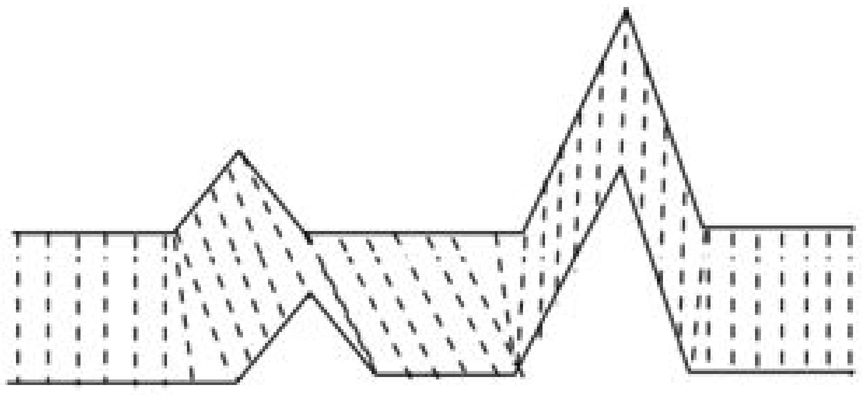
DYNAMIC TIME WARPING (DTW) + HIERARCHICAL CLUSTERING

DYNAMIC TIME WARPING

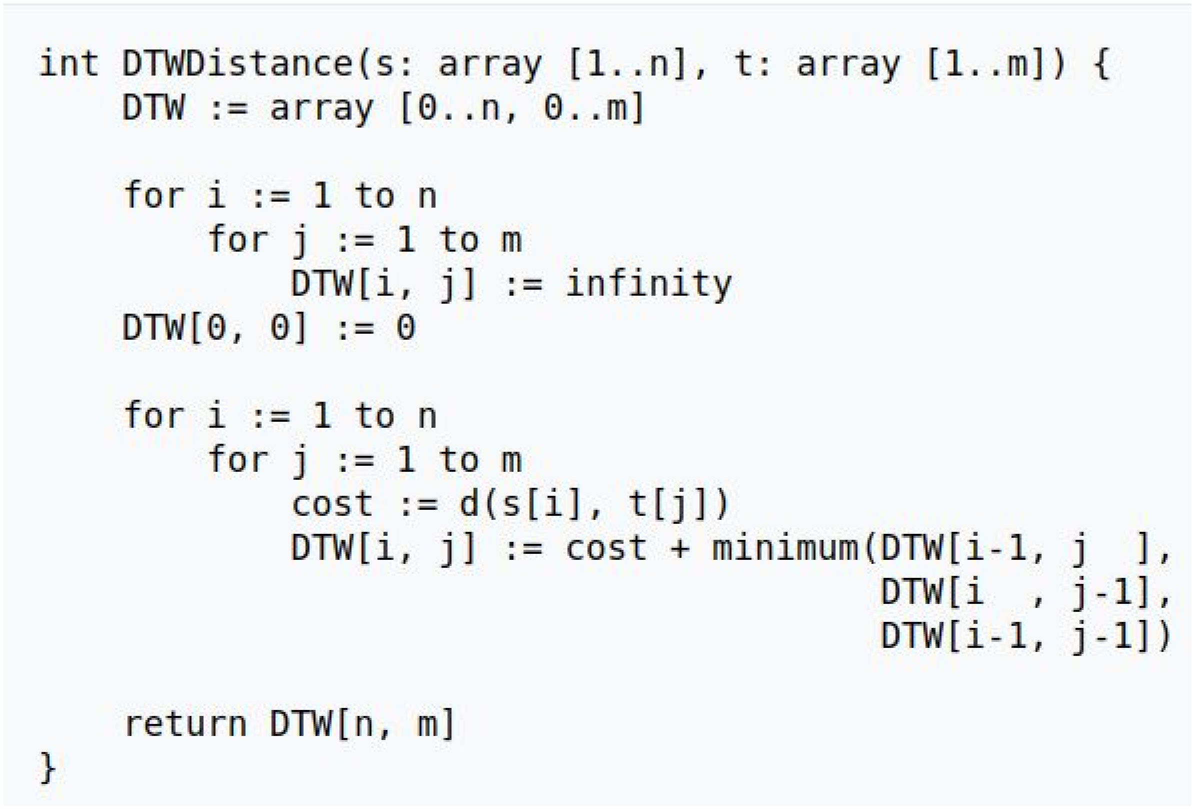
- In time series analysis, dynamic time warping (DTW) is one of the algorithms for measuring a distance metric between two time series, which may vary in speed. For instance, similarities in walking could be detected using DTW, even if one person was walking faster than the other, or if there were accelerations and decelerations during the course of an observation.
- DTW has been applied to time series of video, audio, and graphics data — indeed, any data that can be turned into a linear sequence can be analyzed with DTW.
- The algorithm is subject to certain constraints:
    - Every index from the first sequence must be matched with one or more indices from the other sequence, and vice versa
    - The first index from the first sequence must be matched with the first index from the other sequence (but it does not have to be it's only match)

    - The last index from the first sequence must be matched with the last index from the other sequence (but it does not have to be it's only match)
    - The mapping of the indices from the first sequence to indices from the other sequence must be monotonically increasing, and vice versa, i.e. if j>i are indices from the first sequence, then there must not be two indices l>k in the other sequence, such that index i is matched with index l and index j is matched with index k, and vice versa.
    - An example of dynamic time warping between 2 time series:



- DTW tries to find the optimal match between the 2 time series. The optimal match is denoted by the match that satisfies all the restrictions and the rules and that has the minimal cost, where the cost is computed as the sum of absolute differences or any other distance metric, for each matched pair of indices, between their values.

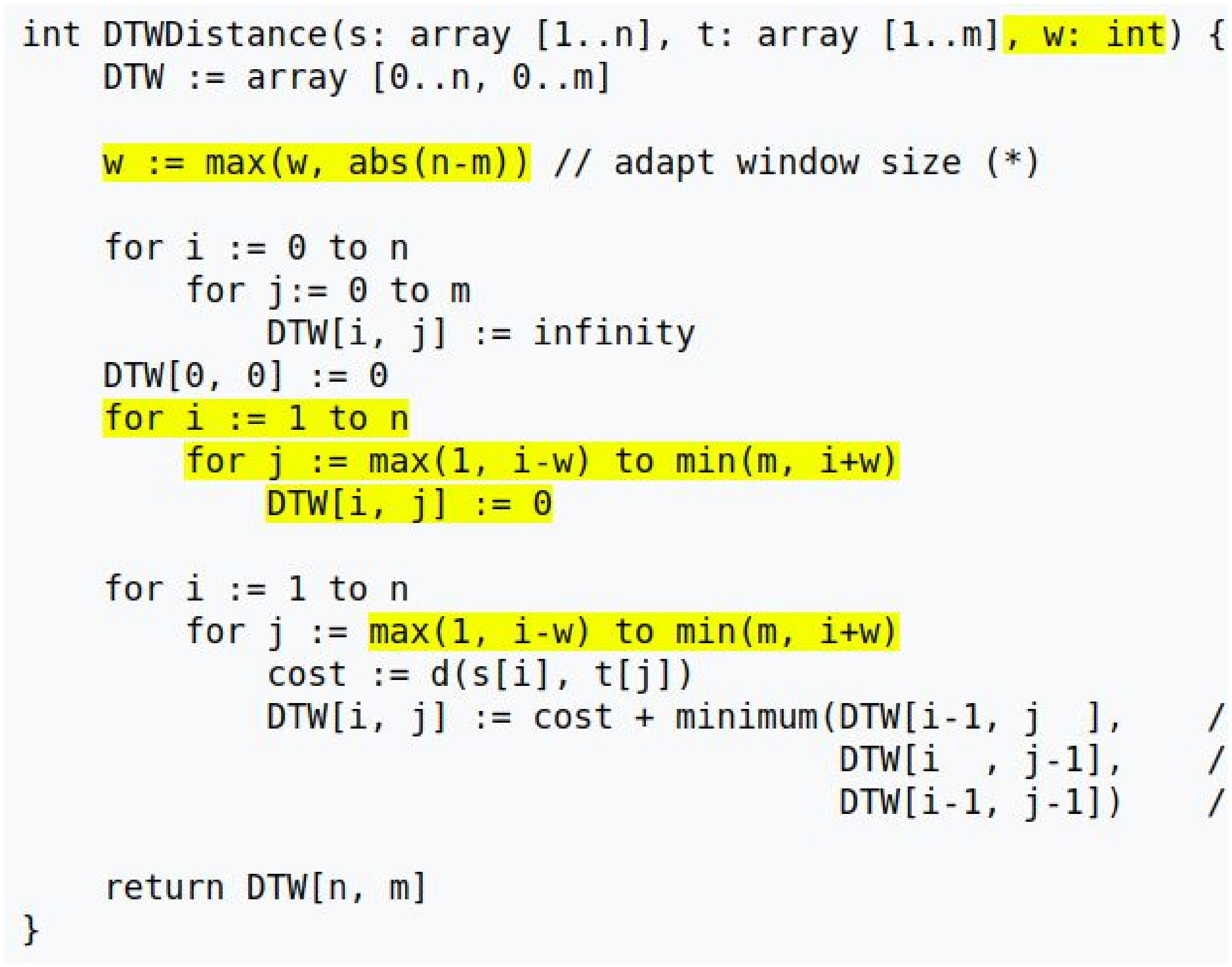
- Although DTW measures a distance-like quantity between two given sequences, it doesn't guarantee the triangle inequality to hold.
- Algorithm to get DTW:



```
int DTWDistance(s: array [1..n], t: array [1..m]) {
    DTW := array [0..n, 0..m]

    for i := 1 to n
        for j := 1 to m
            DTW[i, j] := infinity
    DTW[0, 0] := 0

    for i := 1 to n
        for j := 1 to m
            cost := d(s[i], t[j])
            DTW[i, j] := cost + minimum(DTW[i-1, j  ],
                                        DTW[i  , j-1],
                                        DTW[i-1, j-1])

    return DTW[n, m]
}
```

where DTW[i, j] is the distance between s[1:i] and t[1:j] with the best alignment.

- We sometimes want to add a locality constraint. That is, we require that if s[i] is matched with t[j], then | i - j | is no larger than w, a window parameter. We can easily modify the above algorithm to add a locality constraint as shown below. The changes to the above algorithm are marked in yellow.

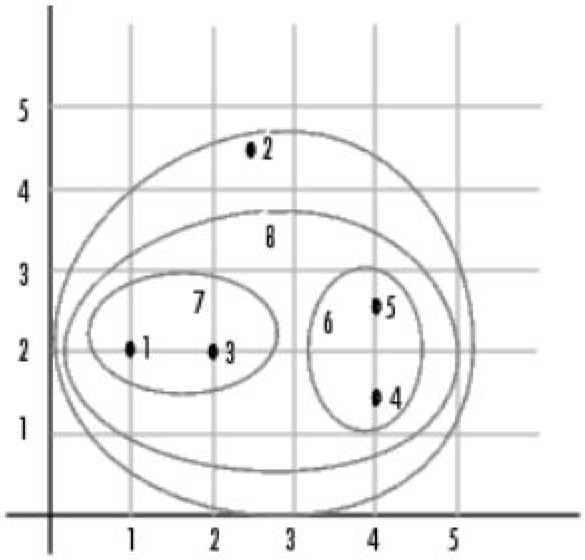


```
int DTWDistance(s: array [1..n], t: array [1..m], w: int) {
    DTW := array [0..n, 0..m]

    w := max(w, abs(n-m)) // adapt window size (*)

    for i := 0 to n
        for j:= 0 to m
            DTW[i, j] := infinity
    DTW[0, 0] := 0
    for i := 1 to n
        for j := max(1, i-w) to min(m, i+w)
            DTW[i, j] := 0

    for i := 1 to n
        for j := max(1, i-w) to min(m, i+w)
            cost := d(s[i], t[j])
            DTW[i, j] := cost + minimum(DTW[i-1, j  ],    /
                                        DTW[i  , j-1],    /
                                        DTW[i-1, j-1])    /

    return DTW[n, m]
}
```

- The time complexity of DTW algorithm is O(NM), where N and M are the lengths of the two input sequences. The same goes for space complexity.

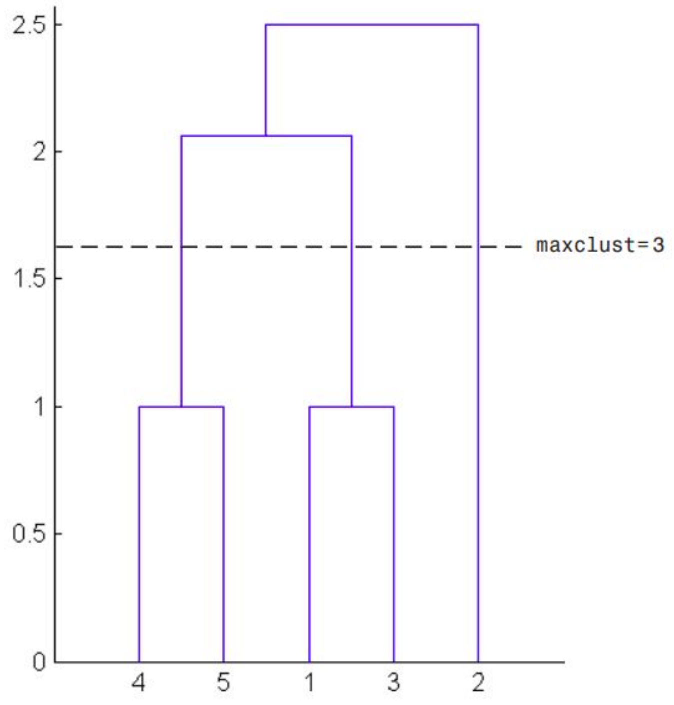
AGGLOMERATIVE HIERARCHICAL CLUSTERING

- Hierarchical clustering groups data over a variety of scales by creating a **cluster tree** or **dendrogram**. The tree is not a single set of clusters, but rather a multilevel hierarchy, where clusters at one level are joined as clusters at the next level. This allows you to decide the level or scale of clustering that is most appropriate for your application. Given below is a figure demonstrating the formation of a cluster tree:



- **ALGORITHM:**
    - Find the dissimilarity (a distance metric) between every pair of objects in the data set.
    - Assume that each point is it’s own cluster.

- Link a pair of clusters that are in closest proximity to form a higher level cluster.
- Repeat the above step until we are left with a single cluster.
- Determine where to cut the hierarchical tree into clusters and assign all the objects below each cut to a single cluster.

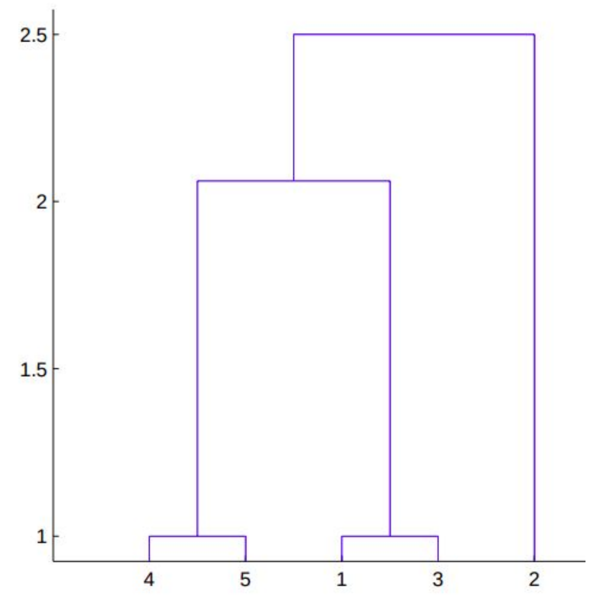


For example, in the above hierarchical tree, the dotted line represents where the tree is cut. We would then end up with 3 sub-trees: {4,5}, {1,3} and {2}, each representing a cluster.

- **DISTANCE MEASURES:** There are many ways to calculate this distance information like Euclidean, Manhattan etc. For the case of time series, we can use Dynamic Time Warping discussed above as the distance measure.

- **LINKAGES:** Once the proximity between objects in the data set has been computed, we can determine how objects in the data set should be grouped into clusters, using a linkage function. Some common linking strategies are:
    - **Single Link Clustering:** The distance between two groups is defined as the distance between the two closest members of each group. The tree built using single link clustering is a minimum spanning tree of the data with edge weights as the distances. So, we can actually implement single link clustering in $O(N^2)$ time, whereas the other variants take $O(N^3)$ time.
    - **Complete Link Clustering:** The distance between two groups is defined as the distance between the two most distant pairs of points. Single linkage can produce clusters with large diameters. Complete linkage represents the opposite extreme: two groups are considered close only if all of the observations in their union are relatively similar. This will tend to produce clusterings with small diameter, i.e., compact clusters.
    - **Average Link Clustering:** Measures the average distance between all pairs. This form of linkage is a compromise between both the above linkage techniques.
- **DENDROGRAMS:** A graphical representation of the hierarchical tree for visualization. In the below figure, the numbers along the horizontal axis represent the indices of the objects in the original data set. The

links between objects are represented as upside-down U-shaped lines. The height of the U indicates the distance between the objects.



- **VERIFICATION OF HIERARCHICAL TREE:** We might want to verify that the distances in the tree reflect the original distances accurately.

  In a hierarchical cluster tree, any two objects in the original data set are eventually linked together at some level. The height of the link represents the distance between the two clusters that contain those two objects. This height is known as the **cophenetic distance** between the two objects.

  If the clustering is valid, the linking of objects in the cluster tree should have a strong correlation with the distances between objects in the distance vector. This correlation is called the **cophenetic correlation coefficient**. The closer the value of the cophenetic correlation coefficient is to 1, the more accurately the clustering solution reflects the data.

- **DECIDING WHERE TO CUT THE TREE:** One way is to compare the height of each link in a cluster tree with the heights of neighboring links below it in the tree.

  A link that is approximately the same height as the links below indicates that there are no distinct divisions between the objects joined at this level of the hierarchy. These links are said to exhibit a high level of **consistency**.

  On the other hand, a link whose height differs noticeably from the height of the links below it indicates that the objects joined at this level in the cluster tree are much farther apart from each other than their components were when they were joined. This link is said to be inconsistent with the links below it.

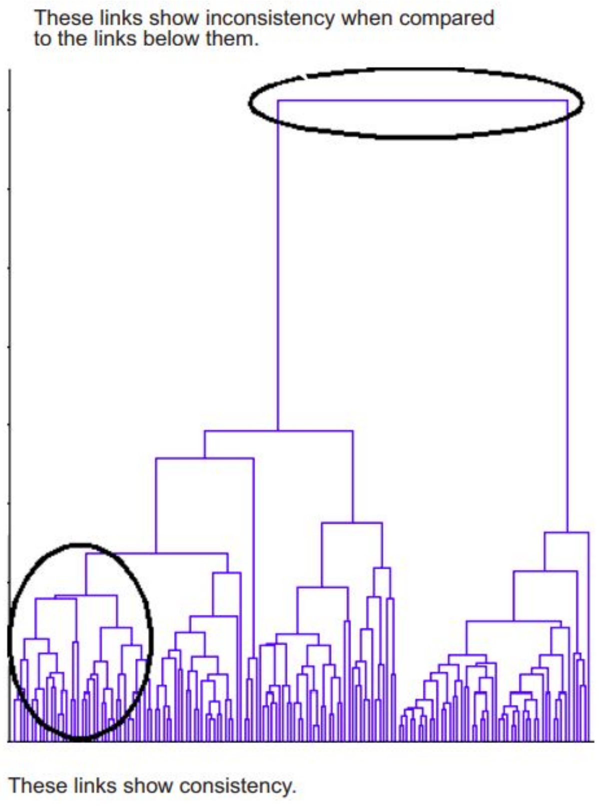
  In cluster analysis, inconsistent links can indicate where to cut the tree. The relative consistency of each link in a hierarchical cluster tree can be quantified and expressed as the **inconsistency coefficient**.

  This function compares each link in the cluster hierarchy with adjacent links that are less than a certain number of levels below it in the cluster hierarchy. This number of levels used to compare is called **depth of the comparison**.

The inconsistency coefficient of a link is calculated as: **[(h - μ) / σ]** where h is the height of the link, μ and σ are the mean and standard deviation of the heights that the link is being compared to.

The objects at the bottom of the cluster tree, called leaf nodes, that have no further objects below them, have an inconsistency coefficient of zero. Clusters that join two leaves also have a zero inconsistency coefficient.

To determine, where to cut the tree, we can decide an upper threshold of the inconsistent coefficient of the links in the tree.


These links show inconsistency when compared to the links below them.
These links show consistency.

- **EVALUATING THE RESULTS:** The **silhouette value** is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.

  For data point $i \in C_i$ (data point i in the cluster C_i), let

  $$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j)$$

  be the mean distance between point i and all other data points in the same cluster, where d(i,j) is the distance between data points i and j in the cluster C(i) (we divide by |C(i)-1| because we do not include the distance d(i,i) in the sum). We can interpret a(i) as a measure of how well the point i is assigned to its cluster.

  For each data point $i \in C_i$, we now define

  $$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j)$$

  to be the smallest mean distance of i to all points in any other cluster, of which i is not a member.

We now define a *silhouette* (value) of one data point i

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{ if } |C_i| > 1$$

and

$$s(i) = 0, \text{ if } |C_i| = 1$$

- **EVALUATION WHEN GROUND TRUTH IS AVAILABLE:**
  - Various metrics are available which measure the similarity between 2 clustering distributions. So, if one clustering distribution is available (like the ground truth), then these metrics can be used to determine how close the predicted clustering is to the ground truth.
  - **Adjusted Rand Index:**
    - Rand Index is calculated as: $R = \frac{a+b}{\binom{n}{2}}$ where a refers to the number of times a pair of elements belongs to a same cluster across two different clustering results, b refers to the number of times a pair of elements are in different clusters across two different clustering results and n is the total no. of points in the dataset.
    - The drawback of Rand Index: If there are a large number of clusters, b will be large even when the clustering is entirely random. This is because, as no. of clusters increase, the likeliness of 2 dis-similar

points to be in a different cluster increases, even when the clustering is way off the mark.

- Contingency Table: Given a set S of n elements, and two clusterings of these elements, namely X = {X_1,X_2,……… X_r} and Y = {Y_1,Y_2,……… Y_s}, the overlap between X and Y can be summarized in a contingency table [n_{ij}] where each entry n_{ij} denotes the number of objects in common between X_i and Y_j

$X \diagdown Y$	Y_1	Y_2	$\cdots$	Y_s	sums
X_1	n_{11}	n_{12}	$\cdots$	n_{1s}	a_1
X_2	n_{21}	n_{22}	$\cdots$	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	n_{r2}	$\cdots$	n_{rs}	a_r
sums	b_1	b_2	$\cdots$	b_s	

- Adjusted Rand Index is:

$$AdjustedIndex = \frac{Index - ExpectedIndex}{MaxIndex - ExpectedIndex}$$

From the above contingency table, this turns out to be

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right] / \binom{n}{2}}$$

- So, the Adjusted Rand Index basically normalizes the normal Rand Index based on the distribution of the 2 clusterings of the dataset.

- **Normalized Mutual Index (NMI):**

Assume two label assignments (of the same N objects), U and V. Their entropy is the amount of uncertainty for a partition set, defined by:

$$H(U) = -\sum_{i=1}^{|U|} P(i)\log(P(i))$$

where $P(i) = |U_i|/N$ is the probability that an object picked at random from U falls into class U_i. Likewise for V:

$$H(V) = -\sum_{j=1}^{|V|} P'(j)\log(P'(j))$$

With $P'(j) = |V_j|/N$. The mutual information (MI) between U and V is calculated by:

$$\text{MI}(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)\log\left(\frac{P(i,j)}{P(i)P'(j)}\right)$$

where $P(i,j) = |U_i \cap V_j|/N$ is the probability that an object picked at random falls into both classes U_i and V_j.

It also can be expressed in set cardinality formulation:

$$\text{MI}(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N}\log\left(\frac{N|U_i \cap V_j|}{|U_i||V_j|}\right)$$

The normalized mutual information is defined as

$$\text{NMI}(U,V) = \frac{\text{MI}(U,V)}{\text{mean}(H(U),H(V))}$$

- **Adjusted Mutual Index (AMI):** Just like Rand Index, NMI may give mis-leading results when no. of clusters are high. So, there is a need to normalize the NMI with respect to the distribution of the 2 clusters:

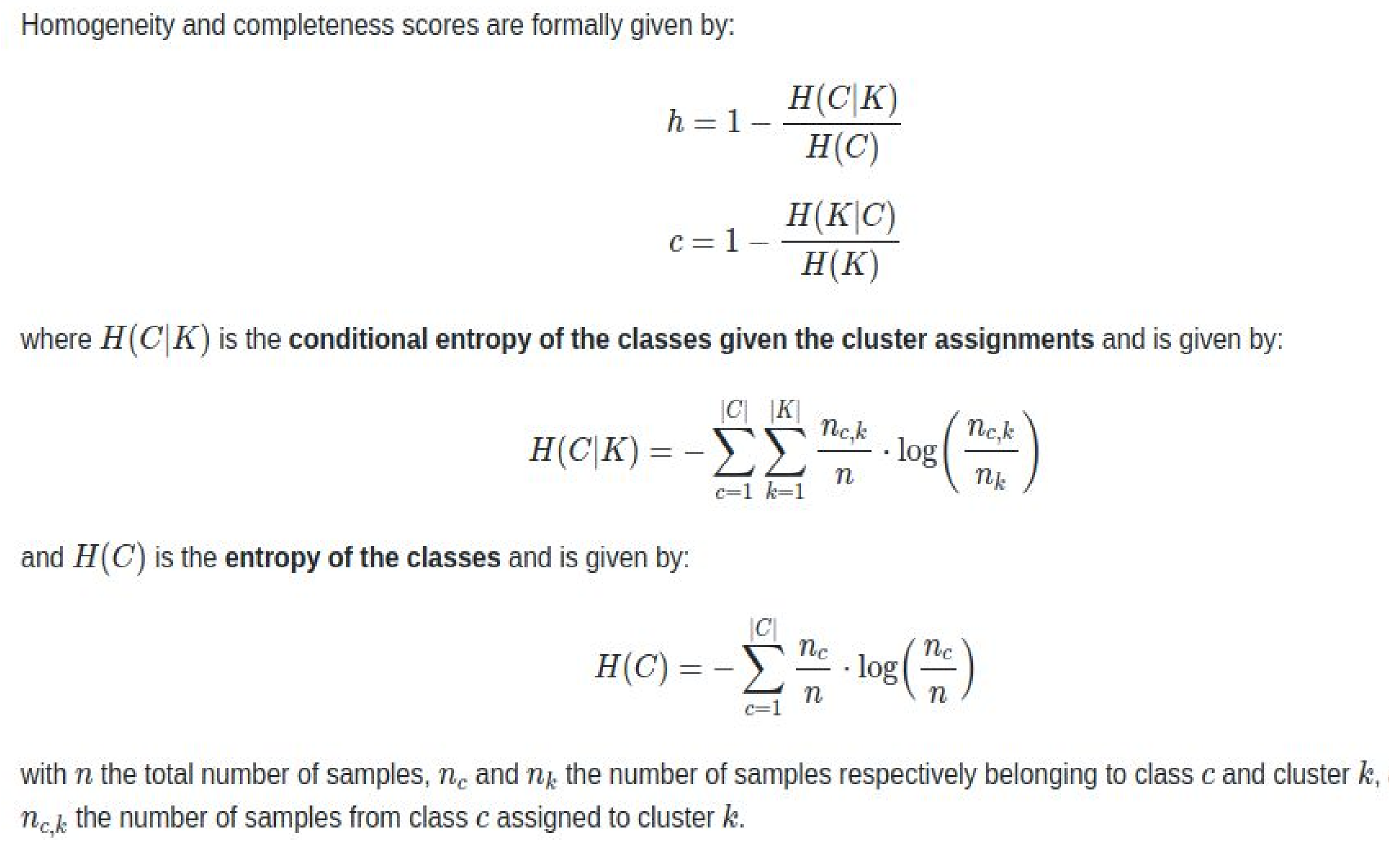
$$\text{AMI} = \frac{\text{MI} - E[\text{MI}]}{\text{mean}(H(U),H(V)) - E[\text{MI}]}$$

○ **Homogeneity, Completeness, V-measure:**

We have two desirable objectives for any cluster assignment:

- Homogeneity: each cluster contains only members of a single class.
- Completeness: all members of a given class are assigned to the same cluster.

These can be calculated quantitatively as given in below figure:



Homogeneity and completeness scores are formally given by:

$$h = 1 - \frac{H(C|K)}{H(C)}$$

$$c = 1 - \frac{H(K|C)}{H(K)}$$

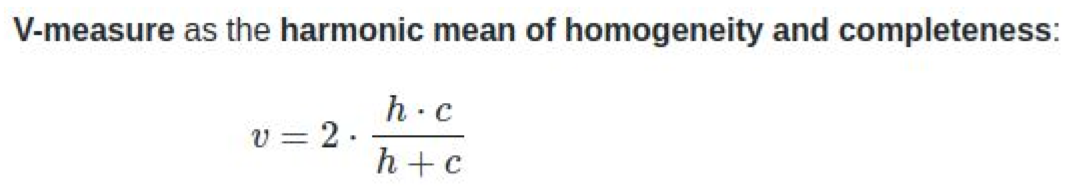
where $H(C|K)$ is the **conditional entropy of the classes given the cluster assignments** and is given by:

$$H(C|K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|} \frac{n_{c,k}}{n} \cdot \log\left(\frac{n_{c,k}}{n_k}\right)$$

and $H(C)$ is the **entropy of the classes** and is given by:

$$H(C) = -\sum_{c=1}^{|C|} \frac{n_c}{n} \cdot \log\left(\frac{n_c}{n}\right)$$

with n the total number of samples, n_c and n_k the number of samples respectively belonging to class c and cluster k,
$n_{c,k}$ the number of samples from class c assigned to cluster k.



V-measure as the **harmonic mean of homogeneity and completeness**:

$$v = 2 \cdot \frac{h \cdot c}{h + c}$$

One disadvantage of these metrics is that they are not normalized with respect to the distributions of the 2 clusters like ARI and AMI.

- **The Fowlkes-Mallows Index (FMI) :**

$$\mathrm{FMI} = \frac{\mathrm{TP}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})}}$$

  Where TP is the number of True Positive (i.e. the number of pair of points that belong to the same clusters in both the true labels and the predicted labels), FP is the number of False Positive (i.e. the number of pair of points that belong to the same clusters in the true labels and not in the predicted labels) and FN is the number of False Negative (i.e the number of pair of points that belongs in the same clusters in the predicted labels and not in the true labels).

- **WHY HIERARCHICAL CLUSTERING , WHY NOT K-MEANS?** DTW is not minimized by the mean. k-means may not converge and even if it converges it will not yield a very good result. The mean is a least-squares estimator on the coordinates. It is designed to minimize variance, not arbitrary distances. So, whenever DTW is used as a distance measure, hierarchical clustering is always preferred to k-means.

EXPERIMENTS-I

- We have used hierarchical clustering on a dataset of weekly sales transactions to demonstrate univariate time series clustering. The dataset has the data of sales of 811 objects for 52 weeks.The dataset is available in the same folder as this report, by the name Sales_Transactions_Dataset_Weekly.csv.
- Testing was done for 3 cases: single link clustering, average link clustering, and complete link clustering. The inconsistency coefficient threshold values were adjusted by trial and error such that optimum silhouette values were obtained. The depth of comparison used in all the 3 cases was 2.
- The experiments described in this section can be run interactively on the python notebook hclustering1.ipynb, which is also present in the same folder as this report.
- The hierarchical clustering was done using 2 distance metrics: dynamic time warping and the normal euclidean distance. The cophenetic correlation and the silhouette value were calculated for both the cases and the results were compared.

EXPERIMENTS-II

- We have used Hierarchical Clustering on a dataset of robot movements described by 6 time-series variables and the resulting consequences, to demonstrate multivariate time series clustering. The dataset has the data for 93 movements of 15 time units. The dataset also, has the ground truth values, which we will use to evaluate our results. The dataset is available in the same folder as this report, by the name lp5.data (The same dataset is used in classification as well).
- Testing was done for 3 cases: single link clustering, average link clustering, and complete link clustering. The inconsistency coefficient threshold values were adjusted by trial and error such that optimum silhouette values were obtained. The depth of comparison used in all the 3 cases was 2.
- The experiments described in this section can be run interactively on the python notebook hclustering2.ipynb, which is also present in the same folder as this report.
- The hierarchical clustering was done using 2 distance metrics: dynamic time warping and the normal euclidean distance. The cophenetic correlation, Adjusted Rand Index, Normalized Mutual Index, Adjusted Mutual Index, V-measure and Fowlkes-Mallows Index were calculated for both the cases and the results were compared.

RESULTS - UNIVARIATE CASE

- **Single Link Clustering:** Inconsistency Coefficient: 0.5

Criteria	DYNAMIC TIME WARPING	EUCLIDEAN
No. of Clusters	668	752
Cophenetic Correlation	0.3535	0.3794
Silhouette Value	0.047	0.010

- **Complete Link Clustering:** Inconsistency Coefficient: 1.1

Criteria	DYNAMIC TIME WARPING	EUCLIDEAN
No. of Clusters	293	282
Cophenetic Correlation	0.7372	0.7130
Silhouette Value	0.0630	-0.0038

- **Average Link Clustering:** Inconsistency Coefficient: 0.8

Criteria	DYNAMIC TIME WARPING	EUCLIDEAN
No. of Clusters	321	309

Cophenetic Correlation	0.7546	0.7567
Silhouette Value	0.1044	0.0007

- As can be seen, in all the 3 cases, though the cophenetic correlation was almost the same for both dynamic time warping and euclidean, the silhouette value was greater for dynamic time warping. Thus, it can be concluded that dynamic time warping is the better distance metric for the time series in this dataset.
- Also, it can be seen that Average Link Clustering works best for this dataset, followed by Complete Link Clustering and Single Link Clustering.

RESULTS - MULTIVARIATE CASE

- **Single Link Clustering:** Inconsistency Coefficient: 1.15

Criteria	**DYNAMIC TIME WARPING**	**EUCLIDEAN**
No. of Clusters	66	96
Cophenetic Correlation	0.9381	0.9644
Adjusted Rand Index	0.1266	0.0875
Adjusted Mutual Index	0.1726	0.1706

Normalized Mutual Index	0.4121	0.4656
Homogeneity, Completeness, V-measure	(0.6087, 0.3115, 0.4121)	(0.8309, 0.3234, 0.4656)
Fowlkes Mallows Score	0.2774	0.2065

- **Complete Link Clustering:** Inconsistency Coefficient: 1.15

Criteria	**DYNAMIC TIME WARPING**	**EUCLIDEAN**
No. of Clusters	29	42
Cophenetic Correlation	0.8446	0.9155
Adjusted Rand Index	0.1899	0.1636
Adjusted Mutual Index	0.3449	0.2972
Normalized Mutual Index	0.4547	0.4472
Homogeneity, Completeness, V-measure	(0.6188, 0.3594, 0.4547)	(0.6661, 0.3366, 0.4472)
Fowlkes Mallows Score	0.3287	0.2955

- **Average Link Clustering:** Inconsistency Coefficient:1.15

Criteria	**DYNAMIC TIME WARPING**	**EUCLIDEAN**
No. of Clusters	27	44
Cophenetic Correlation	0.9685	0.9890

Adjusted Rand Index	0.1632	0.1854
Adjusted Mutual Index	0.3042	0.2945
Normalized Mutual Index	0.4182	0.4496
Homogeneity, Completeness, V-measure	(0.5320, 0.3446, 0.4182)	(0.6517, 0.3431, 0.4496)
Fowlkes Mallows Score	0.3217	0.3192

- As can be seen, though the cophenetic correlation was the greatest for average link clustering, all other metrics favour complete link clustering. Thus, it can be concluded that a complete link is a much better choice for this dataset. This is probably because the clusters in this dataset are expected to be a lot more compact.
- Also, it can be seen that the Euclidean distance metric and DTW work more or less the same. There is no conclusive evidence to suggest that one of the distance metrics is better than the other. This might be because, in this dataset all types of movements occur at the same speed.

CONCLUSION

- This report has given a theoretical introduction to Dynamic Time Warping, an appropriate distance metric for time series. It has also given a theoretical introduction to hierarchical clustering and the various evaluation metrics for clustering.
- The report has demonstrated a practical application of time series clustering on a weekly sales transaction dataset. The results demonstrate the power of dynamic time warping + hierarchical clustering.
- The report has also demonstrated another practical application of time series clustering on a dataset of robotic movements. The results demonstrate that dynamic time warping may not be the better choice for all kinds of datasets. A factor that should be kept in mind is that Euclidean distance calculation (O(N) complexity) is a lot faster than DTW ($O(N^2)$ complexity).
- The results have also demonstrated how different linkage techniques are the best for different types of data. While the first dataset showed average linking is better, complete linking was more suitable for the second dataset.