LAPPEENRANTA UNIVERSITY OF TECHNOLOGY
School of Engineering Science
Computational Engineering and Technical Physics
BM40A0702 Pattern Recognition and Machine Learning

Practical Assignment
Digits 3-D

Supervision:
Prof. Lasse Lensu

Students:

2020

1 Introduction

In this document the report of the 3D-digits classification problem has been presented. The task was to develop a learning system for hand-written digits 0, 1, ..., 9 from the leap motion sensor. The data consists of 1000 samples with 100 samples per digit. The approach and outline for the solution framework are: data parsing and preprocessing, building k-nearest neighbor classifier, finding best parameters for Dynamic Time Warping distance measure and Euclidean distance metric performance by K-fold cross-validation, comparing DTW and Euclidean distances in lack of data case and finally evaluation and comparison of the results.

2 Data

The given training data consists of 1000. mat/.csv “stroke_{0}_{0001}.mat” files where first number refers to a digit, from 0 to 9, and second number represents sample item.

2.1 Data Parsing

Function parseDigits.m utilizes “training_data” folder which must be located in the same directory with the function. ParseDigits.m returns cell array with the structure {1,10}{1,100}. Each cell array contains 100 data samples for digits from “0” to “9”.

2.2 Data Preprocessing

Data preprocessing demo is represented in file dataPreprocessing.m.

- First step in data preprocessing is to obtain information on a data structure. To do so we started with plotting several samples for digit “0”. In the Figure 1, we concluded that it is necessary to center and rescale data samples as they all vary in amplitude and angle. Function normalizeDigits(data) accepts raw data and perform z-score normalization. The result of scaling and centering are represented in Figure 2.

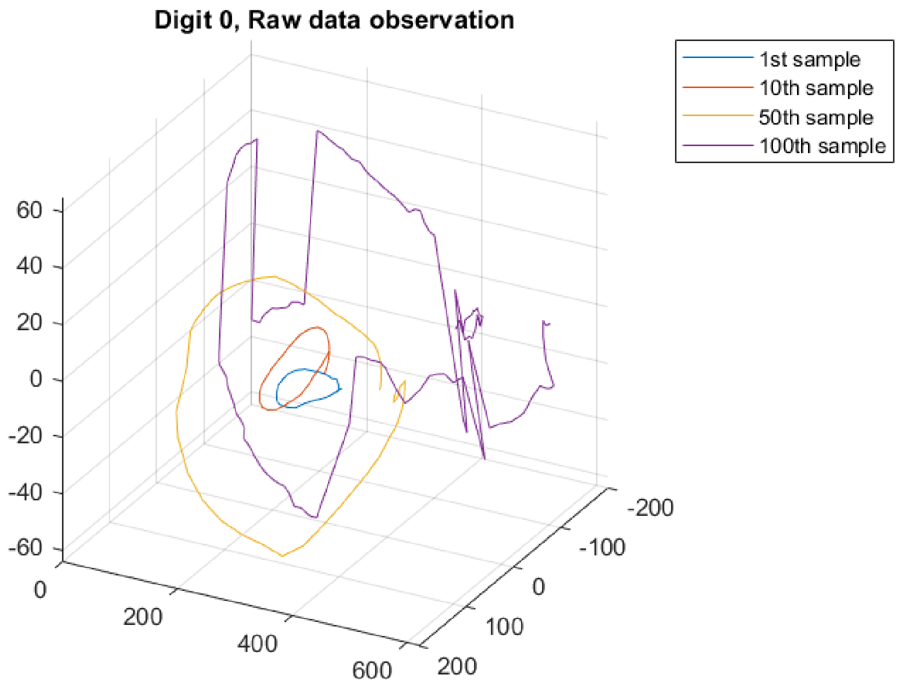

Figure 1: Raw data samples

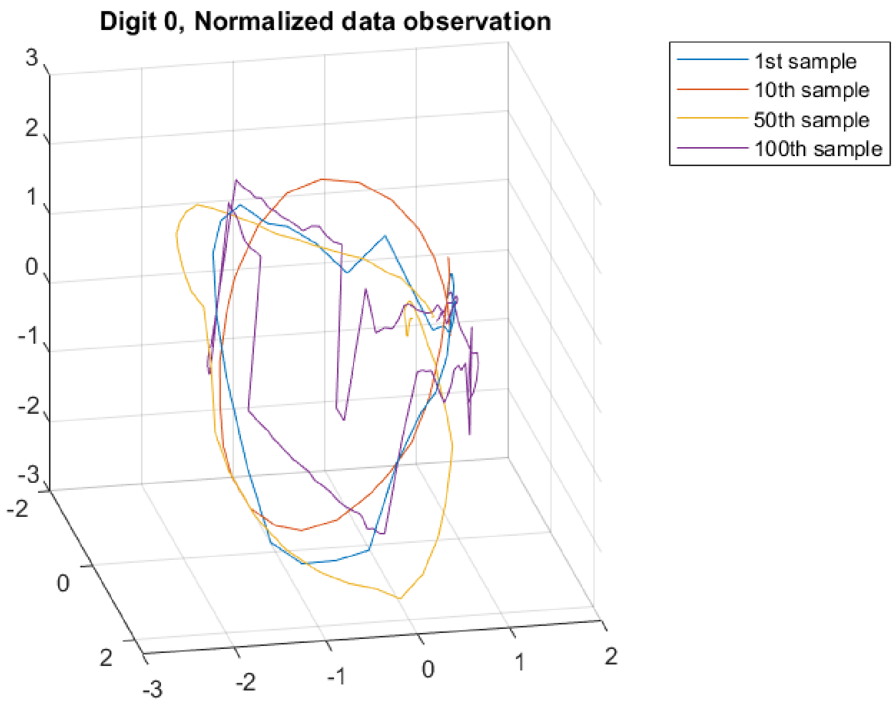

Figure 2: Normalized data

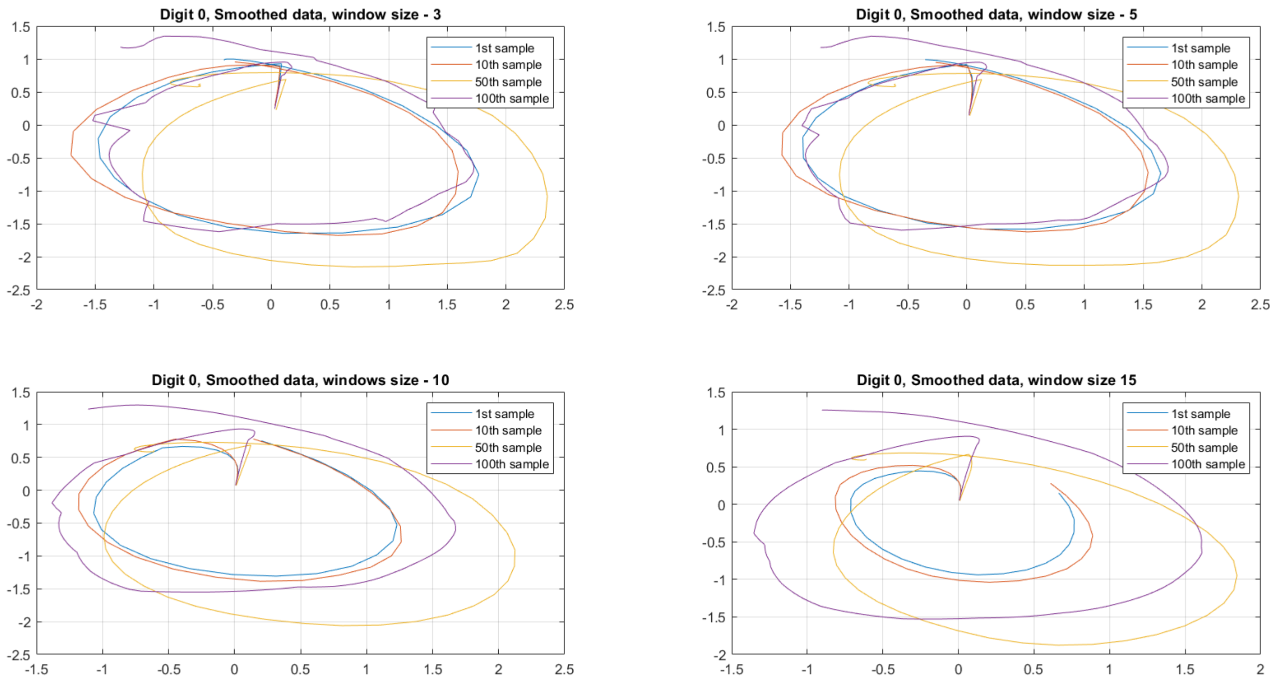

Figure 3: Filtered data

- As Figure 2 shows there is some noise and spikes in data samples. In order to reduce their impact on classification performance, signal filtering can be applied. To get a smooth signal from leap motion data samples, a moving average filter, a simple and common technique, has been implemented. Function smoothDigits(data, windowSize) performs filtering with given windowSize parameter. The result has been depicted in Figure 3.
- To evaluate intrinsic dimensionality, we applied principal component analysis to one of the samples. Variance explained by the first two axis, 'X' and 'Y, was 65.1% and 33.8%, accordingly, while 'Z' axis has only 1%. Therefore, we do not consider third dimension at all.
- Data samples have uneven length. Thus, without resampling, kNN performance with Euclidean distance measure can suffer. To determine resample parameter we measured distribution of length in the data. Figure 4, demonstrates that the most frequent length is in the boundary between 40 and 50, create resampled data with unit length 50. Function resampleDigits.m apply such transformation.

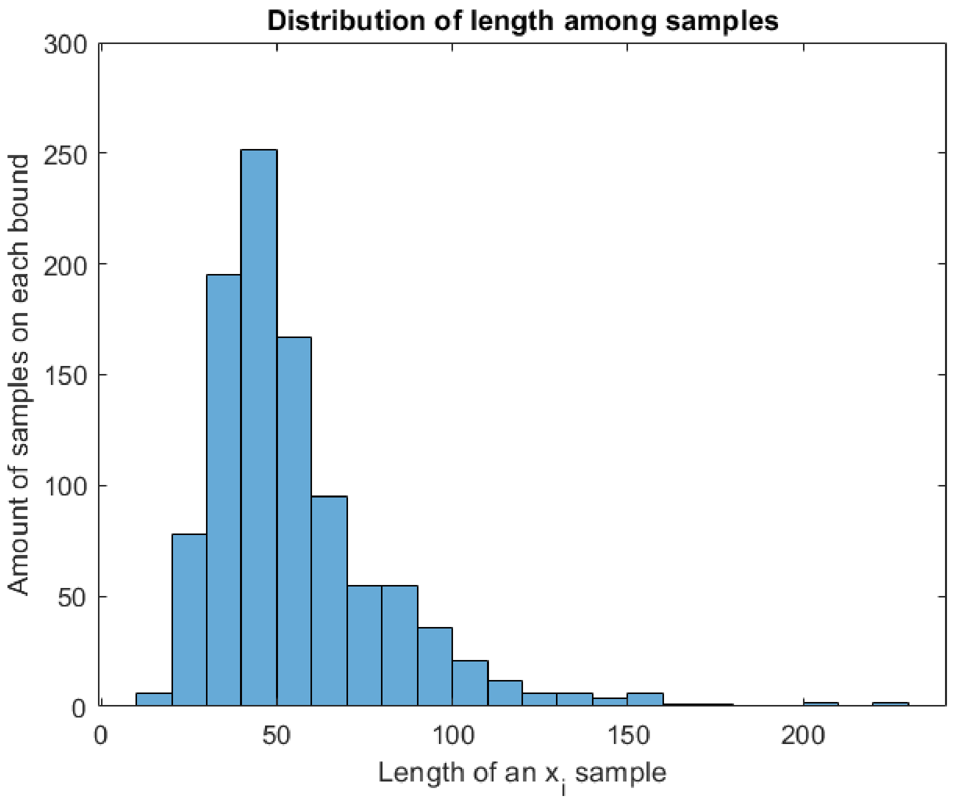

Figure 4: Length Distribution between the data samples

3 Feature Extraction

In this project, we did not consider any feature engineering because:

- kNN is able to classify samples in less than a minute (model built from scratch).
- Accuracy of the classifier is relatively high.
- No instructions about compulsory feature extraction from data.

4 Distance Measures

- **Euclidean distance**

  Euclidean distance on the data set with uneven length between samples demonstrate that lock-step measure (one-to-one mapping of data, figure 5) is very sensitive to noise and misalignment in time, so it is not able to manage local time shifting [2]. Therefore, kNN performance with such data set drops significantly. A possible solution is to resample time-series data, so they have a unit length. Applying such technique helped us to overcome this drawback and increase accuracy of a classifier (Table 1).

- **Dynamic Time Warping**

  Another decision that could be made to handle uneven data sample length is using elastic measure, which allow to “stretch” or “compress” a time-series to arrange a better match with another time ordered data sample, as shown in Figure 5.

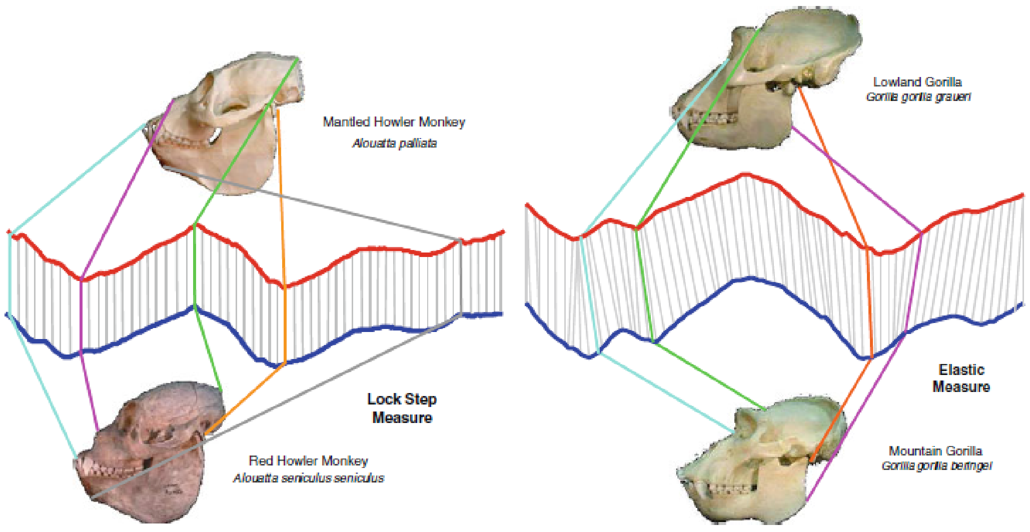

Figure 5: Lock-step measure illustration and elastic measure illustration [2]

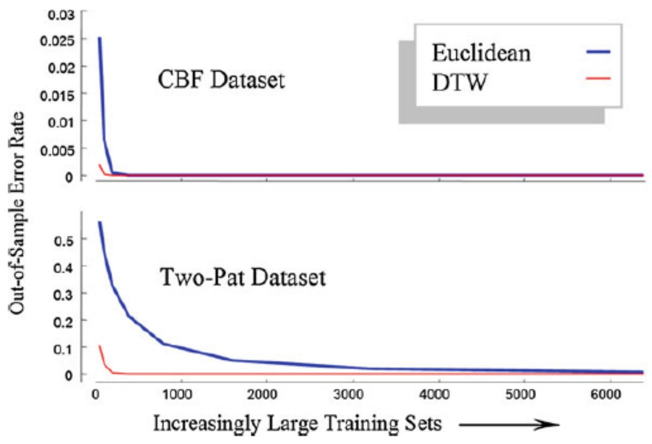

Figure 6: The error rate for 1-NN Classification for increasingly large instantiations of CBF and Two-pats [2]

One of the state-of-the-art solution is a common speech recognition tool [1], Dynamic Time Warping (DTW). It allows us to map data points “one-to many”. However, DTW is sensible to a data size and its time complexity, $O(wn)$, depend on a warping window (w), unlike Euclidean non-parametric distance measure with a time complexity $O(n)$ [2]. One more feature of DTW is that on small data sets its accuracy is much higher than Euclidean Distance (Table 5, Figure 6).

You can observe outcomes for dynamic time warping distance between similar class samples and between dissimilar class samples in Figure 7 and Figure 8. As expected, for alike signals the distance measure is less than for a different ones.

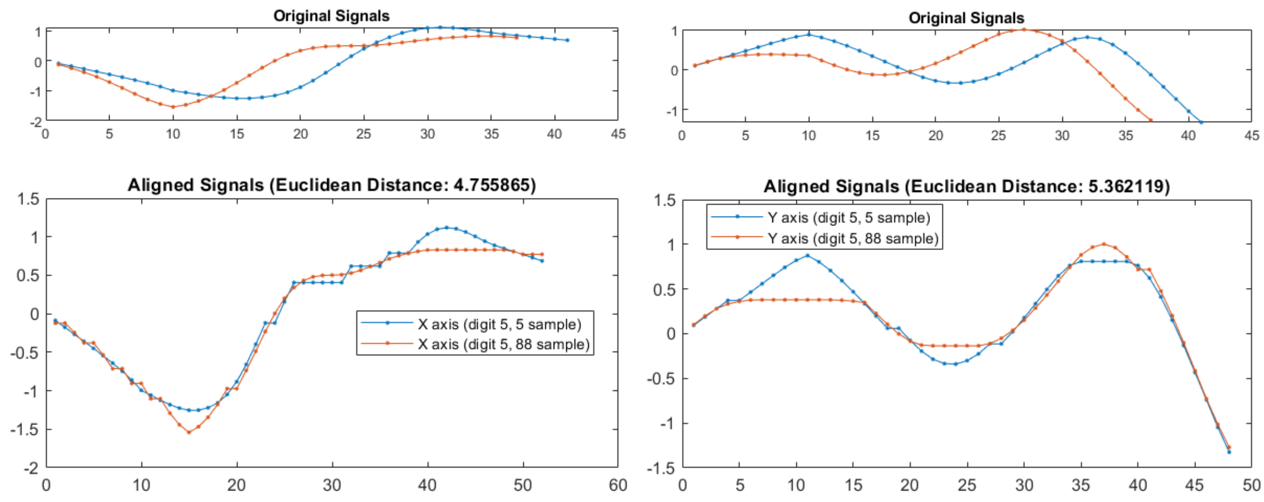

Figure 7: Distances between X and Y signals for the same digit (5), samples №5 and №88

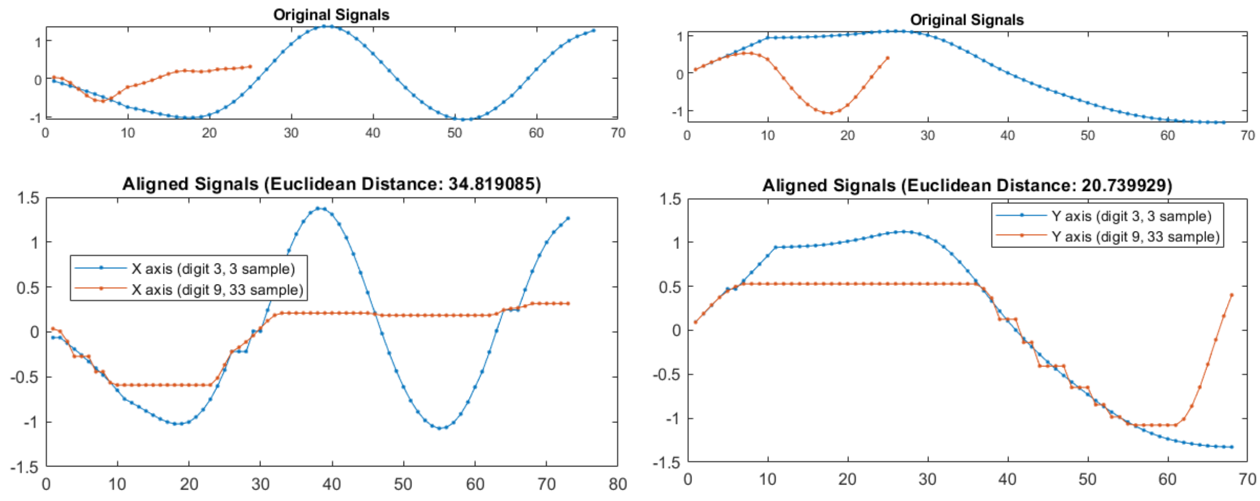

Figure 8: Distances between X and Y signals for digit 3 and 9, samples №3 and №33

5 Classifier Design

File knn.m contains k-nearest neighbor classifier. It has a relatively straightforward implementation algorithm:

- Calculate distance matrix between training and test data samples,
- Sort the distance matrix in ascending order and select k closest neighbors to the given test sample,
- Select the most frequent class among the k neighbors.

A big drawback in this technique is that a time complexity depends on the data set size. The way of computing the distance between given train samples and test samples is the most important feature in this method. In our case we used and compared different distance measures with various data structure to evaluate procs and cons of each approach.

6 Results

File crossValidation.m contains k-Fold cross-validation to evaluate effective number of neighbors in kNN classifier and suitable adjustment window size, classifier accuracy in dependence on the amount of available train data.

- **First set of experiments**

  In first set of experiments we locked the proportion of test and train samples as 0.6 and 0.4, accordingly. Then started k-Fold cross-validation to find out best parameters for Euclidean and DTW distance measures. In the figures circles represent a mean accuracy after 10-fold cross-validation and red dotted line is a mean accuracy between all the experiments.

  In Figure 9 and Table 1, you can observe how varying k, number of neighbors parameter, impacts on the classifier performance with Euclidean distance measure and unit data legnth resampled to 50.

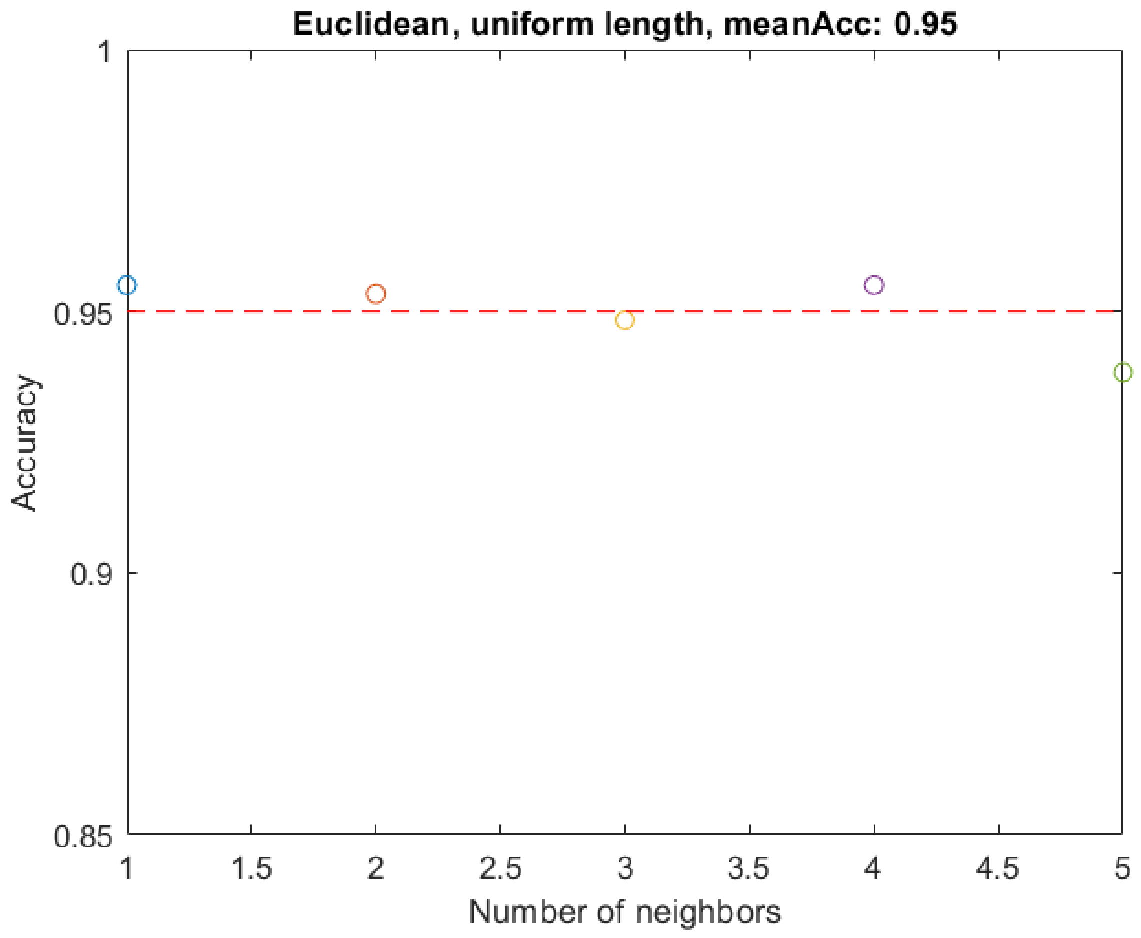

Figure 9: Euclidean Distance, uniform length

The most accurate performance kNN classifier indicates when number of neighbors equals to **1**.

Table 1: Euclidean distance, uniform length

Distance Measure	Train data size	Test data size	Window size	Number of neighbors	Accuracy (10-Fold cross-validation)
Euclidean	0.6	0.4	-	1	**0.9533**
				2	0.9550
				3	0.9483
				4	0.9550
				5	0.9380

In Figure 10 and Table 2, performance of the kNN classifier with Euclidean distance is shown. We can make an assumption that Euclidean distance is very sensitive to misalignment in time [2], therefore accuracy has declined significantly.

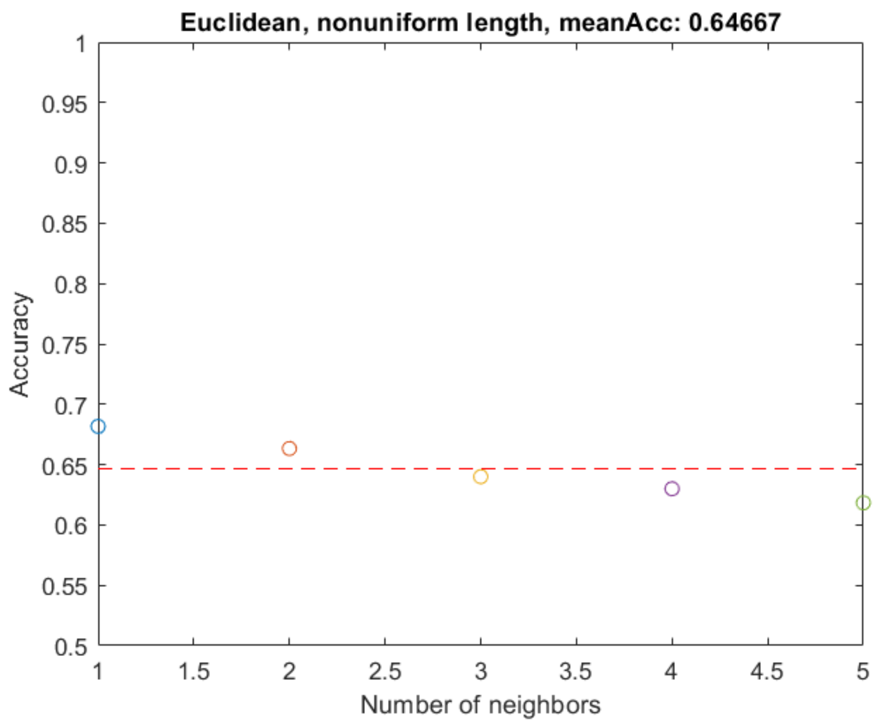

Figure 10: Euclidean distance, nonuniform length

1-NN classifier still outperforms the others, but its efficiency drops faster with uneven data length.

Table 2: Euclidean Distance, nonuniform length

Distance Measure	Train data size	Test data size	Window size	Number of neighbors	Accuracy (10-Fold cross-validation)
Euclidean	0.6	0.4	-	1	**0.6817**
				2	0.6633
				3	0.6400
				4	0.6300
				5	0.6183

Results of the experiments in Figure 11 and Table 3 indicate that kNN classifier with DTW distance measure and uniform sample length becomes slightly better as window size increases and vice versa for number of neighbors.

In Figure 12 and Table 4, outcomes from the classifier with DTW distance measure and nonuniform sample structure are comparable to the previous figure: accuracy slightly better as window size increases and vice versa for number of neighbors.

For all the experiments 1-NN classifier is slightly ahead of the others. DTW and Euclidean

Table 3: DTW, uniform length

Distance Measure	Train data size	Test data size	Window size	Accuracy (10-Fold cross-validation)				
				Number of neighbors				
				1	2	3	4	5
DTW	0.6	0.4	1	**0.9633**	**0.9633**	0.9583	0.9583	0.9533
			2	**0.9677**	0.9633	0.9617	0.9650	0.9567
			3	**0.9683**	0.9650	0.9667	**0.9683**	0.9617
			4	**0.9683**	**0.9683**	0.9717	0.9667	**0.9683**
			5	**0.9733**	0.9717	0.9700	0.9717	0.9700

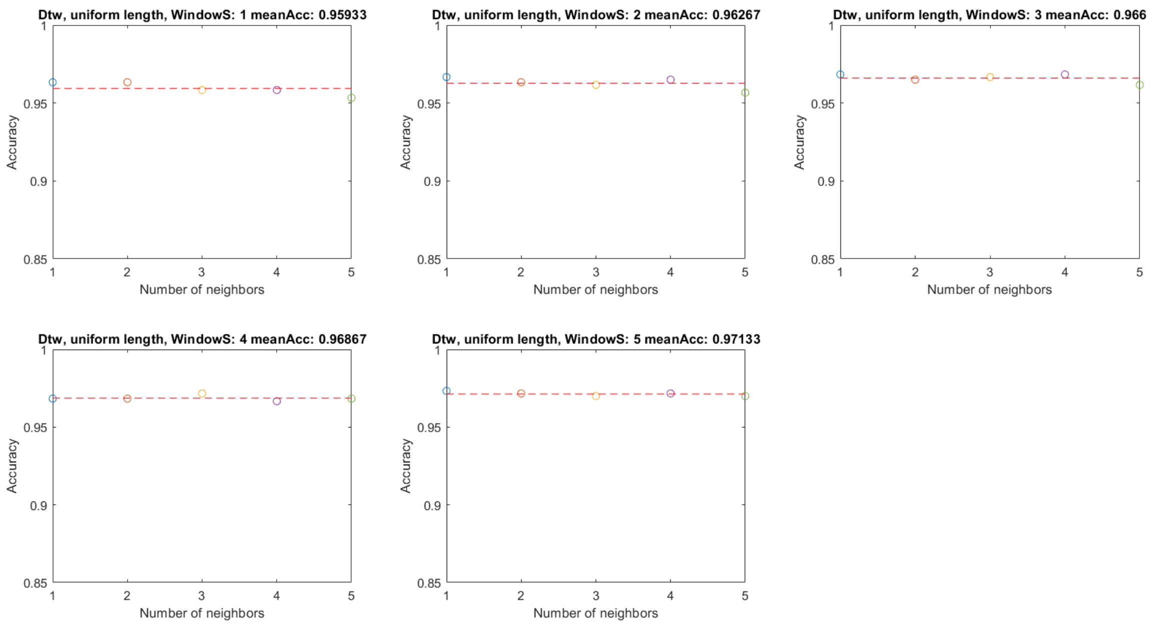

Figure 11: DTW distance measure, uniform length

Table 4: DTW, nonuniform length

Distance Measure	Train data size	Test data size	Window size	Accuracy (10-Fold cross-validation)				
				Number of neighbors				
				1	2	3	4	5
DTW	0.6	0.4	1	0.9517	**0.9533**	0.9500	0.95317	0.95317
			2	**0.9633**	0.9583	0.9550	0.9450	0.9467
			3	**0.9617**	0.9533	0.9616	0.9567	0.9533
			4	**0.9583**	0.9533	0.9567	0.9550	0.9450
			5	**0.9617**	0.9467	0.9533	0.9483	0.9400

measures have comparable results but Euclidean distance requires an appropriate data structure, resampling in our case. Without resampling DTW has much better accuracy in contrast with Euclidean distance. However, DTW distance takes several times more time to compute, hence we can't claim that one of the measures is better.

- **Second set of experiments**

  On this set of experiments, we studied how lack of available train data affects on a classifier performance. We again use implemented crossVlidation.m function Figure 13 and Table 5, confirms that in small amount of train data DTW performs better than Euclidean distance.

Table 5: Impact of train data size on accuracy for different distance measures

Train data size	Test data size	Distance Measure			
		Euclidean		DTF	
		Data Length			
		uniform	nonuniform	uniform	nonuniform
0.8	0.2	0.9650	0.7300	**0.9750**	0.9450
0.4	0.6	0.9517	0.6717	**0.9583**	0.9433
0.2	0.8	0.9325	0.6275	**0.9450**	0.9237
0.1	0.9	0.8978	0.6344	**0.9322**	0.9011
0.05	0.95	0.7716	0.5358	0.8274	**0.8579**

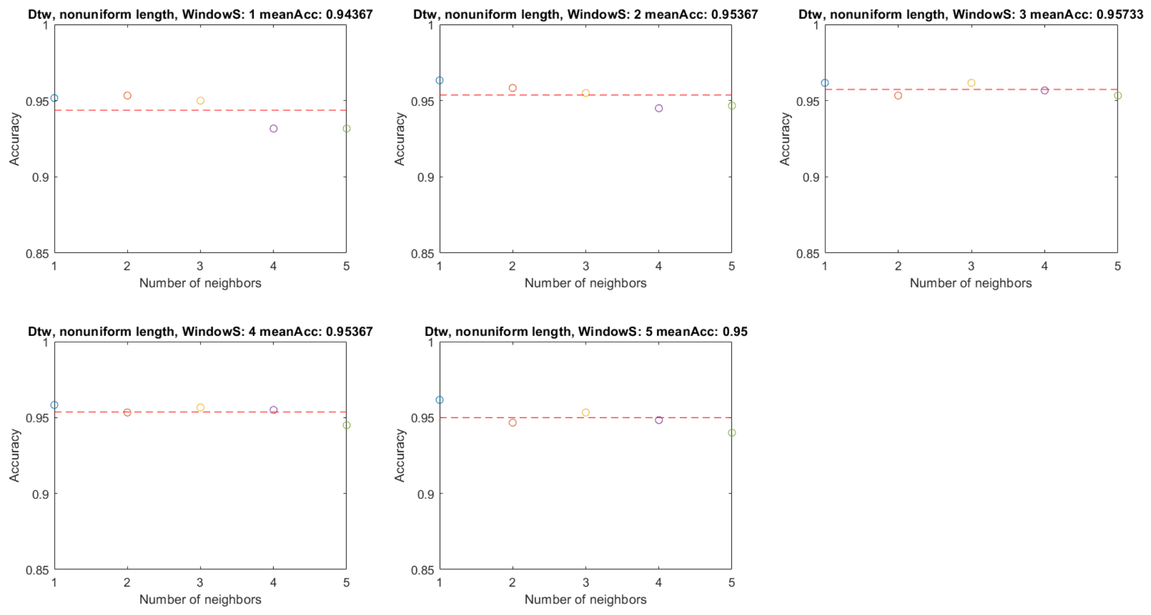

Figure 12: DTW, nonuniform length

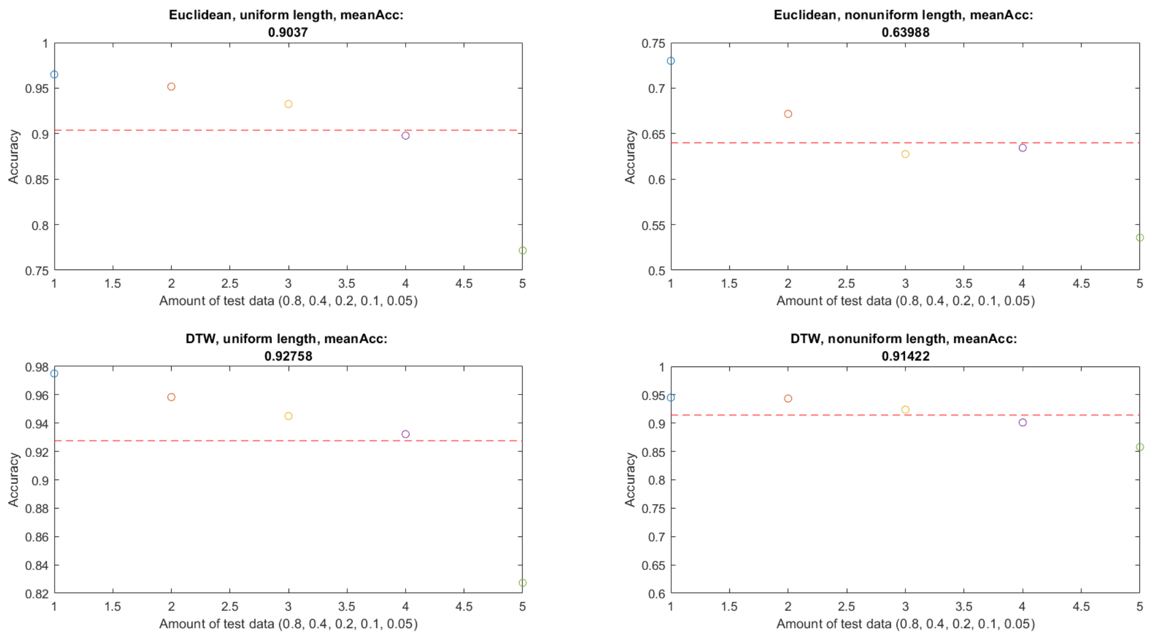

Figure 13: Lack of data modelling

7 Conclusion

During the project we have prepared the data for classification task, designed kNN classifier from scratch. Studied how data structure and its amount influences on kNN classifier with different distance measures (Euclidean and DTW). To sum up, we have implemented a relatively precise and accurate classifier which can perform well even with small number of train samples Fig. 14.

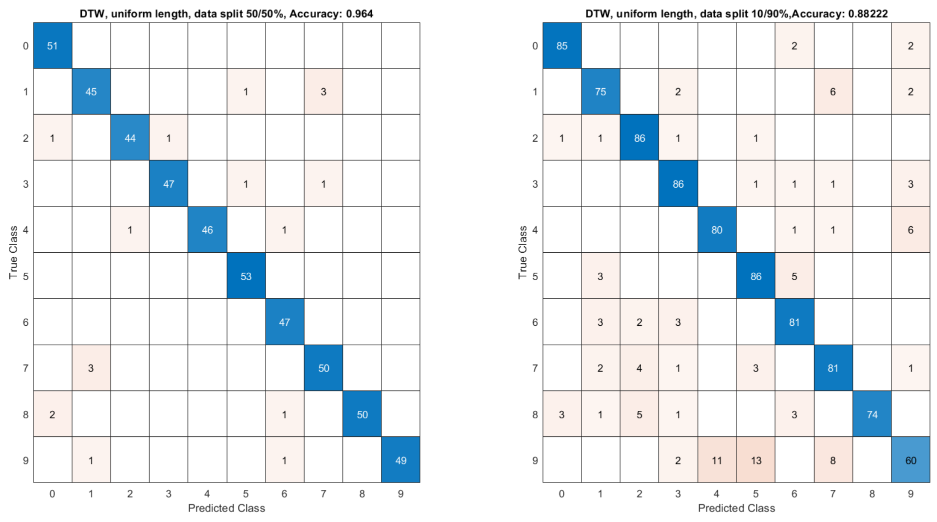

Figure 14: Confusion Matrix for DTW, uniform length

References

[1] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

[2] Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, 2013.