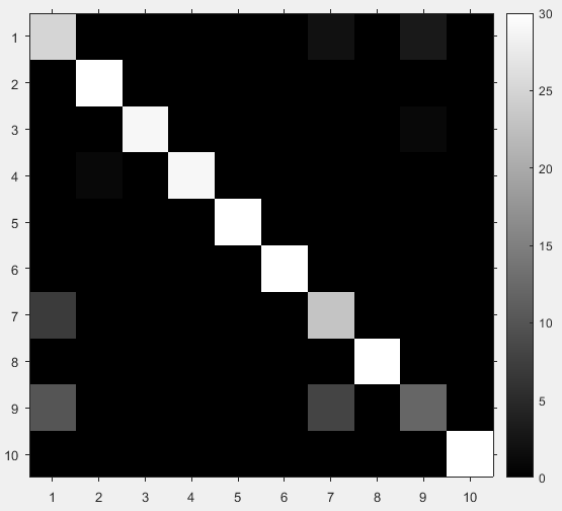

Figure 5 Confusion Matrix (30-fold 5-NN classifier)

6. Discussion

In conclusion, different k-NN non-parametric methods with DTW distances are applied to handle handwritten digit classification problem. Since there's no noise in the training data, the results are comparably good. Otherwise; filtering, edge detection, thinning of the digit image or other preprocessing tasks should be added. To have faster response, the number of data for each stroke is reduced to only two points consisting x and y coordinates of the ink. These results could only offer initial analysis and solutions to a better classification can be done by other machine learning methods.

7. Bibliography

[1] Handwritten Digit Classification. (12 3 2016). Retrieved From: http://www.cis.hut.fi/Opinnot/T-61.3025/Harjoitustyot/Digits/index.html.

[2] Handwriting recognition. (12 3 2016). Retrieved From: https://en.wikipedia.org/wiki/Handwriting_recognition.

[3] E. Alpaydin, in *Introduction to Machine Learning*, London, The MIT Press, 2010, p. 209.

[4] T. Giorgino, "Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package," *Journal of Statistical Software,* 2009.

8. Appendix: Method Implementations in Matlab

Decimate, Centralize, NormalizeSize and DtwDistance Matlab functions are used. Some additional functions are in below:

```
function [F1 C] = f1score( predicted, observed )
C = confusionmat(observed,predicted);
%precision calculation from directly confusion matrix
% 1 is added to prevent division with zero and NaN output
precision = (diag(C)+1) ./ (sum(C,2)+1);
precision = mean(precision);
%recall calculation from directly confusion matrix
recall = (diag(C)+1) ./ (sum(C,1)'+1);
recall = mean(recall);
F1 = 2* (precision*recall)/(precision + recall);
end
```

```
function [prediction] = knn_func( trainingData, trainingLabel, testData,...
KNN, d_max)
    for j = 1:size(testData,1)
        test=reshape(testData(j,:,:),[size(testData,2),size(testData,3)]);
        dist=zeros(1, size(trainingData,1));
        for jj = 1:size(trainingData,1)
            training=reshape(trainingData(jj,:,:),[size(trainingData,2),...
            size(trainingData,3)]);
            %calculate the DTW distance and save it to dist array
            [dist(jj),distM] = DtwDistance(test,training, d_max);
        end
        %sort the distances in order
        [dist,idx] = sort(dist, 2, 'ascend');
        %K nearest neighbors with closest KNN indexes
        dist = dist(:,1:KNN);
        idx = idx(:,1:KNN);
        %majority vote
        prediction(j) = mode(trainingLabel(idx),2);
    end
end
```