Implementing DTW and FastDTW algorithms on measuring time series distance

Reid WANG

Contents

1 Dynamic Time Warping Distance

Introduction to dynamic time warping distance is based on the blog posted by Romain Tavenard DTW is a similarity measure between time series that has been introduced independently in the literature by [Vint68] and [SaCh78], in both cases for speech applications.

Motivation

Traditional distance or similarity metric has two characteristics,

1. Input series A and B should have same length.
2. Calculations are point-wise based.

Dynamic Time warping distance seeks for the temporal alignment that minimize the distance between aligned series.

Problem Formulation

The problem can be formulated as a optimization problem:

$$DTW_q\left(x, x'\right) = \min_{\pi \in \mathcal{A}(x,x')} \left(\sum_{(i,j)\in\pi} d\left(x_i, x'_j\right)^q \right)^{\frac{1}{q}} \tag{1}$$

Here, an alignment path π of length K is a sequence of K index pairs $\left(\left(i_0, j_0\right), \ldots, \left(i_{K-1}, j_{K-1}\right)\right)$ and $\mathcal{A}\left(x, x'\right)$ is the set of all admissible paths. In order to be considered admissible, a path should satisfy the following conditions:

- Beginning (resp. end) of time series are matched together:
  - $\pi_0 = (0, 0)$
  - $\pi_{K-1} = (n-1, m-1)$
- The sequence is monotonically increasing in both i and j and all time series indexes should appear at least once, which can be written:
  - $i_{k-1} \leq i_k \leq i_{k-1} + 1$
  - $j_{k-1} \leq j_k \leq j_{k-1} + 1$

1.1 Dot Product Representation

Another way to represent a DTW path is to use a binary matrix whose non-zero entries are those corresponding to a matching between time series elements. This representation is related to the index sequence representation used above through:

$$\left(A_\pi\right)_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in \pi \\ 0 & \text{otherwise} \end{cases} .$$

This is illustrated in the Figure below where nonzero entries in the binary matrix are represented as dots and the equivalent sequence of matchings is produced on the right:

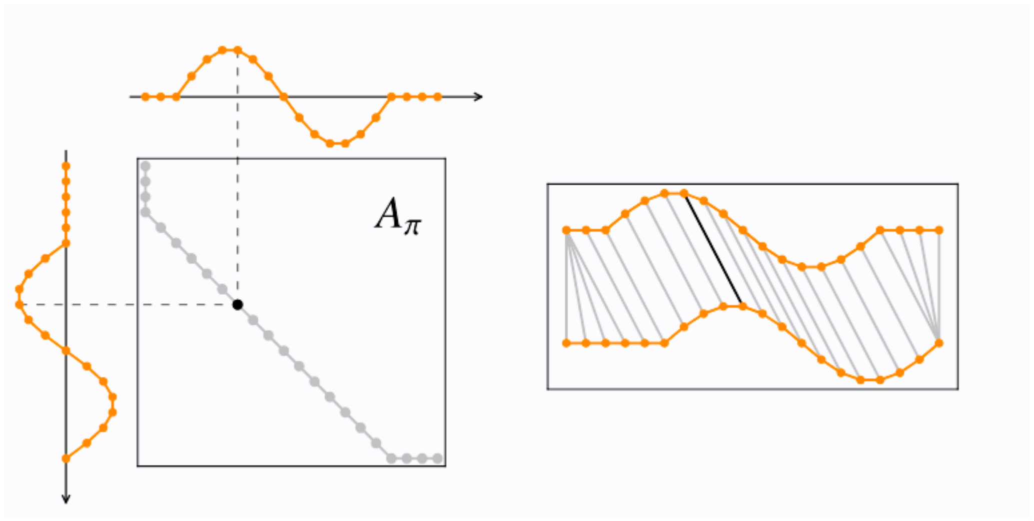

Figure 1.1: Dynamic Time Warping path represented as a binary matrix. (Left) Each dot on the path indicates a nonzero entry in A_π , hence the matching of an element in x with an element in x' . Note the correspondence between this representation and the one on the right.

Under matrix notation, DTW distance can be written as the minimization of a dot product between matrices

$$DTW_q\left(x, x'\right) = \min_{\pi \in \mathcal{A}(x, x')} \left\langle A_\pi, D_q\left(x, x'\right)\right\rangle^{\frac{1}{q}} \tag{1}$$

where $D_q\left(x, x'\right)$ stores distances $d\left(x_i, x'_j\right)$ at the power q.

Solution by Dynamic Programming

An exact solution to this optimization problem in Equation (1.1) can be found using dynamic programming. Dynamic programming relies on recurrence, which consists in linking the solution of a given problem to solutions of (casier) sub-problems. Once this link is known, the dynamic programming approach solves the original problem by recursively solving required sub-problems and storing their solutions for later use (so as not to re-compute subproblems several times).

In the case of DTW, we need to rely on the following quantity:

$$R_{i,j} = DTW_q\left(x_{\rightarrow i}, x'_{\rightarrow j}\right)^q$$

where the notation $x_{\rightarrow i}$ denotes time series x observed up to timestamp i (included). Then, we can observe that:

$$\begin{aligned}
R_{i,j} &= \min_{\pi \in \mathcal{A}\left(x_{\rightarrow i}, x'_{\rightarrow j}\right)} \sum_{(k,l) \in \pi} d\left(x_k, x'_l\right)^q \\
&\overset{*}{=} d\left(x_i, x'_j\right)^q + \min_{\pi \in \mathcal{A}\left(x_{\rightarrow i}, x'_{\rightarrow j}\right)} \sum_{(k,l) \in \pi[:-1]} d\left(x_k, x'_l\right)^q \\
&\overset{**}{=} d\left(x_i, x'_j\right)^q + \min\left(R_{i-1,j}, R_{i,j-1}, R_{i-1,j-1}\right)
\end{aligned}$$

(*) comes from the constraints on admissible paths π : the **last element on an admissible path needs to match the last elements of the series**. Also, (**) results from the continuity conditions on the admissible paths. Indeed, a path that would align time series $x_{\rightarrow i}$ and $x'_{\rightarrow j}$ necessarily encapsulates either:

- a path that would align time series $x_{\rightarrow i-1}$ and $x'_{\rightarrow j}$,
- or a path that would align time series $x_{\rightarrow i}$ and $x'_{\rightarrow j-1}$,

- or a path that would align time series $x_{\rightarrow i-1}$ and $x'_{\rightarrow j-1}$,

as illustrated in the Figure below:

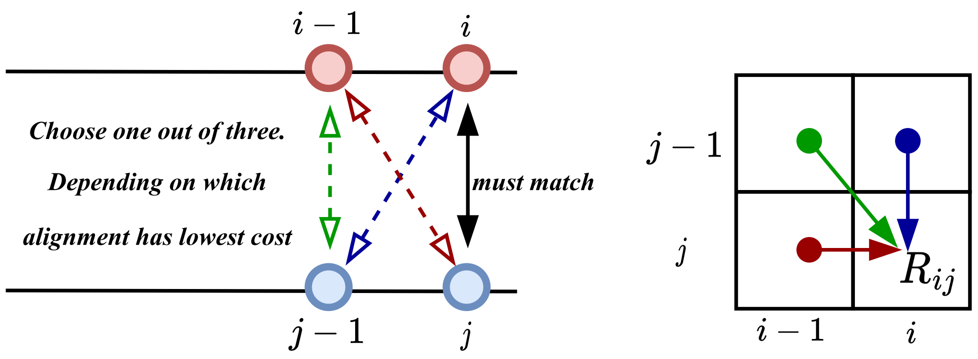

Figure 1.2: DTW transitions

2 FastDTW: DTW in linear time and space

2.1 FastDTW Algorithm

DTW algorithm runs $\mathcal{O}(n^2)$ in time and space, which is quite slow for large data. A simple idea to reduce the computational time is to represent the time series in a lower dimension or with fewer data points.

For example, we can splitting the time series with 16 points into 4 groups, and taking the average value of each group. And use the 4 points (seen as a pattern retrieved from the original series) to calculate the DTW distance with other series. However, by taking average, information within the group are wiped out and the DTW calculated in this way is neither authentic nor decent. (A more decent way might rely on researching in pattern representation and sampling problems).Thus, instead of using DTW directly, we use the warp paths in a lower resolution(fewer data points) as a initial guess of the next search. We continue this procedure (searching around the projected warping paths in a lower resolution and projecting to a higher resolution as an initial guess) until all the data points is resolved.

In FastDTW, we summarize the above procedures into three key operations:

- **Coarsening** - Shrink the size of a time series into half (By taking the average of consecutive points)
- **Projection** - Find the minimum distance warp path as an initial guess for a higher resolution's minimum distance warp path. Projection takes a warp path calculated at a lower resolution and determines what cells in the next higher resolution time series the warp path passes through. Since the resolution is increasing by a factor of two, a single point in the low-resolution warp path will map to at least four points at the higher resolution.
- **Refinement** - Refine the warp path projected from a lower resulution through local adjustments of the warp path

The calculation is actually done through recursive steps. For example, back to the previous example,

1. *Coarsening* - We first splitting the 16 data points into 2 groups, 8 points in each.And taking the average of each group and use this average value to as a lower resolution of the original points. We perform this procedure until there are only 2 points.

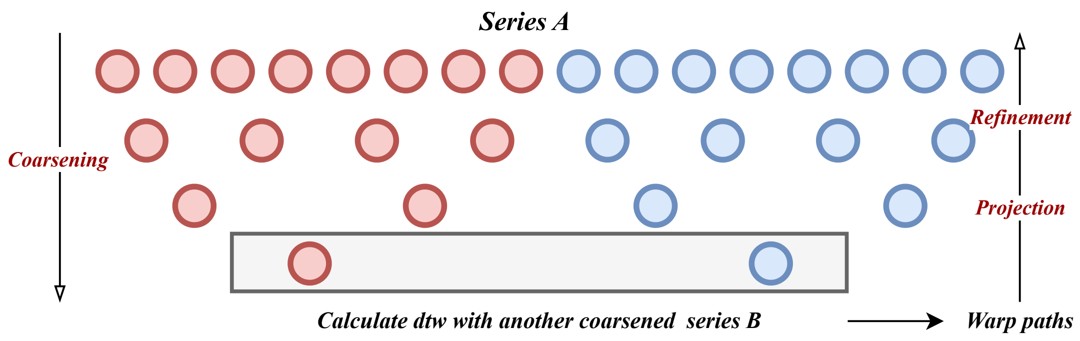

Figure 2.1: FastDTW-Illustration 1 - Coarsening

2. *Projection* - With only 2 points in the group, we can easily calculate the Dynamic Warping distance between two coarsened series.Suppose the warp path with 2 points is given below. For projection, we change every pixel in the left figure into a 2×2 pixels. And drag this 2×2 pixel block along the warp paths to get the projected warp paths in a higher resolution. To increase the accuracy of FastDTW we also add a radius parameter which increases the search area.

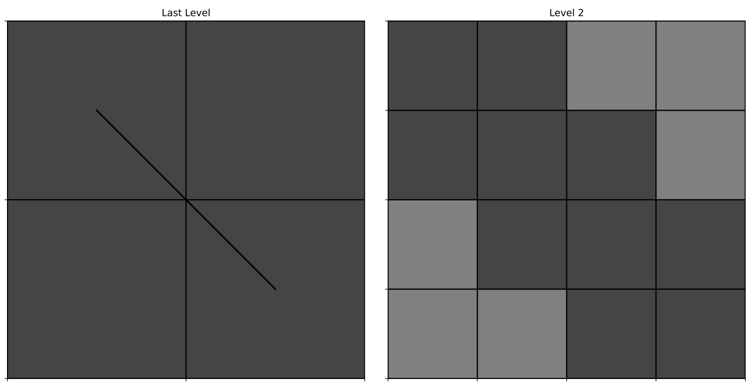

Figure 2.2: FastDTW-Illustration 2 - Projection

The black and grey pixels in Figure 2.2 refers to the pairs that are evaluated during each calculation of warp paths. The difference is that black pixels is the projection of the warp paths from a lower resolution while the grey pixels come from the radius. A more intuitive illustration of the projection and radius method is given as below,

Figure 2.3: Projection and Expanding Searching area by radius

3. *Refinement*- Refinement is done by evaluating the search area after the projection and expanding by radius.By refinement, we get a new warp path. And after that we continue the projection and refinement steps until series becomes the original time series. An illustration of the above example is shown as in Figure 2.12,

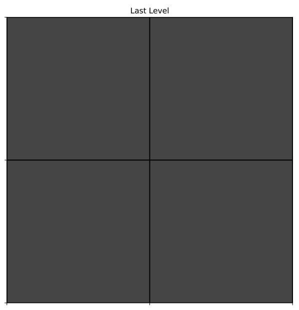

Figure 2.4: Initial Search

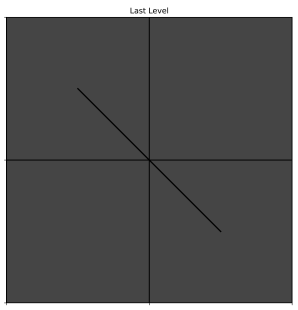

Figure 2.5: Initial Path

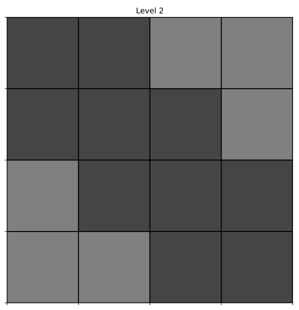

Figure 2.6: Projection 1

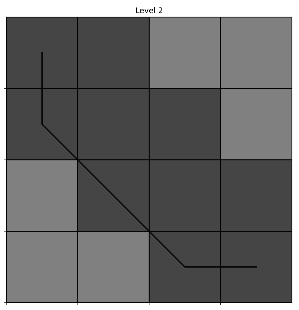

Figure 2.7: Refinement 1

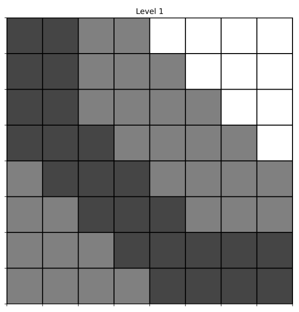

Figure 2.8: Projection 2

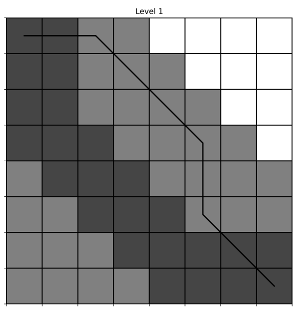

Figure 2.9: Refinement 2

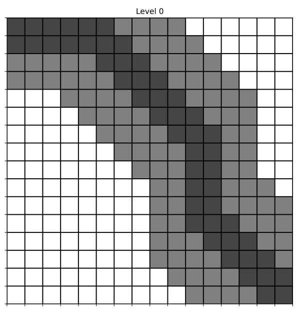

Figure 2.10: Projection 3

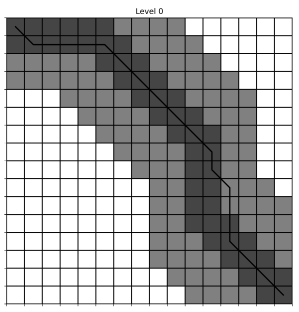

Figure 2.11: Refinement 3

Figure 2.12: FastDTW - Illustration 3 - Refinement

3 Implementation in Python

3.1 Function Documentation

fastdtw(X, Y, radius, level=0, do_plot=False)

- **Description**: Implementation of FastDTW Algorithm based on: *FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space. Stan Salvador and Philip Chan*
- Implementation of Dynamic Time Warping Distance
- **Input**:
  - X: ndarray(M, d) A d-dimensional point cloud with M points
  - Y: ndarray(N, d) A d-dimensional point cloud with N points
  - radius: int Radius of the l-infinity box that determines sparsity structure at each level
  - level: int An int for keeping track of the level of recursion
  - do_plot: boolean Whether to plot the warping path at each level and save to image files
- **Return**:

  (float: cost, ndarray(K, 2): The warping path)

The word ”box” should be boxed.

dtw_brute_backtrace(X, Y , do_plot=False, usage="dtw")

- Input:
  - X: ndarray(M, d) A d-dimensional point cloud with M points
  - Y: ndarray(N, d) A d-dimensional point cloud with N points
  - do_plot: boolean Whether to plot the warping path at each level and save to image files
- Return:

  (float: cost, ndarray(K, 2): The warping path)

4 Other considerations

4.1 Distance to Similarity Metric

The following two ways can be considered,

- **Simulation based**:
  1. Fix the starting and ending points of the alignment matrix(constriants of DTW algorithm). And simulate symmetric random walks from start to end.
  2. Compute average distance from the simulated alignment matrix as maxDTW.
  3. Similarity Score given by:

$$\text{Similarity Score} = \frac{\text{maxDTW} - \text{DTW}}{\text{maxDTW}}$$

- Rough Estimation:

  Use the avaerage distance from the distance matrix multiplied by $\max(N, M)$ as maxDTW.And calculate the similarity score as above

4.2 Others variants of DTW

- ***Soft-DTW: a Differentiable Loss Function for Time-Series*** by Marco Cuturi and Mathieu Blondel[1].

[1]Paper can be found in https://proceedings.mlr.press/v70/cuturi17a/cuturi17a.pdf