Final Project Report

Examining the effects of weight and slope constraint condition in Dynamic Time Warping on 1-nearest-neighbor classification accuracy

and

Shape Averaging Method for multiple time series sequences

Present to

Assoc. Prof. Dr. Chotirat Ratanamahatana

By

6231325521 Thanapat Trachu

6231340921 Pongsapak Pulthasthan

2110430 Time Series Mining and Knowledge Discovery (2022/2)

Department of Computer Engineering

Faculty of Engineering, Chulalongkorn University

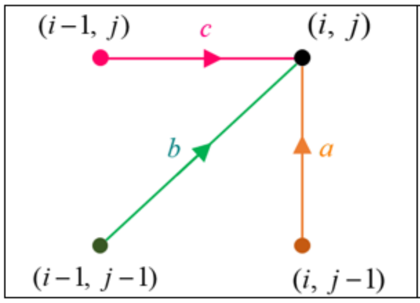
1. Examining the effects of weight and slope constraint condition in Dynamic Time Warping on 1-nearest-neighbor classification accuracy

1.1 Weight (Symmetric Weight VS Asymmetric Weight)

Let A and B are time series that can be expressed as a sequence of feature vectors

$$A = a_1, a_2, \ldots, a_i, \ldots, a_I$$
$$B = b_1, b_2, \ldots, b_i, \ldots, b_J$$

According to Sakoe and Chiba (1978)'s paper [1], the simplest dynamic programming form of Dynamic Time Warping (DTW) algorithm is



$$g(i,j) = \min \begin{cases} g(i,\ j-1) + a \cdot d(i,\ j) \\ g(i-1,\ j-1) + b \cdot d(i,\ j) \\ g(i-1,\ j) + c \cdot d(i,\ j) \end{cases}$$

Simplest DP-form of Dynamic Time Warping (DTW)

where ***g*** is the accumulated distance matrix between time series A and B,
d is the distance matrix between time series A and B, where

$$d(i,\ j) = \left\| a_i - b_j \right\|$$

and ***a, b, c*** are the weights of the distance of the three neighboring cells: ***g(i, j-1)***, ***g(i-1, j-1)***, and ***g(i-1, j)*** respectively

These weights are said to be **symmetric** when **a = c**, such as **(a, b, c) = (1, 1, 1) or (1, 2, 1)**. Therefore, when **a ≠ c**, these weights are said to be **asymmetric**, such as **(a, b, c) = (1, 1, 0) or (0, 1, 1)**

The alignment (warping function/path) F which realizes a mapping from the time axis of pattern A onto that of pattern B is

$$F = (c(1),\ c(2),\ \ldots,\ c(k),\ \ldots,\ c(K))$$

$$\text{where } c(k) = (i(k),\ j(k))$$

The normalized DTW distance can be calculated from

$$\text{Normalized DTW Distance} = D(A, B) = \min_{F}\left[\frac{\sum_{k=1}^{K} d(c(k)) \cdot w(k)}{\sum_{k=1}^{K} w(k)}\right] = \frac{g(I, J)}{\sum_{k=1}^{K} w(k)}$$

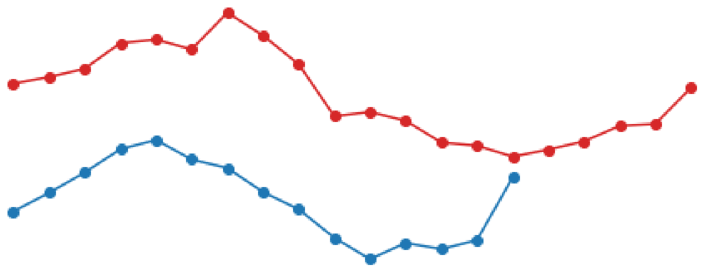
where **w(k)** is a nonnegative distance weighting coefficient, which is intentionally introduced to allow DTW to measure more flexible characteristics between time series A and B.

Next, We conduct some experiments to compare the performance of symmetric form DP-matching and asymmetric form DP-matching. First, we generate two simple mock-up time series and calculate the DTW between them using different weights (a, b, c) to illustrate how these weights might affect the alignment and the cost matrix using some visualizations. Next, the real datasets are being used. The performance is evaluated by comparing the 1-nearest-neighbor classification accuracy obtained from each different weights (a, b, c) in each dataset. There is also a discussion at the end about what form is better (symmetric or asymmetric) and whether we can find the optimal values of these weights to maximize the classification accuracy.

Mock-up Time Series

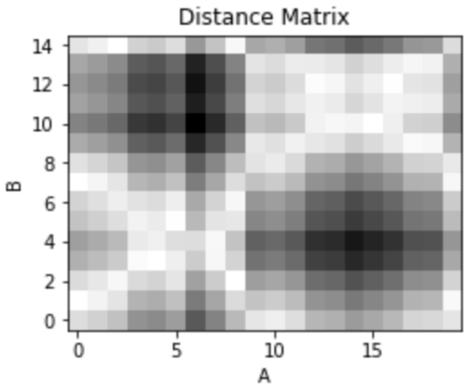
In this section, we use the Python code provided below to generate two simple sinusoidal waves (A and B) whose amplitude is 1.5 with additional random gaussian noise (mean = 0 and variance = 0.3). Note that these waves have the same shape but the things that differentiate between them is their frequency and their length.

```
import numpy as np
np.random.seed(2023)

I = np.linspace(0, 2, 20) #length A = 20
J = np.linspace(0, 1, 15) #length B = 15

A = 1.5 * np.sin(1 * np.pi * I) + np.random.normal(loc = 0, scale = 0.3, size = 20)
B = 1.5 * np.sin(2 * np.pi * J) + np.random.normal(loc = 0, scale = 0.3, size = 15)
```



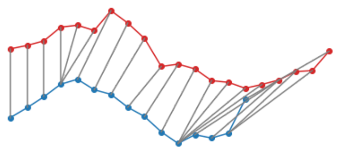
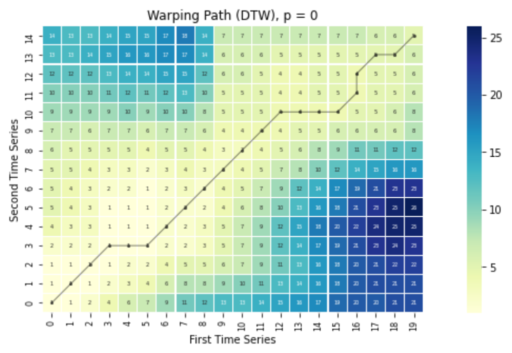
Two Mock-up Generated Time Series (A, B)

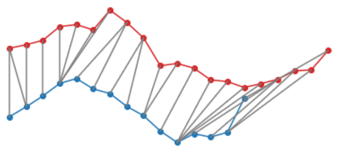
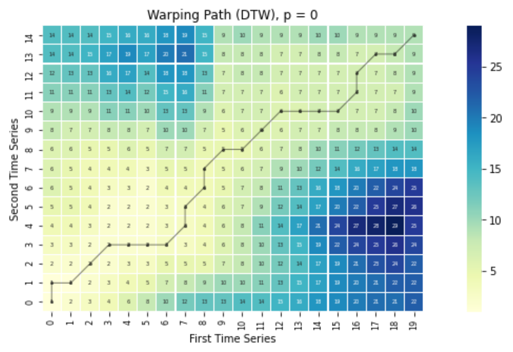
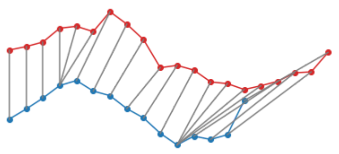
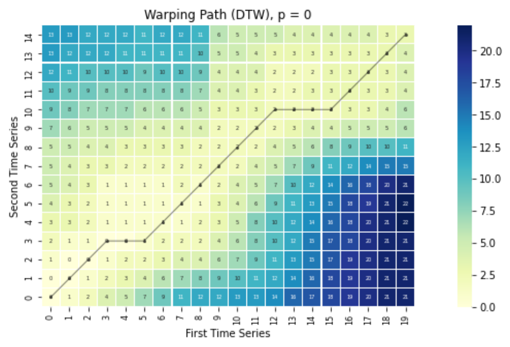
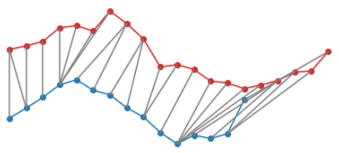
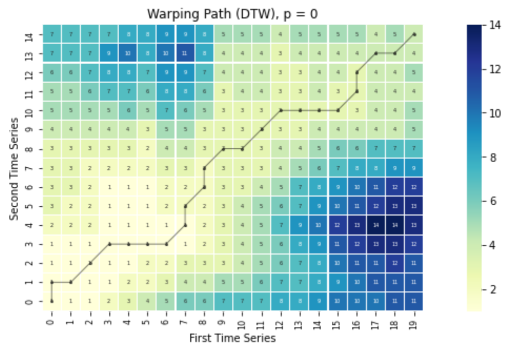
Then, we compute the distance matrix ***d*** of time series A and B which can be visualized as figure below.



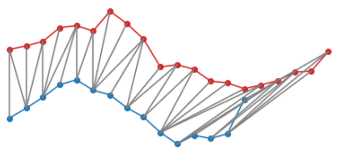
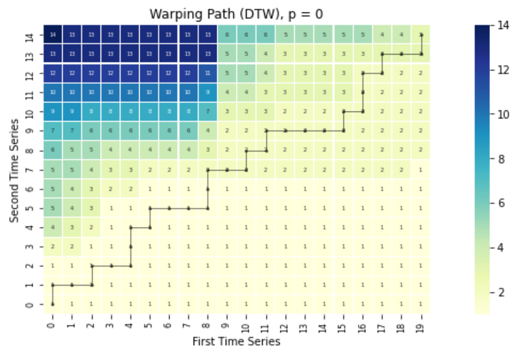
After that, we visualize the DTW alignments obtained from using different weights (a, b, c) in the DTW calculation.

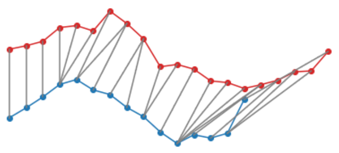
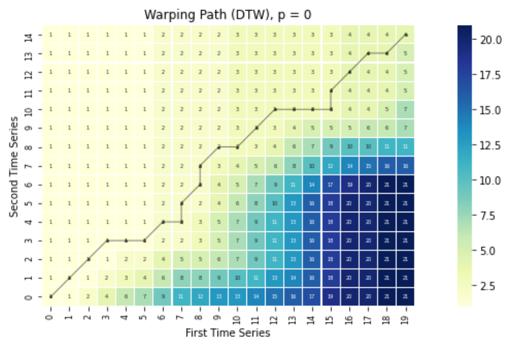
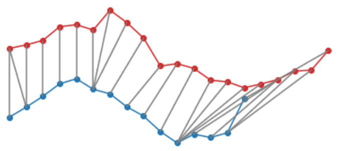
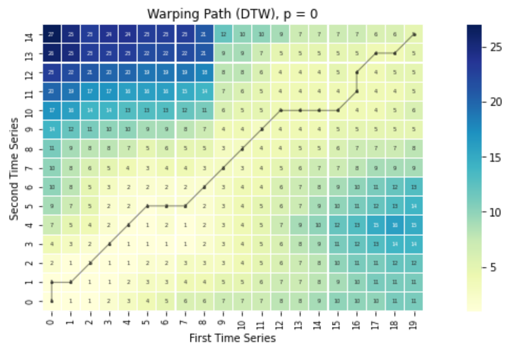
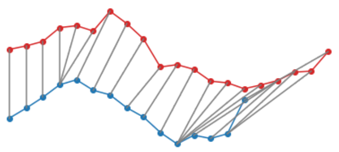
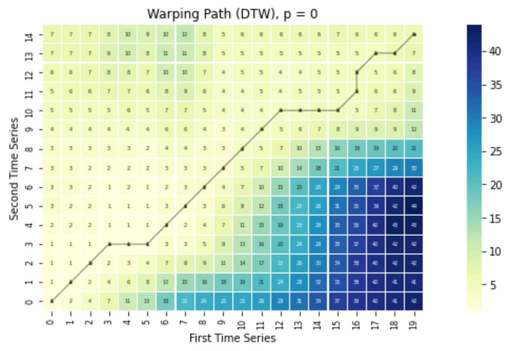
Symmetric Weight

(a, b, c)	**DTW Alignment**	**Cost Matrix *g* with Warping Path**	***D(A,B)***
(1, 1, 1) symmetric		 Warping Path (DTW), p = 0	0.2567

(1, 2, 1) symmetric			0.2454
(1, 0.5, 1) symmetric			0.2480
(0.5, 1, 0.5) symmetric			0.2454

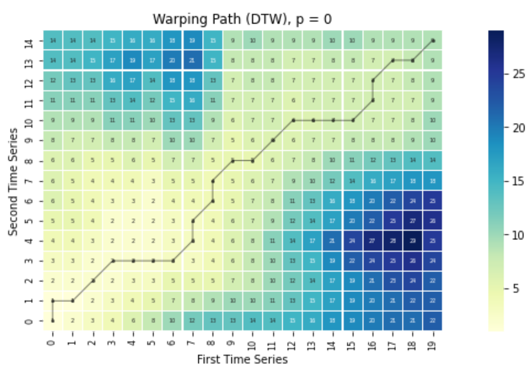
Asymmetric Weight

(a, b, c)	**DTW Alignment**	**Cost Matrix *g* with Warping Path**	***D(A,B)***
(1, 1, 0) asymmetric			0.2073

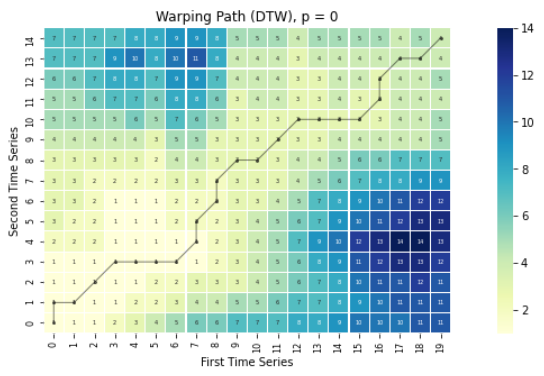
(0, 1, 1) asymmetric			0.2209
(2, 1, 0.5) asymmetric			0.2383
(0.5, 1, 2) asymmetric			0.2329

From the above tables, it is obvious that each weight produces different DTW alignment and also different DTW distance even though it is calculated on the same time series A and B. This is because the different weights have different influences on the decision to select the neighboring cells when computing the DTW cost in dynamic programming.

However, in the symmetric case, note that if the ratio of weight (a : b : c) is the same, it should produce the same DTW Alignment and DTW Normalized Distance. Therefore, we will exclude the weight (0.5, 1, 0.5) out from the experiment as it produces the same result with the weight (1, 2, 1).



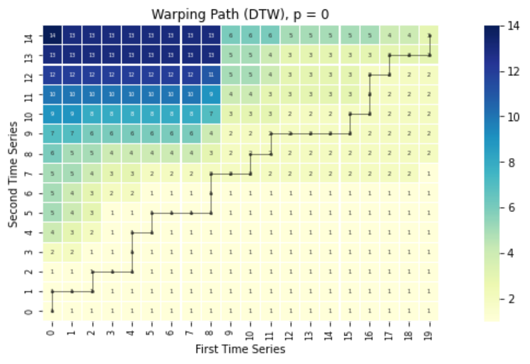
(a, b, c) = (1, 2, 1)
D(A, B) = 0.2454



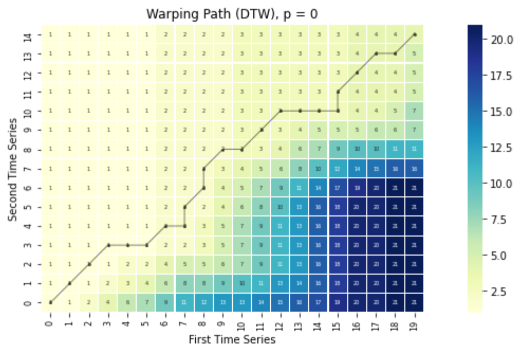
(a, b, c) = (0.5, 1, 0.5)
D(A, B) = 02454

For asymmetric weights, there are interesting cases when some weight is reduced to zero which are: (a, b, c) = (1, 1, 0) and (0, 1, 1). We notice that the dark area (which represents high cost value) in the cost matrix of these weights are only presented only at the top left corner (in case (1, 1, 0)) or right bottom corner (in case (0, 1, 1)), while the dark area in the cost matrix computed from other symmetric weights are shown at both corners.

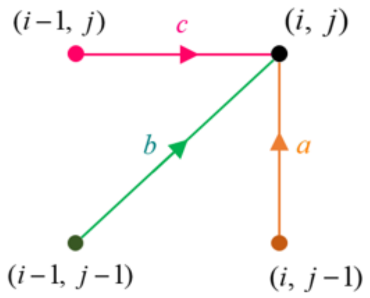
In case (1, 1, 0), when the point in the warping function steps in the direction of the i-axis. This means that some feature vectors a_i are possibly excluded from the calculation which is not good as we should treat both patterns equally. An exclusion of any feature vectors should be avoided as long as possible when using these DTW results to do the other work, especially the classification tasks.



(a, b, c) = (1, 1, 0)



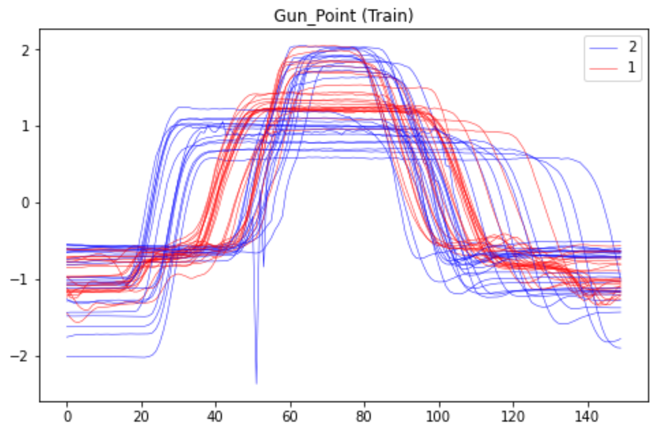
(a, b, c) = (0, 1, 1)

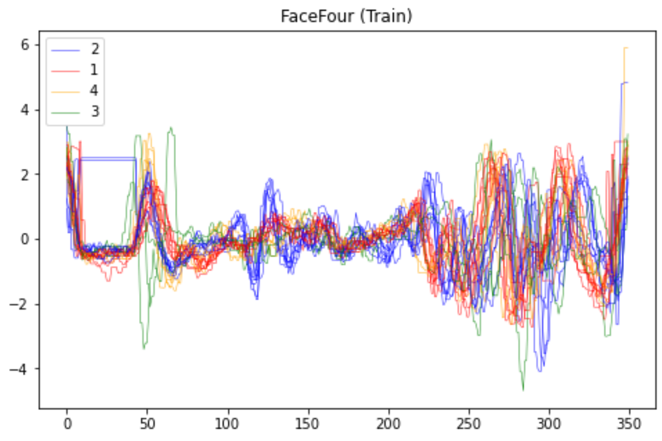


Time Series Dataset

In this part, we investigate how the weight (a, b, c) affects the 1-nearest-neighbor classification accuracy on the well-known time series datasets available on the internet which are 1) Gun-Point 2) Lightning-7 3) Synthetic Control 4) Coffee and 5) FaceFour. The details of these datasets are provided in the table below.

Dataset Name	Number of classes	Size of training set	Size of testing set	Time series Length	Total File Size
Gun-Point	2	50	150	150	484 KB
Lightning-7	7	70	73	319	733 KB
Synthetic Control	6	300	300	60	586 KB
Coffee	2	28	28	286	156 KB
FaceFour	4	24	88	350	629 KB





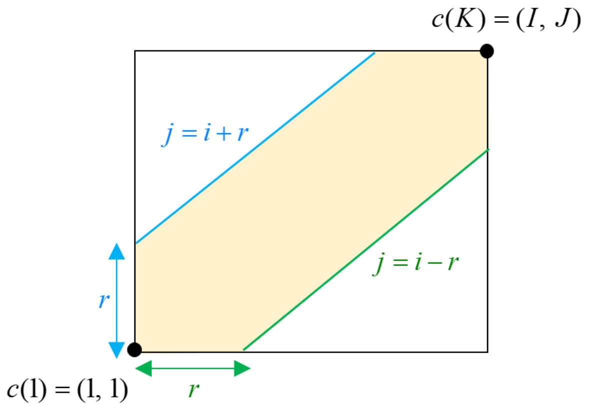
Adjustment Window Condition

As the length of the time series in the datasets mentioned in the above table is getting bigger and the time complexity for computing Dynamic Time Warping (DTW) is $O(n^2)$ where n is the length of the time series.. Therefore, an idea of global constraints are proposed in order to eliminate excessive calculation due to the fact that the time-axis fluctuation in usual cases never causes a too excessive timing difference, and prevent any unreasonable warping.

According to Sakoe and Chiba (1978)'s paper [1], an adjustment window (also known as Sakoe-Chiba Band) is proposed. The condition is

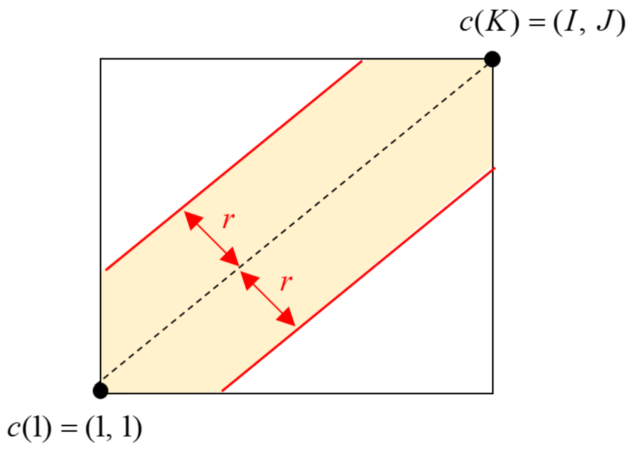
$$|i(k) - j(k)| \leq r$$

where ***r*** is an appropriate positive integer called window length.



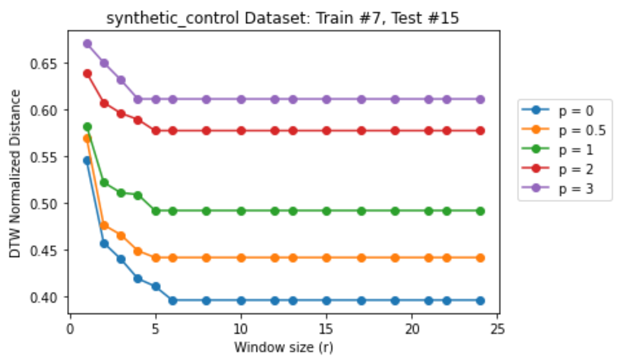
However, to make use of Sakoe-Chiba band properly, we find that we always have to set the window length (***r***) to be higher than the difference between the length of the two series (**r** ≥ |***I-J***|) to ensure that the last warping point in DTW alignment ***c(K) = (I, J)*** is still in the band area in order to satisfy the boundary conditions of the DTW. Although this is fine when the two time series have the same length, we still want to make sure that the last warping point is still in the band area regardless of the difference of the length between those two time series or datasets.

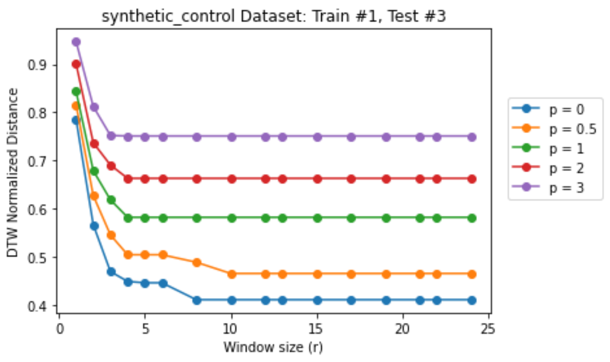
Therefore, we adapt the idea from this paper a little bit in order to ensure that the last warping point ***c(K) = (I, J***) is still in the band. By giving a new definition for the window length (**r**) which is the perpendicular distance from the main diagonal line which is drawn from the first warping point ***c(1) = (1, 1)*** to the last warping point ***c(K) = (I, J)***. We will use this new defined band for the rest of the report.



Note that, in each dataset, the appropriate value of the window length (***r***) is selected from the value 7-10% of the length of each data in that particular dataset. (approximately)

Dataset	Time series Length	Window Length (r)	width (%)
Gun-Point	150	12	8.00%
Lightning-7	319	24	7.52%
Synthetic Control	60	6	10.00%
Coffee	286	28	9.79%
FaceFour	350	28	8.00%

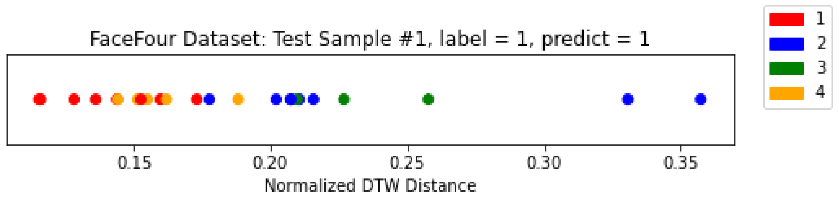




We have sampled data pairs from the Synthetic Control dataset (each time series has length = 60) and calculated the DTW distance by varying the window size (***r***). We found that the calculated distances tend to be stable when the window size (***r***) >= 10% of its length (which is r >= 6 for this dataset)

Evaluation and Result

For each test data, we compute the DTW Normalized Distance among each training data in the particular dataset. The predicted label comes from the label of the training data which has the lowest DTW Normalized Distance.



1-nearest-neighbor classification accuracy

Dataset	**Weight *(a, b, c)***						
	Symmetric			**Asymmetric**			
	(1, 1, 1)	**(1, 2, 1)**	**(1, 0.5, 1)**	**(1, 1, 0)**	**(0, 1, 1)**	**(2, 1, 0.5)**	**(0.5, 1, 2)**
Gun-Point	0.9000	**0.9200**	**0.8467**	0.9067	0.9267	0.9133	**0.9200**
Lightning-7	0.7808	0.7808	0.8219	0.7534	**0.6986**	**0.8493**	0.7260
Synthetic Control	0.9600	0.9400	0.9567	**0.7134**	0.9800	0.8133	**0.9900**
Coffee	0.8214	0.7857	0.8214	**0.6429**	**0.8571**	0.7857	0.7857
FaceFour	0.8181	0.8409	0.7613	**0.4773**	0.8409	0.7159	**0.8523**

Conclusion

From the result in the table above, it is obvious that weight (a, b, c) = (1, 1, 0) achieves the lowest classification accuracy in most datasets. This is not so surprising as it happens from the implementation bias in our code, as we treat the test data as the first time series and the training data as the second time series when calculating the DTW distance between the test data (query) and the training data.

```
for i in range (self.n_train):
        dist_mat = calculate_distance_matrix(X_test, self.X_train[i])
        alignment_cost, normalized_alignment_cost, cost_mat, path =
self.dtwDistance.dp_dtw_sakoe_chiba(dist_mat, dist_weight, p = p, r = r)
```

According to the discussion in the previous section, when the weight (a, b, c) = (1, 1, 0) if the point in the warping path steps in the direction of the i-axis (first time series/test data), some data in the test data are possibly excluded from the calculation which is not good as we cannot match some characteristics of the test data to the available training data, so we cannot give an accurate prediction. That's why weight (1, 1, 0) gives the poorest performance among the other weights. To conclude, we should avoid using zero in the weight as it may skip some important features in the test data which result in poor classification performance.

According to the data in the above table, most of the best accuracy in each dataset is achieved by using the asymmetric weights. However, most of the abysmal accuracy achieved also comes from using weight (1, 1, 0). This shows that it doesn't always guarantee that using asymmetrical weight would give the best performance. Moreover, using symmetric weight also usually gives acceptable performance, but it may not be the best.

However, these ***(a, b, c)*** weights are considered as hyperparameters, this means that there are no all-time optimal values for these weights. Moreover, the window length (***r***) is also a hyperparameter that has an effect with the accuracy. Therefore, we cannot summarize which weight would result in the best performance in all datasets.

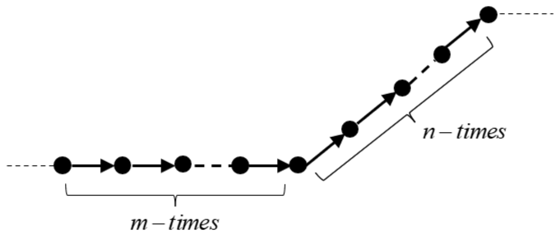
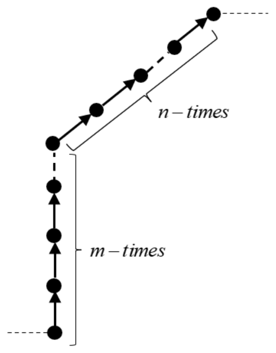
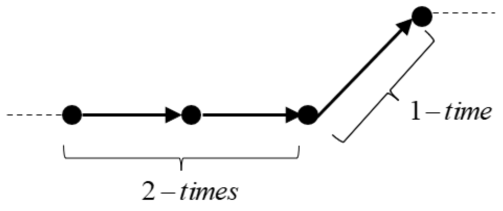
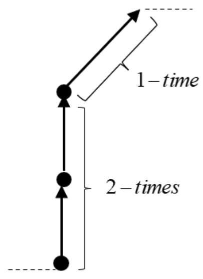
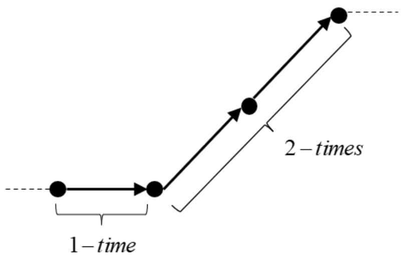
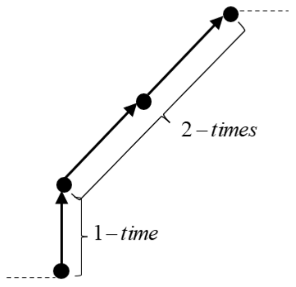
In order to find such optimal values, the thing we can do is to do the grid search to search for all possible combinations of weights in the pre-defined search spaces and find the weight that gives the best accuracy. This is theoretically possible, but in practice, it is very difficult because the Dynamic Time Warping algorithm has $O(n^2)$ time complexity which takes a very long time to compute. These difficulties can be abated by combining other techniques to reduce the computational time, such as reducing the training data by conducting a shape-averaging algorithm to average all the train data in the particular class or using dimensionality reduction techniques or lower bounds techniques to speed up the calculation process.

1.2 Slope Constraint Condition

It has been mentioned in the Sakoe and Chiba (1978)'s paper [1] that neither too steep nor too gentle a gradient should be allowed for the warping path because such deviations may cause an undesirable time-axis warping, especially, when the pattern A is relatively short compared to the pattern B. Therefore, a restriction called a slope constraint condition (***p***) was set.

$$p = \frac{n}{m}$$

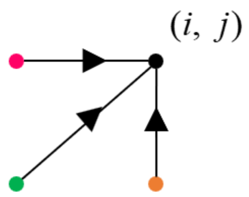
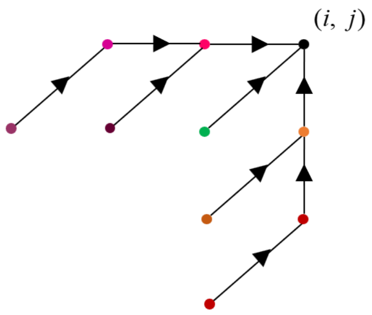
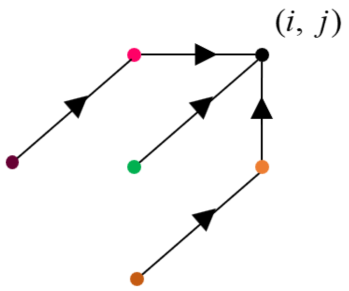
The concrete definition of this restriction is "if point c(k) moves forward in the diction of i (or j)-axis consecutive ***m*** times, then point c(k) is not allowed to step further in the same direction before stepping at least ***n*** times in the diagonal direction".

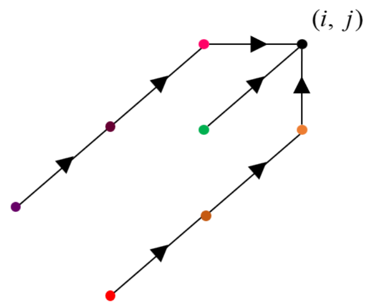
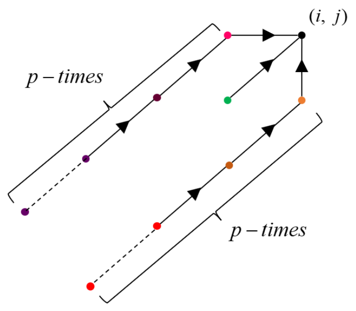
p	Minimum Slope	Maximum Slope
n/m		
1/2		
2		

When p = 0 (which is n = 0), there are no restrictions of the warping function slope, but when p = ∞ (which is m = 0) the warping function is only restricted to the diagonal line (j = i) (similar to the euclidean distance).

In the previous section, we have seen the simplest DP-form of the dynamic time warping algorithm which considers only three neighboring cells. In that case, there is no such slope restriction (p = 0). However, in this section, we will consider the other forms of DP-equation of the dynamic time warping algorithm with the slope constraint condition employed. There are five values of ***p*** we considered in this experiment which are ***p = 0, 1/2, 1, 2, and 3.*** In each variation we only use the symmetric weight written in [1] in consideration, and examine whether this restriction has an effect on the one-nearest-neighbor classification accuracy.

Symmetric DP-Algorithms with Slope Constraint Condition (p = 0, 1/2, 1, 2)

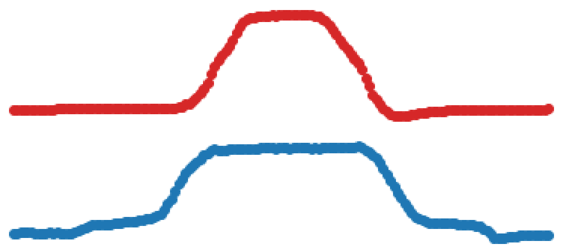
p	**Schematic Explanation**	**DP-equation *g(i, j)* =**
0		$\min\begin{cases} g(i, j-1) + d(i, j) \\ g(i-1, j-1) + 2d(i, j) \\ g(i-1, j) + d(i, j) \end{cases}$
½*		$\min\begin{cases} g(i-1, j-3) + 2d(i, j-2) + d(i, j-1) + d(i, j) \\ g(i-1, j-2) + 2d(i, j-1) + d(i, j) \\ g(i-1, j-1) + 2d(i, j) \\ g(i-2, j-1) + 2d(i-1, j) + d(i, j) \\ g(i-3, j-1) + 2d(i-2, j) + d(i-1, j) + d(i, j) \end{cases}$
1		$\min\begin{cases} g(i-1, j-2) + 2d(i, j-1) + d(i, j) \\ g(i-1, j-1) + 2d(i, j) \\ g(i-2, j-1) + 2d(i-1, j) + d(i, j) \end{cases}$

2		$\min\begin{cases} g(i-2,\ j-3) + 2d(i-1,\ j-2) + 2d(i,\ j-1) + d(i,\ j) \\ g(i-1,\ j-1) + 2d(i,\ j) \\ g(i-3,\ j-2) + 2d(i-2,\ j-1) + 2d(i-1,\ j) + d(i,\ j) \end{cases}$
p (int)		$\min\begin{cases} g(i-p,\ j-1-p) + 2\sum_{k=1}^{p} d(i+1-k,\ j-k) + d(i,\ j) \\ g(i-1,\ j-1) + 2d(i,\ j) \\ g(i-1-p,\ j-p) + 2\sum_{k=1}^{p} d(i-k,\ j+1-k) + d(i,\ j) \end{cases}$

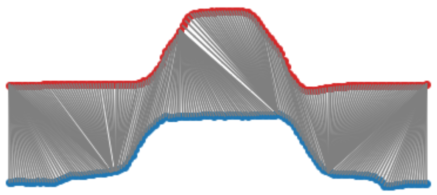
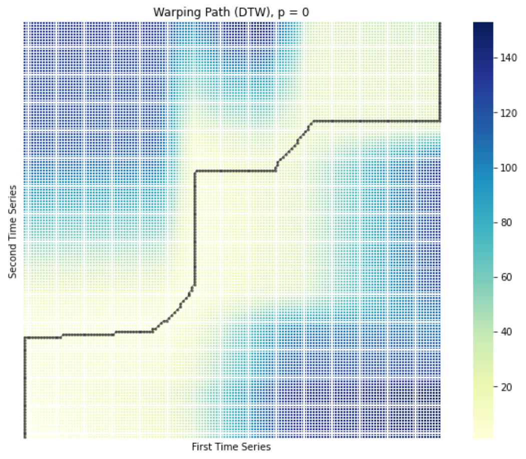
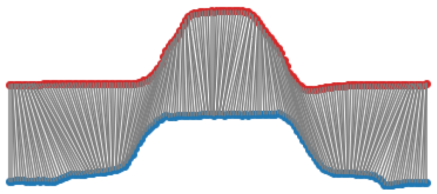
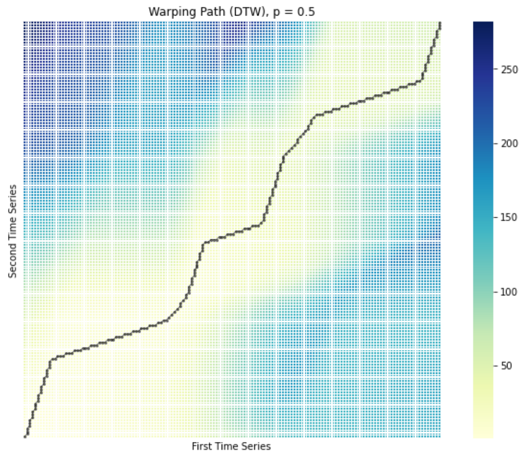
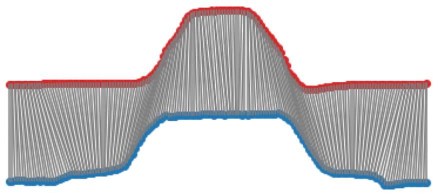
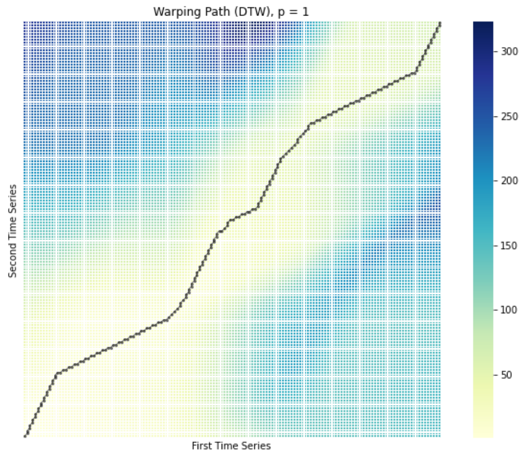
Next, We conduct some experiments to compare the performance of each variation. First, we sample data from the Gun-Point dataset as the mock-up time series and calculate the DTW between them using different slope constraint condition values (***p***) to illustrate how this restriction affects the warping alignment and the cost matrix using some visualizations. Next, the entire datasets are being used to compare the performance by using the 1-nearest-neighbor classification accuracy which is obtained from each different value of slope constraint condition as a metric. Then, there is a discussion at the end about which value of p is better for each classification task.

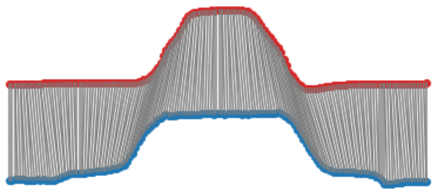
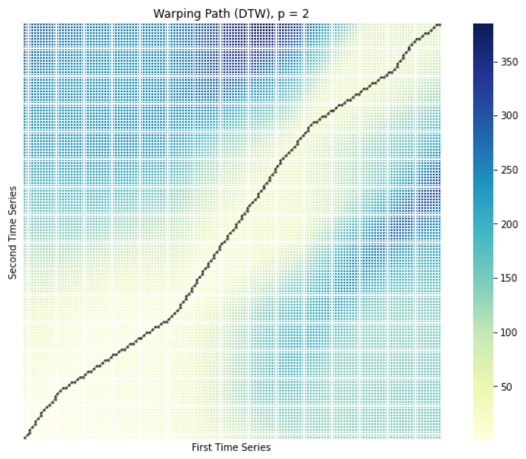
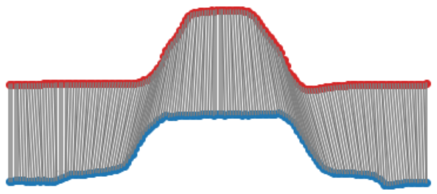
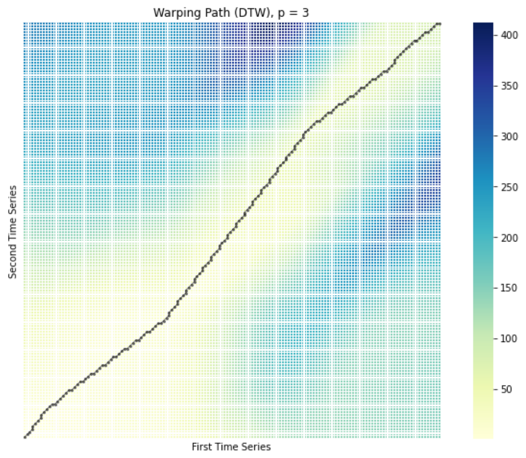
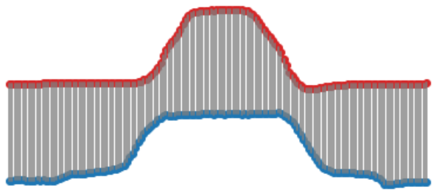
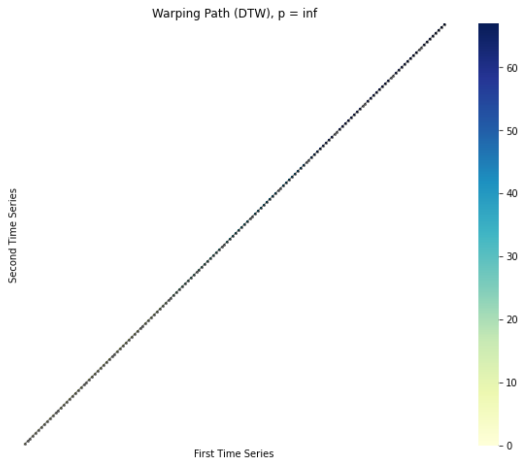
Mock-up Time Series

The two time series (length = 150) in the Gun-Point dataset were picked to use as the mock-up time series in this experiment. We calculate the DTW between these time series and visualize the path obtained from each ***p*** value.



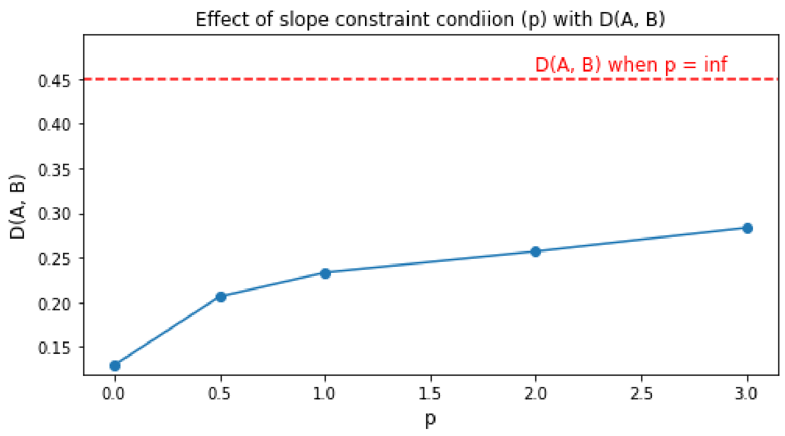
Two Mock-up Time Series from Gun-Point Dataset (A, B)

p	**DTW Alignment**	**Cost Matrix *g* with Warping Path**	***D(A,B)***
0			0.1294
1/2			0.2062
1			0.2332

2			0.2568
3			0.2834
inf			0.4496

From the result in the above table, when the value of ***p = 0*** (no slope constraint condition is employed), there might be a case that the warping path is too steep or too gentle which may cause an undesirable time-axis warping. Having no slope restrictions, this leads to achieving the lowest distance among the other values of ***p*** as shown in the figure below. When the value of ***p*** is risen, the slope of the warping path is getting more restricted due to the definition of slope constraint condition which makes the warping path more sensible and makes the distance to grow slightly. For the case ***p = inf***, it has been shown that the warping path is exactly aligned with the diagonal line and the calculated distance achieves the maximum value compared to the other ***p*** values. This is because it has no flexibility in time-axis when p = inf, this results in comparing point-to-point distance in the calculation.

However, note that in time series related tasks, the preferable value of ***p*** is not too low and too high. For the reason that if the slope condition is too strict, then the dynamic time warping algorithm might not handle the time fluctuation in time series data, but if the slope condition is too lax, it might achieve an undesirable warping function.



Time Series Dataset

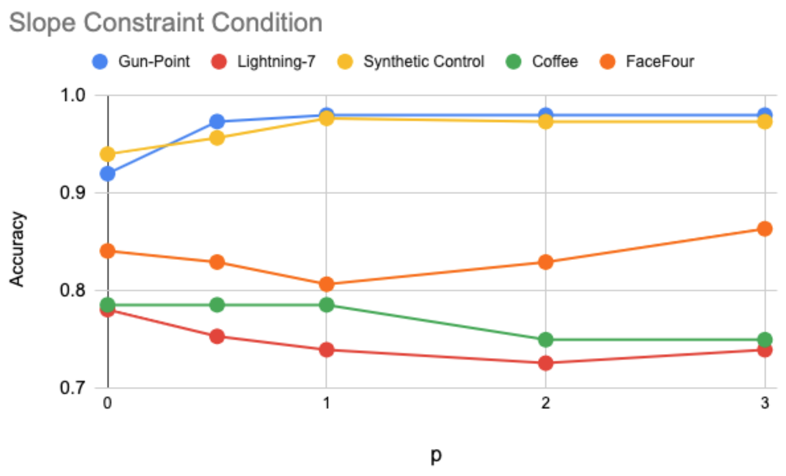
In this part, we investigate how the slope constraint value (***p***) affects the 1-nearest-neighbor classification accuracy on the well-known time series datasets available on the internet which are 1) Gun-Point 2) Lightning-7 3) Synthetic Control 4) Coffee and 5) FaceFour.

1-nearest-neighbor classification accuracy

Dataset	**Slope Constraint Value (p)**				
	p = 0	**p = 1/2**	**p = 1**	**p = 2**	**p = 3**
Gun-Point	**0.9200**	0.9734	**0.9800**	**0.9800**	**0.9800**
Lightning-7	**0.7808**	0.7534	0.7397	**0.7260**	0.7397
Synthetic Control	**0.9400**	0.9567	**0.9767**	0.9733	0.9733
Coffee	**0.7857**	**0.7857**	**0.7857**	**0.7500**	**0.7500**
FaceFour	0.8409	0.8295	**0.8068**	0.8295	**0.8636**

Conclusion

- When the value of p = 0 (no slope condition is employed), there might be a case that the warping path is too steep or too gentle which may cause an undesirable time-axis warping which sometimes can result in poor performance in classification accuracy.
- The preferable value of p should not also be too high. For the reason that if the slope condition is too strict, then the dynamic time warping algorithm might not handle the time fluctuation in time series data.
- To try using p around 1/2, 1, 2 is likely to obtain the optimal accuracy in most datasets. This is because the warping path obtained is more sensible and the restriction is not too lax or too strict.

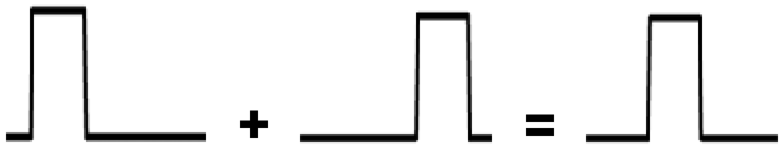


2. Shape Averaging Method for multiple time series sequences

Time series classification was an interesting task which has been used and benefited many industries such as detecting abnormal heart beat signals in healthcare. In the inception of the time series realm, many researchers have used one nearest neighbor classification model with Dynamic Time Warping (DTW) as a distance function to complete the time series classification tasks.

Despite an excellent performance in one nearest neighbor method, the major drawback of this method is a high time complexity, linearly to the number of training data. This will be a huge problem as we usually have a large dataset. To alleviate this problem, a shape averaging method has been proposed to reduce the number of time series in which the test time series is compared with. To specify, shape averaging methods were utilized to obtain represented time series for each class, then we needed to calculate the distance between test time series and represented time series, instead of the whole training data, even though this method also has a trade off, reducing the runtime and also the performance.

Dynamic time warping was a popular shape averaging method in the past as it is used to find alignment between two time series and calculating the mean to obtain the shape averaging time series. Although the DTW shape averaging is met with a high performance, the order of the time series affects the result shape averaging time series and the performance will vary depending on the order of the time series. Therefore, we have an interest in, DTW barycenter averaging, the shape averaging methods which can perform on multiple time series simultaneously.



DTW barycenter averaging (DBA)

In this section, we discuss the algorithm of the DBA shape averaging method. DBA methods abide by the same idea of the DTW shape averaging method, however DBA will choose a pivot time series to avoid the order of the time series problem. The alignment between the pivot time series and every other input time series is calculated, using DTW, and then calculating the mean based on the obtained alignments. Last but not least, we perform these same steps iteratively to obtain the result time series.

```
def dba(multiple_time_series, iter):
  time_series_idx = find_pivot(multiple_time_series)
  T = multiple_time_series[time_series_idx]

  for i in tqdm(range(iter)):
    T = dba_update(multiple_time_series, T)

  return T

def dba_update(multiple_time_series, pivot):
  alignment = [[] for _ in range(pivot.shape[0])]

  for time_series_idx in range(multiple_time_series.shape[0]):
    path = dtw.warping_path(pivot, multiple_time_series[time_series_idx])
    for i, j in path:
      alignment[i].append(multiple_time_series[time_series_idx, j])

  new_pivot = np.array(list(map(lambda ls: sum(ls)/len(ls), alignment)))

  return new_pivot
```

In this experiment, the pivot was initialized, instead of choosing it. Given the data T is a set of k time series sequences with the length l as follows:

$$T = \{T_1, T_2, T_3, \cdots, T_k\}$$

$$T_n = \{T_{n_1}, T_{n_2}, \cdots T_{n_l}\}$$

Then, we sum up all the values within each time series and calculate the mean of these values. The sequence, closest to the mean, will be opted as the initial pivot.

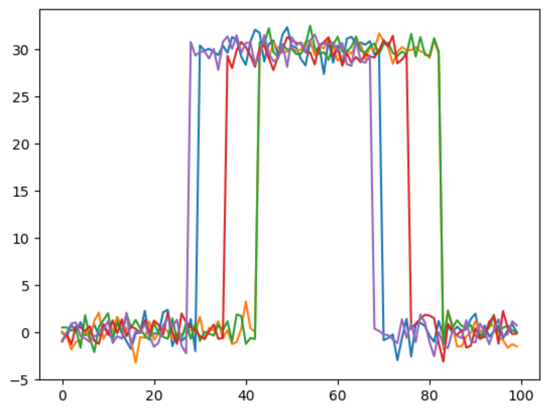
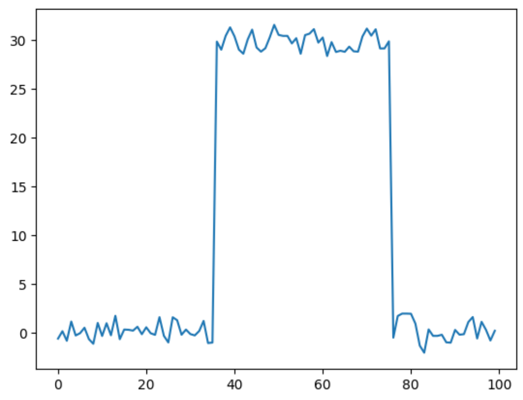
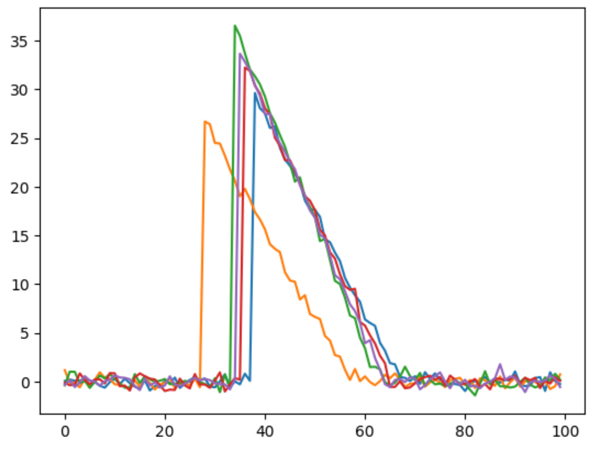
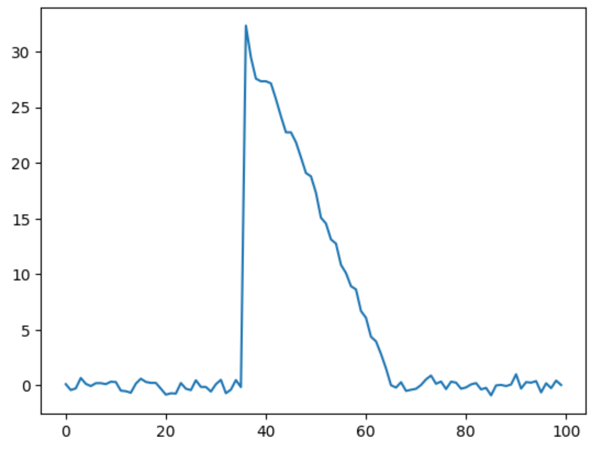
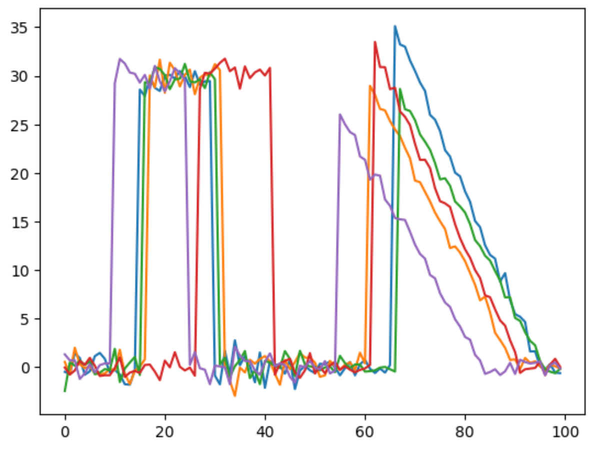
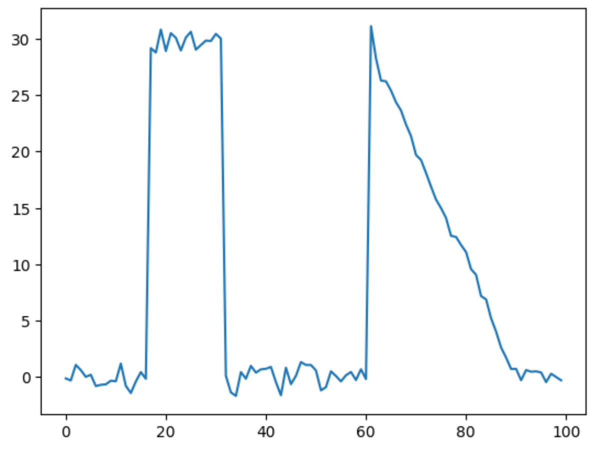
$$S_n = \sum_{i=1}^{l} T_{n_i}$$

$$\text{mean} = \frac{(\sum_{i=1}^{k} S_i)}{k}$$

$$\text{Pivot} = \arg \min_{n \in \{1 \cdots k\}} (|\text{mean} - S_n|)$$

Example result of DBA

In this section, we showed an example result of the DBA shape averaging method. The multiple time series with similar shape were generated and these time series, however, have a different phase, frequency, and amplitude. Furthermore, the gaussian noise was added to these time series. We then performed the DBA shape averaging method on the generated time series with 10 iterations.

Input time series	Result time series
	
	
	

Evaluation

In this section, we evaluated the DBA shape averaging method, compared with amplitude shape averaging method, on multiple dataset, including, Gun Point, Lighting7, synthetic control, and Coffee dataset. The two metrics were used to appraise the DBA method: 1) one nearest neighbor accuracy 2) percentage error. The percentage error was adduced from *Fast and accurate template averaging for time series classification* paper and it can be calculated by using the following equation, given Q and C are original time series and X is a result of averaging time series.

$$\mathrm{PercentageError}(Q, C, X) = \frac{D_{\mathrm{DTW}}(Q, X) - D_{\mathrm{DTW}}(C, X)}{\max\{D_{\mathrm{DTW}}(Q, X), D_{\mathrm{DTW}}\}}$$

One nearest neighbor accuracy

Dataset	DBA	Amplitude
Gun Point	66%	52%
Lighting7	75.71%	41.43%
synthetic control	100%	78%
Coffee	75%	75%

The mean and standard deviation of percentage error of all possible pair

Dataset	DBA	Amplitude
Gun Point	0.1046 +- 0.1012	0.0714 +- 0.0741
Lighting7	0.2408 +- 0.1490	0.1384 +- 0.0972
synthetic control	0.2004 +- 0.1079	0.1623 +- 0.1047
Coffee	0.4040 +- 0.1806	0.0483 +- 0.0441

Discussion

Limitation of the proposed method

The DBA shape averaging methods consist of many hyperparameters to opt for. In addition, our initialization of pivot may not be the most optimal method and it may aggravate the performance, compared to manually choosing the pivot time series. We will leave this topic for further research.

Reference

[1] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 26, no. 1, pp. 43-49, February 1978, doi: 10.1109/TASSP.1978.1163055.

[2] Niennattrakul, Vit & Ratanamahatana, Chotirat. (2007). Inaccuracies of Shape Averaging Method Using Dynamic Time Warping for Time Series Data. Computer. Sci. ICCS 2007. 4487. 513-520. 10.1007/978-3-540-72584-8_68.

[3] P. Sathianwiriyakhun, T. Janyalikit and C. A. Ratanamahatana, "Fast and accurate template averaging for time series classification," 2016 8th International Conference on Knowledge and Smart Technology (KST), Chiang Mai, Thailand, 2016, pp. 49-54, doi: 10.1109/KST.2016.7440530.