Time Elastic Similarity Spaces for Time Series Matching

Background

In general, similarity spaces are extracted embeddings from input data into a high-dimensional feature space where these data can easily be separated using modelling algorithms such as the popular euclidean distances. Nonetheless, when provided with sequence data such as time series data, we are faced with the challenge of the heterogeneity of the data. To solve this, we look for a new way of extracting the matching function using a dynamic similarity measure that is robust enough to extract the patterns in our data. In the literature, the time elastic similarity spaces have been proposed as a way of tackling this kind of sequence data using the Kernelized method. The Frechet's distance, Dynamic Time Warping (DTW), Levenshtein's distance, and the Kernelized DTW have been proposed as a dynamic way of extracting the similarity spaces.

In this study, we evaluated a set of distance metrics -- Frechet distance, DTW, Euclidean distance and the KDTW on a set of hand-written signatures obtained from the MOBISIQ database. The MOBISIQ database contains a set of 83 finger-drawn signatures with 20 expert forgeries. The main idea of this analysis was to develop a framework for verifying finger-drawn signatures based on this reference paper: https://www.hindawi.com/journals/misy/2018/3127042/fig2/.

1. **Levenshtein's distance:**

   In this section, we evaluated two sets of characters and their similarity based on the Levenshtein's distance. The *Levenshtein's* distance is also referred to as *edit distance (insertions, deletions, or substitutions)*, is a pairwise string alignments for measuring the difference between two sequences. Principally, it is the minimum number of single character edits required to change one word into another. In this session of the analysis, we created an alignment matrix that allows us to evaluate the pairwise Levenshtein's distance of two strings. **A =** *"aBigRedHat"* and **B =** *"aRedBigHat"*. The cost of the distances were set to: i.) deletion/ insertion = 1, substitution = 2, and match = 0.

   Results

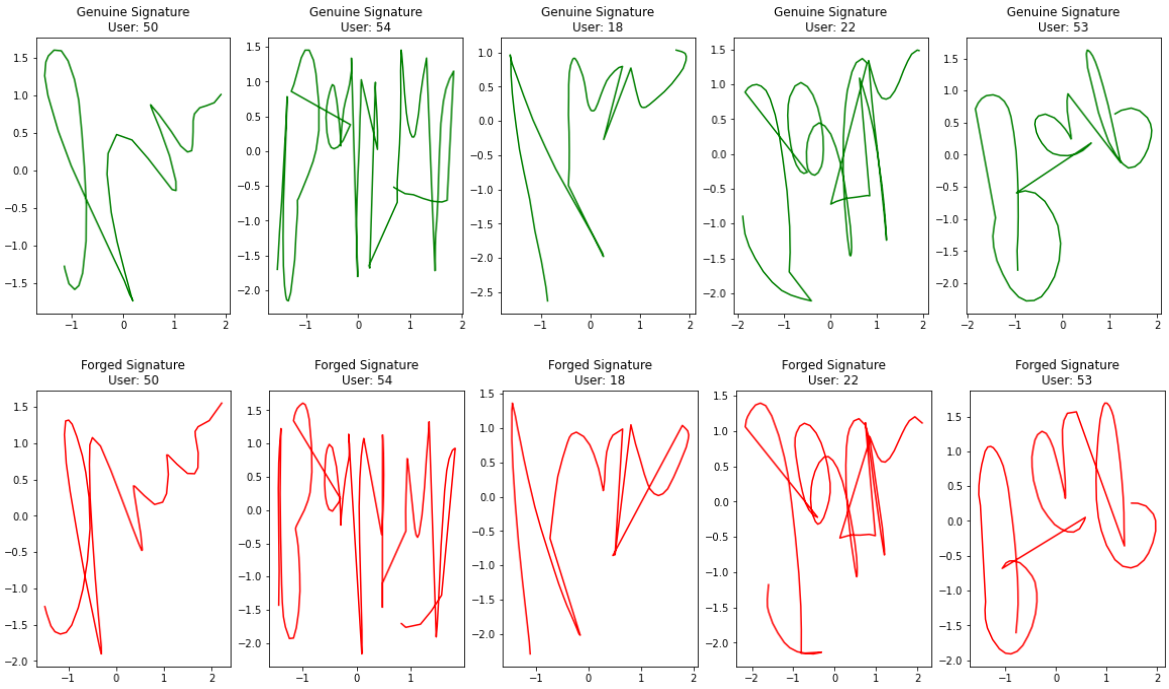
```
Levenshtein's distance:
 [[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
 [1. 2. 3. 4. 3. 4. 5. 6. 7. 8.]
 [2. 3. 4. 5. 4. 3. 4. 5. 6. 7.]
 [3. 4. 5. 6. 5. 4. 3. 4. 5. 6.]
 [4. 3. 4. 5. 6. 5. 4. 5. 6. 7.]
 [5. 4. 3. 4. 5. 6. 5. 6. 7. 8.]
 [6. 5. 4. 3. 4. 5. 6. 7. 8. 9.]
 [7. 6. 5. 4. 5. 6. 7. 6. 7. 8.]
 [8. 7. 6. 5. 6. 7. 8. 7. 6. 7.]
 [9. 8. 7. 6. 7. 8. 9. 8. 7. 6.]]
```

   Observations:

In this case, we observed very few deletions or insertions operations which are usually the case when given two sequences of data with varying length. In our case the length of both strings match. Additionally, the path of the lowest cost was observed along the diagonal of the newly computed similarity space. Overall, the Levenshtein's distance between these two strings is 6.

2. **Signature verification using multiple time-elastic distances and comparison of relative performance.**

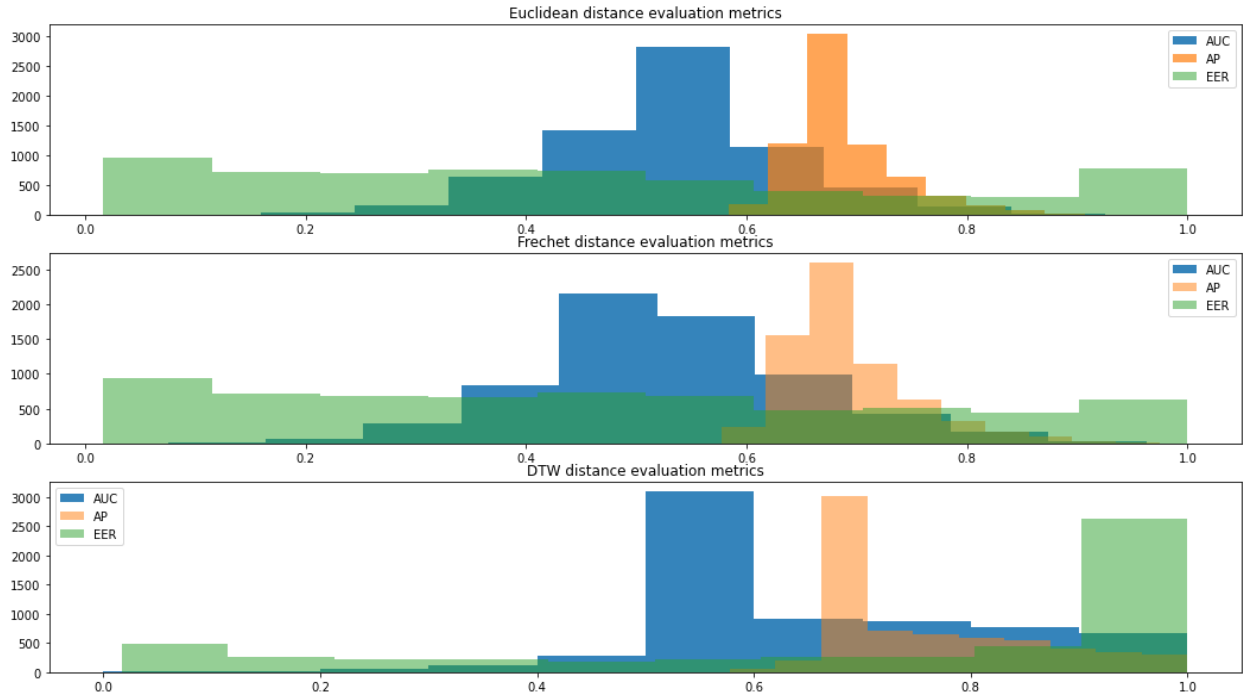
In this section, we evaluated different time-elastic distances and their relative performance in extracting and modelling features from the MOBISIQ database. In a classical machine learning method, the datasets were split into the training and test sets (in this case, the train set contains all the original signatures). Here, in an unsupervised learning manner, we extracted distance features from the X_train and X_test datasets together with the labels. An important step in the feature extraction was to aggregate the 5 genuine signature sets per user into an **average signature (used as the reference signature)** and compared this to the 60 signatures available in the X_test data, containing both the genuine signatures and forged signatures.



Display of the genuine and forged signatures

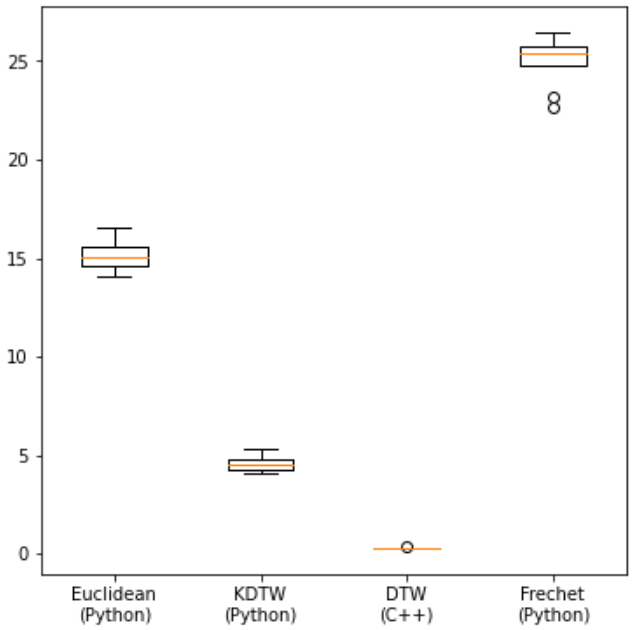
The *Euclidean distance (ED)*, the *DTW distance* and *the Fretchet's distance* were extracted and used as features for training the Nearest Neighbor classifiers. This trained model was evaluated on the 82 other signatures excluding the first user. Afterwards, three e**valuation metrics** were selected to evaluate the outcome of the different models (developed with different distance features) : i.) *Area Under the ROC Curve (AUC), ii.) Average Precision (AP)*

and iii.) *the Equal Error Rate (EER)* measures. The output of these evaluation metric per distance similarity measure is shown in the graphical representation below:

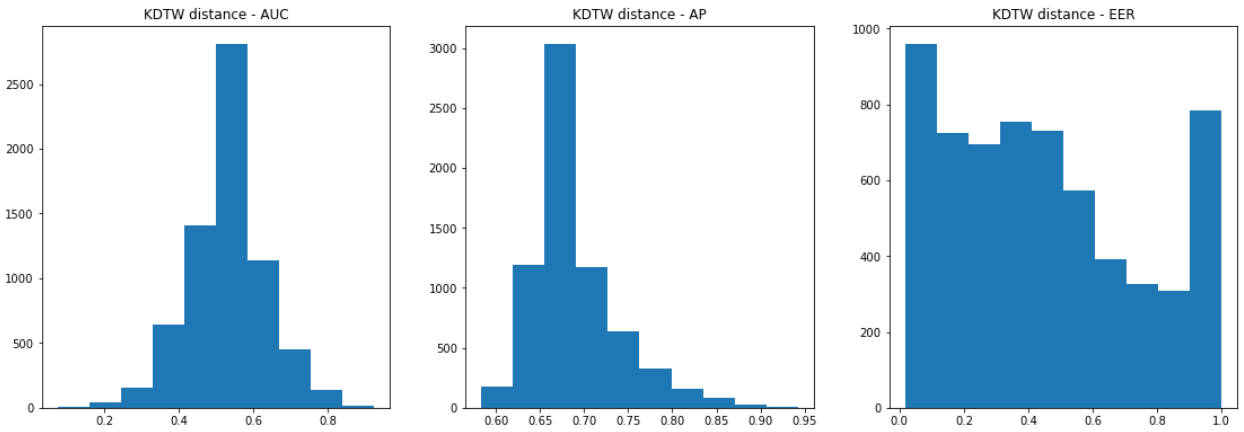


Result presentation of the output of the three evaluation metrics for comparing the distance similarity measures.

Additionally, the runtime of each of the distance similarities were computed and shown in a graphical representation. The frechet similarity metric appears to use the highest runtime for one user (the performance was assumed to be scaled to all the others, hence to save local processing time and resources, this evaluation was limited to just one user.).



3. **KDTW: Regularized Dynamic Time Warping Kernel.**
The optimal parameter (sigma) for the KDTW was obtained through the classical Machine Learning method of Cross validation. The best sigma value observed was 10, reaching a model accuracy of at least 91%. This sigma value was used to train the KDTW distance metric across the whole database of signatures and the performance was noted as shown graphically below.

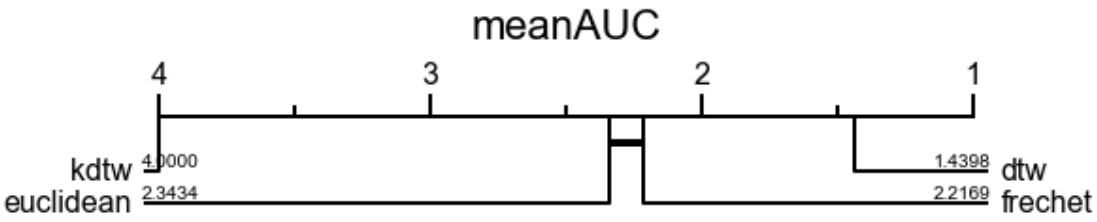


We also evaluated the processing time of the KDTW algorithm, but for one user. This time, the average computation time was about 6.5 seconds.

```
@perform_time
def kdtw_timer():
        path=  "C:\\Users\\rufai\\OneDrive\\Desktop\\School\\UBS\\Big
Data\\Data Mining\\Time Elastic Kernels\\MOBISIG"
    luid = [str(x) for x in pathlib.Path(path).glob("USER*")]
          loadallusers([luid[0]],  ntrain=5,  features=['x',  'y'],
dtfn='kdtw')
    return True

kdtw_timer()[1]
```

4. **Critical Difference Diagrams** (using the Method vs. TaskID)**.**
At this stage, we compared the different performances of the distance metrics using a critical difference diagram developed using the Wilcoxon-Holm method for detecting pairwise significance. We obtain a Critical Diagram of the mean Area Under the Curve.



This Critical Difference diagram shows that the DTW algorithm is the best algorithm over the range of all available data in the MOBISIG database. This method focuses on the relative performances of each of the distance metrics to each other in capturing forged signatures in the database. It is important to note, nonetheless, that in principle the KDTW algorithm has a faster implementation than the DTW algorithm but in this project (the C++ implementation of DTW was considered leading to much faster computation than KDTW).

5. **Proposition of a semi-supervised procedure (using only the training data) for tuning the sigma meta-parameter.**
   In learning problems, like the one presented in this project, where there are very little training sets, an alternative way for estimating optimal parameters for the model could be to use a strong statistical baseline like Bayesian learning or the Gaussian process for estimating the mean and covariance matrices that best captures the complexities and patterns in our data.