LINFO2275 – Data mining and decision making

Gestures Recognition

Project 2 guidelines

Professor: Marco Saerens

Teaching Assistants: Diego Eloi & Nicolas Szelagowski

1 Objective

The objective of this project is to put into practice some of the techniques introduced in the data mining and decision making lectures. This is done through the study of a practical case which requires the use of a scientific programming language, or libraries, related to the statistical processing of data. More precisely, this work aims at applying data mining/statistical learning techniques for supervised classification of three-dimensional recorded hand gestures. This project will be carried out with the same groups as for the previous project (MDP).

2 Problem statement

You are a young data scientist hired by an IT company. Your first work is to implement a hand gesture recognition system for a smart user interface. A three-dimensional tracking of the position vector $\mathbf{r}(t) = [x(t), y(t), z(t)]^{\mathrm{T}}$ of the hand is recorded as a sequence of the three coordinates in function of the time t. Thus, the user is executing a sketch or a symbol with his hand, which is recorded and recognized by the system. Sketches and symbols are rapidly executed freehand drawings (Huang et al., 2019). Here, the sketches are representations of the numbers from 0 to 9 (10 classes or categories).

You are asked to develop a sketch recognition system identifying the category of the sequence, and implementing/testing it with the Python programming language.

3 Datasets

In order to train the recognition system, some publicly available gestures datasets, gathered by and described in (Huang et al., 2019), are used.

For the **first dataset** (dataset01; mandatory), ten users/subjects participated and were each asked to draw ten times the numbers $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ (called "domain 1" data in Huang et al., 2019) in 3D. We therefore have ten repetitions of each of these ten numbers by ten subjects, that is, 1000 sequences of 3D coordinates in total.

For the **second dataset** (dataset02 – called domain 4 in Huang et al., 2019; analyse this dataset only if you have time), each of the ten subjects was asked to draw ten repetitions of three-dimensional figures, like a pyramid for instance (see Huang et al., 2019, for details). Note that the same data collection still contains other drawing types; if you have the time, feel free to investigate these other data sets (as a bonus). All the data are available on internet (on Moodle, only domains 1 and 4 are available).

Note that the second dataset is more difficult to classify than the first one because the figures are really 3-dimensional; you therefore have to start with the first one.

4 Implementation of the models

The project is very open. Notice that you will first have to perform a preliminary *exploratory analysis* of the data as well as some *pre-processing* steps (like, for instance, standardize the signals, PCA of the

gesture signals, etc) in order to facilitate the classification. Note also that we will not use the precise recorded time t of the 3D observations; we consider that the time steps are constant ($t = 1, 2, 3, \dots$) even if this hypothesis is not completely true.

4.1 Baseline methods

You should then implement at least one baseline technique among the *dynamic time warping* (Rabiner et al., 1993; Theodoridis et al., 2009), the *edit-distance* or the *longest common subsequence* (Theodoridis et al., 2009) after having performed a vector quantization (clustering) of the signals for these two last methods. Then, you simply rely on a nearest-neighbor technique. You must implement yourself at least one of these baselines in Python 3 and obtain baseline results on the datasets (you are not allowed to use a library implementing these algorithms, but you are free to use libraries facilitating scientific computation like numpy, etc).

Dynamic time warping can work directly using the 3D coordinates of the signal, but the edit distance, as well as the longest common subsequence, are designed in order to compare strings (sequences of characters). Therefore, to be able to compute the edit-distance, you need to convert your signals into a string. This can be done, for instance, using a clustering technique. More precisely, this is done by gathering every 3D coordinate of every recorded gesture of the training set in one big set of 3D coordinates, on which you are going to apply a clustering (such as a k-means[1]) and record the coordinates of the centroid of each cluster. Then, for each gesture, you will take each spatial 3D point and replace it with the label of the closest centroid. As such, instead of having a series of 3-dimensional spatial coordinates, you will now have a sequence of labels representing the clusters (for example: AAACCCBAA...) for each gesture. You can then use these sequences of labels to compare gestures to each other using the edit distance or the longest common subsequence.

Note that the most rigorous method would require you to perform the clustering inside the cross-validation only on the training set, and then use the centroids for both the training and test set, however, for simplicity purposes, we allow that you just make the clustering once, thus turning all your gestures into strings once, before performing the cross-validation.

4.2 More state-of-the art methods

Finally, you are free to use and test at least one other, more state-of-the-art, method found in the literature, like the $1 recognizer (see the references below – the paper is provided on Moodle), hidden Markov models (Fink, 2014), deep learning techniques (Goodfellow et al., 2016), or any other technique you are interested in. For this second part, you are allowed to use existing implementations/libraries in Python and, of course, refer to them in the report. The idea is to compare the results obtained by this new technique to the baseline technique.

Another interesting possibility would be to perform a multidimensional scaling, or learn a representation, from the pairwise distances (or similarities) between gestures of the training set and then train a logistic regression, a NN, or a SVM on the gesture representations. Then, apply this model on the test set for out-of-training-sample validation. Still another possibility is simply to extract meaningful features from the 3D gesture signals and then rely on a simple supervised classification model.

5 Validation and comparison tests

You will evaluate the classification accuracy of your model by cross-validation on two different task settings: *user-independent* and *user-dependent* gesture identification.

- **user-independent** setting. Here, for the cross-validation, you have to use a leave-one-user-out procedure by using 90% of the data from 9 users for fitting the model (training set), and then testing the model on the remaining 10% data (test set) provided by the remaining user who was

[1] Feel free to use an existing library for performing the clustering.

removed from the training set (there are ten users in total). The same procedure is repeated 10 times by holding all the data of each user in turn for testing (first remove user 1 from the training set and use this user 1 for testing, then remove user 2, etc). Therefore, the evaluation is user-independent because it quantifies the ability of identifying the gestures of a new user who does not appear in the training set. For each compared method, you should record the different accuracies for the different users and then compute the average accuracy and its standard deviation.

- **user-dependent** setting. In the same spirit, you now use a leave-one-gesture-sample-out procedure by using 90% of the data from the 10 users. That is, for each user, we hold out one example of each of the 10 gestures of this user as test fold. This is repeated ten times in a cross-validation setting. It ensures that, for each user, 9 realizations of each of the gestures are used in order to fit the model based on the training set. The performance is then evaluated on test folds each containing only one sample per type of gesture (class) and per user.

In order to determine which class to assign to a given gesture in the test set for the *baseline* methods, you should make a comparison between this gesture and all (or part of if too time-consuming) the gestures present in the training set (user-independent case), or make a comparison with the gestures of the same user from the training set only (user-dependent case). Then, find which ones are the gestures that are the closest matches (k-nearest-neighbors classifier) to the new gesture you want to classify. This must be repeated for all train/test folds in the cross-validation.

We expect that the results will be more accurate in the user-dependent setting. Overall, you should compare the different investigated methods according to their average accuracy and standard deviation in both settings and on both datasets. It is also interesting to provide a confusion matrix showing the gestures (classes) that are easily confused by the best model.

6 Report

Please do not forget to mention your affiliation (SINF, INFO, MAP, STAT, BIR, DATS, DATE, etc) on the cover page, together with your name and group number. You are asked to write a report (PDF) in English of maximum 8 pages (everything included, but some additional, less important, content could be reported in an appendix). This report must have a professional look & feel, like a scientific or a technical report. Therefore, your report should integrate plots, equations, etc, but no screenshot (image capture) of equations and/or screenshot of code outputs. Do not forget to provide references for your sources of information[2] and be consistent in your notations.

Your report must at least contain:

- a short introduction;
- a brief theoretical explanation (with main equations) of the investigated methods;
- a short description of your implementation[3] and methodology;
- a description and a discussion of the comparison results;
- a conclusion.

Your project will be evaluated on the following aspects:

- the quality of your report and code;
- the amount of experimental work (number of datasets and techniques investigated, including potential additional experiments);

[2]Some papers are available on Moodle.

[3]Do not go into details about your code itself. If you still wish to provide details about the code and the functions of your implementation, do not hesitate to integrate a ReadMe file in your zip, but your report should not focus on the code.

- the relevance and description of the investigated methods.

This report must be uploaded before May 18, 2025, 23:59, together with the code (all files zipped together) on Moodle in the section "Assignments". Do not forget to comment your code. This second project accounts for 5 points in the final grade of the course. Your project can be handed behind schedule, but your group will get a penalty of -0.5 (on 5) for each day late (starting at May 18, 2025, midnight. However, we do not accept delays larger than 3 days.

Finally, we recall that a forum, dedicated to mutual assistance in the projects, has been set up on Moodle, on which you can ask questions and answer to questions posted by the other groups.

Some references

- Deisenroth, Aldo Faisal & Cheng Soon Ong (2020) "Mathematics for machine learning". Cambridge University Press.
- Fink, Gernot (2014) "Markov models for pattern recognition". Springer.
- Garcia-Diez, Fouss, Shimbo & Saerens, M. (2011). "A sum-over-paths extension of edit distances accounting for all sequence alignments". Pattern Recognition, 44 (6), pp. 1172-1182.
- Goodfellow, Ian, Bengio, Yoshua & Courville, Aaron (2016). "Deep learning". MIT Press (available at http://www.deeplearningbook.org).
- Huang, Jaiswal & Rai (2019). "Gesture-based system for next generation natural and intuitive interfaces". Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 33 (1), pp. 54-68.
- Rabiner & Juang (1993). "Fundamentals of speech recognition". Prentice-Hall.
- Theodoridis & Koutroumbas (2009). "Pattern recognition, 4th ed". Academic Press.
- Wobbrock, Wilson & Li (2007). "Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes". In Proceedings of the 20th annual ACM symposium on User Interface Software and Technology, pp. 159-168.

Good work !