BUSN 20800: Big Data

Lecture 5: Clustering

Dacheng Xiu

University of Chicago Booth School of Business

Unsupervised Methods

- Supervised versus unsupervised data analysis
- Model-based clustering: **mixture models**
- **K-means** algorithm
- Spectral clustering
- Hierarchical clustering

Supervision

You've seen lots of models for $[y|x]$. Today is about models for x.

The goal in everything we do has been *Dimension Reduction*.

DR: move from high dimensional **x** to low-D summaries.

Dimension reduction can be supervised or unsupervised.

Supervised: Regression and classification

HD **x** is projected through β into 1D $\hat{y}$

Outside info (y) supervises how you simplify **x**.

Unsupervised: Mixture and Factor Models

x is modeled as built from a small number of components.

You're finding the simplest representation of **x** alone.

We always want the same things: low deviance, without overfit.

What is clustering? And why?

Clustering: dividing up data into groups (clusters), so that points inside each group are more "similar" to each other than to points outside the group.
Why cluster? Two main uses

- Summary: (DR) deriving a reduced representation of data
- Discovery: looking for new insights into the data structure. Clustering can also help with predictions. However,

clustering should not be confused with classification!

- **In classification**, we have data for which the groups are known and we try to learn what differentiates them to assign future labels
- **In clustering**, we have data for which the group labels are unknown and try to learn the groups themselves as well as what differentiates them.

Clustering: unsupervised dimension reduction

Group observations into similar 'clusters', and understand the rules behind this clustering.

- Demographic Clusters: Soccer moms, NASCAR dads.
- Consumption Clusters: Jazz listeners, classic rock fans.
  Collaborative Filtering
  Group individuals into clusters, and model average behavior for each.
- Industry Clusters: Competitor groups, supply chains.

Clustering is largely an **exploratory** technique.

- model-based methods (mixture models)
- nonparametric methods (spectral clustering)
- "heuristic" methods (hierarchical clustering)
  Sometimes, it is useful to have clusters organized in a hierarchy.

The K-means Mixture Model

The fundamental model of clustering is a mixture:

observations are random draws from K populations, each with different average characteristics.

Suppose you have K possible means for each observed $\mathbf{x}_i$:

$$\mathbb{E}[\mathbf{x}_i \mid k_i] = \boldsymbol{\mu}_{k_i}, \quad \text{where} \quad k_i \in \{1 \dots K\}$$

e.g., if $k_i = 1$, $\mathbf{x}_i$ is from cluster 1: $\mathbb{E}[x_{i1}] = \mu_{11}$ ... $\mathbb{E}[x_{ip}] = \mu_{1p}$.

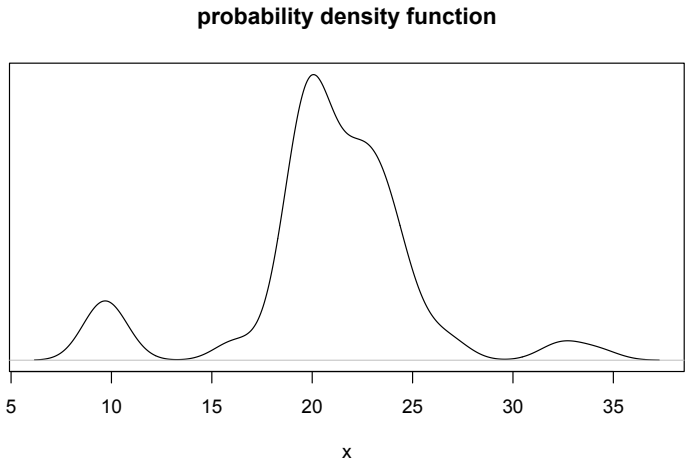
Each mean μ_j has an associated probability or "weight" in the mixture.

For new $\mathbf{x}$ with unknown k,

$$\mathbb{E}[\mathbf{x}] = \mathrm{p}(k = 1)\boldsymbol{\mu}_1 + \ldots + \mathrm{p}(k = K)\boldsymbol{\mu}_K$$

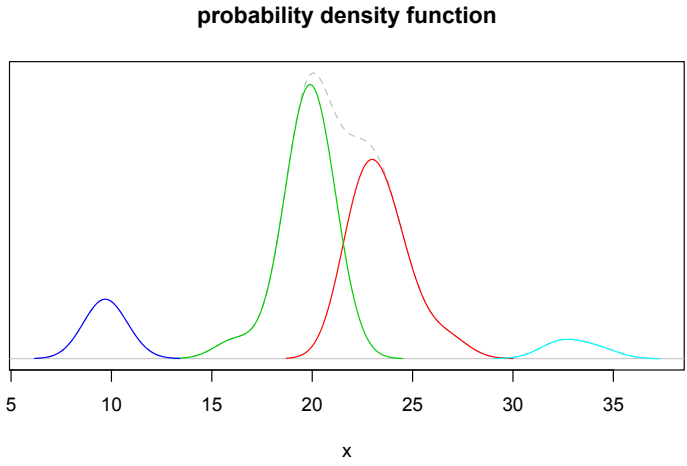
DR: Given $\boldsymbol{\mu}_k$'s, you discuss data in terms of K different types, rather than trying to imagine all possible values for each $\mathbf{x}$.

Mixture Model: without knowing membership 'k'



The *marginal* density has multiple modes; one for each $\boldsymbol{\mu}_k$.

Mixture Model: breaking into K components



Here, we have $K = 4$ different cluster centers. Should it be 5?

K -means: Chicken-egg problem

For given K, the goal is to find clusters so that the within-cluster variability is small.
We do not know cluster memberships k_i and we do not know centroids (K-means) μ_k

Chicken-and-Egg problem

But!

1. **If we knew** k_i : we can easily estimate μ_k
2. **If we knew** μ_k : we can easily estimate k_i

Solution: iterate between 1. and 2.
This strategy relates to the EM algorithm, one of the workhorses of statistical computing.

K-means

K-means algorithm clusters data by fitting a mixture model.

Suppose data $\mathbf{x}_1 \ldots \mathbf{x}_n$ comes from K clusters.

Chicken: If you know membership k_i for each x_i, then estimate

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:k_i=k} \mathbf{x}_i$$

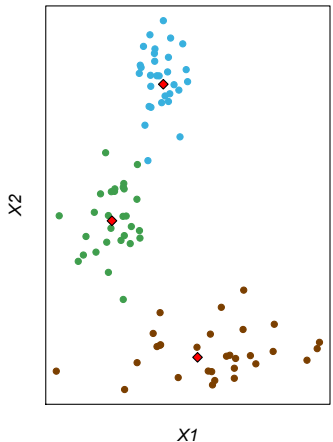
where $\{i : k_i = k\}$ are the n_k observations in group k.

Egg: If you know means μ_k, find $\boldsymbol{k} = k_1 \ldots k_n$ to minimize the sum-of-squares

$$\sum_{k=1} \sum_{i:k_i=k} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^2$$

Mixture deviance: total sums of squares **within** each cluster.

Give K-means the x_i's, and it gives you back the k_i's



In words: K-means

- Label each point based on the closest centroid (mean)
- Replace each centroid by the average of the points in the cluster

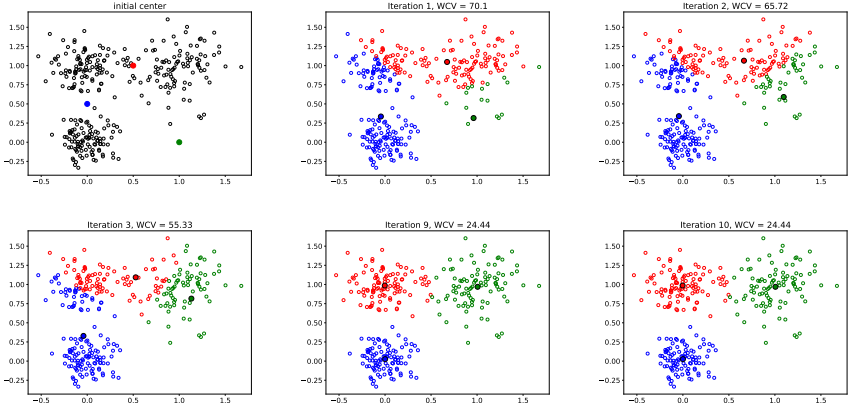
The algorithm starts at random $\boldsymbol{\mu_k}$, and changes k_i's until the sum of squares stops improving.

Solution depends on start location. Try multiple, take the best answer.

Choosing K: For most applications, everything is descriptive. So try a few and use clusters that make sense to you.

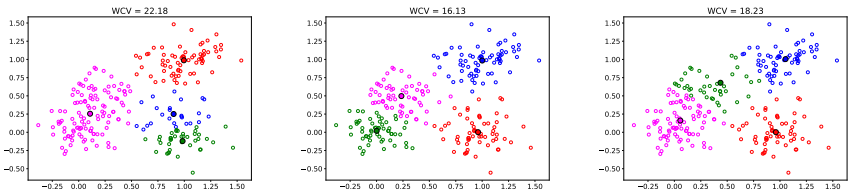
K-means example

Here $X_i = (X_{i1}, X_{i2})$, $n = 300$, and $K = 3$



K-means example, multiple runs

Here $X_i = (X_{i1}, X_{i2})'$, $n = 250$, and $K = 4$, the points are not as well-separated



These are results of running the K-means algorithm with different initial centers (chosen randomly over the range of the X_i 's). We choose the second collection of centers because it yields the smallest within-cluster variation (mixture deviance)

Clusters X (numeric!) into n_clusters groups.

```
from sklearn.cluster import KMeans
kmeans=KMeans(n_clusters=3,init='random',n_init=1,max_iter=10)
kmeans.fit(X)
```

Output each cluster's center coordinates:

```
np.around(kmeans.cluster_centers_,2)
array([[ 0.  ,  0.03],
       [-0.  ,  0.99],
       [ 1.01,  0.97]])
```

Information about cluster labels:

```
from collections import Counter
Counter(kmeans.labels_)
Counter(0: 101, 1: 105, 2: 94)
```

Within cluster sum of squares:

```
round(kmeans.inertia_,2)
24.44
```

Scaling for K-means

The algorithm minimizes total [squared] distance from center, *summed across all dimensions of* **x**.

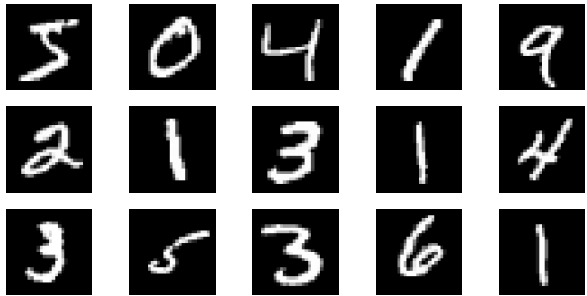
Scale matters: if you replace x_j with $2x_j$, that dimension counts twice as much in determining distance from center
(and will have more influence on cluster membership).

Standard solution is to standardize:

$$\text{cluster on scaled } \tilde{x}_{ij} = \frac{x_{ij} - \bar{x}_j}{\text{sd}(x_j)}$$

The MINST Database of Handwritten Digits

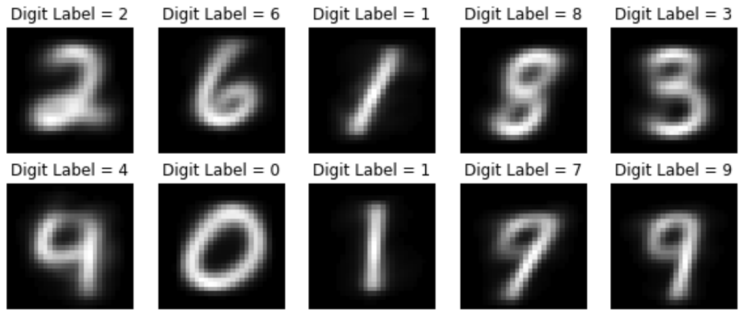
- The MNIST database of handwritten digits, available from Yann LeCun's Website, has a training set of 60,000 examples, and a test set of 10,000 examples.
- The digits have been size-normalized and centered in a fixed-size image.
- The training data: $60,000 \times 784$ matrix; the testing data: $10,000 \times 784$ matrix.



10-means clustering on MNIST

The most natural choice of K is 10. Why?

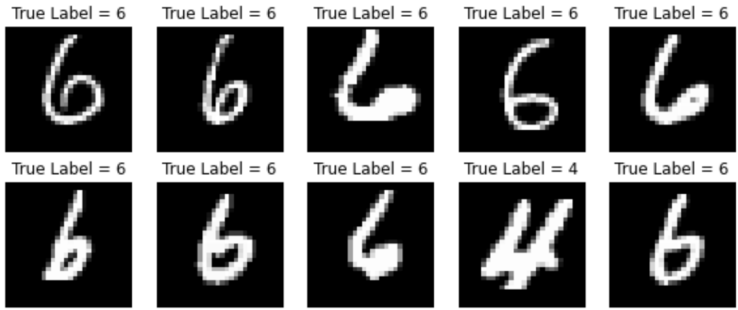
Display the center of each cluster:



If using the majority label in the clusters as the predicted label, the model has the accuracy of : 57.84 % on training data and 59.45 % on testing data.

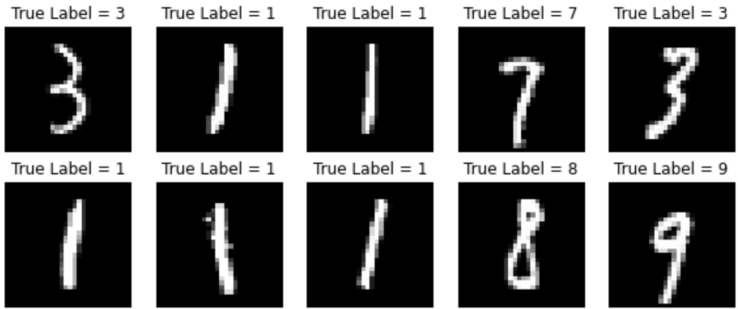
10-means clustering on MNIST

10 random images of the cluster whose majority label is 6



10-means clustering on MNIST

10 random images of the cluster whose majority label is 1



Choosing K

1st order advice: most often clustering is an exploration exercise, so choose the K that makes the most sense to you.

But, we can apply data-based model building here:

1. Enumerate models for $k_1 < k_2 < \ldots < k_K$.
2. Use a selection tool to choose the best model for new $\mathbf{x}$.

Step one is easy. Step two is a tougher.

For example, for CV you'd want to have high OOS $\mathrm{p}_{k_i}(\mathbf{x}_i)$.

But you don't know k_i! This is a *latent* variable.

There's no ground truth like y to compare against.

AIC and BIC for K-means

We *can* use IC to select K.

- Deviance is (as always) D=-2logLHD, which for K-means is the total sum of squares (slide 10).
- df is the number of μ_{kj}: $K \times p$. (where p is dimension of $\mathbf{x}$)

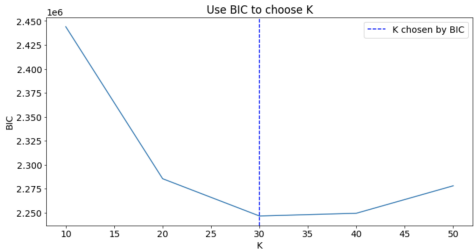
Then our usual AIC and BIC formulas apply.

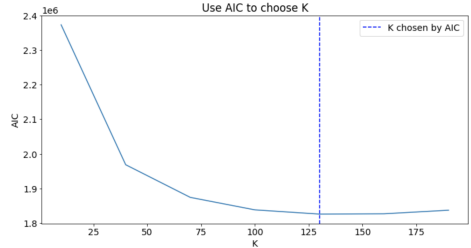
Beware: the assumptions behind both AIC and BIC calculations are only roughly true for K-means.

These tools are lower quality here than in regression.

You're often better off just using descriptive intuition.

BIC and AIC for MNIST K-means





BIC likes $K = 30$, AIC likes 130.
Both are way more complicated than is useful.

With $K = 30$, the model has the accuracy of : 75.085% on training data, and 76.29% on testing data.

Re-visiting the K-means model

In minimizing sums-of-squares, K-means targets the model

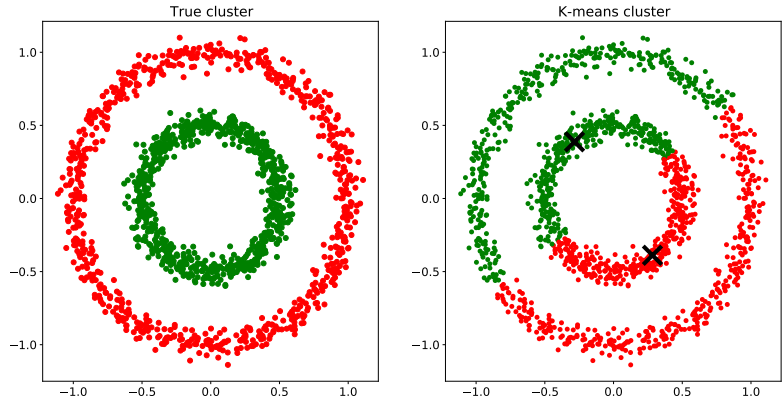
$$\mathrm{p}_k(\mathbf{x}) = \prod_j \mathrm{N}(x_j \mid \mu_{kj}, \sigma^2)$$

$\Rightarrow$ independence across dimensions (no multicollinearity) and uniform variance (same σ for all j, which is why scale matters).

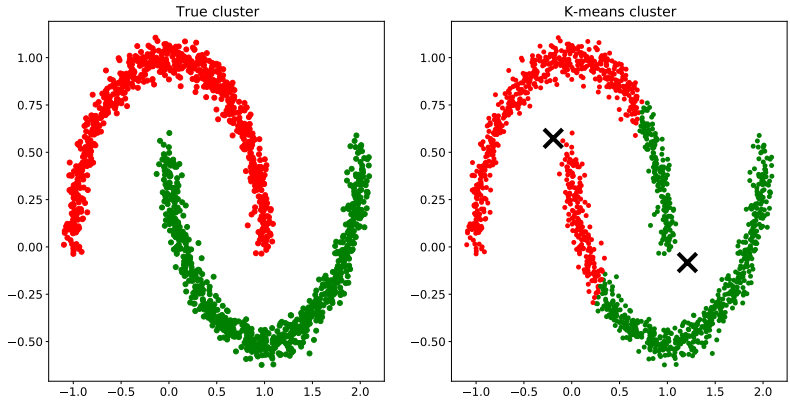
Despite being a silly model, this tends to do a decent job of clustering when **x** consists of *continuous* variables.

It does a worse job for **x** made up of dummies or counts.

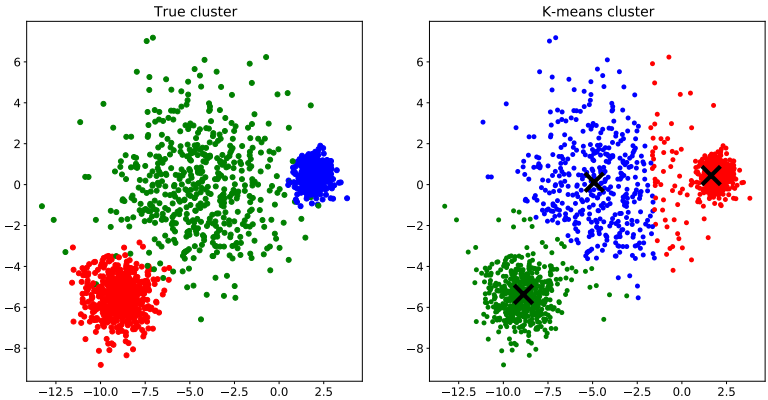
When K-means fail: non-Gaussian



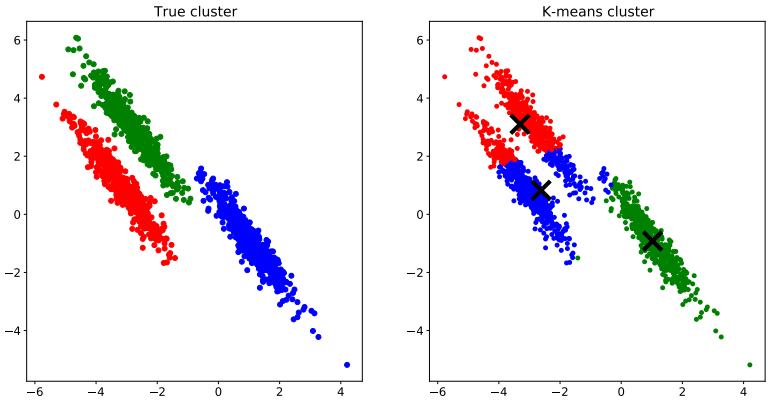
When K-means fail: non-Gaussian



When K-means fail: unequal variances



When K-means fail: anisotropic data



Solution: From K-means to spectral clustering

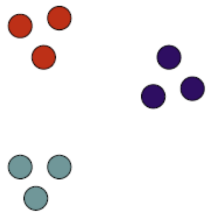
Spectral clustering is a generalization of the standard clustering methods, such as K-means, and it can tackle with the above situations.

Goal: Given data points X_1, ..., X_n, we have a measure of how similar each pair of point (X_i, X_j) is, say $W_{i,j}$, and we partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

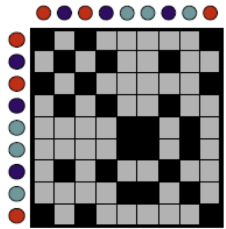
It has a close connection with graph clustering

Similarity Graph: G(V,E,W)

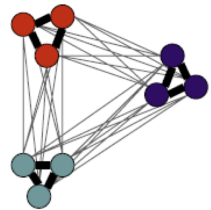
V – Vertices (Data points)
E – Edge if similarity > 0
W - Edge weights (similarities)



Data



Similarities



Similarity graph

Weighted adjacency matrix

K-nearest neighbors graph: we connect v_i and v_j with an edge if v_i is among the K-nearest neighbors of v_j or if v_j is among the K-nearest neighbors of v_i. This ensures that W is symmetric.

fully connected graph: we simply connect all points with positive similarity with each other, and we weight all edges by $W_{i,j} = exp(-d^2(i,j)/c)$, where $d(i,j)$ is the dissimilarity (like distance).

Spectral clustering algorithm

We can then use the W matrix to obtain a new matrix $L = G - W$, where G is diagonal with $G_{i,i} = \sum_j W_{i,j}$.

This matrix L has several beautiful properties, which we will take advantage of for designing the clustering algorithm below:

- Compute the first K eigenvectors associated with the smallest eigenvalues of L:

$$V_{n \times K} = [v_1, v_2, \ldots, v_K].$$

- For the ith data point, we "represent" it by K coordinates:

$$(v_{i,1}, v_{i.2}, \ldots, v_{i,K}).$$

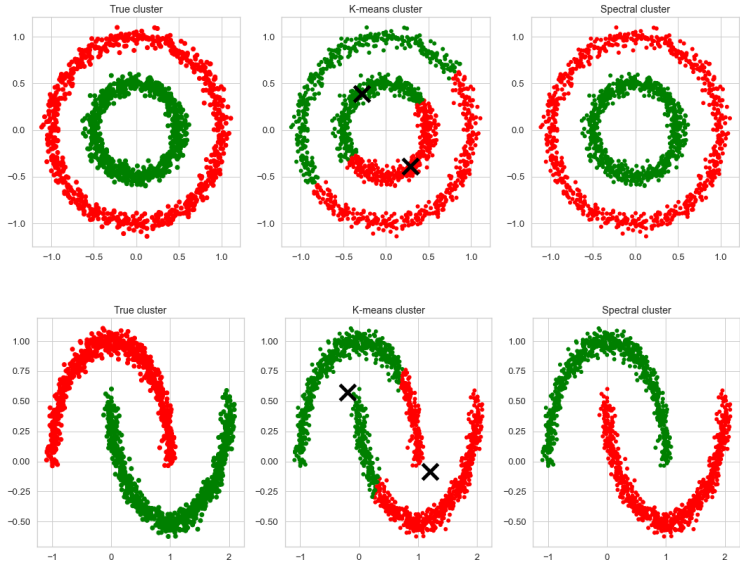
- Perform K-means on the new data points.

Why spectral clustering works?

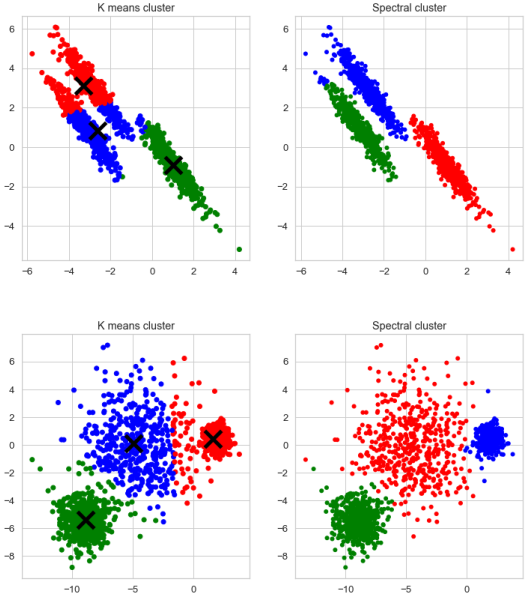
- Think of $P = G^{-1}W$ as a transition matrix. (It can be shown that all elements in each row sum up to 1. Intuitively, this tells us how likely it is to move from one point to another)
- Note that:
$$v^T L v = \frac{1}{2} \sum_{i,j} W_{i,j}(v_i - v_j)^2$$
- So eigenvectors associated with small eigenvalues should have small $|v_i - v_j|$ when $W_{i,j}$ is large. (The points are similar!)

Spectral clustering's a powerful tool when we don't have access to a vector of measurements for each data point. (We just need know how 'similar' they are.)

Spectral clustering



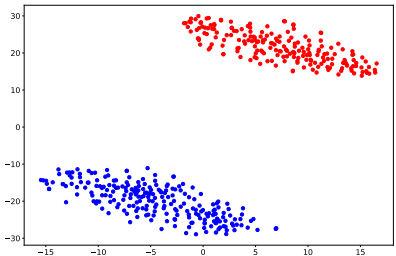
Spectral clustering



Example: clustering legislators based on votes

- Data for U.S. House of Representatives Roll Call Votes. Each line corresponds to a single member of the House, and it contains the results for that member on 669 roll calls: 1 for Yea, -1 for Nay, 0 for no vote.
- We have 434 Representatives.
- Goal: We use voting behavior to reveal legislators' political view (liberal or conservative).

Clustering results



	Name	second_eig	Group
355	Schakowsky	-0.065527	D
429	Woolsey	-0.064166	D
229	Lee	-0.064044	D
11	Baldwin	-0.063970	D
275	Miller, George	-0.063947	D
...	...	...	...
28	Blackburn	0.049626	R
136	Foxx	0.049632	R
3	Akin	0.049699	R
256	McHenry	0.050037	R
288	Neugebauer	0.050160	R

434 rows × 3 columns

From K-means to hierarchical clustering

Recall two properties of K-means clustering:

- It fits exactly K clusters (as specified)
- Final clustering assignment depends on the chosen initial cluster centers

Hierarchical clustering is an alternative that does not rely on any underlying model

- Hierarchical clustering produces a sequence of nested cluster memberships.
- No need to choose initial starting positions and the number of clusters.
- Data points that are similar will end up in the same cluster.

There are different ways to measure similarity.

At one end, all points are in their own cluster, at the other end, all points are in one cluster

Agglomerative vs divisive

Two types of hierarchical clustering algorithms

Agglomerative (i.e., bottom-up):

- Start with all points in their own group
- Until there is only one cluster, repeatedly: merge the two groups that have the smallest dissimilarity

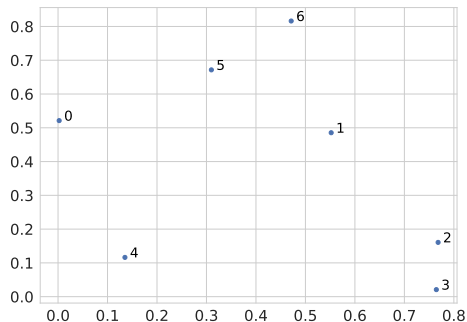
Divisive (i.e., top-down):

- Start with all points in one cluster
- Until all points are in their own cluster, repeatedly: split the group into two resulting in the biggest dissimilarity

Agglomerative strategies are simpler, we'll focus on them.

Simple example

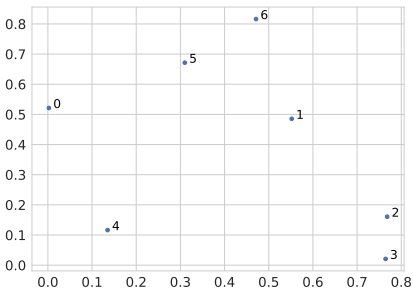
Given these data points, an agglomerative algorithm might decide on a clustering sequence as follows:

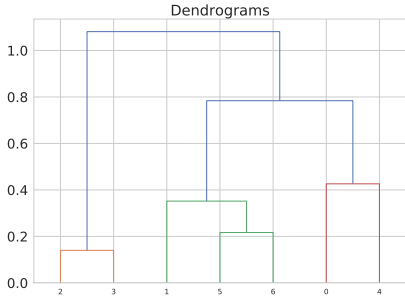


- Step 1: {0}, {1}, {2}, {3}, {4}, {5}, {6};
- Step 2: {0}, {1}, {2, 3}, {4}, {5}, {6};
- Step 3: {0}, {1}, {2, 3}, {4}, {5, 6};
- Step 4: {0}, {1, 5, 6}, {2, 3}, {4};
- Step 5: {0, 4}, {1, 5, 6}, {2, 3};
- Step 6: {0, 4, 1, 5, 6}, {2, 3};
- Step 7: {0, 1, 2, 3, 4, 5, 6}.

Eendrogram

We can also represent the sequence of clustering assignments as a dendrogram:





Note that cutting the dendrogram horizontally partitions the data points into clusters

What's a dendrogram?

Dendrogram: convenient graphic to display a hierarchical sequence of clustering assignments. This is simply a tree where:

- Each node represents a group
- Each leaf node is a singleton (i.e., a group containing a single data point)
- Root node is the group containing the whole data set
- Each internal node has two children nodes, representing the the groups that were merged to form it

Remember: the choice of similarity measure determines how we merge groups of points

Hierarchical Clustering of WSJ News

We have data from 180 topics, each being a vector of length 18,433.

See full graph on this website. Source: Bybee, Kelly, Manela, and Xiu, The Structure of Economic News (2021).