Clustering

Overview

- Supervised vs. Unsupervised Learning
- K-means
  - Algorithm
  - Choosing K (# of clusters)
- Review: Curse of Dimensionality

- Hierarchical clustering
  - Algorithm
  - Choosing K (# of clusters)

Unsupervised Learning

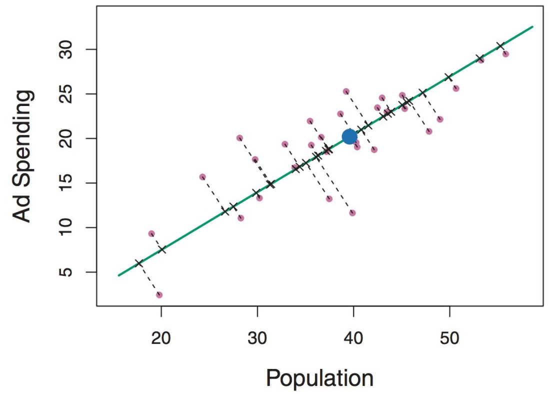
- No response variable, y
  - Just based on predictors, X_1, X_2, X_3, …, X_p
- A fuzzy endeavor...
  - Not cross-validating
    - to choose best “model” in usual sense
    - to know how well you’re doing
- Can be useful as
  - preprocessing step for supervised learning
  - better understand features

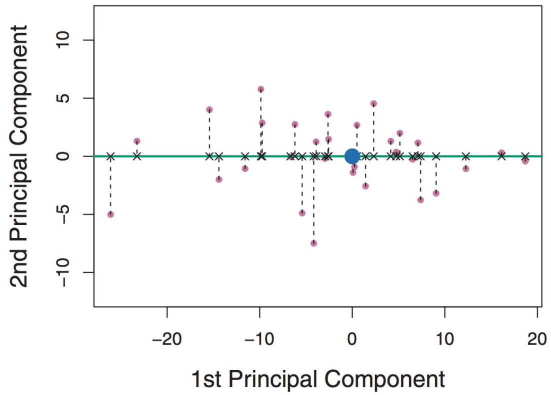
Unsupervised Learning

Two most common and contrasting unsupervised techniques

PCA

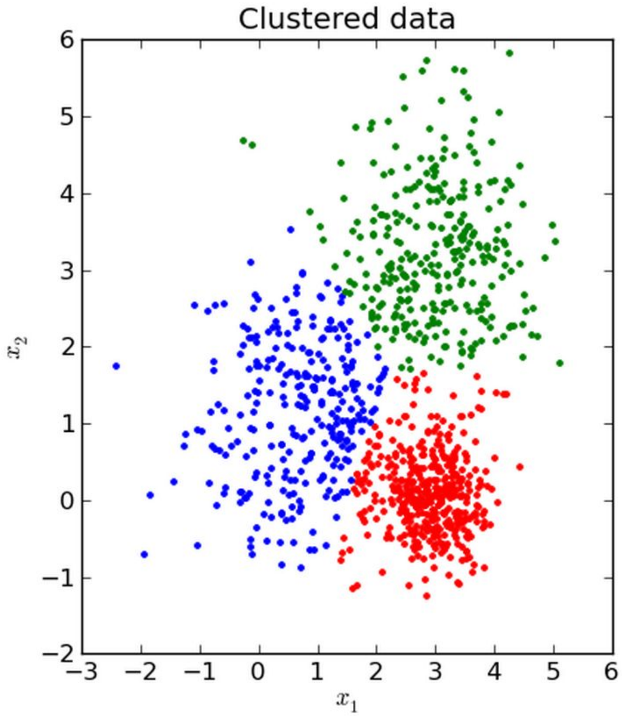
Low-dim representation of data that explains good fraction of variance





Clustering

Find homogenous subgroups among data



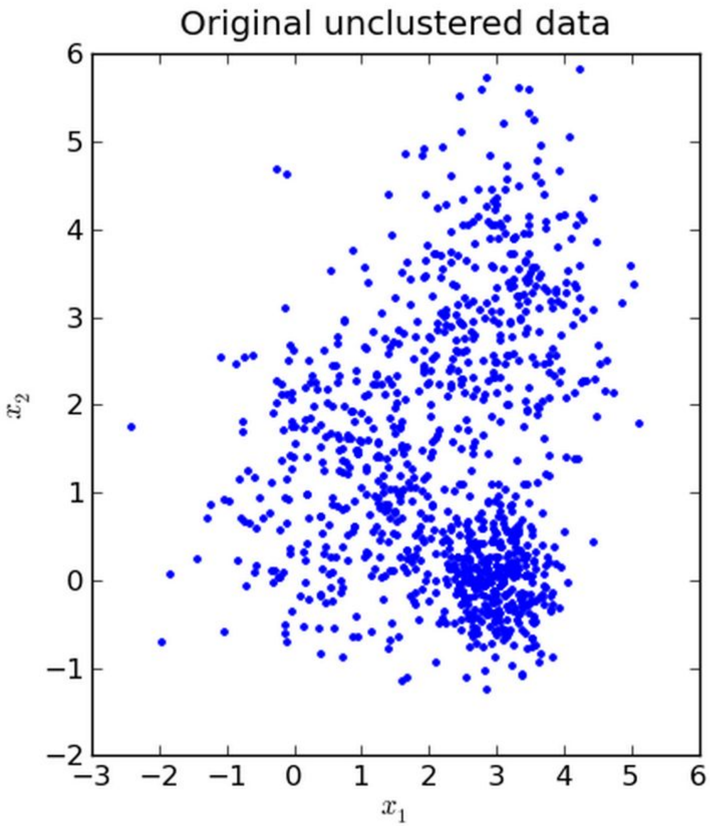
Supervised Learning

- **The label is the supervisor!**
  No label ⇔ Not supervised
  - K-means clustering is not supervised learning, nor is hierarchical clustering
  - PCA is not supervised learning
  - Though again, both can be used in supervised learning!

- Supervised Learning
  - Linear, Logistic, Lasso, Ridge
  - Decision Trees, Bagging, Random Forest, Boosting
  - SVM
  - kNN

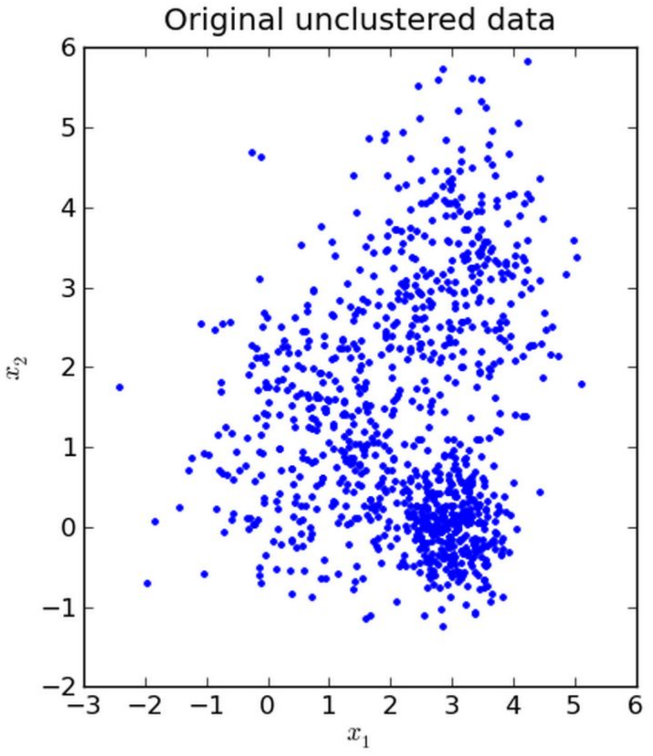
What is clustering?

Divide data into distinct subgroups such that observations within each group are quite similar



What is clustering?

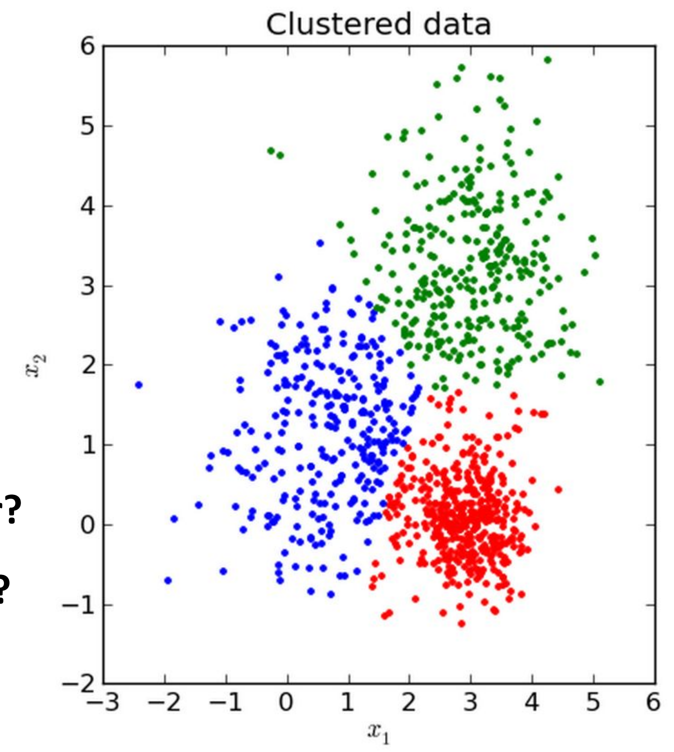
Divide data into distinct subgroups such that observations within each group are quite similar



How many subgroups?

What's considered similar?

How am I even doing this?

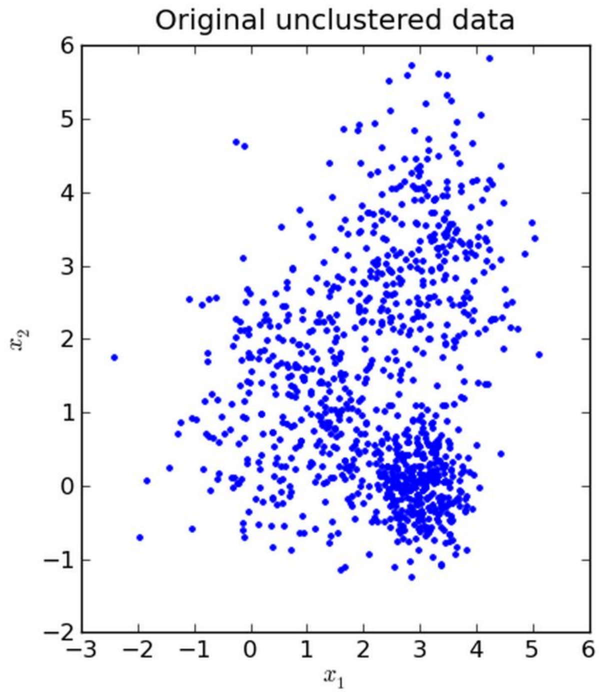


K-means

Idea: Want "within-cluster variation" to be small

Suppose: A fixed K, say K=3. Want to assign each of *n* data point to one of 3 clusters, such that "within-cluster variation" is smallest

- There are K^n possible choices! Pretty unwieldy



K-means

- Again, want to partition data into K subgroups while minimizing within-cluster variation
- More formally....

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \underline{\mathrm{WCV}(C_k)} \right\}$$

where WCV for k-th cluster is the sum of all the pairwise Euclidean distances-squared

$$\mathrm{WCV}(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

$|C_k|$ is number of observations in k-th cluster

K-means

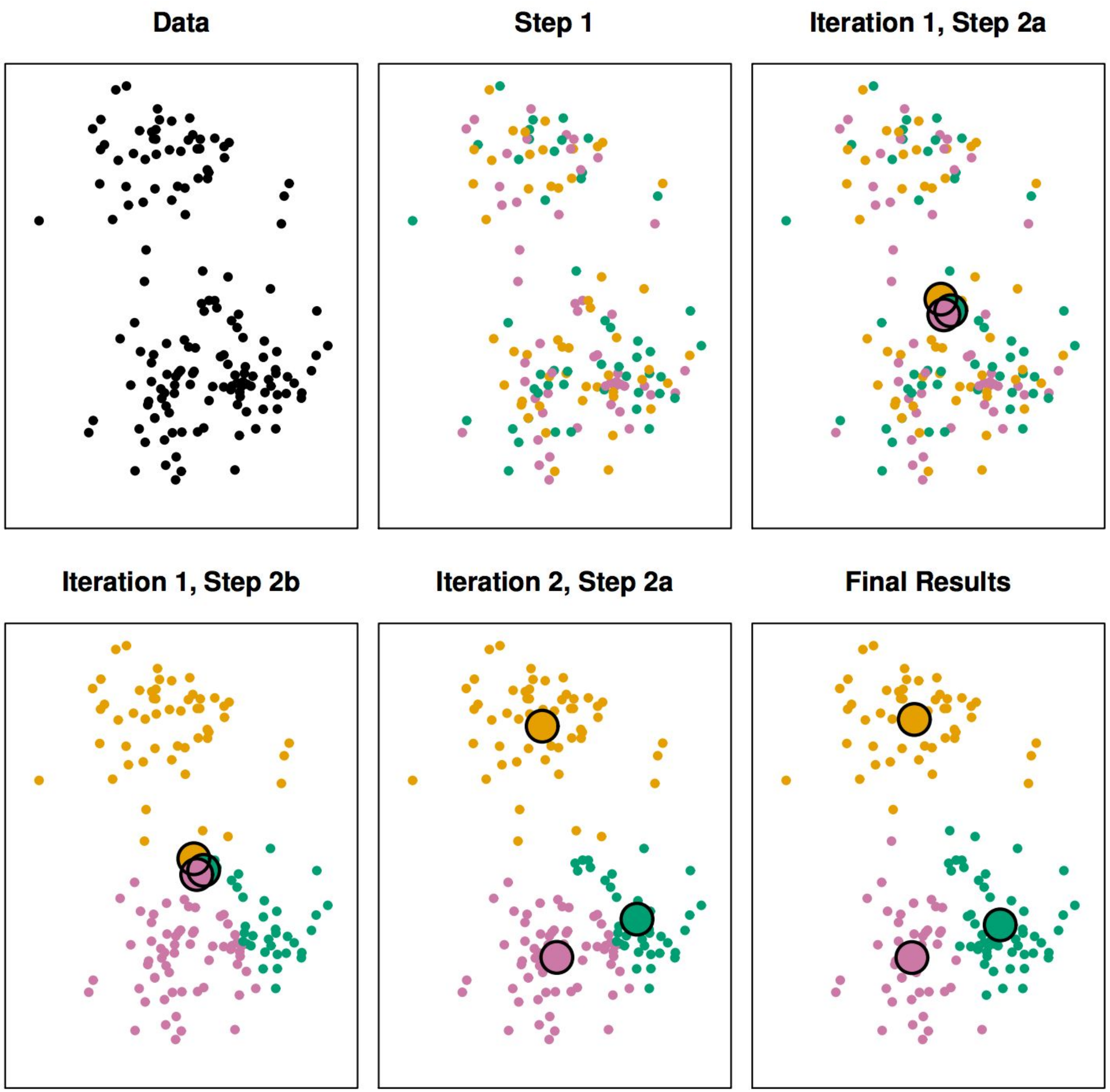
Altogether, we're picking $C_1, \ldots C_K$ such that

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$$

But again the problem is that there are K^n ways. Too many!

K-means algorithm

For K=3…



K-means algorithm

(1) Randomly assign number, from 1 to K, to each data point.**

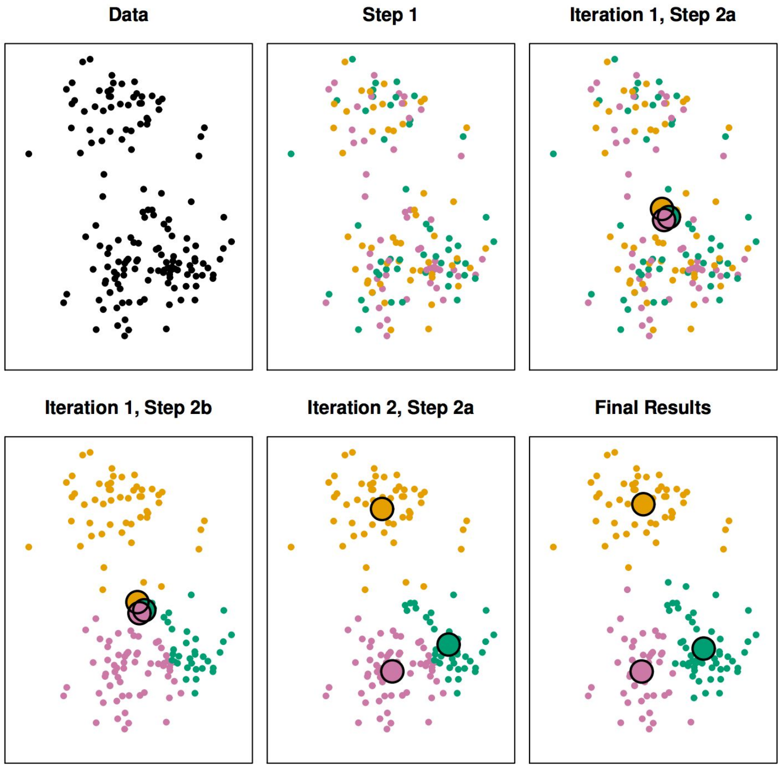
(2) Repeat until cluster assignments stop changing

  a. For each of K clusters, compute cluster **centroid** by taking vector of p feature means

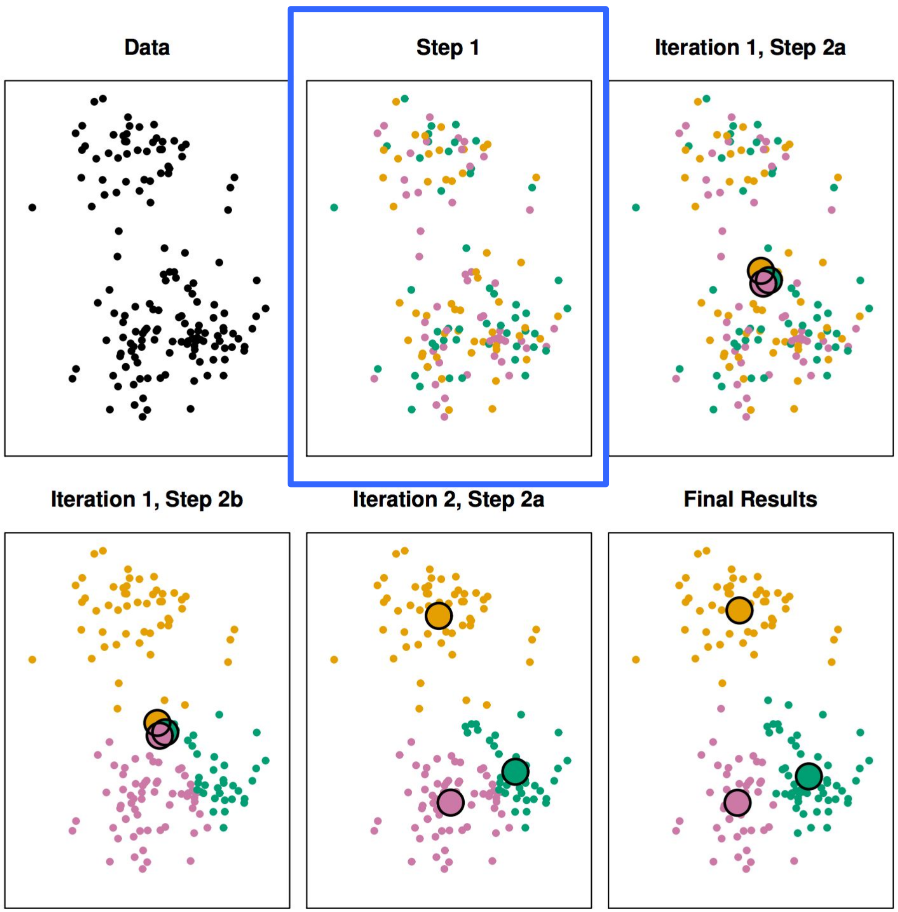
  b. Assign data point to cluster for which centroid is closest (Euclidean)

**Other initializations possible: choose K random data points to be initial centroids.

See also: K-means++



K-means algorithm



Finds local optimum!

Results depend on random initialization**

Solution

Try multiple initializations and pick one with lowest

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$$

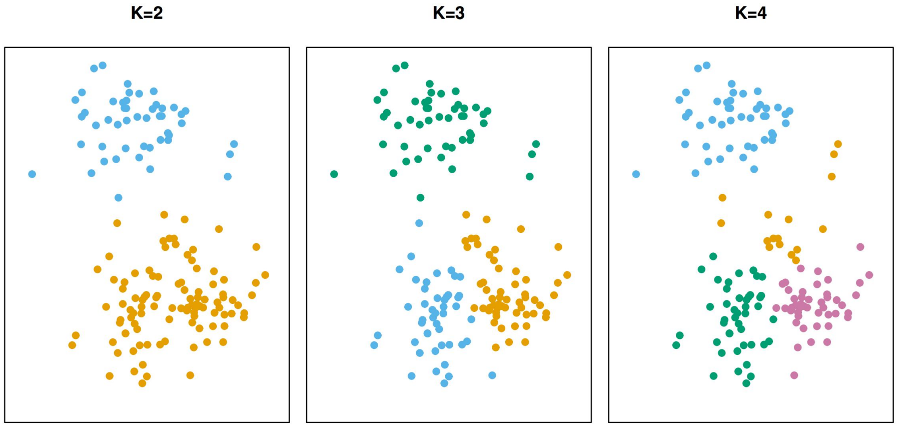
** Again, consider smarter initializations such as kmeans++ http://en.wikipedia. org/wiki/K-means%2B%2B

Choosing K

- No easy answer
- A fuzzy endeavor
  - May just want K similar groups
  - But more often, want something useful or interpretable that exposes some interesting aspect of data
    - Presence/absence of natural distinct groups
    - Descriptive statistics about groups
  - Ex. Are there certain segments of my market that tend to be alike?
    - Ex. middle-aged living in suburbs who log-in infrequently

Choosing K

- Fuzziness aside, there are many methods we can employ to choose K
- Three popular ones:
  - "Elbow" method
  - GAP statistic
  - Silhouette Coefficient

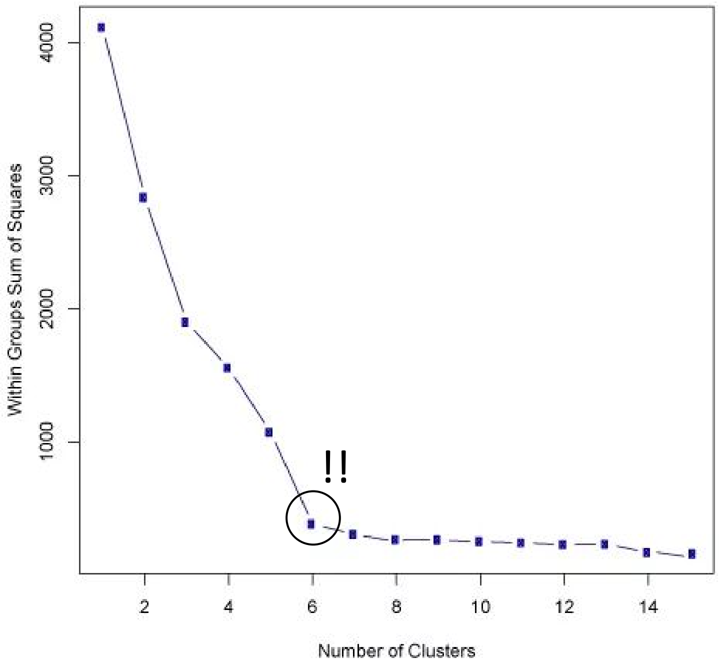


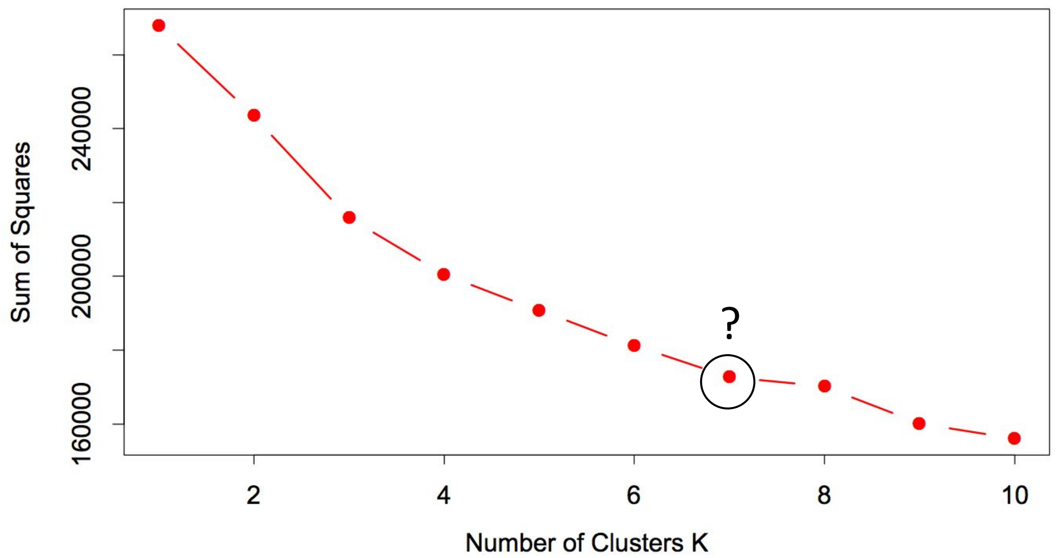
Choosing K – "Elbow" method

- <u>Same Idea</u>: Choose a number of clusters so that adding another cluster doesn't give us that much more

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} ||x_i - x_{i'}||^2$$

Within Cluster Point Scatter
A natural loss function is the sum pairwise distances of the points within each cluster, summed over all clusters.

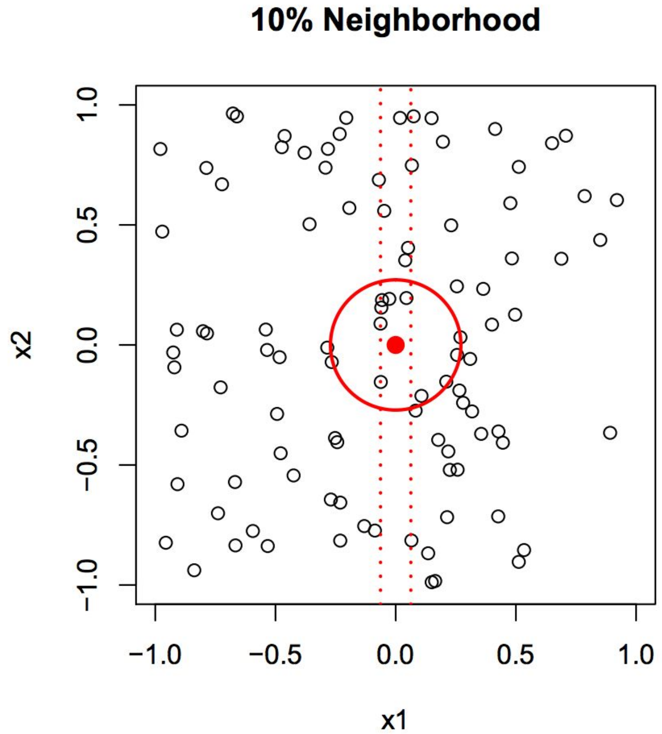


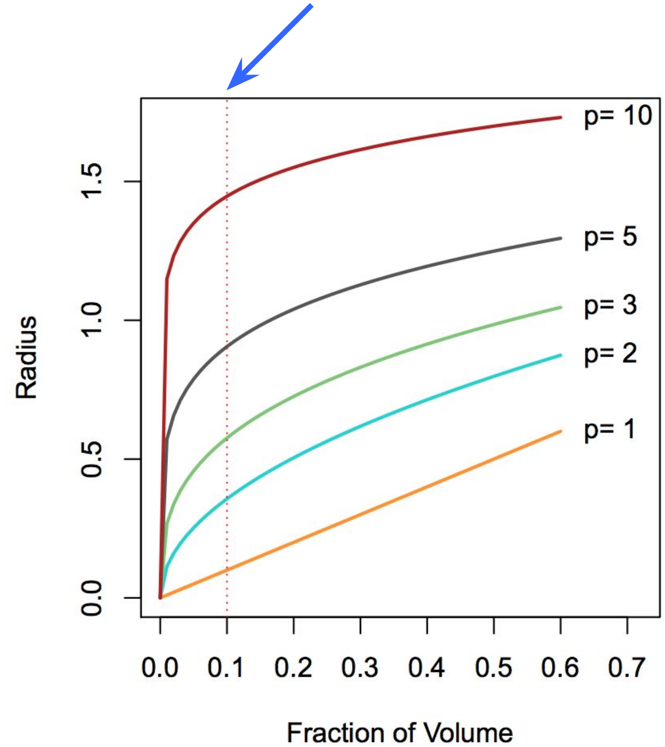


Curse of Dimensionality

- Distance models (kNN, k-means, hierarchical clustering) problematic in high-dimensions/features
- Nearest neighbors can be "far" in high dimensions
- Sparsity of sample points!

Curse of Dimensionality



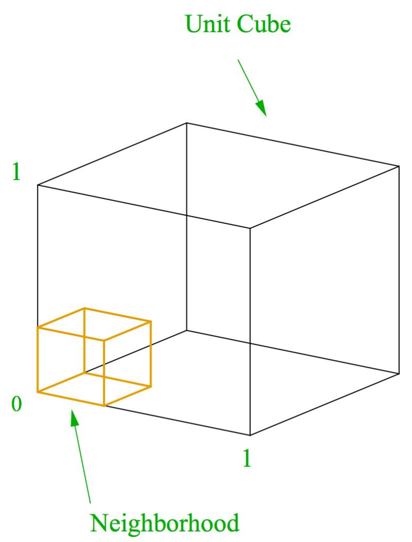


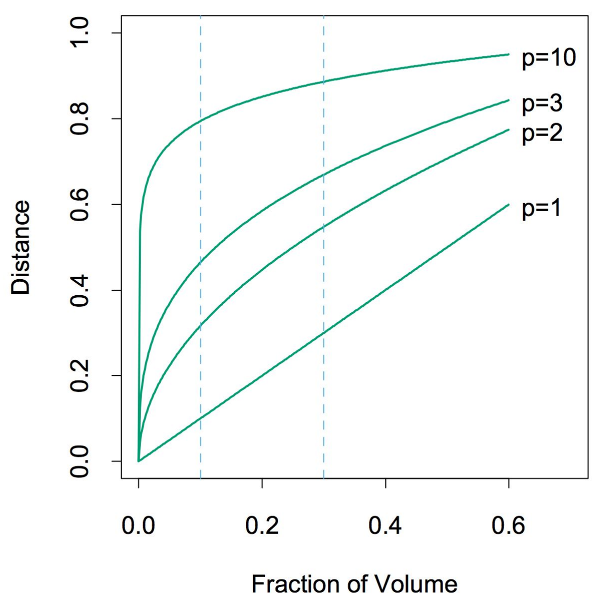
- p = 1 just involves variable x1
- p = 2 involves x1 and x2
  - Notice radius of circle in 2 dimensions is much bigger than radius in 1 dimension

Curse of Dimensionality

Another way to think about dimensionality and its curse

- Hyper-cubical neighborhood about target point to capture fraction ***v*** of the the unit volume
- Expected edge length will be: $$e_p(v) = v^{1/p}$$

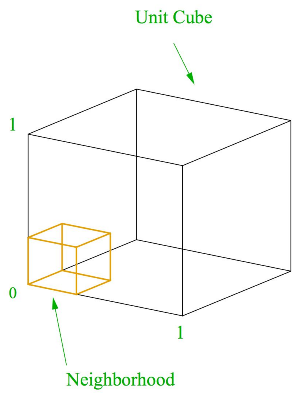


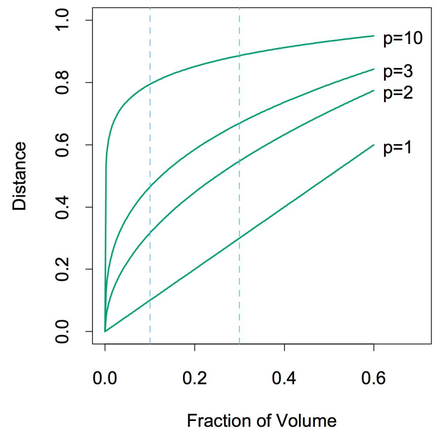


Can you work out out the 10% neighborhood for the unit cube case?

How much more data do we need to compensate for increasing dimensions (p)?

Curse of Dimensionality





Expected edge length

$$\boxed{e_p(v) = v^{1/p}}$$

Sampling density proportional to

$$\boxed{N^{1/p}}$$

p is dimensions of input space
N is number of points

Edge length example: Suppose interested in a v = 10% neighborhood

p = 1 $\longrightarrow$ edge = (0.1)^1 = 0.1
p = 10 $\longrightarrow$ edge = (0.1)^(1/10) = 0.794

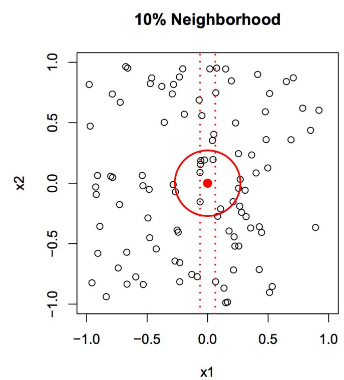
Sampling density example: How to achieve equivalent density in higher dimensions

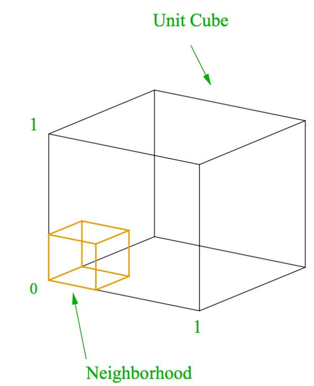
If N_1 = 100 represents dense sample for a single dim feature space
To achieve same density for 10 inputs, we need N_{10} = 100^10 points

Curse of Dimensionality - Takeaways

- k-means, or any method involving this sort of distancing, suffers majorly from curse of dimensionality
  - Nearest neighbors “far” in high dimensions (even for p = 10)

- We can mathematically think of idea of “far” and sparsity of points in high dimensions using both radii approach and hypercube approaches





- It takes **a lot** of data to make up for increase in dimensions

Expected edge length

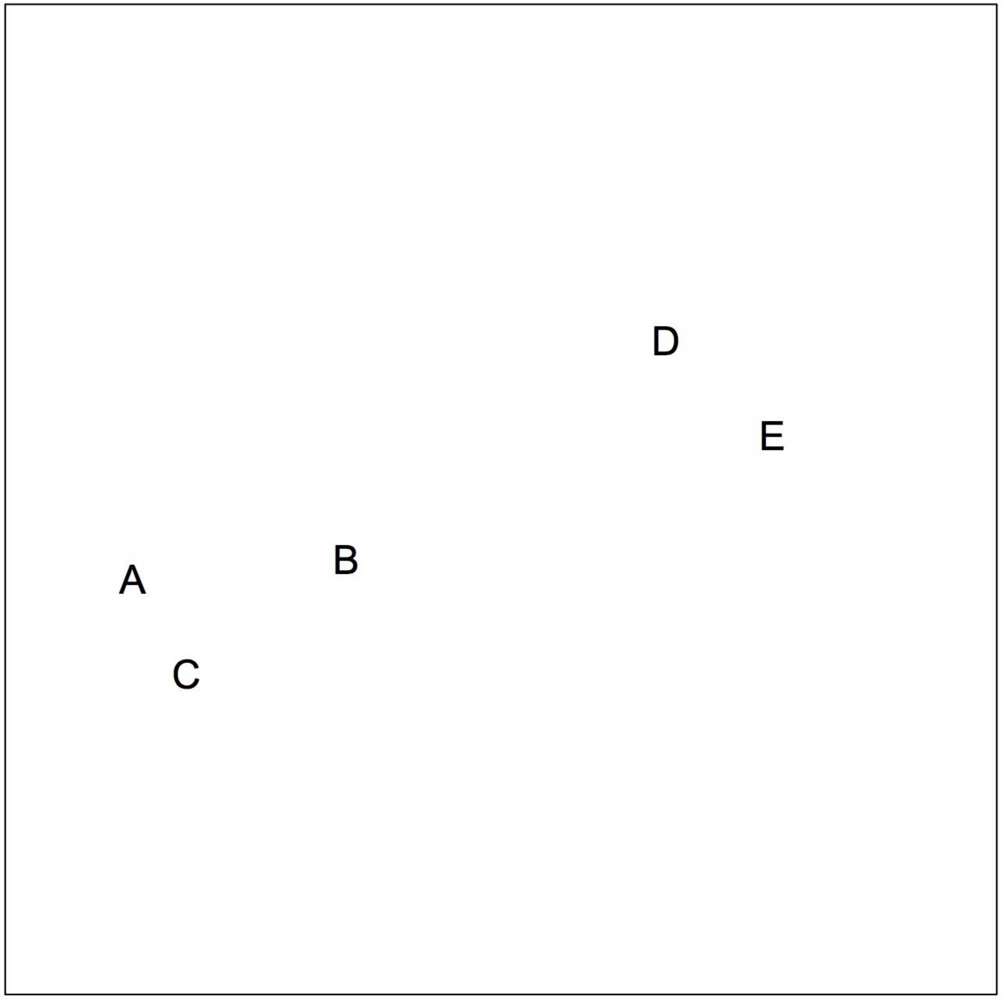
$$e_p(v) = v^{1/p}$$

Sampling density $\propto$ to

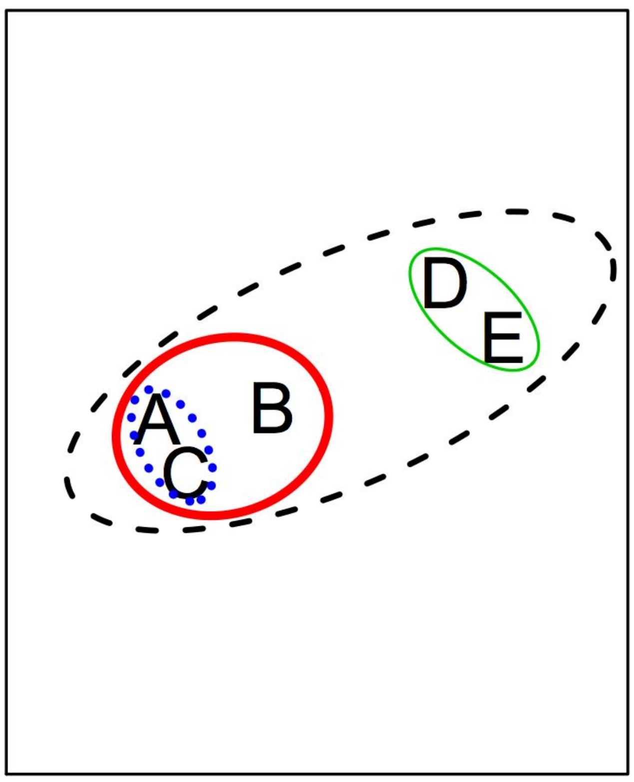
$$N^{1/p}$$

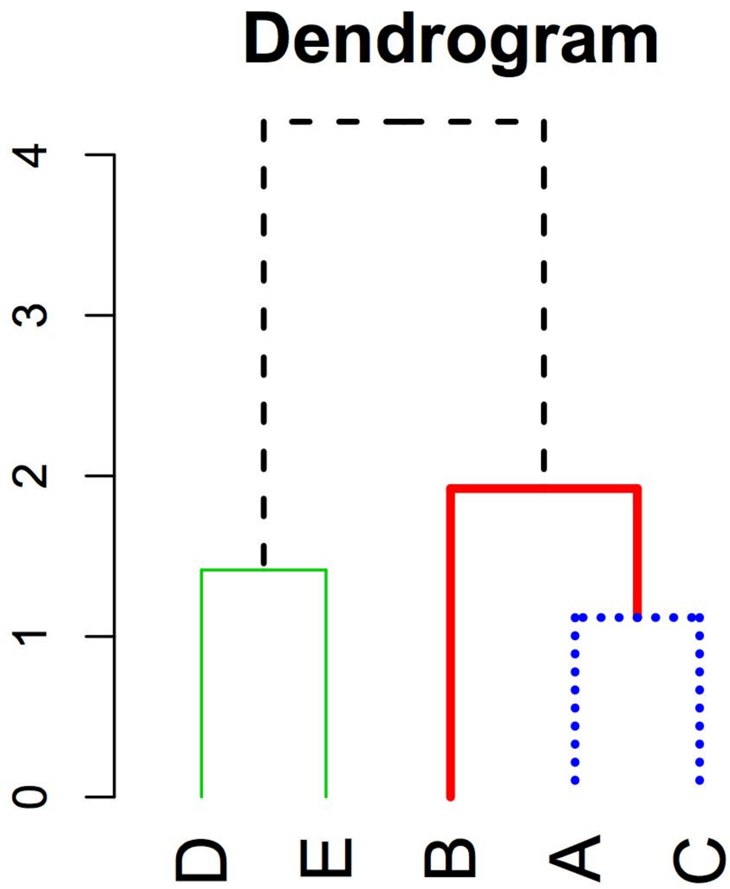
Hierarchical Clustering

Hierarchical Clustering



Hierarchical Clustering





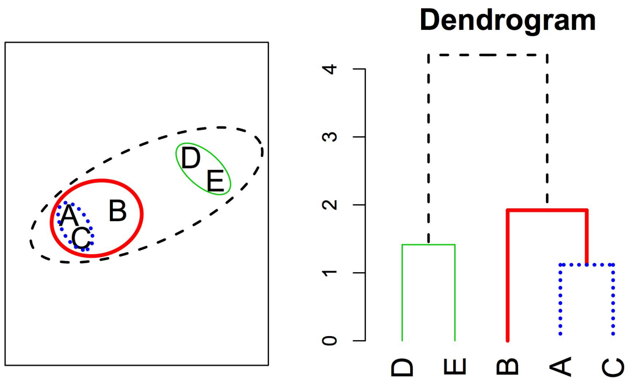
Hierarchical Clustering

<u>Algorithm</u>

(1) Each point as its own cluster

(2) Merge closest clusters

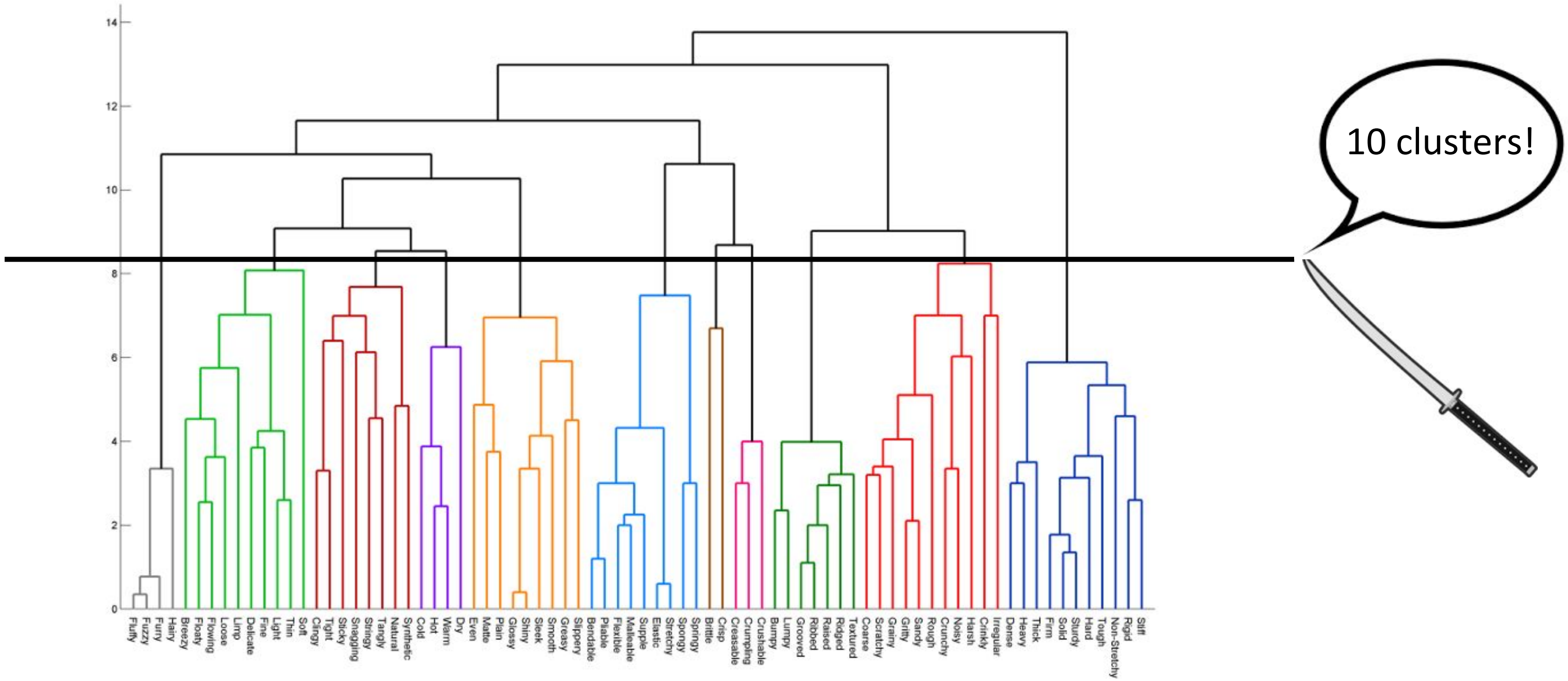
(3) End when all points in single cluster



Notice

- Skipped over the notion of “distance” between clusters
- Height of fusion tells you how close clusters are!
  - A and C are pretty close, at around 1.2
  - Red and Green are not that close, fusing at around 4.1

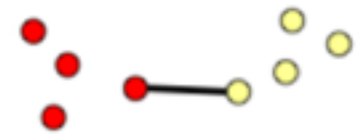
Varying K



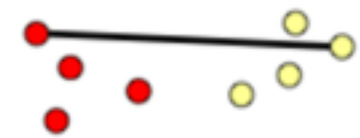
- In contrast to K-means, don’t have to choose K from the start!
  - Depending on where precisely we cut, we have anywhere from 1 to n clusters
- Choosing K: Can again use Elbow Method, Gap Statistic, Silhouette
  - But notice the heights give you sense of separation of clusters depending on cut.

Cluster distance measures

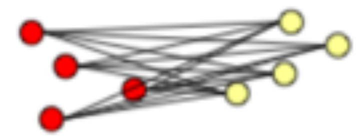
- Single link: $D(c_1, c_2) = \min_{x_1 \in c_1, x_2 \in c_2} D(x_1, x_2)$
  - distance between closest elements in clusters
  - produces long chains a→b→c→…→z



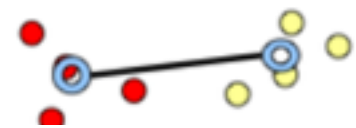
- Complete link: $D(c_1, c_2) = \max_{x_1 \in c_1, x_2 \in c_2} D(x_1, x_2)$
  - distance between farthest elements in clusters
  - forces "spherical" clusters with consistent "diameter"



- Average link: $D(c_1, c_2) = \sum_{x_1 \in c_1} \sum_{x_2 \in c_2} D(x_1, x_2)$
  - average of all pairwise distances
  - less affected by outliers

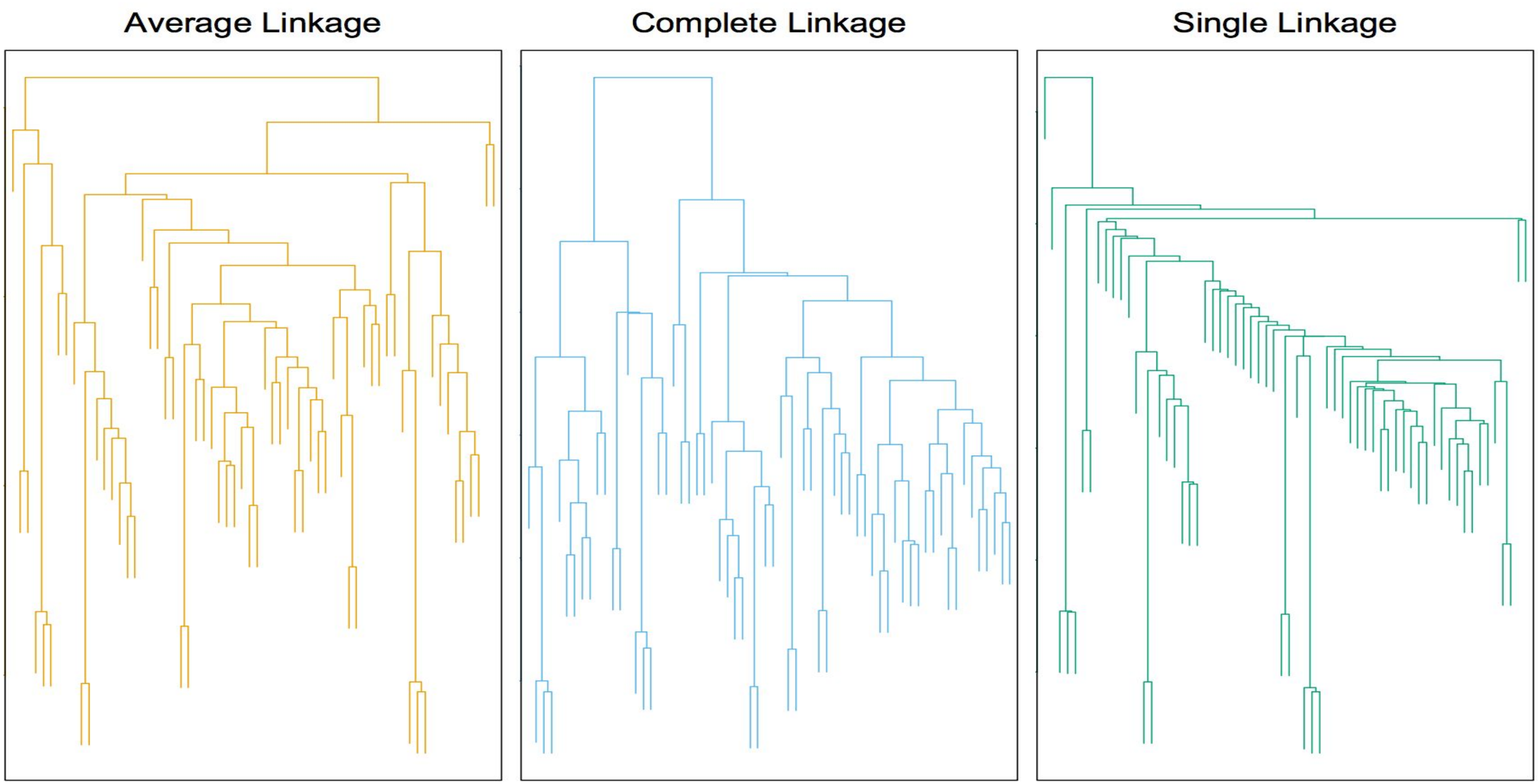


- Centroids: $D(c_1, c_2) = D\left(\left(\frac{1}{|c_1|} \sum_{x \in c_1} \vec{x}\right), \left(\frac{1}{|c_2|} \sum_{x \in c_2} \vec{x}\right)\right)$
  - distance between centroids (means) of two clusters



Most commonly used: Complete and Average

Distance between two clusters?



- Not too sensitive to outliers
- Compromise between complete linkage and single

- More sensitive to outliers
- May violate “closeness”

- Less sensitive to outliers
- Handles irregular shapes fairly naturally

Questions

- What is the curse of dimensionality?
  - Why is it particularly bad for kNN and clustering?
  - Pick either the radius or cube interpretation and provide a volume based explanation of the curse

- Describe the K-means algorithm in steps

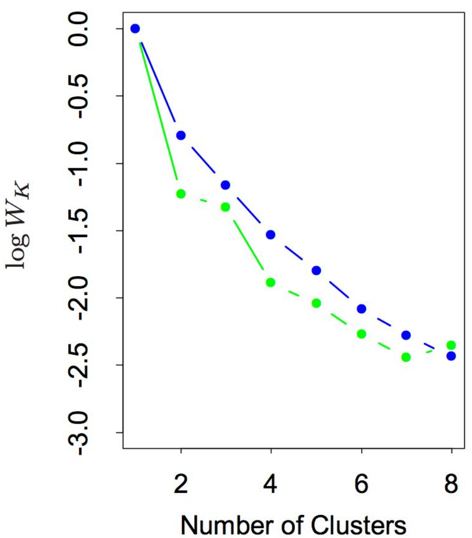
- Describe the Hierarchical clustering algorithm in steps
  - What is the height of the dendrogram?
  - Contrast with K-means

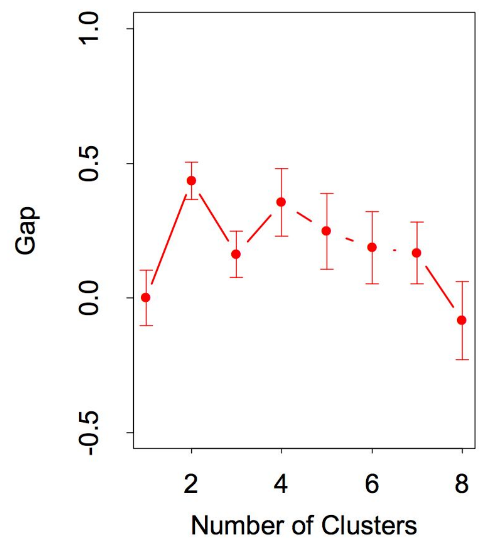
- Choosing K is no trivial task! What are ways of choosing K?
  - Describe Elbow method
  - Bonus (more advanced, can get away with just Elbow method): Describe GAP statistic, Silhouette Coefficient

Appendix

Choosing K – GAP Statistic

- Arguably best method!
- <u>Idea</u>: Compare within-cluster scatter $W_1, \ldots, W_k$ to uniformly distributed rectangle containing data. Find largest gap.
  - Notice as number of clusters increase, within cluster scatter decreases
  - What happens when number of clusters is number of points?





<u>Three Steps to the Gap Statistic</u>

(1) Observed vs. Expected value of $\log(W_k)$ over 20 simulations from uniform data

(2) Translate curves so that $\log(W_k) = 0$ for k=1

(3) Gap statistic K* is smallest K producing gap within one standard deviation of gap at K+1

Choosing K – Silhouette Coefficient

General method for interpreting and validating clusters of data

For each observation i:

- a(i) = average dissimilarity of i with all other data points within same cluster
  - A measure of how well i is assigned to the cluster
  - The smaller a(i) is, the better the assignment
- b(i) = lowest average dissimilarity of i to any other cluster, of which i is not member.
  - Other cluster can be thought of as a "neighboring cluster"

silhouette(i) = [b(i) – a(i)] / max{a(i), b(i)}

-1 < silhouete(i) < 1

Want a(i) small, b(i) large $\longrightarrow$ Want silhouette large

- near 1, dense and well separated
- near 0, overlapping clusters; could well belong to another cluster
- near -1, misclustered

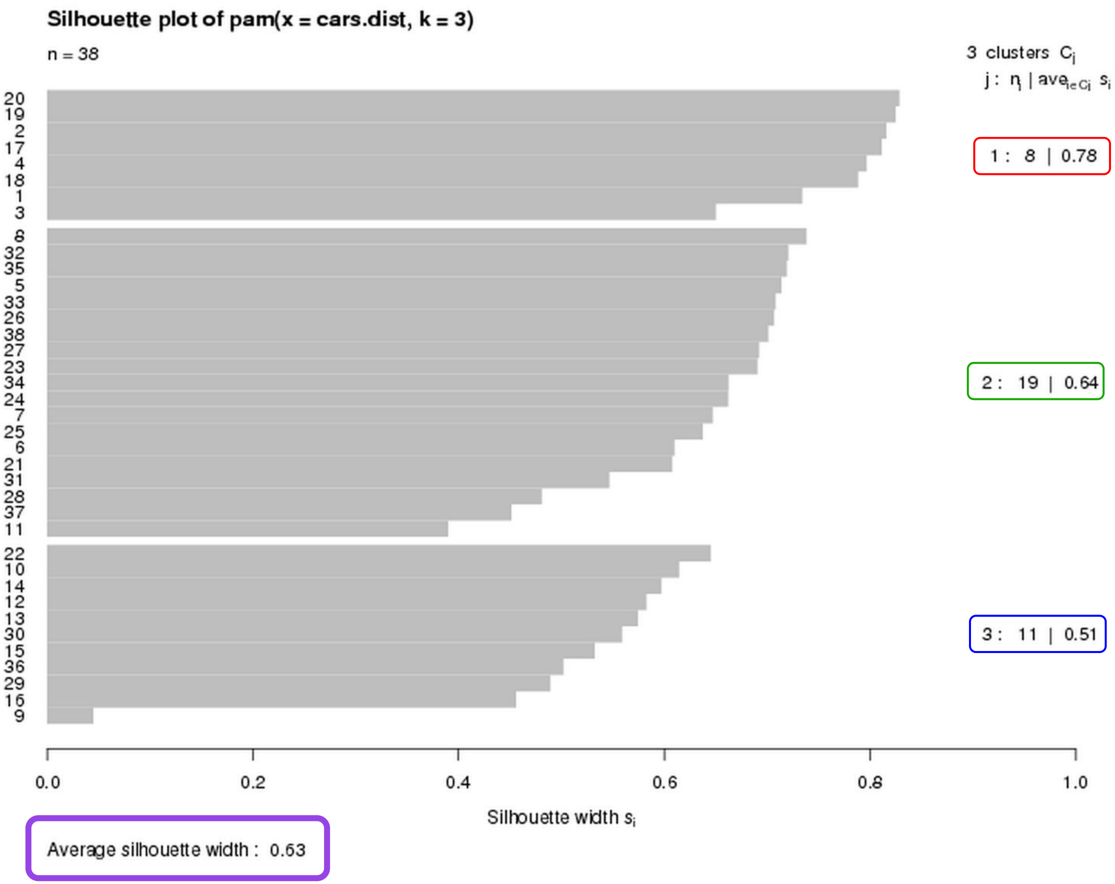
Silhouette Coefficient

silhouette(i) = [b(i) – a(i)] / max{a(i), b(i)}

-1 < silhouete(i) < 1

Want a(i) small, b(i) large → Want silhouette large

- near 1, dense and well separated
- near 0, overlapping clusters; could well belong to another cluster
- near -1, misclustered



38 data points

3 clusters

- 1st cluster has 8 data points and average silhouette of 0.78
- 2nd cluster has 19 points, 0.64
- 3rd cluster has 11 points, 0.51
- Overall average silhouette **0.63**

<u>Guidelines for Overall Avg Silhouette</u>

Range	Interpretation
0.71 – 1.0	Strong structure found
0.51 – 0.7	Reasonable structure
0.26 – 0.5	Structure weak/artificial
< 0.25	No substantial structure

Some Additional Considerations

- Standardize features?
  - Yes, probably.
  - How to deal with categorical?
- Outliers can be problematic
  - Especially using squared Euclidean as a distance metric
  - What if small subset of observations quite different from all others?
    - Kmeans and hierarchical clustering FORCES every data-point into clusters, potentially distorting clusters
    - Mixture models (‘soft clustering’) are attractive alternative as they accommodate outliers
- Generally not very robust
  - Can test by clustering subsets of data

K-means – a few more notes

Simple, elegant method, but can be problematic in a lot of ways

- Only intended for **quantitative** features (think centroid calculation for categorical data) and squared **Euclidean** distance (which is not robust to outliers)

One alternative is K-medoids

- Worth reading up a bit more about http://web.stanford.edu/~hastie/local.ftp/Springer/OLD/ESLII_print4.pdf page 515
- Computationally more intensive (requires large proximity matrix computation)
- But, handles categorical features more naturally (though still must define distance metric for mixed data rather carefully), and more robust to outliers.

Within Cluster Point Scatter

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} d(x_i, x_{i'})$$

Within Cluster Point Scatter
A natural loss function is the sum pairwise distances of the points within each cluster, summed over all clusters. In particular, we could specify d(xi, xi') to be Euclidean

---

Let $d_{ii'} = d(x_i, x_{i'})$

$$T = \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} d_{ii'} = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \left(\sum_{C(i')=k} d_{ii'} + \sum_{C(i')\neq k} d_{ii'} \right)$$

Total Point Scatter

$$T = W(C) + B(C)$$

$$B(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')\neq k} d_{ii'}$$

Between Cluster Point Scatter

Within Cluster Point Scatter

It can be shown that

$$
\begin{aligned}
W(C) &= \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} ||x_i - x_{i'}||^2 \\
&= \sum_{k=1}^{K} N_k \sum_{C(i)=k} ||x_i - \bar{x}_k||^2,
\end{aligned}
$$

where

$\bar{x}_k = (\bar{x}_{1k}, \ldots, \bar{x}_{pk})$ is mean vector associated with k-th cluster

$N_k = \sum_{i=1}^{N} I(C(i) = k)$