Unsupervised methods (clustering)

INFO 523 - Lecture 10

Dr. Greg Chism

One main question for this and the next lecture:

What is modelling?

Mapping problems to machine learning tasks

- Predicting what customers might buy, based on past transactions
- Identifying fraudulent transactions
- Determining price elasticity (the rate at which a price increase will decrease sales, and vice versa) of various products or product classes
- Determining the best way to present product listings when a customer searches for an item
- Customer segmentation: grouping customers with similar purchasing behavior
- AdWord valuation: how much the company should spend to buy certain AdWords on search engines
- Determine which variable has larger effects on a particular outcome

Solving classification problems

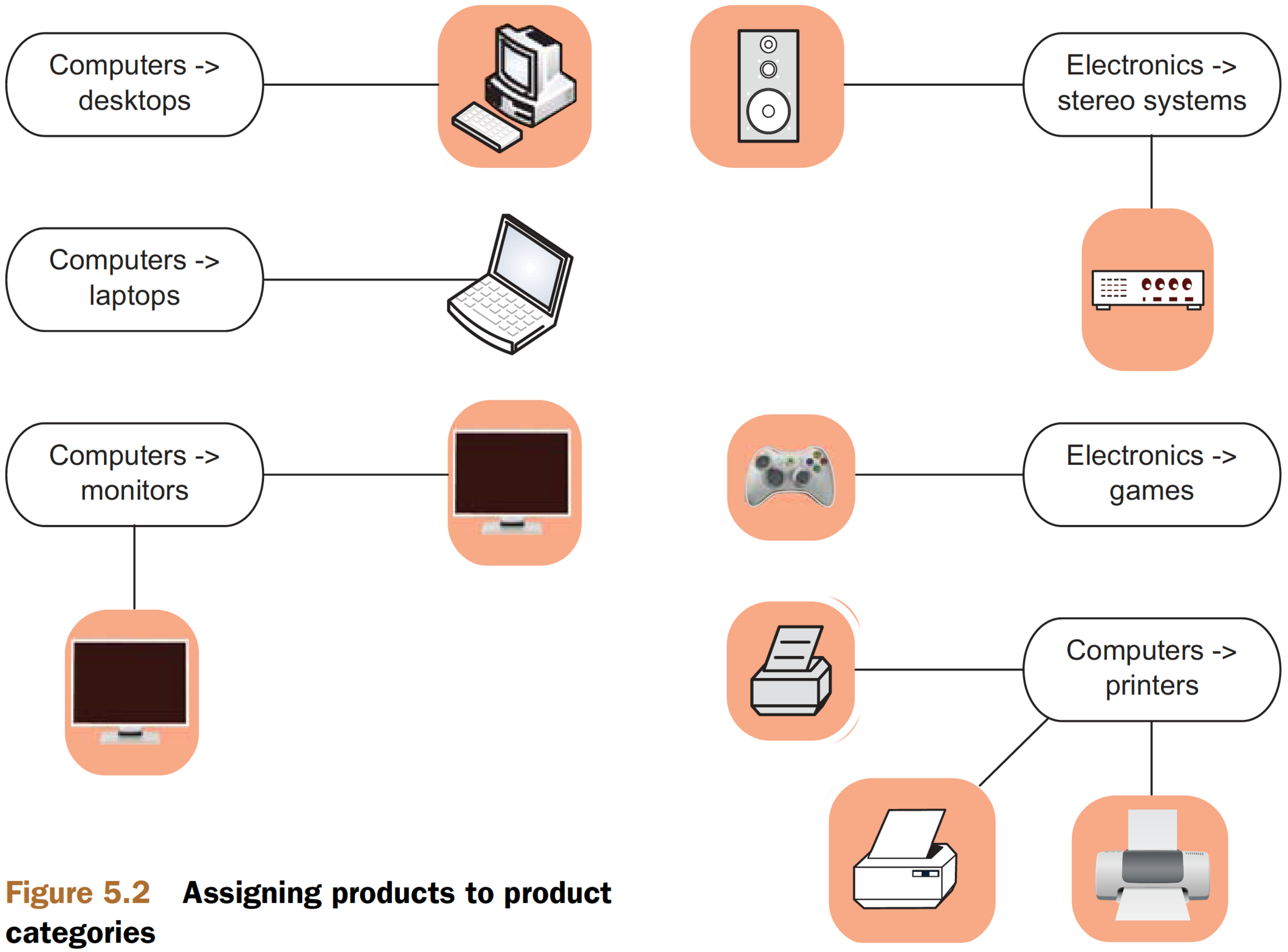


Figure 5.2 Assigning products to product categories

Solving classification problems

- Classification itself is an example of what is called *supervised learning*.
- Multicategory vs. two-category classification
  - using binary classifiers to solve multicategory problems : building one classifier for each category (a “one versus rest” classifier)
  - in most cases a suitable multiple-category implementation better than multiple binary classifiers: using the package *mlogit* instead of the base method *glm*() for logistic regression

Solving classification problems

- Some common classification methods
  - Naive Bayes : a good first attempt at solving the product categorization problem.
  - Decision trees : an important extension of decision trees - random forests
  - Logistic regression : estimate class probabilities (the probability that an object is in a given class) in addition to class assignments
  - Support vector machines : SVMs make fewer assumptions about variable distribution than do many other methods

Solving scoring problems

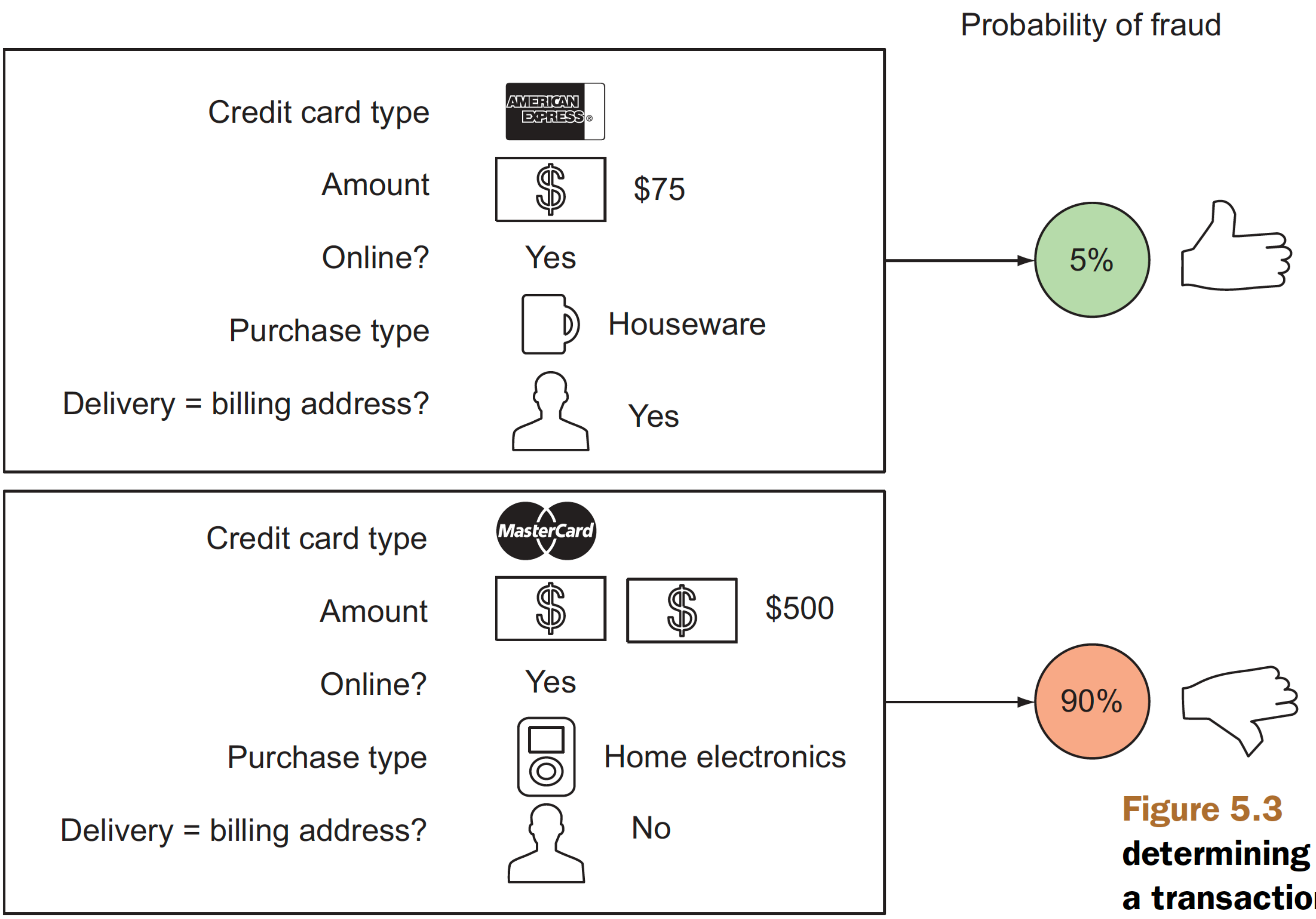


Figure 5.3 Notional example of determining the probability that a transaction is fraudulent

Solving scoring problems

- Linear regression
  - Linear regression builds a model such that the predicted numerical output is a linear additive function of the inputs.
  - a good first model to try when trying to predict a numeric value
- Logistic regression
  - Logistic regression always predicts a value between 0 and 1
  - Ie, if what you want to estimate is the probability that a given transaction is fraudulent or legitimate.

Working without known targets

- *unsupervised learning*: there's not (yet) a specific outcome that you want to predict
  - discover similarities and relationships in the data. rather than predicting outputs based on inputs
- Common clustering methods
  - K-means clustering
  - A priori algorithm for finding association rules
  - Nearest neighbor

WHEN TO USE BASIC CLUSTERING

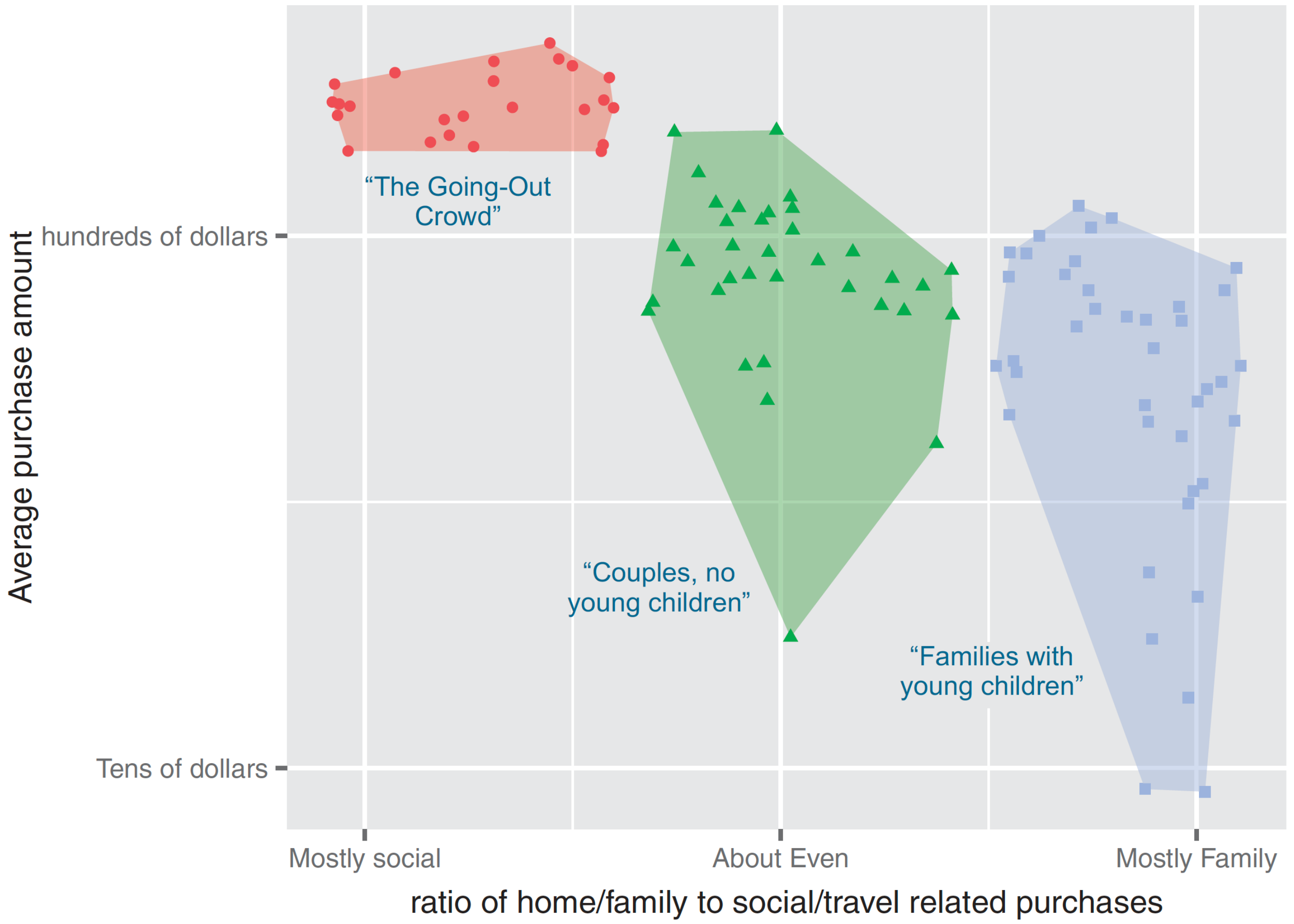
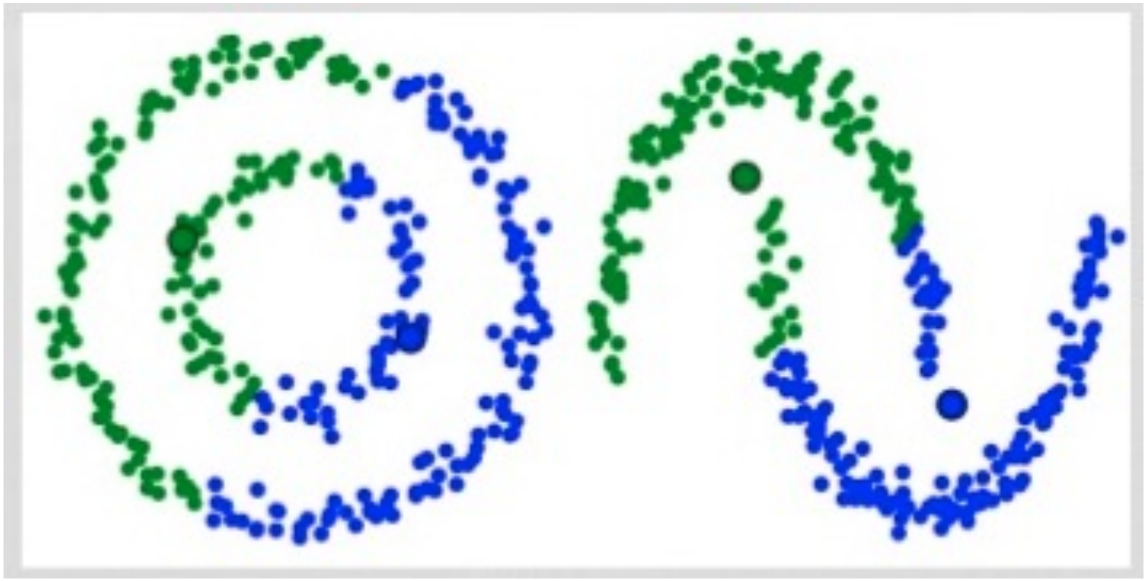


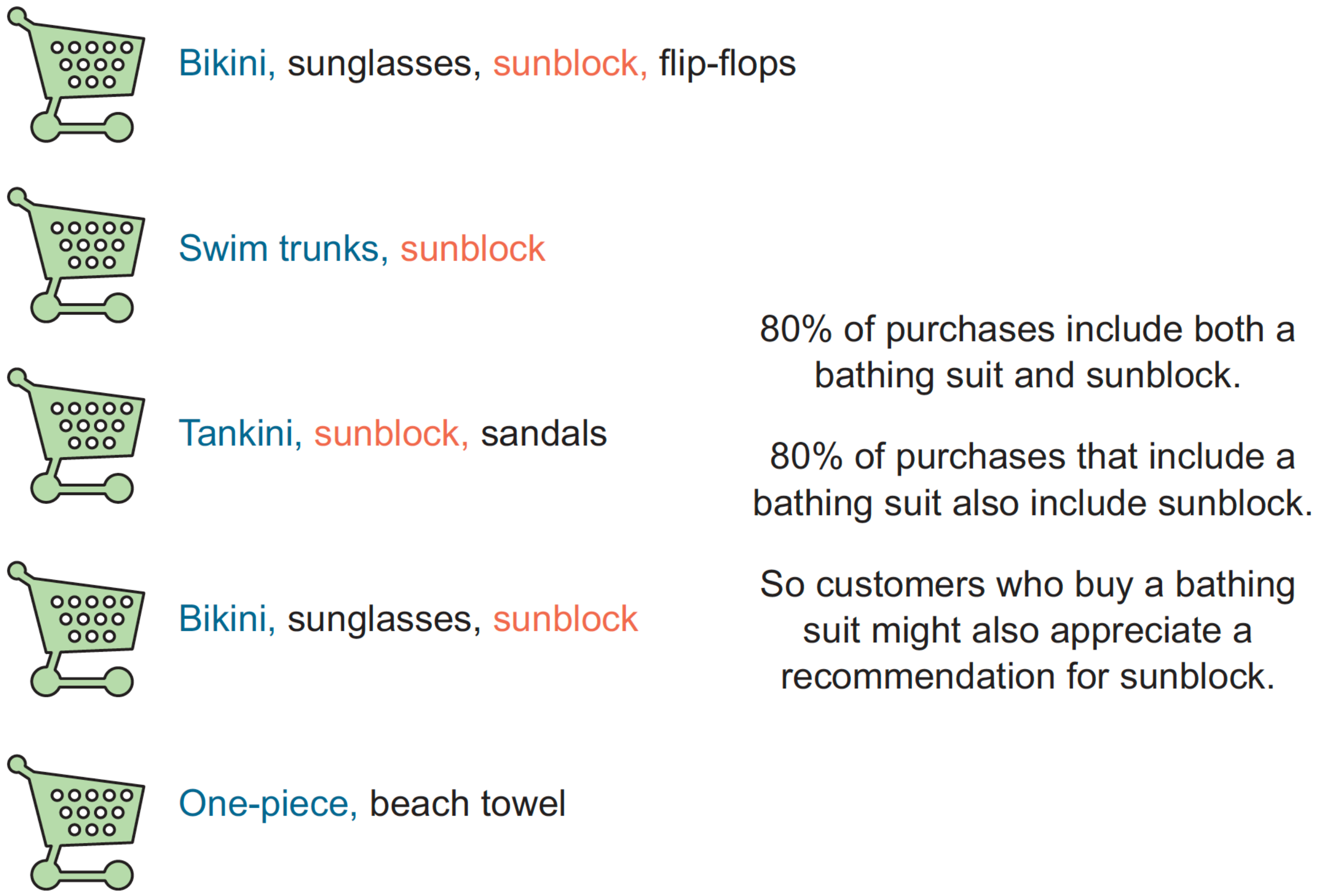
Figure 5.4 **Notional example of clustering your customers by purchase pattern and purchase amount**

WHEN TO USE BASIC CLUSTERING



Sometimes basic clustering algorithms (k-means) fail…

WHEN TO USE ASSOCIATION RULES


Bikini, sunglasses, sunblock, flip-flops
Swim trunks, sunblock
Tankini, sunblock, sandals
Bikini, sunglasses, sunblock
One-piece, beach towel

80% of purchases include both a bathing suit and sunblock.

80% of purchases that include a bathing suit also include sunblock.

So customers who buy a bathing suit might also appreciate a recommendation for sunblock.

Figure 5.5 Notional example of finding purchase patterns in your data

WHEN TO USE NEAREST NEIGHBOR METHODS

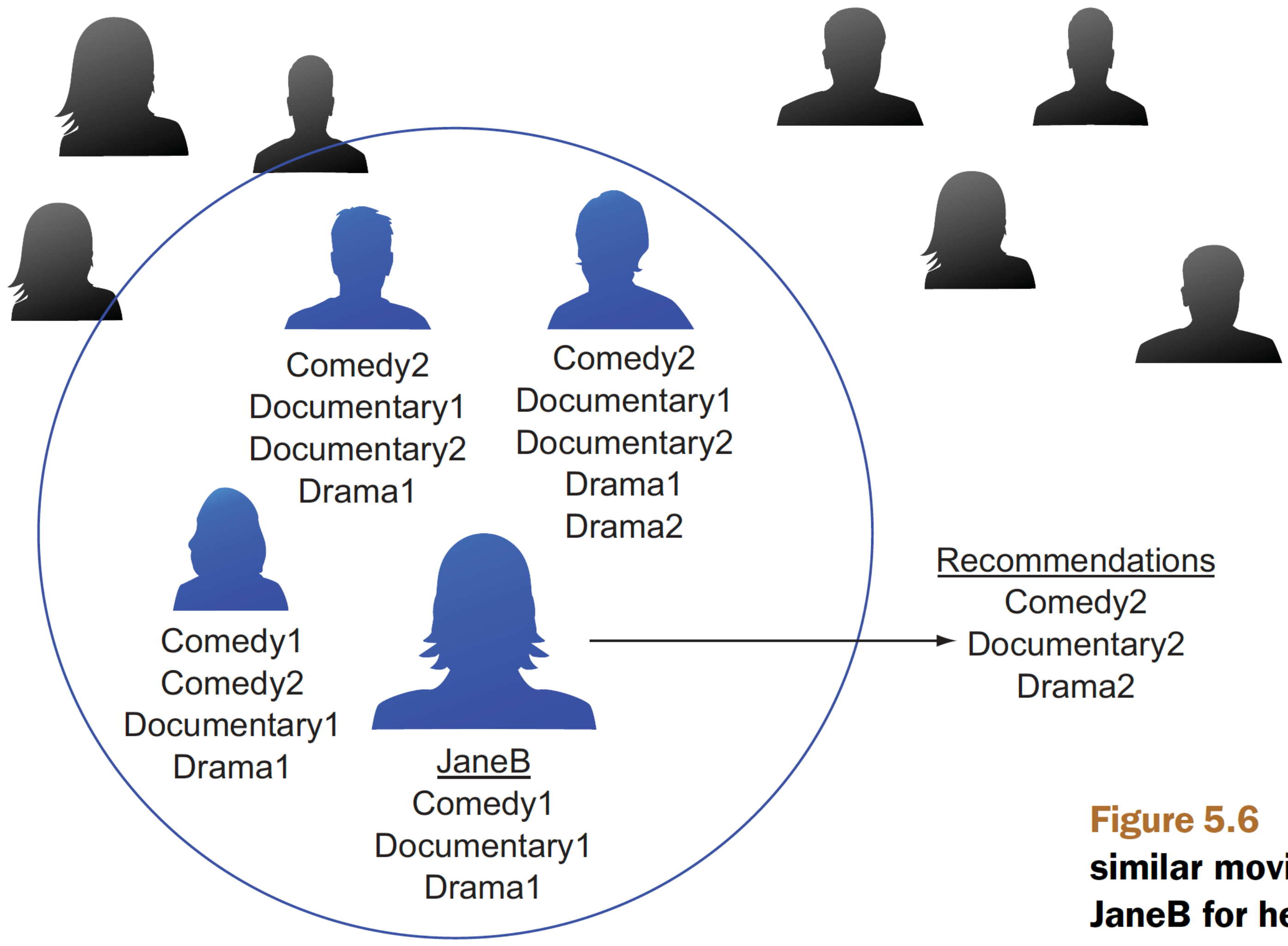


Figure 5.6 Look to the customers with similar movie-watching patterns as JaneB for her movie recommendations.

Unsupervised methods

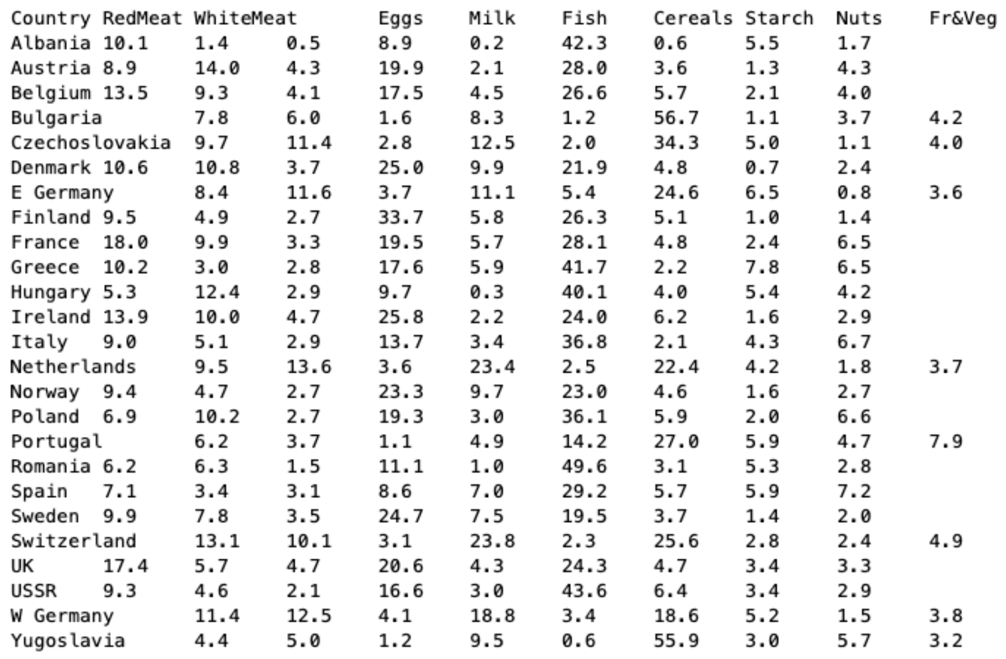
- Methods to discover unknown relationships in data…Central to data mining!
- Two main classes of unsupervised methods
  - *Cluster analysis*: finds groups in your data with similar characteristics
    - Hierarchical clustering
    - k-means
  - *Association rule*: mining finds elements or properties in the data that tend to occur together.

Distances – Many indexes...

- Euclidean distance (length of the line segment between points, L2 distance)
  - edist(x, y) <- sqrt((x[1]-y[1])^2 + (x[2]-y[2])^2 + ...)
  - When all the data is real-valued (quantitative)
- Hamming distance
  - hdist(x, y) <- sum((x[1] != y[1]) + (x[2] != y[2]) + ...)
  - For categorical and binary variables (male /female , or small /medium /large)
- Manhattan (city block) distance : L1 distance
  - mdist(x, y) <- sum(abs(x[1]-y[1]) + abs(x[2]-y[2]) + ...)
  - Number of horizontal and vertical units from one point to the other.
- Cosine similarity
  - dot(x, y) <- sum(x[1]*y[1] + x[2]*y[2] + ...)
  - cossim(x, y) <- dot(x, y)/(sqrt(dot(x,x)*dot(y,y)))
  - 1 - 2*acos(cossim(x,y))/pi)

Let’s go over a worked example!

- A small dataset from 1973 on protein consumption from nine different food groups in 25 countries in Europe => group the countries based on patterns in their protein consumption.
- protein <- read.table(,…)
- What should you do first?
  - vars.to.use <- colnames(protein)[-1]
  - pmatrix <- scale(protein[,vars.to.use])
  - pcenter <- attr(pmatrix, "scaled:center")
  - pscale <- attr(pmatrix, "scaled:scale")



```
Country RedMeat WhiteMeat       Eggs    Milk    Fish    Cereals Starch  Nuts    Fr&Veg
Albania 10.1    1.4     0.5     8.9     0.2     42.3    0.6     5.5     1.7
Austria 8.9     14.0    4.3     19.9    2.1     28.0    3.6     1.3     4.3
Belgium 13.5    9.3     4.1     17.5    4.5     26.6    5.7     2.1     4.0
Bulgaria        7.8     6.0     1.6     8.3     1.2     56.7    1.1     3.7     4.2
Czechoslovakia  9.7     11.4    2.8     12.5    2.0     34.3    5.0     1.1     4.0
Denmark 10.6    10.8    3.7     25.0    9.9     21.9    4.8     0.7     2.4
E Germany       8.4     11.6    3.7     11.1    5.4     24.6    6.5     0.8     3.6
Finland 9.5     4.9     2.7     33.7    5.8     26.3    5.1     1.0     1.4
France  18.0    9.9     3.3     19.5    5.7     28.1    4.8     2.4     6.5
Greece  10.2    3.0     2.8     17.6    5.9     41.7    2.2     7.8     6.5
Hungary 5.3     12.4    2.9     9.7     0.3     40.1    4.0     5.4     4.2
Ireland 13.9    10.0    4.7     25.8    2.2     24.0    6.2     1.6     2.9
Italy   9.0     5.1     2.9     13.7    3.4     36.8    2.1     4.3     6.7
Netherlands     9.5     13.6    3.6     23.4    2.5     22.4    4.2     1.8     3.7
Norway  9.4     4.7     2.7     23.3    9.7     23.0    4.6     1.6     2.7
Poland  6.9     10.2    2.7     19.3    3.0     36.1    5.9     2.0     6.6
Portugal        6.2     3.7     1.1     4.9     14.2    27.0    5.9     4.7     7.9
Romania 6.2     6.3     1.5     11.1    1.0     49.6    3.1     5.3     2.8
Spain   7.1     3.4     3.1     8.6     7.0     29.2    5.7     5.9     7.2
Sweden  9.9     7.8     3.5     24.7    7.5     19.5    3.7     1.4     2.0
Switzerland     13.1    10.1    3.1     23.8    2.3     25.6    2.8     2.4     4.9
UK      17.4    5.7     4.7     20.6    4.3     24.3    4.7     3.4     3.3
USSR    9.3     4.6     2.1     16.6    3.0     43.6    6.4     3.4     2.9
W Germany       11.4    12.5    4.1     18.8    3.4     18.6    5.2     1.5     3.8
Yugoslavia      4.4     5.0     1.2     9.5     0.6     55.9    3.0     5.7     3.2
```

Hierarchical clustering

Euclidean distance (L1 distance)

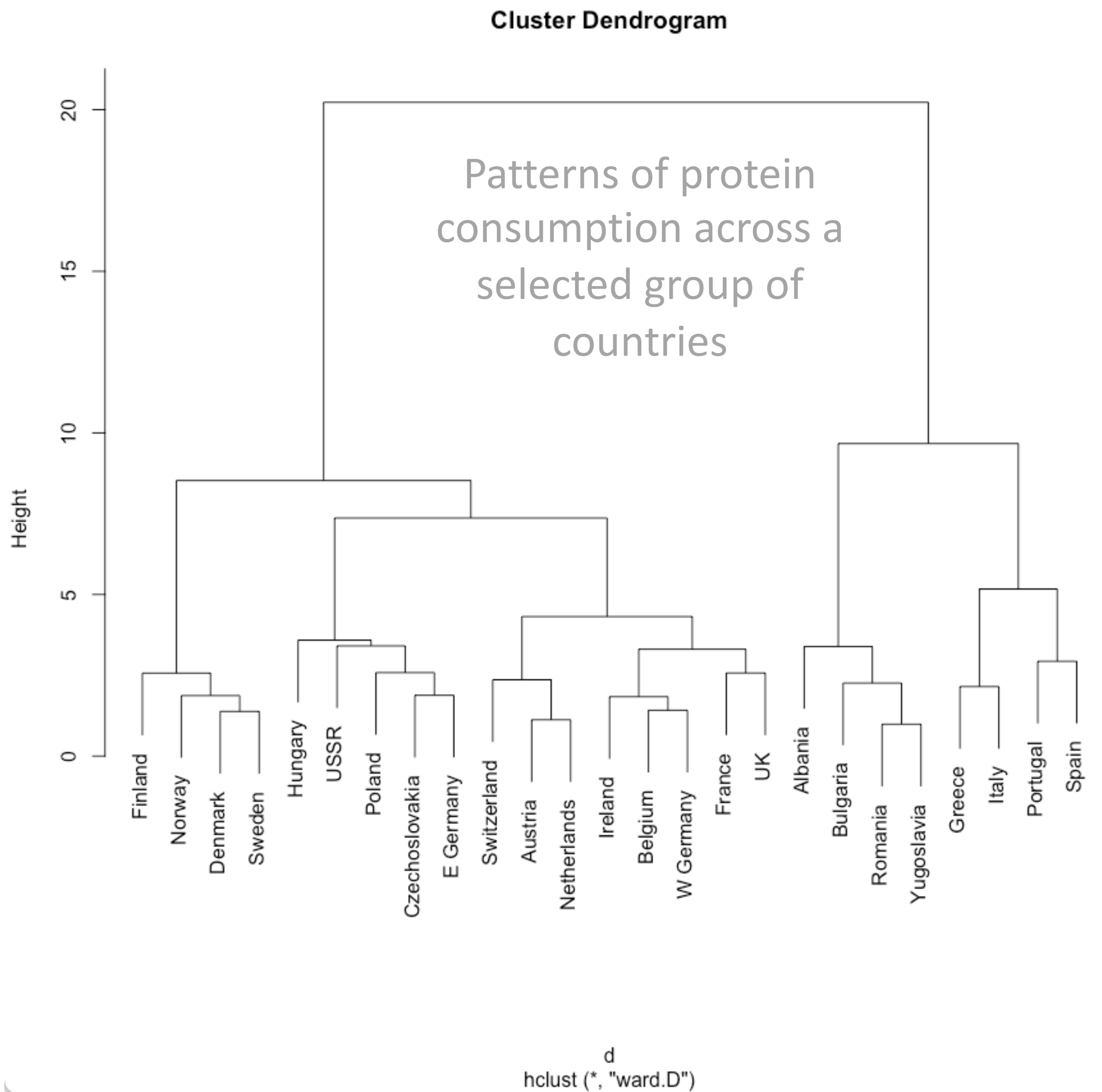
Scaled/centered matrix!

- d <- dist(pmatrix, method="euclidean")
- pfit <- hclust(d, method="ward.D")
- plot(pfit, labels=protein$Country)

Ward's minimum variance: compact and spherical clusters

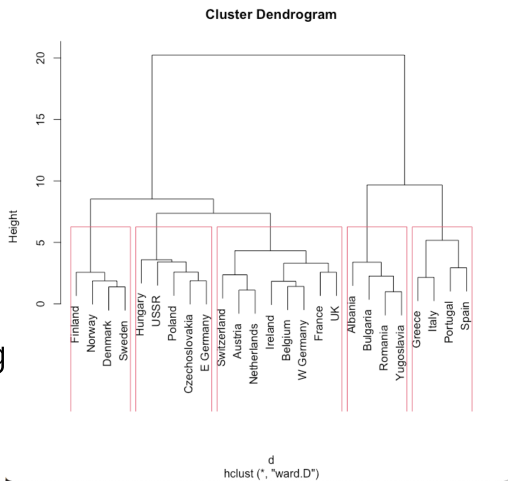
Hierarchical clustering

Distance metric: Euclidean
Cluster algorithm: Ward's



Hierarchical clustering

- The dendrogram suggests five clusters (or more? Or less? Let's talk about that later!) = > draw the rectangles on the dendrogram
  - rect.hclust(pfit, k=5)



- Extracting the clusters found by hclust()

```
groups <- cutree(pfit, k=5)
print_clusters <- function(labels, k) {
        for(i in 1:k) {
                print(paste("cluster", i))
                print(protein[labels==i,c("Country","RedMeat","Fish","Fr.Veg
        }
}
print_clusters(groups, 5)
```

VISUALIZING CLUSTERS…again, many options!

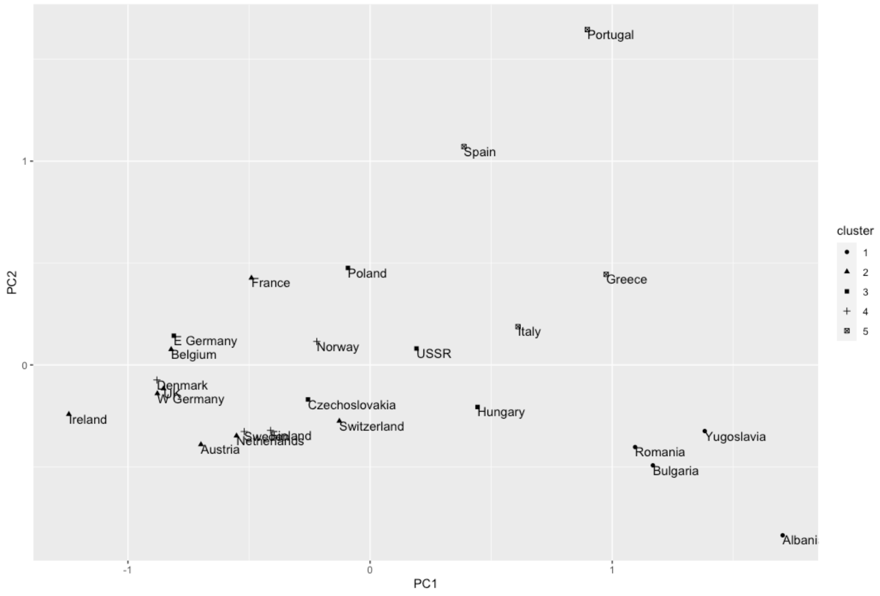
- Let's check out a PCA: prcomp()
- PCA

```
library(ggplot2)
princ <- prcomp(pmatrix)
nComp <- 2
project <- predict(princ,
newdata=pmatrix)[,1:nComp]
```

- Plot

```
project.plus <-
cbind(as.data.frame(project),
cluster=as.factor(groups),
country=protein$Country)
ggplot(project.plus, aes(x=PC1,
y=PC2)) +
    geom_point(aes(shape=cluster)) +
    geom_text(aes(label=country),
    hjust=0, vjust=1)
```

Why do we need to use the predict function here?

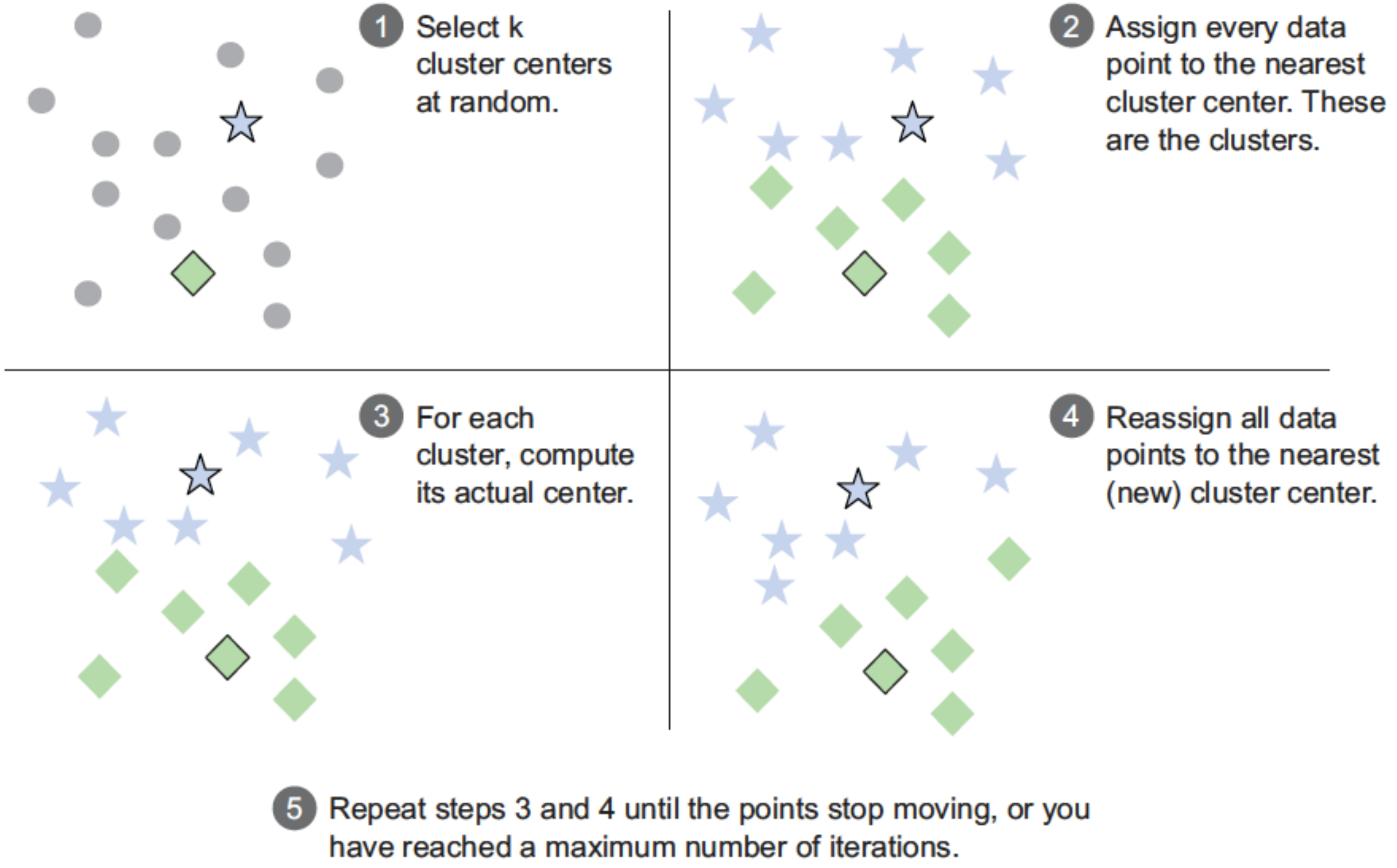


The k-means algorithm

- The data is all numeric and the distance metric is squared Euclidean (in theory, you could also run it with other distance metrics)
- Plus side…
  - easy to implement
  - can be faster than hierarchical clustering on large datasets
  - works best on data that looks like a mixture of Gaussians
- Negative side
  - must pick *k* in advance
  - fairly unstable : the final clusters depend on the initial cluster centers. => This algorithm isn't guaranteed to have a unique "final solution".
  - Not a probabilistic algorithm…

The k-means procedure

Cluster Dendrogram



1 Select k cluster centers at random.
2 Assign every data point to the nearest cluster center. These are the clusters.
3 For each cluster, compute its actual center.
4 Reassign all data points to the nearest (new) cluster center.
5 Repeat steps 3 and 4 until the points stop moving, or you have reached a maximum number of iterations.

Running k-means with k=5

```
pclusters <- kmeans(pmatrix, centers = 5, nstart=25, iter.max=100)
summary(pclusters)
pclusters$centers
pclusters$size
groups <- pclusters$cluster
print_clusters(groups, kbest.p)
```

We will generate 25 initial configurations

Whether a given cluster is *real*?

- Whether a cluster represents true structure depends on the plausible variations in the dataset…
- clusterboot() (fpc package)
  - use bootstrap resampling to evaluate how stable a given cluster is
  - Jaccard coefficient : The Jaccard similarity between two sets A and B
    - the number of elements in the intersection of A and B / the number of elements in the union of A and B.

Basic general strategy

1. Cluster the data as usual.
2. Draw a new dataset (of the same size as the original) by resampling the original dataset with replacement (meaning that some of the data points may show up more than once, and others not at all). Cluster the new dataset.
3. For every cluster in the original clustering, find the most similar cluster in the new clustering (the one that gives the maximum Jaccard coefficient) and record that value.
   - If this maximum Jaccard coefficient is less than 0.5, the original cluster is considered to be dissolved.

- Repeat steps 2–3 *many* times.
- Do NOT forget about the difference between <u>training</u> and <u>testing</u> datasets!

Cluster stability

- The cluster stability of each cluster in the original clustering is the mean value of its Jaccard coefficient over all the bootstrap iterations.
- clusters with a stability value
  - 0.6 <= : unstable.
  - 0.6 ~ 0.75 : measure a pattern in the data, but there isn't high certainty about which points should be clustered together.
  - 0.85 >= : highly stable (they're likely to be real clusters).

clusterboot() on the protein data

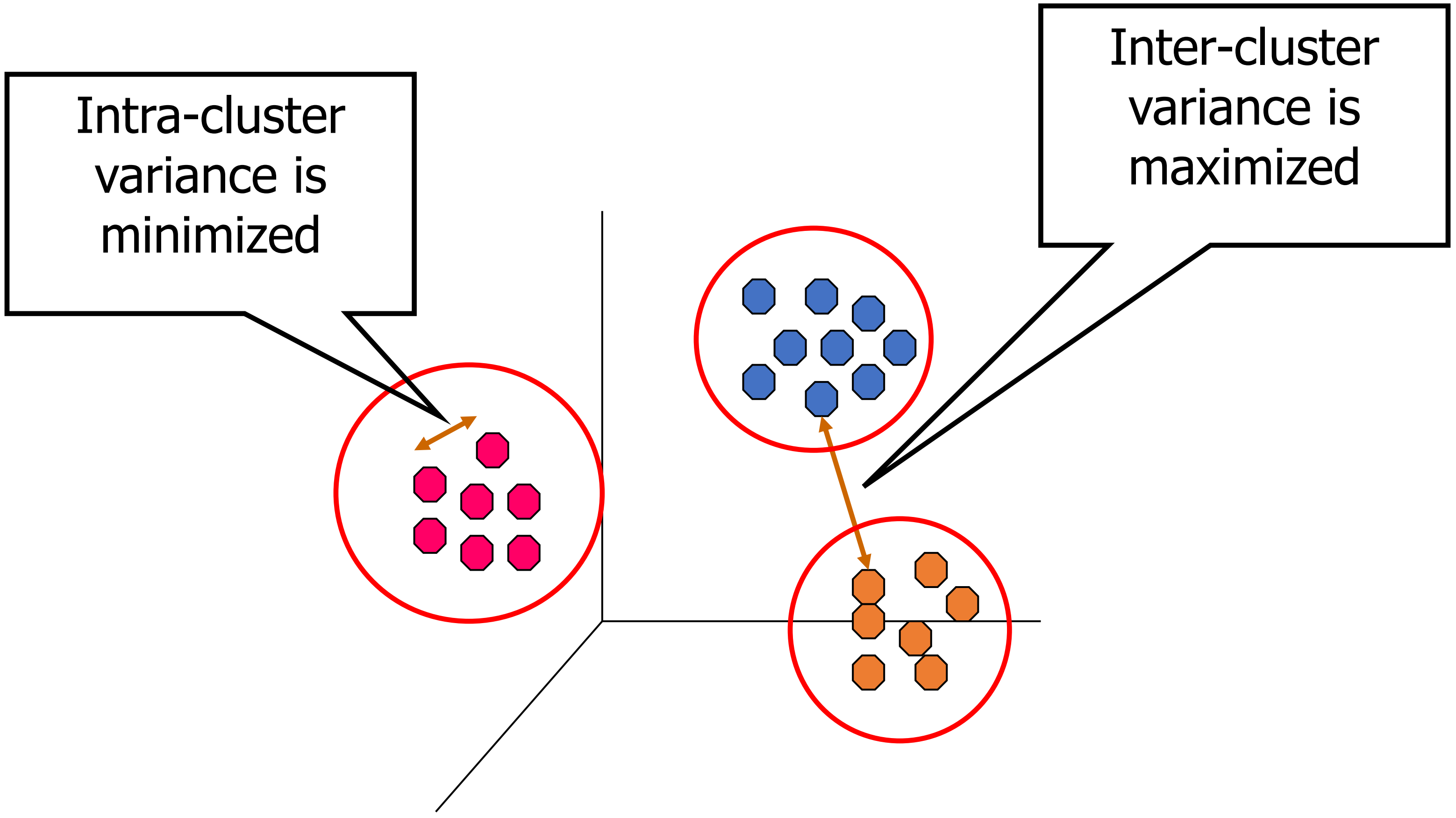
```
library(fpc)
cboot.hclust <-
clusterboot(pmatrix,clustermethod=hclustCBI,method="ward",
k=5)
summary(cboot.hclust$result)
groups<-cboot.hclust$result$partition
print_clusters(groups, 5)
cboot.hclust$bootmean
cboot.hclust$bootbrd
```

PICKING THE NUMBER OF CLUSTERS

- Cross validation (CV) – extremely common!
- Internal indexes (to minimize or maximize.
- For instance:
  - Variances of within cluster and between clusters
  - Rate-distortion method
  - F-ratio
  - Davies-Bouldin index (DBI)
  - Bayesian Information Criterion (BIC)
  - Silhouette Coefficient

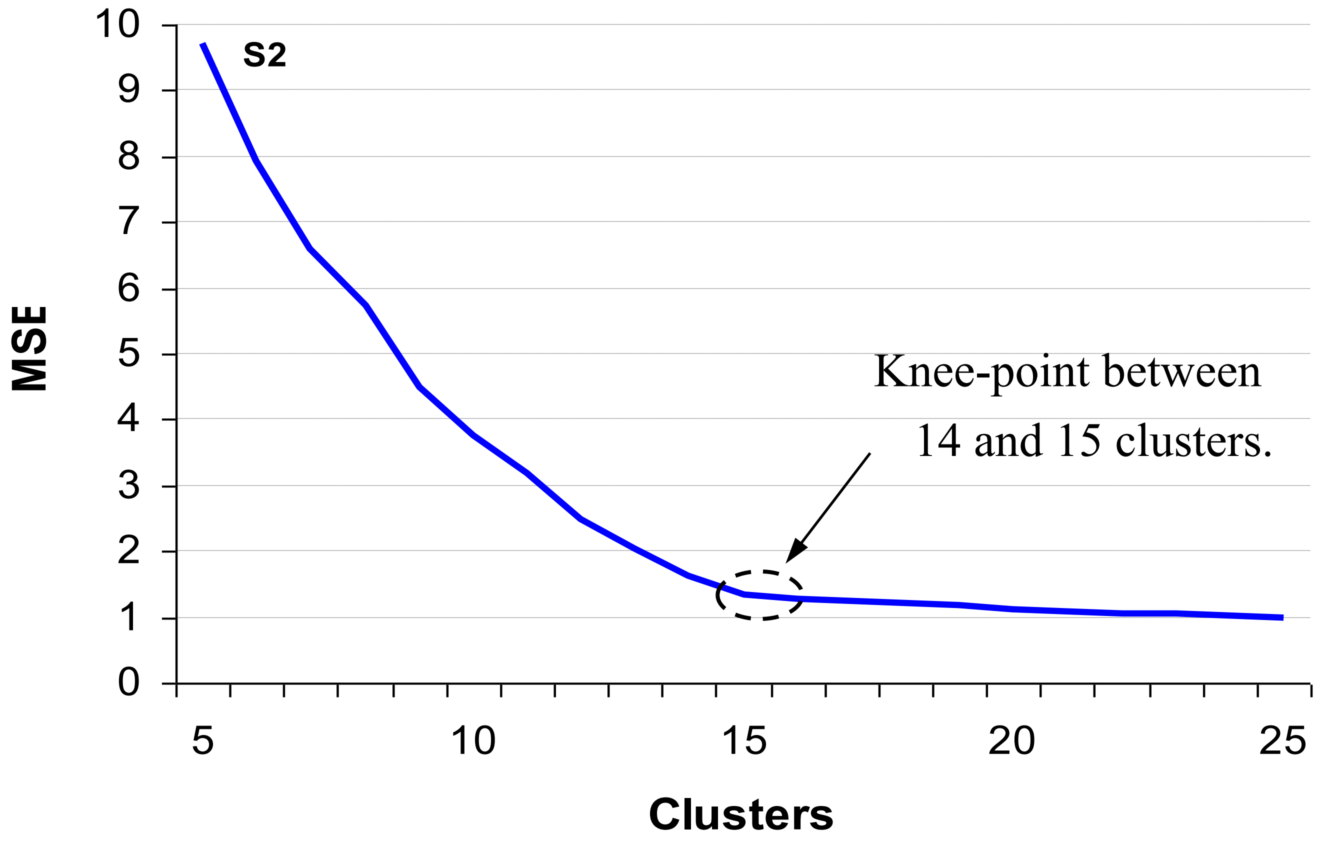
The general idea (e.g. sum of squared errors)

- Minimize within cluster variance
- Maximize between cluster variance


Intra-cluster variance is minimized
Inter-cluster variance is maximized

Sum of squared errors

- The more clusters the smaller the value.
- Small knee-point near the correct value.
- But how to detect?



Cluster validation

Supervised classification:

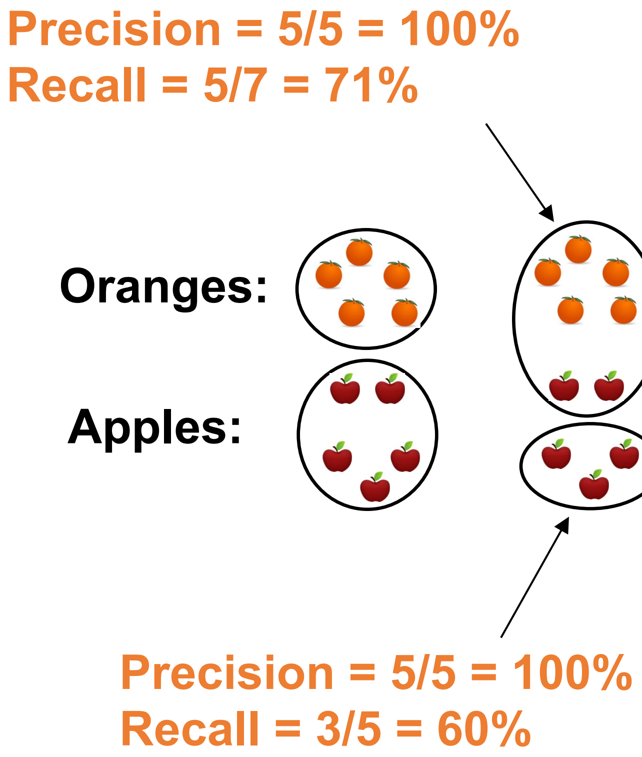
- Ground truth class labels known
- Accuracy, precision, recall

Cluster analysis:

- No class labels

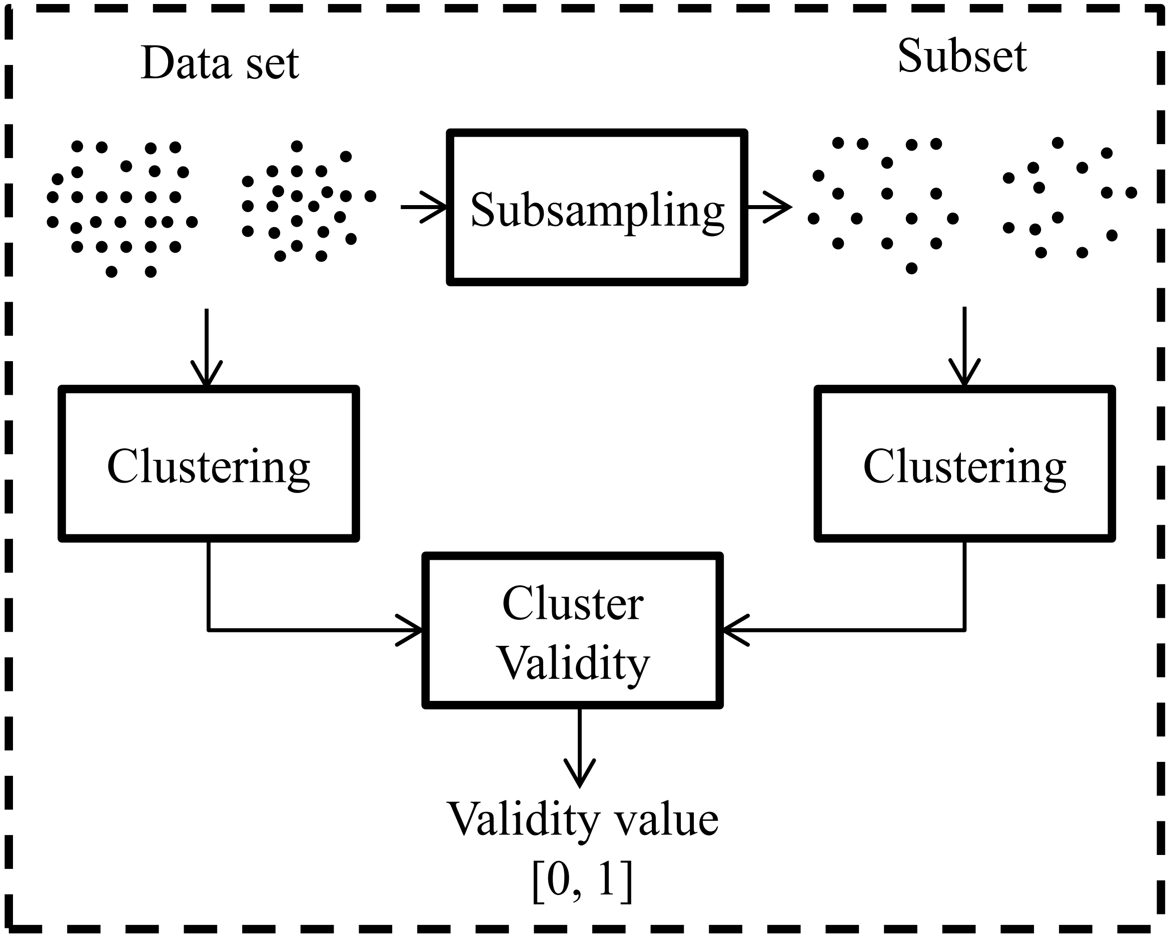
Validation need to:

- Compare clustering algorithms
- **Solve the number of clusters**
- Avoid finding patterns in noise

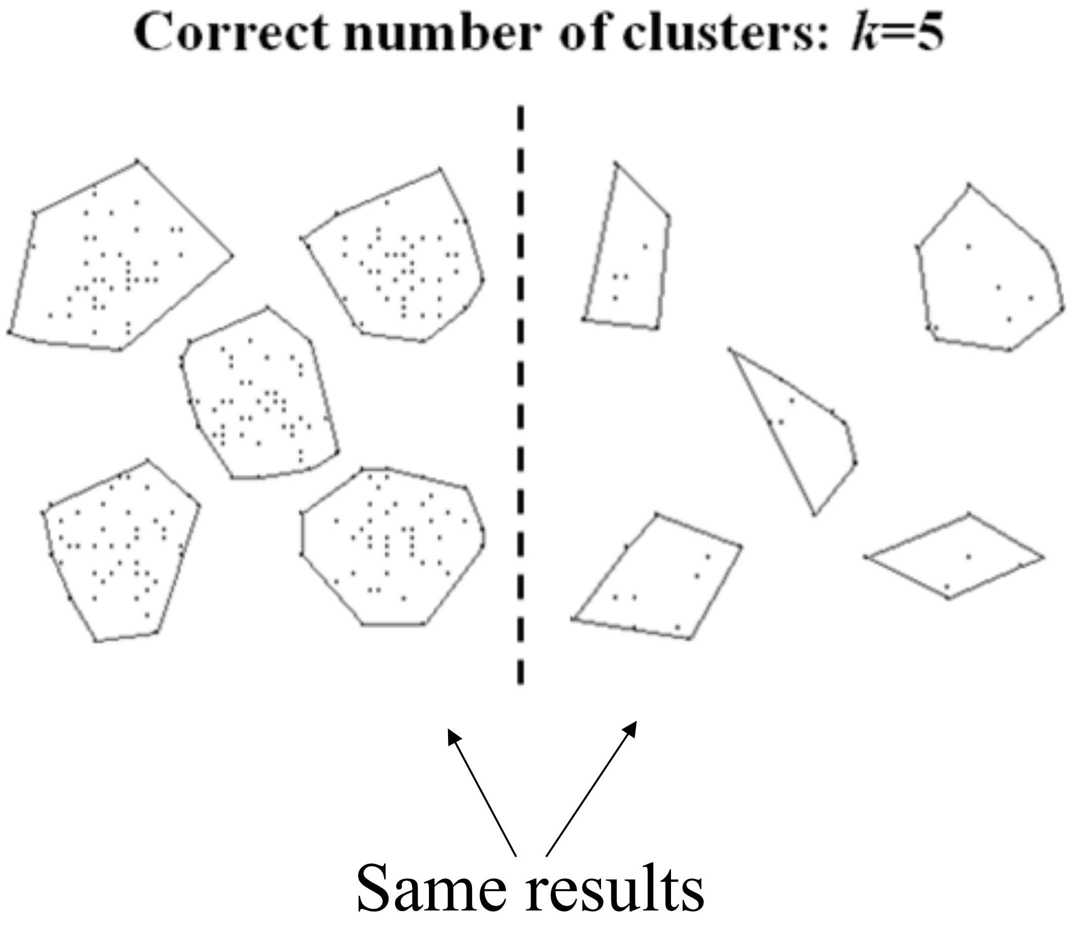


Cross-validation

Compare clustering of full data against sub-sample

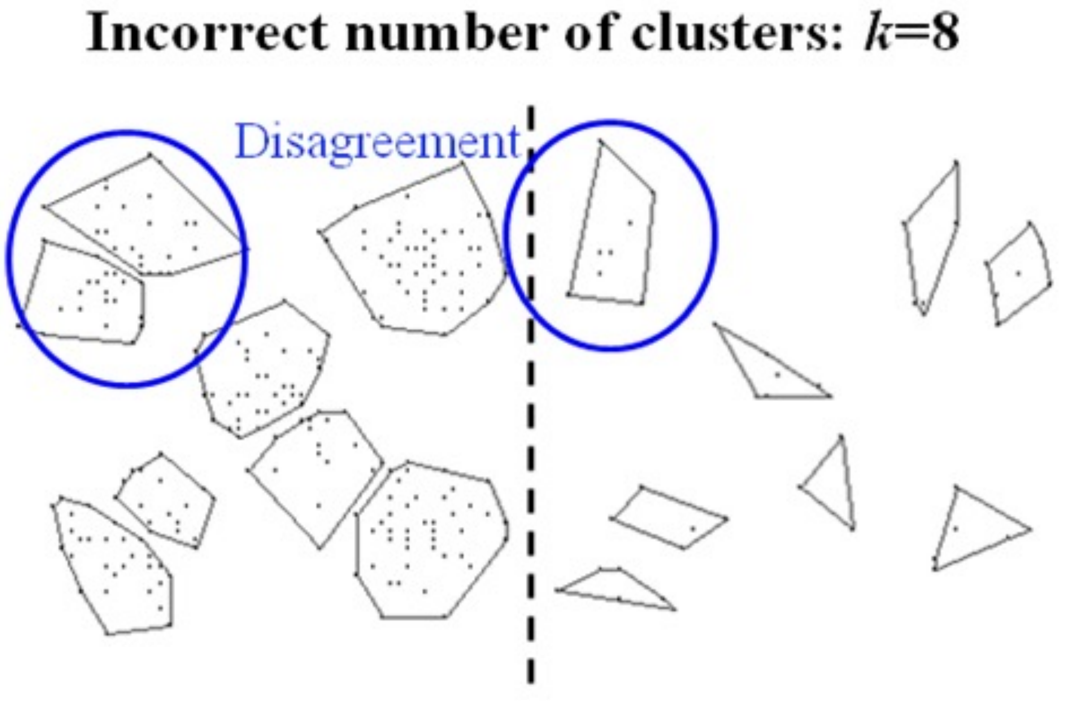


Cross-validation: Correct



Cross-validation

Incorrect



Different results

Stability approach in general

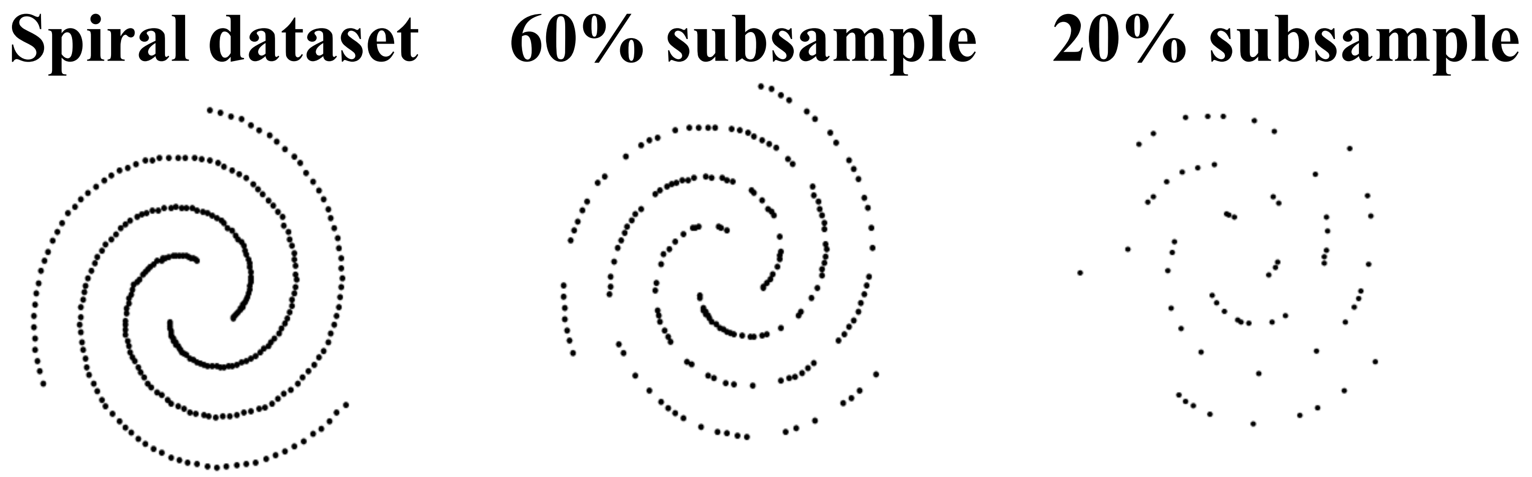
1. Add randomness
2. Cross-validation strategy
3. Solve the clustering
4. Compare clustering

Adding randomness

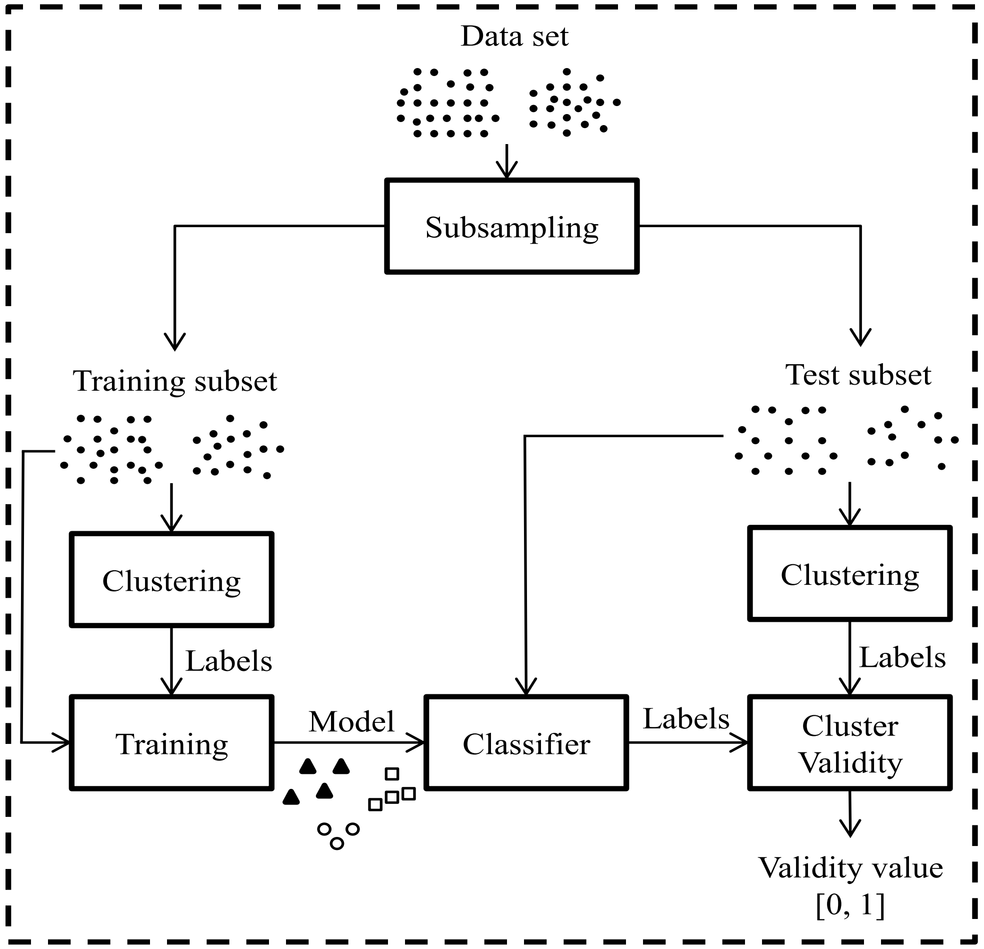
- Three choices:
  1. Subsample
  2. Add noise
  3. Randomize the algorithm
- What subsample size?
- How to model noise and how much?
- Use k-means?

Sub-sample size

- Too large (80%): same clustering always
- Too small (5%): may break cluster structure
- Recommended 20-40%



Classification approach

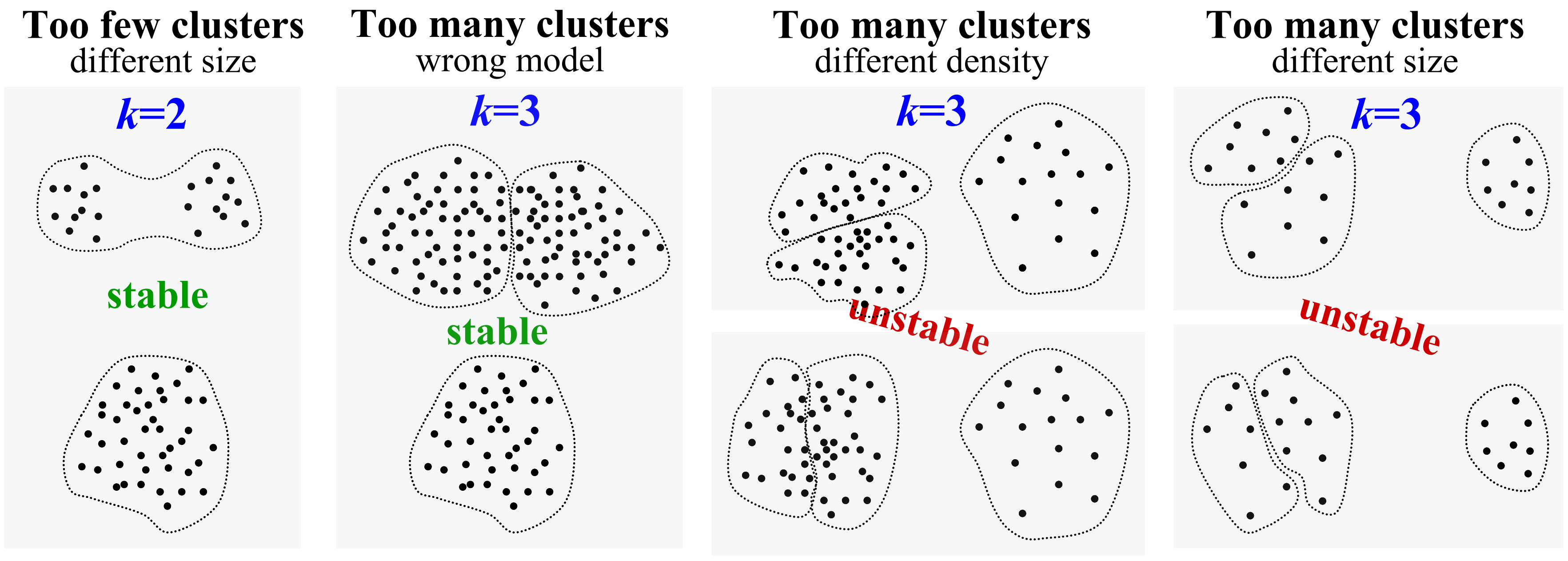


Problem

Stability can also come from other reasons:

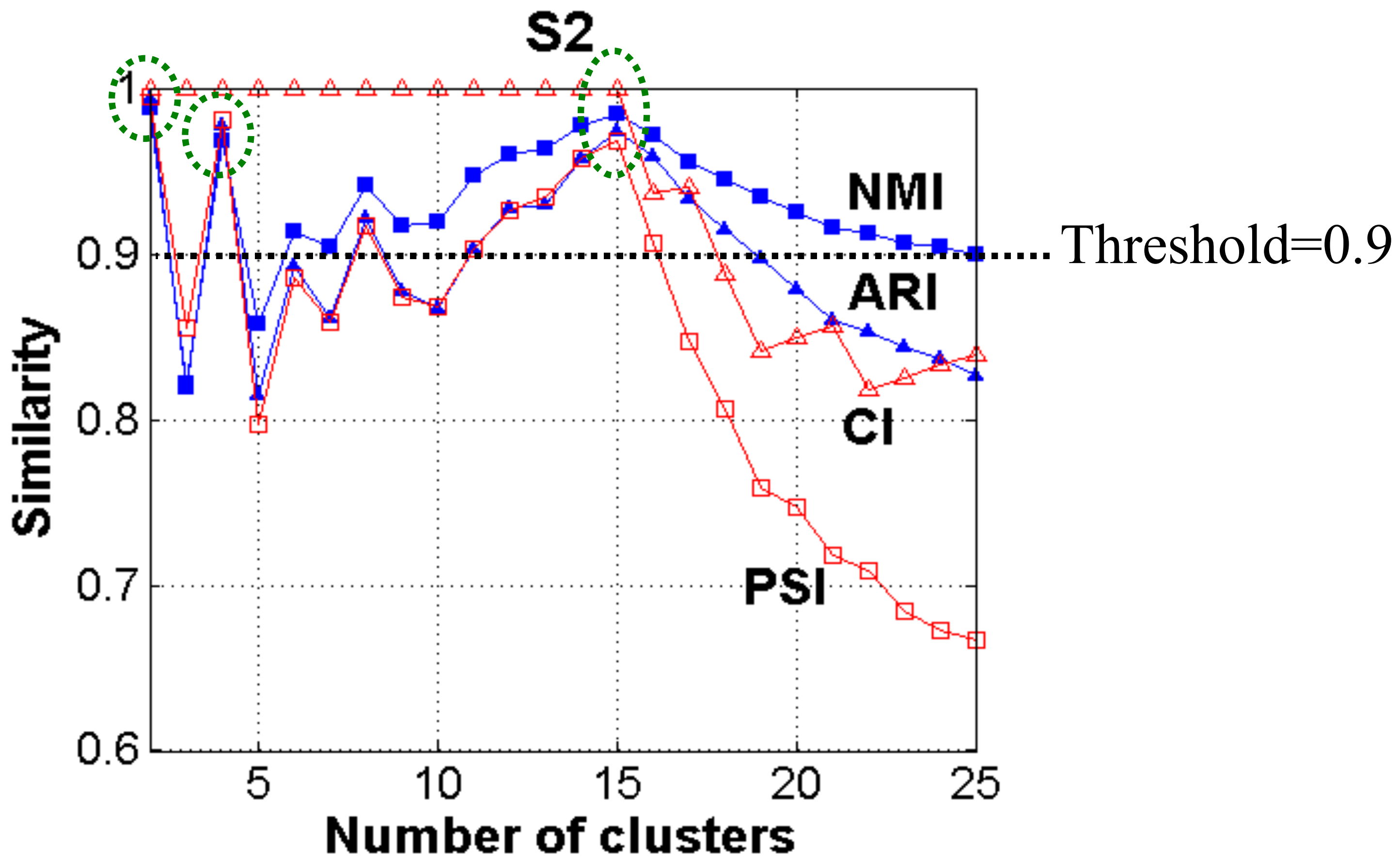
- Different cluster sizes
- Wrong cluster model

Happens when *k*<*k**



Solution

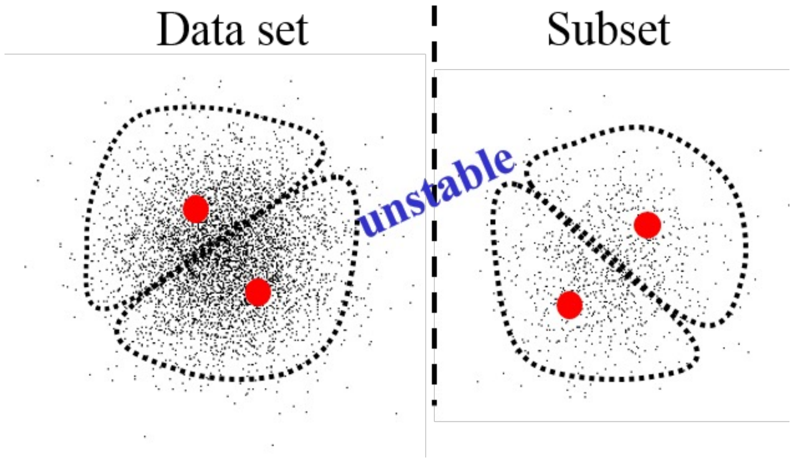
Instead of selecting *k* with maximum stability, select last *k* with stable result.



Effect of cluster shapes

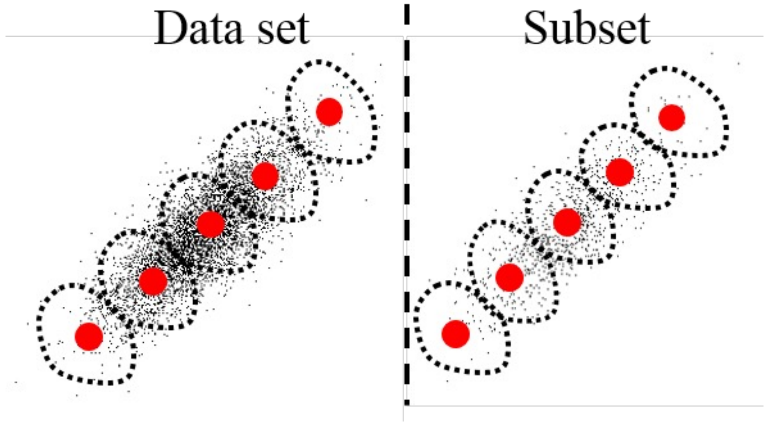
Correct model:

- Works ok.



Wrong model:

- Elliptical cluster Minimizing TSE would find 5 spherical clusters



Clustering takeaways

- The goal of clustering is to discover or draw out similarities among subsets of your data.
- In a good clustering, points in the same cluster should be more similar (nearer) to each other than they are to points in other clusters.
- When clustering, the units that each variable is measured in matter. Different units cause different distances and potentially different clusterings.
- Clustering is often used for data exploration or as a precursor to supervised learning methods.
- Different clustering algorithms might give different results. You should consider different approaches, with different numbers of clusters.
- NOT to forget probabilistic vs non-probabilistic algorithms.