Practical block III: Human Behavior Analysis

Object Recognition

University of Barcelona

May 15, 2016

Authors:
Camps, Julià

Contents

1 Introduction

In this practical exercise we will deal with experimenting with several of the characteristics of the Dynamic Time Warping method for gesture recognition on the Microsoft Research Cambridge-12 Kinect^{TM} gesture data set on pairs of images.

The aim for this exercise is to implement and test the DTW method for computing the gesture recognition on the mentioned dataset.

We dispose of a gestures dataset, with the coordinates of different key points of the human skeleton representation. By means of computing the Euclidean distance between different compositions of this positions, we are able to obtain a cost measure of similarity between two poses, and, since each gesture is composed by a sequence of poses, we have been able to successfully apply DTW as a distance measure for computing a k-NN over a reduced representation of the dataset.

2 Description of the DTW method

Dynamic time warping (DTW) is an algorithm for measuring similarity between two temporal sequences which may vary in time or speed. For instance, similarities in walking patterns could be detected using DTW, even if one person was walking faster than the other, or if there were accelerations and decelerations during the course of an observation.

Since, the DTW method is an algorithm for measuring similarity between two sequences considering variations in time or speed, with the aim to find the optimal match between them. We have implemented the method in order to classify gestures by the similarity of the movements of different parts of the human body.

In order to apply this algorithm there are some requirements:

- The data, must be representable in sequences of elements.
- A similarity/distance measure is needed, in order to compute the cost needed in order to consider each element of one sequence to be corresponding to a certain element of the other sequence.

2.1 The Euclidean distance

For the Matlab implementation, the distance measure used was the Euclidean distance. In order to use the Euclidean distance, since, a sequence is a set of frames, and each frame contains a set of points. Since, the euclidean distance has to be calculated between two points, the total distance between two frames, was the sum of all the Euclidean distances computed for each pair of the same key-point in both frames.

Thus, we can define formally the similarity distance between two frames as:

$$dist(f_1, f_2) = \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{m} (x_{ij} - y_{ij})^2} \tag{1}$$

where, $f1$ and f_2 are frames (or poses) in two different sequences, and the function *dist* will return the cost for being f_2 a transformation of f_1. Where, n is the number of key-points composing a pose (or frame) in any of the given sequences, m is the number of dimensions of a given equivalent pair of key-points x_i and y_i (each from one the the compared frames f_1 and f_2, respectively).

2.2 The DTW algorithm implementation

In the following part of this section the steps of the DTW algorithm are presented.

1 Create a matrix M size $n \times m$, where n is the length of the first sequence, and m is the length of the second one.

2 Initialise all the matrix M values to infinity (or any unreasonably large number).

3 Fill up the matrix with the distances required to reach each transformation state. This step is performed according to the following:

$$M_{i,j} = cost + min(M_{i-1,j}, M_{i,j-1}, M_{i-1,j-1})$$

where each of the elements in the minimum operator correspond to the following cases respectively: insertion, deletion and match.

4 The cost corresponds to the similarity function, by adding this value, we are accumulating the transformation effort (or cost needed for assuming some transformation) through the matrix M.

5 Finally, at the last cell of the matrix (i.e. cell $M_{n,m}$) we obtain the cost for assuming the complete transformation of both sequences to be the same.

3 The k-NN with DTW gestures classifier

In this section the code implemented for applying the DTW as the similarity measure, on an k-NN algorithm, for gesture classification is commented.

The method implemented, for the described purpose, was composed by the following steps:

1 Load all the data for the experiments (see section 4).

2 Format the data as a list of "sequences" (we need the sequence representation in order to apply the DTW algorithm), so that an element of the sequence is a full frame of the gesture, and that each sequence contains a gesture, itself.

3 Perform the cross-validation over the full set of instances, but at each iteration separating the testing part from the training, in order to fill the confusion matrix with the testing results.

This process is repeated for each different gestures type, in order to obtain results of when trying to classify any of the gestures, appearing in the dataset reviewed, for this practical work.

4 Experiments description

In this section we will show the configurations used for the experiments, which are also discussed. The results obtained are presented and discussed in section 5.

In order to classify the gestures it was decided to load a set of different gestures and test the k-NN implementation presented in section 3 using cross-validation with the leave-one-out[1] strategy.

The data used for the experiments is described in the following table:

Data	Gesture name	# Samples
P1_1_1_p19	Crouch or hide	10
P1_1_2_p06	Shoot a pistol	10
P1_1_3_p19	Throw an object	10
P1_1_4_p06	Change weapon	10
P1_1_5_p19	Kick	10
P1_1_6_p06	Put on night vision goggles	10
P1_1_7_p22	Start Music/Raise Volume (of music)	10
P1_1_8_p06	Navigate to next menu	10
P1_1_9_p21	Wind up the music	10
P1_1_10_p19	Take a Bow to end music session	10
P1_1_11_p21	Protest the music	10
P1_1_12_p19	Move up the tempo of the song	10

Table 1: Different parameters configurations for the experiments.

In table 1 the different configurations for the experiments of this report are exposed. I decided to use a representative set of each different type of gestures in order to be considering the full space, and avoid loosing interesting gestures comparisons. However, as can be noticed, not all the full dataset was used for the experiments. This simplification was needed due to hardware restrictions of the memory available in the machine used for testing the experiments.

For the k-NN parameters, it was decided to perform two different experiments:

1 **k = 1:** In order to find possible elements that even, being mainly similar with the elements of their class, in average, they had a strong relation with some element of a different class.

2 **k = 3:** In order to find outliers, since this time a misclassified element should have at least the 2 nearest elements being from a different class, rather than only one.

[1]Leave-p-out cross-validation (LpO CV) involves using p observations as the validation set and the remaining observations as the training set. This is repeated on all ways to cut the original sample on a validation set of p observations and a training set. Leave-one-out cross-validation (LOOCV) is a particular case of leave-p-out cross-validation with p = 1. It is also called the Jackknife.

5 Discussion on the results

From the experiments results we can conclude that the method for gesture classification implemented, performs very well with the used dataset, as can bee seen in the confusion matrix shown in table 2 and table 3.

Classification Results

Real Gestures	1	2	3	4	5	6	7	8	9	10	11	12
1	9									1		
2		9		1								
3			11		1							
4		1		9								
5					10							
6						9		1				
7							11					
8						1		9				
9									9		1	
10			1							8		
11									1		11	
12					2						1	7

Table 2: Confusion matrix obtained for the k-NN tests using leave-one-out strategy, for $k = 1$.

In table 2 we can observe the confusion matrix obtained when testing the k-NN over the gestures dataset, using $k = 1$ nearest neighbours. This test by itself shows how good is the model, however, we expect that some misclassifications occur, since we are just using one-nearest-neighbour, and this strategy, usually implies misclassifications with elements placed in the border of a class, or even, being misclassified due to outliers of different classes. Therefore, it was decided to perform another test with $k = 3$, and by analysing the differences of the misclassifications among both methods, we will be able to distinguish the real outliers from the elements having an outlier very close to them from a different class. Although this two experiments will benefit from each other in order to explain how is the data being tested and the model built, it should be noticed that if there exists intersection between classes, we will not be able to handle this problem by means of k-NN with the proposed representation and similarity method.

The total accuracy retrieved using $k = 1$ was of 90.32%, having the worst represented class being class 12, which retrieved only 70% of accuracy.

	Classification Results											
Real Gestures	1	2	3	4	5	6	7	8	9	10	11	12
1	9									1		
2		9		1								
3			11		1							
4		1		9								
5					10							
6		1				9						
7							11					
8		1						9				
9									9		1	
10			1							8		
11									1		11	
12			1		2						1	6

Table 3: Confusion matrix obtained for the k-NN tests using leave-one-out strategy, for $k = 3$.

In table 3 we can observe the confusion matrix obtained when testing the k-NN over the gestures dataset, using $k = 3$ nearest neighbours. From this table, moreover than only observing the proposed method accuracy, which is shown in table **??**, it is interesting to notice that those misclassified elements, if all classes are enough different in order to ensure that this method should perform with 100% accuracy, are indicating outlier elements of the subset. Since, almost all elements of every class have been correctly classified, and therefore, we are likely to have all classes "well enough" represented (although we doubt that class 12 is well represented, since has only 60% of accuracy).

The total accuracy retrieved using $k = 3$ was of 89.52%. This result was totally unexpected. Since, it was expected to improve the accuracy, using more neighbours, therefore the only possible consequence of this problem is that some of the classes might be not well enough represented, or even in some cases, there must exist some overlapping between different classes. We believe that this problem could be fixed, by means of using more data, however, as we already stated, it was not possible to consider using the full dataset, due to the lack of more computational resources.

6 Conclusions and Future work

From this exercise we can conclude that the DTW is a very useful method for comparing data represented as sequences, and that could be used for unsupervised or semi-supervised methods, since does not requires to have the the testing sequences defined, it is able to find the sub-sequence, in the full data, which is more likely to contain the gesture searched.
The k-NN implementation using the DTW showed great results for the gestures classification problem, with the tested dataset. However, we noticed that this solution did not scale appropriately, due to the enormous number of operations needed to build the distances matrix.
Here are presented some possible improvements in order to handle the performance limitations of this approach:

- A good option for improving the performance of this method, could be considering using clustering over the training set, in order to reduce the search space. In order to explore these possibilities it would be interesting to start with a simple approach, using k-means over the training set, for building a smaller model for applying the k-NN with DTW.
- Another possibility could be performing an heuristic exploration search over the matrix, instead of generating the full matrix. Or exploring several clusters at the same time and give priority to the ones reaching better results at the first iterations using a threshold for deciding if a sequence belong to a certain cluster, without the need of exploring all clusters to the end.

Although, we haven not studied yet any algorithm that can deal with this problem in a straightforward notion, while preserving interesting scalability properties. The reviewed techniques in this exercise showed very good results, it is just the performance issue problem, that has been commented on the first part of this section. The solution to this problem can be also achieved by applying probabilistic graphs based techniques, which solve the scalability problem, by approximating the result instead of ensuring the optimal transformation relation given a set of sequences.