

## Hierarchical Clustering

Prof. Marco Alvarez, Computer Science  
University of Rhode Island

## Hierarchical clustering

- ▶ Unsupervised learning technique that groups observations into nested clusters
- ▶ Key features:
  - does not require a predefined number of clusters
  - creates a hierarchy of clusters, often visualized as a dendrogram
    - can be agglomerative (bottom-up), or divisive (top-down)
  - widely used in various domains (gene expression, phylogenetic trees, community detection, etc.)

## Agglomerative vs divisive

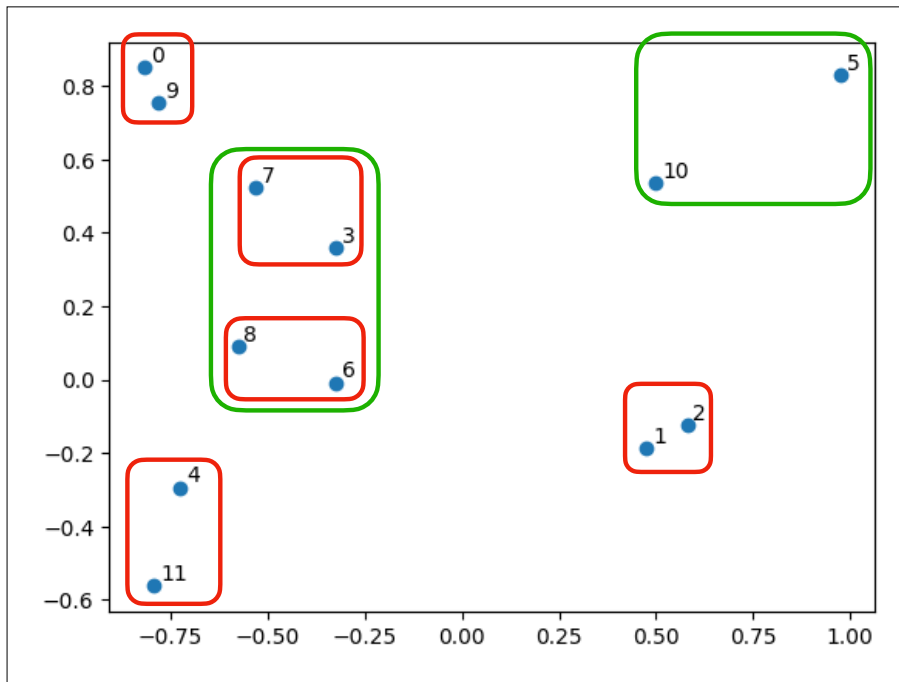
- ▶ Agglomerative approach (**bottom-up**)
  - start with observations as singleton clusters and progressively merge them until a final single cluster is created
  - most common form
- ▶ Divisive approach (**top-down**)
  - start with all observations on a single cluster and progressively split the clusters until all observations are singleton clusters

## HAC algorithm

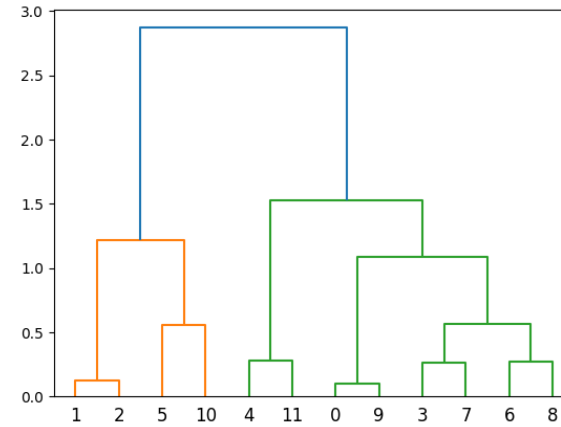
- ▶ Initially each observation is considered a cluster
- ▶ Repeat
  - **merge**: find the closest pair of clusters and merge it into a single cluster
  - if all observations are in a single cluster stop

Distance Metrics:  
- Euclidean distance  
- Manhattan distance  
- Minkowski distance  
- Hamming distance  
- Cosine similarity  
- Correlation coefficient

Linkage Criteria:  
- Single-linkage  
- Complete-linkage  
- Average-linkage  
- Centroid-linkage  
- Ward's method



## Dendrogram

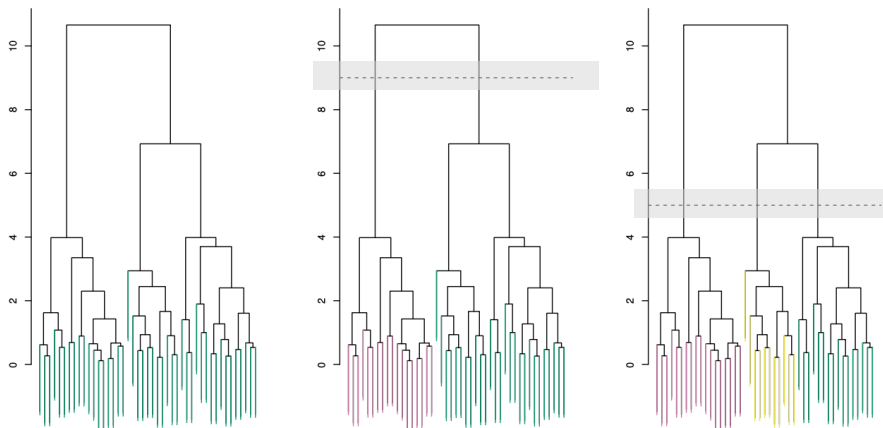


A dendrogram represents the hierarchy of clusters

Y-axis represents the distance at which clusters are merged

Horizontal cuts through the dendrogram represent cluster groupings at various thresholds

## HAC with complete-linkage



An Introduction to Statistical Learning, 2nd Ed

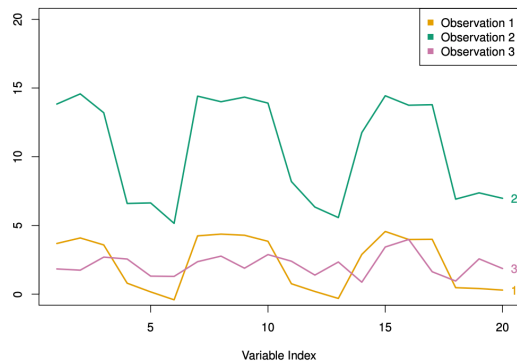
### Pairwise metrics

See the [Pairwise metrics, Affinities and Kernels](#) section of the user guide for further details.

<code>metrics.pairwise.additive_chi2_kernel(X[, Y])</code>	Compute the additive chi-squared kernel between observations in X and Y.
<code>metrics.pairwise.chi2_kernel(X[, Y, gamma])</code>	Compute the exponential chi-squared kernel between X and Y.
<code>metrics.pairwise.cosine_similarity(X[, Y, ...])</code>	Compute cosine similarity between samples in X and Y.
<code>metrics.pairwise.cosine_distances(X[, Y])</code>	Compute cosine distance between samples in X and Y.
<code>metrics.pairwise.distance_metrics()</code>	Valid metrics for pairwise_distances.
<code>metrics.pairwise.euclidean_distances(X[, Y, ...])</code>	Compute the distance matrix between each pair from a vector array X and Y.
<code>metrics.pairwise.haversine_distances(X[, Y])</code>	Compute the Haversine distance between samples in X and Y.
<code>metrics.pairwise.kernel_metrics()</code>	Valid metrics for pairwise_kernels.
<code>metrics.pairwise.laplacian_kernel(X[, Y, gamma])</code>	Compute the laplacian kernel between X and Y.
<code>metrics.pairwise.linear_kernel(X[, Y, ...])</code>	Compute the linear kernel between X and Y.
<code>metrics.pairwise.manhattan_distances(X[, Y, ...])</code>	Compute the L1 distances between the vectors in X and Y.
<code>metrics.pairwise.nan_euclidean_distances(X)</code>	Calculate the euclidean distances in the presence of missing values.
<code>metrics.pairwise.pairwise_kernels(X[, Y, ...])</code>	Compute the kernel between arrays X and optional array Y.
<code>metrics.pairwise.polynomial_kernel(X[, Y, ...])</code>	Compute the polynomial kernel between X and Y.
<code>metrics.pairwise.rbf_kernel(X[, Y, gamma])</code>	Compute the rbf (gaussian) kernel between X and Y.
<code>metrics.pairwise.sigmoid_kernel(X[, Y, ...])</code>	Compute the sigmoid kernel between X and Y.
<code>metrics.pairwise.paired_euclidean_distances(X, Y)</code>	Compute the paired euclidean distances between X and Y.
<code>metrics.pairwise.paired_manhattan_distances(X, Y)</code>	Compute the paired L1 distances between X and Y.
<code>metrics.pairwise.paired_cosine_distances(X, Y)</code>	Compute the paired cosine distances between X and Y.
<code>metrics.pairwise.paired_distances(X, Y, *[, ...])</code>	Compute the paired distances between X and Y.
<code>metrics.pairwise_distances(X[, Y, metric, ...])</code>	Compute the distance matrix from a vector array X and optional Y.
<code>metrics.pairwise_distances_argmin(X, Y, *[, ...])</code>	Compute minimum distances between one point and a set of points.
<code>metrics.pairwise_distances_argmin_min(X, Y, *)</code>	Compute minimum distances between one point and a set of points.
<code>metrics.pairwise_distances_chunked(X[, Y, ...])</code>	Generate a distance matrix chunk by chunk with optional reduction.

## What kind of distance to consider?

- We have been using Euclidean distances, however, the choice is **very important**

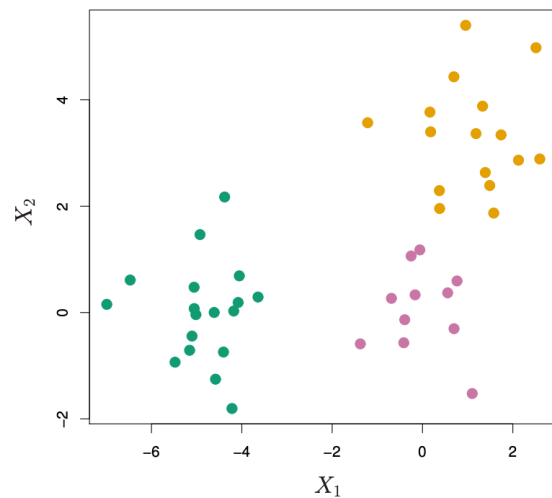


An Introduction to Statistical Learning, 2nd Ed

## Distances between clusters

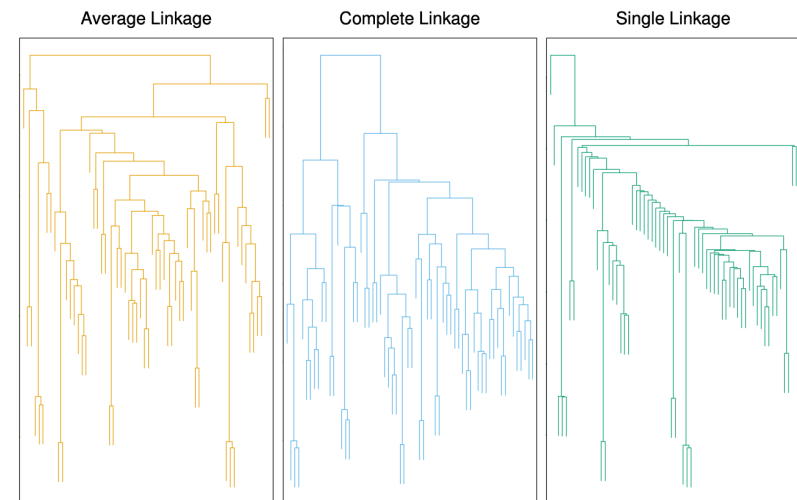
Single-linkage:	$D(A, B) = \arg \min_{\mathbf{x}_1 \in A, \mathbf{x}_2 \in B} d(\mathbf{x}_1, \mathbf{x}_2)$
Complete-linkage:	$D(A, B) = \arg \max_{\mathbf{x}_1 \in A, \mathbf{x}_2 \in B} d(\mathbf{x}_1, \mathbf{x}_2)$
Average-linkage:	$D(A, B) = \frac{1}{ A   B } \sum_{\mathbf{x}_1 \in A, \mathbf{x}_2 \in B} d(\mathbf{x}_1, \mathbf{x}_2)$
Centroid-linkage:	$D(A, B) = \ \mu_A - \mu_B\ _2^2$
Ward's method:	Minimizes variance within clusters

## Consider this data



An Introduction to Statistical Learning, 2nd Ed

## Example



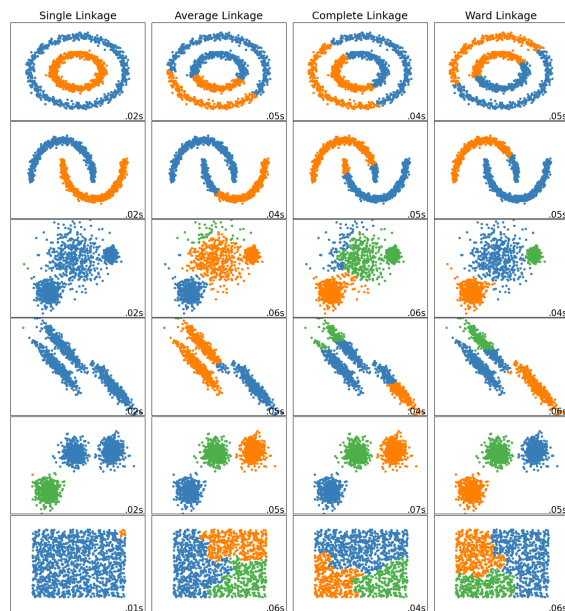
An Introduction to Statistical Learning, 2nd Ed

# Limitations and considerations

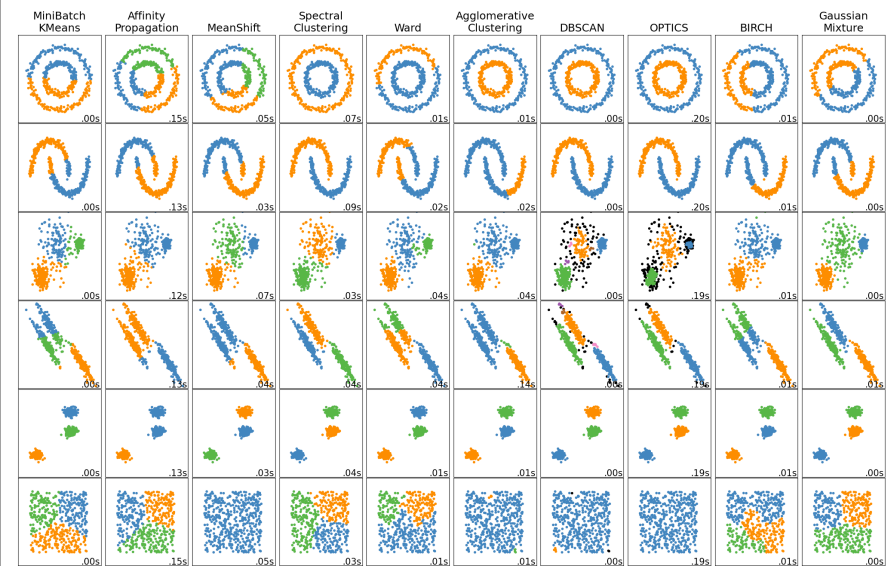
- ▶ **Computational complexity**
  - can be computationally expensive for large datasets
- ▶ **Sensitivity to noise and outliers**
  - can be sensitive to noise and outliers in the data
- ▶ **Choice of distance metric and linkage criterion**
  - distance metric and linkage criterion can significantly affect the results
- ▶ **Optimal number of clusters**
  - no definitive method for determining the optimal number of clusters

## HAC Notebook

<https://colab.research.google.com/drive/13ENliESw1VT7IWNXeB0Q-7tyMuHPppuM>



From scikit-learn



From scikit-learn