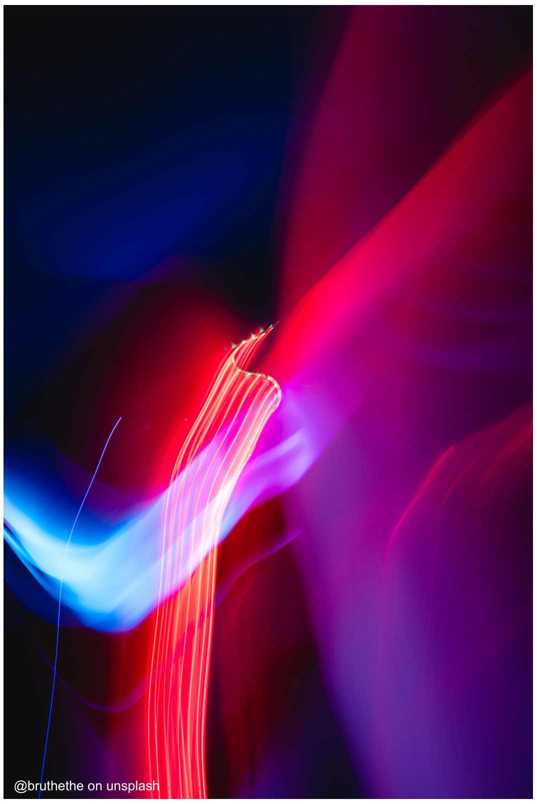
LINFO2275: Data mining and decision making

Implementation of a hand gesture recognition system



Group 9 :
CAPON Nathan (DATS)
HEROUFOSSE Gauthier (BIR)
SCHERPEREEL Colin (BIR)

Teaching :
Pr. SAERENS Marco
COURTAIN Sylvain
LELEUX Pierre

Academic year 2021-2022

1 Introduction

In recent years, gesture recognition has been widely used in human-computer interaction systems, prosthetic control, and game and virtual reality interfaces. As automation processes have become an essential need for the present world, the integration of smart interfaces relying on natural and intuitive sensors into the everyday life has seen a rapid increase (Huang et al. 2019). Gesture recognition systems are part of this set of people-centric applications that display a large potential of development in the next decades. In particular, the automotive industry is expected to play an important role in this growth, thanks to the potential benefits of gesture-controlled systems to improve user experience, to reduce driver workload, and to increase vehicle safety (Wendorf 2019). As gestures are an essential part of human communication, gesture-based systems turn out to be more intuitive than other kinds of interfaces. In particular, even a basic implementation relying on sketches and symbols (rapidly executed freehand drawings) can provide user-friendly interactions with a relatively high accuracy.

In this project, we develop a basic sketch recognition system in *Python*. In particular, we compare the performance of two different methods to reach this purpose: dynamic time warping and a support vector machine (SVM) implementation.

2 Methods

The data used to train and test the sketch recognition model consists in a part of the hand-gesture dataset, gathered by and described in (Huang et al. 2019). More precisely, we use the "domain 1" data corresponding to sketches of numbers ranging from 0 to 9, that are called categories or classes. These sketches are produced by 10 different users with 10 repetitions for each class. In total, 1000 sequences of the hand position vector $\mathbf{r}(t) = [x(t), y(t), z(t)]^T$ are therefore recorded as a function of time t.

2.1 Dynamic time warping

The first approach used to develop a sketch recognition system combines dynamic time warping (DTW) with a nearest-neighbors technique. On the one hand, DTW is used to compute a distance metric between a sequence of the testing set and reference sequences of the training set. On the other hand, the nearest-neighbors technique is used to retrieve the reference sequences that are the most similar to the testing one in order to identify to which category it belongs. The main benefit of DTW is that the distance metric it provides can account for time distortions and warping of the signal.

The DTW distance used in this recognition system is based on the following elements:

- A set of k multi-dimensional time series (MDTs) T_i, $i \in \{1, ..., k\}$ that correspond each to M individual time series Q_j, $j \in \{1, ..., M\}$ with n_j observations, $T = [Q_1 = (q_{1,1}, ..., q_{1,n_1}), ..., Q_M = (q_{M,1}, ..., q_{M,n_M})]$,

- When comparing two sequences Q_A and Q_B, an $(n_A \times n_B)$ matrix, whose entry (i, j) is the squared Euclidian distance $d(i, j) = (q_{A,i} - q_{B,j})^2$ between $q_{A,i}$ and $q_{B,j}$,

- A recursive function defining the cumulative distance $D(i, j)$ of the adjacent elements and that is used to represent the warping cost, *Note:* contrary to what was presented during the lectures, and based on (Shokoohi-Yekta et al. 2017), we don't add twice $d(i,j)$ when jumping from (i,j) to $(i-1,j-1)$.

$$\begin{cases} D(i,j) & = d(i,j) + min\{D(i-1,j-1), D(i-1,j), D(i,j-1)\}, \quad \begin{cases} i & = 1,...,n_A \\ j & = 1,...,n_B \end{cases} \\ D(1,1) & = d(1,1) \end{cases}$$

- A warping path $P = (p_1, p_2, ..., p_s)$ that is a set of contiguous matrix elements defining the DTW distance between Q_A and Q_B. The path relies on the cumulative distance $D(i,j)$ to minimize the total warping cost, $DTW(Q_A, Q_B) = \sqrt{D(n_A, n_B)}$,

  The warping path is also subject to several constraints:

  ➢ P must start and finish in diagonally opposite corner cells of the matrix,

  ➢ The steps in P are restricted to adjacent cells,

  ➢ The points in P must be monotonically spaced in time.

2.1.1 Multi-dimensional case

Generalizing the DTW approach to the multi-dimensional case is described in (Shokoohi-Yekta et al. 2017). Two different DTW metrics can be used to compare two multi-dimensional sequences $T_A = [Q_{A,1}, ..., Q_{A,M}]$ and $T_B = [Q_{B,1}, ..., Q_{B,M}]$:

- DTW$_I$ is the cumulative distance of all dimensions independently measured.
  If $DTW(Q_{A,m}, Q_{B,m})$ is the DTW distance of the m^{th} dimension of T_A and T_B, DTW$_I$ can be written as:

$$DTW_I(T_A, T_B) = \sum_{m=1}^{M} DTW(Q_{A,m}, Q_{B,m})$$

- DTW$_D$ is calculated in a similar way to DTW for 1D time series, except that $d(i,j)$ is redefined as the cumulative squared Euclidian distance of M data points. If $q_{A,i,m}$ is the i^{th} element of the m^{th} dimension of T_A and $q_{B,j,m}$ is the j^{th} element of the m^{th} dimension of T_B:

$$d(i,j) = \sum_{m=1}^{M} (q_{A,i,m} - q_{B,j,m})^2$$

In order to account for the fact that the users may draw the numbers in different ways (i.e., from top to bottom or from bottom to top), we compute both DTW$_I$ and DTW$_D$ twice for each comparison with a training sample (once forward in time and once backward in time). We only keep the best values (the minimal distances) as the final results of the comparison.

While these two approaches to adapt DTW to MDTs can produce significantly different results, there is no evidence that one surpasses the other in most cases (Shokoohi-Yekta et al. 2017). The performance associated to these two metrics are therefore compared as two distinct recognition models.

2.1.2 Euclidian distance

For comparison purpose, we also create a very basic recognition system based on the Euclidian distance between the testing and training MDTs. The Euclidian distance between the M-dimensional time series T_A and T_B, with respective lengths n_A and n_B writes:

$$ED(T_A, T_B) = \sum_{i=1}^{n} \sqrt{\sum_{m=1}^{M} (q_{A,i,m} - q_{B,i,m})^2}, \quad n = min\{n_A, n_B\}$$

2.1.3 K-Nearest neighbors

The last step of the approach is to use a K-nearest neighbors (KNN) algorithm to assign a category to the testing sample. Based on the previous distance metrics, we keep the K training sequences that show the lowest distance to a given testing sequence. We then compare the recognition performances reached when using two different criteria:

- The category assigned to the testing sample is always the one of the nearest neighbor (K=1).

- The testing sequence is assigned to the category of the majority among 3 nearest neighbors (K=3) if at least 2 of the 3 references share the same category. Otherwise, the category of the nearest neighbor is chosen.

2.2 Support Vector Machine

To find alternatives to DTW, we decided to try to implement an algorithm using a Support Vector Machine (SVM).

Support vector machines are known as supervised learning models that analyse data for classification or regression. They look for the best decision boundaries to maximise the width of the gap between classes. Once the model is fully trained, new examples are classified into the most probabilistic category. SVMs can also effectively perform non-linear classification by using a kernel to map the data into much higher dimensions, and have a better chance of separating them with a plane. We decided to try two strategies with an SVM: to train it either by using all the digits drawn by all the individuals or to train the model several times by separating the individuals.

2.2.1 Data preprocessing

First, to be able to train the classifier, we need to preprocess our data. We chose to split the 3D space of our data into subspaces associated with a number. This allows us to reduce the dimensional triplet sequence to a sequence of numbers.
Then, the sequences should be reduced to a fixed duration that is determined by the shortest sequence. Finally, the set of sequences provide us with the database to be used to train and test our SVM classifier.

2.2.2 Kernel choice

After the data preprocessing, we chose to try the first strategy (global classification) with a linear classifier. Because it didn't work, we had to use a kernel to access higher dimensions. We decided to compare the kernel types available in the *sklearn* library, i.e. either 'polynomial' or 'rbf' or 'sigmoid' and see which one obtains the best results. As the RBF kernel provided the best results for the first strategy, we used this kernel for the separation by individuals.

2.3 Validation

The classification accuracy of the different approaches are assessed by cross-validation with a leave-one-user-out procedure. Ten iterations are therefore completed by using each time 90% of the sketch data as our training set and 10% of the data as our testing set (corresponding to the 100 sequences of one user). The evaluation is therefore user-independent. For each of the above-mentioned approaches, an average accuracy and its standard deviation are computed from the 10 iterations. The accuracy of one iteration is defined as the ratio between the number of testing sequences that have been classified in the right category and the total number of testing samples given to the model. In addition, a confusion matrix showing the classes that are easily confused by the best model is provided.

3 Implementation

The implementation of the different recognition models is done in *Python*. Regarding the DTW approach, finding the best warping path is achieved by using dynamic programming. For the SVM training, we chose to use the *Python* package *sklearn*. More details can be found in the code itself, as many annotations and a complete description of each function, as well as of their inputs and outputs, are provided.

4 Results

4.1 Dynamic time warping

The performance of the first batch of recognition models based on dynamic time warping is presented in Table 1. The average accuracy and the standard deviation computed from the 10 iterations with the leave-one-user-out procedure are provided for 12 different situations. As explained in section 2.1.3, 2 nearest-neighbors criteria are compared in this table (K=1 and K=3). On the other hand, the 3 distance metrics (Euclidian distance, DTW$_I$ and DTW$_D$) are computed twice for each case: only comparing samples in a time-forward manner (F), and choosing the minimal distance between time-forward and time-backward comparisons (BF). We recall that the Euclidian distance is only computed to compare the performance of the DTW approaches with a very basic system.

First, it is clear that the performance of the DTW approaches are very disappointing. Not only does their accuracy not exceed 10.7%, which is a very poor score for a recognition system, but they also are outperformed by the very basic Euclidian distance. It may seem that the application of DTW to the multi-dimensional case is not to be recommended for this kind of hand gesture recognition system.

Ignoring the poor performance, we can observe that the two nearest-neighbors criteria provide very similar results. However, as the mean accuracy is higher in the (K=1) case and as the standard deviation is higher in the (K=3) case, we will focus on the results given with the first criterion. Assigning the category of the very nearest-neighbor to a testing sample thus happens to be more efficient overall.

Another thing to point out is that the implementation of a time-forward and time-backward comparison between the testing and the reference sequences brings a bit more performance. For instance, we can observe an increase of 11.4% on the mean accuracy for the Euclidian distance and + 10.3% with DTW$_D$. Regarding both DTW$_I$ and DTW$_D$, the BF case also reduces the standard deviation. Considering the computational cost that this measure takes (all calculations are doubled), the importance of its implementation might be discussed.

All in all, the comparison between DTW$_D$ and DTW$_i$ is also very clear. The former seems to outperform the latter in all cases presented in this table. In addition, as DTW$_I$ is more computationally consuming, DTW$_D$ happens to be far more efficient in this context.

Table 1: Comparison of the performance of the recognition models based on DTW

		Euclidian		DTW_I		DTW_D	
		F	BF	F	BF	F	BF
K=1	Average accuracy [%]	11.4	12.7	4.5	4.5	9.7	10.7
	Standard deviation [%]	4.61	5.59	4.39	4.39	4.45	3.87
K=3	Average accuracy [%]	11.1	12.6	4.3	4.2	9.3	9.2
	Standard deviation [%]	4.23	4.86	4.63	4.62	7.31	6.23

4.2 Support vector machine

The results for the first strategy are presented below. To determine the best kernel to classify our data, we proceeded to a Grid search with the different parameters of the SVC function. You can find more details about those on the sklearn library. It concludes to the following results :

Best parameters:

- Polynomial : [C=100, coef0=0, degree=2, gamma='scale']
- RBF : [C=50, coef0= 0.0, gamma='scale']
- Sigmoid : [C=100, gamma='scale', kernel='sigmoid']

Test accuracy:

- Polynomial : 67 %
- RBF : 75 %
- Sigmoid : 15 %

Remark : Not having strong knowledge of the kernel theory, we decided to restrict ourselves to the kernel types available in the *sklearn* library, i.e. either 'polynomial' or 'rbf' or 'sigmoid'.

We then tested the robustness of our model by computing the test accuracy using cross-validation with the RBF kernel. We obtained a test accuracy of **60 %**, which is probably due to the fact that each individual has a special way to draw a digit. Indeed, the strategy with separated individuals gave us better results, as shown below :

Table 2: Comparison of the performance of the strategies based on SVM

Individual	1	2	3	4	5	6	7	8	9	10
Test accuracy [%]	78	62	83	89	56	66	71	66	94	80

We observe that this method has better performance than the previous. However, this depends of the individual concerned but, in most of the case, this statement is true and allow us to say that the second strategy is more suitable for our task.

Finally, we can conclude by pointing out that our results with this second strategy are quite high, around **80 %** accuracy in average and so we can claim that SVM is a good model to classify our data.

5 Conclusion

In this project, we have tried different strategies to implement a gesture recognition system in *Python*. We used two different approaches based on dynamic programming and machine learning respectively. We built the Dynamic Time Warping model from scratch using literature resources. The accuracy of the classification was evaluated by cross-validation with a leave-one-user-out procedure. For this particular case, Dynamic Time Warping gave us poor results and does not seem to be an appropriate approach. This seems a bit surprising since many gesture recognition models using Dynamic Time Warping are found in the literature. On the other hand, the trained support vector machine model gave us relatively good results, with an accuracy of 80% for the best implementation. Although the results are not satisfactory for Dynamic Time Warping, there is still much research to be done in this area.

References

Huang, J., P. Jaiswal, and R. Rai (2019). "Gesture-based system for next generation natural and intuitive interfaces". In: *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 33.1, pp. 54–68. DOI: `10.1017/S0890060418000045`. URL: `https://www.cambridge.org/core/product/identifier/S0890060418000045/type/journal_article`.

Shokoohi-Yekta, M. et al. (2017). "Generalizing Dynamic Time Warping to the Multi-Dimensional Case Requires an Adaptive Approach". In: *Data mining and knowledge discovery* 31.1, pp. 1–31. DOI: `https://doi.org/10.1007/s10618-016-0455-0`. URL: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5668684/`.

sklearn (library). *Support Vector Machines*. URL: `https://scikit-learn.org/stable/modules/svm.html#svm-kernels`. (accessed: 16.05.2022).

Wendorf, M. (2019). *How Gesture Recognition Will Change Our Relationship With Tech Devices*. URL: `https://interestingengineering.com/how-gesture-recognition-will-change-our-relationship-with-tech-devices`.