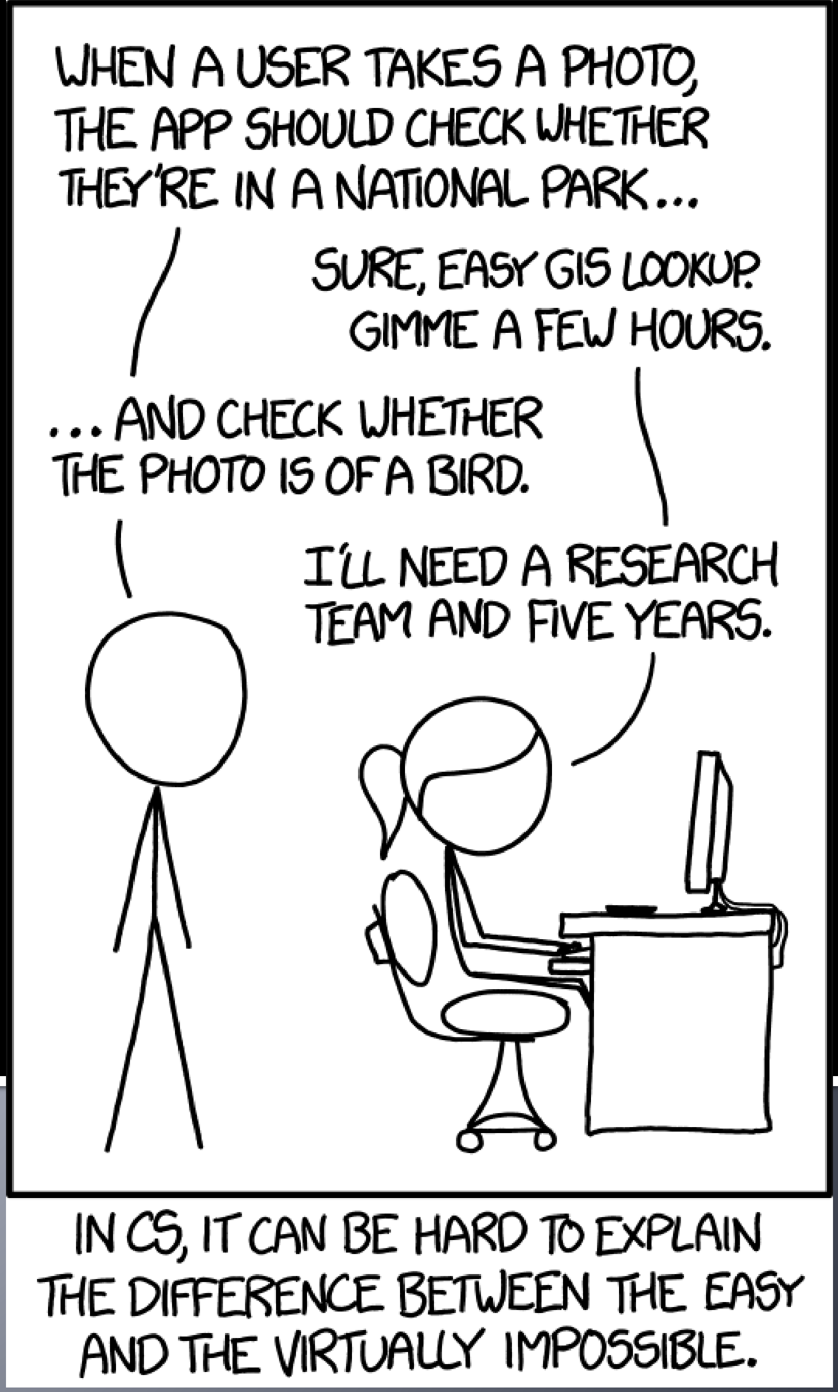
INFO 251: Applied Machine Learning

Unsupervised Learning



Course Outline

- Causal Inference and Research Design
  - Experimental methods
  - Non-experiment methods
- Machine Learning
  - Design of Machine Learning Experiments
  - Linear Models and Gradient Descent
  - Non-linear models
  - Neural models
  - Fairness and Bias
  - Practicalities and summary
  - **Unsupervised Learning**
- Special topics

Outline

- **Unsupervised learning: intro**
- Cluster Analysis
  - *k*-Means clustering
  - Hierarchical clustering
  - Clustering in Python
- Dimensionality Reduction
  - Intuition
  - Principal Component Analysis
  - Example: Eigenfaces

Unsupervised Learning

- Why unsupervised learning?
  - Can't always obtain labeled data
  - Obtaining labeled data can be *expensive*
  - Useful in exploratory analysis
  - Can reduce noise and complexity of problems
    - Particularly methods for dimensionality reduction, which will be covered in a subsequent lecture?
  - Can be used as precursor to supervised learning

Cluster analysis: basic principles

- Idea: Find natural groupings in data
  - Form of unsupervised learning
  - Oftentimes, "correct" groupings are unknown
  - **Key idea:** Samples within a group should be more similar than samples in different groups

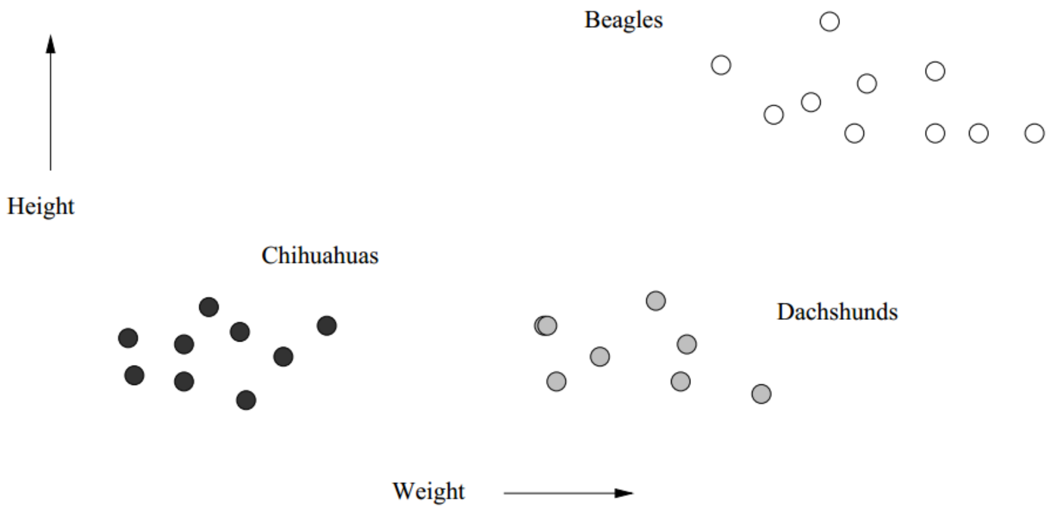
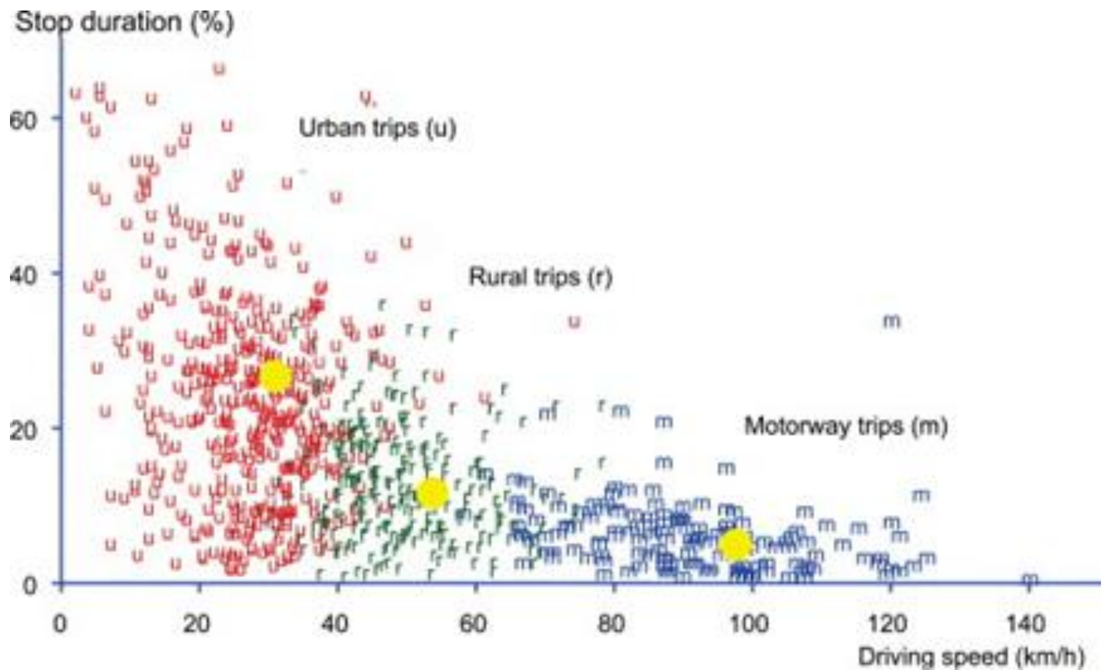
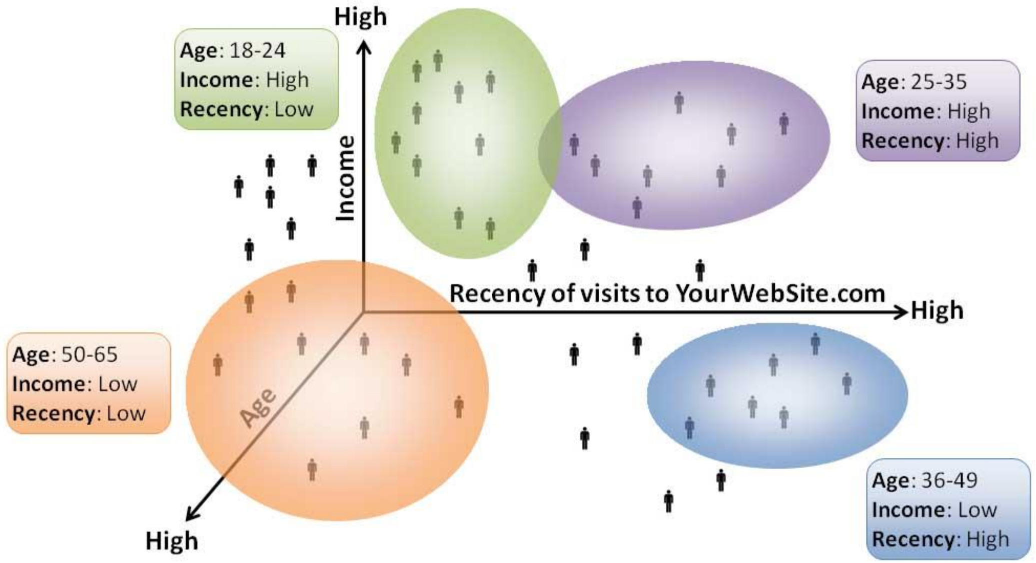


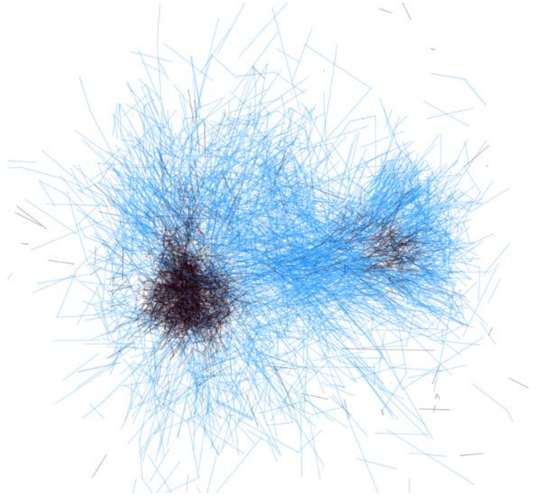
Figure 7.1: Heights and weights of dogs taken from three varieties



Other common applications

- Market segmentation
- Customer/voter profiling
- Social network analysis
- Image segmentation
- Land use analysis and city planning
- Crime analysis
- Recommender systems (coming soon!)
- ...



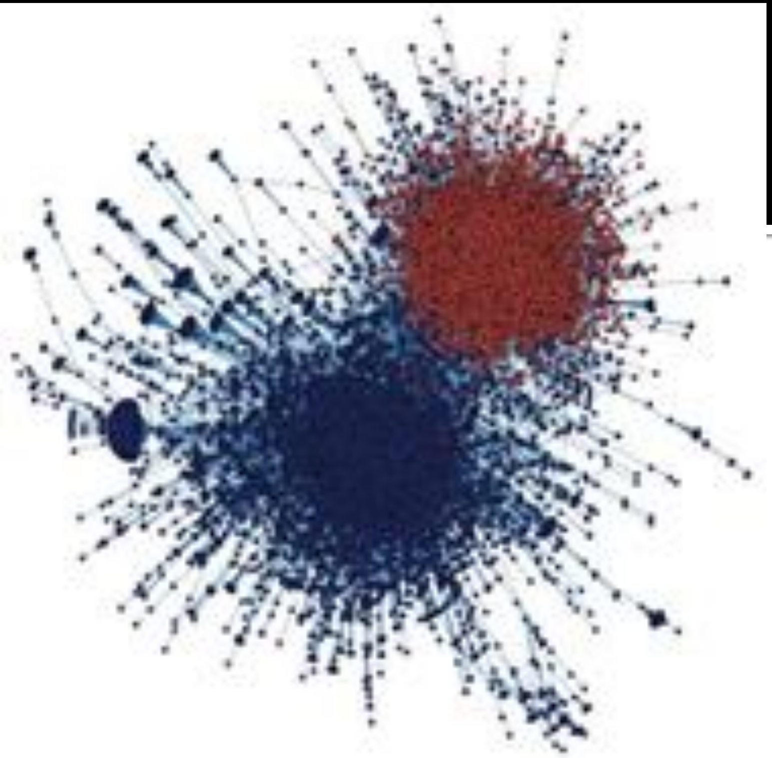


Cluster analysis: key components

- Key ingredients
  - A distance/similarity/dissimilarity function
  - A "loss function" to evaluate clusters
  - An algorithm to optimize loss function

- See: Hastie et al, The Elements of Statistical Learning, Chapter 14

Distance Metrics



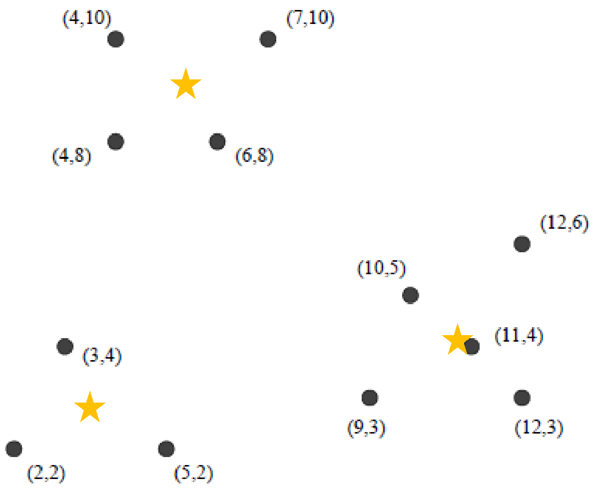
- Requirements of a distance metric
  - Non-negative
  - Symmetric
  - Satisfies triangle inequality
- Examples
  - L^n-Norm: $D^n(x_i, x_j) = \sqrt[n]{\sum_{k=1}^{K}(x_{ik} - x_{jk})^n}$
  - Also: cosine similarity, edit distance, etc.
- (typically, you should standardize your features)

k-Means Clustering

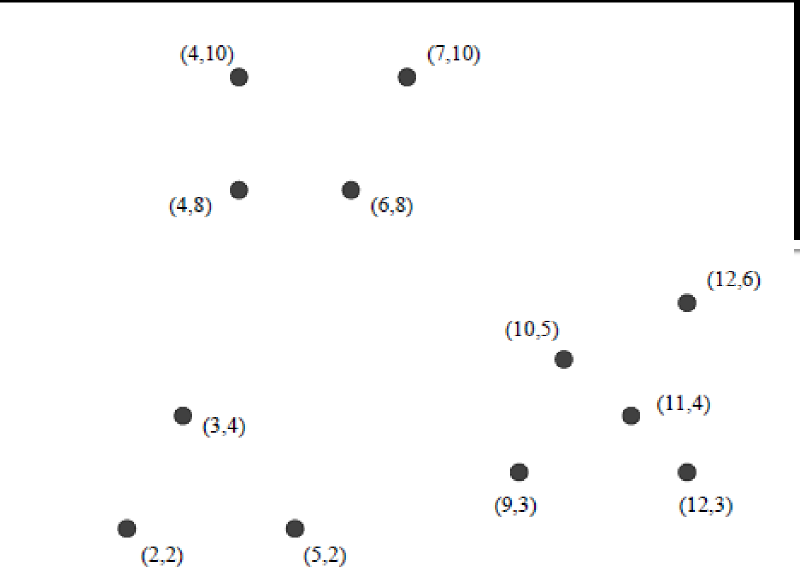
- We want to assign all points/observations to K clusters, where K is determined a priori
- Each cluster has a centroid
- Loss function:

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \|x_i - \mu_k\|_2^2$$

(sum of squared distances between each point and the centroid to which it is assigned)



k-Means Algorithm



- Algorithm to minimize loss:

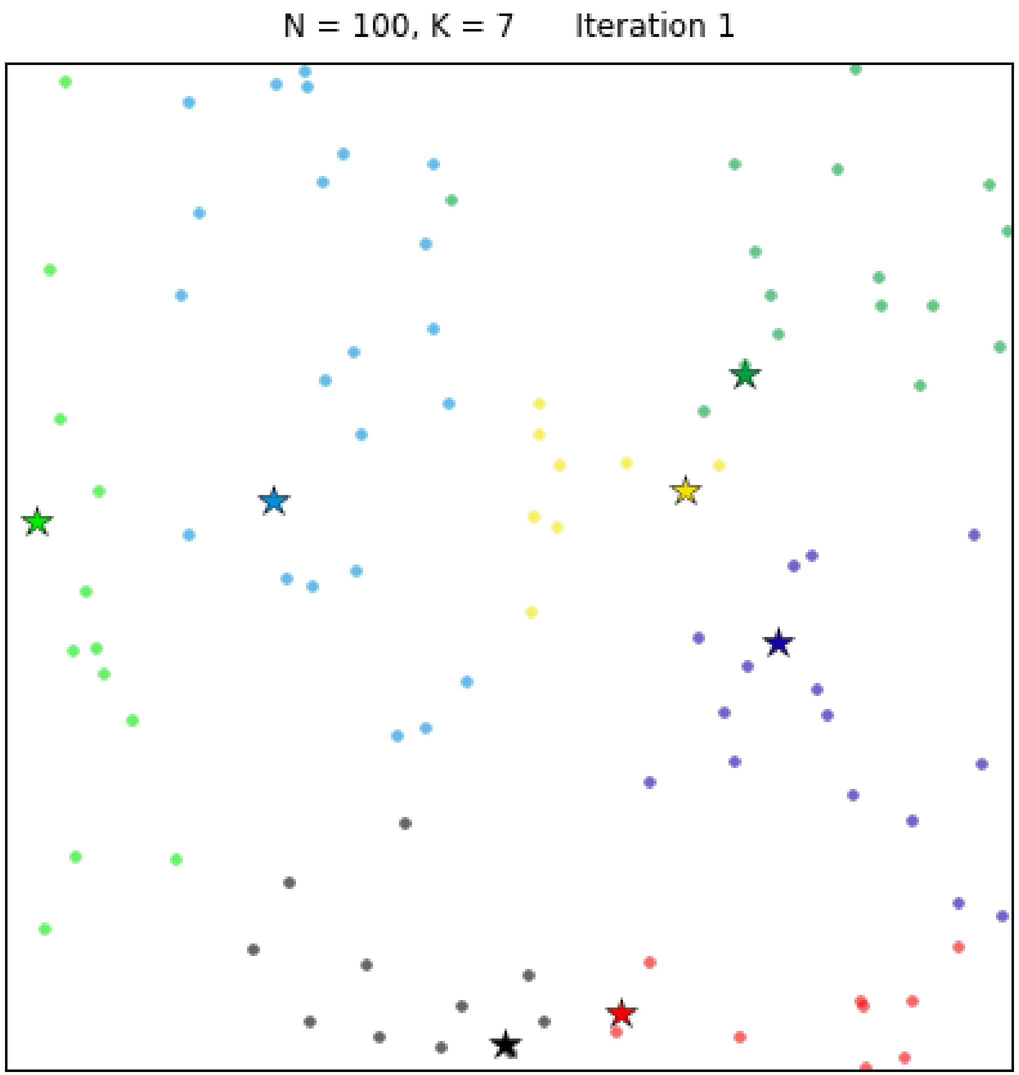
1. Choose centroids μ_k
2. Assign each data point to nearest centroid
3. Re-align centroid to center of mass
4. Repeat steps 2+3 until complete

- This process will converge to a *local* minimum

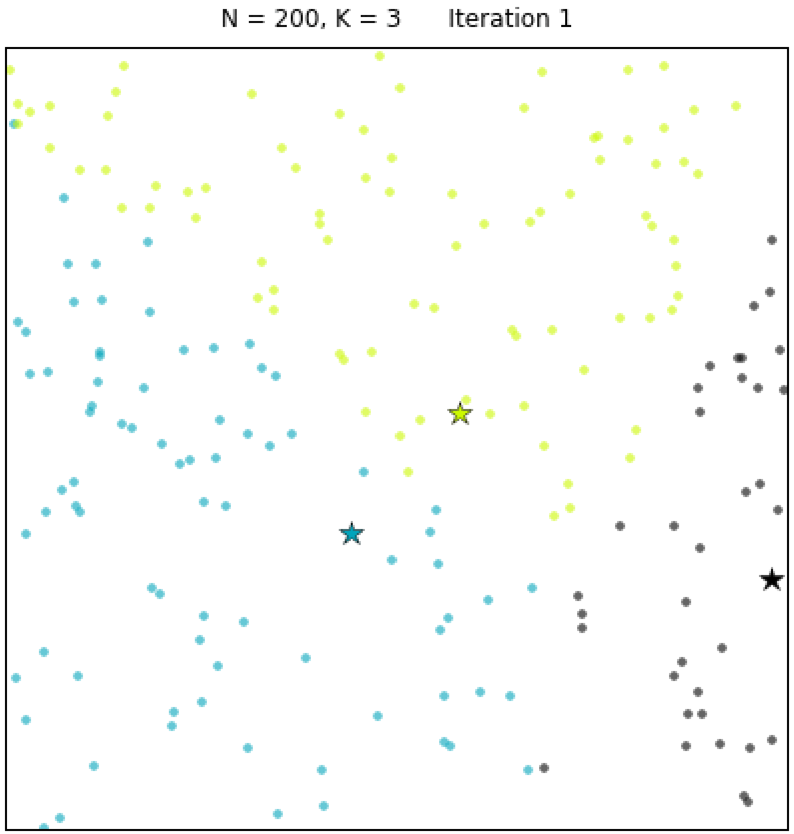
```
Initially choose k points that are likely to be in
    different clusters;
Make these points the centroids of their clusters;
FOR each remaining point p DO
    find the centroid to which p is closest;
    Add p to the cluster of that centroid;
    Adjust the centroid of that cluster to account for p;
END;
```

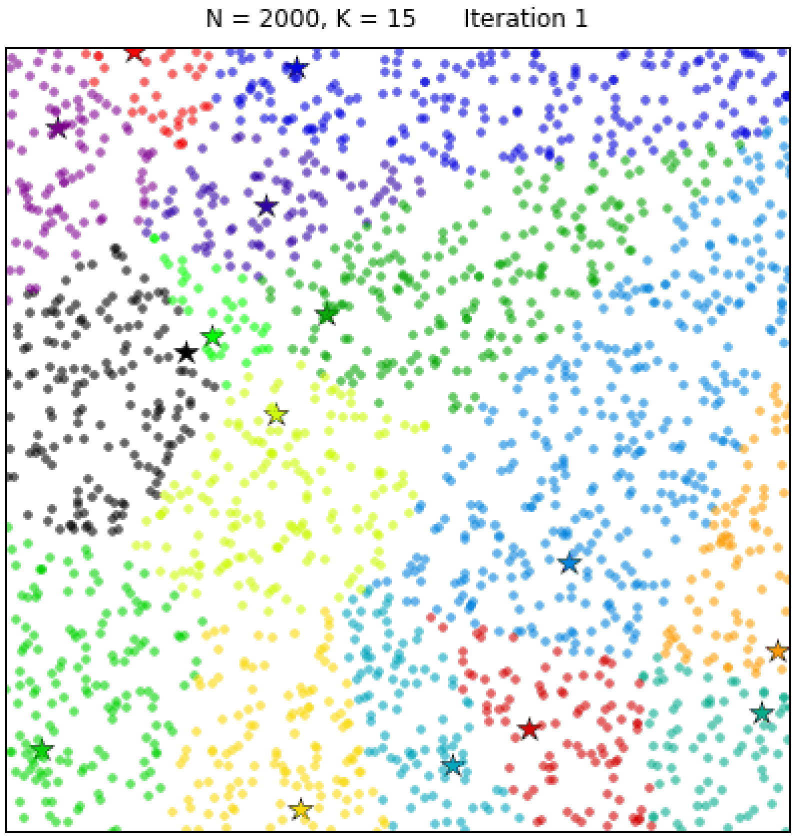
- What's it look like?

k-Means Clustering



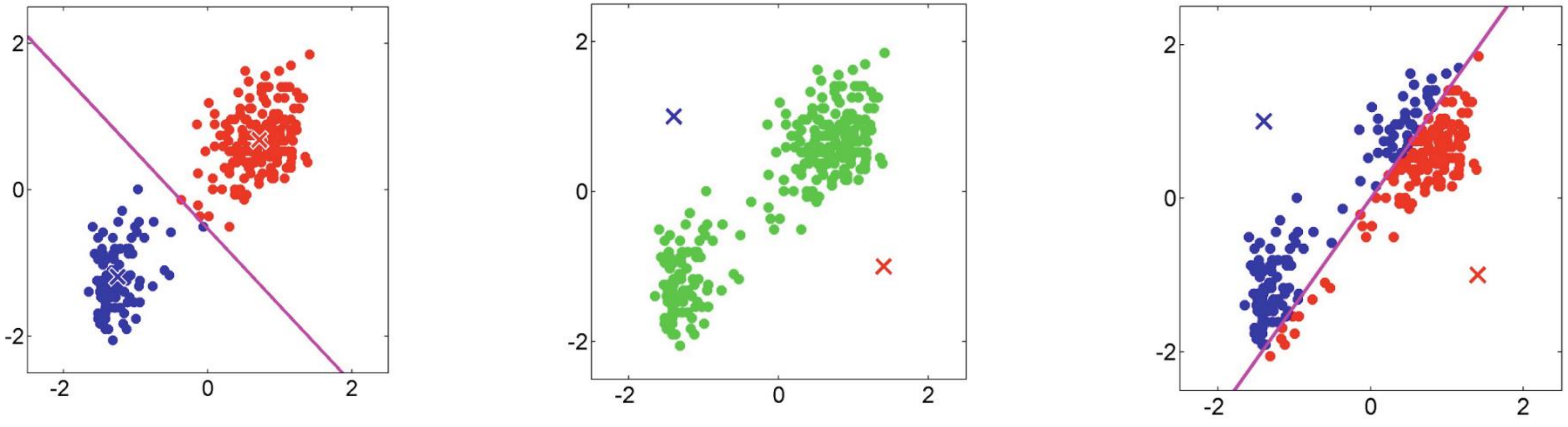
k-Means Clustering





k-Means: Details

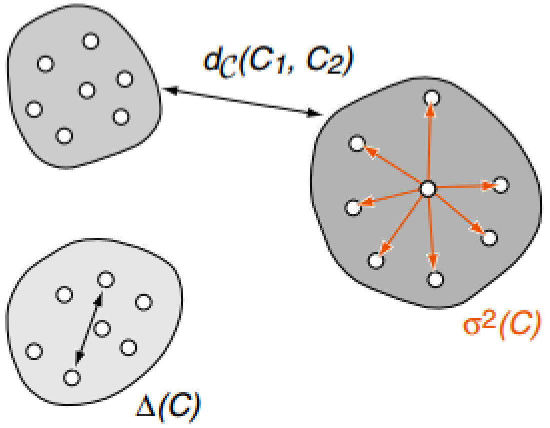
- Which distance function to use?
  - As with kNN, no one-size-fits-all answer
- How to initialize clusters?
  - Place randomly (in space, or choose from points)
  - Choose far apart



k-Means: Details

- How many clusters?
  - Structural knowledge important
  - Loss will decrease as *k* increases
- Cross-Validation?
  - Cross-validation: train on part of data, evaluate performance on test set
  - Requires some external measure of performance!
  - i.e., what fraction of points ended up in “correct” cluster?
    - Mutual Information: measure of information shared between a clustering and a ground-truth classification

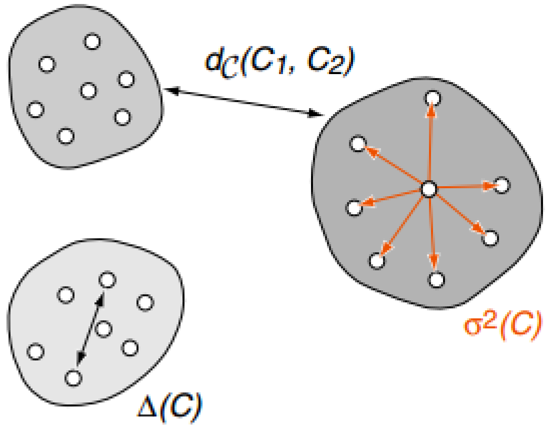
k-Means: Details



- What if there is no external measure of performance?
  - Note: this is the most common situation!
- Intuitively, a few goals:
  - Minimize distances within clusters
    - Intra-cluster correlation (average distance to cluster centroid)
  - Maximize distances between clusters
    - Relates to the "linkage" function
  - Maximize stability
    - Cluster stability: How much do partitions change with different sub-samples of data?

k-Means: Details

- Davies-Bouldin Index: $$DB = \frac{1}{n}\sum_{i=1}^{n}\max_{i \neq j}\left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right)$$
  - c_i is centroid of cluster i
  - σ_i is avg. distance of elements in cluster i to c_i
  - $d(c_i, c_j)$ is distance between c_i and c_j
  - Pop quiz: minimize DB or maximize DB?



- See also (Hastie et al): Gap Statistic, Dunn index, edge correlation, silhouette scores, elbow criteria, expected density, Hopkins statistic

k-Means: Perspective

- Pros
  - Fast, reasonable approximation for spherical data
  - Intuitive, guaranteed to converge
  - Each point assigned to exactly one cluster
- Cons
  - Each point assigned to exactly one cluster
    - Assignment can be sensitive to *k*, initialization
    - Clusters can be sensitive to outliers
  - Clusters must be spherical
  - Choice of *k* is not always apparent

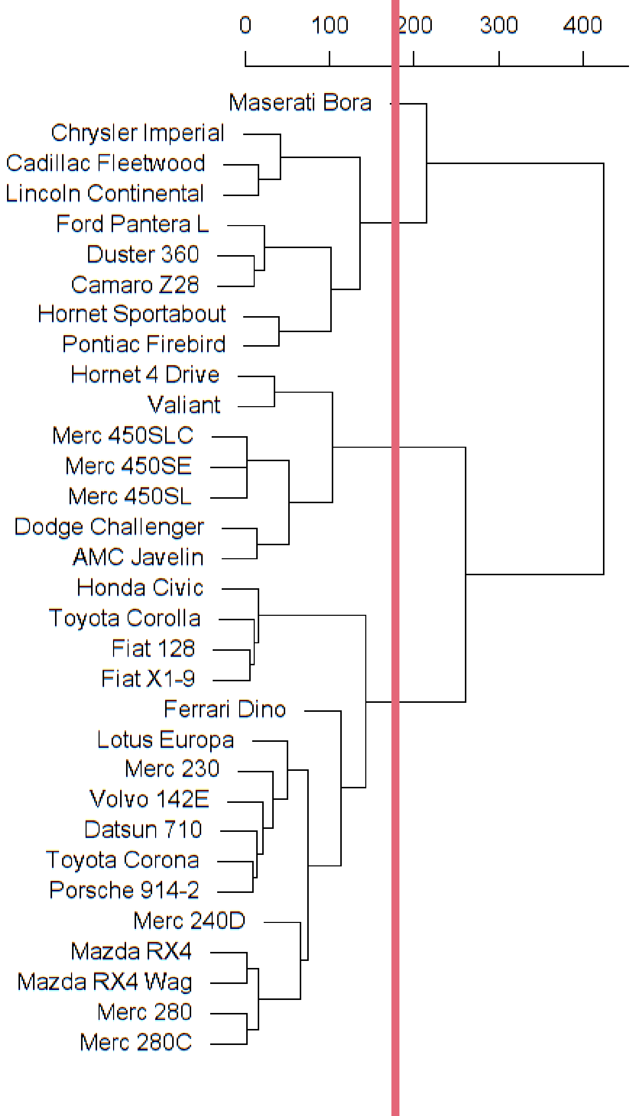
Hierarchical Clustering

- "Bottom up" (agglomerative) approach
  - Groups are merged that have smallest inter-cluster distance
- Doesn't require *k*
- Basic procedure:

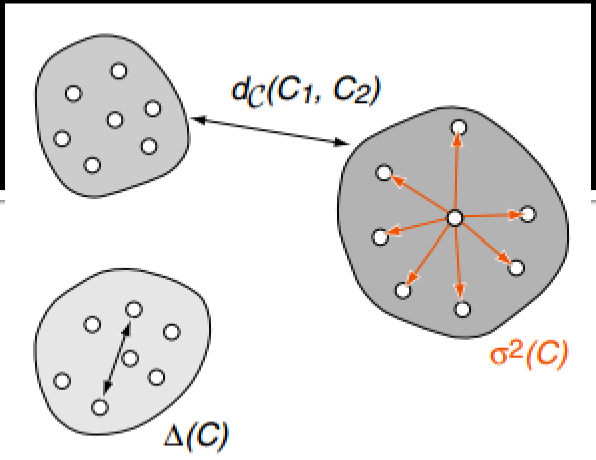
```
WHILE it is not time to stop DO
    pick the best two clusters to merge;
    combine those two clusters into one cluster;
END;
```

Hierarchical Clustering

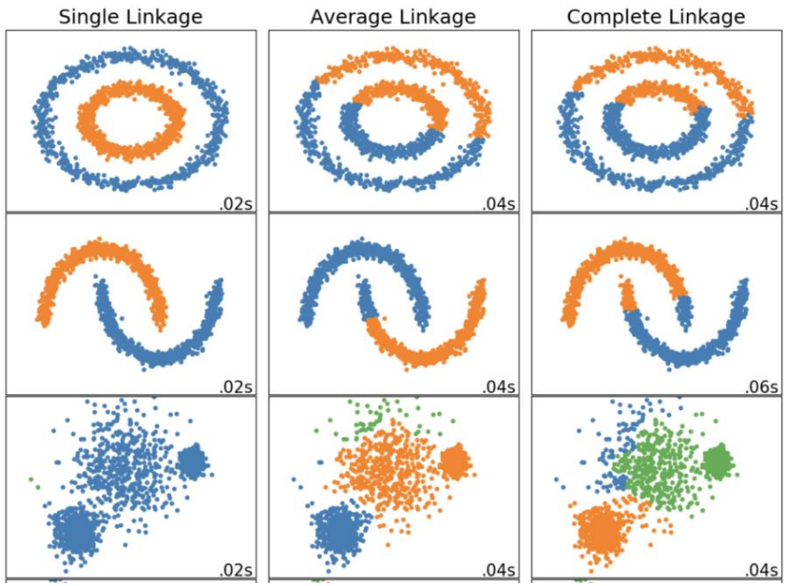
- Creates a hierarchy
  - Represented as a "dendrogram"
- "Cut point" creates clusters
  - e.g. to fit a given *k*
  - e.g., based on natural divisions (such as when inter-cluster distance exceeds a threshold)



Hierarchical Clustering



- Important considerations:
  - Inter-point distances and inter-cluster distance
    - Average linkage
      - Distance between clusters is average of all point distances
    - Complete (maximum) linkage:
      - Intra-C distance measured betw/ two points farthest apart
      - Creates spherical clusters
    - Single (minimum) linkage
      - Intra-C distance measured between two closest points
      - Creates extended clusters

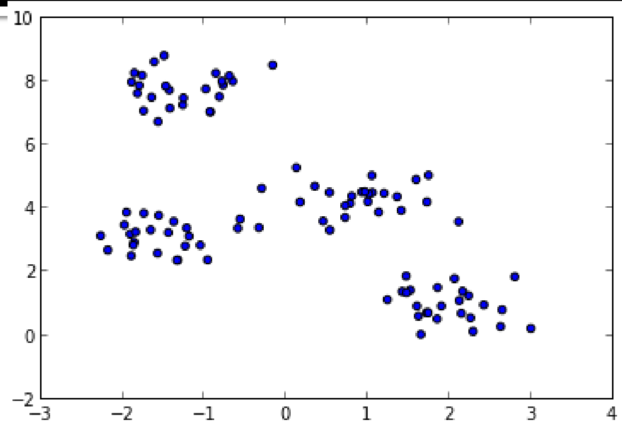


Cluster analysis: Summary

- A common method for finding groups in data
  - Useful for discovering structure
  - Very intuitive, easy to implement, scalable
- Key modeling parameters
  - Algorithm choice: *k*-means, hierarchical, …
  - Distance metric (and feature space!)
  - Linkage function
  - Global optimization criteria (e.g. ICC)

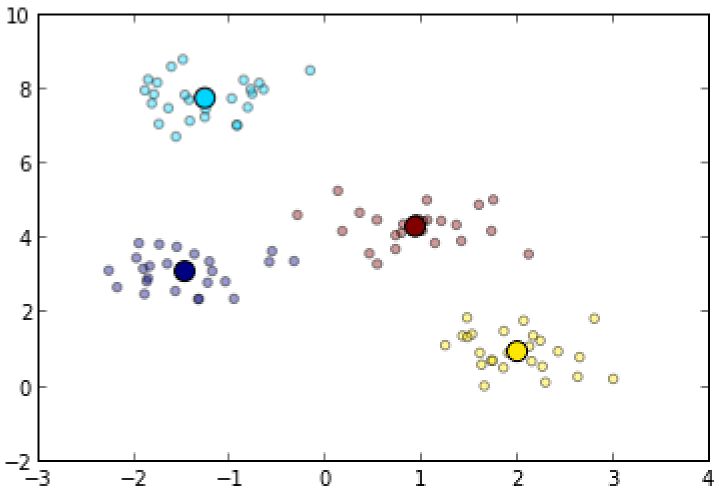
Cluster analysis in Python

```
import matplotlib
import matplotlib.pyplot as plt
import sklearn.datasets as datasets
plt.jet() # set the color map

# create a dataset
X, Y = datasets.make_blobs(centers=4, cluster_std=0.5, random_state=0)

# see what it looks like
plt.scatter(X[:,0], X[:,1]);
```




```
from sklearn.cluster import KMeans
kmeans = KMeans(4, random_state=8)
Y_hat = kmeans.fit(X).labels_

plt.scatter(X[:,0], X[:,1], c=Y_hat, alpha=0.4)
mu = kmeans.cluster_centers_
plt.scatter(mu[:,0], mu[:,1], s=100, c=np.unique(Y_hat))
print mu
```

```
[[-1.47935679  3.11716896]
 [-1.26811733  7.76378266]
 [ 1.99186903  0.96561071]
 [ 0.92578447  4.32475792]]
```



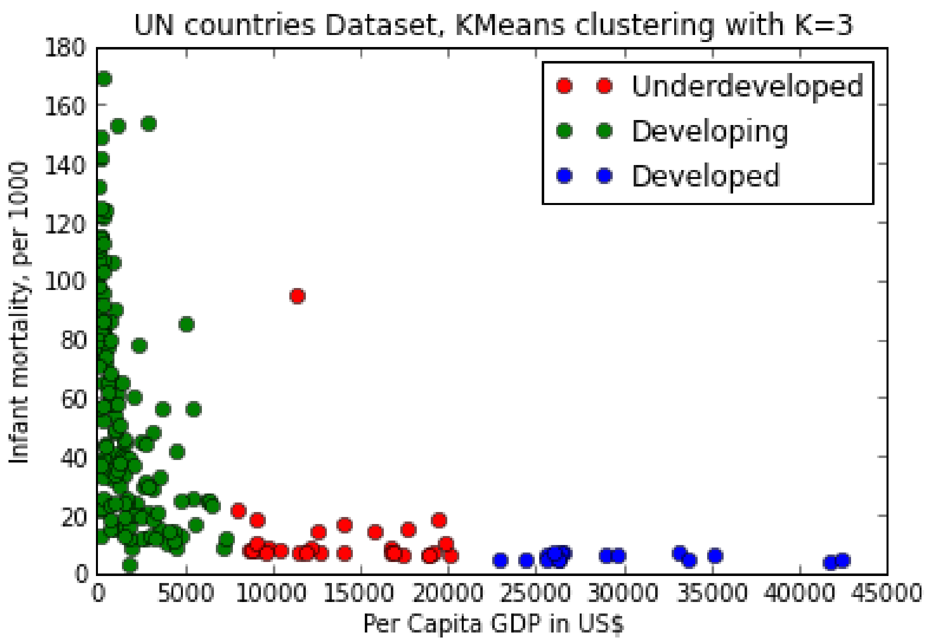
Cluster analysis in Python

```python
import pandas as pd
df = pd.read_csv('../datasets/UN.csv')
print('----')
# print the raw column information plus summary header
print(df)
print('----')
# look at the types of each column explicitly
print('Individual columns - Python data types')
[(x, type(df[x][0])) for x in df.columns]
```

```
----
<class 'pandas.core.frame.DataFrame'>
Int64Index: 207 entries, 0 to 206
Data columns (total 14 columns):
country                   207  non-null values
region                    207  non-null values
tfr                       197  non-null values
contraception             144  non-null values
educationMale             76  non-null values
educationFemale           76  non-null values
lifeMale                  196  non-null values
lifeFemale                196  non-null values
infantMortality           201  non-null values
GDPperCapita              197  non-null values
economicActivityMale      165  non-null values
```

Cluster analysis in Python

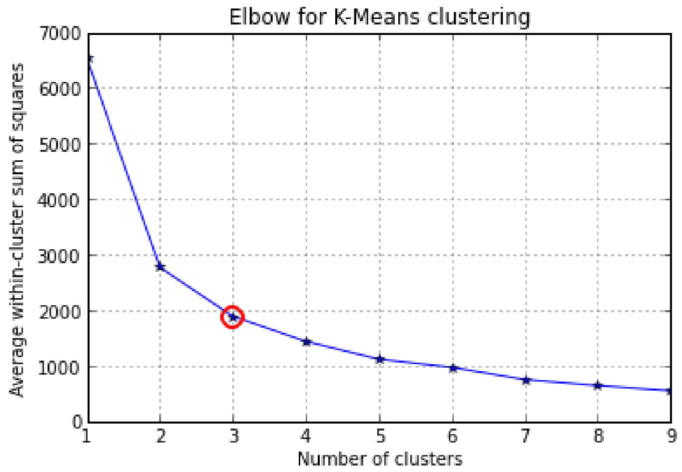
```
from sklearn.cluster import KMeans
km = KMeans(3)                          # initialize
km.fit(X)
c = km.predict(X)                       # classify into three clusters

import kmeans as mykm                   # loads some helper code
# plot column 3 (GDP), vs column 2 (infant mortality)
(pl0,pl1,pl2) = mykm.plot_clusters(X,c,3,2)
```



Cluster analysis in Python

```
import numpy as np
from scipy.cluster.vq import kmeans,vq
from scipy.spatial.distance import cdist
import matplotlib.pyplot as plt

K = range(1,10)

KM = [kmeans(X,k) for k in K]                          # apply kmeans 1 to 10
centroids = [cent for (cent,var) in KM]                # get cluster centroids

D_k = [cdist(X, cent, 'euclidean') for cent in centroids]   # compute distance matrix

cIdx = [np.argmin(D,axis=1) for D in D_k]              # assign points to nearest centroid
dist = [np.min(D,axis=1) for D in D_k]                 # get distance to centroids

avgWithinSS = [sum(d)/X.shape[0] for d in dist]
```

```
kIdx = 2
# plot elbow curve
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(K, avgWithinSS, 'b*-')
ax.plot(K[kIdx], avgWithinSS[kIdx], marker='o', markersize=12,
      markeredgewidth=2, markeredgecolor='r', markerfacecolor='None')
plt.grid(True)
plt.xlabel('Number of clusters')
plt.ylabel('Average within-cluster sum of squares')
tt = plt.title('Elbow for K-Means clustering')
```



Cluster analysis in Python

```
from scipy.spatial.distance import pdist, squareform
from scipy.cluster.hierarchy import linkage, dendrogram
countries = (2,7,8,9,10,12,13,14,15,16,18,19,21,23,24,25,26,27,28,29,171,172,194,195)
print df.ix[countries,(0,2,9,10)]
```

```
          country   tfr  GDPperCapita  economicActivityMale
2         Algeria  3.81          1531                  76.4
7       Argentina  2.62          8055                  76.2
8         Armenia  1.70           354                  65.0
9       Australia  1.89         20046                  74.0
10        Austria  1.42         29006                  69.5
12        Bahamas  1.95         12545                  81.2
13        Bahrain  2.97          9073                  88.2
14     Bangladesh  3.14           280                  88.8
15       Barbados  1.73          7173                  73.4
16        Belarus  1.40           994                  76.4
18         Belize  3.66          2569                  79.0
19          Benin  5.83           391                  90.0
21        Bolivia  4.36           909                  74.1
23       Botswana  4.45          3640                  75.4
24         Brazil  2.17          4510                  84.0
25         Brunei  2.70         16683                  82.2
```

Cluster analysis in Python

```
features = df.ix[countries,(2,9,10)]
names = df.ix[countries,(0)].tolist()

data_dist = pdist(normalize(features))          # compute distance matrix
data_link = linkage(data_dist)                  # compute cluster linkages

dendrogram(data_link,labels=names, color_threshold=1, leaf_rotation=90)
plt.xlabel('Countries')
plt.ylabel('Distance')
```