Unsupervised Learning

Fabio G. Cozman - fgcozman@usp.br

November 16, 2018

What can we do?

- We just have a dataset with features (no labels, no response).
  - We want to "understand" the data... no easy to define this.

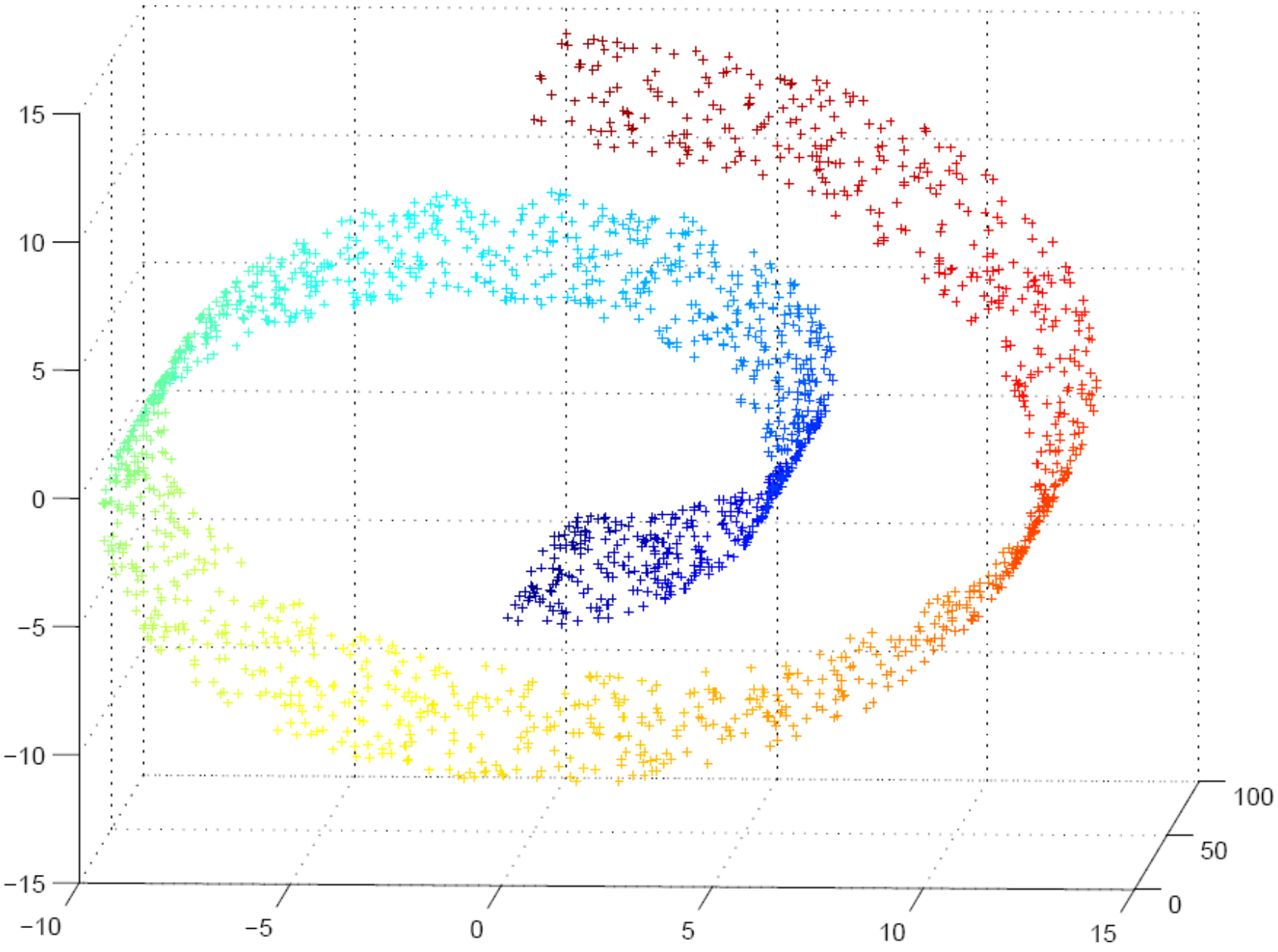
What can we do?

- We just have a dataset with features (no labels, no response).
  - We want to "understand" the data... no easy to define this.
- Possible ideas:
  - Find association rules that indicate relationships amongst variables (market basket analysis).
  - Find compact ways to summarize observations (representation learning).
  - Cluster the data.

Representation learning

- Often it is possible to find compact representations that capture most of the content of observations.
  - Dimension is reduced: easier to visualize, easier to understand.
  - Easier to capture complex patterns (manifolds).
  - In essense, the idea is to "learn (the real) features".

A data manifold



Principal Component Analysis (PCA)

- We have features $X_1, \ldots, X_n$, suitably normalized (zero mean and unit variance).

Principal Component Analysis (PCA)

- We have features $X_1, \ldots, X_n$, suitably normalized (zero mean and unit variance).
- The *first principal component* Z_1 is a linear combination

$$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + \cdots + \phi_{n1} X_n,$$

where $\sum \phi_{i1}^2 = 1$, *and*
such that Z_1 has the largest possible variance.

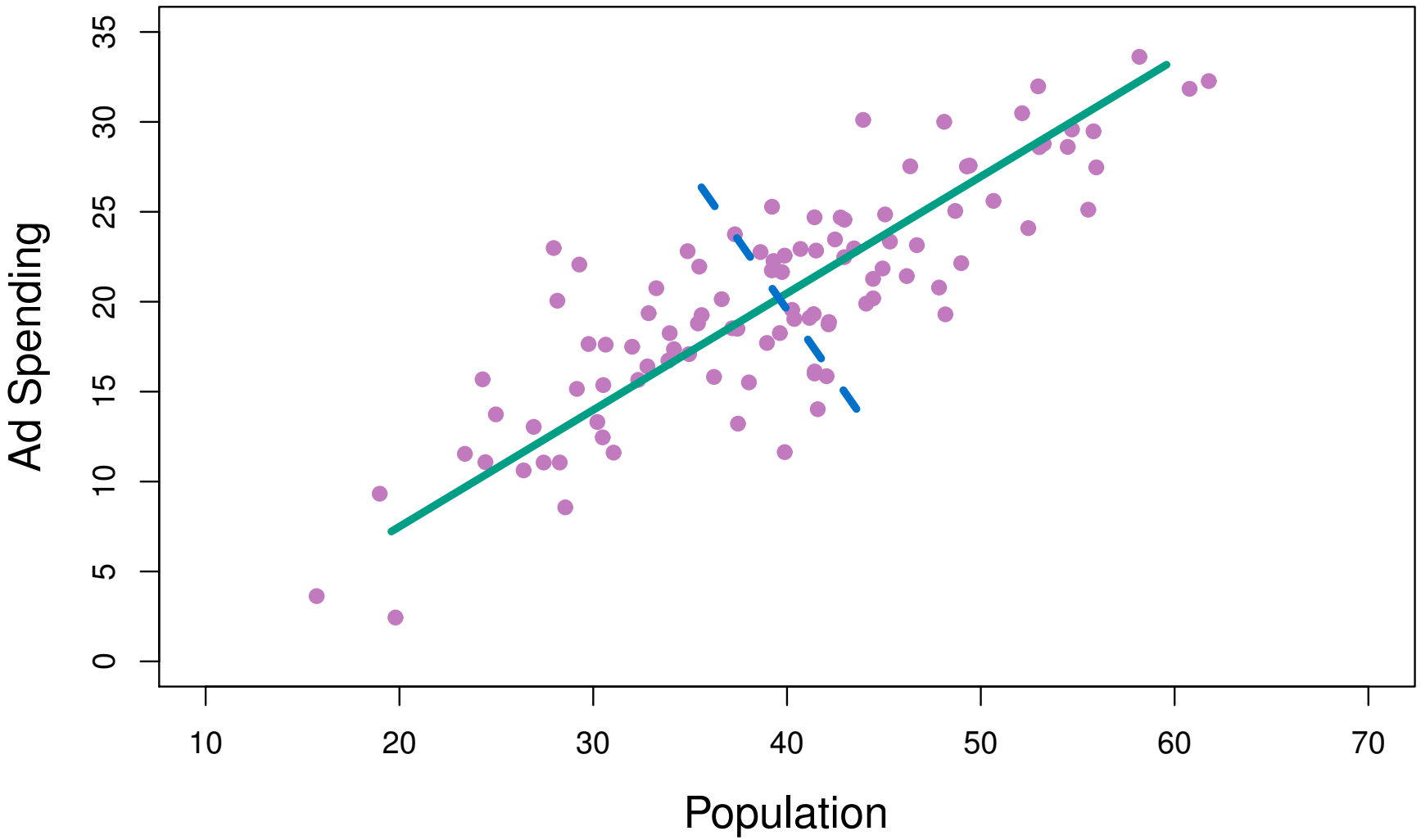
Principal Component Analysis (PCA)

- We have features $X_1, \ldots, X_n$, suitably normalized (zero mean and unit variance).
- The *first principal component* Z_1 is a linear combination

$$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + \cdots + \phi_{n1} X_n,$$

  where $\sum \phi_{i1}^2 = 1$, *and*
  such that Z_1 has the largest possible variance.
- The next principal components similarly maximize variance, being uncorrelated with previous ones (orthogonal to previous ones...).

A two-dimensional example



Loadings

- The coefficients ϕ_{ij} are called *loadings*.
- As features are centered, the mean of Z_i is zero as well.
  - Hence sample variance of Z_1 is

$$\sum_{j=1}^{N} \left(\sum_{i=1}^{n} \phi_{n1} x_{ij} \right)^2 .$$

Building the representation

- To find the loadings for the first principal component, maximize sample variance:

$$\max_{\phi_{11},\ldots,\phi_{n1}} \sum_{j=1}^{N} \left(\sum_{i=1}^{n} \phi_{n1} x_{ij} \right)^2 ,$$

subject to $\sum_{i=1}^{n} \phi_{i1}^2 = 1$.

- Quadratic problem (can be solved).

Building the representation, continued

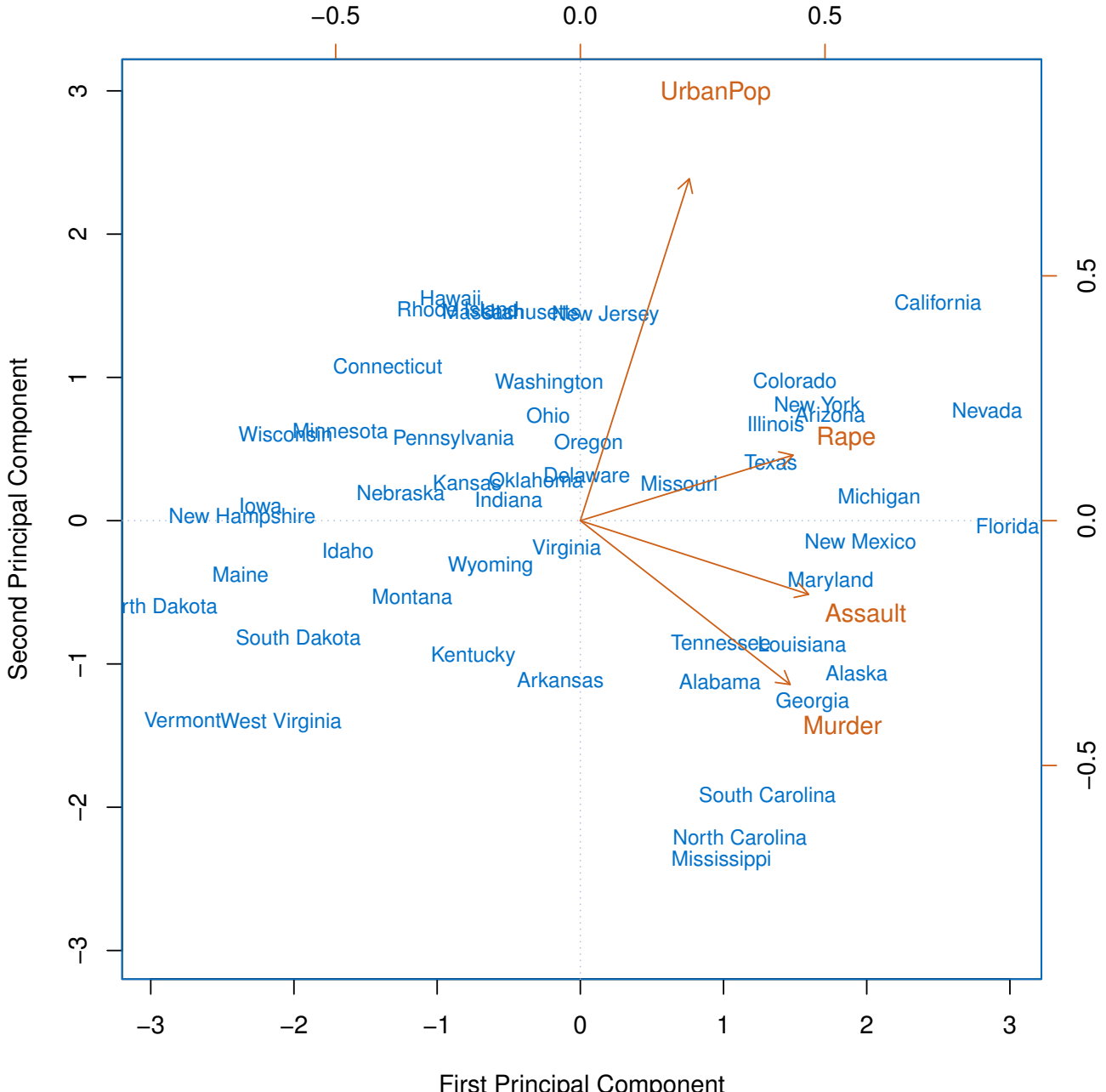
- After finding Z_1, find Z_2: same problem, but in orthogonal space.
- Again, quadratic problem (can be solved).
- Then repeat for other principal components.

Building the representation, continued

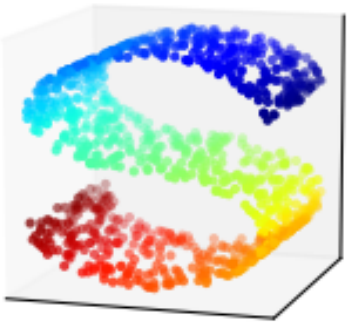
- After finding Z_1, find Z_2: same problem, but in orthogonal space.
- Again, quadratic problem (can be solved).
- Then repeat for other principal components.

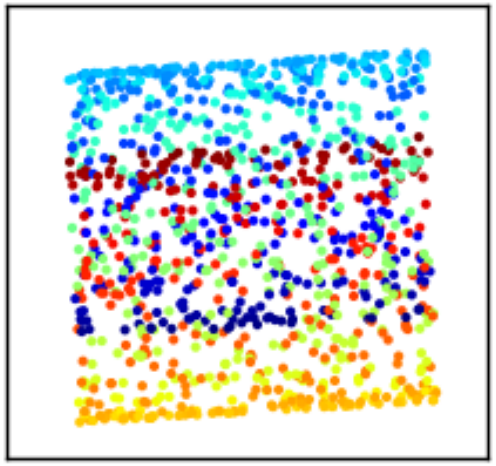
- In the end, an orthogonal basis that spans the feature space.
- Actually, there are fast algorithms that use matrices.

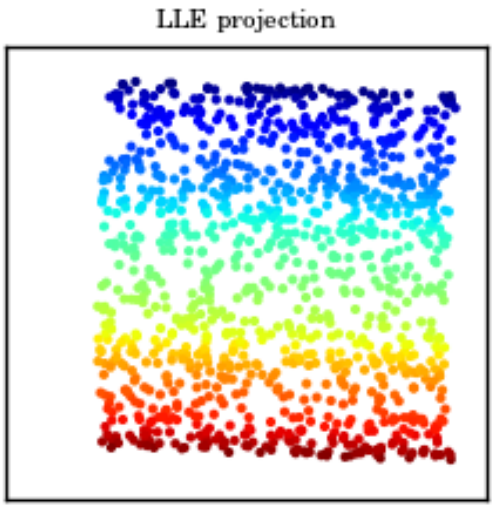
Example: the USArrests dataset

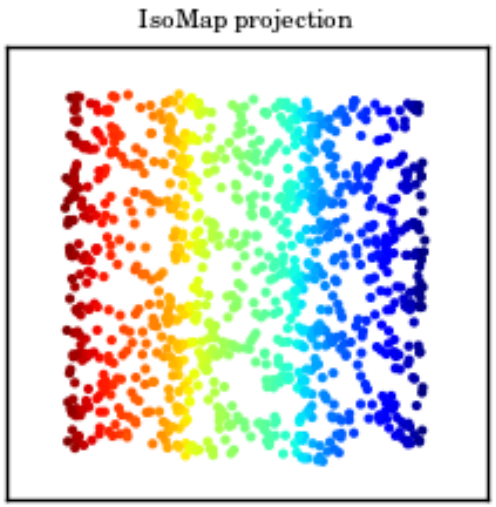


Other manifold learning techniques









From AstroML documentation, by Jake VanderPlas
(http://www.astroml.org/book_figures/chapter7/fig_S_manifold_PCA.html).

Another interpretation for PCA

1. First loading vector yields the line that is closest to the N observations.

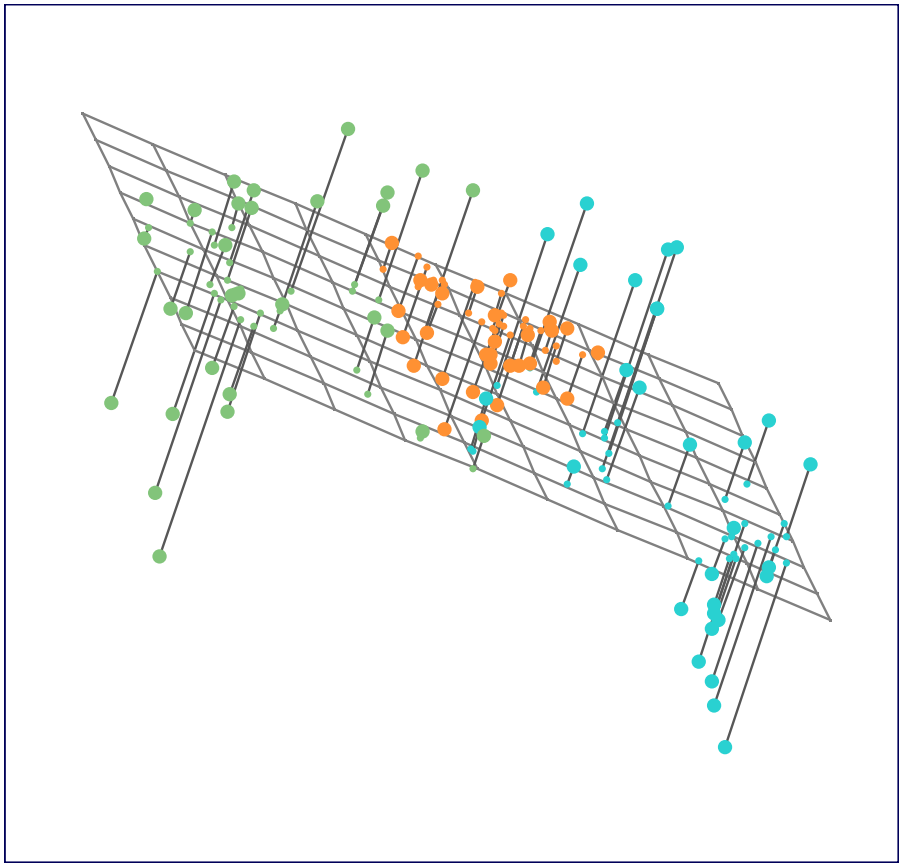
Another interpretation for PCA

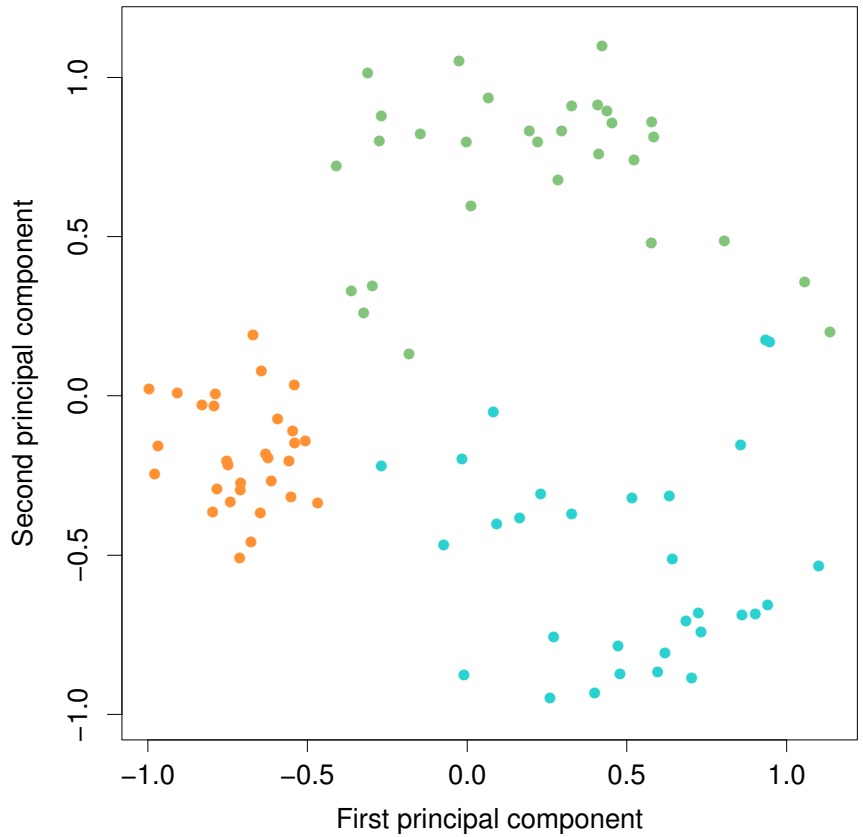
1. First loading vector yields the line that is closest to the N observations.
2. First and second loading vectors yield the plance that is closest to the N observations.

Another interpretation for PCA

1. First loading vector yields the line that is closest to the N observations.
2. First and second loading vectors yield the plance that is closest to the N observations.
3. And so on.

Example in three dimensions





The important idea

- If we retain just a few principal components, we explain most of the variance in the data.
- Hopefully by keeping a few principal components we have better features.

Explained variance

- The *total variance* is $\sum_{i=1}^{n}(1/N)\sum_{j=1}^{N}x_{ij}^2$.
- The *variance explained* by principal component Z_k is

$$(1/N)\sum_{j=1}^{N}z_{jk}^2.$$

Explained variance

- The *total variance* is $\sum_{i=1}^{n}(1/N)\sum_{j=1}^{N}x_{ij}^2$.
- The *variance explained* by principal component Z_k is

$$(1/N)\sum_{j=1}^{N}z_{jk}^2.$$

- The *proportion of variance explained* by Z_k is

$$\frac{(1/N)\sum_{j=1}^{N}z_{jk}^2}{\sum_{i=1}^{n}(1/N)\sum_{j=1}^{N}x_{ij}^2}.$$

Main practical question

- How many principal components to keep?
- No easy answer.
  - Check proportion of explained variance; stop when it is “small” enough.
  - Try a few possibilities.

PCA in supervised learning

- PCA is a tool to explore data.
- But it can also be used to produce a compact set of features.
  - Any classifier can be preceded by some feature summarization through PCA.

Really doing PCA

- Principal components are produced by eigenvector computation.

Really doing PCA

- Principal components are produced by eigenvector computation.

- Take matrix $\mathbf{A}$ and suppose

$$\mathbf{Av} = \lambda \mathbf{v}.$$

Then λ is an eigenvalue, and $\mathbf{v}$ is an eigenvector.

Suppose matrix $\mathbf{A}$ is symmetric

- Then eigenvalues are real; eigenvectors are orthogonal and real.
- Build matrix with eigenvectors

$$\mathbf{U} = [\mathbf{u}_1 \mathbf{u}_2 \dots \mathbf{u}_n]$$

and diagonal matrix with eigenvalues

$$\mathbf{D} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}.$$

- Then:

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{U}^T.$$

Covariance matrix

- Consider the matrix

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \dots \\ \mathbf{x}_N \end{bmatrix}$$

and compute a symmetric matrix

$$\frac{1}{N-1}\mathbf{X}^T\mathbf{X}.$$

- Each element of this matrix is a "covariance"

$$\frac{1}{N-1}\sum_{i=1}^{N} x_{ij}x_{kj}.$$

Decomposition

- Compute matrix $\mathbf{U}$ of eigenvectors of $\mathbf{X}^T\mathbf{X}$, sorted from largest to smallest eigenvalue.
- Then principal components are

$$\begin{bmatrix} Z_1 \\ \vdots \\ Z_n \end{bmatrix} = \mathbf{U}^T \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix}.$$

- The most numerically stable way to compute this transformation is through the *Singular Value Decomposition* (SVD) of $\mathbf{X}$.

A summary:

1. Compute $\mathbf{m} = \sum_{j=1}^{N} \mathbf{x}_j$ and

$$\mathbf{S} = (\mathbf{X} - \mathbf{m})^T (\mathbf{X} - \mathbf{m}).$$

2. Compute the eigenvalues and eigenvectors of $\mathbf{S}$.
3. Order the eigenvectors by the value of their eigenvalues (from largest to smallest).
4. Select the K eigenvectors with largest eigenvalues, and build the matrix

$$\mathbf{U} = [\mathbf{u}_1 \dots \mathbf{u}_K].$$

5. Then define

$$\mathbf{Z} = \mathbf{U}^T (\mathbf{X} - \mathbf{m}).$$

Reconstruction

- Suppose we have $\mathbf{m}$ and $\mathbf{U}$.
- Suppose we have a point $\mathbf{x}$; then we compute

$$\mathbf{z} = \mathbf{U}^T(\mathbf{x} - \mathbf{m}).$$

- We can now see the "approximate" point

$$\hat{\mathbf{x}} = \mathbf{m} + \mathbf{U}\mathbf{z}.$$

(See Barber's book, Figures 15.1 and 15.3.)

Clustering

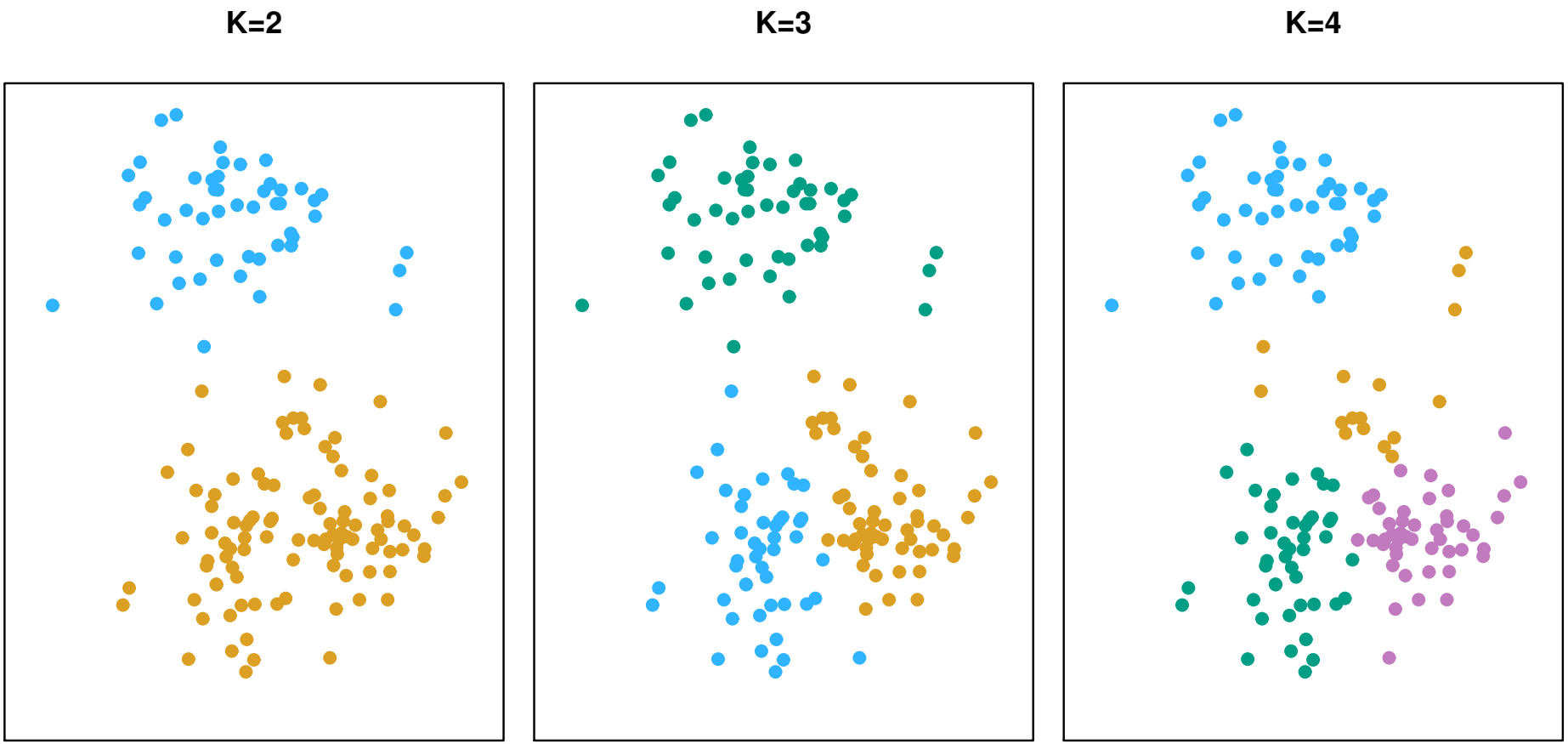
- The goal is to find subgroups of the dataset; each sugbroup is a *cluster*.
  - Often clustering appears as synonym for unsupervised learning.
- Examples:
  - Clustering species in biology experiment.
  - Clustering clients of supermarket based on purchases.

A few points

- Clustering is not easy to evaluate.
  - A bit subjective.
  - It depends on measure of "similarity".
  - It must somehow maximize intra-cluster similarity, and minimize inter-cluster similarity.

- Two important methods:
  - K-means.
  - hierarchical clustering.

Example



(Obtained with K-means, discussed later.)

K-means: Setup

- Features $X_1, \ldots, X_n$; clusters $C_1, \ldots, C_K$ to be found (all disjoint).

K-means: Setup

- Features $X_1, \ldots, X_n$; clusters $C_1, \ldots, C_K$ to be found (all disjoint).
- We must find clusters that

$$\max_{C_1,\ldots,C_k} \sum_{k=1}^{K} s(C_k)$$

for some measure of intra-cluster similarity s.

- One "reasonable" measure:

$$s(C_k) = -\frac{1}{|C_k|} \sum_{x',x'' \in C_k} \sum_{i=1}^{n} (x_i' - x_i'')^2.$$

K-means: Setup

- Features $X_1, \ldots, X_n$; clusters $C_1, \ldots, C_K$ to be found (all disjoint).
- We must find clusters that

$$\max_{C_1,\ldots,C_k} \sum_{k=1}^{K} s(C_k)$$

for some measure of intra-cluster similarity s.

  - One "reasonable" measure:

$$s(C_k) = -\frac{1}{|C_k|} \sum_{x',x'' \in C_k} \sum_{i=1}^{n} (x_i' - x_i'')^2.$$

- K-means offers an approximate solution to the resulting maximization problem.

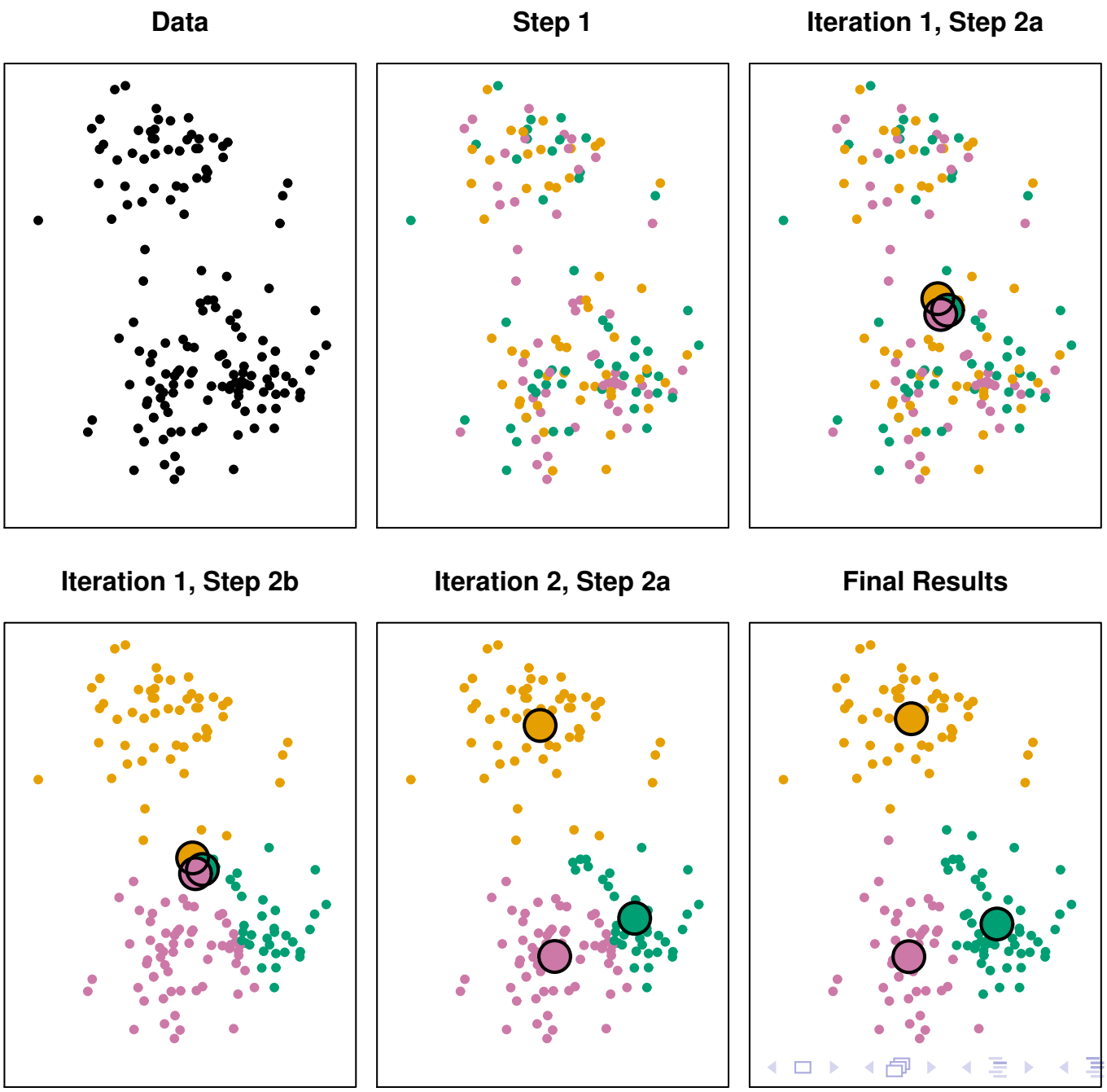
K-means

Input: dataset; distance measure; number K.

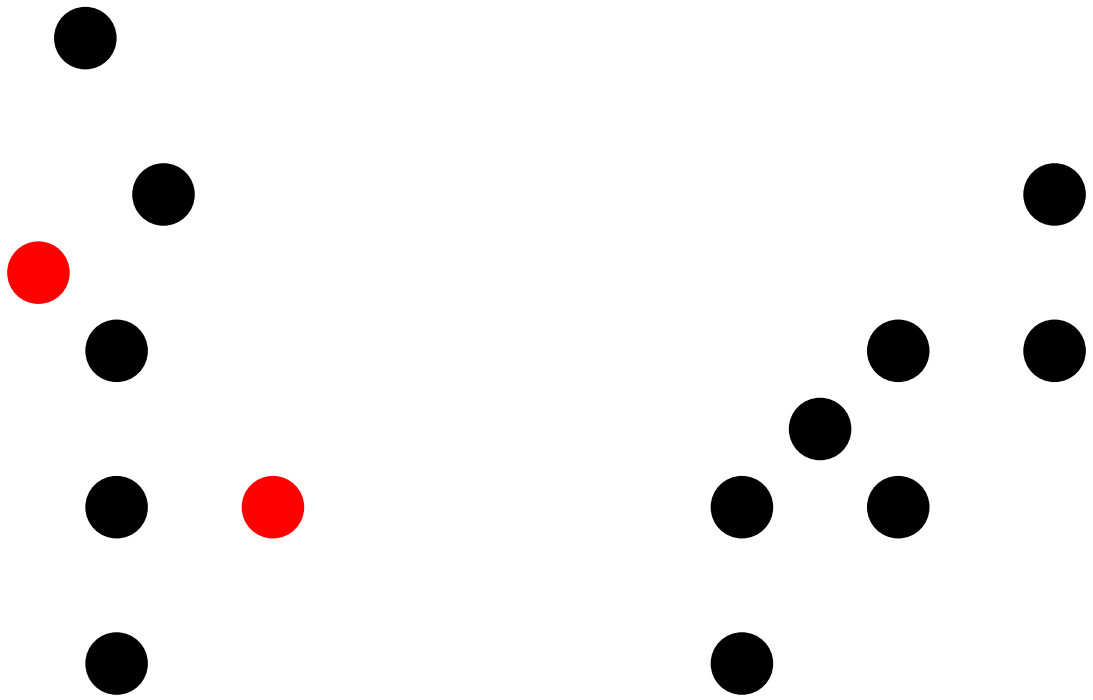
1. Select initial assignment to K clusters:
   - One possibility: assign each point randomly.
   - Another one: randomly select K points as centroids and assign points to closest centroids.
2. Iterate until stopping criterion is met:
   1. Compute centroid for each cluster.
   2. Assign each point to closest centroid (use distance measure).

Note: centroid of points is the average across coordinates.

Example


Data
Step 1
Iteration 1, Step 2a
Iteration 1, Step 2b
Iteration 2, Step 2a
Final Results

Another example



Red points: randomly selected centroids at initial step.

When to stop?

- When a maximum number of iterations is run.
- When centroids stop changing (or display little change).
- When the measure $\sum_{k=1}^{K} s(C_k)$ displays little change.

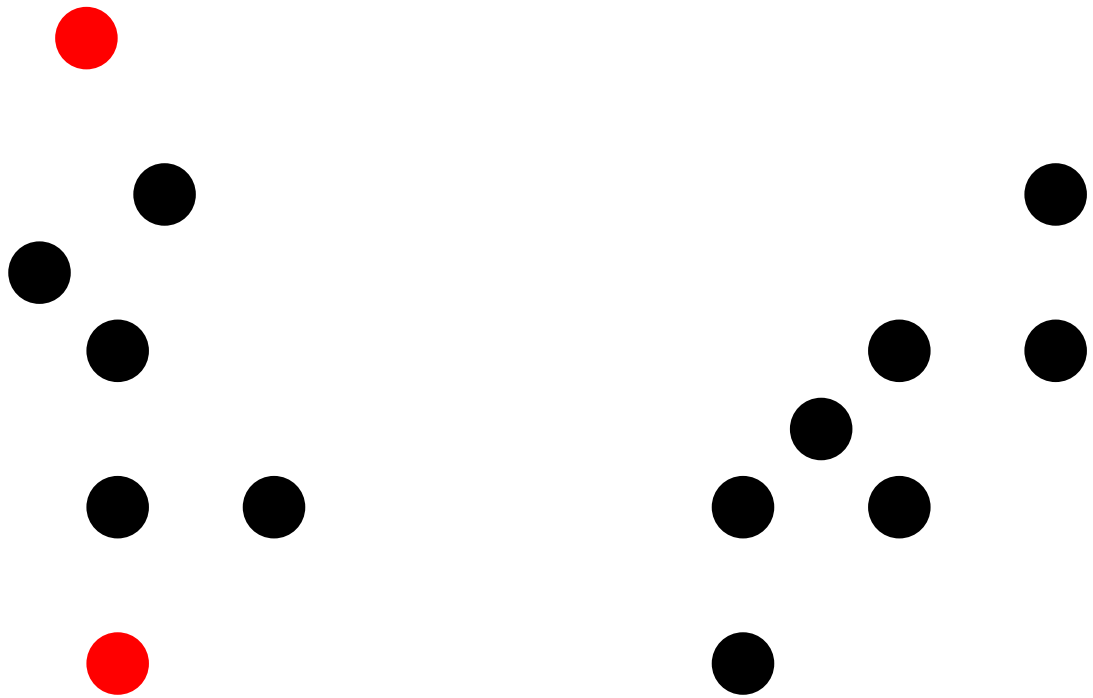
K-means: strengths

- Easy to understand, easy to implement.
- Cost is proportional to number of points, number of clusters, and number of iterations.

K-means: weaknesses

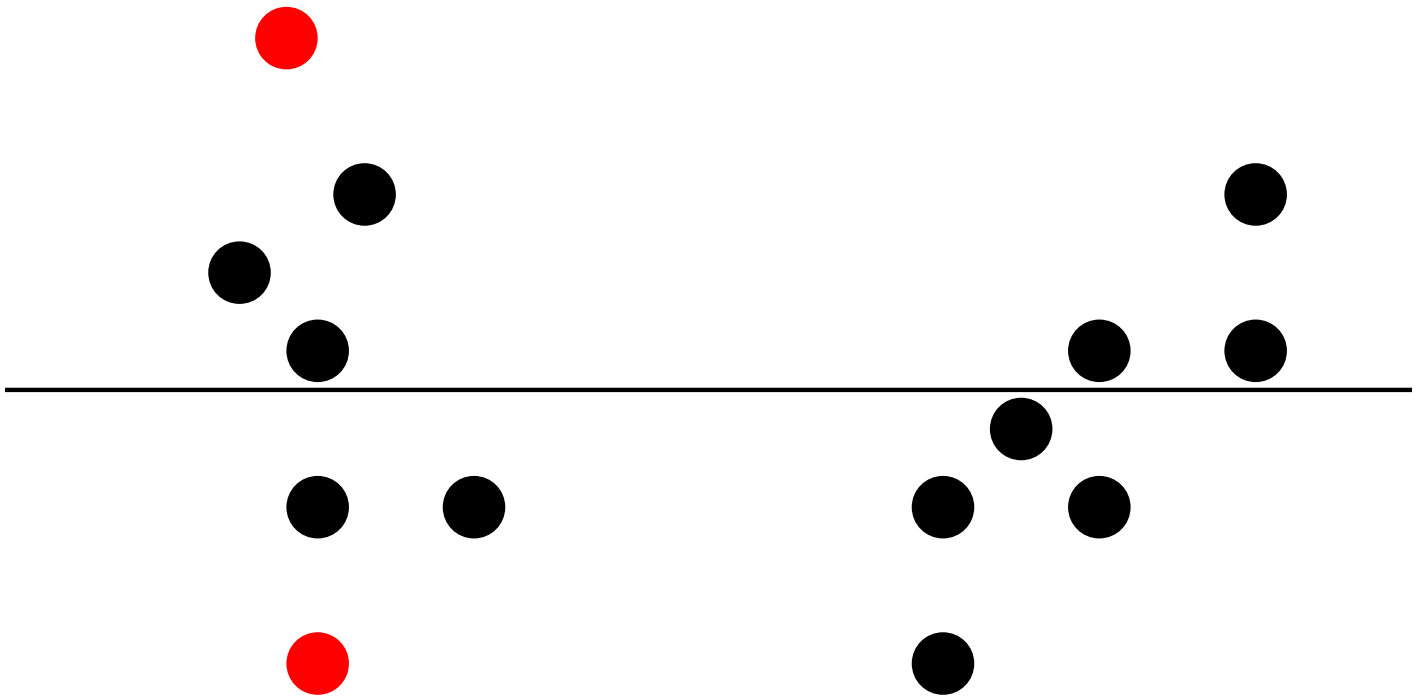
- Requires K.
- Results depend on initial guess.
- Sensitive to outliers.
- No real theoretical guarantees; stopping is somewhat ad hoc.
- Centroid computation may be meaningless for categorical data.

Sensitivity to initial guess



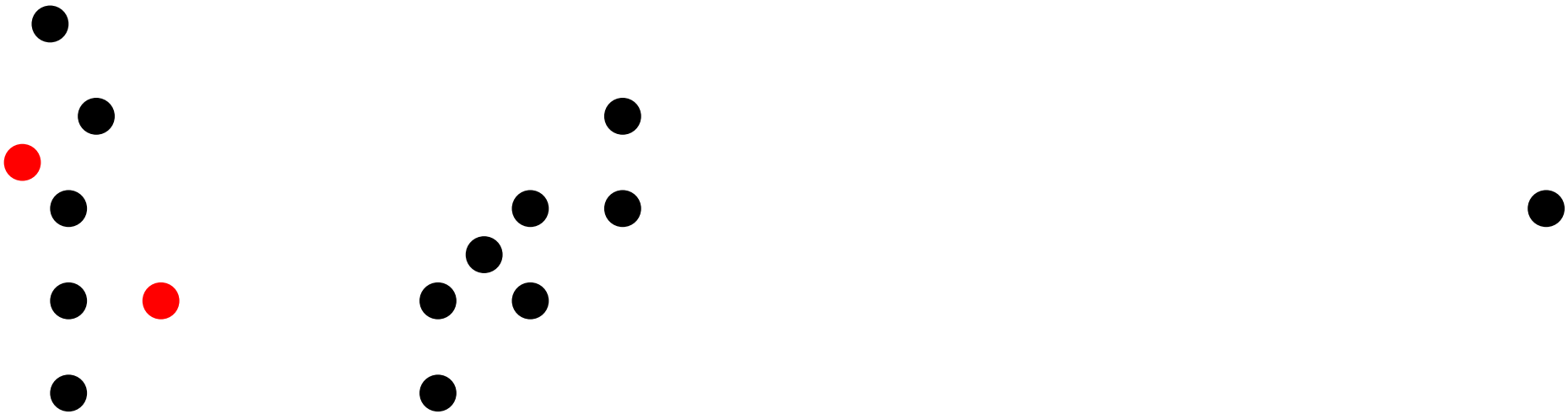
Red points: randomly selected centroids at initial step.

Sensitivity to initial guess



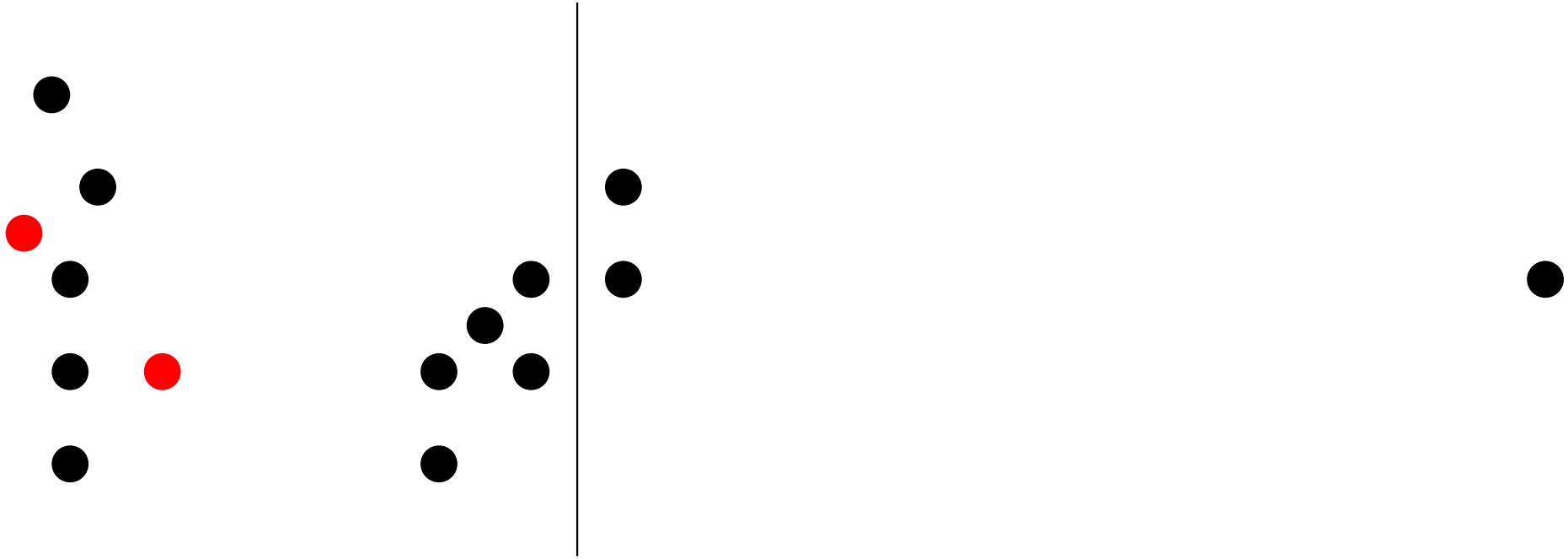
Red points: randomly selected centroids at initial step.

Sensitivity to outliers



Red points: randomly selected centroids at initial step.

Sensitivity to outliers



Red points: randomly selected centroids at initial step.

Some techniques

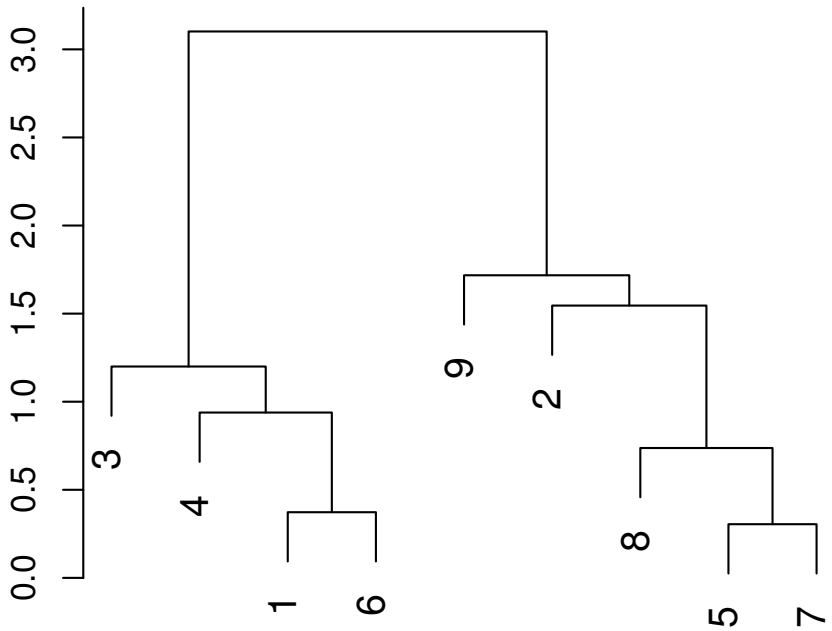
- Run K-means several times, then select best clustering according to $\sum_{k=1}^{K} s(C_k)$.
- Remove points that are "too far" from the cloud of points in dataset.
- Instead of centroid, use a more "robust" representation for clusters...
  - select a few points randomly to compute centroid...
  - use K-medoids (medoid is a point whose sum of distances to other points in cluster is minimal).

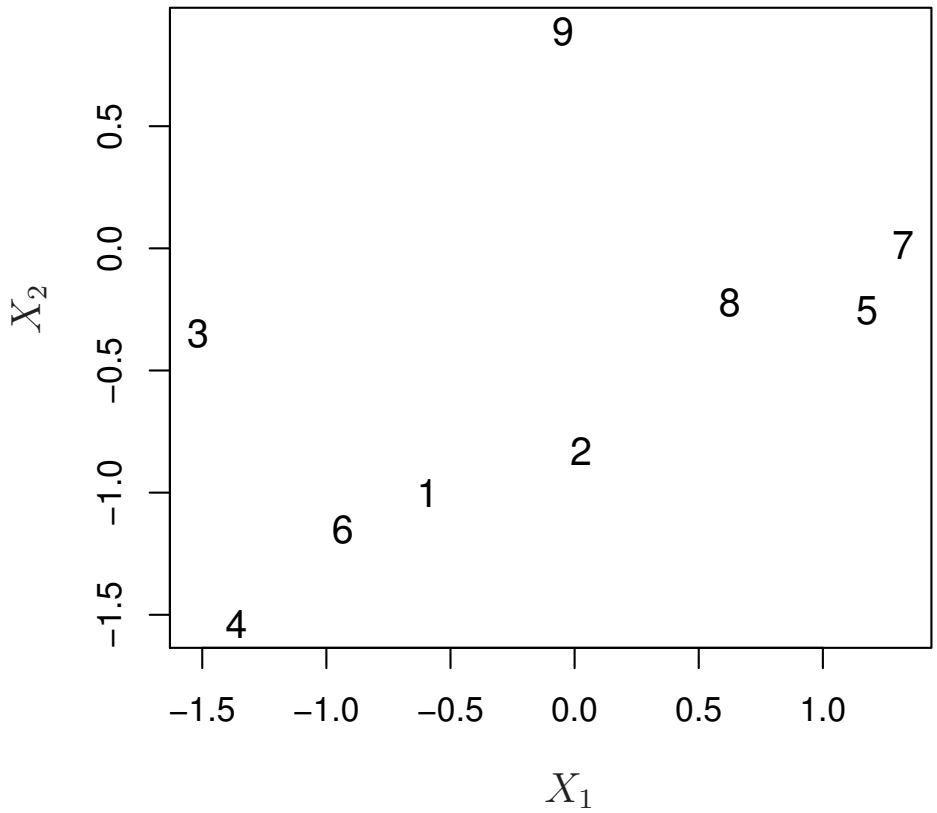
Example

Hierarchical clustering

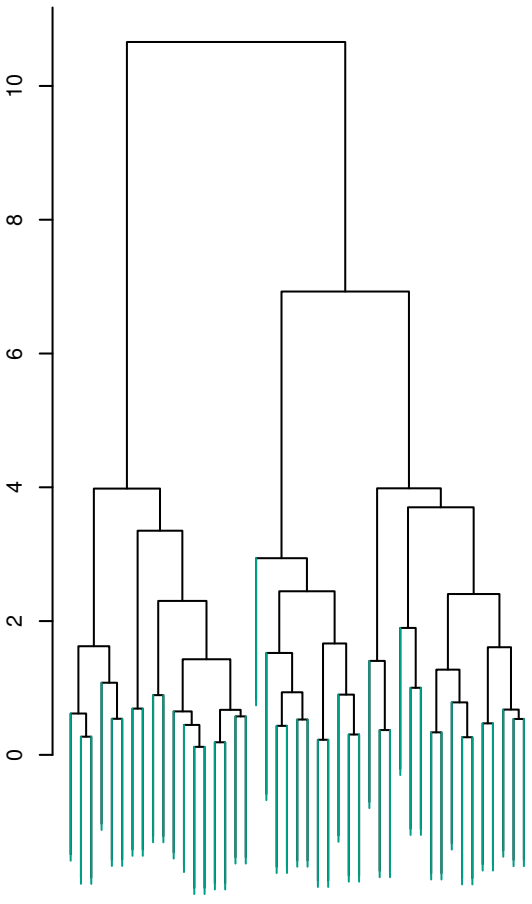
- Main idea: build a representation for all possible clusters, instead of working with a fixed K.
- Usual representation is *dedrogram*.
- Dendrograms can be build bottom-up (agglomerative) or top-down (divisive).
  - Most popular (by far): agglomerative clustering.

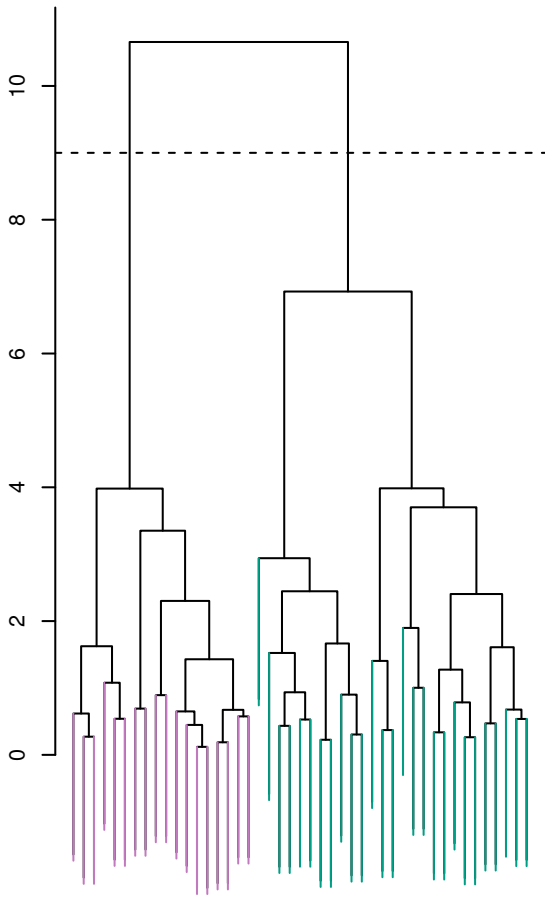
A dendrogram

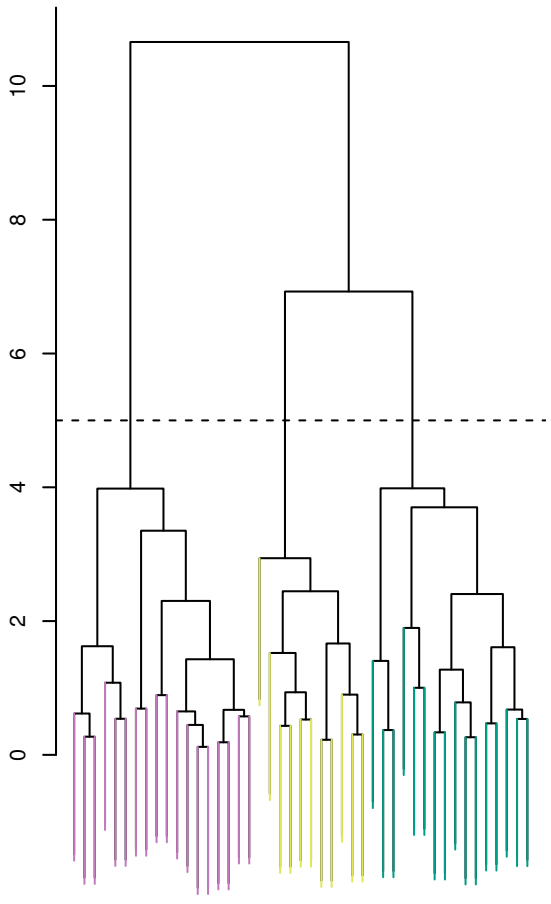




Interpreting a dendrogram







Agglomerative clustering

1. Compute distance between all N points, using some given distance measure.
2. Repeat whenever possible:
    1. Find the two closest clusters.
    2. Fuse these clusters in the dendrogram (height is the distance between them).

What is the distance between clusters?

Complete linkage: distance between two points (one in each cluster) that are most distant.

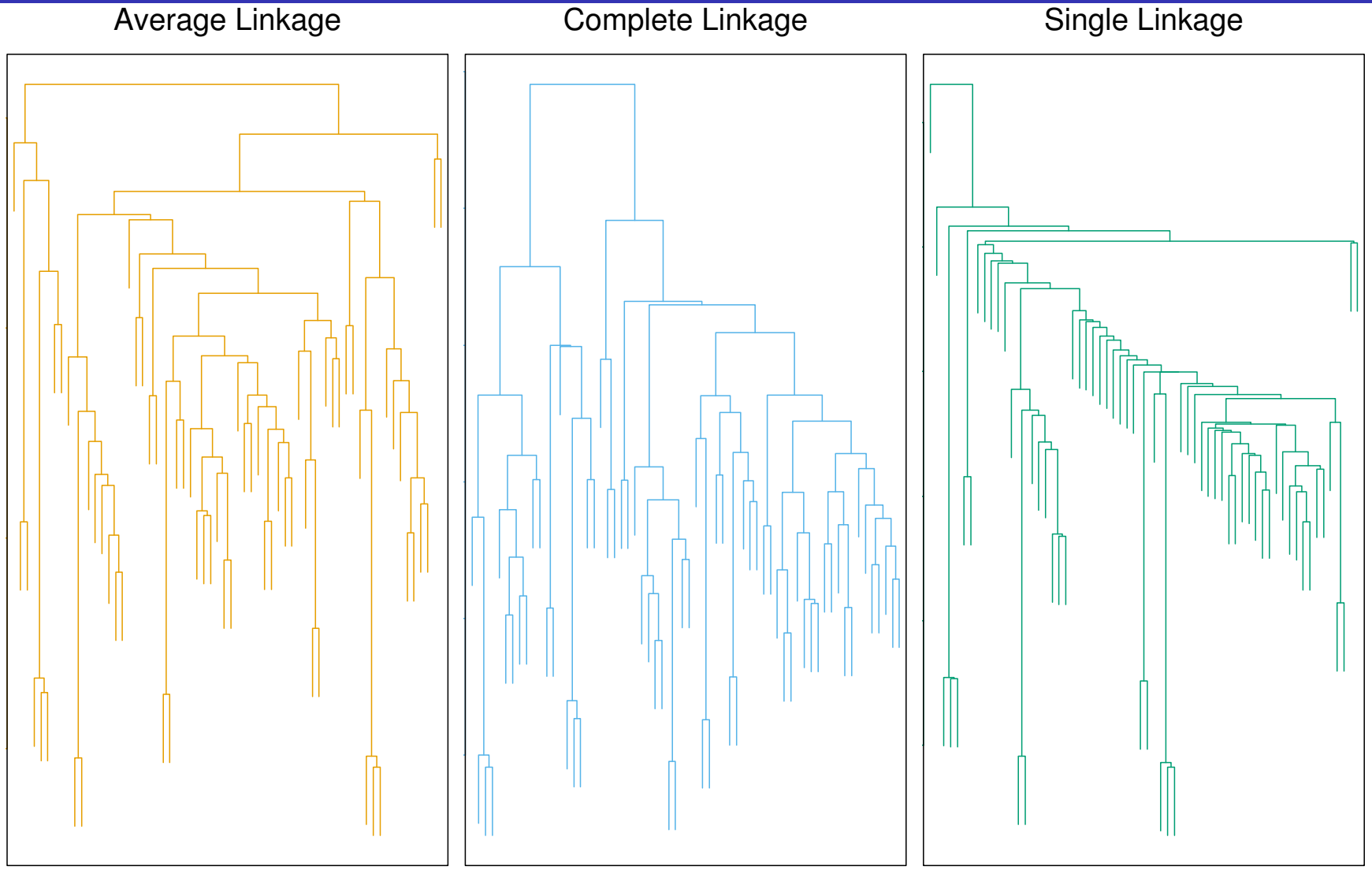
Single linkage: distance between two points (one in each cluster) that are closest.

Average linkage: average of the distance between t all pairs of points (one in each cluster).

Centroid linkage: distance between centroids of both clusters.

Centroid linkage may be a bit difficult to represent.

Example



Note: single linkage may lead to "long clusters"...

Distance measures

- Common: (weighted/squared) Euclidian distance.

$$\sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2}.$$

- (weighted) Manhattan distance:

$$|x_1 - y_1| + \cdots + |x_n - y_n|.$$

- (weighted) Chebyshev distance:

$$\max\left(|x_1 - y_1|, \ldots, |x_n - y_n|\right).$$

(Note: usually good idea to standardize.)

Distance measures: categorical data

- Observations of binary variables (0s and 1s): just the proportion of disagreements.
  - Jaccard coefficient: useful when only of the labels is relevant (the other label is the "normal" one):

$$\frac{\text{number of disagreements}}{\text{number of disagreements} + \text{number of agreements of "relevant" class}}.$$

- For categorical variables:
  - Proportion of disagreements.
  - Use one-hot encoding and techniques for binary variables.

Latent models

- A different scheme is to assume a statistical model.
- For instance, suppose there is class variable with K labels, and $\mathbb{P}(X = x|Y = y)$ for each label.
  - Estimate the probablities, then assign labels.
  - This is a *mixture model*.
  - The class variable is a *latent variable*.

Latent models

- A different scheme is to assume a statistical model.
- For instance, suppose there is class variable with K labels, and $\mathbb{P}(X = x | Y = y)$ for each label.
  - Estimate the probablities, then assign labels.
  - This is a *mixture model*.
  - The class variable is a *latent variable*.
- There are very complex latent models around (often called *topic models*).

We are not discussing latent models in this course...

Evaluating a clustering

- Not easy; quite subjective and application-dependent.
  - Try a few things and compare the results!
  - Visualize the clusters (projections).
  - Check sum of inter/intra-cluster distances.
  - Use a global metric (for instance, *silhoutte*).
- Many algorithms have been proposed; hard to compare them.
  - K-means and dendrograms are still the champions.

Silhoutte

- For each point x:
  - $a(x)$ is the average distance between x and all other points in the same cluster.
  - Find the average distance between x and all other points in each other cluster; denote by $b(x)$ the smallest of these averages.

Silhoutte

- For each point x:
  - $a(x)$ is the average distance between x and all other points in the same cluster.
  - Find the average distance between x and all other points in each other cluster; denote by $b(x)$ the smallest of these averages.
- Define

$$s(x) = \begin{cases} 1 - a(x)/b(x) & \text{if } a(x) < b(x); \\ 0 & \text{if } a(x) = b(x); \\ b(x)/a(x) - 1 & \text{if } a(x) > b(x). \end{cases}$$

  - Note: $s(x) \in [-1, 1]$ (1 is "good"; -1 is "bad").

Silhoutte

- For each point x:
  - $a(x)$ is the average distance between x and all other points in the same cluster.
  - Find the average distance between x and all other points in each other cluster; denote by $b(x)$ the smallest of these averages.
- Define

$$s(x) = \begin{cases} 1 - a(x)/b(x) & \text{if } a(x) < b(x); \\ 0 & \text{if } a(x) = b(x); \\ b(x)/a(x) - 1 & \text{if } a(x) > b(x). \end{cases}$$

  - Note: $s(x) \in [-1, 1]$ (1 is "good"; -1 is "bad").
- Take the average of $s(x)$ for all points as the measure (the silhoutte) of the clustering.
- A higher silhoutte is better than a lower one...

A final note

Some of the figures in this presentation are taken from *An Introduction to Statistical Learning, with applications in R* (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.