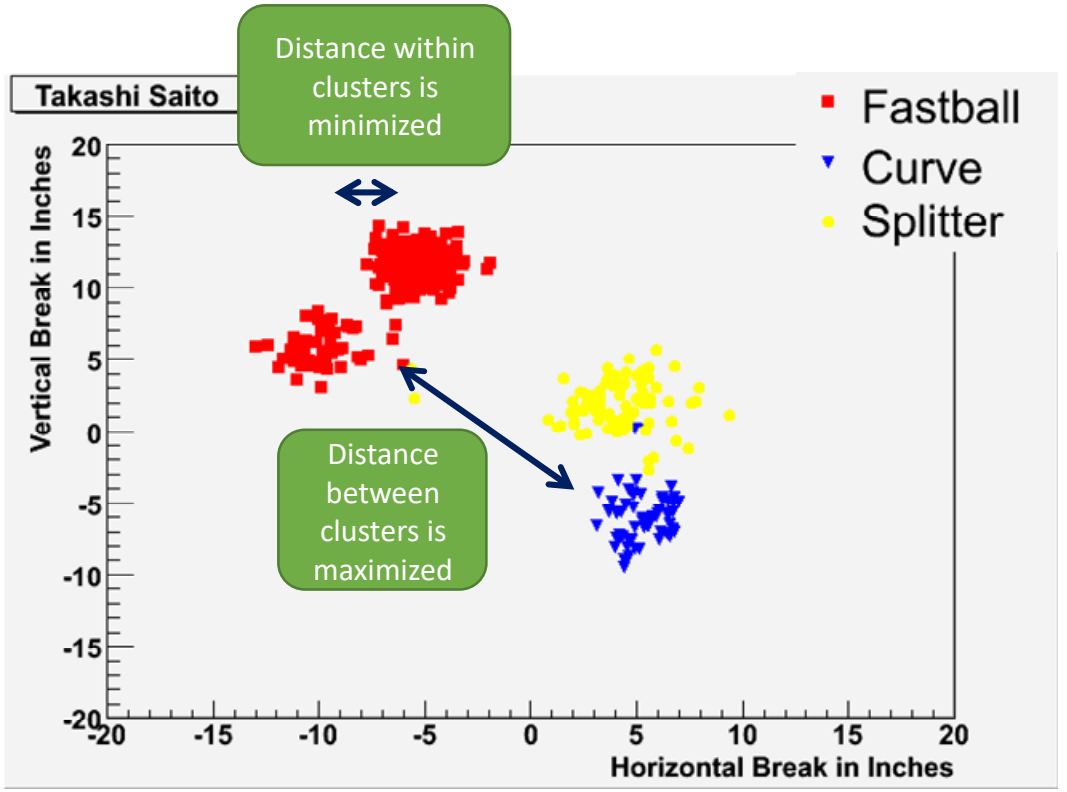
Clustering

What is Clustering?

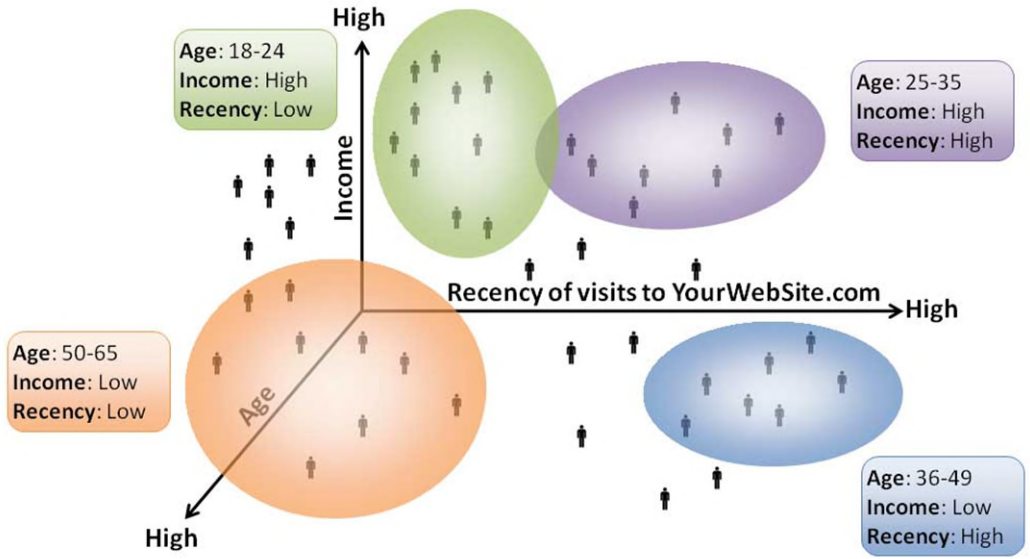
- Grouping data so that elements within a group will be
  - Similar (or related) to one another
  - Dissimilar (or unrelated) from elements in other groups.



http://www.baseball.bornbybits.com/blog/uploaded_images/Takashi_Saito-703616.gif

Clustering

- Used to determine distinct groups of data
- Based on data across multiple dimensions
- Uses
  - Customer segmentation
  - Identifying patient care groups
  - Performance of business sectors



Here you have four clusters of web site visitors.

What does this tell you?

from http://www.datadrivesmedia.com/two-ways-performance-increases-targeting-precision-and-response-rates/

Applications

Understanding

- Group related documents for browsing
- Create groups of similar customers
- Discover which stocks have similar price fluctuations

Summarization

- Reduce the size of large data sets
- Those similar groups can be treated as a single data point

Even more examples

Marketing

- Discover distinct customer groups for targeted promotions

Insurance

- Finding “good customers” (low claim costs, reliable premium payments)

Healthcare

- Find patients with high-risk behaviors

What cluster analysis is NOT

Manual ("supervised") classification

- People simply place items into categories

Simple segmentation

- Dividing students into groups by last name

Main idea:

The clusters must come from the data, not from external specifications.

Creating the "buckets" beforehand is categorization, but not clustering.

Clusters can be ambiguous

How many clusters?

6

2

4

The difference is the threshold you set.
How distinct must a cluster be to be it's own cluster?

Two clustering techniques

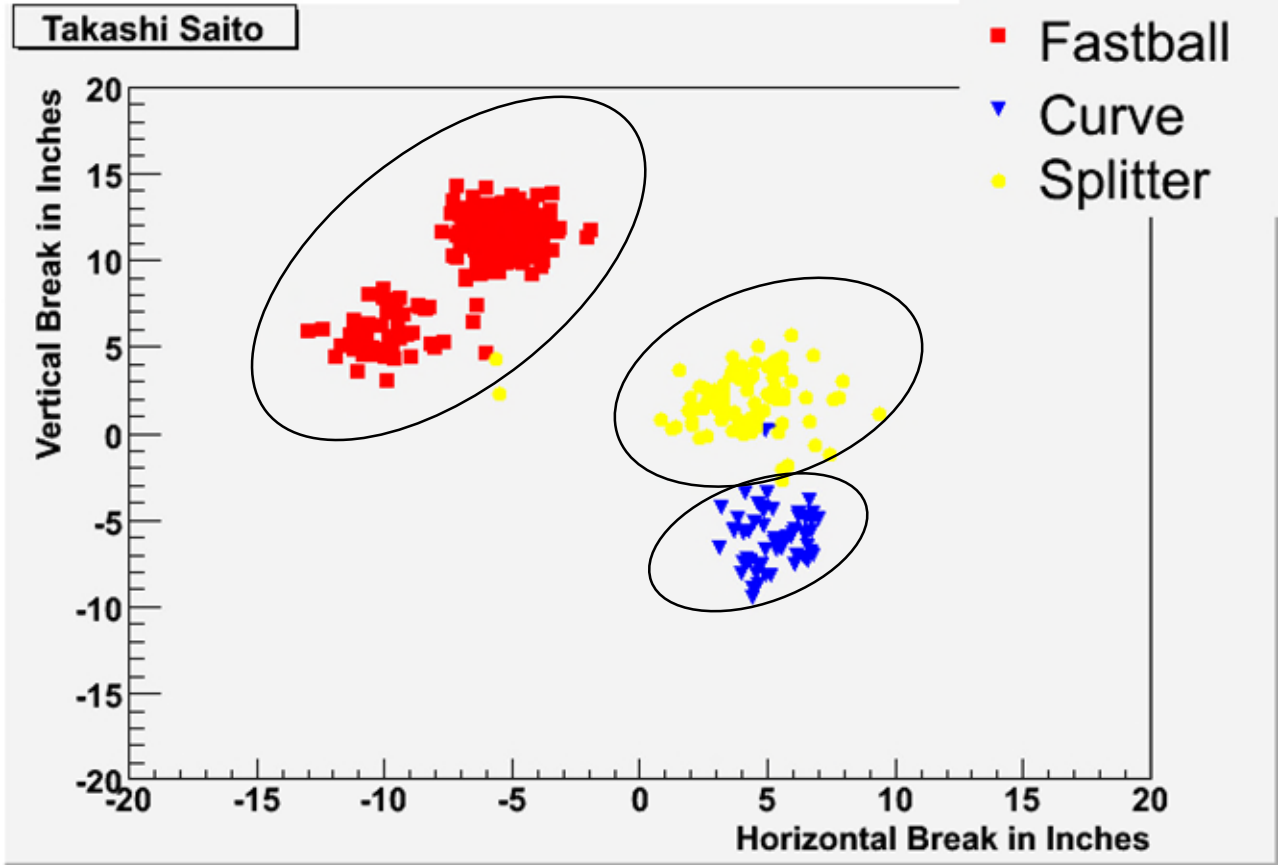
Partition

- Non-overlapping subsets (clusters) such that each data object is in exactly one subset

Hierarchical

- Set of nested clusters organized as a hierarchical tree

Partitional Clustering

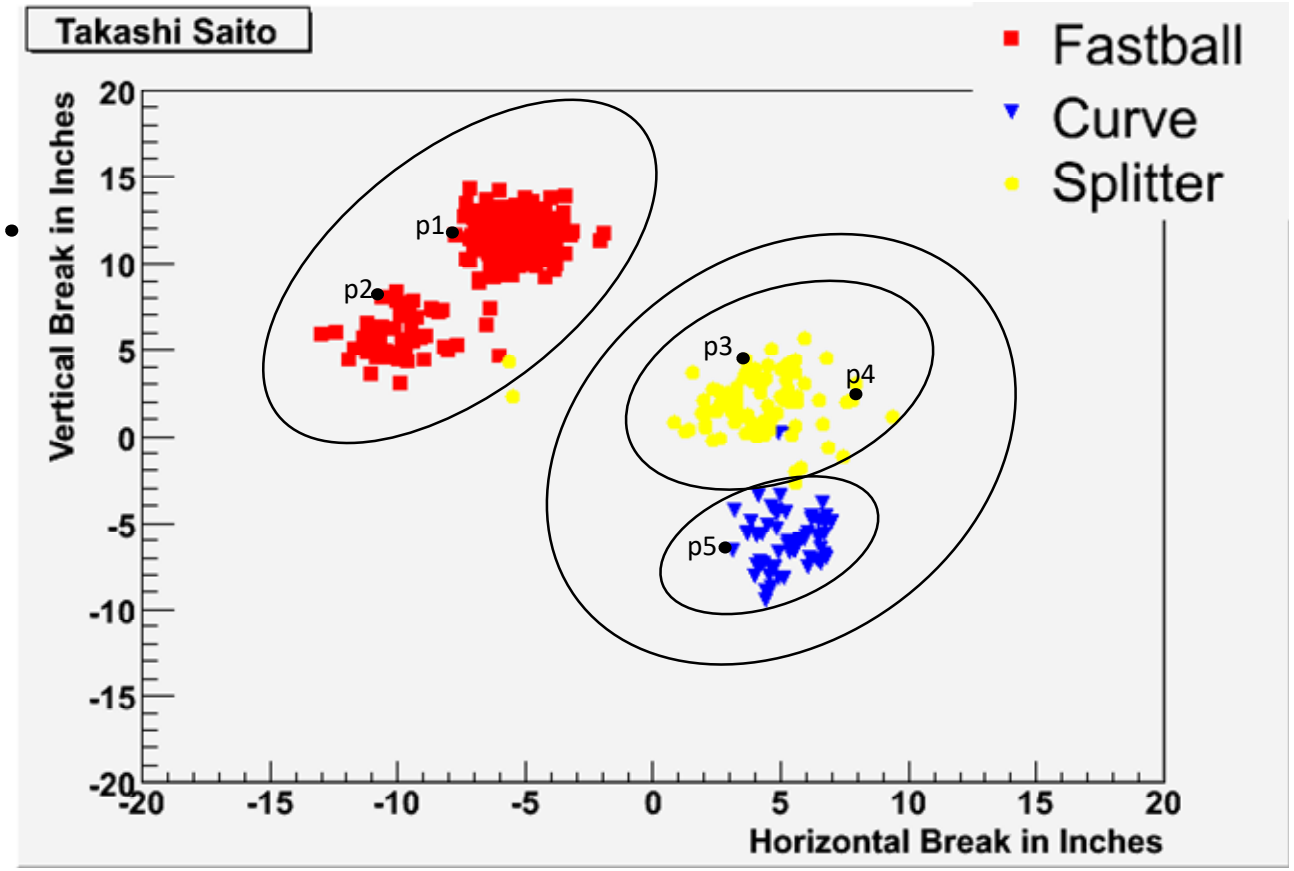


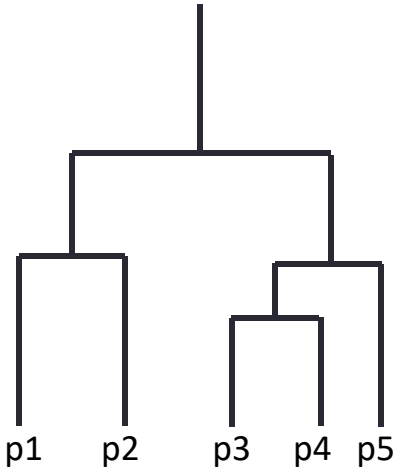
Three distinct groups emerge, but…

…some curveballs behave more like splitters.

…some splitters look more like fastballs.

Hierarchical Clustering





This is a dendrogram

Tree diagram used to represent clusters

Clusters can be ambiguous

How many clusters?

6

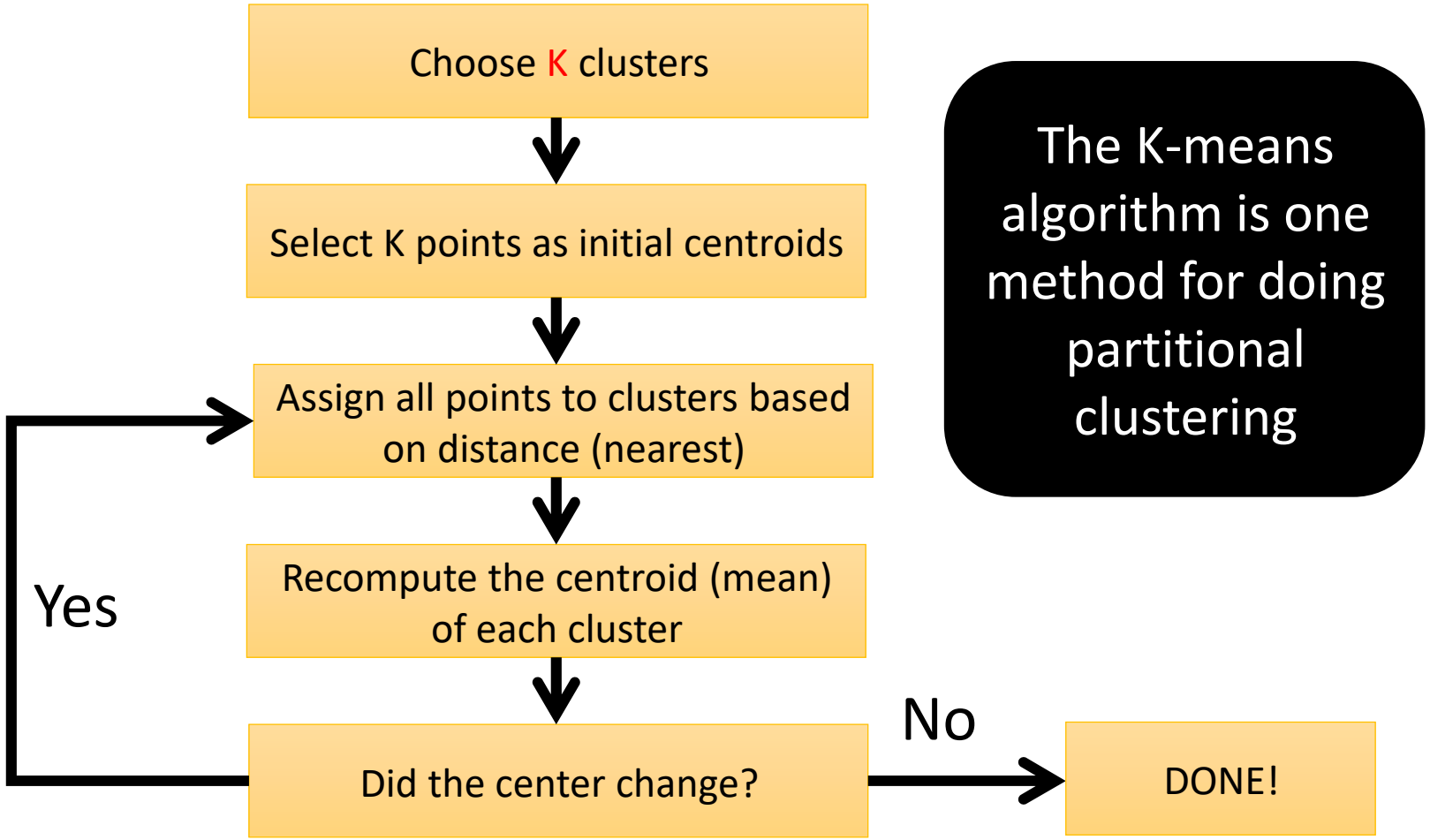
2

4

The difference is the threshold you set.
How distinct must a cluster be to be it's own cluster?

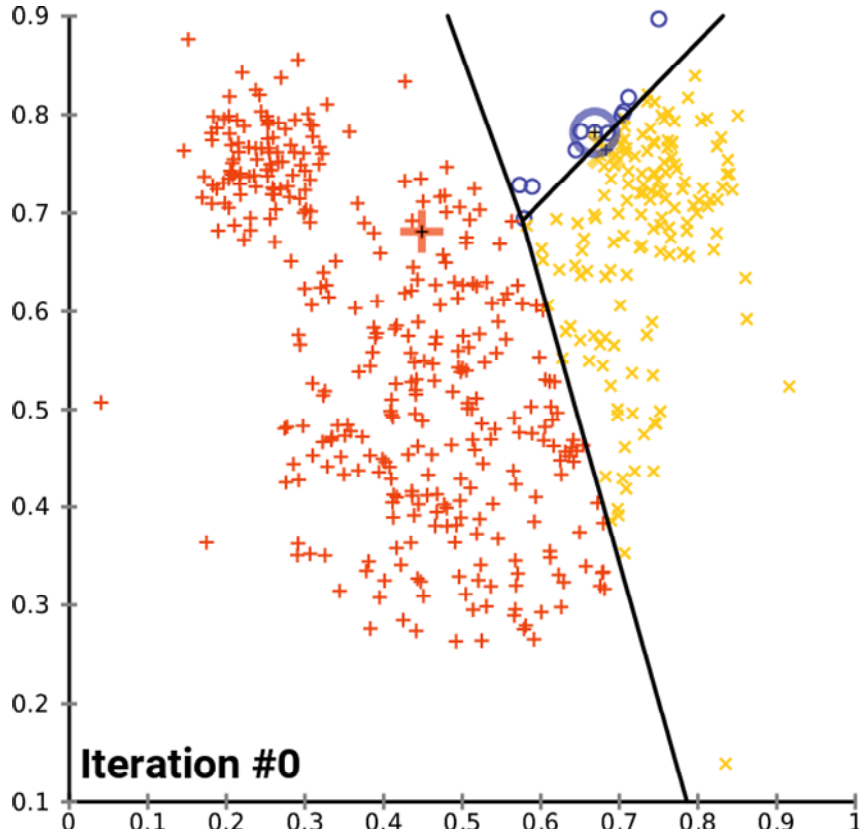
adapted from Tan, Steinbach, and Kumar. Introduction to Data Mining (2004)

K-means (partitional)



The K-means algorithm is one method for doing partitional clustering

K-Mean clustering



Choosing the initial centroids

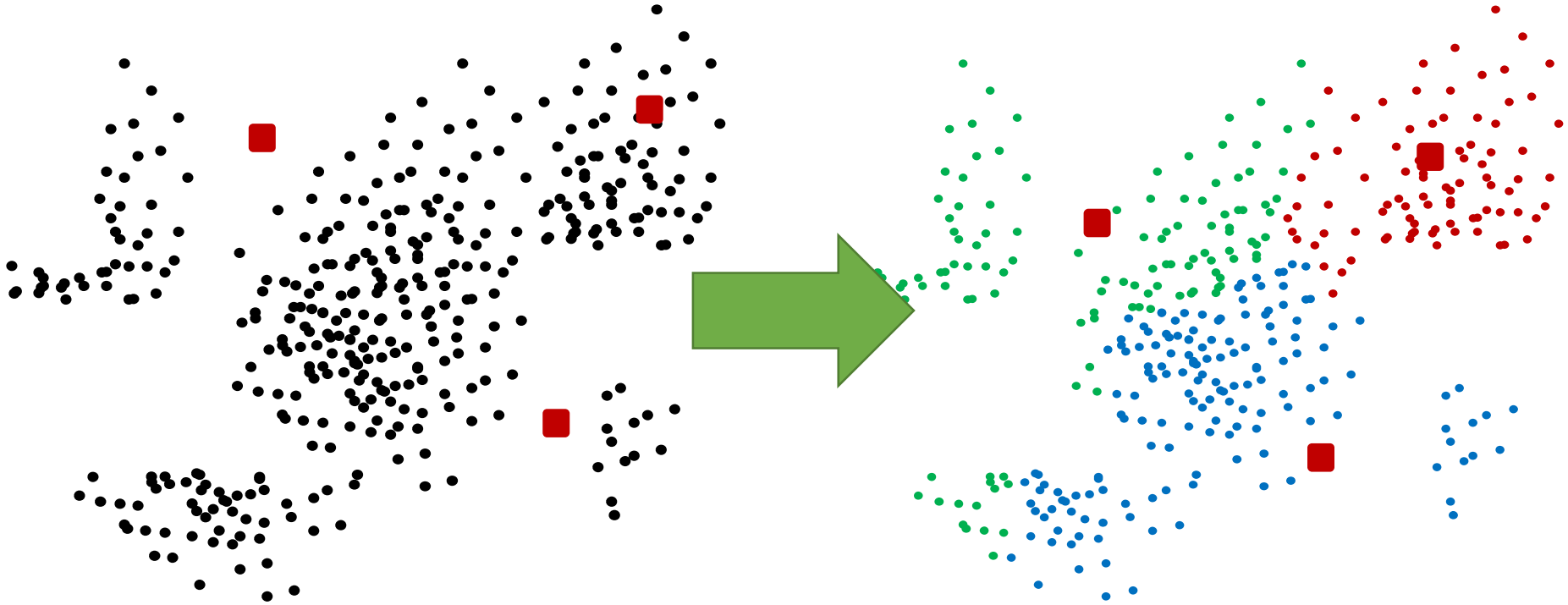
It matters

- Choosing the right number
- Choosing the right initial location

Bad choices create bad groupings

- They won't make sense within the context of the problem
- Unrelated data points will be included in the same group

Example of Poor Initialization



This may “work” mathematically but the clusters don’t make much sense.

Limitations of K-Means Clustering

K-Means gives unreliable results when

- Clusters vary widely in size
- Clusters vary widely in density
- Clusters are not in rounded shapes
- The data set has a lot of outliers

The clusters may never make sense.
In that case, the data may just not be well-suited for clustering!

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes	Distances between nearest points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large `n_clusters` and `n_samples`	Large dataset, outlier removal, data reduction.	Euclidean distance between points

Scikit-learn for K-Mean clustering

- K-Mean clustering can be with scikit-klearn easily :

```
from sklearn.cluster import KMeans
```

- You need to first initialize the classifier by creating a kmean classification model object:

```
kmeans = KMeans(n_clusters=k)
```

n_clusters is the only required argument to specify the number of clusters you want.

Important arguments

- Besides the number of clusters, there are few important arguments you should be aware of:
- algorithm: default “auto”, “full” or “elkan”.
- max_iter: a number, default 300. Specify the maximum number of iterations of the k-means algorithm for a single run. Depending on your dataset, it may never converge.
- n_jobs: default "None" or number. Specify how many concurrent processes/threads should be used for parallelized routines. None means to use 1, -1 means to use all processor.

- `kmeans = KMeans(algorithm='auto', max_iter=300, n_clusters=4, n_jobs=1)`
- https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

The Iris example

```
import pandas as pd
import numpy as np
import seaborn as sns
iris = sns.load_dataset('iris')
```

Let's do some exploration first.

Since clustering works with unlabelled data, we need to first remove the label "species" from our dataset:

```
iris_train = iris[['sepal_length','sepal_width','petal_length','petal_width']]
columns = iris.columns[0:4]
iris_train = iris[columns]
```

- Initialize a basic k-mean classifier. KMeans() creates a classifier object.

kmeans = KMeans(n_clusters=3, max_iter = 100)

- Compute the model's parameters based on the "training" data. fit() performs the learning process and returns a computed classifier.

kmeans.fit(iris_train)

- Use the computed classifier to "predict" the classification of your data. It returns a series of predicted labels.

prediction = kmeans.predict(iris_train)

Evaluation

- Since clustering is an unsupervised learning, there is no "right" answer to be tested against. You are not trying to predict if one observation is accurately classified into a specific group. E.g. row 1 belongs to setosa. Instead, you are grouping similar observations into the same group, whether this group is setosa or not is not relevant.
- Therefore, if you have some observations with known groups, the evaluation of clustering can be done with V-measure.
- If you have no observation with known groups, then the evaluation can be done with Silhouette Coefficient

V-measure

- V-measure is an score by calculating the mean of homogeneity and completeness. The measurements are computed by comparing the memberships of known groups to memberships of predicted groups.
- homogeneity: each cluster contains only members of a single class.
- completeness: all members of a given class are assigned to the same cluster.
- from sklearn import metrics
- metrics.homogeneity_score(labels_true, labels_pred)
- metrics.completeness_score(labels_true, labels_pred)
- metrics.v_measure_score(labels_true, labels_pred)

```
from sklearn import metrics
metrics.homogeneity_score(iris['species'], iris['predicted']) #0.75148
metrics.completeness_score(iris['species'], iris['predicted']) #0.76498
metrics.v_measure_score(iris['species'], iris['predicted']) #0.75817
```

- Bounded scores: 0.0 is as bad as it can be, 1.0 is a perfect score.
- No assumption is made on the cluster structure

Silhouette Coefficient

- Silhouette Coefficient can be used when there is no known labels. It only relies on assessing the distance between the observations within the same group and across different groups.
  - **a**: The mean distance between a sample and all other points in the same class. (similarity within the same cluster)
  - **b**: The mean distance between a sample and all other points in the *next nearest cluster*. (dissimilarity between different clusters)
  - s=(b-a)/max(a,b)
- metrics.silhouette_score(X, labels, metric='euclidean')

- from sklearn import metrics
- metrics.silhouette_score(iris_train, prediction, metric='euclidean') #0.55281
- The score is bounded between -1 for incorrect clustering and +1 for highly dense clustering.

Finding the right number of clusters

- No golden rules.
- Reality and cost.
- Comparing the evaluation metrics.