Search Engines

Classification and Clustering

- Classification and clustering are classical pattern recognition / machine learning problems
- Classification
  - Asks “what class does this item belong to?”
  - *Supervised learning* task
- Clustering
  - Asks “how can I group this set of items?”
  - *Unsupervised learning* task
- Items can be documents, queries, emails, entities, images, etc.
- Useful for a wide variety of search engine tasks

Classification

- Classification is the task of automatically applying labels to items
- Useful for many search-related tasks
  - Spam detection
  - Sentiment classification
  - Online advertising
- Two common approaches
  - Probabilistic
  - Geometric

How to Classify?

- How do humans classify items?
- For example, suppose you had to classify the "healthiness" of a food
  - Identify set of *features* indicative of health
    - fat, cholesterol, sugar, sodium, etc.
  - *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"
  - Finally, *classify* the item based on the evidence
    - If "healthiness factor" is above a certain value, then deem it healthy

Classes of Classifiers

- Types of classifiers
  - Generative (Naïve-Bayes)
  - Discriminative (SVMs)
  - Non-parametric (nearest neighbor)
- Types of learning
  - Supervised (Naïve-Bayes, SVMs)
  - Semi-supervised (Rocchio, relevance models)
  - Unsupervised (clustering)

Ontologies

- Ontology is a labeling or categorization scheme
- Examples
  - Binary (spam, not spam)
  - Multi-valued (red, green, blue)
  - Hierarchical (news/local/sports)
- Different classification tasks require different ontologies

Generative vs. Discriminative

- Generative models
  - Assumes documents and classes are drawn from joint distribution P(*d*, *c*)
  - Typically P(*d*, *c*) decomposed to P(*d* | *c*) P(*c*)
  - Effectiveness depends on how P(*d*, *c*) is modeled
  - Typically more effective when little training data exists
- Discriminative models
  - Directly model class assignment problem
  - Do not model document “generation”
  - Effectiveness depends on amount and quality of training data

Naïve Bayes Classifier

- Probabilistic classifier based on Bayes' rule:

$$\begin{aligned} P(C|D) &= \frac{P(D|C)P(C)}{P(D)} \\ &= \frac{P(D|C)P(C)}{\sum_{c\in\mathcal{C}} P(D|C=c)P(C=c)} \end{aligned}$$

- *C* is a random variable corresponding to the class
- *D* is a random variable corresponding to the input (e.g. document)

Probability 101: Random Variables

- Random variables are non-deterministic
  - Can be discrete (finite number of outcomes) or continues
  - Model uncertainty in a variable
- P(*X* = *x*) means “the probability that random variable X takes on value x”
- Example:
  - Let X be the outcome of a coin toss
  - P(X = heads) = P(X = tails) = 0.5
- Example: *Y* = 5 - 2*X*
  - If *X* is random, then *Y* is random
  - If *X* is deterministic then *Y* is also deterministic
    - *Note:* “Deterministic” just means P(X = x) = 1.0!

Naïve Bayes Classifier

- Documents are classified according to:

$$\begin{aligned}\text{Class}(d) &= \arg\max_{c\in\mathcal{C}} P(c|d) \\ &= \arg\max_{c\in\mathcal{C}} \frac{P(d|c)P(c)}{\sum_{c\in C} P(d|c)P(c)}\end{aligned}$$

- 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*

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

Estimating P(*d* | *c*)

- P(*d* | *c*) is the probability that document *d* is observed given the class is known to be *c*
- Estimate depends on the *event space* used to represent the documents
- What is an event space?
  - The set of all possible outcomes for a given random variable
  - For a coin toss random variable the event space is *S* = {heads, tails}

Multiple Bernoulli Event Space

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

Multiple Bernoulli Document 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

Multiple-Bernoulli: Estimating P(*d* | *c*)

- P(*d* | *c*) is computed as:

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

- Laplacian smoothed estimate:

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

- Collection smoothed estimate:

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

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 Document 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:

$$\begin{aligned} P(d|c) &= P(|d|)\left(tf_{w_1,d}, tf_{w_2,d}, \ldots, tf_{w_{\mathcal{V}},d}\right)! \prod_{w \in \mathcal{V}} P(w|c)^{tf_{w,d}} \\ &\propto \prod_{w \in \mathcal{V}} P(w|c)^{tf_{w,d}} \end{aligned}$$

- Laplacian smoothed estimate:

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

- Collection smoothed estimate:

$$P(w|c) = \frac{tf_{w,c} + \mu \frac{cf_w}{|C|}}{|c| + \mu}$$

Support Vector Machines

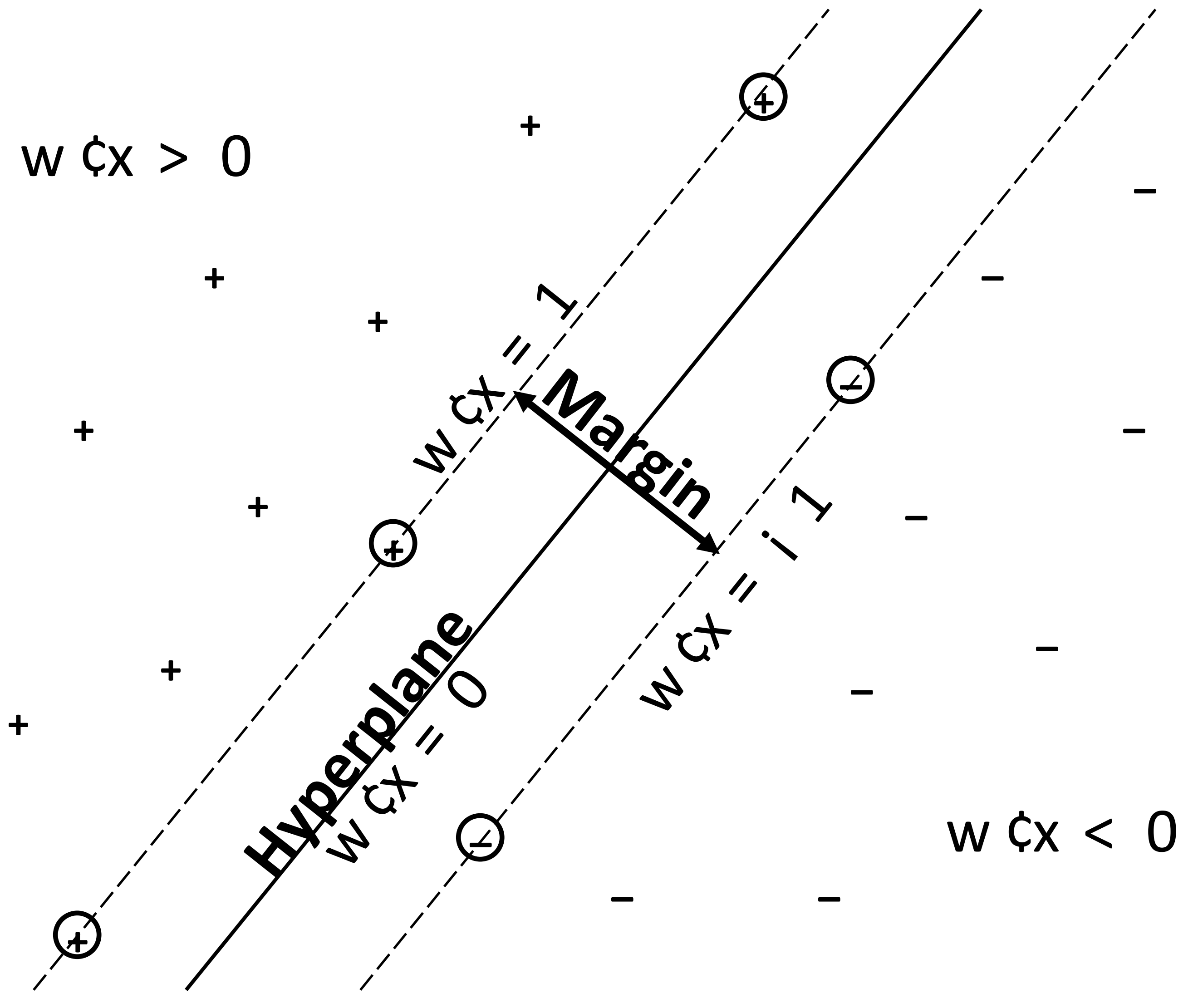
- Based on geometric principles
- Given a set of inputs labeled '+' and '-', find the "best" hyperplane that separates the '+'s and '-'s
- Questions
  - How is "best" defined?
  - What if no hyperplane exists such that the '+'s and '-'s can be perfectly separated?

"Best" Hyperplane?

- First, what is a hyperplane?
  - A generalization of a line to higher dimensions
  - Defined by a vector *w*
- With SVMs, the best hyperplane is the one with the *maximum margin*
- If x^+ and x^- are the closest '+' and '-' inputs to the hyperplane, then the margin is:

$$\mathrm{Margin}(w) = \frac{|w \cdot x^-| + |w \cdot x^+|}{||w||}$$

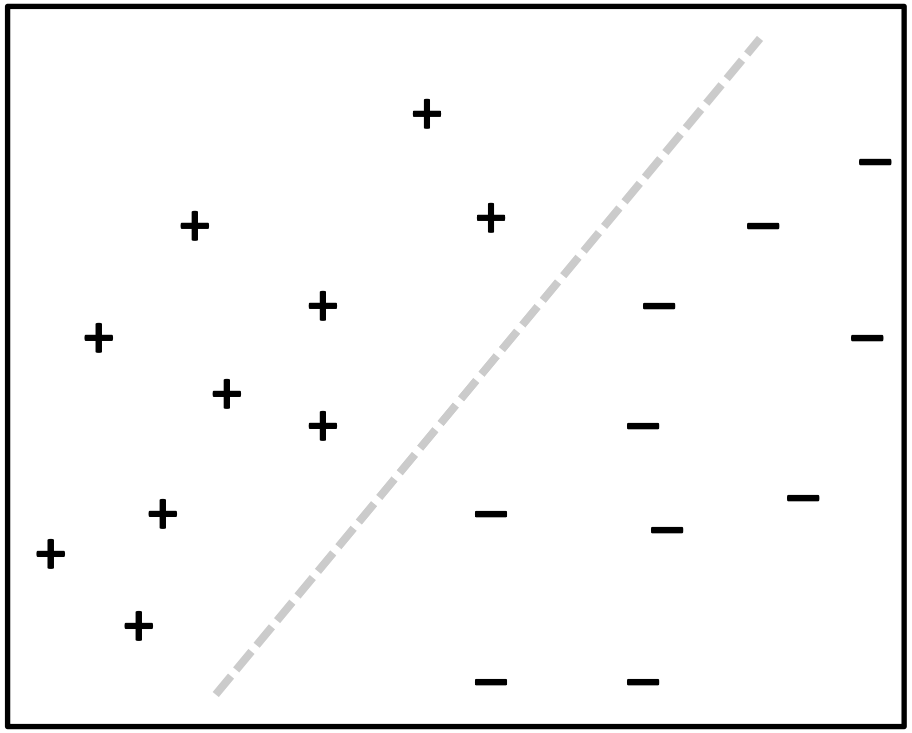
Support Vector Machines

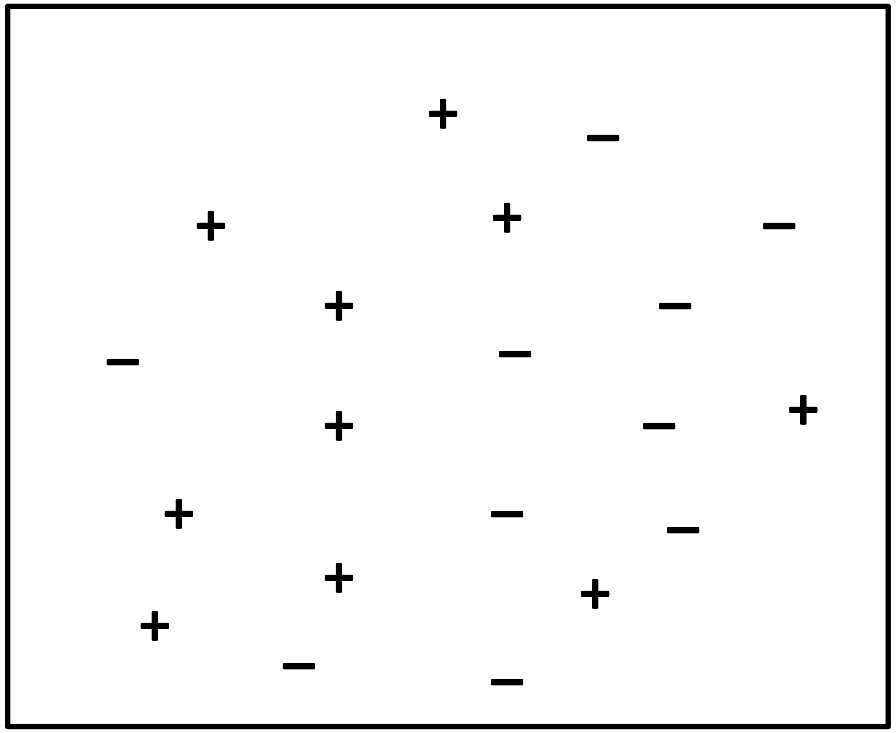


"Best" Hyperplane?

- It is typically assumed that $|w \cdot x^{-}| = |w \cdot x^{+}| = 1$, which does not change the solution to the problem
- Thus, to find the hyperplane with the largest margin, we must maximize $\frac{2}{||w||}$.
- This is equivalent to minimizing $||w||^2$.

Separable vs. Non-Separable Data


Separable


Non-Separable

Linear Separable Case

- In math:

$$\begin{aligned}
&minimize: && \tfrac{1}{2}||w||^2 \\
&subject\ to: && \\
& && w \cdot x_i \geq 1 \quad \forall i \text{ s.t. Class}(i) = + \\
& && w \cdot x_i \leq -1 \quad \forall i \text{ s.t. Class}(i) = -
\end{aligned}$$

- In English:
  - Find the largest margin hyperplane that separates the ‘+’s and ‘-’s

Linearly Non-Separable Case

- In math:

$$
\begin{aligned}
&\textit{minimize}: \quad \tfrac{1}{2}||w||^2 + C\textstyle\sum_{i=1}^{N}\xi_i \\
&\textit{subject to}:
\end{aligned}
$$

$$
\begin{aligned}
w \cdot x_i &\geq 1 - \xi_i & \forall i \text{ s.t. Class}(i) = + \\
w \cdot x_i &\leq -1 + \xi_i & \forall i \text{ s.t. Class}(i) = \text{-} \\
\xi_i &\geq 0 & \forall i
\end{aligned}
$$

- In English:
  - ξ_i denotes how misclassified instance *i* is
  - Find a hyperplane that has a large margin and lowest misclassification cost

The Kernel Trick

- Linearly non-separable data may become linearly separable if transformed, or mapped, to a higher dimension space
- Computing vector math (i.e., dot products) in very high dimensional space is costly
- The kernel trick allows very high dimensional dot products to be computed efficiently
- Allows inputs to be implicitly mapped to high (possibly infinite) dimensional space with little computational overhead

Kernel Trick Example

- The following function maps 2-vectors to 3-vectors:

$$\Phi(x) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

- Standard way to compute $\Phi(w) \cdot \Phi(x)$ is to map the inputs and compute the dot product in the higher dimensional space
- However, the dot product can be done entirely in the original 2-dimensional space:

$$\begin{aligned} \Phi(w) \cdot \Phi(x) &= w_1^2x_1^2 + 2w_1w_2x_1x_2 + w_2^2x_2^2 \\ &= (w \cdot x)^2 \end{aligned}$$

Common Kernels

- The previous example is known as the polynomial kernel (with *p* = 2)
- Most common kernels are linear, polynomial, and Gaussian
- Each kernel performs a dot product in a higher implicit dimensional space

Kernel Type	Value	Implicit Dimension
Linear	$K(x_1, x_2) = x_1 \cdot x_2$	N
Polynomial	$K(x_1, x_2) = (x_1 \cdot x_2)^p$	$\begin{pmatrix} N+p-1 \\ N \end{pmatrix}$
Gaussian	$K(x_1, x_2) = \exp -\|\|x_1 - x_2\|\|^2 / 2\sigma^2$	Infinite

Non-Binary Classification with SVMs

- One versus all
  - Train “class *c* vs. not class *c*” SVM for every class
  - If there are K classes, must train *K* classifiers
  - Classify items according to:

$$\mathrm{Class}(x) = \arg\max_{c} w_c \cdot x$$

- One versus one
  - Train a binary classifier for every pair of classes
  - Must train *K*(*K*-1)/2 classifiers
  - Computationally expensive for large values of *K*

SVM Tools

- Solving SVM optimization problem is not straightforward
- Many good software packages exist
  - SVM-Light
  - LIBSVM
  - R library
  - Matlab SVM Toolbox

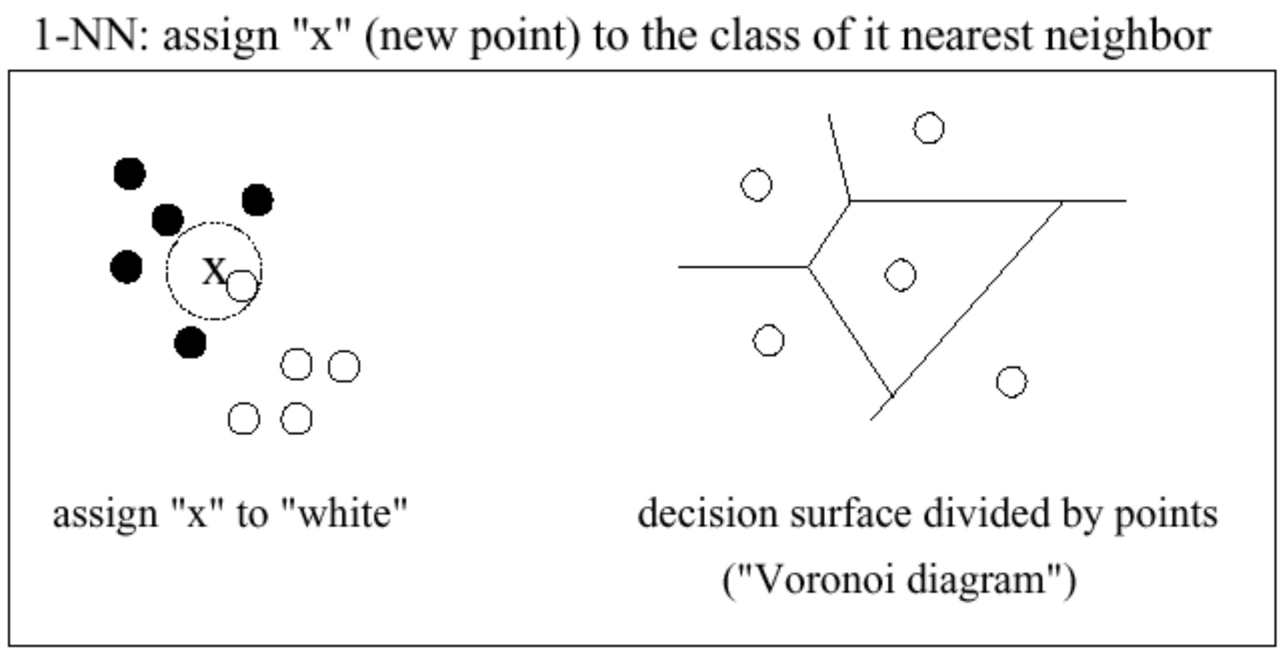
Nearest Neighbor Classification

- *Instance-Based Learning, Lazy Learning*
- well-known approach to pattern recognition
- initially by Fix and Hodges (1951)
- applied to text categorization in early 90's
  - strong baseline in benchmark evaluations
- among top-performing methods in TC evaluations
- Assigns the class to an item based on the neighbors of the item and their class

Key Components of Nearest Neighbor

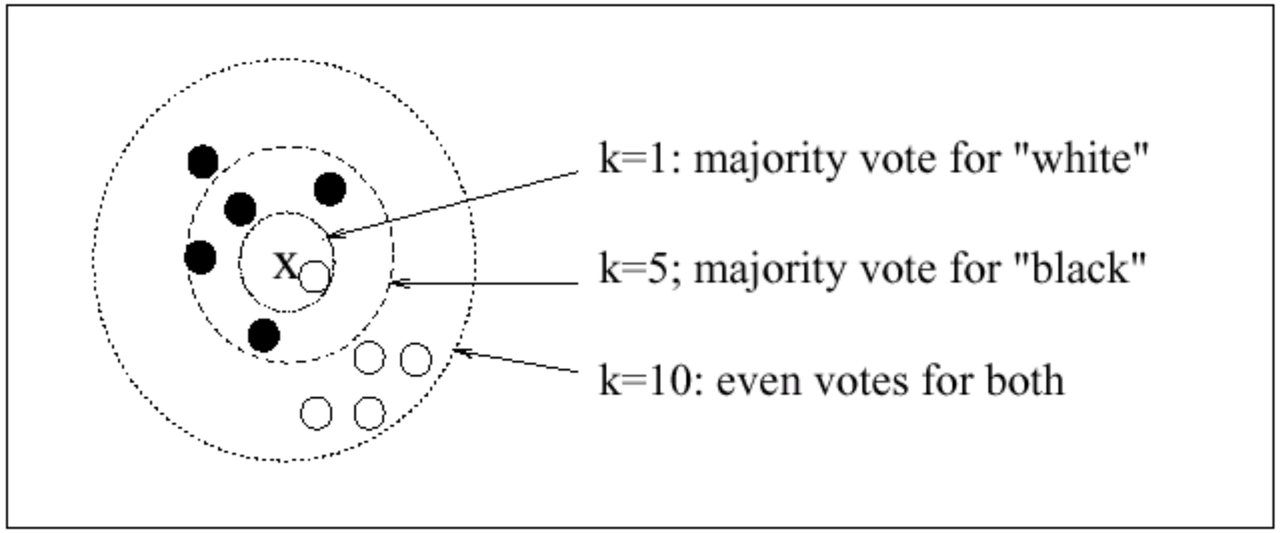
- “Similar” item: We need a functional definition of “similarity” if we want to apply this automatically.
- How many neighbors do we consider?
- Does each neighbor get the same weight?
- All categories in neighborhood? Most frequent only? How do we make the final decision?

1-Nearest Neighbor (graphically)


1-NN: assign "x" (new point) to the class of it nearest neighbor

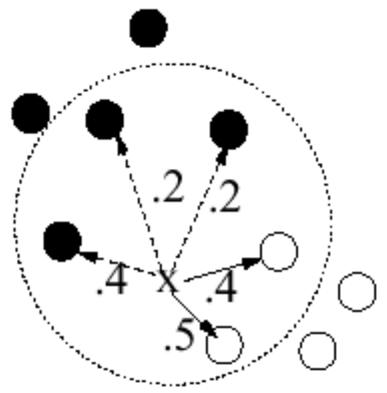
K-Nearest Neighbor using a *majority* voting scheme

K-Nearest Neighbor using a *majority* voting scheme


x
k=1: majority vote for "white"
k=5; majority vote for "black"
k=10: even votes for both

K-NN using a weighted-sum voting Scheme

k-NN using a weighted-sum voting scheme



kNN (k = 5)

Assign "white" to x because the weighted sum of "whites" is larger then the sum of "blacks".

Each neighbor is given a weight according to its nearness.

Category Scoring for Weighted-Sum

- The score for a category is the sum of the similarity scores between the point to be classified and all of its k-neighbors that belong to the given category.
- To restate: $score(c \mid x) = \sum_{d \in kNN\, of\ x} sim(x,d)\, I(d,c)$

  where *x* is the new point; *c* is a class (*e.g. black* or *white);*
  d is a classified point among the k-nearest neighbors of *x*;
  sim(x,d) is the similarity between *x* and *d;*
  I(d,c) = 1 iff point *d* belongs to class *c*;
  I(d,c) = 0 otherwise.

The kth Nearest Neighbor Rule (Fix and Hodges, 1951)

- Define a metric to measure “closeness” between any two points.
- Fix *k* (empirically chosen).
- Given a new point *x* and a training set of classified points.
  - Find the *k* nearest neighbors (kNN) to *x* in the training set.
  - Classify *x* as class *y* if more of the nearest neighbors are in class *y* than in any other classes (*majority vote*).

kNN for Text Categorization (Yang, SIGIR-1994)

- Represent documents as points (vectors).
- Define a similarity measure for pairwise documents.
- Tune parameter *k* for optimizing classification effectiveness.
- Choose a voting scheme (e.g., weighted sum) for scoring categories
- Threshold on the scores for classification decisions.

Thresholding for Classification Decisions

- Alternative thresholding strategies:
  - Rcut: For each document to be categorized, rank candidate categories by score, and assign YES to the top-*m* categories (where *m* is some fixed number).
  - Pcut: Applies only when we have a whole batch of documents to be categorized. Make the category assignments proportional to the category distribution in the training set (i.e. if 1/4th of the training documents were in the category "Coffee Maker" then we will assign 1/4th of the documents in this batch to the "Coffee Maker" category).
  - Scut: For each category, choose a threshold score (empirically). Any document with a category score that surpasses its respective threshold will be predicted to be a member of that category.

Possible Similarity Measures

- Cosine similarity

$$\cos(\vec{x}, \vec{y}) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \times \sqrt{\sum_i y_i^2}}$$

- Euclidean distance
- Kernel functions
- Kullback-Leibler distance (distance between two probability distributions)

Key Components (revisited)

- Functional definition of "similarity"
  - e.g. cos, Euclidean, kernel functions, ...
- How many neighbors do we consider?
  - Value of *k* determined *empirically*
- Does each neighbor get the same weight?
  - Weighted-sum or not
- All categories in neighborhood? Most frequent only? How do we make the final decision?
  - Rcut, Pcut, or Scut

Approaches to Automated Text Categorization

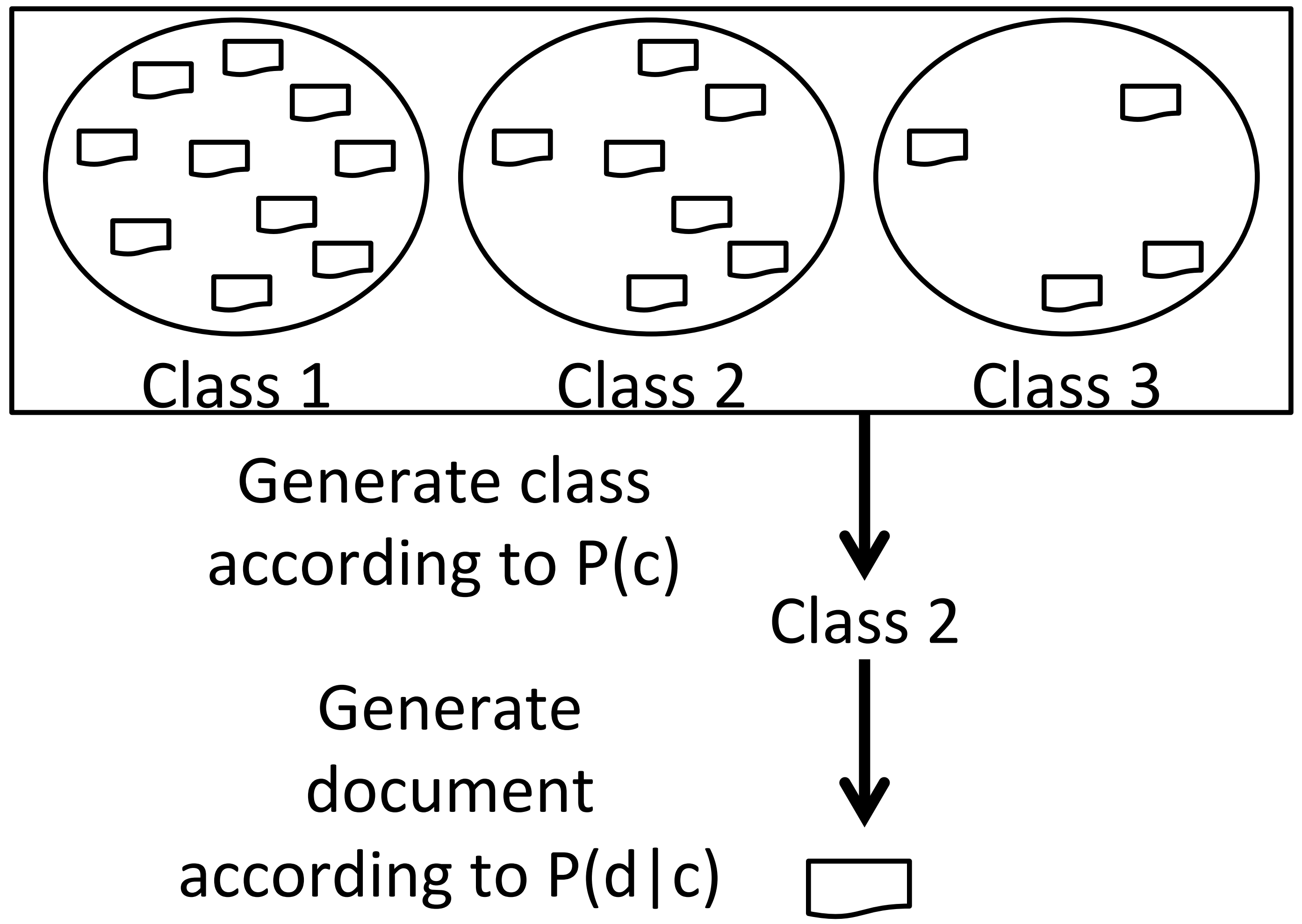
- Regression based on Least Squares Fit (1991)
- Nearest Neighbor Classification (1992)
- Bayesian Probabilistic Models (1992)
- Symbolic Rule Induction (1994)
- Neural Networks (1995)
- Rocchio approach (traditional IR, 1996)
- Support Vector Machines (1997)
- Boosting or Bagging (1997)
- Hierarchical Language Modeling (1998)
- First-Order-Logic Rule Induction (1999)
- Maximum Entropy (1999)
- Hidden Markov Models (1999)
- Error-Correcting Output Coding (1999)

...

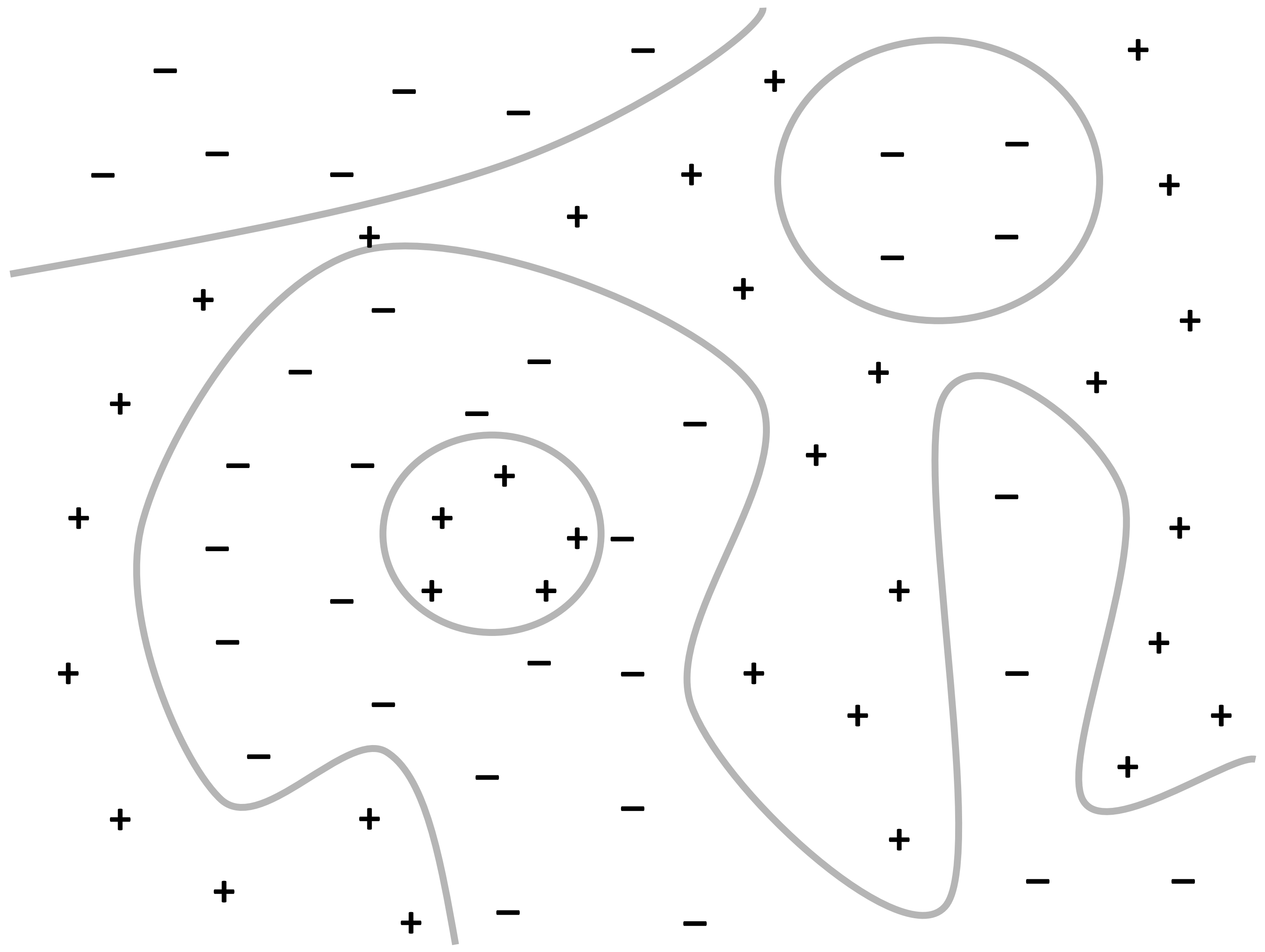
Evaluating Classifiers

- Common classification metrics
  - Accuracy (precision at rank 1)
  - Precision
  - Recall
  - F-measure
  - ROC curve analysis
- Differences from IR metrics
  - “Relevant” replaced with “classified correctly”
  - Microaveraging more commonly used

Naïve Bayes Generative Process


Class 1
Class 2
Class 3
Generate class according to P(c)
Class 2
Generate document according to P(d|c)

Nearest Neighbor Classification



Feature Selection

- Document classifiers can have a very large number of features
  - 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

Information Gain

- Information gain is a commonly used feature selection measure based on information theory
  - It tells how much "information" is gained if we observe some feature
- Rank features by information gain and then train model using the top *K* (*K* is typically small)
- The information gain for a Multiple-Bernoulli Naïve Bayes classifier is computed as:

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

Classification Applications

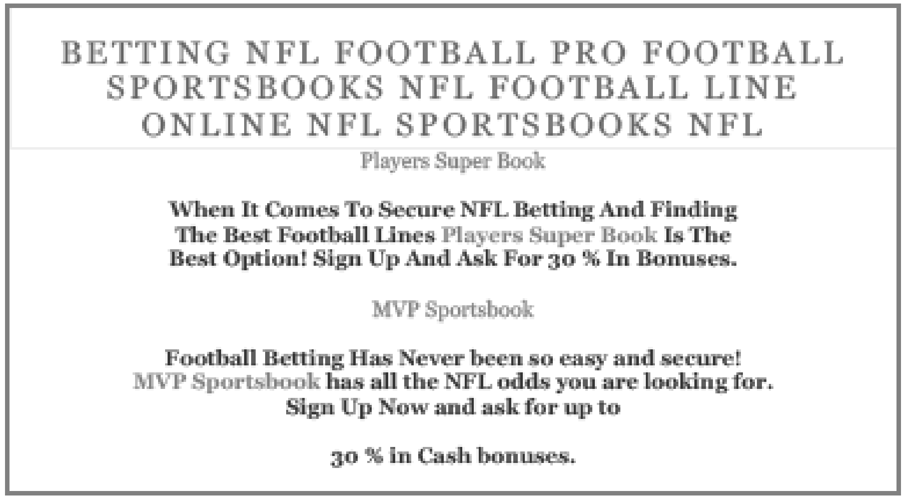
- Classification is widely used to enhance search engines
- Example applications
  - Spam detection
  - Sentiment classification
  - Semantic classification of advertisements
  - Many others not covered here!

Spam, Spam, Spam

- Classification is widely used to detect various types of spam
- There are many types of spam
  - Link spam
  - Term spam

Spam Example

Website:


BETTING NFL FOOTBALL PRO FOOTBALL SPORTSBOOKS NFL FOOTBALL LINE ONLINE NFL SPORTSBOOKS NFL

Players Super Book

When It Comes To Secure NFL Betting And Finding The Best Football Lines Players Super Book Is The Best Option! Sign Up And Ask For 30 % In Bonuses.

MVP Sportsbook

Football Betting Has Never been so easy and secure! MVP Sportsbook has all the NFL odds you are looking for. Sign Up Now and ask for up to

30 % in Cash bonuses.

Term spam:

pro football sportsbooks nfl football line online nfl sportsbooks nfl football gambling odds online pro nfl betting pro nfl gambling online nfl football spreads offshore football gambling online nfl gamblibg spreads online football gambling line online nfl betting nfl sportsbook online online nfl betting spreads betting nfl football online online football wagering online gambling online gambling football online nfl football betting odds offshore football sportsbook online nfl football gambling …

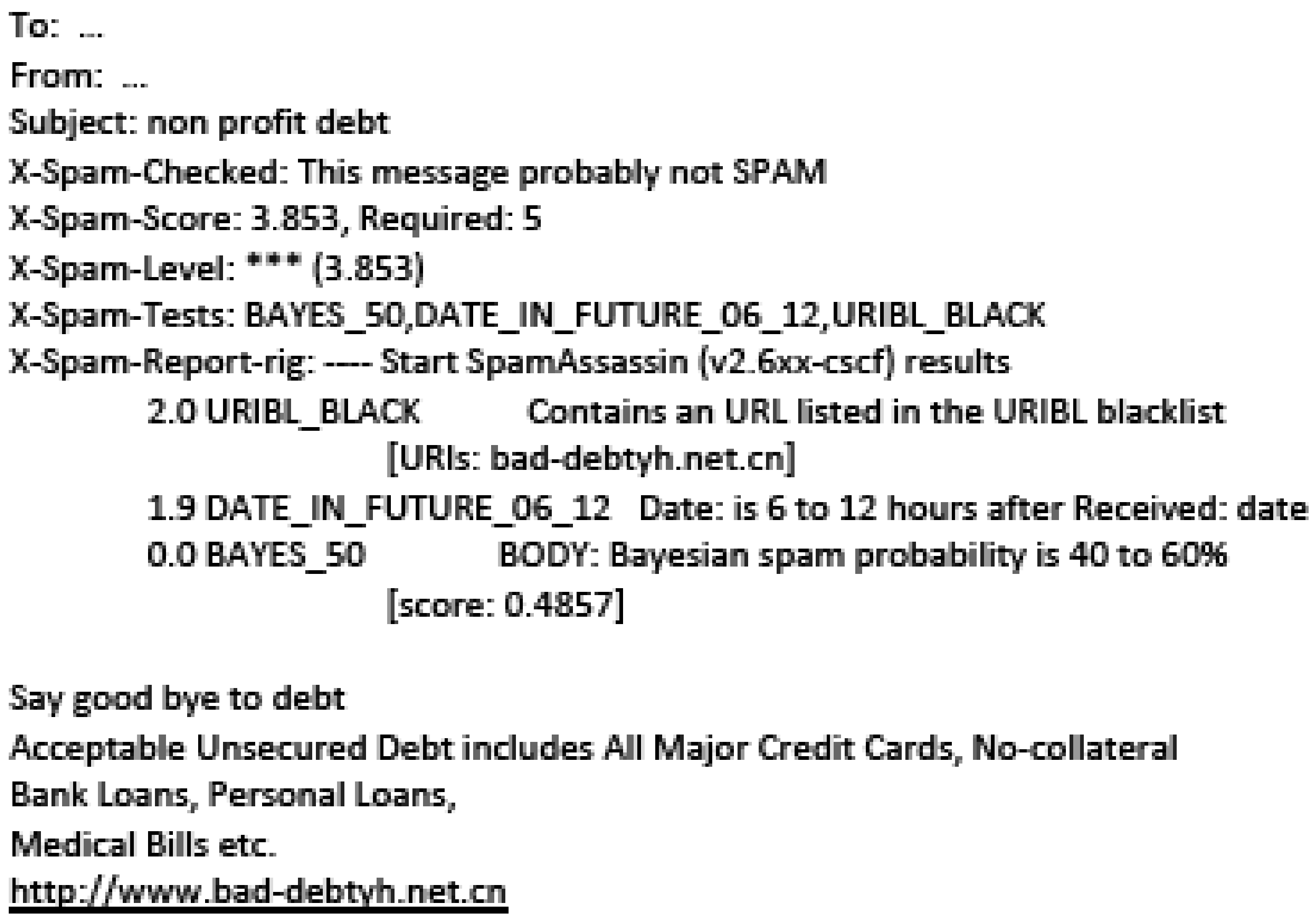
Link spam:

MVP Sportsbook Football Gambling Beverly Hills Football Sportsbook
Players SB Football Wagering Popular Poker Football Odds
Virtual Bookmaker Football Lines V Wager Football Spreads
Bogarts Casino Football Point Spreads Gecko Casino Online Football Betting
Jackpot Hour Online Football Gambling MVP Casino Online Football Wagering
Toucan Casino NFL Betting Popular Poker NFL Gambling
All Tracks NFL Wagering Bet Jockey NFL Odds
Live Horse Betting NFL Lines MVP Racebook NFL Point Spreads
Popular Poker NFL Spreads Bogarts Poker NFL Sportsbook …

Spam Detection

- Useful features
  - Unigrams
  - Formatting (invisible text, flashing, etc.)
  - IP address
- Different features are useful for different spam detection tasks
- Email and web page spam are by far the most widely studied, well understood, and easily detected types of spam

Example Spam Assassin Output



```
To:  ...
From:  ...
Subject: non profit debt
X-Spam-Checked: This message probably not SPAM
X-Spam-Score: 3.853, Required: 5
X-Spam-Level: *** (3.853)
X-Spam-Tests: BAYES_50,DATE_IN_FUTURE_06_12,URIBL_BLACK
X-Spam-Report-rig: ---- Start SpamAssassin (v2.6xx-cscf) results
        2.0 URIBL_BLACK           Contains an URL listed in the URIBL blacklist
                      [URIs: bad-debtyh.net.cn]
        1.9 DATE_IN_FUTURE_06_12   Date: is 6 to 12 hours after Received: date
        0.0 BAYES_50             BODY: Bayesian spam probability is 40 to 60%
                      [score: 0.4857]

Say good bye to debt
Acceptable Unsecured Debt includes All Major Credit Cards, No-collateral
Bank Loans, Personal Loans,
Medical Bills etc.
http://www.bad-debtyh.net.cn
```

Sentiment

- Blogs, online reviews, and forum posts are often opinionated
- Sentiment classification attempts to automatically identify the polarity of the opinion
  - Negative opinion
  - Neutral opinion
  - Positive opinion
- Sometimes the strength of the opinion is also important
  - “Two stars” vs. “four stars”
  - Weakly negative vs. strongly negative

Classifying Sentiment

- Useful features
  - Unigrams
  - Bigrams
  - Part of speech tags
  - Adjectives
- SVMs with unigram features have been shown to be outperform hand built rules

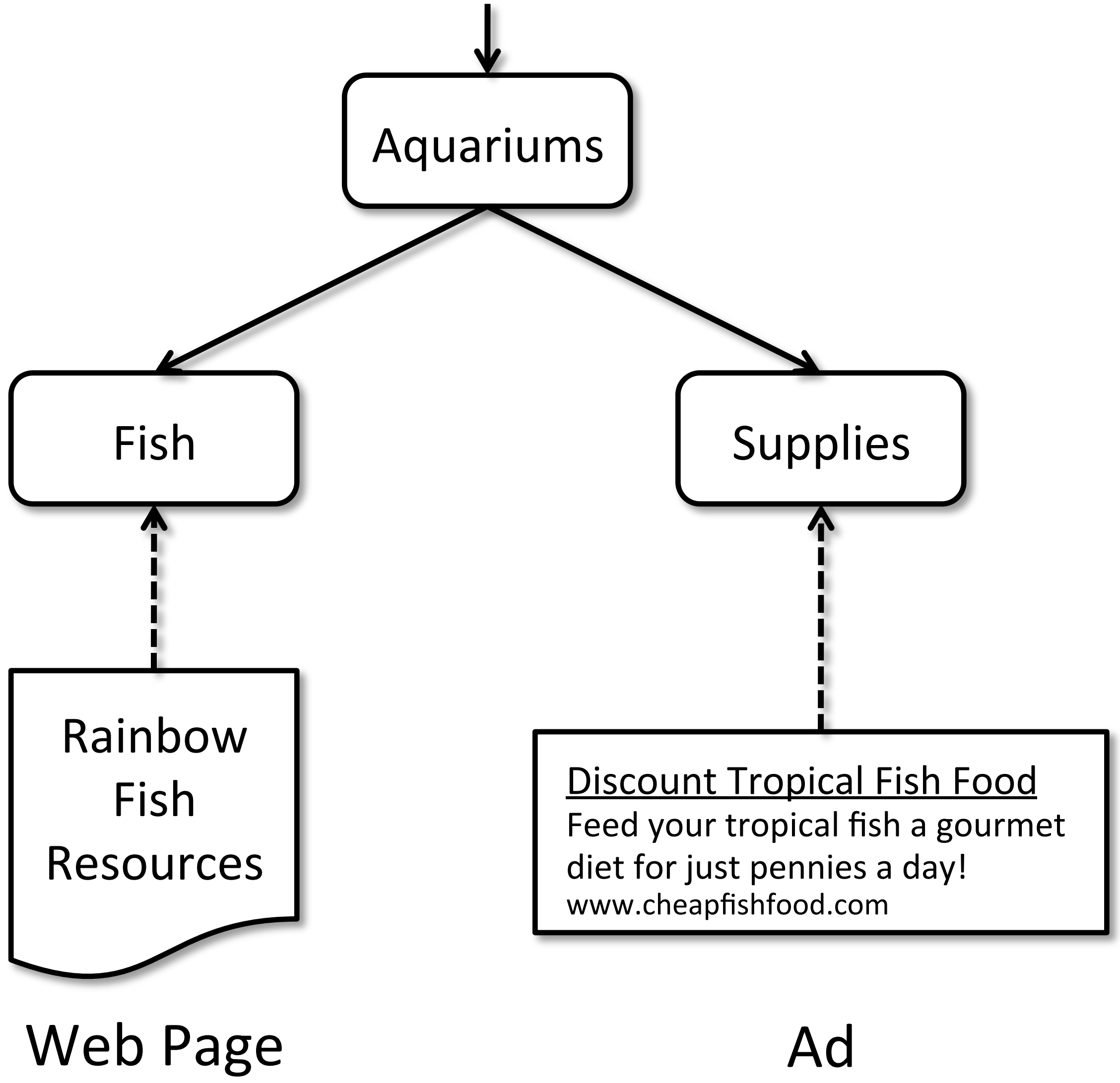
Classifying Online Ads

- Unlike traditional search, online advertising goes beyond “topical relevance”
- A user searching for ‘tropical fish’ may also be interested in pet stores, local aquariums, or even scuba diving lessons
- These are semantically related, but not topically relevant!
- We can bridge the semantic gap by classifying ads and queries according to a semantic hierarchy

Semantic Classification

- Semantic hierarchy ontology
  - Example: Pets / Aquariums / Supplies
- Training data
  - Large number of queries and ads are manually classified into the hierarchy
- Nearest neighbor classification has been shown to be effective for this task
- Hierarchical structure of classes can be used to improve classification accuracy

Semantic Classification



Aquariums
Fish
Supplies
Rainbow Fish Resources
Discount Tropical Fish Food
Feed your tropical fish a gourmet diet for just pennies a day!
www.cheapfishfood.com
Web Page
Ad

Clustering

- A set of unsupervised algorithms that attempt to find latent structure in a set of items
- Goal is to identify groups (clusters) of similar items
- 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 and how similarity is defined!

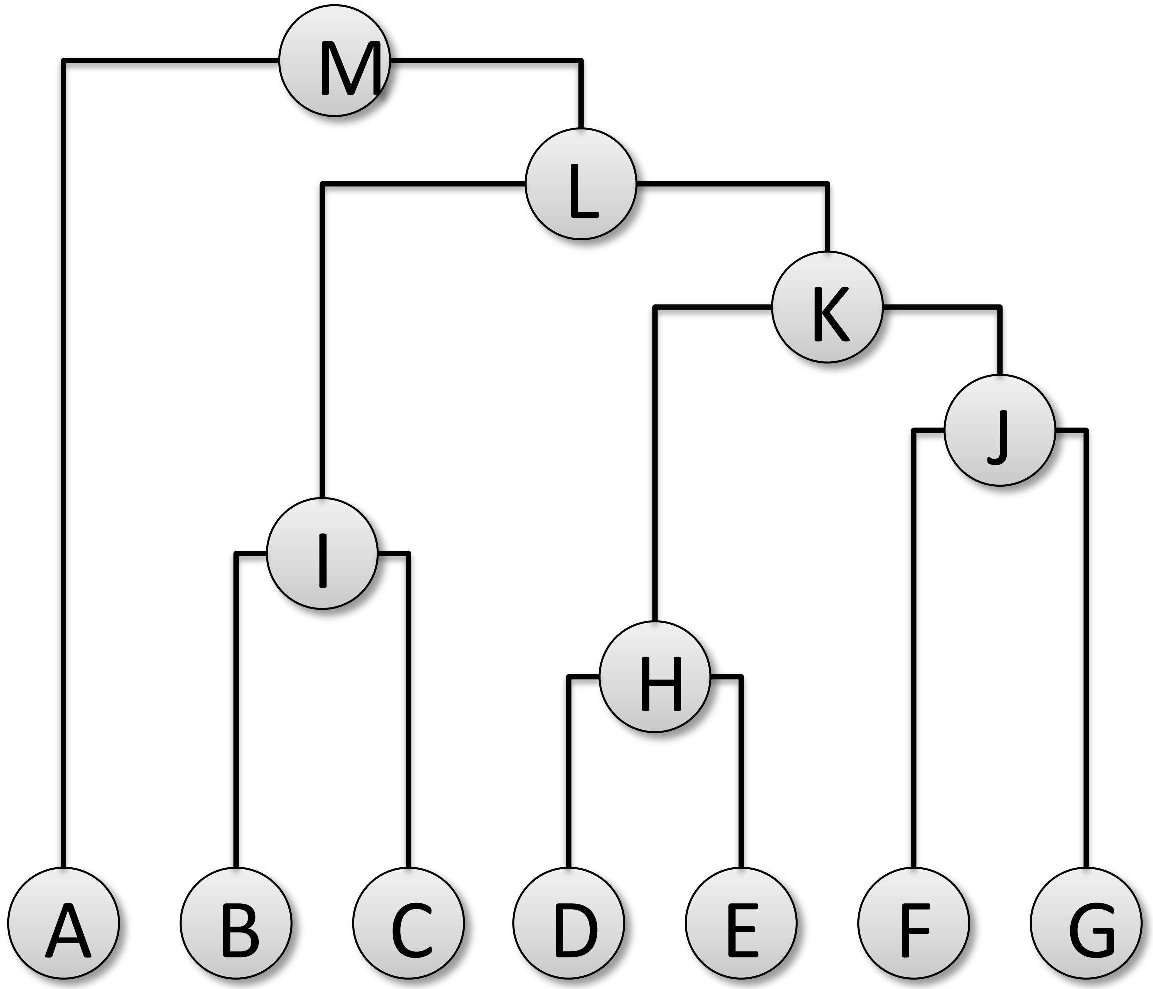
Clustering

- 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)
  3. Determine what makes a “good” clustering
  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
  - 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* (# items) singleton clusters
- Two types of hierarchical clustering
  - Divisive (“top down”)
  - Agglomerative (“bottom up”)
- Hierarchy can be visualized as a *dendogram*

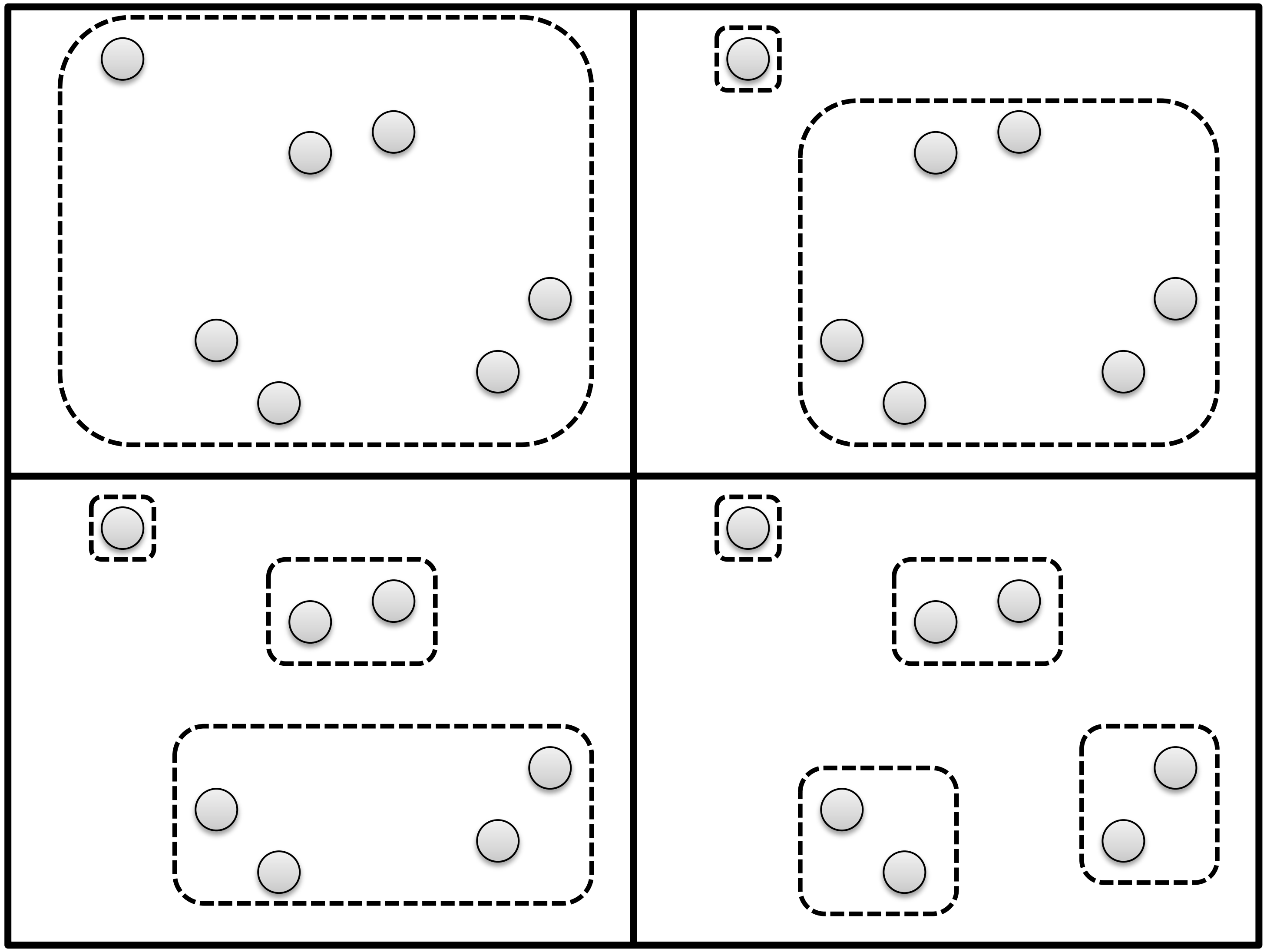
Example Dendrogram



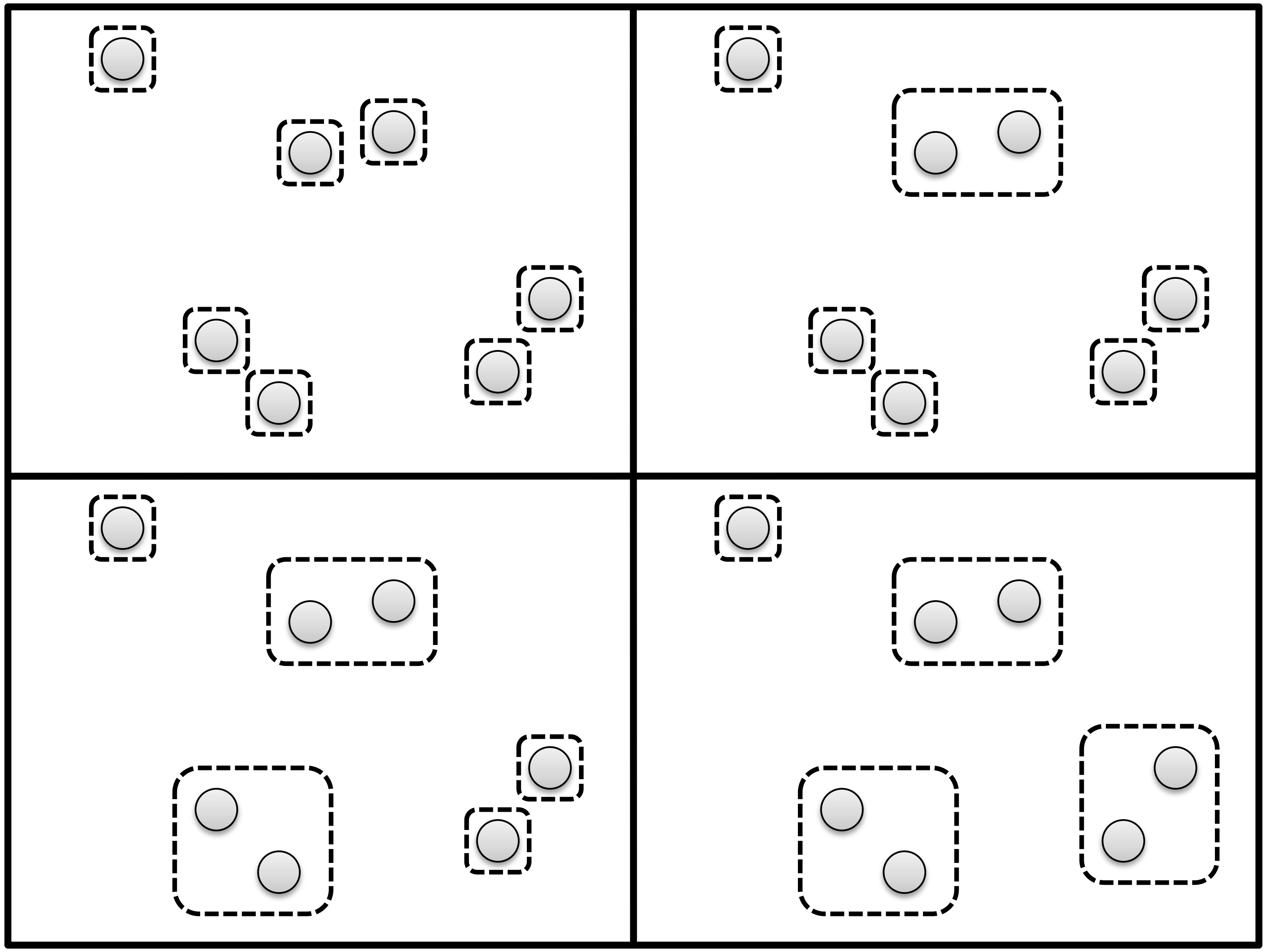
Divisive and Agglomerative Hierarchical Clustering

- Divisive
  - Start with a single cluster consisting of all of the items
  - Until only singleton clusters exist…
    - **Divide** an existing cluster into two new clusters
- Agglomerative
  - Start with *N* (# items) singleton clusters
  - Until a single cluster exists…
    - **Combine** two existing cluster into a new cluster
- How do we know how to divide or combined clusters?
  - Define a division or combination cost
  - Perform the division or combination with the lowest cost

Divisive Hierarchical Clustering



Agglomerative Hierarchical Clustering



Clustering Costs

- 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

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

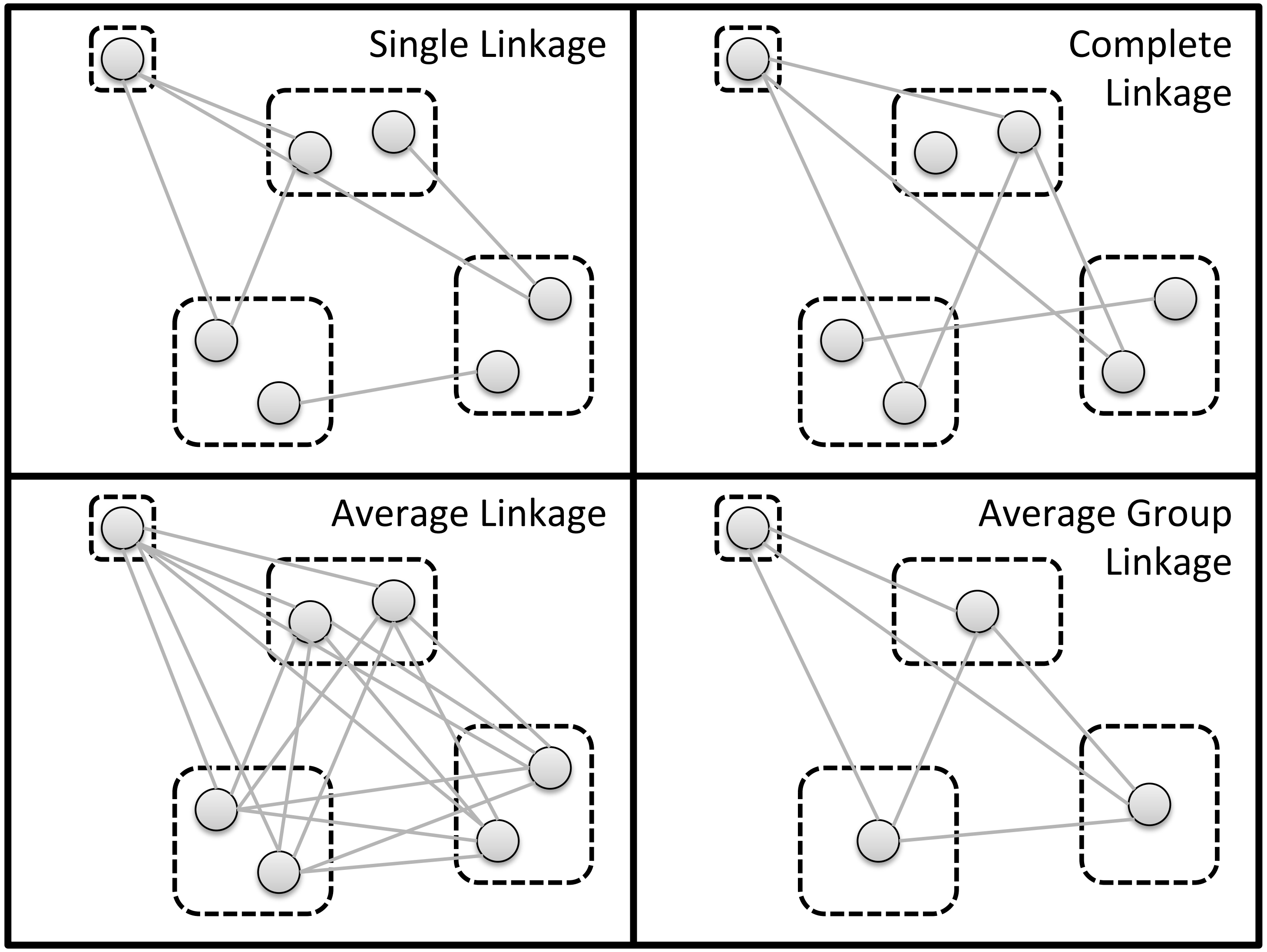
- Average linkage

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

- Average group linkage

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

Clustering Strategies



Agglomerative Clustering Algorithm

Algorithm 1 Agglomerative Clustering

```
procedure AgglomerativeCluster($X_1, \ldots, X_N$, $K$)
    $A[1], \ldots, A[N] \leftarrow 1, \ldots, N$
    $ids \leftarrow \{1, \ldots, N\}$
    for $c = N$ to $K$ do
        $bestcost \leftarrow \infty$
        $bestclusterA \leftarrow$ undefined
        $bestclusterB \leftarrow$ undefined
        for $i \in ids$ do
            for $j \in ids - \{i\}$ do
                $c_{i,j} \leftarrow COST(C_i, C_j)$
                if $c_{i,j} < bestcost$ then
                    $bestcost \leftarrow c_{i,j}$
                    $bestclusterA \leftarrow i$
                    $bestclusterB \leftarrow j$
                end if
            end for
        end for
        $ids \leftarrow ids - \{bestClusterA\}$
        for $i = 1$ to $N$ do
            if $A[i]$ is equal to $bestClusterA$ then
                $A[i] \leftarrow bestClusterB$
            end if
        end for
    end for
end procedure
```

K-Means Clustering

- Hierarchical clustering constructs a hierarchy of clusters
- K-means always maintains exactly *K* clusters
  - Clusters represented as centroids (“center 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 1
- Tends to converge quickly
- Can be sensitive to choice of initial centroids
- Must choose *K*

K-Means Clustering Algorithm

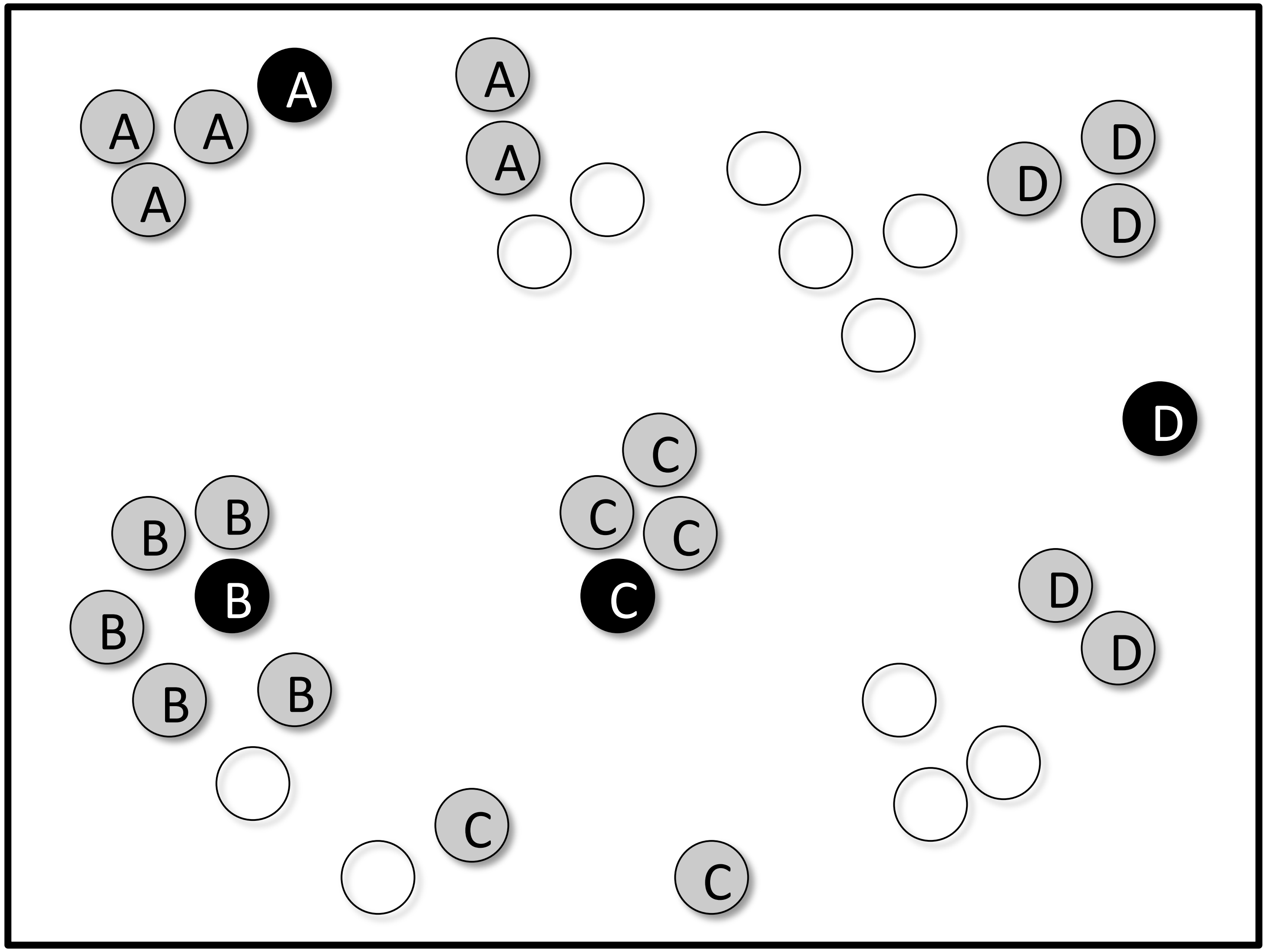
Algorithm 1 K-Means Clustering

```
 1: procedure KMeansCluster($X_1, \ldots, X_N$, $K$)
 2:     $A[1], \ldots, A[N] \leftarrow$ initial cluster assignment
 3:     repeat
 4:         $change \leftarrow false$
 5:         for $i = 1$ to $N$ do
 6:             $\hat{k} \leftarrow \arg\min_k dist(X_i, C_k)$
 7:             if $A[i]$ is not equal $\hat{k}$ then
 8:                 $A[i] \leftarrow \hat{k}$
 9:                 $change \leftarrow true$
10:             end if
11:         end for
12:     until $change$ is equal to $false$ return $A[1], \ldots, A[N]$
13: end procedure
```

K-Nearest Neighbor Clustering

- Hierarchical and K-Means clustering partition items into clusters
  - Every item is in exactly one cluster
- K-Nearest neighbor clustering forms one cluster per item
  - The cluster for item *j* consists of *j* and *j*'s *K* nearest neighbors
  - Clusters now overlap

5-Nearest Neighbor Clustering



Evaluating Clustering

- Evaluating clustering is challenging, since it is an ***unsupervised*** learning task
- If labels exist, can use standard IR metrics, such as precision and recall
- Other measures such as "cluster precision", which is defined as:

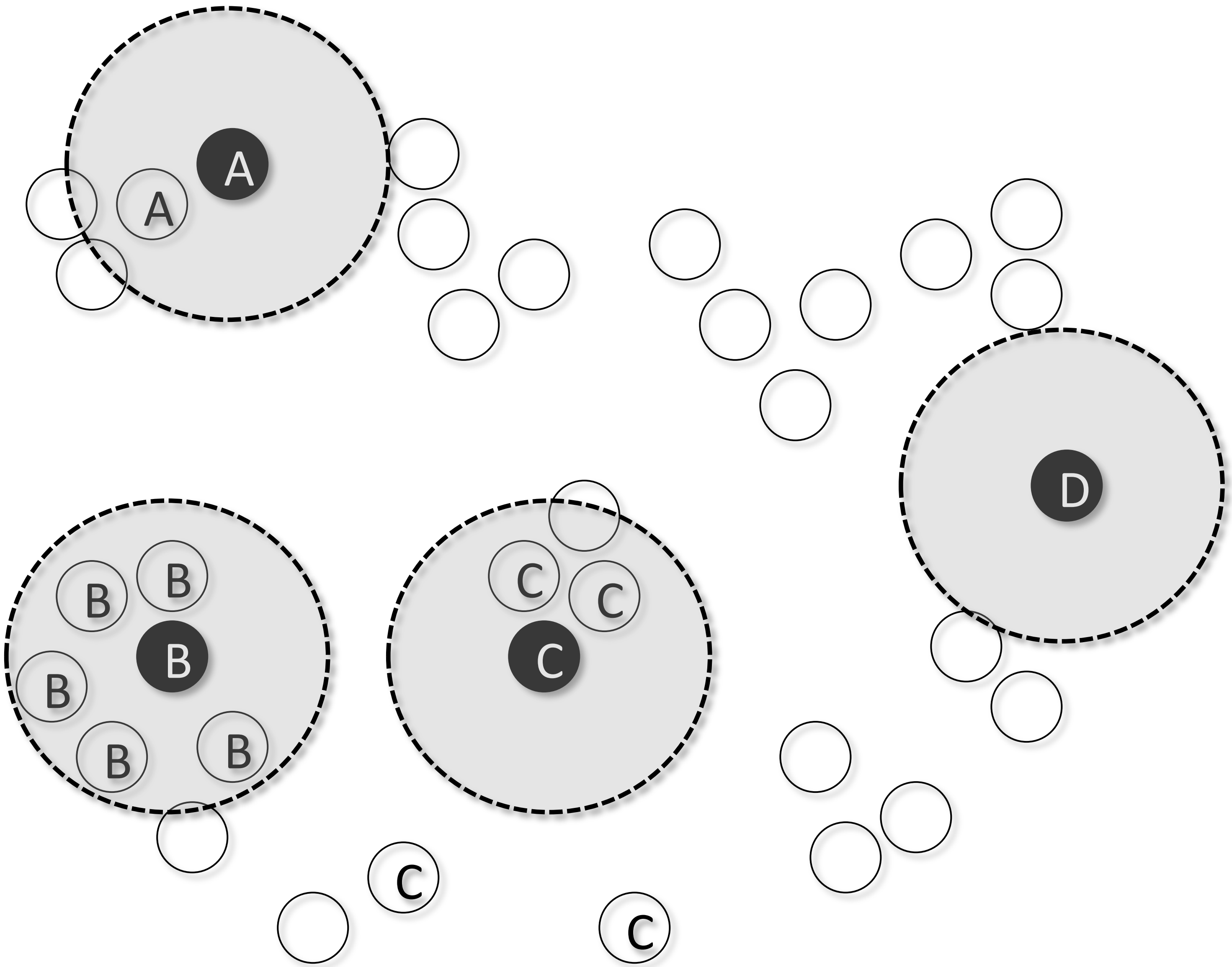
$$ClusterPrecision = \frac{\sum_{i=1}^{K} |\text{MaxClass}(C_i)|}{N}$$

- Another option is to evaluate clustering as part of an end-to-end system

How to Choose K?

- K-means and hierarchical clustering require us to choose *K*, the number of clusters
- No theoretically appealing way of choosing *K*
- Depends on the application and data
- Can use hierarchical clustering and choose the best level of the hierarchy to use
- Can use adaptive *K* for K-nearest neighbor clustering
  - Define a ‘ball’ around each item
- Difficult problem with no clear solution

Adaptive Nearest Neighbor Clustering



Clustering and Search

- Cluster hypothesis
  - "Closely associated documents tend to be relevant to the same requests" – van Rijsbergen '79
- Tends to hold in practice, but not always

But Does It Help Retrieval?

- Cluster retrieval

$$P(Q|C_j) = \prod_{i=1}^{n} P(q_i|C_j)$$

- Smoothing with hard clusters

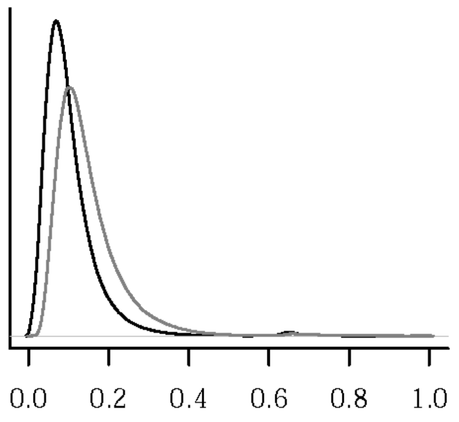
$$P(w|D) = (1 - \lambda - \delta)\frac{f_{w,D}}{|D|} + \delta\frac{f_{w,C_j}}{|C_j|} + \lambda\frac{f_{w,Coll}}{|Coll|}$$

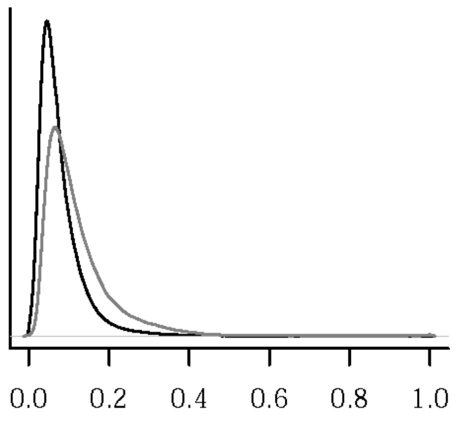
- Smoothing with soft clusters

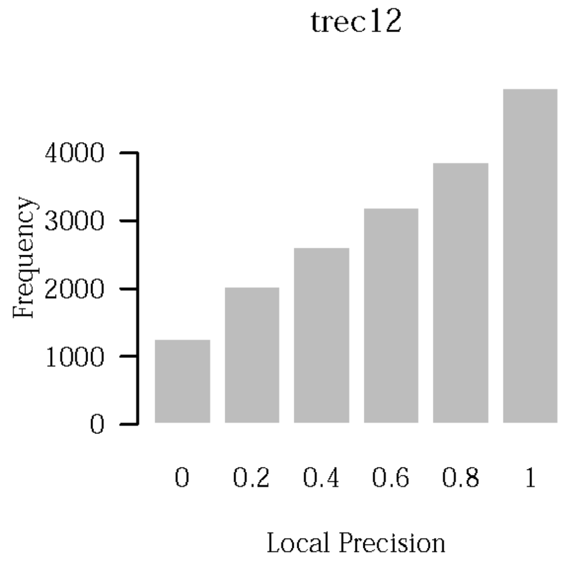
$$P(w|D) = (1 - \lambda - \delta)\frac{f_{w,D}}{|D|} + \delta\sum_{C_j}\frac{f_{w,C_j}}{|C_j|}P(D|C_j) + \lambda\frac{f_{w,Coll}}{|Coll|}$$

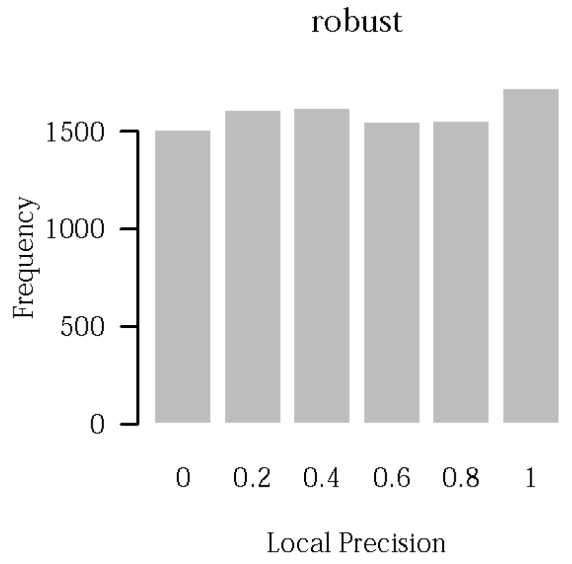
- Last two more effective (cf. topic models)

Testing the Cluster Hypothesis


0.0
0.2
0.4
0.6
0.8
1.0


0.0
0.2
0.4
0.6
0.8
1.0


trec12
Frequency
Local Precision


robust
Frequency
Local Precision