Clustering

Overview

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

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

Unsupervised Learning

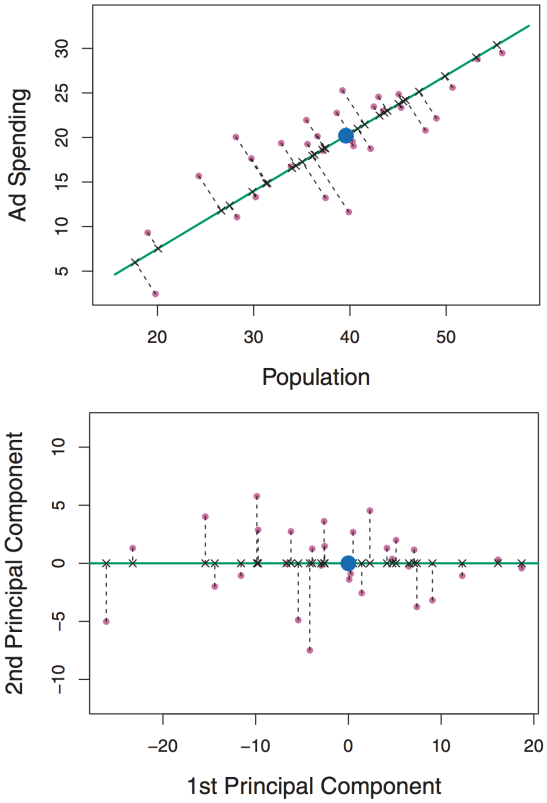
- No response variable, y
  - Just based on predictors, X_1, X_2, X_3, …, X_p
- A fuzzy endeavor...
  - Not cross-validating to choose best "model" in usual sense
  - Not cross-validating to know how well you're doing
- Can be useful as
  - ✓ preprocessing step for supervised learning
  - ✓ better understand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

Unsupervised Learning

Two most common and contrasting unsupervised techniques

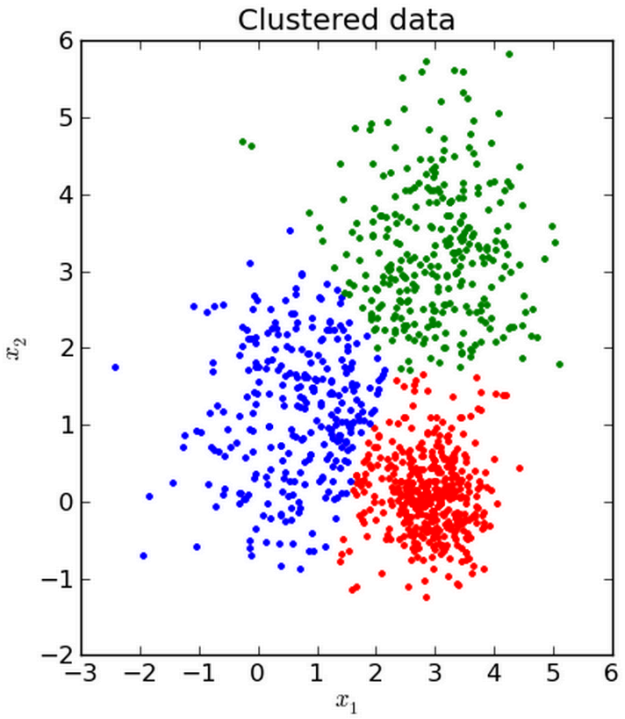
PCA

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

Curse of Dimensionality

First let's take a detour and re-visit Linear Regression and k-th Nearest Neighbor

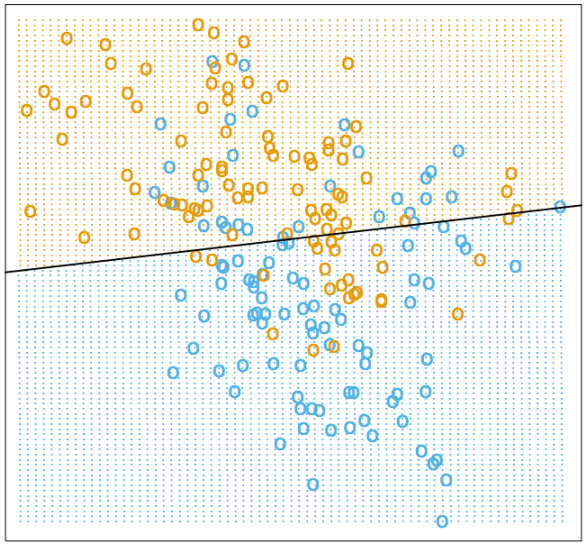
Linear Models

- Very structured
- Stable but possibly inaccurate
- Low Variance, High Bias

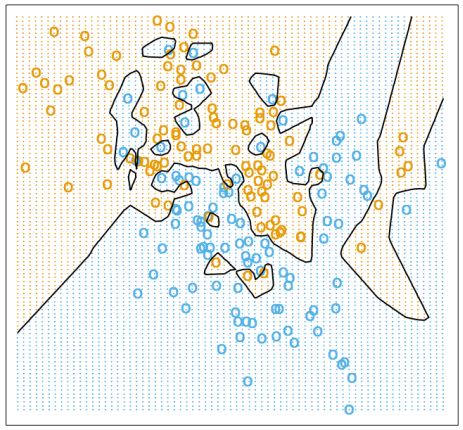
k-th Nearest Neighbor

- Very mildly structural
- Often accurate, but unstable
- High Variance, Low Bias

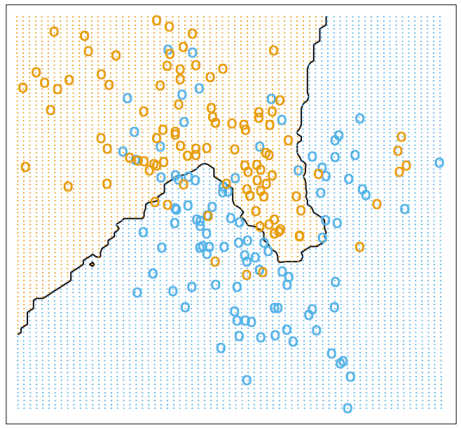
Linear Regression with 0/1 Response



1-Nearest Neighbor



15-Nearest Neighbor

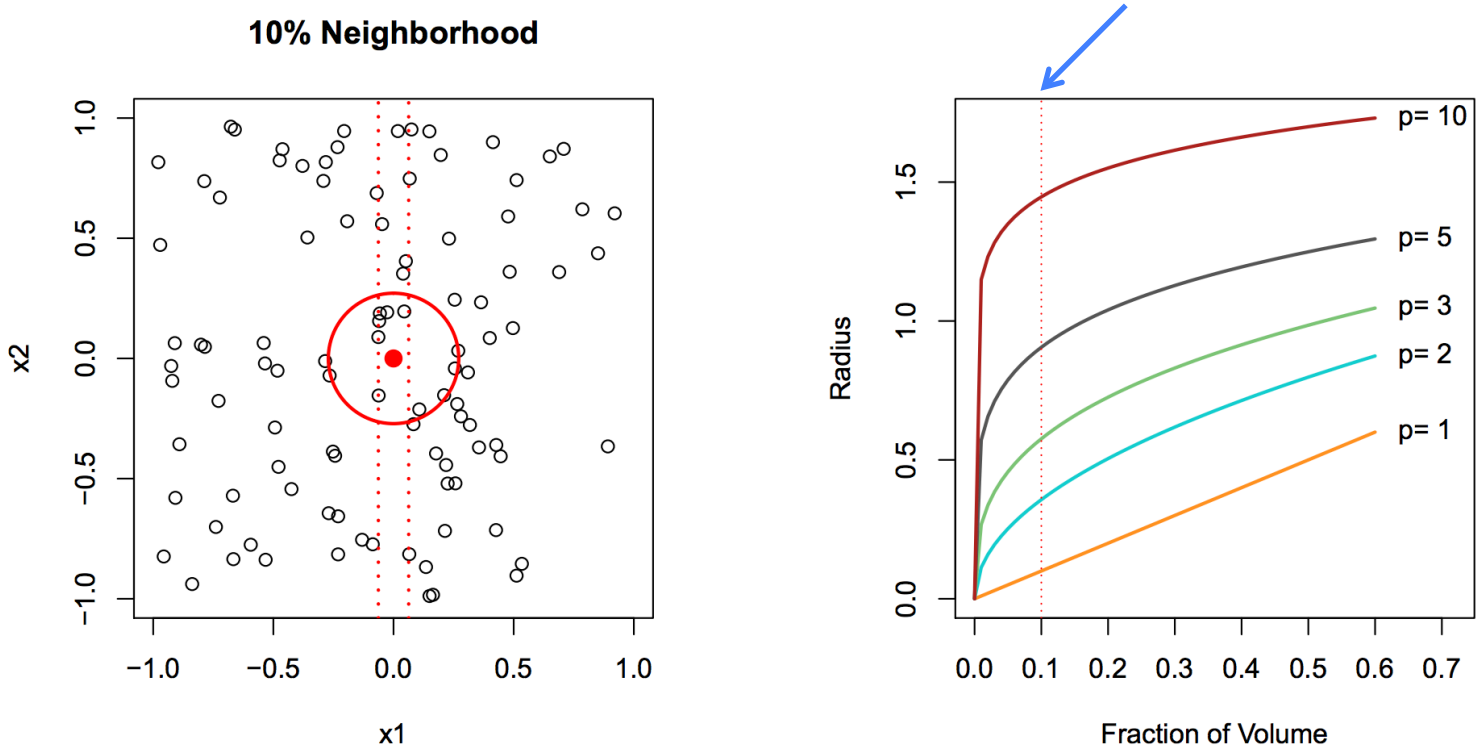


Curse of Dimensionality

- kNN is problematic in high-dim spaces
  - Though can be pretty good for $p \leq 4$ and N on the large side
- Nearest neighbors can be "far" in high dimensions
- Need to get a reasonable fraction of the N values of y_i to average to bring down the variance

Okay….what's "far"?
To start, let's consider 10% to be a reasonable fraction

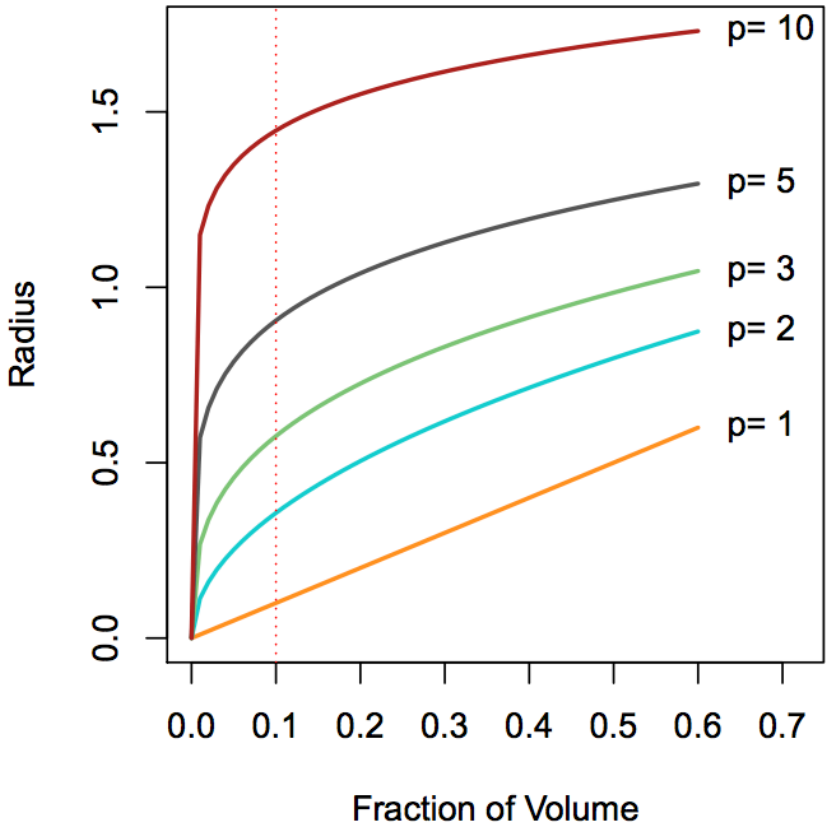
Curse of Dimensionality



- 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

Can you work out some of the points on the plot?



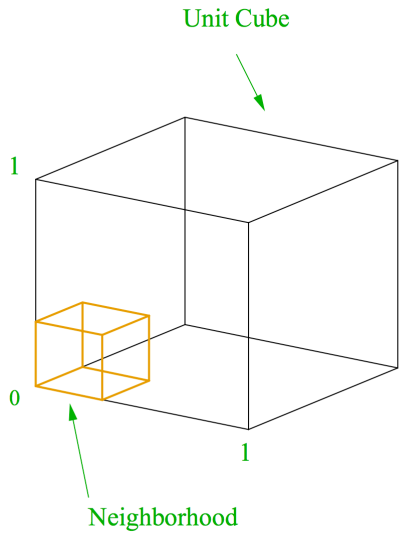
Dimension	Volume of a ball of radius R	Radius of a ball of volume V
0	1	All balls have volume 1
1	$2R$	$V/2$
2	πR^2	$\frac{V^{1/2}}{\sqrt{\pi}}$
3	$\frac{4}{3}\pi R^3$	$\left(\frac{3V}{4\pi}\right)^{1/3}$
4	$\frac{\pi^2}{2}R^4$	$\frac{(2V)^{1/4}}{\sqrt{\pi}}$
5	$\frac{8\pi^2}{15}R^5$	$\left(\frac{15V}{8\pi^2}\right)^{1/5}$
6	$\frac{\pi^3}{6}R^6$	$\frac{(6V)^{1/6}}{\sqrt{\pi}}$
7	$\frac{16\pi^3}{105}R^7$	$\left(\frac{105V}{16\pi^3}\right)^{1/7}$
8	$\frac{\pi^4}{24}R^8$	$\frac{(24V)^{1/8}}{\sqrt{\pi}}$
9	$\frac{32\pi^4}{945}R^9$	$\left(\frac{945V}{32\pi^4}\right)^{1/9}$
10	$\frac{\pi^5}{120}R^{10}$	$\frac{(120V)^{1/10}}{\sqrt{\pi}}$

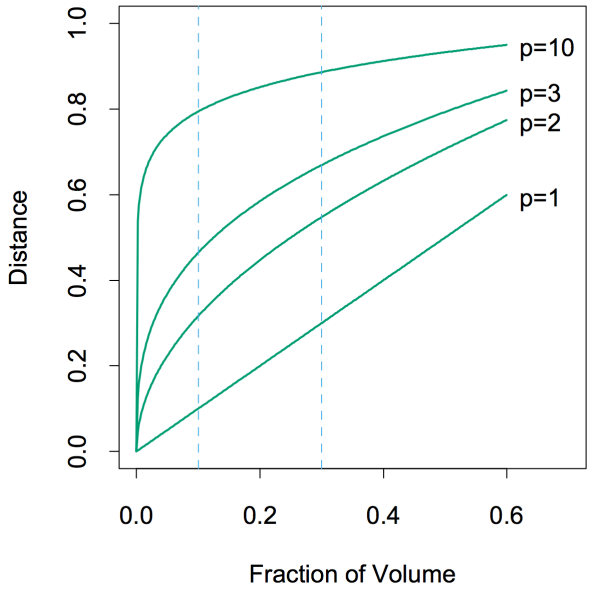
Curse of Dimensionality

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:

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

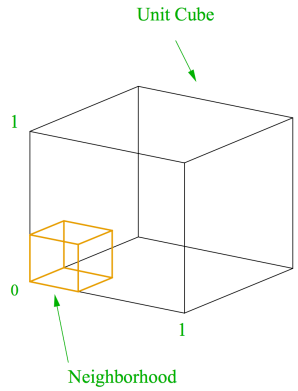


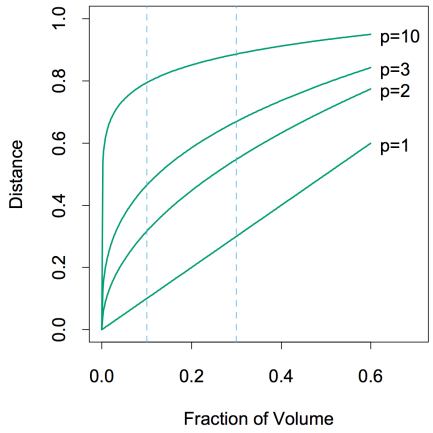


Can you work out out the 10% neighborhood for the unit cube case?

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

Curse of Dimensionality





Expected edge length

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

Sampling density proportional to

$$N^{1/p}$$

p is dimensions of input space
N is number of points

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

p = 1 → edge = (0.1)^1 = 0.1
p = 10 → edge = (0.1)^(1/10) = 0.794

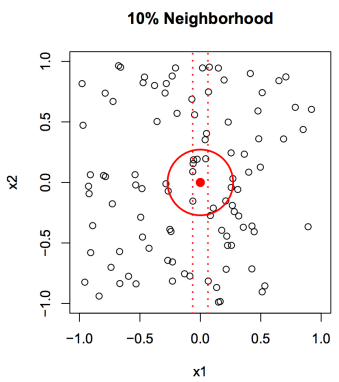
Sampling density example: How to achieve equivalent density in higher dimensions

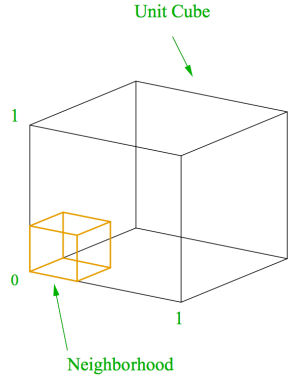
If $N_1 = 100$ represents dense sample for a single dim feature space
To achieve same density for 10 inputs, we need $N_{10} = 100^{10}$ points

Curse of Dimensionality - Takeaways

- kNN, or any method involving this sort of distancing, suffers majorly from curse of dimensionality
  - Nearest neighbors "far" in high dimensions (even for p = 10)
  - As we'll see, k-means and hierarchical clustering fall prey to curse.

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





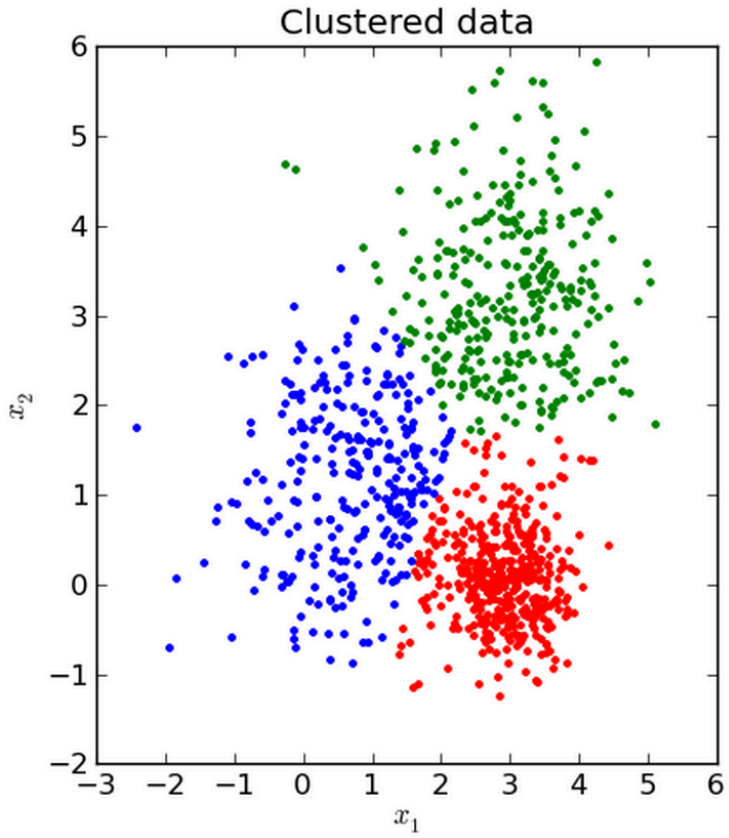
- It takes **a lot** of data to make up for increase in dimensions

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

Sampling density $\propto$ to $N^{1/p}$

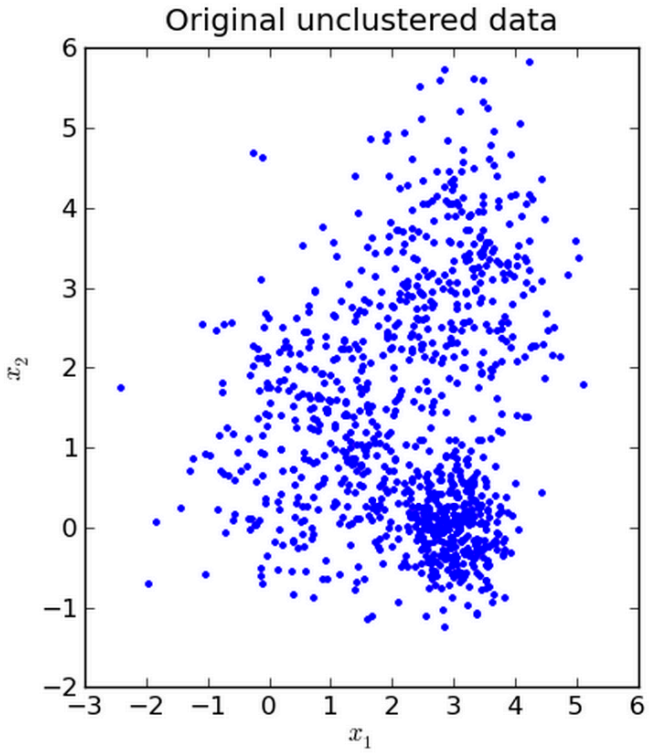
What is clustering?

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



What is clustering?

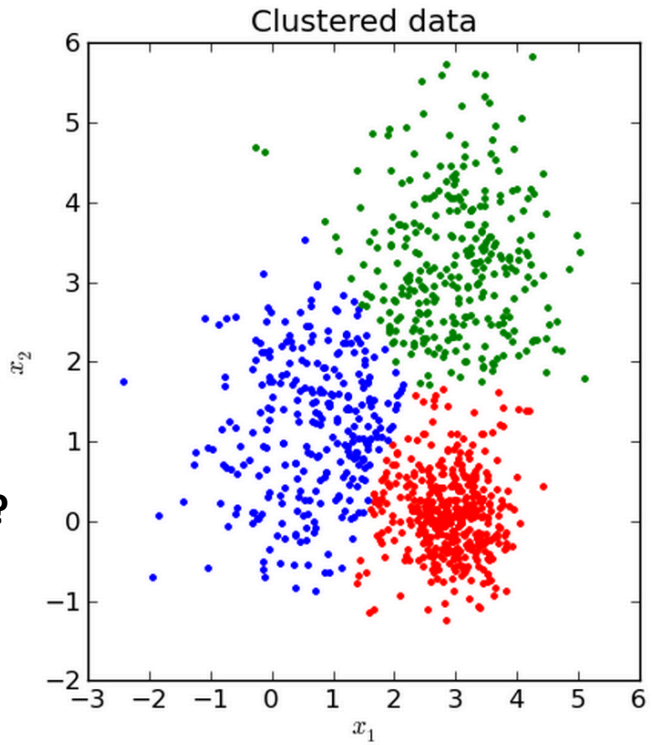
Divide data into distinct subgroups such that observations within each group are quite similar



How many subgroups?

What's considered similar?

How am I even doing this?



Two most popular approaches

K-means

Hierarchical clustering

We'll go over

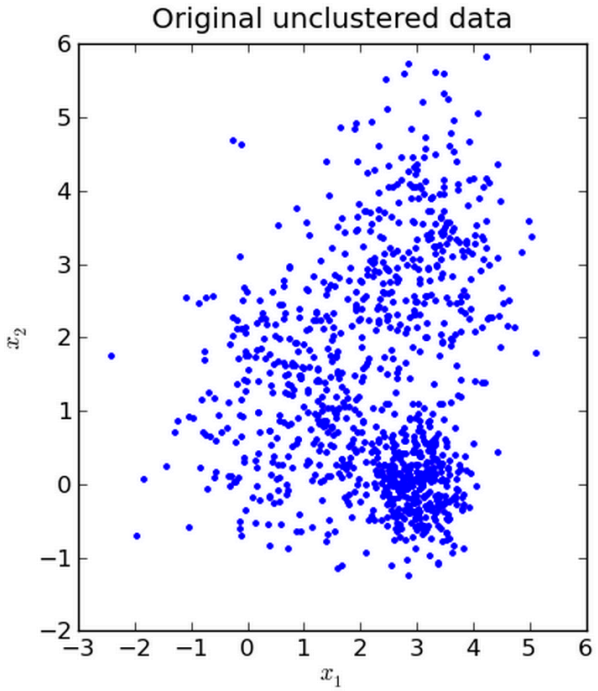
- Algorithm
- Choosing K
- Special considerations

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….

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

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

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

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

K-means

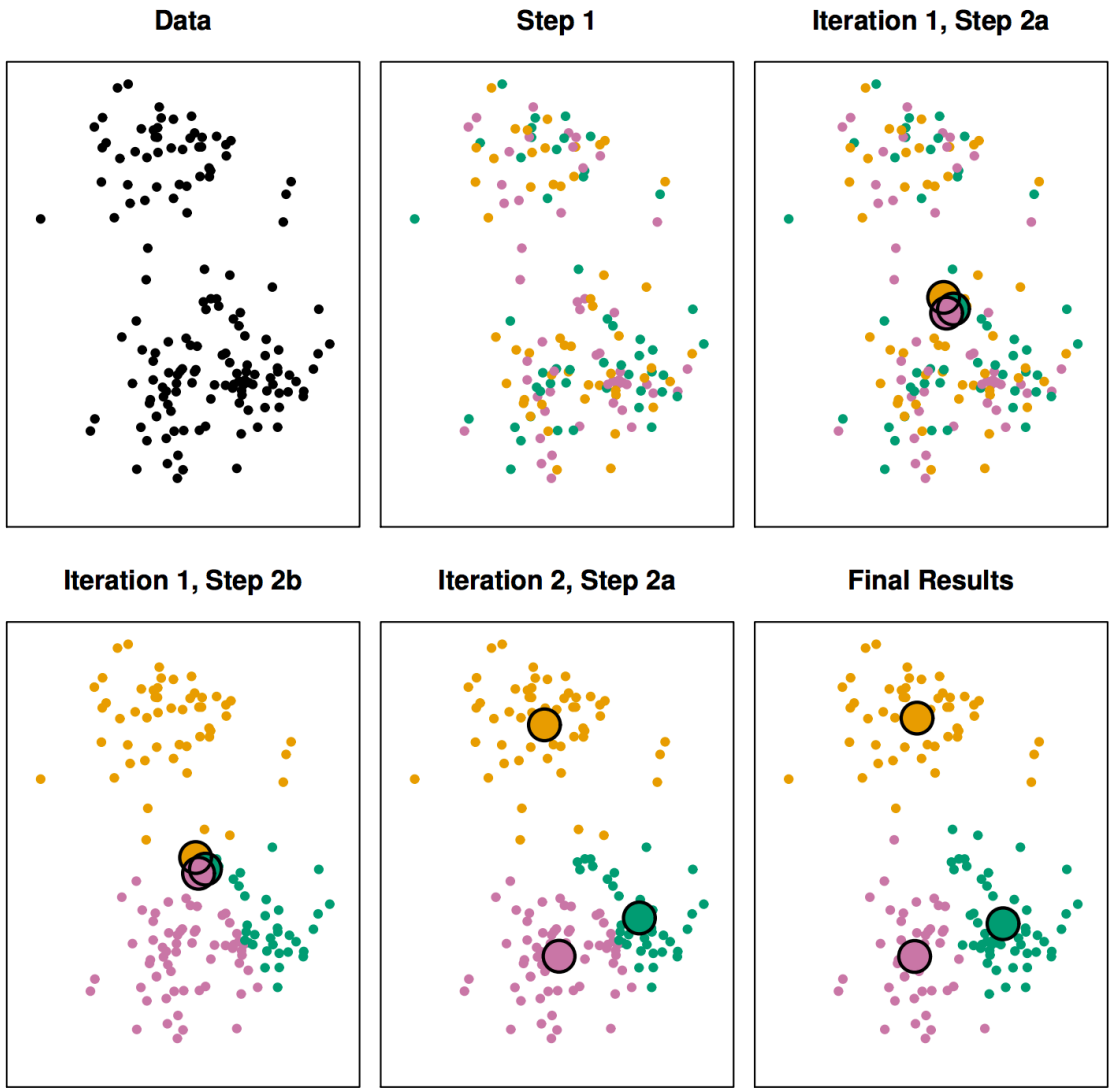
Altogether, we're picking C_1, … C_K such that

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

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

K-means algorithm

For K=3…



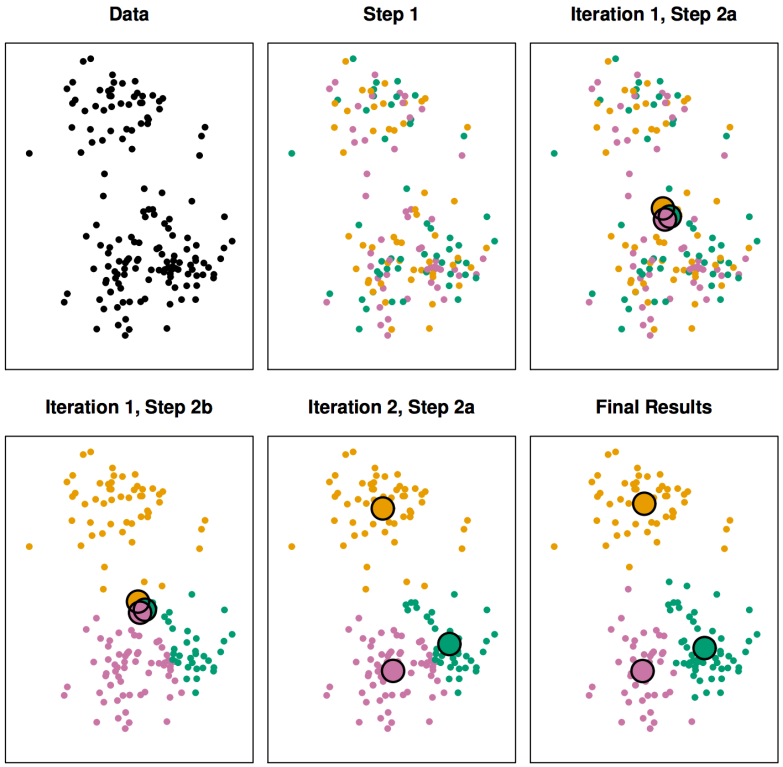
K-means algorithm

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

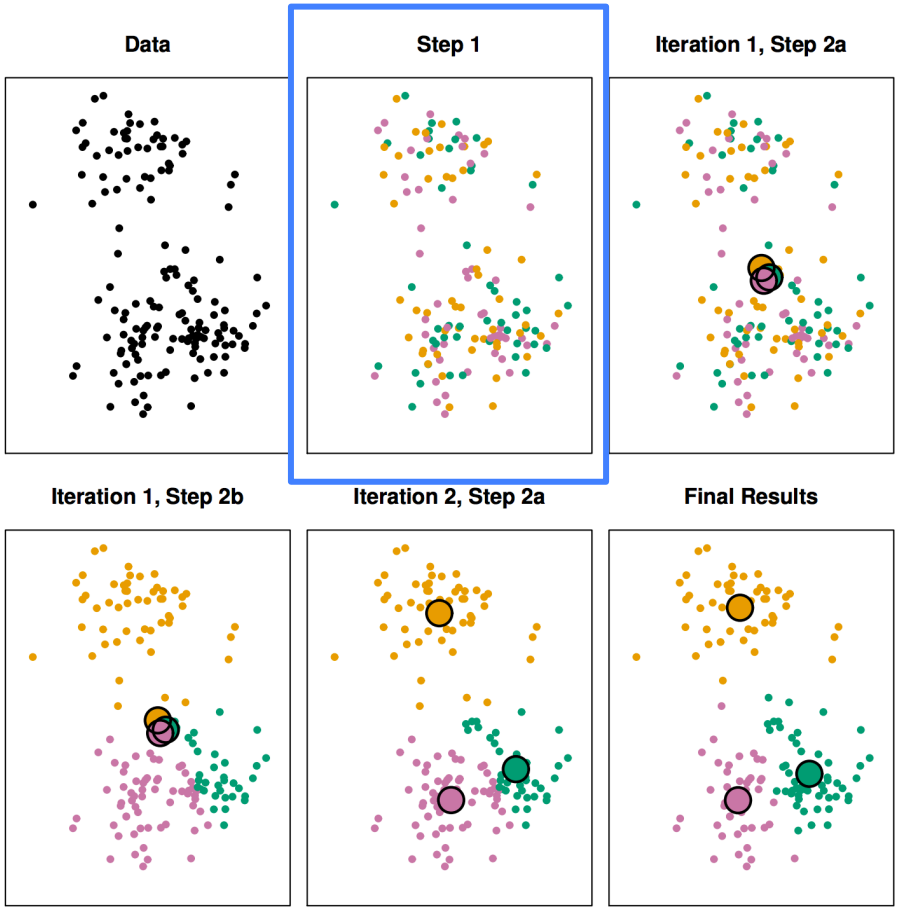
(2) Repeat until cluster assignments stop changing

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)



K-means algorithm



Finds local optimum!

Results depend on random initialization

Solution

Try multiple initializations and pick one with lowest

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

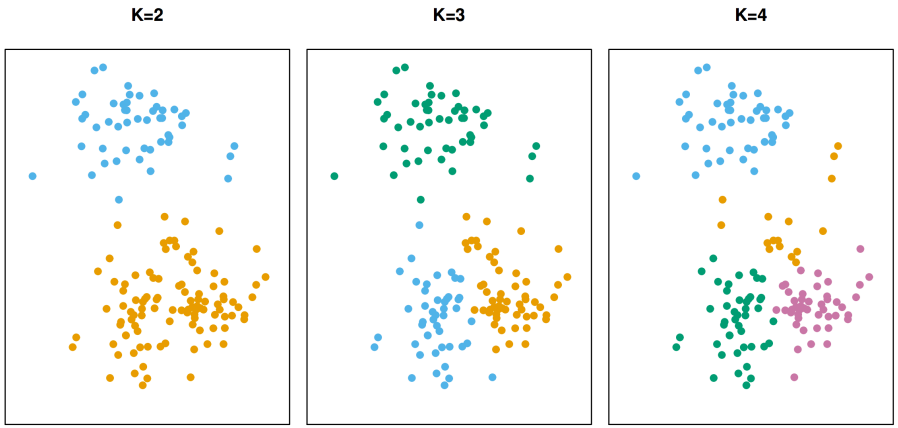
* Also could consider smarter initializations such as kmeans++ http://en.wikipedia. org/wiki/K-means%2B%2B

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
- We'll go over three popular ones
  - "Elbow" method
  - GAP statistic
  - Silhouette Coefficient

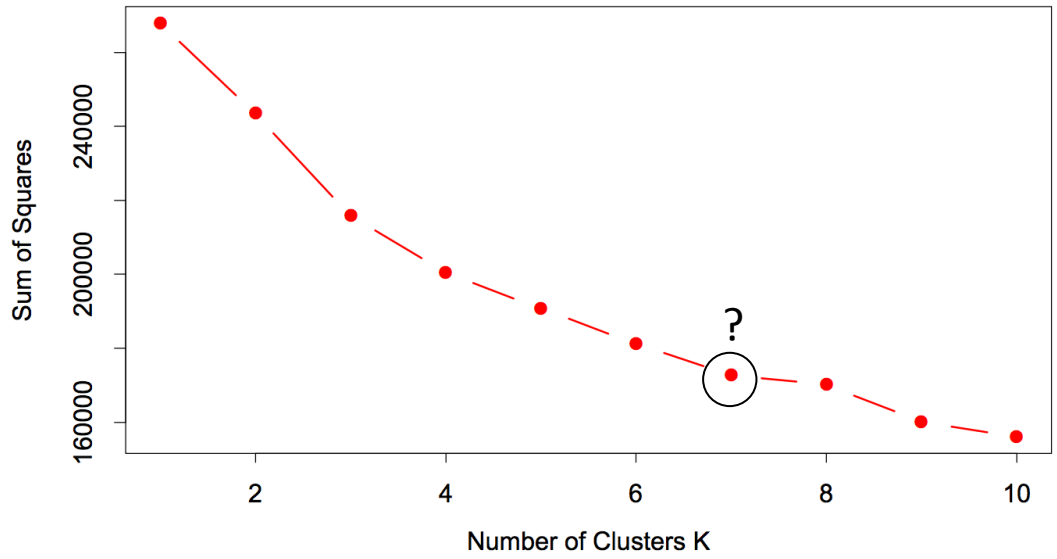


Choosing K – "Elbow" method

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

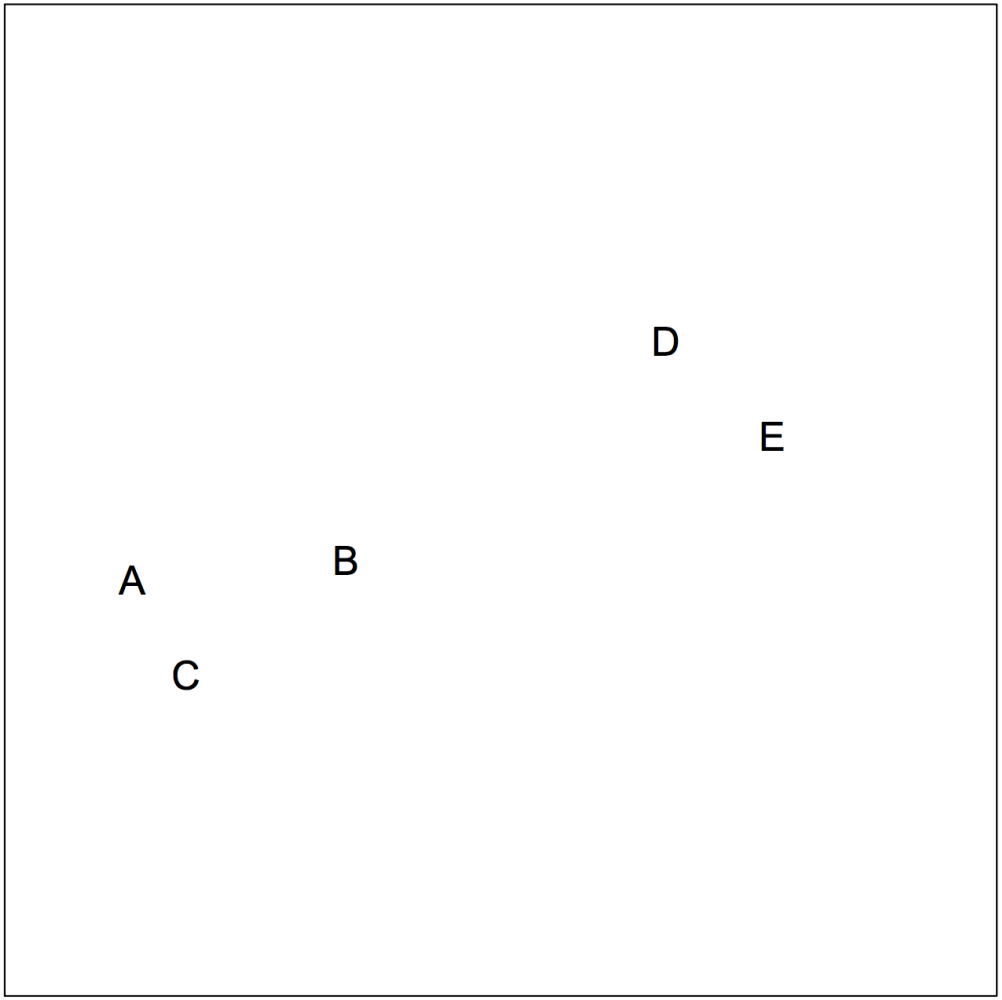
$$W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')=k}||x_i - x_{i'}||^2$$

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

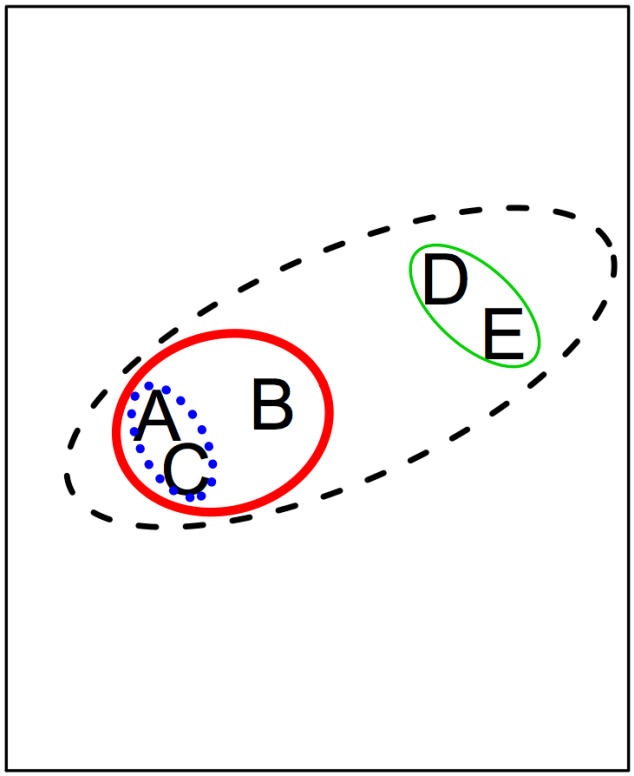


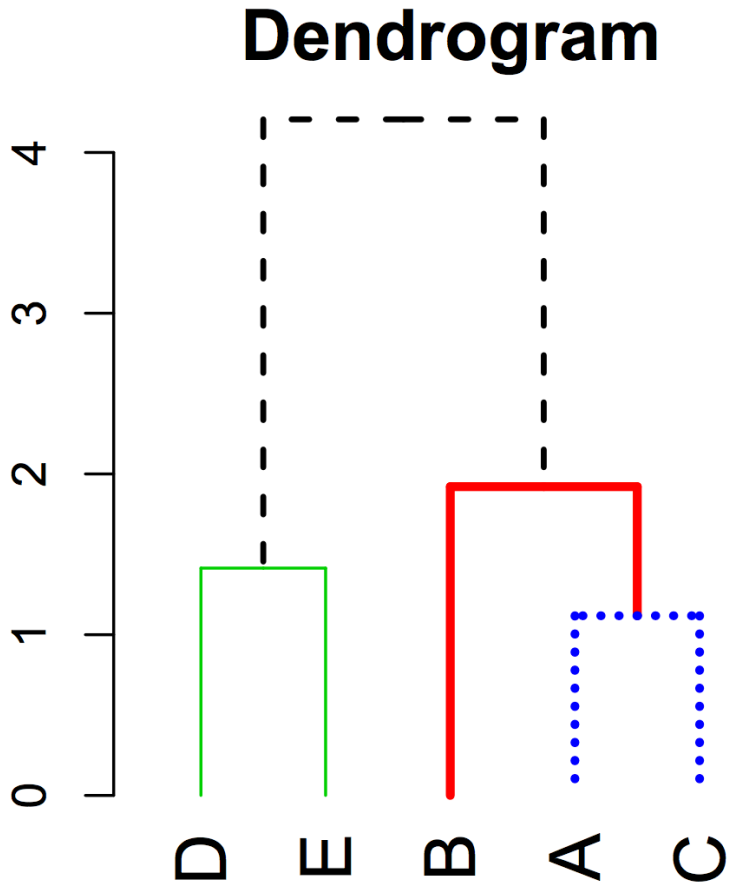
Hierarchical Clustering

Hierarchical Clustering



Hierarchical Clustering





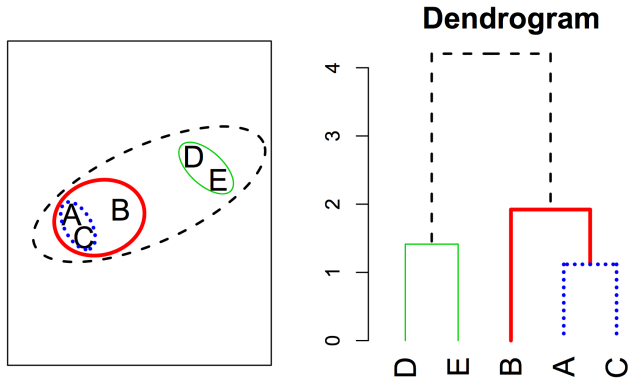
Hierarchical Clustering

Algorithm

(1) Each point as its own cluster

(2) Merge closest clusters

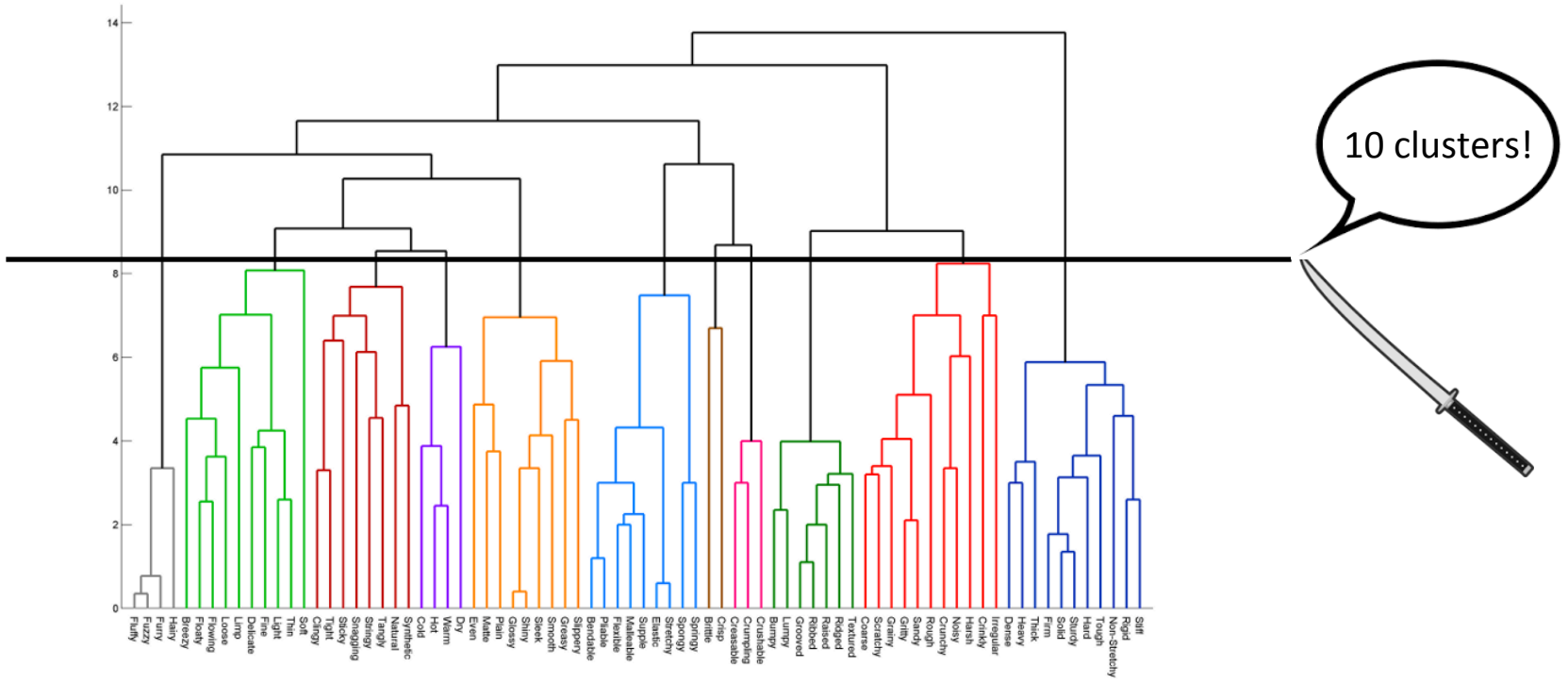
(3) End when all points in single cluster



Notice

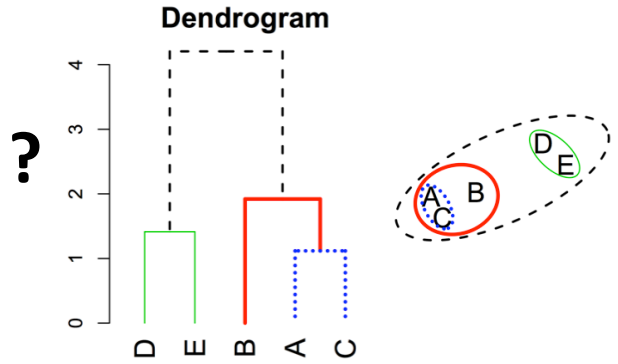
- Skipped over the notion of “distance” between clusters
- Height of fusion tells you how close clusters are!
  - A and C are pretty close, at around 1.2
  - Red and Green are not that close, fusing at around 4.1

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.

Distance between two clusters?



☺ Tends to be balanced → (more commonly used)

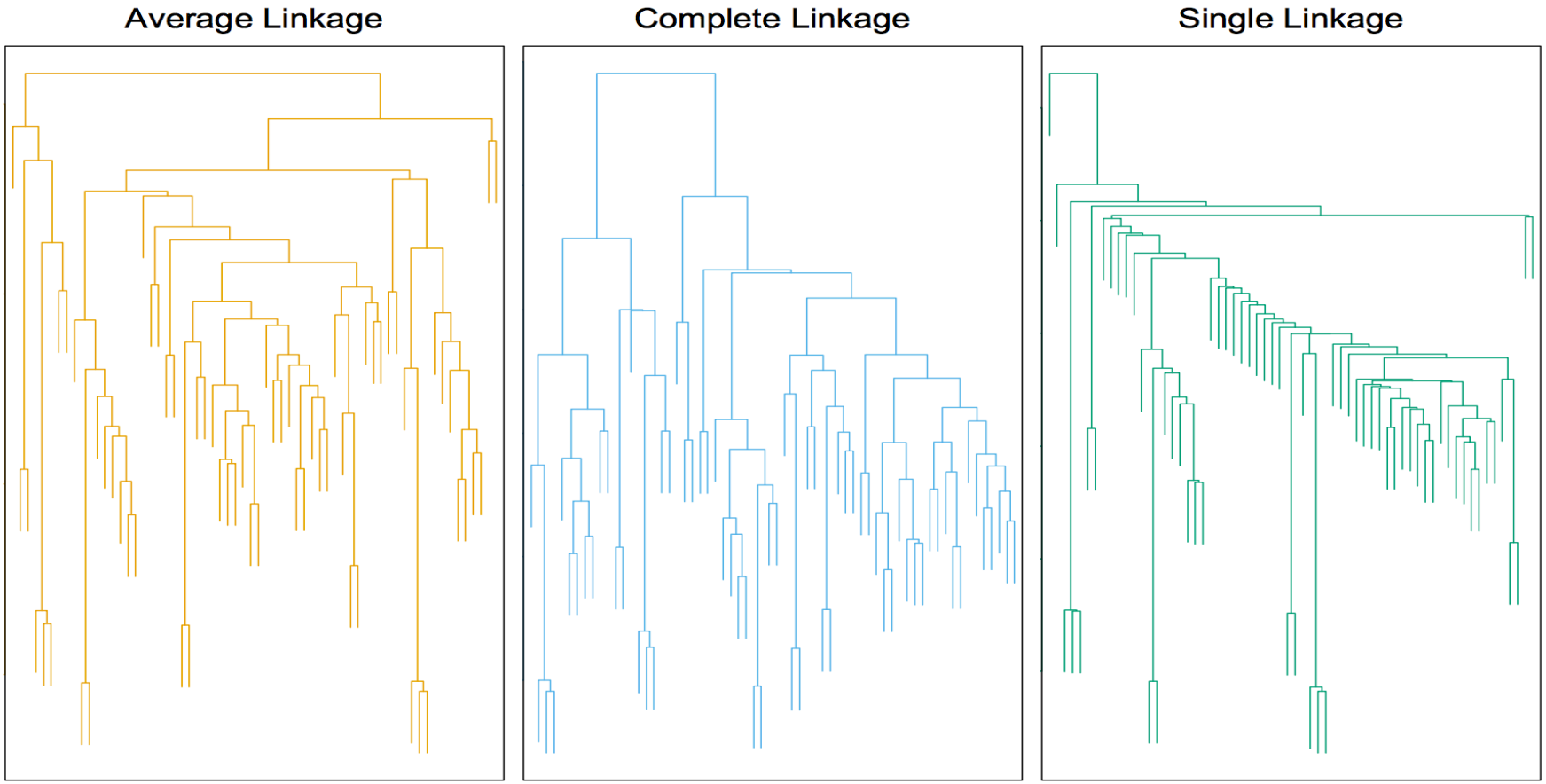
Extended trailing clusters → (less commonly used)

☺ Tends to be balanced → (more commonly used)

(not as commonly used, though popular in Genomics)

Linkage	*Description*
Complete	Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
Single	Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time.
Average	Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable *inversions*.

Distance between two clusters?



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

- Less sensitive to outliers
- May violate to “closeness”

- More sensitive to outliers
- Handles irregular shapes fairly naturally

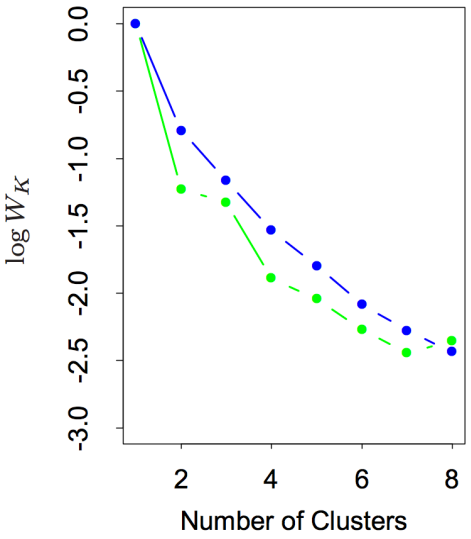
Questions

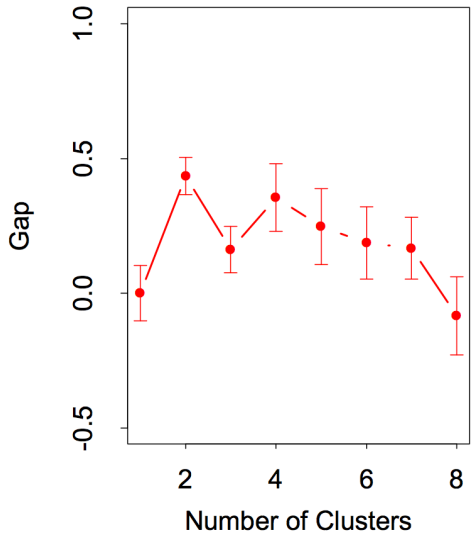
- What is the curse of dimensionality?
  - Why is it particularly bad for kNN and clustering?
  - Pick either the radius or cube interpretation and provide a volume based explanation of the curse
- Describe the K-means algorithm in steps
- Describe the Hierarchical clustering algorithm in steps
  - What 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!
- Idea: 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?





Three Steps to the Gap Statistic

(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 → Want silhouette large

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

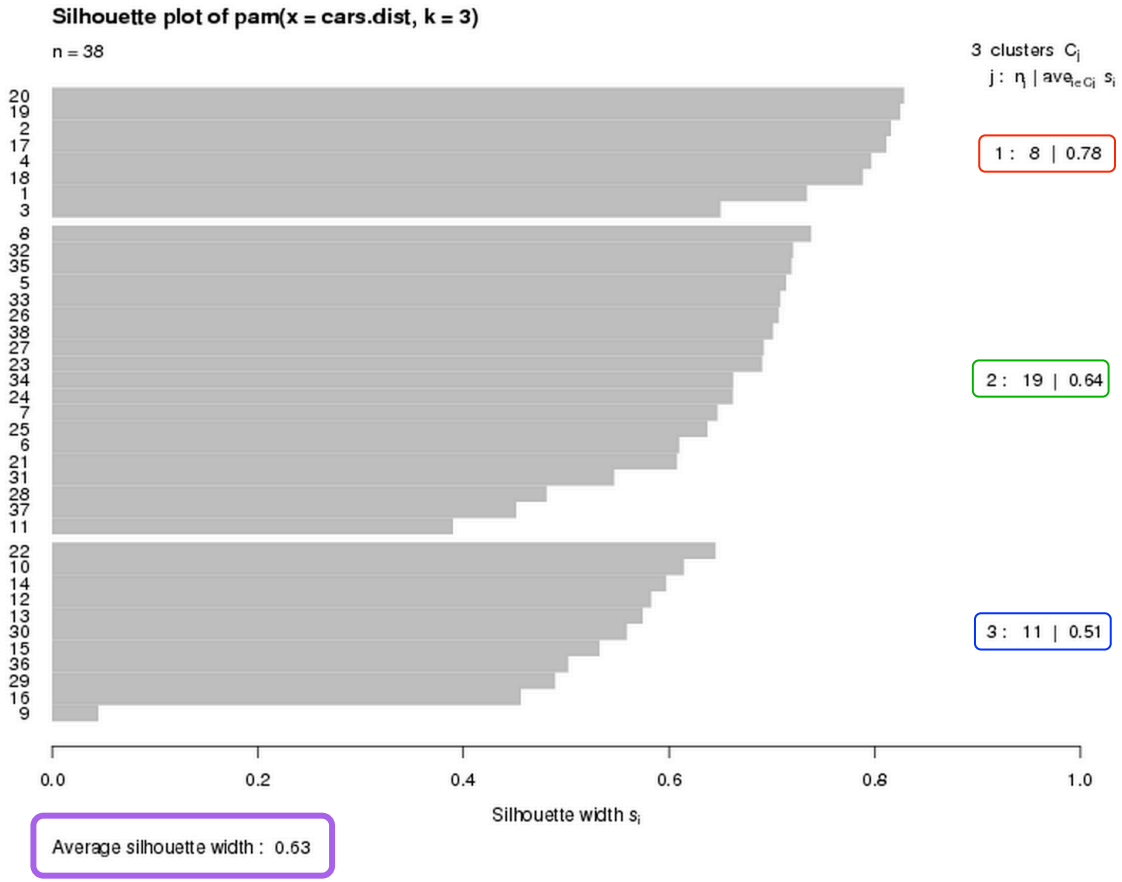
Silhouette Coefficient

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**

Guidelines for Overall Avg Silhouette

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

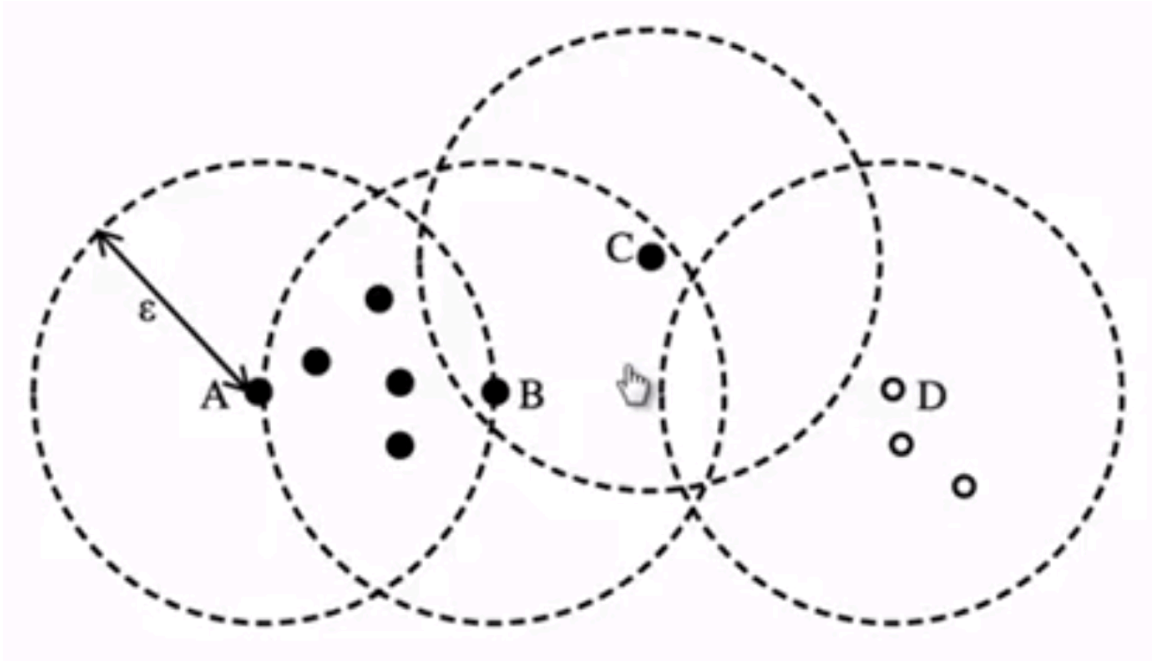
- Only intended for **quantitative** features (think centroid calculation for categorical data) and squared **Euclidean** distance (which is not robust to outliers)

One alternative is **K-medoids**

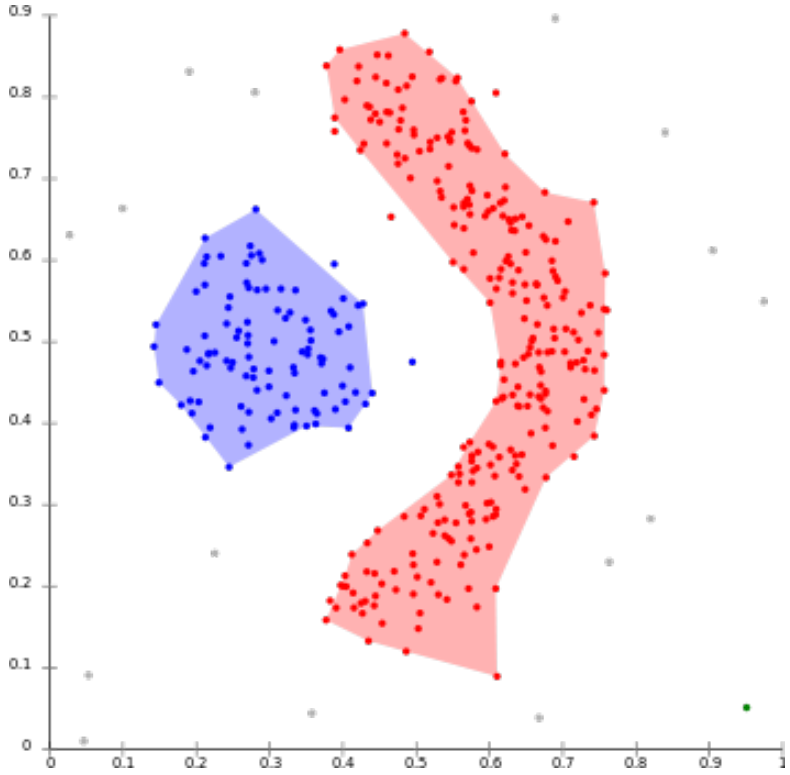
- Minimizes pairwise dissimilarities and chooses one of data points as center, or "medoid"
  - Worth reading up a bit more about http://web.stanford.edu/~hastie/local.ftp/Springer/OLD/ESLII_print4.pdf page 515
- Computationally more intensive (large proximity matrix computation)
- But, handles categorical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

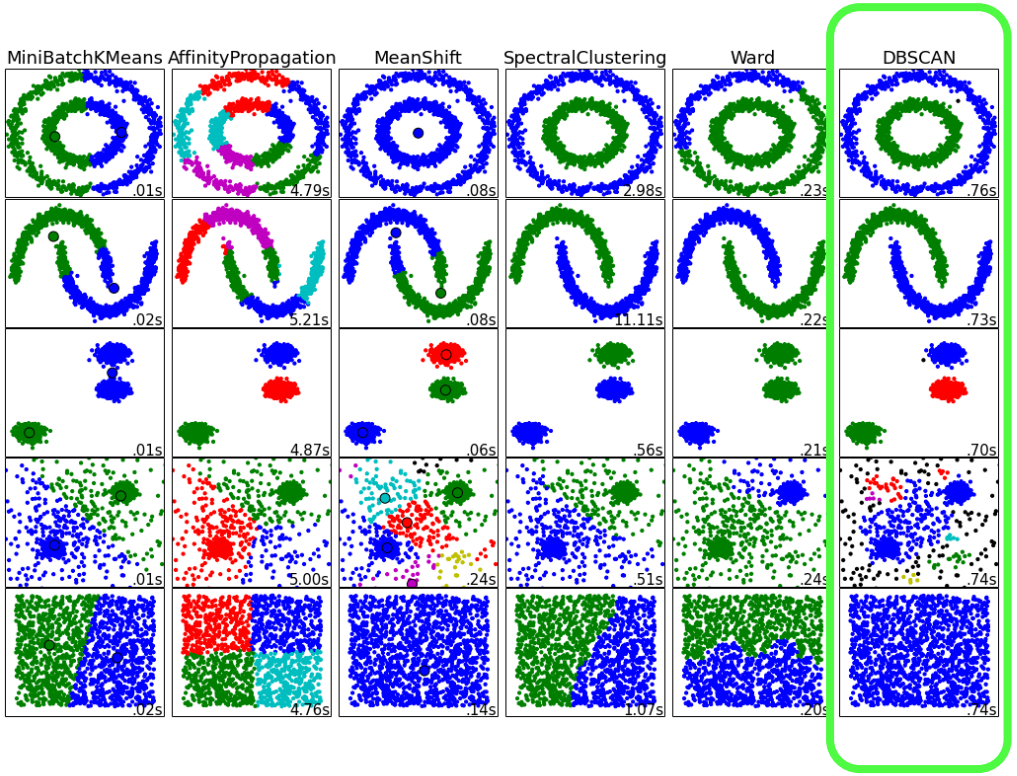
DBSCAN

- A very popular clustering algorithm as well!
- “Density-based spatial clustering of applications with noise”
- Groups together close together points, marks low density regions as outliers



DBSCAN


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


MiniBatchKMeans
AffinityPropagation
MeanShift
SpectralClustering
Ward
DBSCAN
.01s
4.79s
.08s
2.98s
.23s
.76s
.02s
5.21s
.08s
11.11s
.22s
.73s
.01s
4.87s
.06s
.56s
.21s
.70s
.01s
5.00s
.24s
.51s
.24s
.74s
.02s
4.76s
.14s
1.07s
.20s
.74s

Suppose you want to cluster similar shoppers, to show items and ads that they'll like …

Shopper	Computers	Keyboards	Peanut Butter	Oreos
Aditi	1	2	0	0
Rohit	0	0	30	50
Aaron	0	0	50	50
Jia	0	0	0	1
Jack	2	4	10	20
William	3	6	0	0
….	…	…	…	…

- Who is Aditi most "similar" to in Euclidean sense?
- Who is Jack most "similar" to?
  - Do we care more about selling a jar of Peanut Butter or a Computer?
- What can we do so that distance isn't just based on Peanut Butter and Oreos?
  - But William is still far from Aditi…

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

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