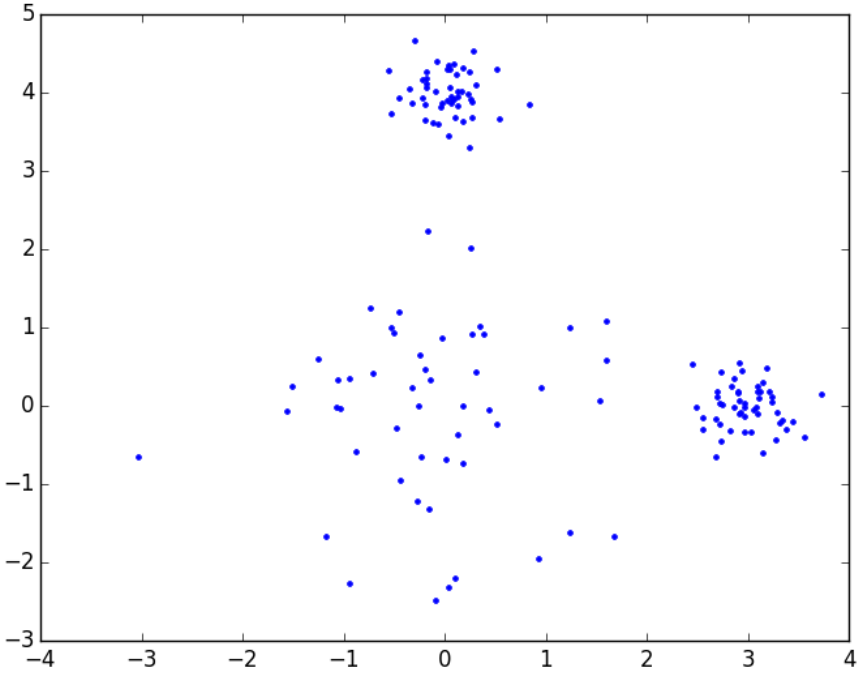
Clustering

- Divide an (unlabeled) dataset into a fixed number of clusters (or groups) such that the instances within each cluster are similar to one another and different from the instances in other clusters
- Purpose: Discover the underlying structure in some dataset, or to find an optimal reduced representation
- k-means
- Gaussian Mixture Models with E-M algorithm
- Hierarchical Clustering

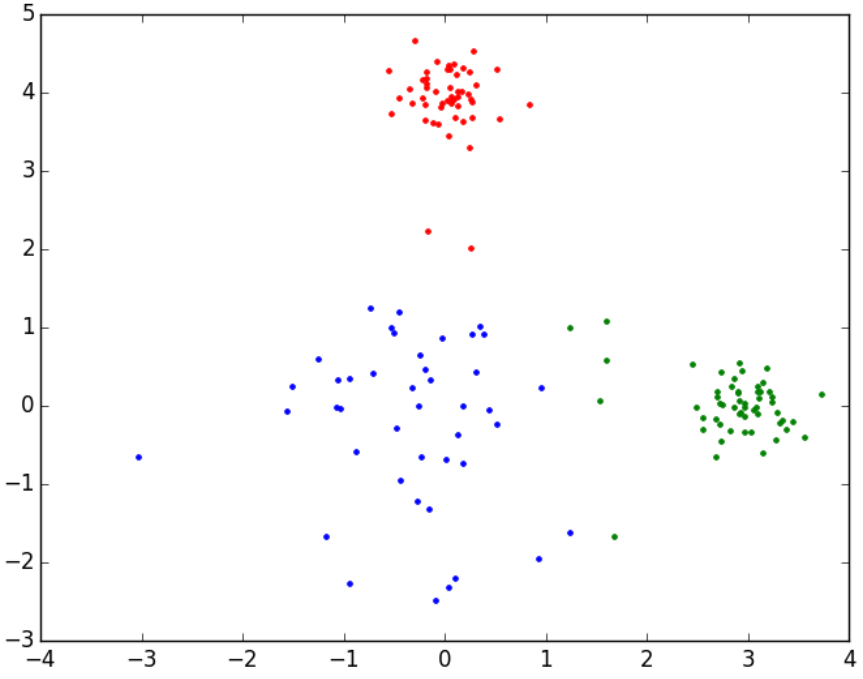
Example

- Easily visualized for two-dimensional data



Distinction between clustering and classification

- In classification problems we know already some set of classes to which each of our data instances belongs
- In clustering, conversely we have a completely flat dataset it is up to us to discover the structure



Clustering Function

- If we write our data instances $a_1, \ldots, a_n$ then we need some metric to quantify how well our data is clustered
- We can use the euclidean distance for this task: $d(a_i, a_j) = ||a_i - a_j||_2^2$
- Assume that we know in advance the number of clusters we want to fit to our data K
- A clustering is a function that assigns a label to each data instance: $f : \text{data} \rightarrow \{1, \ldots, K\}$

Objective for Clustering

- The clustering problem can also be formulated as an optimization
- Here f is the classification and n_k is the number of instances assigned to class k

$$J = \frac{1}{2} \sum_{k=1}^{K} \frac{1}{n_k} \sum_{f(i)=k, f(j)=k} d(a_i, a_j) \tag{1}$$

Clustering for Euclidean Distance

- It can be shown that the objective reduces as follows for the case of the Euclidean Distance, where A_k is the average data instance in group k

$$J = \sum_{k=1}^{K} \sum_{f(i)=k} ||a_i - A_k||_2^2 \tag{2}$$

- This suggests a more general approach where we iteratively minimize first over the groupings then the cluster centers

K-Means algorithm

$$J = \sum_{k=1}^{K} \sum_{f(i)=k} ||a_i - F_k||_2^2 \tag{3}$$

- Fix the number of clusters and pick an initial (random) guess of the cluster centers F_k
- Follow two steps until convergence:
- Assign each point to the nearest cluster center
- Recalculate the cluster centers as the average of all of the points in the cluster

Aspects of the K-Means Algorithm

- The cluster centers become a representative point for *all* of the points in a given cluster
- The minimum distance criterion partitions the space into a set of convex polyhedra
- Although the algorithm is guaranteed to converge the solution can be highly dependent upon the initial conditions, often desirable to run the algorithm multiples times and choose the result with the best (smallest) within cluster variance
- Another name for k-means is *vector quantization*
- Combinatorial dependence on the size of the dataset: infeasible to check all possible solutions

K-Means in scipy

- Scipy provides two options for implementing the k-means algorithm: `scipy.cluster.vq.kmeans` and `sklearn.cluster.KMeans`

Clustering Recap

- Last lecture: introduced clustering and a specific clustering algorithm called k-means
- *Clustering*: Process of dividing a dataset into clusters or groups such that all the instances in each cluster are similar to one another
- Properties of k-means: 1) number of clusters specified in advance (k) 2) Assignment dependent upon random initialization algorithm
- Algorithm: a) Cluster centers compute assignment b) Assignment compute new cluster centers

k-means versus Hierarchical Clustering

- Today we will introduce a new type of clustering algorithm called hierarchical clustering
- Unlike k-means the assignment will not depend upon initialization of the algorithm
- Also the number of clusters is not specified in advance
- Input data: measurement of distances between all pairs of data points, $(x_1, x_2, \ldots, x_n)$, use the matrix of distances, $d(x_i, x_j)$

Top-down and Bottom-up Clustering

- There are two types of hierarchical clustering algorithms: *agglomerative* and *divisive*
- Agglomerative: Begin with all data instances in separate clusters and repeatedly merge clusters until we have a single group (merge two 'nearest' clusters)
- Divisive: Opposite of agglomerative, begin with single large group and split until all clusters are singleton
- We will look agglomerative algorithms today
- From the definition we need a way to measure distances between two clusters (not only between data instances): this is known as the *linkage*

Linkages

- Given two groups of data instances $A = \{x_1, x_2, x_3\}$ and $B = \{x_4, x_5, x_6\}$
- Compare A and B using the distances their elements
- Distance here can be any metric we choose including the euclidean distance
- Goal: some function d that takes clusters (groups) as arguments and returns a number representing the dissimilarity of those clusters, $d(A, B)$

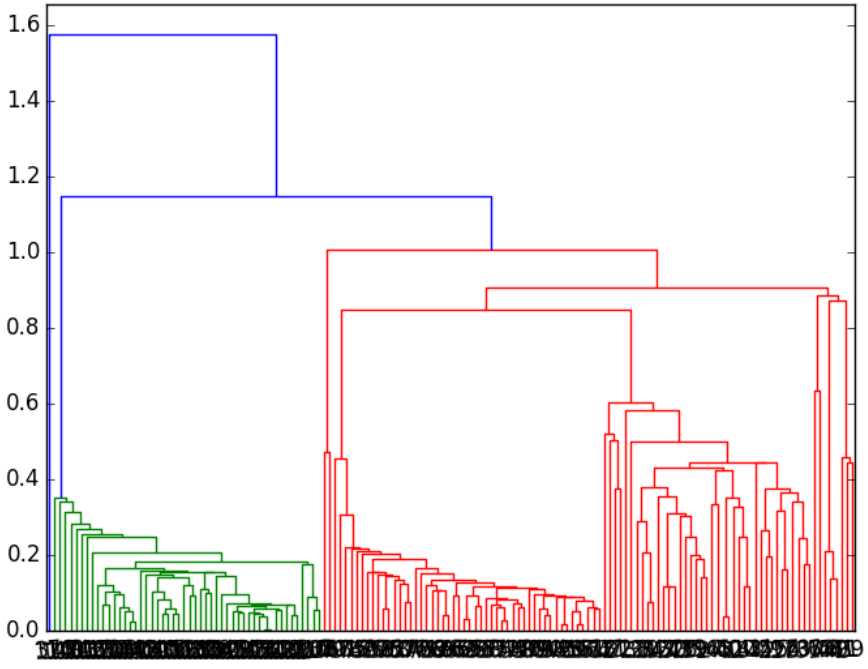
Single Linkage

- In single linkage we return the *smallest* distance between two groups
- Two groups are 'close' together if any of their points are near to each other

$$d(A, B) = \min_{i \in A, j \in B} d(x_i, x_j) \tag{4}$$

Visualizing Hierarchical Clustering with Dendrograms

- A *dendrogram* is a natural visual representation of the hierarchical clustering process (single linkage example)
- Cutting a dendrogram a fixed height gives a clustering of the data



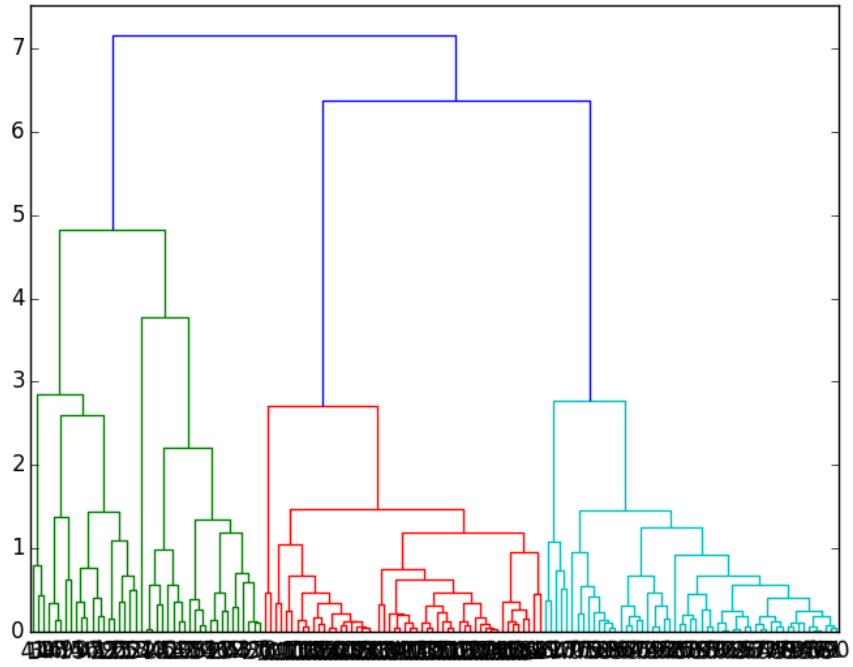
Dendrograms

- Tree where each node represents a cluster
- Leaf nodes represent singletons
- Root node contains all the data instances
- The height at which we draw a node is proportional to the dissimilarity of the two nodes
- Also the order of data instances is usually permuted so that the tree is more visually appealing (closer nodes placed near each other)

Complete Linkage

- In complete linkage we return the *maximum* distance between two groups
- Distance of the farthest pair

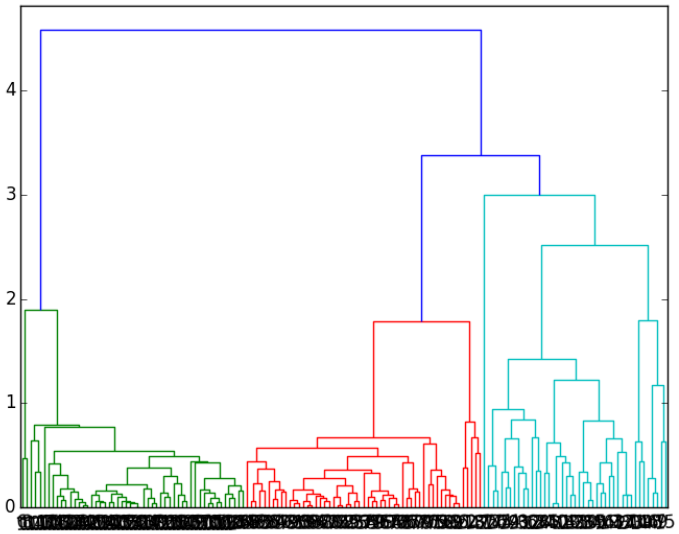
$$d(A, B) = \max_{i \in A, j \in B} d(x_i, x_j) \tag{5}$$



Average Linkage

- In complete linkage we return the *average* distance between all pairs of points in the two groups
- Distance of the farthest pair

$$d(A, B) = \frac{1}{|A||B|} \sum_{i \in A, j \in B} d(x_i, x_j) \tag{6}$$



Properties of hierarchical clustering

- As we run the algorithm (i.e. as we merge more nodes together) the distances between the clusters we are merging is always increasing
- For examples it is convenient to use the euclidean distance $||x - y||_2$ but in general we have many different distance metrics to choose from

Properties of hierarchical clustering

- As we run the algorithm (i.e. as we merge more nodes together) the distances between the clusters we are merging is always increasing
- For examples it is convenient to use the euclidean distance $||x - y||_2$ but in general we have many different distance metrics to choose from

Defining different metrics

- Different ways to compute distance between data instances

$$d(x, y) = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2}, \text{ Euclidean} \tag{7}$$

$$d(x, y) = \sum_{i=1}^{N}|x_i - y_i|, \text{ Taxicab} \tag{8}$$

$$d(x, y) = \max_i |x_i - y_i|, \text{ Chebyshev} \tag{9}$$

$$d(x, y) = \left(\sum_{i=1}^{N}(x_i - y_i)^p\right)^{1/p}, \text{ Minkowski} \tag{10}$$

Problems with Single and Complete Linkages

- Single linkage can result in clusters that are too spread out while complete linkage suffers from chaining
- Average linkage is a compromise between single and complete although it also has its own difficulties for example hard to interpret dendrogram

Clustering in Python

- Hierarchical clustering and dendrogram visualization is provided in python via the module `scipy.cluster.hierarchy`

```
from scipy.spatial.distance import pdist, squareform
from scipy.cluster.hierarchy import fcluster, linkage, dendrogram
```

- `pdist`: will take our data as argument and return an object representing the distances between all of the different observations
- `linkage`: will take the object returned from `pdist` and perform the agglomerative clustering we have been discussing, specify whether we want single, complete, or average linkages

Clustering in Python (cont.)

- `fcluster()`: takes the object returned from `linkage` and returns an array representing the assignment
- `fcluster(Z,t=5.,criterion='distance')`: we specify arguments to control how clusters are formed (in this case we are using distance) and the threshold (where we cut the dendrogram)
- `fcluster()`: criterion to 'maxclust' then the threshold will determine the number of clusters, threshold found automatically
- `dendrogram`: take the linkage matrix and plot the dendrogram