Data Clustering

Héctor Corrada Bravo

University of Maryland, College Park, USA

DATA606: 2020-04-12

Center for Bioinformatics & Computational Biology
University of Maryland

Motivating Example

Time series dataset of mortgage affordability as calculated and distributed by Zillow: https://www.zillow.com/research/data/.

The dataset consists of monthly mortgage affordability values for 76 counties with data from 1979 to 2017.

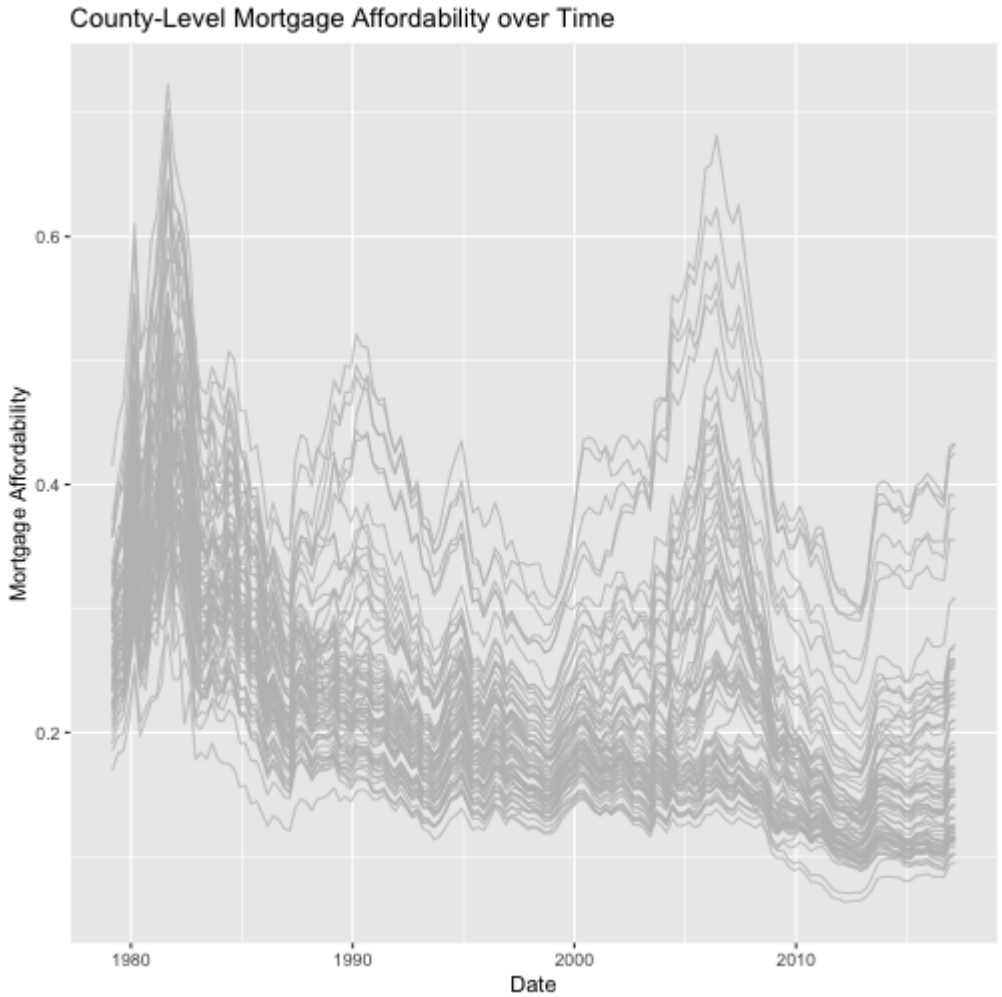
Motivating Example

> "To calculate mortgage affordability, we first calculate the mortgage payment for the median-valued home in a metropolitan area by using the metro-level Zillow Home Value Index for a given quarter and the 30-year fixed mortgage interest rate during that time period, provided by the Freddie Mac Primary Mortgage Market Survey (based on a 20 percent down payment)."

Motivating Example

> "Then, we consider what portion of the monthly median household income (U.S. Census) goes toward this monthly mortgage payment. Median household income is available with a lag. "

Motivating Example



Can we partition counties into groups of counties with similar value trends across time?

Cluster Analysis

The high-level goal of cluster analysis is to organize objects (observations) that are *similar* to each other into groups.

We want objects within a group to be more *similar* to each other than objects in different groups.

Cluster Analysis

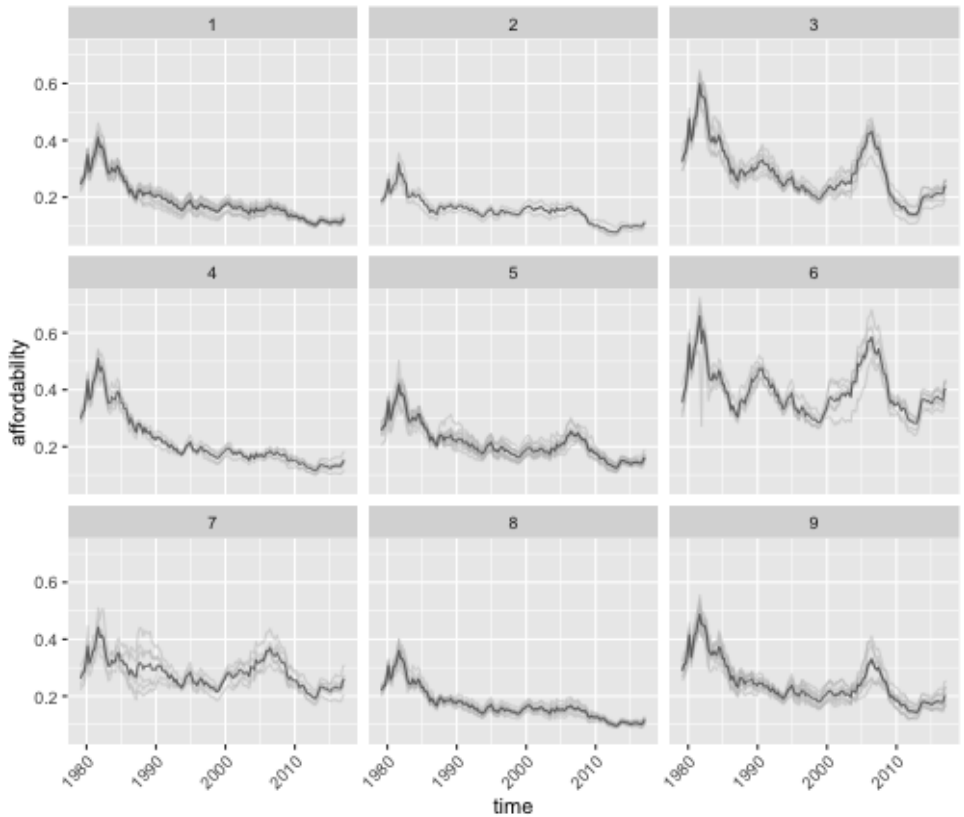
The high-level goal of cluster analysis is to organize objects (observations) that are *similar* to each other into groups.

We want objects within a group to be more *similar* to each other than objects in different groups.

Central to this high-level goal is how to measure the degree of *similarity* between objects.

A clustering method then uses the *similarity* measure provided to it to group objects into clusters.

Cluster Analysis



Result of the k-means algorithm partitioning the data into 9 clusters.

The darker series within each cluster shows the average time series within the cluster.

Dissimilarity-based Clustering

For certain algorithms, instead of similarity we work with dissimilarity, often represented as distances.

When we have observations defined over attributes, or predictors, we define dissimilarity based on these attributes.

Dissimilarity-based Clustering

Given measurements x_{ij} for $i = 1, \ldots, N$ observations over $j = 1, \ldots, p$ predictors.

Suppose we define a dissimilarity $d_j(x_{ij}, x_{i'j})$, we can then define a dissimilarity between objects as

$$d(x_i, x_{i'}) = \sum_{j=1}^{p} d_j(x_{ij}, x_{i'j})$$

Dissimilarity-based Clustering

In the k-means algorithm, and many other algorithms, the most common usage is squared distance

$$d_j(x_{ij}, x_{i'j}) = (x_{ij} - x_{i'j})^2$$

We can use different dissimilarities, for example

$$d_j(x_{ij}, x_{i'j}) = |x_{ij} - x_{i'j}|$$

which may affect our choice of clustering algorithm later on.

Dissimilarity-based Clustering

For categorical variables, we could set

$$d_j(x_{ij}, x_{i'j}) = \begin{cases} 0 \text{ if } x_{ij} = x_{i'j} \\ \\ 1 \text{ o.w.} \end{cases}$$

Dissimilarity-based Clustering

If the values the categorical variable have an intrinsic similarity

Generalize using symmetric matrix L with elements

$L_{rr'} = L_{r'r}$,
$L_{rr} = 0$ and
$L_{rr'} \geq 0$ otherwise.

This may of course lead to a dissimilarity that is not a proper distance.

Limitations of Distance-based clustering

When working with distances, we are quickly confronted with the *curse of dimensionality*.

One flavor: in high dimensions: all points are equi-distant

The curse of dimensionality

Consider the case where we have many covariates. We want to use a distance-based clustering method.

The curse of dimensionality

Consider the case where we have many covariates. We want to use a distance-based clustering method.

Basically, we need to define distance and look for small multi-dimensional "balls" around the data points. With many covariates this becomes difficult.

The curse of dimensionality

Imagine we have equally spaced data and that each covariate is in $[0, 1]$. We want something that clusters points into *local* regions, containg some reasonably small amount of data points (say 10%). Let's imagine these are high-dimensional cubes.

The curse of dimensionality

Imagine we have equally spaced data and that each covariate is in $[0, 1]$. We want something that clusters points into *local* regions, containg some reasonably small amount of data points (say 10%). Let's imagine these are high-dimensional cubes.

If we have p covariates and we are forming p-dimensional cubes, then each side of the cube must have size l determined by $l \times l \times \cdots \times l = l^p = .10$.

The curse of dimensionality

If the number of covariates is p=10, then $l = .1^{1/10} = .8$. So it really isn't local! If we reduce the percent of data we consider to 1%, $l = 0.63$. Still not very local.

The curse of dimensionality

If the number of covariates is p=10, then $l = .1^{1/10} = .8$. So it really isn't local! If we reduce the percent of data we consider to 1%, $l = 0.63$. Still not very local.

If we keep reducing the size of the neighborhoods we will end up with very small number of data points in each cluster and require a large number of clusters.

K-means Clustering

A commonly used algorithm to perform clustering is the K-means algorithm.

It is appropriate when using squared Euclidean distance as the measure of object dissimilarity.

$$d(x_i, x_{i'}) = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

$$= \left\| x_i - x_{i'} \right\|^2$$

K-means Clustering

K-means partitions observations into K clusters, with K provided as a parameter.

Given some clustering, or partition, C, denote cluster assignment of observation x_i to cluster $k \in \{1, \ldots, K\}$ is denoted as $C(i) = k$.

K-means Clustering

K-means partitions observations into K clusters, with K provided as a parameter.

Given some clustering, or partition, C, denote cluster assignment of observation x_i to cluster $k \in \{1, \ldots, K\}$ is denoted as $C(i) = k$.

K-means minimizes this clustering criterion:

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

K-means Clustering

This is equivalent to minimizing

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} N_k \sum_{i:\, C(i)=k} \|x_i - \bar{x}_k\|^2$$

with:

- $\bar{x}_k = (\bar{x}_{k1}, \ldots, \bar{x}_{kp})$
- $\bar{x}_{kj}$ is the average of predictor j over the observations assigned to cluster k,
- N_k is the number of observations assigned to cluster k

K-means Clustering

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} N_k \sum_{i:\, C(i)=k} \|x_i - \bar{x}_k\|^2$$

Minimize the total distance given by each observation to the mean (centroid) of the cluster to which the observation is assigned.

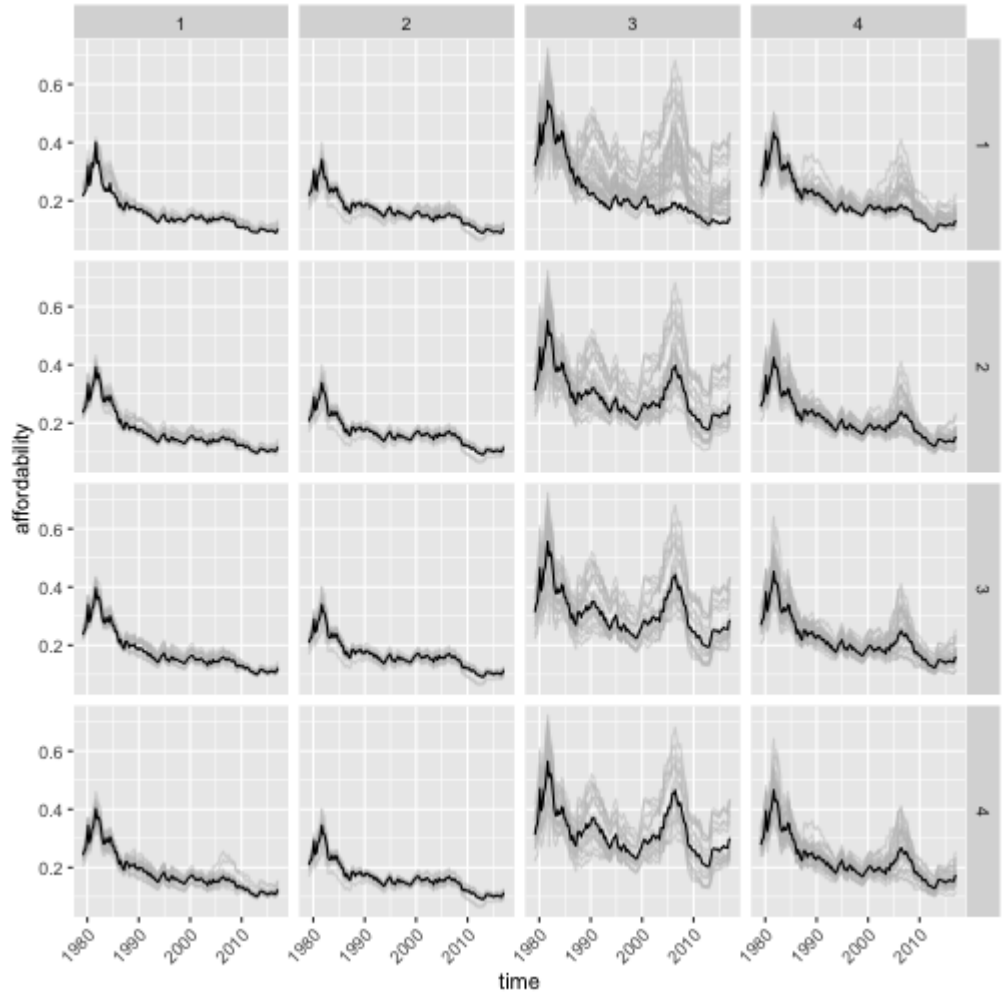
K-means Clustering

An iterative algorithm is used to minimize this criterion

1. Initialize by choosing K observations as centroids $m_1, m_2, \ldots, m_k$
2. Assign each observation i to the cluster with the nearest centroid, i.e., set $C(i) = \arg\min_{1 \le k \le K} \left\| x_i - m_k \right\|^2$
3. Update centroids $m_k = \bar{x}_k$
4. Iterate steps 2 and 3 until convergence

K-means Clustering

Here we illustrate the k-means algorithm over four iterations on our example data with $K = 4$.



K-means Clustering

Criterion $W(C)$ is reduced in each iteration so the algorithm is assured to converge.

Not a convex criterion, the clustering we obtain may not be globally optimal.

In practice, the algorithm is run with multiple initializations (step 0) and the best clustering achieved is used.

K-means Clustering

Also, selection of observations as centroids can be improved using the K-means++ algorithm:

1. Choose an observation as centroid m_1 uniformly at random
2. To choose centroid m_k, compute for each observation i not chosen as a centroid the distance to the nearest centroid
   $d_i = \min_{1 \leq l < k} \left\| x_i - m_l \right\|^2$
3. Set centroid m_k to an observation randomly chosen with probability
   $\frac{e_i^d}{\sum_{i'} e_{i'}^d}$
4. Iterate steps 1 and 2 until K centroids are chosen

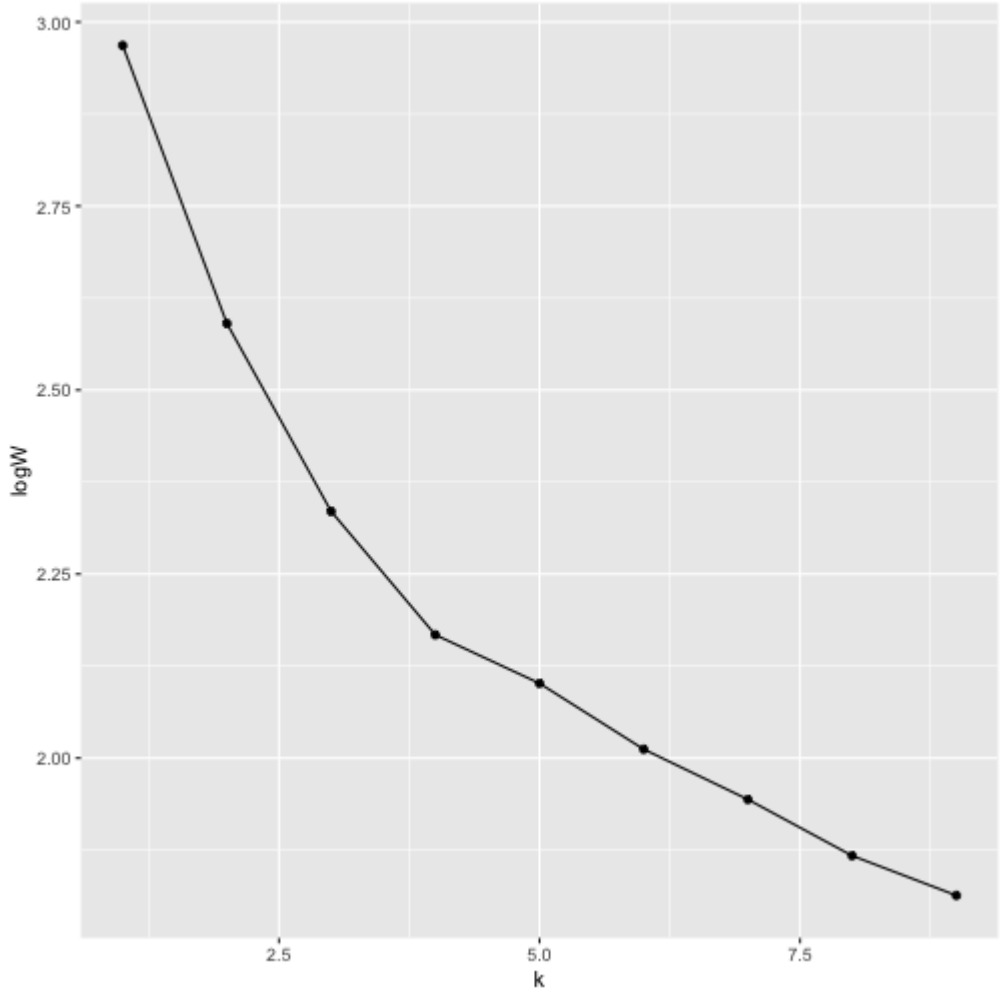
Choosing the number of clusters

The number of parameters must be determined before running the K-means algorithm.

There is no clean direct method for choosing the number of clusters to use in the K-means algorithm (e.g. no cross-validation method)

Choosing the number of clusters

Looking at criterion $W(C)$ alone is not sufficient as the criterion will become smaller as the value of K is reduced.



Choosing the number of clusters

We can use properties of this plot for ad-hoc selection.

Suppose there is a true underlying number K^* of clusters in the data,

- improvement in the $W_K(C)$ statistic will be fast for values of $K \leq K^*$
- slower for values of $K > K^*$.

Choosing the number of clusters

Improvement in the $W_K(C)$ statistic will be fast for values of $K \leq K^*$

In this case, there will be a cluster which will contain observations belonging to two of the true underlying clusters, and therefore will have poor within cluster similarity.

As K is increased, observations may then be separated into separate clusters, providing a sharp improvement in the $W_K(C)$ statistic.

Choosing the number of clusters

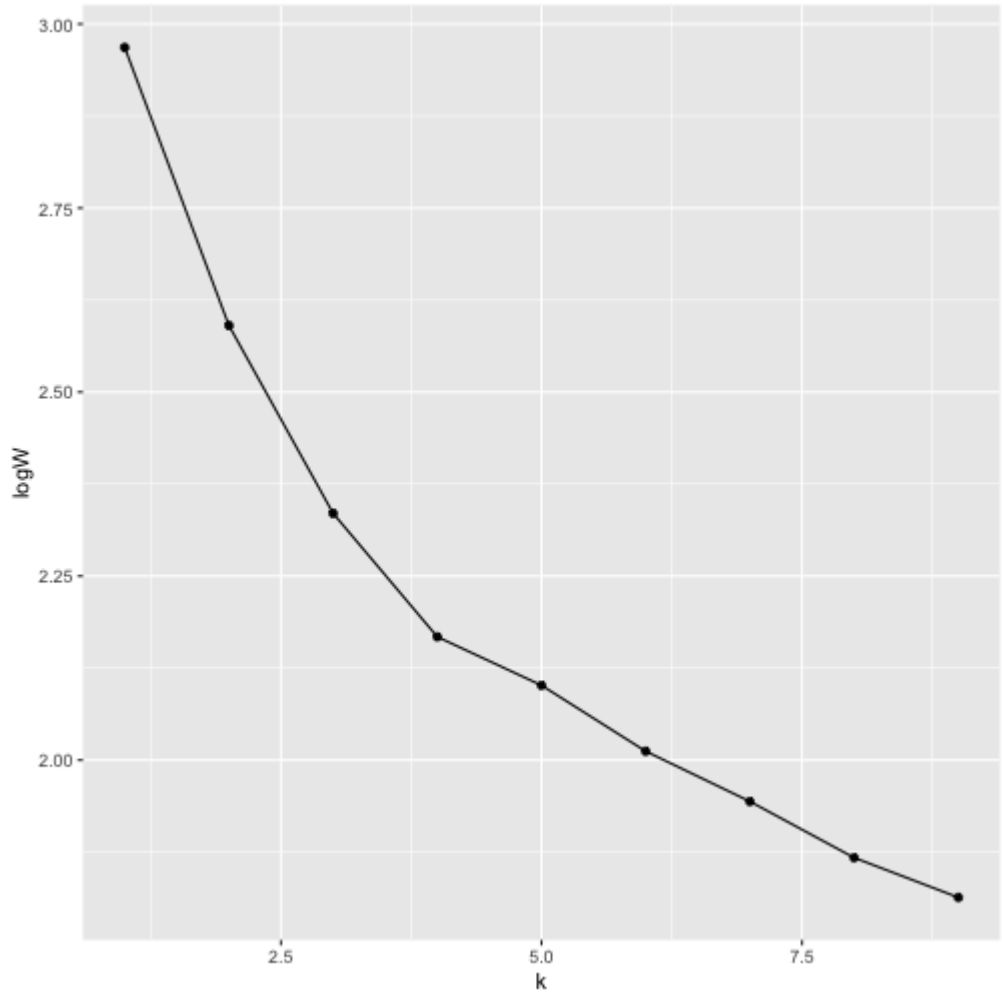
Improvement in the $W_K(C)$ statistic will be slower for values of $K > K^*$

In this case, observations belonging to a single true cluster are split into multiple cluster, all with generally high within-cluster similarity,

Splitting these clusters further will not improve the $W_K(C)$ statistic very sharply.

Choosing the number of clusters

The curve will therefore have an inflection point around K^*.



Choosing the number of clusters

The *gap statistic* is used to identify the inflection point in the curve.

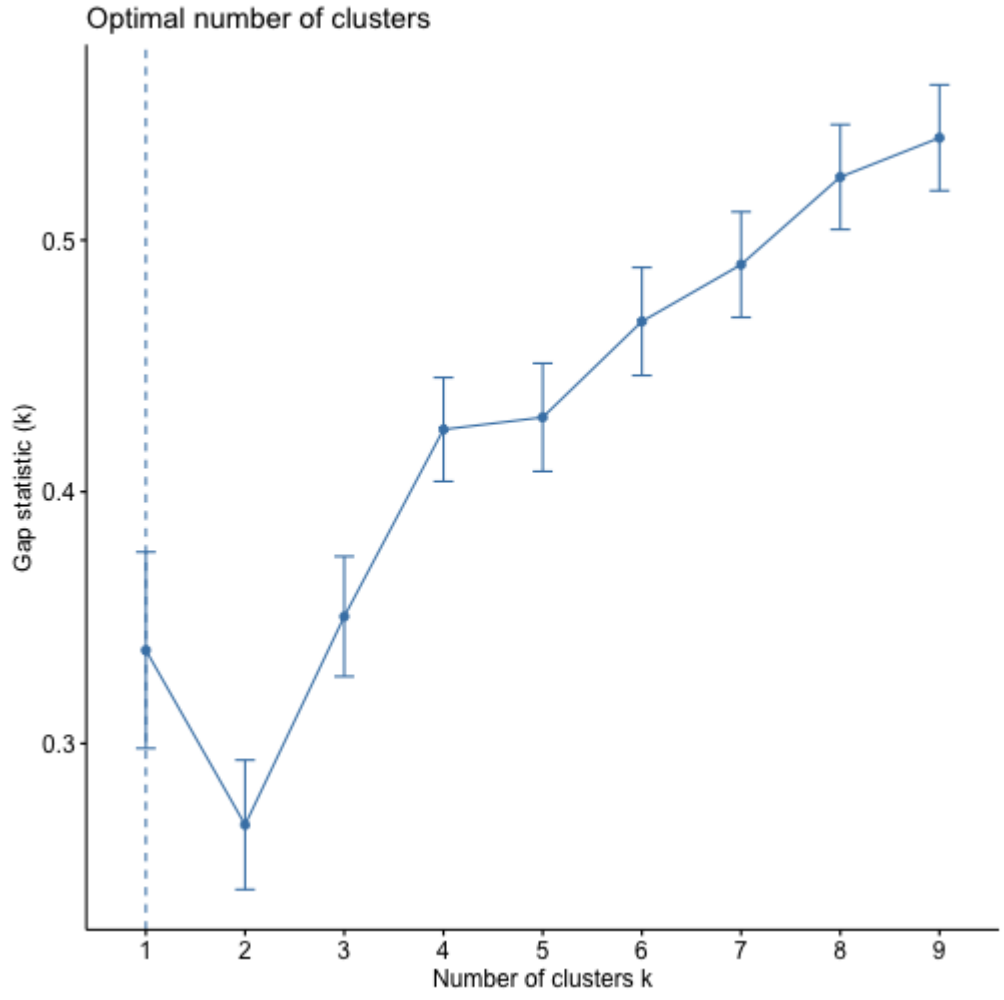
It compares the behavior of the $W_K(C)$ statistic based on the data with the behavior of the $W_K(C)$ statistic for data generated uniformly at random over the range of the data.

Chooses the K that maximizes the gap between these two $W_K(C)$ curves.

Choosing the number of clusters

For this dataset, the gap statistic suggests there is no clear cluster structure and therefore $K = 1$ is the best choice.

A choice of $K = 4$ is also appropriate.



K-medioids clustering

A variant of the same algorithm for distance that are not Euclidean

Input: Distance matrix $d(x_i, x_k)$ between observations
Output: Cluster assignments, and *a representative data point* for each cluster (medioid)

Advantage: Can apply to situations where distances between observations are available but not feature vectors (e.g., network data)

K-medioids clustering

1. Initialize by choosing K observations as medioids $m_1, m_2, \ldots, m_k$
2. Assign each observation i to the cluster with the nearest medioid, i.e., set $C(i) = \arg\min_{1 \leq k \leq K} d(x_i, m_k)$
3. Update medioids $m_k = \arg\min_{x_i \text{ s.t. } C(i)=k} \sum_{j \text{ s.t. } C(j)=k} d(x_i, x_j)$
4. Iterate steps 2 and 3 until convergence

Large-scale clustering

Cost of K-means as presented:

Each iteration: Compute distance for each point to centroid $O(knp)$

Large-scale clustering

Cost of K-means as presented:

Each iteration: Compute distance for each point to centroid $O(knp)$

This implies we have to do multiple passes over entire dataset.

Not good for massive datasets

Large-scale clustering

Cost of K-means as presented:

Each iteration: Compute distance for each point to centroid $O(knp)$

This implies we have to do multiple passes over entire dataset.

Not good for massive datasets

Can we do this in "almost" a single pass?

Large-scale clustering (BFR Algorithm)

1. Select k points as before
2. Process data file in chunks:
   - Set chunk size so each can be processed in main memory
   - Will use memory for workspace so not entire memory available

Large-scale clustering (BFR Algorithm)

For each chunk

- All points *sufficiently* close to the centroid of one of the k clusters is assigned to that cluster (*Discard Set*)

Large-scale clustering (BFR Algorithm)

For each chunk

- All points *sufficiently* close to the centroid of one of the k clusters is assigned to that cluster (*Discard Set*)
- Remaining points are clustered (e.g., using k-means with some value of k. Two cases
  - Clusters with more than one point (*Compressed Set*)
  - Singleton clusters (*Retained Set*)

Large-scale clustering (BFR Algorithm)

For each chunk

- All points *sufficiently* close to the centroid of one of the k clusters is assigned to that cluster (*Discard Set*)
- Remaining points are clustered (e.g., using k-means with some value of k. Two cases
  - Clusters with more than one point (*Compressed Set*)
  - Singleton clusters (*Retained Set*)
- Try to merge clusters in *Compressed Set*

Large-scale clustering (BFR Algorithm)

What is sufficiently close?

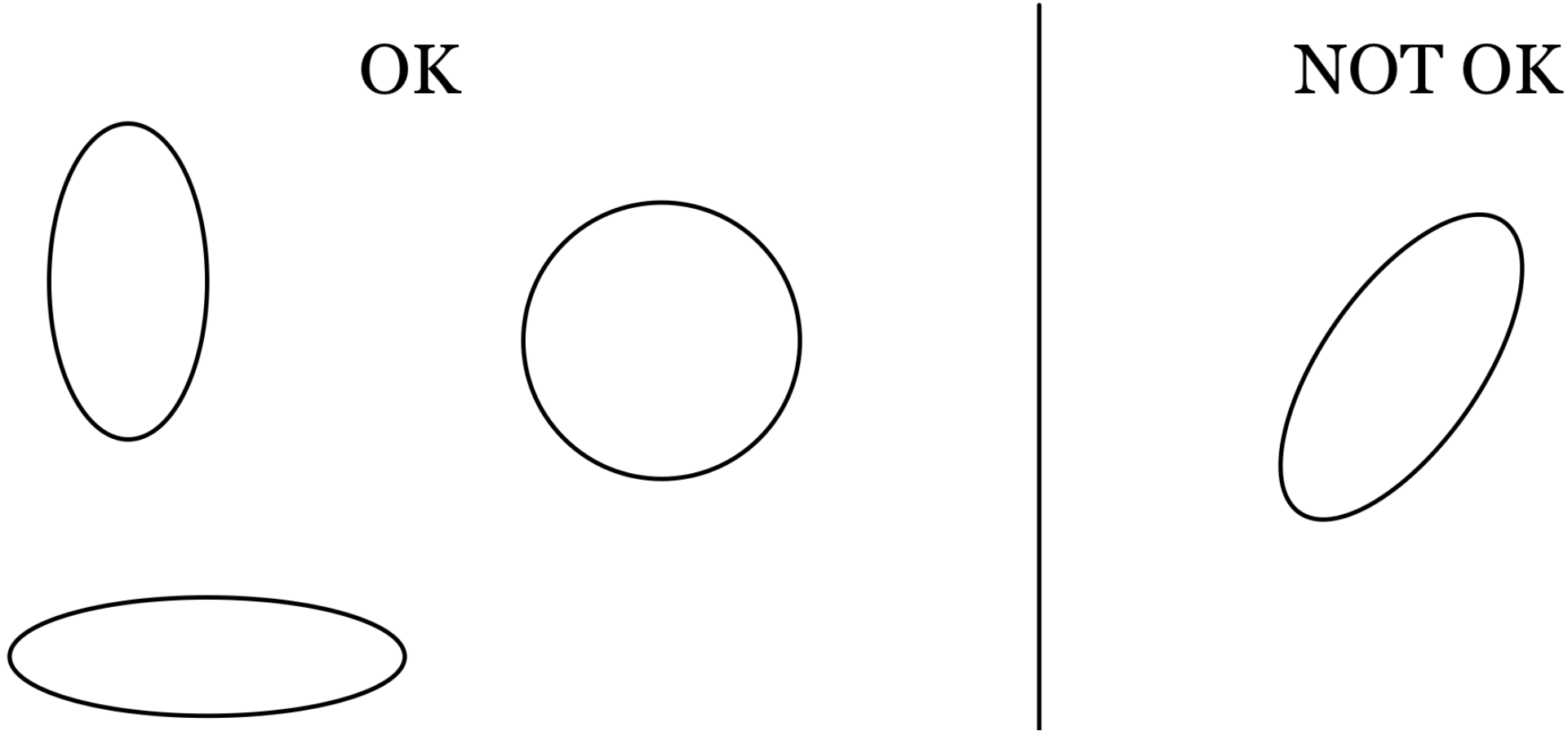
- "Weighted" distance to centroid below some threshold.

$$\sqrt{\sum_{i=1}^{d} \frac{(p_i - c_i)^2}{\sigma_i}}$$

- c_i: cluster mean for feature i
- σ_i: cluster standard deviation of feature i

Large-scale clustering (BFR Algorithm)

Assumption: points are distributed along axis-parallel ellipses



Large-scale clustering (BFR Algorithm)

Under this assumption, we only need to store means and variances to calculate distances

Large-scale clustering (BFR Algorithm)

Under this assumption, we only need to store means and variances to calculate distances

We can do this by storing for each cluster j:

- N_j number of points assigned to cluster
- s_{ij} sum of values of feature i in cluster j
- s_{ij}^2 sum of squares of values of feature i in cluster j

Large-scale clustering (BFR Algorithm)

Under this assumption, we only need to store means and variances to calculate distances

We can do this by storing for each cluster j:

- N_j number of points assigned to cluster
- s_{ij} sum of values of feature i in cluster j
- s_{ij}^2 sum of squares of values of feature i in cluster j

Constant amount of space to represent cluster

Large-scale clustering (BFR Algorithm)

Under this assumption, we only need to store means and variances to calculate distances

We can do this by storing for each cluster j:

- N_j number of points assigned to cluster
- s_{ij} sum of values of feature i in cluster j
- s_{ij}^2 sum of squares of values of feature i in cluster j

Constant amount of space to represent cluster

Exercise. show these are sufficient to calculate weighted distance

Large-scale clustering (BFR Algorithm)

This is used to represent (final) clusters in *Discard Set* and (partial) clusters in *Compressed Set*

Only points explicitly in memory are those in the *Retained Set*

Large-scale clustering (BFR Algorithm)

This is used to represent (final) clusters in *Discard Set* and (partial) clusters in *Compressed Set*

Only points explicitly in memory are those in the *Retained Set*

Points outside of *Retained Set* are never kept in memory (written out along with cluster assignment)

Large-scale clustering (BFR Algorithm)

Merging clusters in *Compressed Set*

Example: Merge if the variance of combined clusters is sufficiently close to variance of separate clusters

Large-scale clustering (BFR Algorithm)

After all data is processed:

- Assign points in *Retained Set* to cluster with nearest centroid
- Merge partial clusters in *Compressed Set* with final clusters in *Discarded Set*

Large-scale clustering (BFR Algorithm)

After all data is processed:

- Assign points in *Retained Set* to cluster with nearest centroid
- Merge partial clusters in *Compressed Set* with final clusters in *Discarded Set*

Or,

Flag all of these as *outliers*

Summary

- Clustering algorithms used to partition data into groups of similar observations
- K-means and K-mediods: iterative algorithms to minimize a partition objective function
- Optimization solution depends on initialization: K-means++ improved initialization
- BFR Algorithm: how to solve for massive datasets in "almost" one pass of algorithm