



Open your mind. LUT.
Lappeenranta **University of Technology**

LUT Computer Vision and Pattern Recognition Laboratory

Pattern Recognition and Machine Learning Recognition BM40A0702

Practical Assignment

Digits 3-D

Name	Student No.	Role
Tuo Yang	589715	Pre-processing, DTW, documentation
Suzan Yemane	592919	Data classification, data visualization, documentation
Subhashree Rautray	592964	Data classification, classification visualization, documentation

Date: 07-12-2020

Contents

1. Problem description.....	3
2. Solution description.....	3
2.1 Data processing phase.....	3
2.1.1 First operation of data processing.....	4
2.1.2 Second operation of data processing.....	5
2.1.3 Third operation of data processing.....	5
2.1.4 Data preprocessing results:.....	6
2.2 Classifier designing, training and prediction phase(KNN+DTW).....	6
2.2.1 DTW algorithm procedures:.....	8
2.2.2 Example to explain this algorithm.....	9
2.2.3 Modification to DTW for decreasing the amount of calculation:.....	9
2.3 Prediction and evaluation.....	10
3. Experimental results	10
Reference.....	12

1. Problem description

To begin with, we have got 1000 three dimensional stoke-coordinated data of digits ranging from 0 to 9, 100 samples for each digit, and they are generally N by 3 matrices, which is time series coordinates data collected using the index finger when write one digit in the three- dimensional space. Our goal is to design one pattern recognition system which can recognize the hand-written digits, also to classify them correctly based on the given 3-D location time series.

2. Solution description

So we have designed one classifier using k-Nearest neighbors with DTW (Dynamic Time Warping) to realize this recognition task, below are the steps which we have followed to developed the classifier .

2.1 Data processing phase

We got some abnormal samples inside the dataset, when we draw take X and Y coordinates of each data sample and use them to make 2D plots, there are some samples' plots which shape was not like the digit shape they are marked with.

Therefore, we draw 2D digits' figures of all samples out, because of limitation of space, we did not paste all digits' figure here, we only take digit 0's 2D figure as the example, as we can see from figure 2.1, sample 33 and sample 100's shape was totally not like digit 0 shape. so for other digits samples we have checked, there still such kind of abnormal samples also exists.

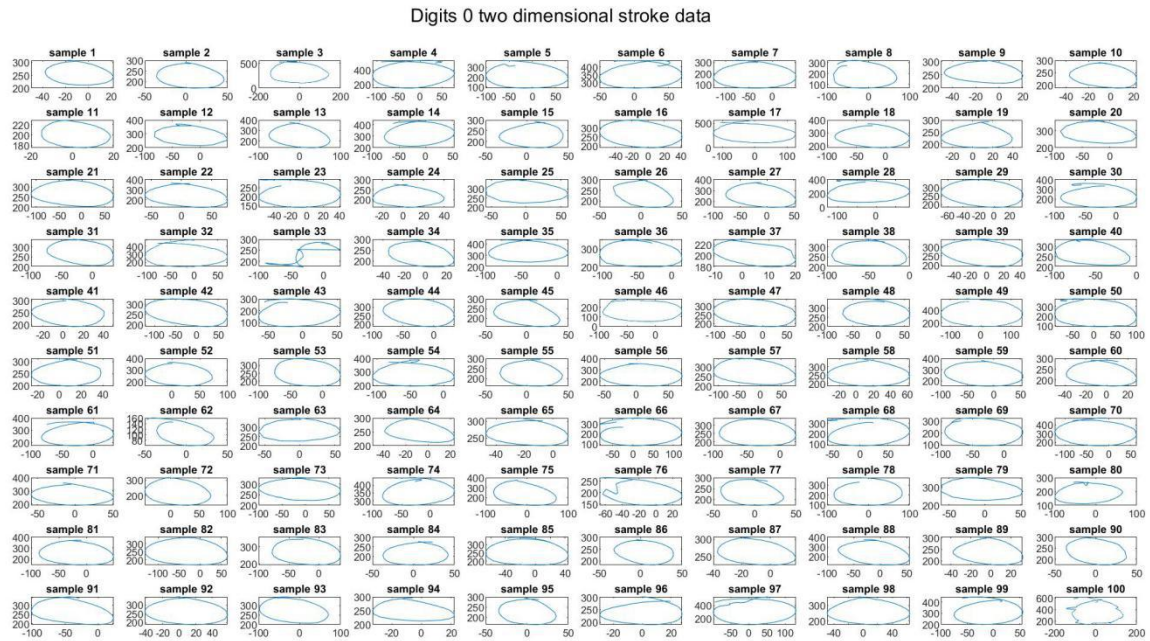


Figure 2.1 Digit 0 two-dimensional stroke data

2.1.1 First operation of data processing

That's the reason why we need to remove such kind of samples from the whole data sets, which is one crucial part of data preprocessing, because once these samples would be used as training data set for the classifier, its data will let the classifier learn wrong patterns(shape) from these samples thus making wrong classification for the testing dataset samples.

All indexes of abnormal samples are shown as figure 2.2 below:

```
removal_indices = [33 100 103 107 114 121 122 133 ...
                  138 146 162 164 167 168 169 171 177 ...
                  178 183 191 192 195 200 279 442 444 ...
                  446 468 499 593 623 667 677 694 764 ...
                  821 922];|
```

Figure 2.2 indexes of abnormal samples we need to remove from the whole dataset

2.1.2 Second operation of data processing

After removing the abnormal data samples, the next step of data preprocessing is to split the whole dataset into training dataset and testing dataset, and we have kept the percentage between the training dataset and testing dataset is 8:2(this percentage, according to our experiments, which will make classifier achieve the best classification performance), so we need randomly take 80% and 20% data samples from the whole dataset as training dataset and testing dataset respectively. However, a new problem arises, if we just randomly take 80% data samples from dataset, it's most likely the samples we have taken will not cover all digits' data samples, which could make classifier not be able to recognize all digits' patterns ranging from 0 to 9 when using trained model to classify new samples.

Therefore, a more reasonable strategy we have adopted, that is, randomly taking 80% and 20% from each digit's data samples respectively as sub-training dataset and sub-testing dataset, then combine each digit's such kind of datasets together, here so we can get two general training dataset and testing dataset. Meanwhile, we need to add class labels to each data sample first, then we can use the method mentioned above to randomly take data samples.

Two cell matrixes have been gained after this step, one 770 by 2 training dataset matrix and one 193 by testing dataset matrix, each matrix's first row presents data samples, each sample is a N by 3 matrix presenting stroke time series data in 3-D space (N depends on how many coordinates inside the sample data), the second row of matrix presenting corresponding class labels for each data sample.

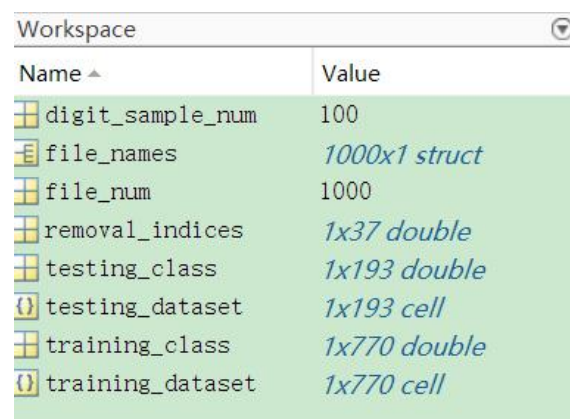
2.1.3 Third operation of data processing

To make training and testing data samples more in line with real data, and prevent the classifier from overfitting, we need to mess up the order of training and testing

samples by using matlab function randperm, after this processing, separately take all samples' class labels from the second row of training and testing samples as two class label vectors.

2.1.4 Data preprocessing results:

Four new variables will be generated after doing three processing steps mentioned to all data samples, just as the figure 2.3 below shows:



The image shows a screenshot of the MATLAB Workspace window. It contains a table with two columns: 'Name' and 'Value'. The variables listed are: digit_sample_num (100), file_names (1000x1 struct), file_num (1000), removal_indices (1x37 double), testing_class (1x193 double), testing_dataset (1x193 cell), training_class (1x770 double), and training_dataset (1x770 cell). The table is highlighted with a green background.

Name	Value
digit_sample_num	100
file_names	1000x1 struct
file_num	1000
removal_indices	1x37 double
testing_class	1x193 double
testing_dataset	1x193 cell
training_class	1x770 double
training_dataset	1x770 cell

Figure 2.3 training and testing datasets after preprocessing

Notes for figure 2.3:

Four important variables need to be observed:

training_dataset: including all training data samples inside, each cell element presents one sample (N by 3 time series data).

training_class: corresponding class labels for all training dataset samples.

testing_dataset: including all testing data samples inside, each cell element presents one sample (N by 3 time series data).

testing_class: corresponding class labels for all testing dataset samples.

2.2 Classifier designing, training and prediction phase

In fact, all data samples inside the data set are time series data, so the classification problem in this case can be taken as this problem: How to classify time series data?

Time sequential data could be taken as one high-dimensional data, the dimension is

the length of the time sequence. Compared with the general input vector, there is a little difference for the time sequential vector that its different dimensions are not independent from each other, inversely they are highly-related. Take one stroke data as the example, we assume the data is a 32 by 3 matrix, each row's data presents one coordinate data at one moment, but each column of this row data X Y Z coordinates are not related with each other, instead each row's whole data is highly related with each other because they are connected with the time, so the dimension of this matrix as input vector should be 32.

SVM (Support Vector Machine), Naïve Bayesian such kinds of classification algorithm generally will assume that the portions of input vectors are independent from each other, so it's not suitable directly using them on time-series classification.

KNN (K nearest neighbors) this kind of supervised algorithm, which is simple and rude but workable. It depends on similarity calculation between sample points, sample points here refer to each time sequential data. Similarity's calculation is equal to the distance calculation between samples, so the core of the question has become how to calculate the similarity between time sequences. In fact, the distance calculation between time sequences could be realized by using DTW (Dynamic Time Warping) algorithm. Calculating distances between two time series data by aligning their length. Let us get back to the case, we could use KNN to realize multi-class classification here, as we have talked in the last paragraph, we need to calculate the similarity(distance) between samples, but all data samples are time series data which size is N by 3(N is totally different for each sample), so we cannot directly use common distances like Euclidean, Cosine, Mahalanobis or Chebychev. So that's the reason why we chose DTW here, for KNN, we would not spend a lot of time explaining here, the optimal k's value, is identified as 2 by experiments, when this model will get the highest classification performance. but for DTW, we will introduce the algorithm give a simple example to explain how to calculate DTW distance between time series.

2.2.1 DTW algorithm procedures:

1. First there must be two or more known time sequences need pairwise comparison, so we assume that there are two sequences $A = \{a_1, a_2, a_3, \dots, a_m\}$
 $B = \{b_1, b_2, b_3, \dots, b_n\}$, the dimension $m > n$.
2. Then use the Euclidean distance to calculate the distance between every two

$D(a_1, b_1)$	$D(a_1, b_2)$...	$D(a_n, b_1)$
$D(a_2, b_1)$	$D(a_2, b_2)$...	$D(a_n, b_2)$
...
$D(a_m, b_1)$	$D(a_m, b_2)$...	$D(a_m, b_n)$

points in each sequence $D(a_i, b_j)$ where i 's range $[1, m]$, j 's range $[1, n]$, draw the figure 2.4 shown as below:

figure 2.4 Euclidean distance matrix between two time sequences

3. Next step is to find the shortest path from the figure above. From $D(a_1, a_2)$ along with a certain path to $D(a_m, b_n)$. This certain path satisfies conditions like: If current point is $D(a_i, b_j)$, the next point must be chosen from points $D(a_{i+1}, b_j)$, $D(a_i, b_{j+1})$ and $D(a_{i+1}, b_{j+1})$, and the path must be the shortest one. Calculation of shortest path length is realized by using dynamic programming solution, in other words, considering distance from the three upper left corner points like $(i-1, j)$, $(i-1, j-1)$ and $(i, j-1)$ ones to (i, j) 's shortest distance when calculating the shortest path to the point (i, j) .
4. Next using backtracking algorithm to output the shortest path points from $D(a_1, b_1)$ to $D(a_m, b_n)$. The summation of them is the DTW distance.

Notes: For 3-D time series data, the Euclidean distance would be the distance between two three-dimensional coordinates, we can directly use three-dimensional Euclidean distance formula to get the result.

2.2.2 Example to explain this algorithm,

Two column vectors $a = [8 \ 9 \ 1]'$, $b = [2 \ 5 \ 4 \ 6]'$ are known:

To get: the shortest distance between two vectors using dynamic time warping.

Firstly, calculating Euclidean distance matrix for corresponding pairwise points:

Secondly, calculating the cumulative distance (from 6 to 5)

6	3	4	2
7	4	5	3
1	4	3	5

6	9	13	15
13	10	14	16
14	14	13	18

Calculating ways will be like this:

$$D(1,1) = d(1,1) = 6$$

$$D(1,2) = D(1,1) + d(1,2) = 9$$

...

$$D(2,2) = \min(D(1,2), D(1,1), D(2,1)) + d(2,2) = 6 + 4 = 10$$

...

$$D(m, n) = \min(D(m-1, n), D(m-1, n-1), D(m, n-1)) + d(m, n)$$

2.2.3 Modification to DTW for decreasing the amount of calculation:

The algorithm above calculation complexity is $O(m*n)$, m and n respectively are lengths of two sequences. It can be accelerated by adding local matching constraints, that is, a certain point in the sequence can only be paired with points within the range w before and after its position. This move will greatly decrease the calculation amount DTW distance.

2.3 Prediction and evaluation

After using KNN+DTW distance to train the model by using training dataset, the labels for testing dataset could be predicted by using this model, we can use the prediction labels to make comparison with real labels to calculate the right-classified rate and visualize this prediction results meanwhile.

3. Experimental results

Totally we got 193 testing data samples here, after using KNN+DTW algorithm to make prediction, prediction label vector C for testing will be gained by this model, and we can use the formula below to calculate the right classified rate:

```
C = data_classify(training_dataset, training_class, testing_dataset, 2);  
correct_classified_rate = sum(C == testing_class)/length(testing_class);
```

Figure 3.1 right-classified rate calculating formula (testing dataset)

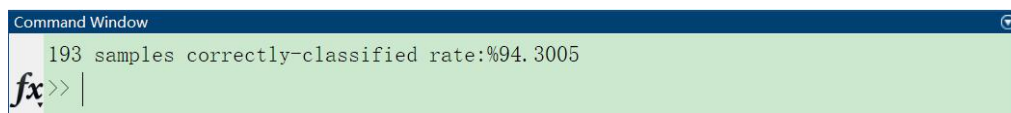


Figure 3.2 right-classified rate of testing dataset

The right classified rate is(usually stable around 94%) .

The visualization of classification results for testing dataset is shown as figure 3.3 and figure 3.4, the subplot titles which is marked with red presents misclassified samples:

Classification results for 193 samples

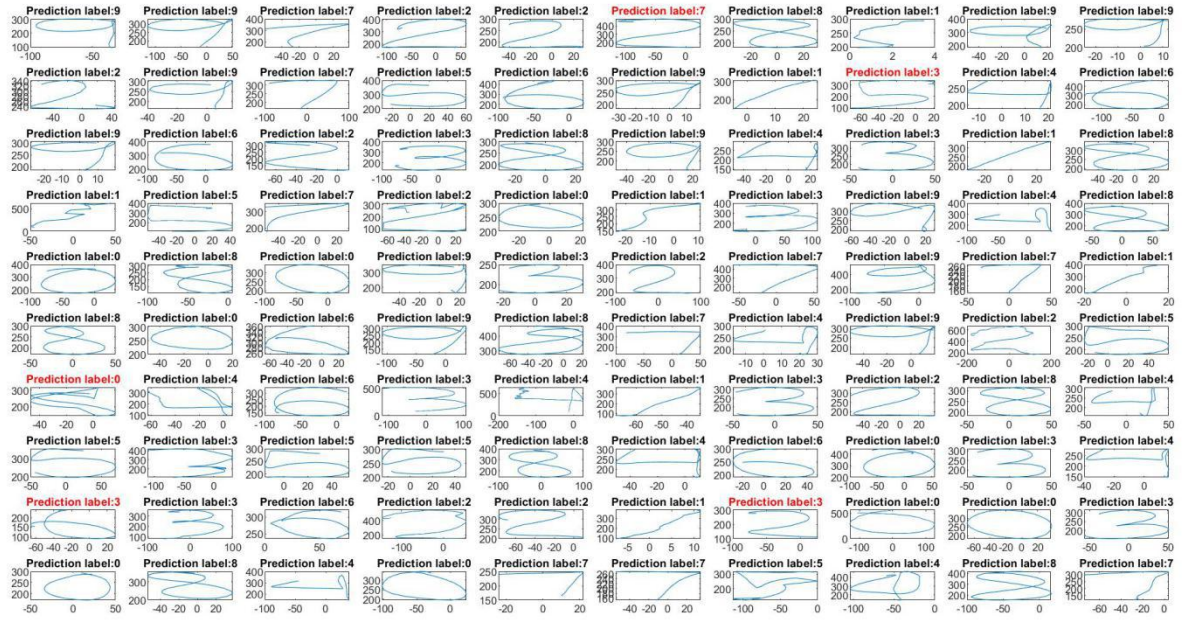


Figure 3.3 prediction labels from 1st to 100th testing samples

Classification results for 193 samples



Figure 3.4 prediction labels from 101th to 193th testing sample

Reference

Time-Series Analysis: Wearable Devices Using DTW and KNN

Link:

<https://www.sflscientific.com/data-science-blog/2016/6/4/time-series-analysis-fitbit-using-dtw-and-knn>

Dynamic time warping

Link: https://en.wikipedia.org/wiki/Dynamic_time_warping

KNN:

Link: <https://www.unite.ai/what-is-k-nearest-neighbors/>