Master Thesis

nicoletta.farabullini

October 2021

This work is a development of a previous master project. What they did there was to analyze body movements in videos and compare a database based on a query. The analysis of each video is divided into frames, each frame includes a certain body position. Individual body poses are extracted from each position and the angles between joints are calculated, e.g. from neck to left shoulder. The output is a vector of angle for each frame, frames are then compared against the query by mean of cosine similarity.

1 Problems

The analysis of these videos is done by iterating over each frame for all videos. This is not an optimized method as it can become quite expensive with a large database. Additionally, we would like to provide a UI with a ranking of the best k videos in a efficient and fast manner.

1.1 Early analysis

Two techniques were initially investigated: LB_Keough and LCSS model. The former was chosen as it provides an accurate ranking. The paper relating to the second method states that "Even though we may miss the best match to our query, we are going to find a close match in less time". This is not optimal for our goals as we want an accurate ranking. Additionally, the LB_Keough method shows promising outcome in terms of analysis of very large datasets.

1.2 LB_Keough

LB_Keough performs early data pruning by calculating a lower bound distance and then computes the exact DTW distance if data passes the "best so far" condition. The early pruning is done by enveloping the query with using the Sakoe-Chiba distance and divide this envelope into different Minimum Binding Rectangles (MBRs). MBRs contain are built based on the maximum value in the upper envelope and the minimum in the lower for a chunk of the data set. The distance between these query blocks and the candidate series is then calculated. If this distance is less than the best-so-far condition, the DTW

distance is calculated. This technique is applied to time series and makes use of the Euclidean distance.

Question 1 from Sven: how do we incorporate angle similarity in Euclidean distance?

Question 2 from Sven: Each video will consist of a set of vectors. How do we move from comparing two individual vectors to a multi-dimensional case?

Question from Alex and Renato: how long does it take now to analyse the videos in the DB with the current implementation fo teh analysis and how does it compare with this new approach?

1.3 Question 1 from Sven - proposed solution

Cosine similarity is defined as such:

$$cosSim = \frac{\vec{x} \cdot \vec{y}}{\mid \vec{x} \mid\mid \vec{y} \mid} = \frac{\sum_i \vec{x_i} \cdot \sum_i \vec{y_i}}{\sqrt{\sum_i \vec{x_i^2}}\sqrt{\sum_i \vec{y_i^2}}} \tag{1}$$

If we look at the squared Euclidean distance:

$$ed(x,y)^2 = (\sqrt{(x-y)^2})^2 = (x-y)^2 = (\sum_i (x_i - y_i))^2 = \sum_i x_i^2 + \sum_i y_i^2 - \sum_i 2 * x_i \cdot y_i \tag{2}$$

If we now do the same process but with x and y being divided by their respective sums:

$$ed(\hat{x},\hat{y})^2 = \sum_i (\frac{x_i}{\sqrt{\sum_i x_i^2}} - \frac{y_i}{\sqrt{\sum_i y_i^2}})^2 \tag{3}$$

Folding this out:

$$ed(\hat{x},\hat{y})^2 = \sum_i (\frac{x_i}{\sqrt{\sum_i x_i^2}})^2 + \sum_i (\frac{y_i}{\sqrt{\sum_i y_i^2}})^2 - \sum_i \frac{2 * x_i \cdot y_i}{\sqrt{\sum_i x_i^2} * \sqrt{\sum_i y_i^2}} \tag{4}$$

Repositioning summations:

$$ed(\hat{x},\hat{y})^2 = \frac{\sum_i x_i^2}{\sum_i x_i^2} + \frac{\sum_i y_i^2}{\sum_i y_i^2} - 2 * \frac{\sum_i x_i \cdot y_i}{\sqrt{\sum_i x_i^2} * \sqrt{\sum_i y_i^2}} \tag{5}$$

The first two components of the sum in Equation(5) can be simplified to be one:

$$ed(\hat{x},\hat{y})^2 = 2 - 2 * \frac{\sum_i x_i \cdot y_i}{\sqrt{\sum_i x_i^2} * \sqrt{\sum_i y_i^2}} = 2 * (1 - \frac{\sum_i x_i \cdot y_i}{\sqrt{\sum_i x_i^2} * \sqrt{\sum_i y_i^2}}) \tag{6}$$

Equation(6) is directly proportional to Equation(1):

$$ed(\hat{x},\hat{y})^2 = 2 * (1 - cosSim(x,y)) \tag{7}$$

In other words, the Euclidean distance can be expressed in terms of cosine similarity if the two vectors are normalized to unit length.

But what does this mean when it comes to angle vectors? Potential solution: how about we switch from polar to Cartesian coordinates? In our case, x and y would be angles, e.g. ϕ with respective Cartesian coordinates z and w:

$$z = r * cos(\phi) \tag{8}$$

$$r = \sqrt{z^2 + w^2} \tag{9}$$

$$\phi = cos^{-1} * \frac{z}{\sqrt{z^2 + w^2}} \tag{10}$$

This last Equation can be plugged into Equation(6).

1.4 Question from Alex - resolved

Still working on setting up the MSc project on my computer. I am running into some errors due to my OS system (nothing wrong with the project). I will try to install it on my other computer which runs on Ubuntu.

Update: I succeed to install on my Ubuntu machine. I ran a sample script with the main functions without the usage of the UI. for 85 short videos it takes approximately 10 seconds.

1.5 Question 2 from Sven

The multi-dimensional approach in the paper merges multiple MBRs and extends the dimension of the blocks from 2D to 3D (this is the high-level idea). Can this be applied in our case? I am working on this.

Alternatively, we could adapt some version of the LB_Keough from the paper:

Algorithm 1 LB_Keough

```
best_so_far = infinity
for i in all vectors in single video do
  LB_dist = LB_Keough_dist(C[i], Q)
  if LB_dist < best_so_far then
    true_dist = DTW(C[i], Q)
    if true_dist < best_so_far then
      best_so_far = true_dist
      index_of_best_match = i
    end if
  end if
end for
```

We use LB_Keough for individual vectors in a video, then we loop over all of the videos. We save each bsf and index for each video in a list. We sort the list and see which indexes correspond to the best matches.