Unsupervised Learning with Clustering

K-means Clustering

Why Clustering?

Interpret and label clusters

Identify important features

Characterize new points by the closest cluster (or nearest neighbors)

Use the cluster assignments as a compression or summary of the data

Clustering

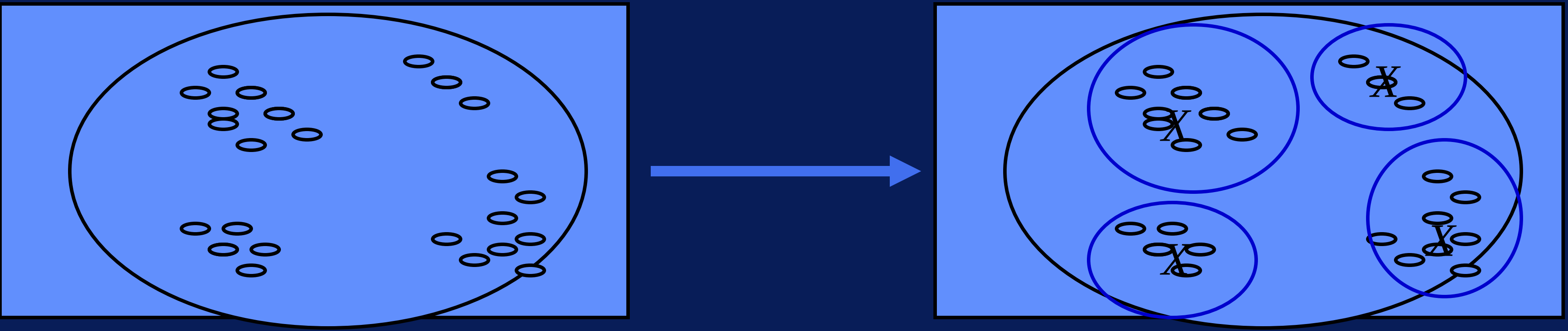
Basic idea: Group similar things together

Unsupervised Learning

K-means

- Partitioning instances into k (disjoint) clusters
- Measure of similarity

2D Clustering Example



Clustering

Unsupervised: no target value to be predicted

Differences ways clustering results can be produced/represented/learned

- Exclusive vs. overlapping
- Deterministic vs. probabilistic
- Hierarchical vs. flat
- Incremental vs. batch learning

Many Clustering Techniques

K-means clustering

Hierarchical clustering

Conceptual clustering

Probability-based clustering

Bayesian clustering

Common uses of Clustering

Often used as an exploratory data analysis tool

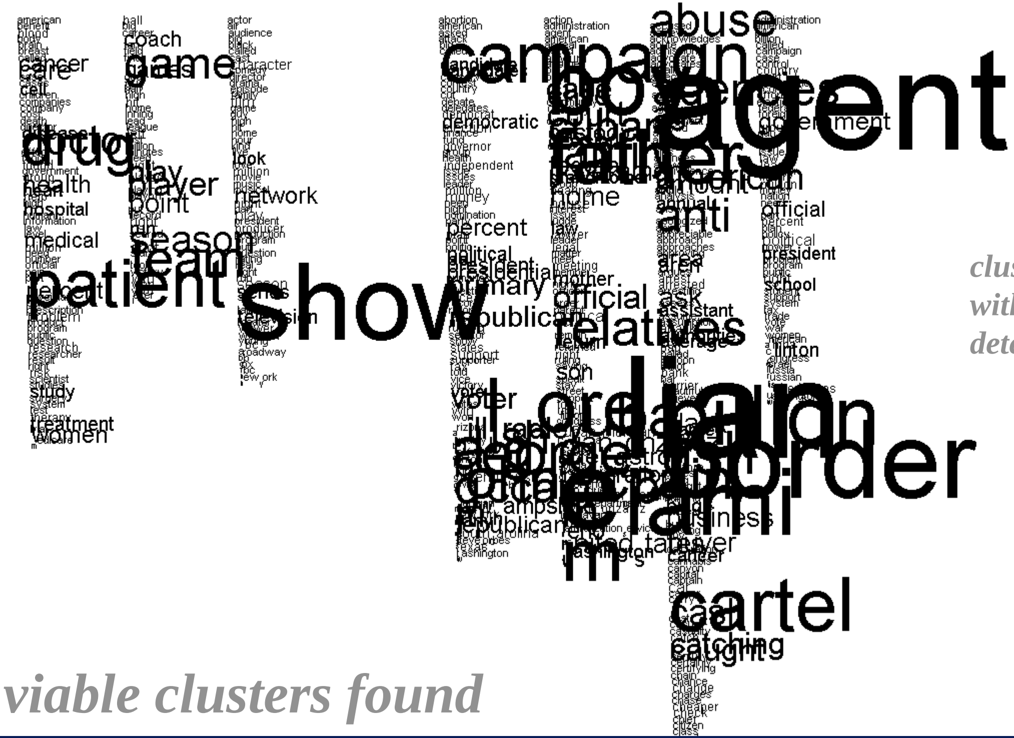
In one-dimension, a good way to quantify real-valued variables into k non-uniform buckets

Used on acoustic data in speech understanding to convert waveforms into one of k categories (known as Vector Quantization)

Also used for choosing color palettes on old fashioned graphical display devices

Color Image Segmentation

K-means Example



cluster means shown with coordinates determining fontsize

7 viable clusters found

Clustering Objective

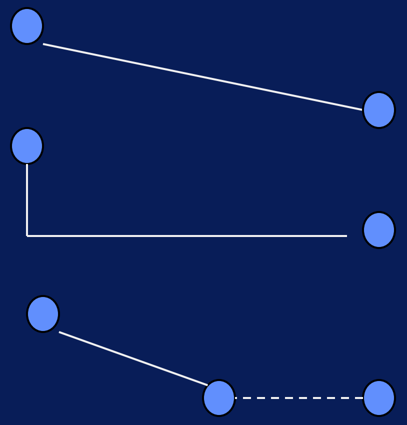
Find subsets that are similar within cluster and dissimilar between clusters

Similarity defined by distance measures

Euclidean distance

Manhattan distance

Mahalanobis
(Euclidean w/dimensions
rescaled by variance)



Similarity Defined by Distance Measures

Euclidean distance =

sqrt[(a1 - b1)^2+ (a2 – b2)^2+...)]

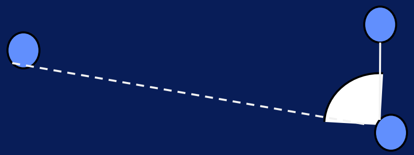
Manhattan distance

[|a1 - b1|+ |a2 – b2|+...)]

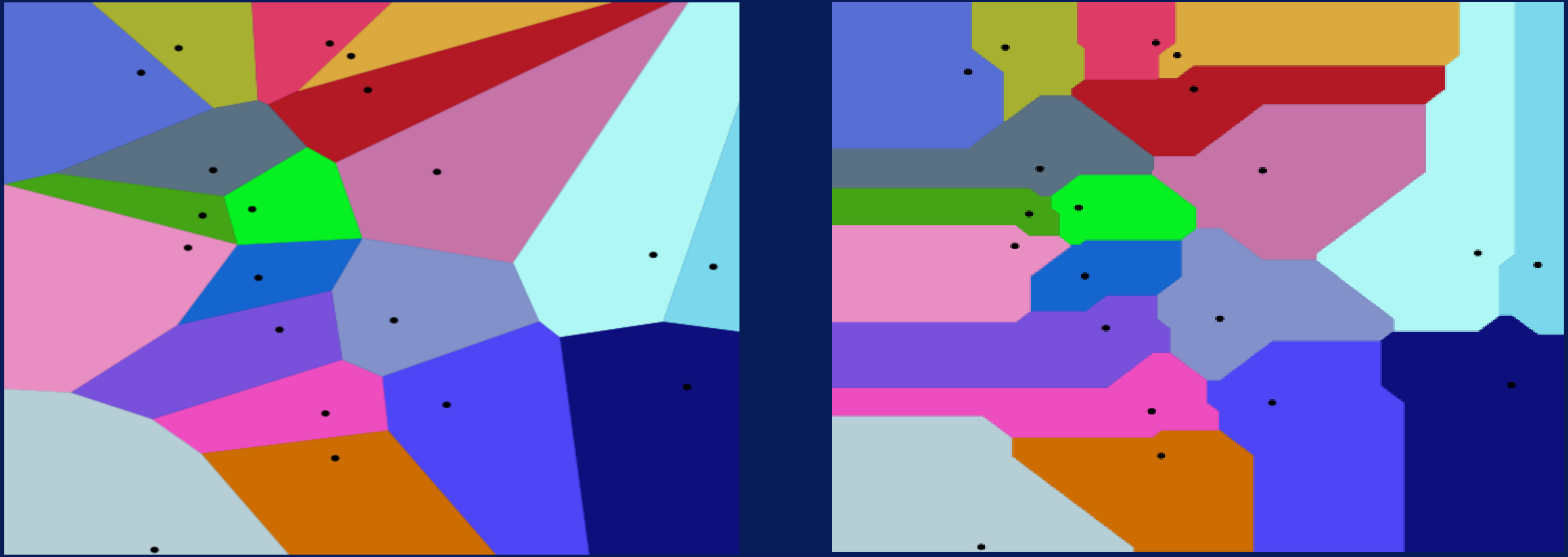
Cosine (insensitive to size)

Euc Dist/

sqrt[(a1)^2 +(a2)^2..]*sqrt[(b1)^2+...)]



Euclidean Vs. Manhattan



The *k*-means Algorithm
Iterative Distance Based Clustering

Clusters the data into k groups
where k is specified in advance

The *k*-means Algorithm

Iterative Distance Based Clustering

1. Cluster centers are chosen at random
2. Instances are assigned to clusters based on their distance to the cluster centers
3. Centroids of clusters are computed – “means”
4. Go to 1st step until convergence

A simple, effective, standard method

Start with K initial cluster centers

Loop:

Assign each data point to nearest cluster center

Calculate mean of cluster for new center

Stop when assignments don’t change

K-means Issues

How to choose K?

How to choose initial centers?

Will it always stop?

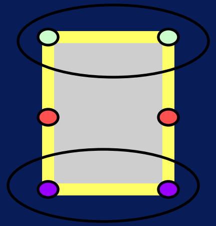
K-Means Pros & Cons

- **Simple and reasonably effective**
- **The final cluster centers do not represent a global minimum but only a local one**
- **Result can vary significantly based on initial choice of seeds**
  - Completely different final clusters can arise from differences in the initial randomly chosen cluster centers
- **Algorithm can easily fail to find a reasonable clustering**

Getting Trapped in a Local Minimum

Example: four instances at the vertices of a two-dimensional rectangle

- Local minimum: two cluster centers at the midpoints of the rectangle's long sides



Simple way to increase chance of finding a global optimum: restart with different random seeds

Clustering

- **Partition unlabeled examples into disjoint subsets of *clusters*, such that:**
  - Examples within a cluster are very similar
  - Examples in different clusters are very different
- **Discover new categories in an *unsupervised* manner (no sample category labels provided)**

K-Means Algorithm

Let d be the distance measure between instances

Select k random instances $\{s_1, s_2, \ldots s_k\}$ *as seeds*

Until clustering converges or other stopping criterion:

    For each instance x_i:

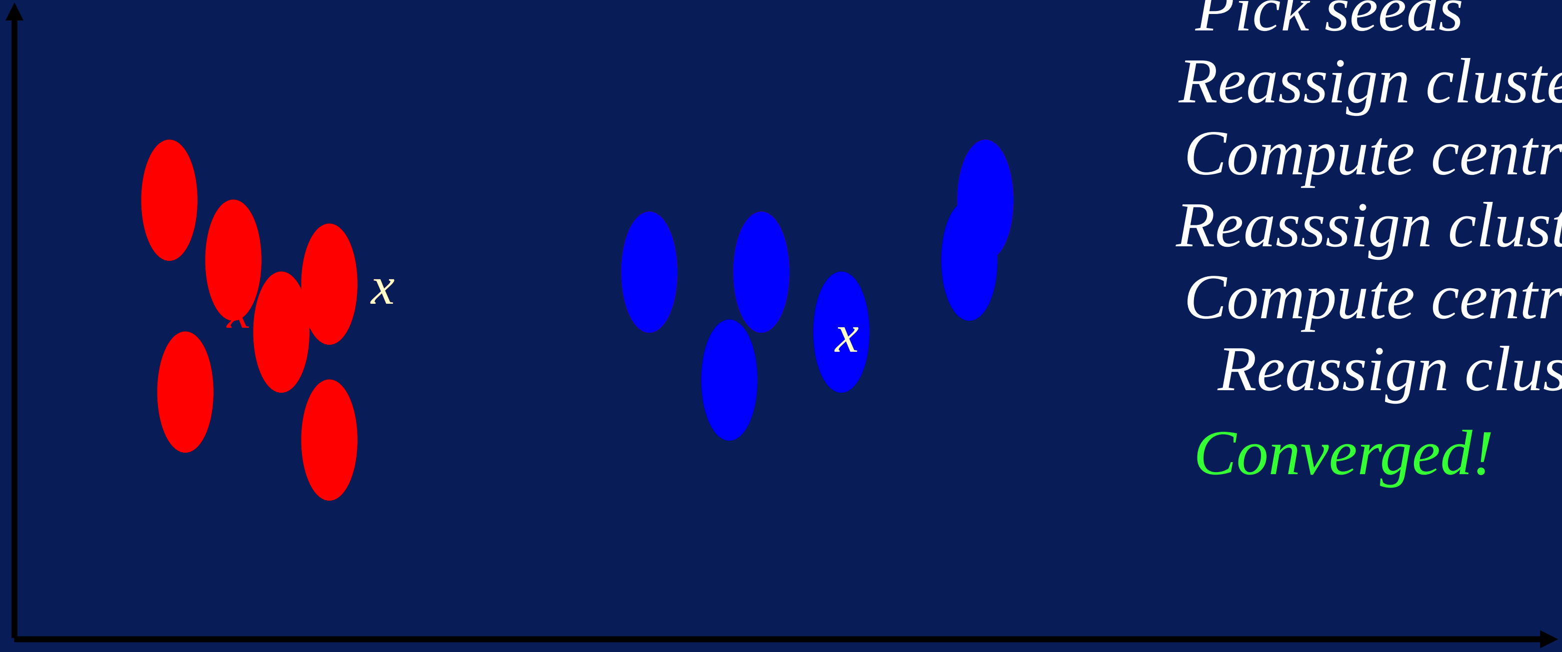
        Assign x_i *to the cluster* c_j *such that* $d(x_i, s_j)$ *is minimal*

    (Update the seeds to the centroid of each cluster)

    For each cluster c_j

        $s_j = \mu(c_j)$

K Means Example
(K=2)



Pick seeds
Reassign clusters
Compute centroids
Reasssign clusters
Compute centroids
Reassign clusters
Converged!

Random Seed Choice

Results can vary based on random seed selection

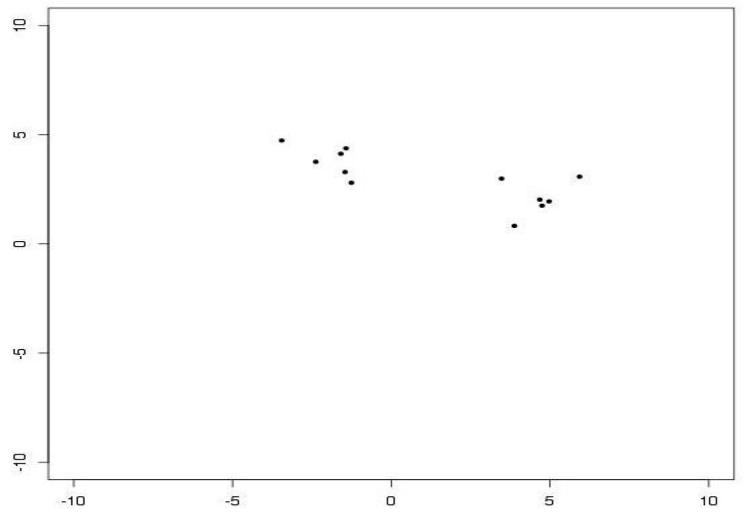
Some seeds can result in poor convergence rate, or convergence to sub-optimal clusters

Select good seeds using a heuristic or the results of another method

K-means Example

For K=1, using Euclidean distance, where will the cluster center be?

X_2

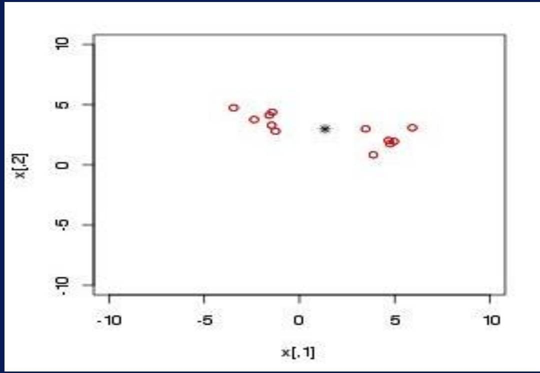


X_1

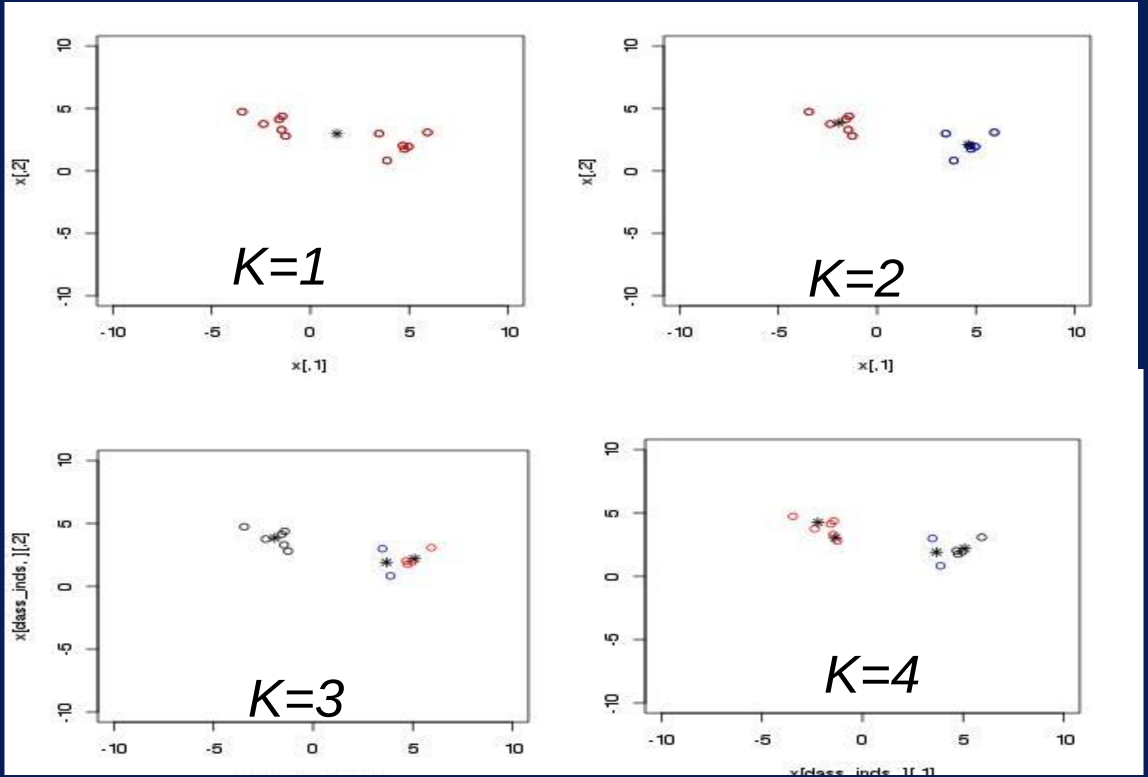
K-means Example

For K=1, the overall mean minimizes Sum Squared Error (SSE), aka Euclidean distance

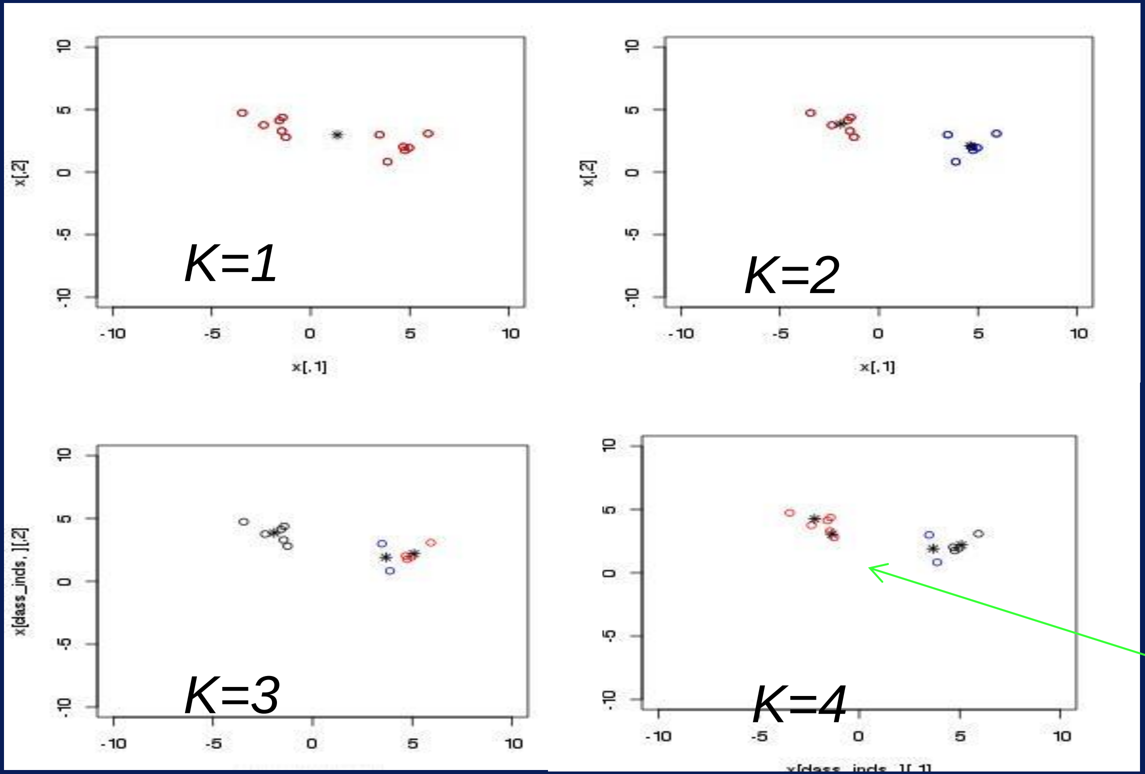
Simple example:
#choose 1 data point as initial K centers
#10 is max loop iterations
#1 is number of initial sets to try



K-means Example



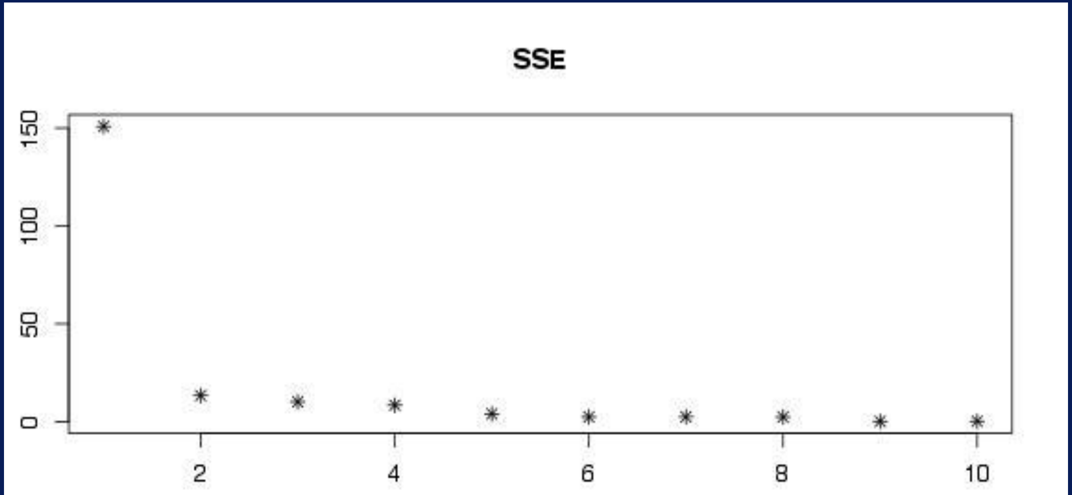
K-means Example



As K increases individual points get a cluster

Choosing K for K-means

Total Within Cluster SSE



K=1 to 10

Not much improvement after K=2 (“elbow”)

K-means Clustering Issues

Scale

Dimensions with large numbers may dominate distance metrics

Outliers

Outliers can pull cluster mean, K-mediods uses median instead of mean

K-Means Clustering Summary

Labeled clusters can be interpreted by using supervised learning - train a tree or learn rules

Can be used to fill in missing attribute values

All methods have a basic assumption of independence between the attributes

- Some methods allow the user to specify in advanced that two of more attributes are dependent and should be modeled with a joint probability

Call out box for Labels

Call out box for definitions

Call out box important points/takeaways

Call out box for drawing attention (or some other thing)