LINFO2275 : project 2

Hand gesture recognition

Rousseau Mathieu, 67001800

UCLouvain
Belgium
21/05/2023

1 Introduction

We want to implement a hand gesture recognition system for a smart user interface. A 3D tracking of the position vector $\tilde{\mathbf{r}}(t) = (x(t), y(t), z(t))^T$ of the hand is recorded as a sequence of the 3 coordinates in function of the time t[1]. The user is executing a sketch that is recorded and recognized by the system.

The system will be trained in order to recognize two different kind of sketches.

A first dataset contains the records of the numbers $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$ in three dimensions. We asked 10 users to draw 10 times each of these numbers. We have therefore a total of 1000 sequences.

A second dataset contains the records of three-dimensional figures. As for the precedent dataset, 10 users have drawn each figure 10 times.

2 Preprocessing

We first preprocessed the data. We embedded the datas in a list of 1000 entries. Each entry consist in a matrix of dimension (*n_ timepoints*, *vec_ dimensions*) where *n_ timepoints* is the length of the time series and *vec_ dimensions* is the dimension of the position vector which is 3 in our case. Notice that each time series has different length, that's important for what is to follow. After that, we compressed the time series to improve the computation time of the algorithms. To do that, we performed a vector quantization by applying a k-means algorithm where each data points is replaced with its closest centroid where the number of centroids correspond to a fraction of the time series length. We decided to reduce the length by half.

3 Dynamic time warping

3.1 Introduction

Given a time series of three-dimensional unknown vector data points, we would like to predict the sketch represented by these points. In order to do this, we used a K-Nearest Neighbors approach where we find the k most similar time series in the dataset and use the most represented target value among the k time series and use it as the prediction output.

To find the k most similar time series, we use a distance function as the similarity measure. However, we cannot use the often used euclidean distance as it produces pessimistic results when dealing with time series of different lengths.

Dynamic time warping - a often used technique in speech recognition problem - is a solution to this problem, it finds the optimum alignment between two time series by "warping" non-linearly a time series stretching or shrinking its time axis. Once we found the optimal alignment, we can determine the similarity between these twos.

Let $s = (s_1, \dots, s_n)$ and $t = (t_1, \dots, t_m)$ be two time series of length n and m respectively where s_i, t_j are the position vector $\in \mathbb{R}^3$ for $i = 1, \dots, n$ and $j = 1, \dots, m$. We define a warping of order $n \times m$ as a sequence of k points $x = (x_1, \dots, x_k)$ where $x_l = (i_l, j_l) \in [1, n] \times [1, m]$. Each points x_l of the path aligns the vector s_{i_l} with the vector t_{i_l}.

The following constraints apply,

- **boundary conditions** : $x_1 = (1, 1)$ and $x_k = (n, m)$. This ensures that every index of the time series are used in the warp path.

1. Note that we will not consider the precise recorded time of the observations.

- **locality** : $x_{l+1} - x_l \in \{ (1,0), (1,1), (0,1) \}, \quad \forall l \in [1, k-1]$.

The locality condition implies a monotonicity condition : $s_1 \leq \dots \leq s_n$ and $t_1 \leq \dots s_m$. This monotonicity implies that the lines representing the warp path do not overlap.

Among all the warping path we can construct, want to find the optimal warping path x^* that consists in the warping path that minimizes the square root of the following cost given for any warping path x,

$$C_x(s,t) = \sum_{l=0}^{k} ||s_{i_l} - t_{j_l}||_2^2 \tag{1}$$

We can then compute the DTW distance by computing the total cost of the warping path x^*.

In practice, to find the optimal path, we use a dynamic programming approach which consists in the following algorithm,

1. We define a distance matrix $D \in \mathbb{R}^{n \times m}$ where $D_{i,j}$ is the minimum-distance for the warping path between $s' = (s_1, \dots, s_i)$ and $t' = (t_1, \dots, t_j)$.
2. Each entry is computed by using the following recursion formula,

   $$D_{i,j} = \sqrt{||s_i - r_j||} - \min\left(D_{i-1,j}, D_{i-1,j-1}, D_{i,j-1}\right) \tag{2}$$

   with the initial condition $D_{1,1} = 0$
3. The entry $D(n,m)$ contains the minimum-distance for the warping path between s and t.

This algorithm has $\mathcal{O}(nm)$ time complexity. This can be improved by the use of a window of size $r > 0$ that will constrain the amount of warping allowed. In particular, it restrains the distance in the time axis that a point s_i, $i = 1, \dots, n$ can be mapped to in t.

We can use the so-called *"Sakoe-Chiba Band"* window which assumes that the best path will not stray too far away from the main diagonal of the distance matrix. Let w be the window size, the window constraint implies the following,

$$j - w \leq i \leq j + w$$

where i and j are two points aligning in s and t respectively. The number of cells that needs to be computed in the distance matrix is therefore reduced. However, this constraints does not work so well if we have time series stopping at radically different time. For such time series, the warping path can stray far away from the linear one implying the need to fill (almost) the whole distance matrix.

3.2 Results

3.2.1 User-independent cross-validation

3.2.2 User-dependent cross-validation

Appendix