Lecture 8. Clustering Algorithms

Zhonghai Lu

KTH Royal Institute of Technology

May 3, 2024

Agenda

1. Supervised learning vs. Unsupervised learning
2. K-means clustering
3. Hierarchical clustering
4. Dynamic Time Warping (DTW)
5. Clustering examples

Supervised learning

Given a set of p features $X_1, X_2. \cdots, X_p$ measured on n observations, and a response Y also measured on those same n observations, the goal is then to predict Y using $X_1, X_2. \cdots, X_p$.

- Classification for discrete responses;
- Regression for continuous responses such as time series, real values.

Unsupervised learning

Given a set of p features $X_1, X_2. \cdots, X_p$ measured on n observations.

- We are not interested in prediction, because we do not have an associated response variable Y.
- Clustering: a broad class of methods for discovering unknown subgroups in data.

Clustering

- Clustering refers to a very broad set of techniques for finding subgroups, or clusters, in a data set.
- We seek to partition them into distinct groups so that the observations within each group are quite similar to each other, while observations in different groups are quite different from each other.
- Clustering is classification without labels (ground truth).

A big family of clustering methods from scikit-learn

Method name	Parameters	Scalability	Use case	Geometry (metric used)
K-Means	number of clusters	Very large n_samples, medium n_clusters with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium n_samples, small n_clusters	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large n_samples, medium n_clusters	Non-flat geometry, uneven cluster sizes	Distances between nearest points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large n_clusters and n_samples	Large dataset, outlier removal, data reduction.	Euclidean distance between points

Common clustering approaches

We focus on perhaps the two best-known clustering approaches: K-means clustering and Hierarchical clustering.

- In K-means clustering, we seek to partition the observations into a pre-specified number K of clusters, which are distinct and non-overlapping.
- In hierarchical clustering, we do not know in advance how many clusters we want; in fact, we end up with a tree-like visual representation of the observations, called a *dendrogram*, that allows us to view at once the clusterings obtained for each possible number of clusters, from 1 to n.

K-means clustering

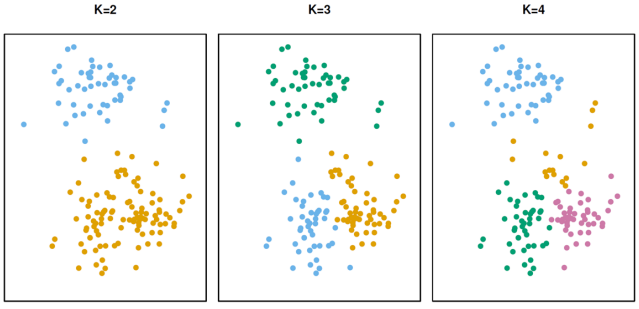


Figure: K-means clustering example

- A simulated data set with 150 observations in two-dimensional space.
- The results of applying K-means clustering with different values of K, the number of clusters.
- The color of each observation indicates the cluster to which it was assigned.
- There is no ordering of the clusters. The cluster coloring is arbitrary.
- These cluster labels were not used in clustering; instead, they are the outputs of the clustering procedure.

How it works?

The idea behind K-means clustering is that a good clustering C_k is one for which the within-cluster variation W is as small as possible.

$$\min_{C_1, C_2, \cdots, C_k} \sum_{k=1}^{K} W(C_k)$$

W measures the amount by which the observations within a cluster differ from each other.

How it works?

- One most common choice involves squared Euclidean distance.
- W is the sum of all of the pairwise squared Euclidean distances between the observations in the kth cluster, divided by the total number of observations $|C_k|$ in the kth cluster.

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

How to solve the optimization problem?

Partition the observations into K clusters such that the objective is minimized.

- There are almost K^n ways to partition n observations into K clusters.
- A simple algorithm can be shown to provide a local optimum—a pretty good solution—to the optimization problem.

A heuristic

The algorithm is guaranteed to decrease the value of the objective at each step.

- Initial cluster assignment: Randomly assign a number from 1 to K to each of the observations.
- Iterate the Expectation-Maximization loop until the assignments stop changing.
  - **Expectation**: For each of the K clusters, compute the cluster centroid. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster.
  - **Maximization**: Assign each observation to the cluster whose centroid is closest. The closest is defined by the chosen distance measure, in this case, the Euclidean distance.

Progress illustration

The progress of the K-means algorithm on the example with K=3.

- Top left: the observations are shown.
- Top center: in Step 1 of the algorithm, each observation is randomly assigned to a cluster.
- Top right: in Step 2(a) E-step, the cluster centroids are computed. These are shown as large colored disks. Initially, the centroids are almost completely overlapping because the initial cluster assignments were chosen at random.
- Bottom left: in Step 2(b) M-step, each observation is assigned to the nearest centroid.
- Bottom center: Step 2(a) E-step is once again performed, leading to new cluster centroids.
- Bottom right: the results obtained after ten Expectation-Maximization iterations.

Progress illustration

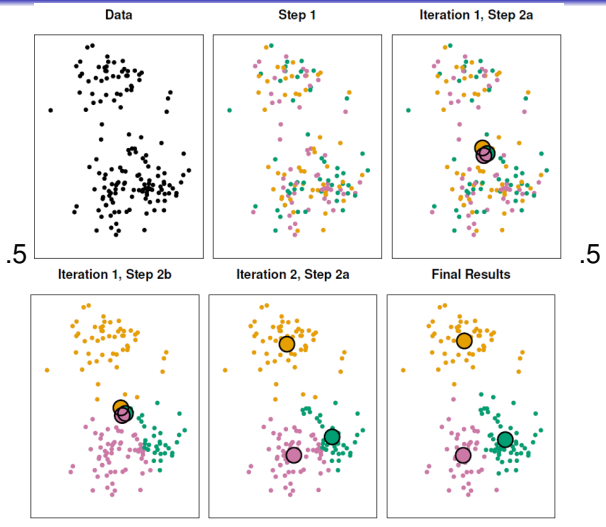


Figure: The progress of the K-means algorithm on the example with K=3.

Effect of initial random assignment

- K-means clustering performed six times on the data with K = 3, each time with a different random assignment of the observations in Step 1 of the K-means algorithm. Above each plot is the value of the objective.
- Three different local optima were obtained, one of which resulted in a smaller value of the objective and provides better separation between the clusters.
- Those labeled in red all achieved the same best solution, with an objective value of 235.8.

Figure: The effect of initialization

Hierarchical clustering

- One potential disadvantage of K-means clustering is that it requires us to pre-specify the number of clusters K.
- Hierarchical clustering is an alternative approach which does not require that we commit to a particular choice of K.
- Hierarchical clustering has an added advantage over K-means clustering in that it results in an attractive tree-based representation of the observations, called a dendrogram.

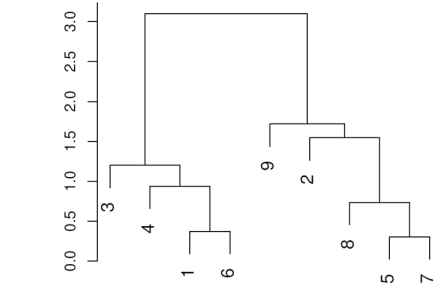
Agglomerative clustering

By bottom-up or agglomerative clustering, which is the most common type of hierarchical clustering, a dendrogram (generally depicted as an upside-down tree) is built starting from the leaves and combining clusters up to the trunk.

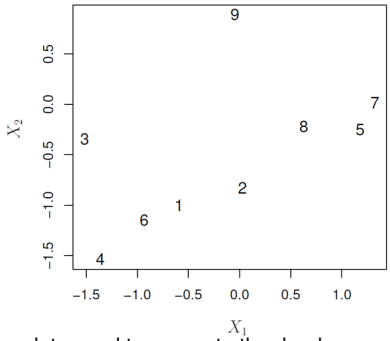
- In dendrogram, each leaf of the dendrogram represents one of the observations. As we move up the tree, some leaves begin to fuse into branches. These correspond to observations that are similar to each other.
- For any two observations, we can look for the point in the tree where branches containing those two observations are first fused.
  The height of this fusion, as measured on the vertical axis, indicates how different the two observations are.
- Thus, observations that fuse at the very bottom of the tree are quite similar to each other, whereas observations that fuse close to the top of the tree will tend to be quite different.

Dendrogram

An illustration of how to properly interpret a dendrogram.



A dendrogram generated using Euclidean distance and complete linkage. Observations 5 and 7 are quite similar to each other, as are observations 1 and 6. However, observation 9 is not more similar to observation 2 than it is to observations 8, 5, and 7, even though observations 9 and 2 are close together in terms of horizontal distance. This is because observations 2, 8, 5, and 7 all fuse with observation 9 at the same height, approximately 1.8.



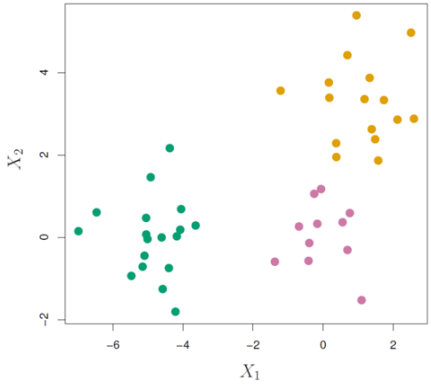
The 9 raw data used to generate the dendrogram confirm that indeed, observation 9 is not more similar to observation 2 than it is to observations 8, 5, and 7.

Dendrogram-based clustering

- To identify clusters on the basis of a dendrogram, we make a horizontal cut across the dendrogram.
- The distinct sets of observations beneath the cut can be interpreted as clusters.

Dendrogram-based clustering

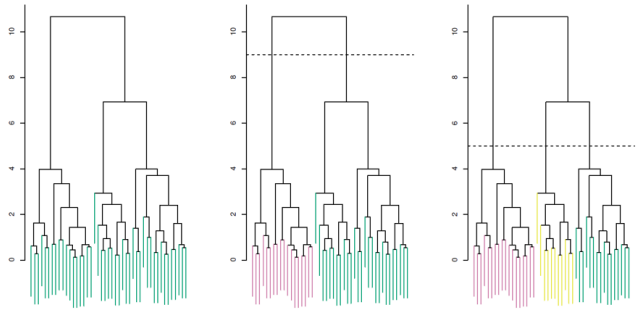
Data set: 45 observations generated in 2D space.



In reality there are three distinct classes, shown in separate colors. However, we will treat these class labels as unknown and will seek to cluster the observations in order to discover the classes from the data.

Dendrogram-based clustering

Left: Dendrogram obtained from hierarchically clustering the data with complete linkage and Euclidean distance.



Center: the dendrogram cut at a height of nine (indicated by the dashed line). This cut results in two distinct clusters, shown in different colors.
Right: the dendrogram cut at a height of five, resulting in three distinct clusters, shown in different colors. Note that the colors were not used in clustering, but are simply used for display purposes.

Hierarchical clustering algorithm

The algorithm proceeds iteratively.

- Starting out at the bottom of the dendrogram, each of the n observations is treated as its own cluster. The 2 clusters most similar to each other are then fused so that there are now $n - 1$ clusters.
- Next the 2 clusters most similar to each other are fused again, so that there now are $n - 2$ clusters.
- The algorithm proceeds in this fashion until all of the observations belong to one single cluster, and the dendrogram is complete.

Hierarchical clustering algorithm

The algorithm proceeds iteratively.

- Begin with n observations and a measure (such as the Euclidean distance) of all the pairwise dissimilarities. Treat each observation as its own cluster.
- For $i = 1, 2, \cdots, n$:
  (a) Examine all pairwise inter-cluster dissimilarities among the i clusters, and identify the pair of clusters that are least dissimilar or most similar.
  (b) Fuse these two clusters. The dissimilarity between these two clusters indicates the height in the dendrogram at which the fusion should be placed.
  (c) Compute the new pairwise inter-cluster dissimilarities among the $i - 1$ remaining clusters

Illustration

- Top Left: initially, there are nine distinct clusters, 1, 2, . . . , 9.
- Top Right: the two clusters that are closest, 5 and 7, are fused into a single cluster.
- Bottom Left: the two clusters that are closest, 6 and 1, are fused into a single cluster.
- Bottom Right: the two clusters that are closest using complete linkage, 8 and the cluster 5, 7, are fused into a single cluster.
  How did we determine that the cluster 5, 7 should be fused with the cluster 8?

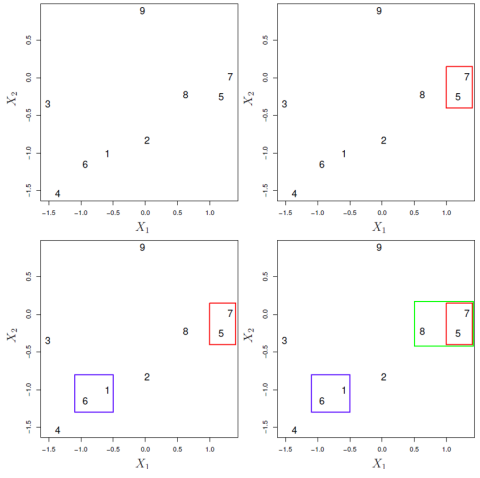


Figure: The impact of linkage

Linkage

- We have a concept of the dissimilarity between pairs of observations, but how do we define the dissimilarity between two clusters if one or both of the clusters contains multiple observations?
- The concept of dissimilarity between a pair of observations needs to be extended to a pair of groups of observations.
- This extension is achieved by developing the notion of linkage, which defines the dissimilarity between two groups of observations.

Linkage summary

A summary of the four most commonly used types of linkage in hierarchical clustering.

Linkage	*Description*
Complete	Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
Single	Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time.
Average	Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable *inversions*.

Figure: The summary of linkage

Impact of Linkage

Dendrogram typically depends quite strongly on the type of linkage used.

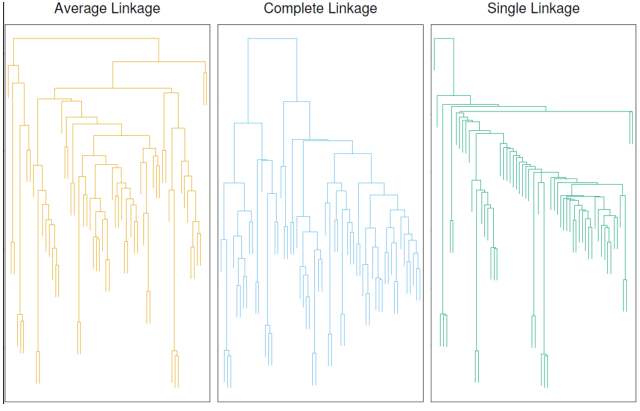


Figure: The impact of linkage

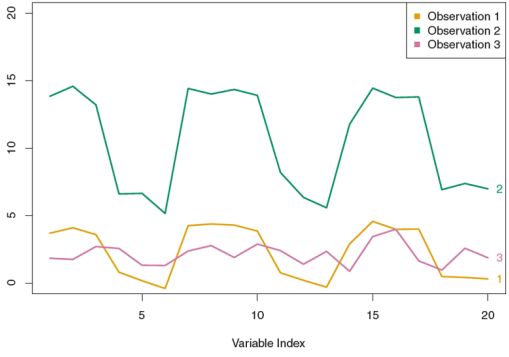
Dissimilarity measures

- Euclidean distance.
- Correlation-based distance considers two observations to be similar if their features are highly correlated, even though the observed values may be far apart in terms of Euclidean distance.
  E.g. Correlation-based distance focuses on the shapes of observation profiles rather than their magnitudes.

Example

Three observations with measurements on 20 variables are shown.

- Observations 1 and 3 have similar values for each variable and so there is a small Euclidean distance between them. But they are very weakly correlated, so they have a large correlation-based distance.
- Observations 1 and 2 have quite different values for each variable, and so there is a large Euclidean distance between them. But they are highly correlated, so there is a small correlation-based distance between them.



Other issues

- Dissimilarity measures
- Data scaling
  Consider whether or not the variables should be scaled to have standard deviation one before the dissimilarity between the observations is computed.
- Soft clustering
  Since K-means and hierarchical clustering force every observation into a cluster, the clusters found may be heavily distorted due to the presence of outliers that do not belong to any cluster.
  Mixture models are an attractive approach for accommodating the presence of such outliers. These amount to a soft version of K-means clustering.
- Robustness
  Clustering methods generally are not very robust to perturbations to the data. For instance, suppose that we cluster n observations, and then cluster the observations again after removing a subset of the n observations at random. One would hope that the two sets of clusters obtained would be quite similar, but often this is not the case!

Practical issues (1)

In the case of K-means clustering, how many clusters should we look for in the data?

- Each of these decisions can have a strong impact on the results obtained. In practice, we try several different choices, and look for the one with the most useful or interpretable solution.
- With these methods, there is no single right answer—any solution that exposes some interesting aspects of the data should be considered.

Practical issues (2)

In order to perform clustering, some decisions must be made.

- Should the observations or features first be standardized in some way? For instance, maybe the variables should be centered to have mean zero and scaled to have standard deviation one.
- In the case of hierarchical clustering,
  - What dissimilarity measure should be used?
  - What type of linkage should be used?
  - Where should we cut the dendrogram in order to obtain clusters?

Distance metrics used in the Euclidean space

Let two variables $\overrightarrow{x}$ and $\overrightarrow{y}$ be vectors of length n; x_i and y_i the ith values of x and y.

- Euclidean distance.

$$d(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- Manhattan distance (City block distance).

$$d(\vec{x}, \vec{y}) = \sum_{i=1}^{n} |x_i - y_i|$$

- Maximum distance: the maximum value of the distances of the attributes.

$$d(\vec{x}, \vec{y}) = \max_{1 \leq i \leq n} |x_i - y_i|$$

Distance metrics used in the Euclidean space

Let two variables $\overrightarrow{x}$ and $\overrightarrow{y}$ be vectors of length n; x_i and y_i the ith values of x and y.

- Minkowski distance, called L_p-norm.

$$d(\vec{x}, \vec{y}) = \sqrt[p]{\sum_{i=1}^{n}(x_i - y_i)^p}$$

The Euclidean distance ($p = 2$), Manhattan distance ($p = 1$), and maximum distance ($p = \infty$).

- Mahalanobis distance

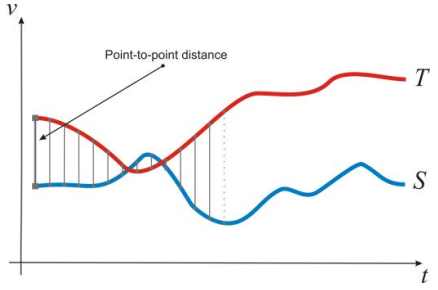
$$d(\vec{x}, \vec{y}) = \sqrt{(x - y)C^{-1}(x - y)^T}$$

where C is the covariance matrix.

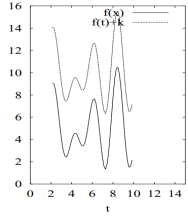
Weakness of Euclidean distance

Euclidean distance and its variants present several weaknesses for measuring similarity of time-series signals.

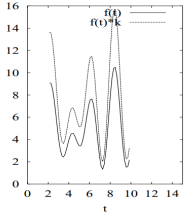
- It compares only time series of the same length.
- It is very sensitive with respect to six signal transformations: shifting, uniform amplitude scaling, uniform time scaling, uniform bi-scaling, time warping and non-uniform amplitude scaling.
- It does not handle outliers or noise.



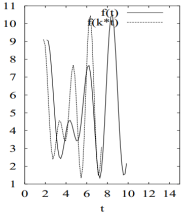
Signal transformations



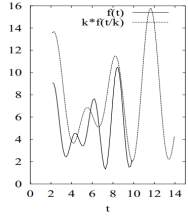
(a) Shifting



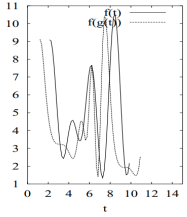
(b) Uniform Amplitude Scaling



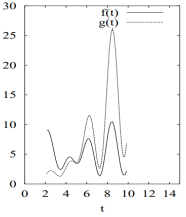
(c) Uniform Time Scaling



(d) Uniform Bi-scaling



(e) Time Warping



(f) Non-uniform Amplitude Scaling

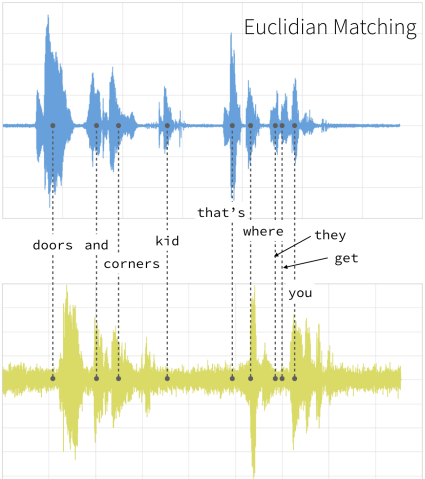
C. S. Perng, H. Wang, S. R. Zhang and D. S. Parker, "Landmarks: a new model for similarity-based pattern querying in time series databases," 16th International Conference on Data Engineering, 2000, pp. 33-42.

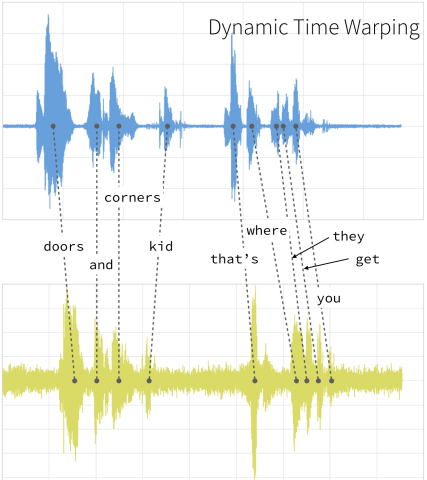
Dynamic time warping (DTW) distance

- Dynamic Time Warping (DTW) gives more robustness to the similarity computation.
- DTW does not do uniform point-to-point pairing but allows many-to-one and one-to-many pairing when computing distance.
- The main strength of this distance measure is that it allows to compare signals of different length, and recognize similar shapes, even if they present signal transformations, such as shifting and/or scaling.

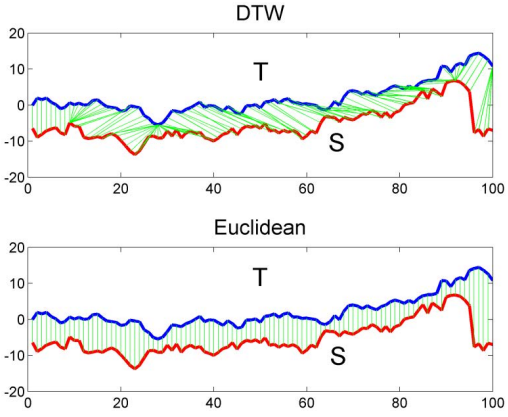
Why DTW? Euclidean vs. DTW

How similar are two audio clips saying the same words, but different delays among the words?





Euclidean vs. DTW



DTW definition

Given two time series $T = \{t_1, t_2, ..., t_n\}$ and $S = \{s_1, s_2, ..., s_m\}$ of length n and m, respectively, define a $n \times m$ distance matrix:

$$distMatrix = \begin{bmatrix} d(T_1, S_1) & d(T_1, S_2) & \cdots & d(T_1, S_m) \\ d(T_2, S_1) & d(T_2, S_2) & \cdots & d(T_2, S_m) \\ \cdots & & & \\ \cdots & & & \\ d(T_n, S_1) & d(T_n, S_2) & \cdots & d(T_n, S_m) \end{bmatrix}$$

where $distMatrix(i, j)$ corresponds to the Euclidean distance of ith point of T and jth point of S, $d(T_i, S_j)$, where $1 \leq i \leq n$ and $1 \leq j \leq m$.

DTW objective

The objective of DTW is to find the warping path $W = \{w_1, w_2, \cdots, w_k, \cdots, w_K\}$ of contiguous elements in *distMatrix*, $\max(n, m) < K < m + n - 1$, and $w_k = distMatrix(i, j)$, such that it minimizes the following function

$$DTW(T, S) = min\left(\sqrt{\sum_{k=1}^{K} w_k}\right)$$

DTW: Constraints

The warping path is subject to several constraints.
Given $w_k = (i, j)$ and $w_{k-1} = (i', j')$ with i, $i' \leq n$ and j, $j' \leq m$:

- Boundary conditions. $w_1 = (1, 1)$ and $w_K = (n, m)$.
- Continuity. i–$i' \leq 1$ and j–$j' \leq 1$.
- Monotonicity. i–$i' \geq 0$ and j–$j' \geq 0$.

The warping path

The warping path can be efficiently computed using dynamic programming.

- A cumulative distance matrix γ of the same dimension as the *distMatrix*, is created to store in the cell (i, j) the following value:

$$\gamma(i,j) = d(T_i, S_j) + min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\}$$

- The overall complexity is relative to the computation of all distances in *distMatrix*, i.e., $O(nm)$.
- The last element of the warping path, w_K corresponds to the distance calculated with the DTW method.

The warping path

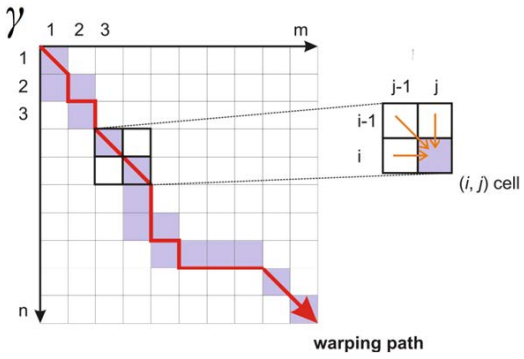


Figure: Warping path computation using dynamic programming. The lavender cells corresponds to the warping path. The red arrow indicates its direction.

- The warping distance at the (i, j) cell will consider, besides the distance between T_i and S_j, the minimum value among adjacent cells at positions: $(i-1, j-1)$, $(i-1, j)$ and $(i, j-1)$.
- The Euclidean distance between two time series can be seen as a special case of DTW, where the path's elements belong to the γ matrix diagonal.

DTW calculation example

Algorithm steps: (1) Compute distance matrix, (2) Compute accumulated distance matrix using dynamic programming, (3) find a warping path, thus alignment (pairing) pattern.

	3	5	1	3
2	1	9	1	1
1	4	16	0	4
5	4	0	16	4
4	1	1	9	1

	3	5	1	3
2	1	10	11	12
1	5	17	10	14
5	9	5	21	14
4	10	6	14	**15**

	3	5	1	3
2	1	10	11	12
1	5	17	10	14
5	9	5	21	14
4	10	6	14	**15**

(a) (b) (c) (d)

Figure: An example of non-unique warping paths.

(a) Time series x and y. (b) Local distance (cost) matrix, $(x_i - y_i)^2$. (c) Accumulated cost gives the DTW distance. DTW(x,y)=$\sqrt{15}$. The grey shaded cells show 2 different optimal warping paths. (d) Alignments of x and y by the two warping paths. Source: Optimal Warping Paths are unique for almost every Pair of Time Series by Brijnesh J. Jain and David Schultz.

Yet another simple example

Calculate the distance of the two sequences:

$$x = \{3, 1, 2, 2, 1\}$$

$$y = \{2, 0, 0, 3, 3, 1, 0\}$$

- What is the Euclidean distance of the two series?
- What is the DTW distance of the two series?

The example is from "An Illustrative Introduction to Dynamic Time Warping". https://towardsdatascience.com/an-illustrative-introduction-to-dynamic-time-warping-36aa98513b98

A simple example: Euclidean distance

How to deal with the no-matching points?

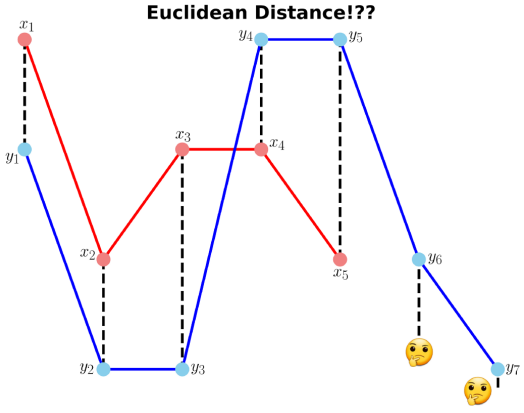
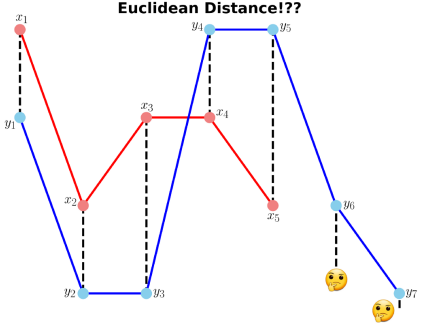
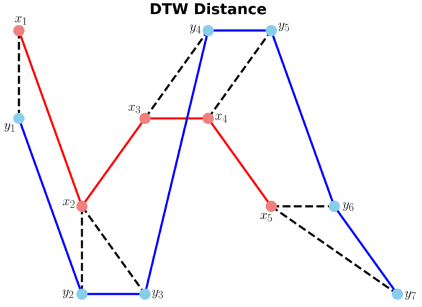


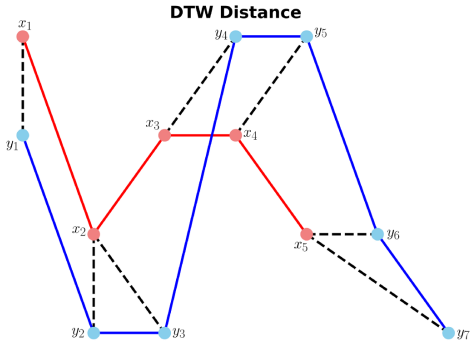
Figure: Sequences with different lengths

A simple example: Euclidean distance vs. DTW distance





A simple example: How to calculate the DTW distance?



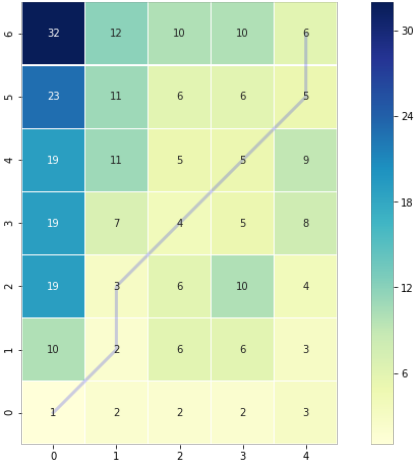
- Use the Python package fastdtw
  pip install fastdtw

- Compute first the squared Euclidean distance matrix for all-to-all pairs.
- Based on the distance matrix, compute the accumulated cost matrix while obtaining the warping path.

```
eu_dist=euclidean_distance_matrix(x,y)
print(eu_dist)
[[1. 1. 0. 0. 1.]
 [9. 1. 4. 4. 1.]
 [9. 1. 4. 4. 1.]
 [0. 4. 1. 1. 4.]
 [0. 4. 1. 1. 4.]
 [4. 0. 1. 1. 0.]
 [9. 1. 4. 4. 1.]]
```

```
dtw_distance, warp_path=fastdtw(x,y,dist=euclidean)
>>> dtw_distance: 6.0
>>> warp_path: [(0,0),(1,1),(1,2),(2,3),(3,4),(4,5),(4,6)]
```

A simple example: Warping path and visualization



- Visualize the accumulated cost matrix along with the warping path.
- The value of the ending cell on the warping path is the DWT(x,y) distance.

```
cost_matrix=accumulated_cost_matrix(x,y)
print(np.flipud(cost_matrix))
# Flipping the cost matrix as heatmap values.
>>> [[32. 12. 10. 10.  6.]
     [23. 11.  6.  6.  5.]
     [19. 11.  5.  5.  9.]
     [19.  7.  4.  5.  8.]
     [19.  3.  6. 10.  4.]
     [10.  2.  6.  6.  3.]
     [ 1.  2.  2.  2.  3.]]
```

Figure: Accumulated cost matrix and warping path

An example of DTW alignment (1)

- DTW algorithm has become popular being very efficient as the time-series similarity measure which minimizes the effects of shifting and distortion in time.
- It allows “elastic” transformation of time series in order to detect similar shapes with different phases.

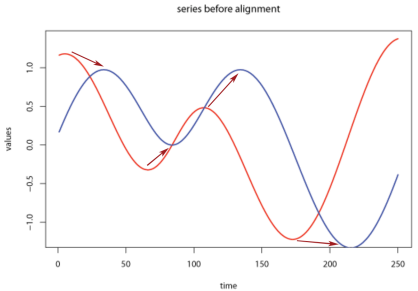


Figure: Two raw time series. Arrows show the desirable points of alignment.

Pavel Senin. ”Dynamic Time Warping Algorithm Review”, December 2008.

An example of DTW alignment (2)

Illustration of the optimal alignment of the two time series.

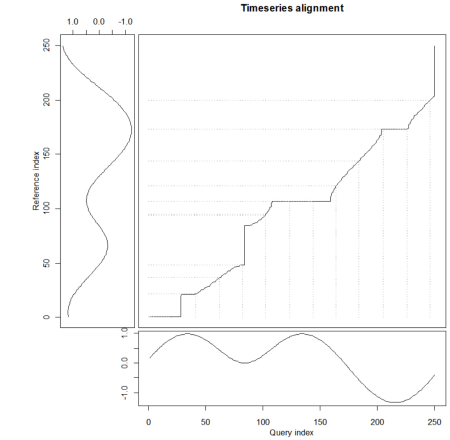


Figure: The optimal warping path aligning time

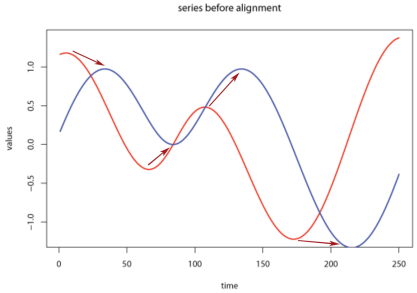


Figure: Two raw time series. Arrows show the desirable points of alignment.

Left figure: The lower curve is the blue series while the left curve is the red series.

An example of DTW alignment (3)

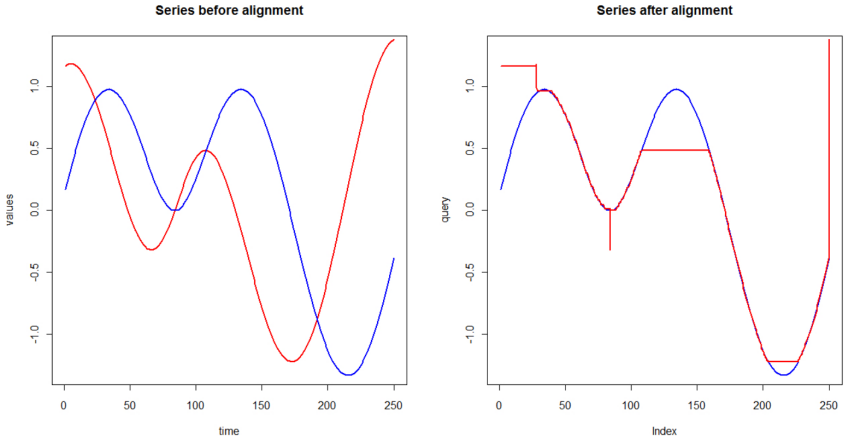


Figure: The optimal alignment of the time series: before vs. after alignment

DTW references

- Understanding Dynamic Time Warping. https://databricks.com/blog/2019/04/30/understanding-dynamic-time-warping.html
- Programatically understanding dynamic time warping (DTW). https://nipunbatra.github.io/blog/ml/2014/05/01/dtw.html#Visualizing-the-distance-matrix
- An Illustrative Introduction to Dynamic Time Warping. https://towardsdatascience.com/an-illustrative-introduction-to-dynamic-time-warping-36aa98513b98
- DTW function in Matlab https://www.mathworks.com/help/signal/ref/dtw.html
- Cassisi, Carmelo & Montalto, Placido & Aliotta, Marco & Cannata, Andrea & Pulvirenti, Alfredo. (2012). "Similarity Measures and Dimensionality Reduction Techniques for Time Series Data Mining". In Chapter 3 of book "Advances in Data Mining Knowledge Discovery and Applications".

Clustering examples

- Time-series clustering
- Clustering in Scikit-learn
- Clustering demo

Time series clustering

Given multiple time series, clustering means partitioning the series into different groups.

- $\vec{X} = x_1, x_2, \cdots, x_n$
- $\vec{Y} = y_1, y_2, \cdots, y_n$
- $\vec{Z} = z_1, z_2, \cdots, z_n$

A time-series is treated as a multi-dimensional vector.

Clustering in Scikit-learn

Clustering of unlabeled data can be performed with the module sklearn.cluster.

- sklearn.cluster.Kmeans
- sklearn.cluster.AgglomerativeClustering

Comparison of the K-Means and MiniBatchKMeans clustering algorithms
Hierarchical clustering: structured vs unstructured ward
`http://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering`

sklearn.cluster.Kmeans

```
>>> from sklearn.cluster import KMeans
>>> import numpy as np
>>> X = np.array([[1, 2], [1, 4], [1, 0],
...               [4, 2], [4, 4], [4, 0]])
>>> kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
>>> kmeans.labels_
array([0, 0, 0, 1, 1, 1], dtype=int32)
>>> kmeans.predict([[0, 0], [4, 4]])
array([0, 1], dtype=int32)
>>> kmeans.cluster_centers_
array([[ 1.,  2.],
       [ 4.,  2.]])
```

http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

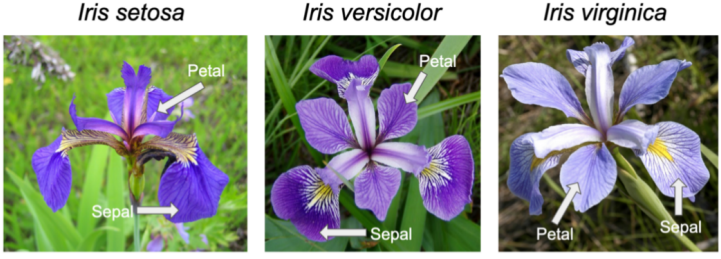
IPython Demo[1]

- A DTW example `http://localhost:8889/notebooks/OneDrive%20-%20KTH/IPython/IL2233VT22/Lec8_clustering/dtw_example.ipynb`
- The simple clustering example `http://localhost:8888/notebooks/IL2233VT22/Lec8_clustering/simple_example.ipynb`
- Clustering the Iris flower data set. `http://localhost:8888/notebooks/IL2233VT22/Lec8_clustering/iris_clustering.ipynb`
- Clustering a time-series data set, BME. `http://localhost:8888/notebooks/IL2233VT22/Lec8_clustering/bme_clustering.ipynb`

[1] The demo code is only for teacher.

How to recognize iris flowers?

- How to identify different iris flowers?



Petals & Sepals for Iris setosa, Iris versicolor, and Iris virginica (Sources: 1, 2, 3, Licenses: Public Domain, CC BY-SA 3.0 & CC BY-SA 2.0)

Figure: Iris flowers

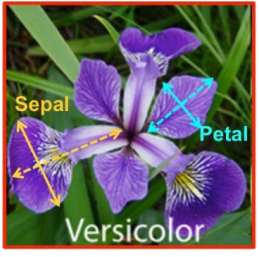


Figure: Measurement

The Iris flower dataset

- The dataset contains a set of 150 records under five attributes - sepal length, sepal width, petal length, petal width and species.
- There are three species, ['setosa' 'versicolor' 'virginica'], each with 50 records.

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	0 (Iris-setosa)
4.9	3.	1.4	0.2	0 (Iris-setosa)
4.7	3.2	1.3	0.2	0 (Iris-setosa)
...	...	...	...	1 (Iris-versicolor)
...	...	...	...	2 (Iris-virginica)
5.9	3.0	5.1	1.8	2 (Iris-virginica)

https://archive.ics.uci.edu/ml/datasets/iris

Iris: Clustering steps

- Load the dataset from sklearn and visualize the data in 2 or 3 dimensions
- Define a K-means model with 3 clusters
- Fit the 3-cluster model to the data set (you can try using 2 or 3 or all 4 features of the data set for clustering)
- Visualize the estimated cluster labeling and true cluster labeling
- Output cluster labels (the cluster labels need to be re-mapped)
- Generate a confusion matrix

`http://localhost:8888/notebooks/IL2233VT22/Lec8_clustering/iris_clustering.ipynb`

BME data from UCI

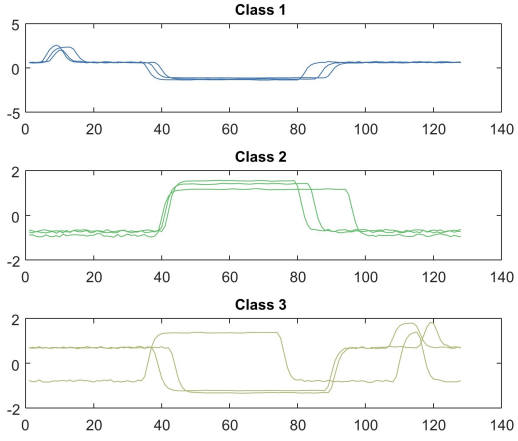
BME is a synthetic data set with three classes. The length of each series is 128.
Train size: 30. Test size: 150. Missing value: No

- One class is characterized by a small positive bell arising at the initial period (Class 1: Begin), one does not have any bell (Class 2: Middle), one has a positive bell arising at the final period (Class 3: End).
- All series are constituted by a central plate. The central plates may be positive or negative. The discriminant is the presence or absence of a positive peak, or at the beginning of series or at the end.

Data source: University Joseph Fourier (Grenoble I). `http://www.timeseriesclassification.com/description.php?Dataset=BME`

BME data from UCR

Class 1: Begin - Class 2: Middle - Class 3: End



References

- Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani. “An Introduction to Statistical Learning: with Applications in R”, 2013, Springer.
  `http://www-bcf.usc.edu/~gareth/ISL/data.html`