How do we group resources?

CS 437/537: INTRODUCTION TO INFORMATION RETRIEVAL

Classification and Clustering

Classification and clustering are classical pattern recognition and machine learning problems

- Often applied to items: documents, emails, queries, entities & images
- Useful for a wide variety of search engine tasks

Classification, also referred to as categorization

- Asks “what class does this item belong to?”
- Supervised learning task (automatically applies labels to data/items)

Clustering

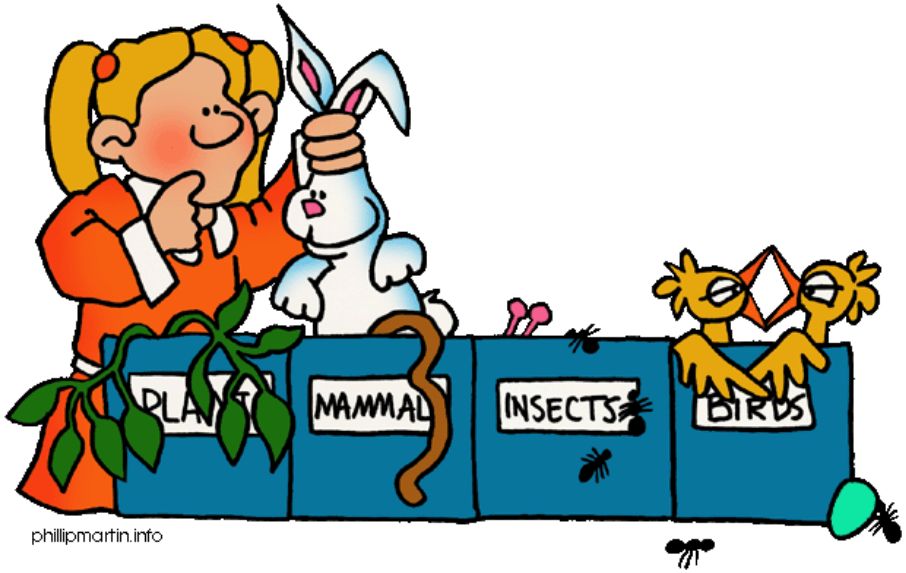
- Asks “how can I group this set of items?”
- Unsupervised learning task (grouping related items together)

Classification

Classification is the task of automatically applying labels to items

Useful for many IR-related tasks

- Spam detection
- Sentiment classification
- Online advertising
- Topic mapping



How to classify?

Example: suppose you had to classify the healthiness of a food

- Identify set of features indicative of health: fat, cholesterol, sugar, sodium
  - Extract features from foods
  - Read nutritional facts, chemical analysis, etc.
  - Combine evidence from the features into a hypothesis
  - Add health features together to get “healthiness factor”
- Classify the item based on the evidence
  - If “healthiness factor” is above a certain value, then deem it healthy

Naïve Bayes classifier

Documents are classified according to

$$Class\ (d) = argmax_{c \in C}\ P(c|d)$$

$$= argmax_{c \in C}\ \frac{P(d|c)P(c)}{P(d)}$$

$$= argmax_{c \in C}\ \frac{P(d|c)P(c)}{\sum_{c \in C} P(d|c)P(c)}$$

Must estimate P(d|c) and P(c)

- P(c) is the probability of observing class c
- P(d|c) is the probability that document d is observed given the class is known to be c

Naïve Bayes classifier

Probabilistic classifier based on Bayes' rule:

- C is a random variable corresponding to the class (input)

$$P(C|D) = \frac{P(D|C)P(C)}{P(D)}$$

$$= \frac{P(D|C)P(C)}{\sum_{c \in C} P(D|C = c)P(C = c)}$$

Based on the *term independence assumption*, the Naïve Bayes' rule yields:

$$P(c \mid d) = \frac{P(d \mid c)\, P(c)}{\sum_{c \in C} P(d \mid c)\, P(c)} = \left.\frac{\prod_{i=1}^{n} P(w_i \mid c)\, P(c)}{\sum_{c \in C} \prod_{i=1}^{n} P(w_i \mid c)\, P(c)}\right\} \text{(Chain rule)}$$

Estimating $P(c)$

P(c) is the probability of observing class c

- Estimated as the proportion of *training* documents in class c

$$P(c) = \frac{N_c}{N}$$

- N_c is the number of training documents in class c
- N is the total number of training documents

No real training, in the machine learning sense

M-B: Event space

Multiple Bernoulli for word probabilities

- Documents are represented as *binary vectors*
  - One entry for *every word in the vocabulary*
  - Entry i = 1, if word i occurs in the document; 0, otherwise
- Multiple Bernoulli distribution is a natural way to model distributions over binary vectors
  - Same model as in traditional probabilistic retrieval model

M-B: Representation

document *id*	cheap	buy	banking	dinner	the	*class*
1	0	0	0	0	1	not spam
2	1	0	1	0	1	spam
3	0	0	0	0	1	not spam
4	1	0	1	0	1	spam
5	1	1	0	0	1	spam
6	0	0	1	0	1	not spam
7	0	1	1	0	1	not spam
8	0	0	0	0	1	not spam
9	0	0	0	0	1	not spam
10	1	1	0	1	1	not spam

M-B: Estimating P($d|c$)

P(d|c) is computed (in the M-B model) as

$$P(d|c) = \prod_{w \in V} P(w|c)^{\delta(w,d)}(1 - P(w|c))^{1-\delta(w,d)}$$

- where δ(w, d) = 1 iff term w occurs in d, 0 otherwise
  - P(d|c) = 0 if term w never occurred in c in the training set
  - This is known as the "*data sparseness*" problem, which can be solved by "*smoothing*" methods

Laplacian smoothed estimate:

$$P(w|c) = \frac{df_{w,c} + 1}{N_c + 1}$$

- where $df_{w,c}$ denotes the number of documents in c including term w and N_c is the number of documents that belong to class c

Multinomial: Event space

Documents are represented as vectors of term frequencies

- One entry for *every word in the vocabulary*
- Entry i = number of times that term i occurs in the document

Multinomial distribution is a natural way to model distributions over frequency vectors

- Same event space as used in the language modeling retrieval model

Multinomial: Representation

document *id*	cheap	buy	banking	dinner	the	*class*
1	0	0	0	0	2	not spam
2	3	0	1	0	1	spam
3	0	0	0	0	1	not spam
4	2	0	3	0	2	spam
5	5	2	0	0	1	spam
6	0	0	1	0	1	not spam
7	0	1	1	0	1	not spam
8	0	0	0	0	1	not spam
9	0	0	0	0	1	not spam
10	1	1	0	1	2	not spam

Multinomial: Estimating P(d|c)

P(d|c) is computed as:

$$P(d|c) = \quad \propto \quad \prod_{w \in \mathcal{V}} P(w|c)^{tf_{w,d}}$$

Laplacian smoothed estimate:

$$P(w|c) = \frac{tf_{w,c} + 1}{|c| + |V|}$$

Number of Terms *w* in Class *c*

|*V*| is the number of *distinct terms* in the *training documents*

where $|c|$ is the number of *terms* in the *training documents* of class *c*

Multinomial vs. M-B Models

Multinomial model tends to outperform the Multiple-Bernoulli model

Implementing both models is relatively straightforward

Both classifiers are

- *Efficient*, since their statistical data can be stored in memory
- *Accurate* in document classification
- *Popular* and attractive choice as a general-purpose classifier

Feature selection

Document classifiers can have a very large number of features, such as indexed terms

- Not all features are useful
- Excessive features can increase computational cost of training and testing

Feature selection methods reduce the number of features by choosing the *most useful features*

- Feature selection can significantly *improve efficiency* (in terms of storage and processing time) while *not hurting* the *effectiveness* much (in addition to eliminating noise)

Information gain

IG is a commonly used feature selection measure

- It is the expected reduction in entropy caused by partitioning the examples according to an attribute (word)
  - Based on information theory
  - Tells how much "information" is gained (about a class) by observing some feature
  - Characterizes the (im)purity of a set of examples

In practice:

- Rank features by IG and then train model using the top *K* (typically small) attributes (words)
- The IG for a MNB classifier is computed as

Conditional Entropy

$$
\begin{aligned}
IG(w) &= H(C) - H(C|w) \\
&= -\sum_{c \in \mathcal{C}} P(c) \log P(c) + \sum_{w \in \{0,1\}} P(w) \sum_{c \in \mathcal{C}} P(c|w) \log P(c|w)
\end{aligned}
$$

Entropy of $P(c)$

Information gain

Example. The IG for the term "cheap"

$$IG(w) = -\sum_{c \in C} P(c) \log P(c) + \sum_{w \in \{0,1\}} P(w) \sum_{c \in C} P(c|w) \log P(c|w)$$

$$\begin{aligned}
IG(cheap) = & -P(spam)\log P(spam) - P(\overline{spam})\log P(\overline{spam}) + \\
& P(cheap)P(spam|cheap)\log P(spam|cheap) + \\
& P(cheap)P(\overline{spam}|cheap)\log P(\overline{spam}|cheap) + \\
& P(\overline{cheap})P(spam|\overline{cheap})\log P(spam|\overline{cheap}) + \\
& P(\overline{cheap})P(\overline{spam}|\overline{cheap})\log P(\overline{spam}|\overline{cheap}) \\
= & -\frac{3}{10}\log\frac{3}{10} - \frac{7}{10}\log\frac{7}{10} + \frac{4}{10}\cdot\frac{3}{4}\log\frac{3}{4} \\
& + \frac{4}{10}\cdot\frac{1}{4}\log\frac{1}{4} + \frac{6}{10}\cdot\frac{0}{6}\log\frac{0}{6} + \frac{6}{10}\cdot\frac{6}{6}\log\frac{6}{6} \\
= & \ 0.2749
\end{aligned}$$

where $P(\overline{cheap})$ denotes $P(cheap = 0)$,

$P(\overline{spam})$ denotes $P(not\ spam)$,

$0 \log 0 = 0$, and

$IG(buy) = 0.0008$, $IG(banking) = 0.04$, $IG(dinner) = 0.36$, $IG(the) = 0$

Clustering

A set of unsupervised algorithms that attempt to find latent structure in a set of items

Goal: identify groups (clusters) of similar items, given a set of unlabeled instances

- Suppose I gave you the shape, color, vitamin C content and price of various fruits and asked you to cluster them
  - What criteria would you use?
  - How would you define similarity?

Clustering is very sensitive to

- how items are represented
- how similarity is defined

Clustering

General outline of clustering algorithms

1. Decide how items will be represented (e.g., feature vectors)
2. Define similarity measure between pairs or groups of items (e.g., cosine similarity, Euclidian distance)
3. Determine what makes a “good” clustering (e.g., using intra- & inter-cluster similarity measures)
4. Iteratively construct clusters that are increasingly “good”
5. Stop after a local/global optimum clustering is found

Steps 3 and 4 differ the most across algorithms

Hierarchical clustering

Constructs a hierarchy of clusters

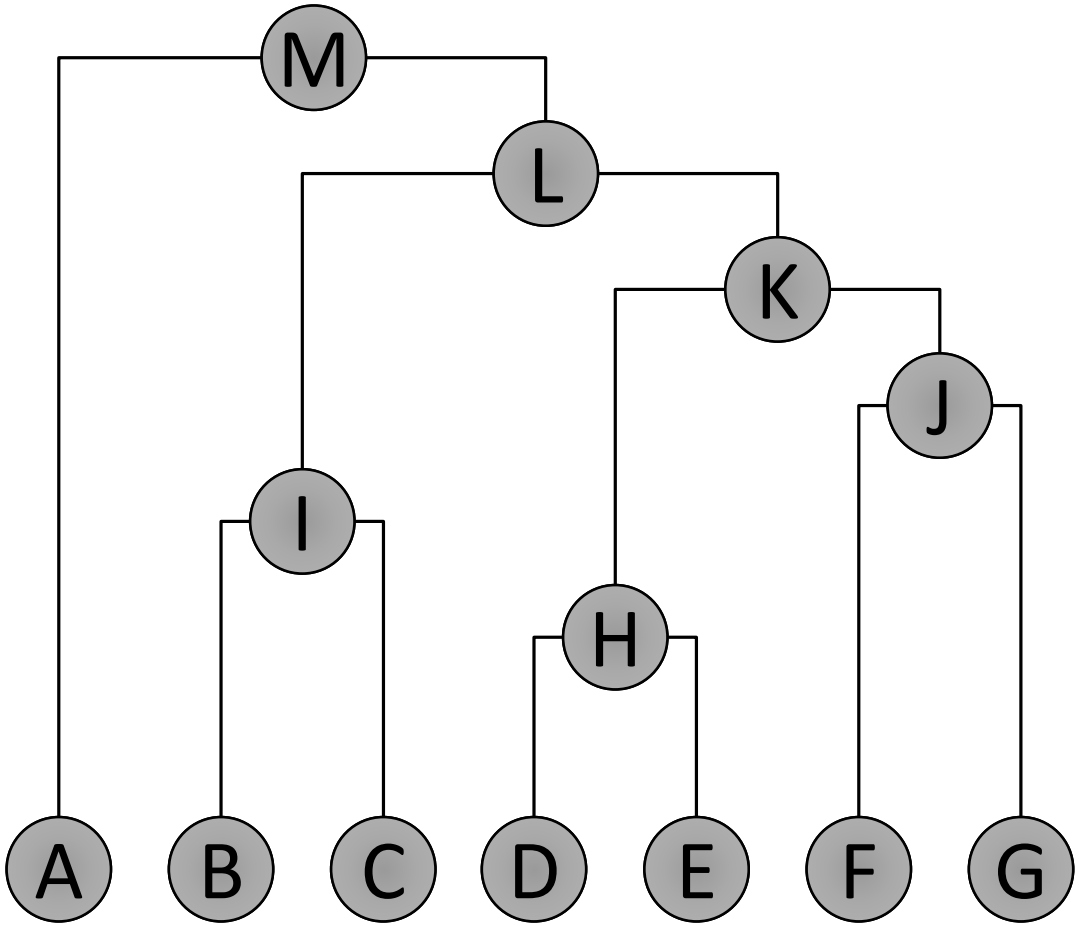
- Starting with some initial clustering of data & iteratively trying to improve the “quality” of clusters
- The top level of the hierarchy consists of a single cluster with all items in it
- The bottom level of the hierarchy consists of N (number of items) singleton clusters

Different objectives lead to different types of clusters

Two types of hierarchical clustering

- Divisive (“top down”)
- Agglomerative (“bottom up”)

Clustering as a dendrogram



Agglomerative vs. divisive

Divisive

- Start with a single cluster consisting of all of the items
- Until only singleton clusters exist
- Divide an existing cluster into two (or more) new clusters

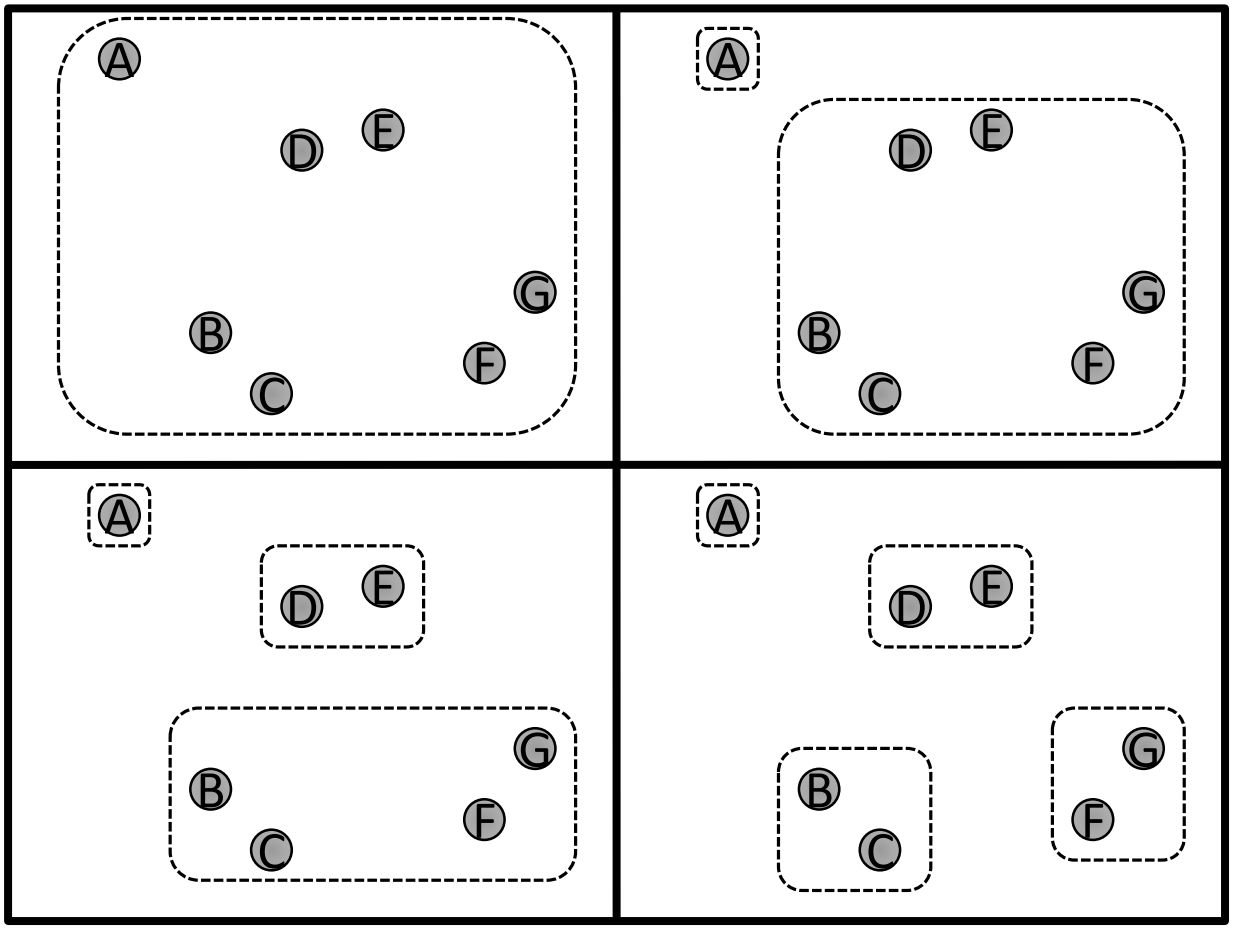
Agglomerative

- Start with N (number of items) singleton clusters
- Until a single cluster exists
- Combine two (or more) existing cluster into a new cluster

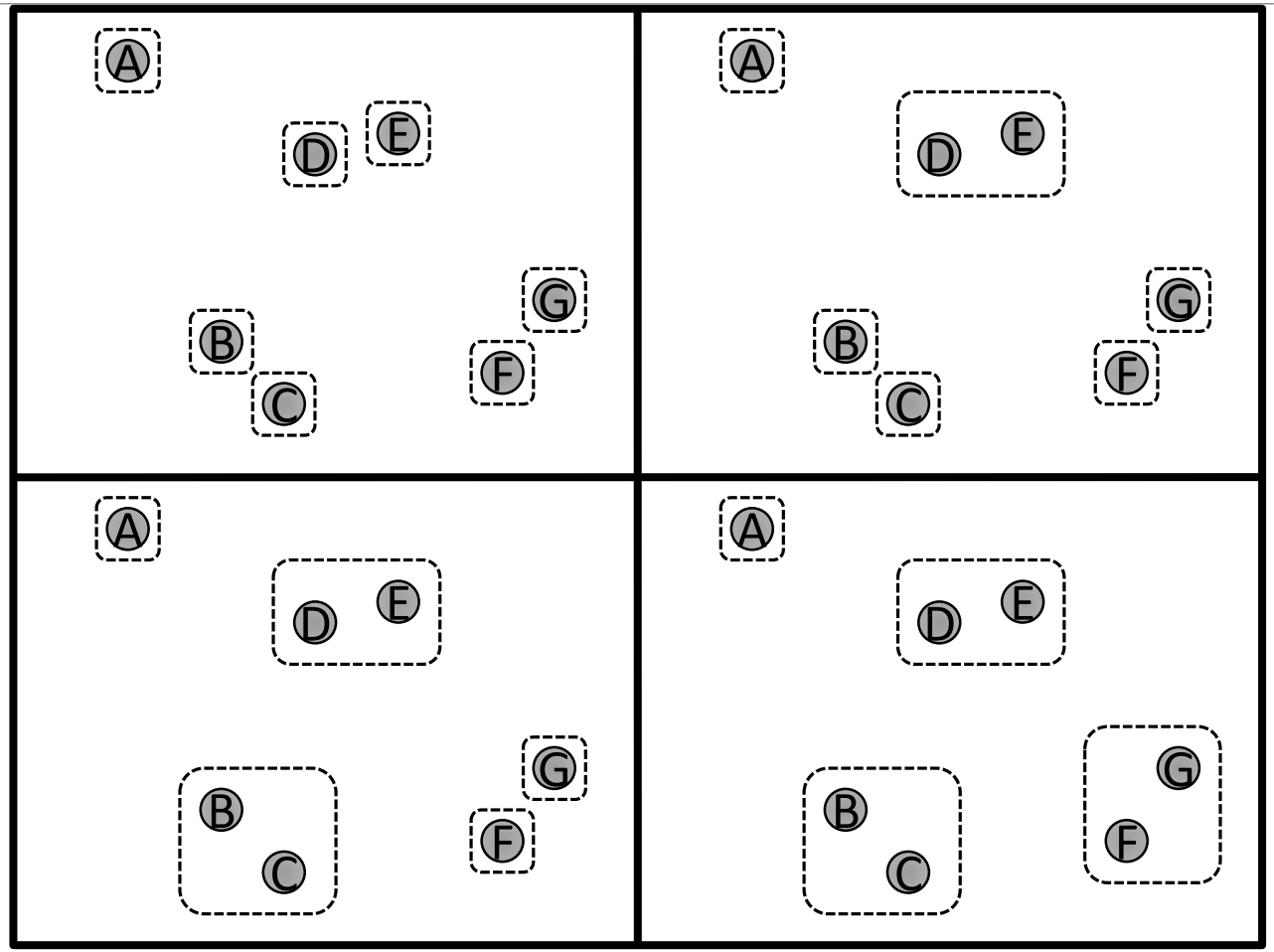
How do we know how to divide or combine clusters?

- Define a division or combination cost
- Perform the division or combination with the lowest cost

Divisive



Agglomerative



Clustering cost

Cost is a measure to capture how *expensive* is it to merge 2 clusters

- Single linkage

$$COST(C_i, C_j) = \min\{dist(X_i, X_j) | X_i \in C_i, X_j \in C_j\}$$

- Complete linkage

$$COST(C_i, C_j) = \max\{dist(X_i, X_j) | X_i \in C_i, X_j \in C_j\}$$

- Average linkage

$$COST(C_i, C_j) = \frac{\sum_{X_i \in C_i, X_j \in C_j} dist(X_i, X_j)}{|C_i||C_j|}$$

- Average group linkage

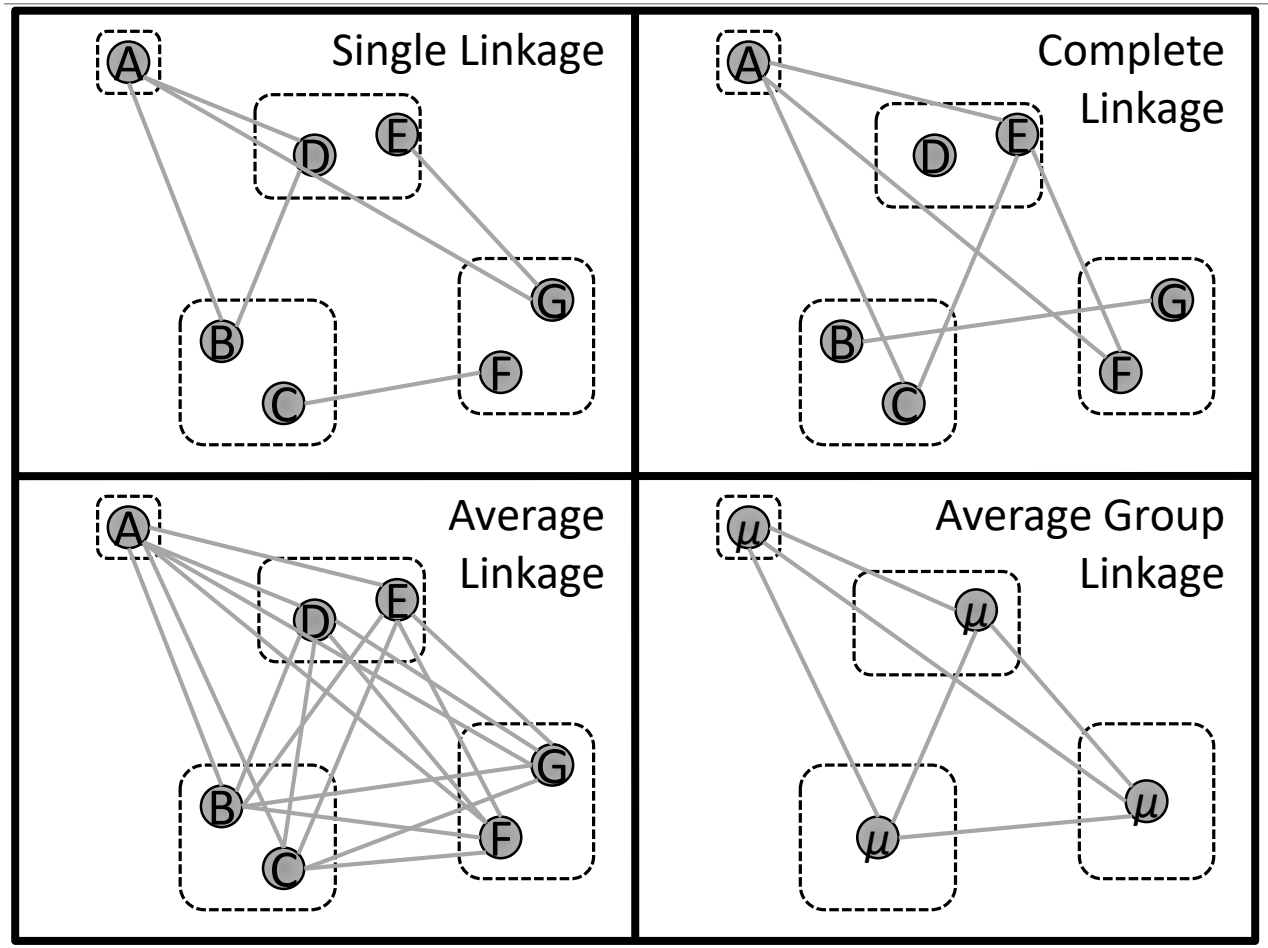
$$COST(C_i, C_j) = dist(\mu_{C_i}, \mu_{C_j})$$

where μ_C is the centroid of cluster C

Euclidean distance

$$D(p,q) = D(q,d) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Visualizing cost



Generally, *Average-Link* Clustering yields the best effectiveness

Cost choice

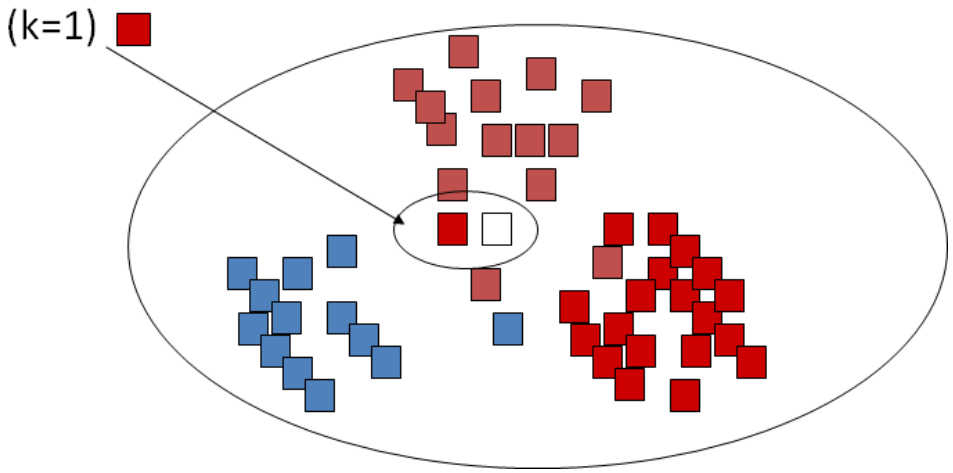
The choice of the best clustering technique/strategy requires experiments & evaluation

- Single linkage
  - Could result in “ very long” or “spread-out” clusters
- Complete linkage
  - Clusters are more compact than Single Linkage
- Average linkage
  - A compromise between Single & Complete Linkage
- Average group linkage
  - Closely related to the Average Linkage approach

K-Nearest Neighbor Clustering

K-Nearest neighbor (KNN) clustering forms one cluster per item

- The cluster for item j consists of j and the K nearest neighbors of j
- Clusters can overlap



Lazy Learner
All computation deferred until classification (***no training process***)

KNN: How does it work?

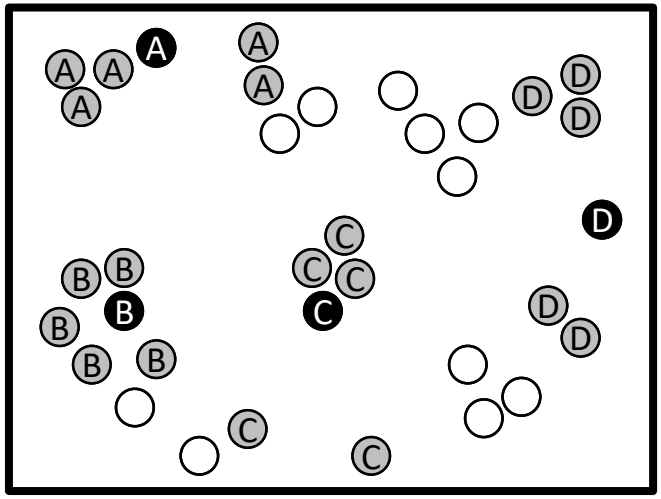
Requires 3 things:

- Training data (***samples***)
- Distance ***metric*** to compute distance between records
- The value of ***k***: the number of nearest neighbors to retrieve

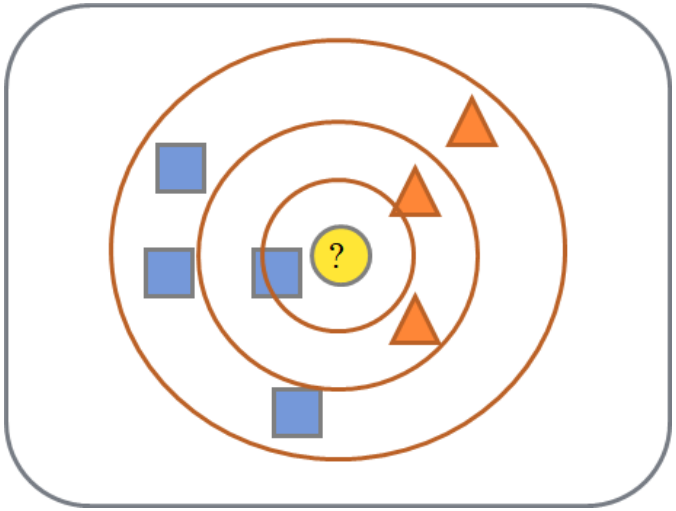
To classify an unknown instance:

- Compute distance to other training instances
- Identify K nearest neighbors
- Use
  - Class labels of nearest neighbors to determine the class label of unknown instance
  - Nearest neighbors to form a cluster

Examples



Clustering
K = 5



Majority Voting
K=1; square class
K=3; triangle class
K=7; square class

Drawbacks and Applications

Drawbacks

- Often fails to find meaningful clusters
  - In sparse areas of the input space, the instances assigned to a cluster are far away (e.g., D in the 5-NN example)
  - In dense areas, some related instances may be missed if K is not large enough (e.g., B in the 5-NN example)
- Computational expensive since it computes distances between each pair of instances
- Can generate ties if used for labeling (unless K is odd)

Applications

- Emphasize finding a small number (rather than all) of closely related instances, i.e., precision over recall
- Content-based image retrieval (e.g., given a query image, find visually similar images)
- Non-personalized news recommendations (e.g., given a current reading, what other articles might be alike)

K-Means clustering

Hierarchical clustering constructs a hierarchy of clusters

K-means always maintains exactly K clusters

- Clusters are represented by their centroids ("centers of mass")

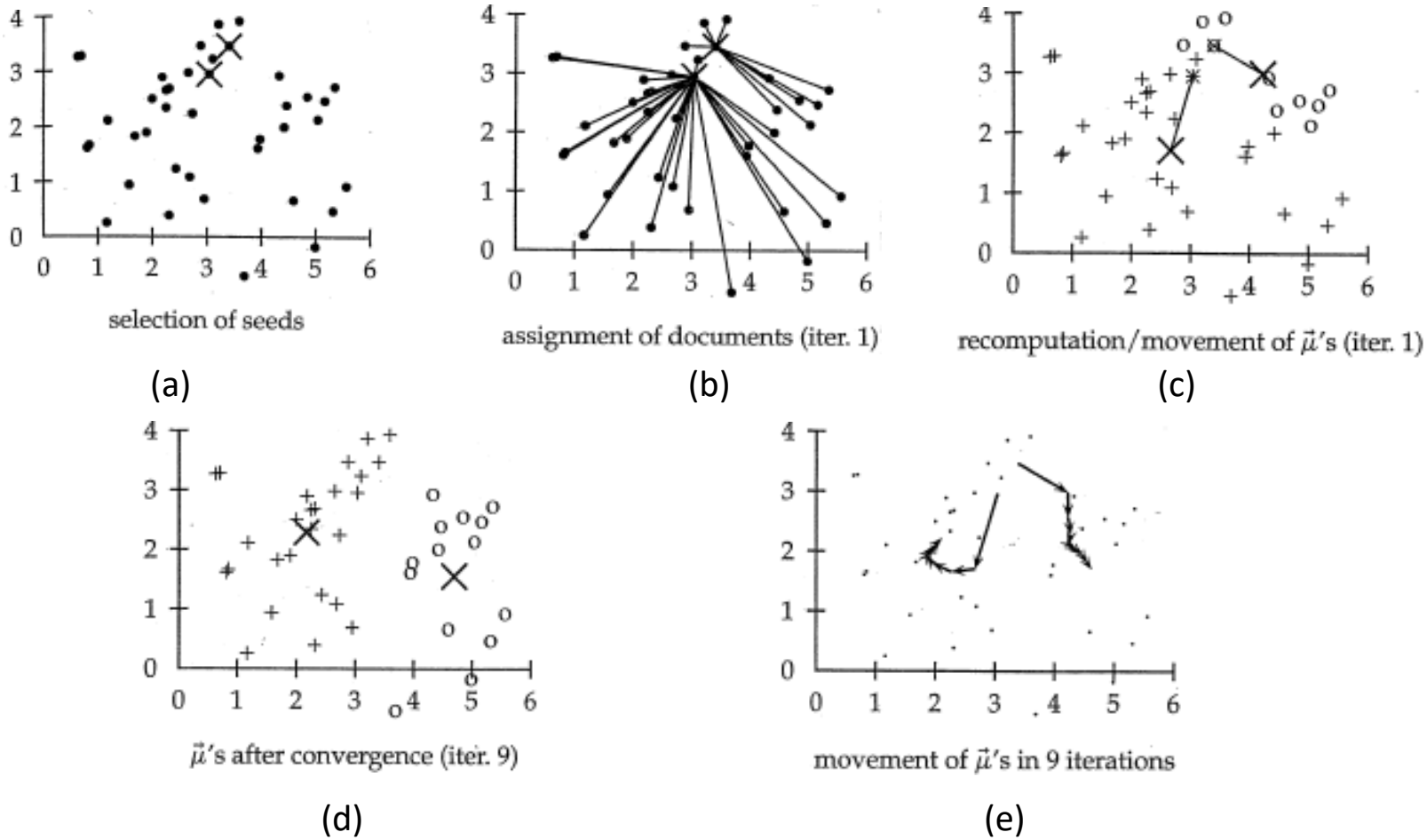
Basic algorithm:

- Step 0: Choose K cluster centroids
- Step 1: Assign points to closet centroid
- Step 2: Re-compute cluster centroids
- Step 3: Go to Step 1

Tends to converge quickly

Can be sensitive to choice of *initial centroids*
Must choose K to begin with ☹

Visualizing K-Means



K-Means Clustering Algorithm

Algorithm 1 K-Means Clustering

```
procedure KMeansCluster(X_1, ..., X_N, K)
    A[1], ..., A[N] ← initial cluster assignment   (* Either randomly or using
    repeat                                          some knowledge of the data *)
        change ← false
        for i = 1 to N do
            k̂ ← arg min_k dist(X_i, C_k)            (* Each instance is assigned to the
            if A[i] is not equal k̂ then                closest cluster *)
                A[i] ← k̂
                change ← true                       (* The cluster of an instance changes;
            end if                                     proceeds *)
        end for
    until change is equal to false return A[1], ..., A[N]
end procedure
```

How to choose K?

K-means and K nearest neighbor clustering require us to choose K

No theoretically appealing way of choosing K

Depends on the application and data; often chosen experimentally to evaluate the quality of the resulting clusters for various values of K

Can use hierarchical clustering and choose the best level

Can use adaptive K for K-nearest neighbor clustering

- Larger (Smaller) K for dense (spare) areas
- Challenge: choosing the boundary size

Difficult problem with no clear solution

Evaluation

Classification

- Precision = proportion of correctly labeled / total instances
  - Given that we have ground truth (*known labels*) we can compute

Cluster

- If ground truth is available – external criterion: Purity
  - Each *cluster* is assigned to the *class* which is *most frequent* in the cluster
  - The *accuracy* of the assignment is measured by counting the number of correctly assigned documents divided by *N*, the total number of documents to be clustered
- Otherwise – internal criterion: Intra and Inter cluster similarity
  - Attaining high intra-cluster similarity (documents within a cluster are similar)
  - Achieving low inter-cluster similarity (documents from different clusters are dissimilar)