Information Processing Technology of Internet of Things

Chapter 3
Information Retrieval

Wu Liu

Beijing Key Lab of Intelligent Telecomm. Software and Multimedia
Beijing University of Posts and Telecommunications

3.5 Text Classification

3.5.1 Introduction

Introduction

- Ancient problem for librarians
  - storing documents for later retrieval
- With larger collections, need to **label the documents**
  - assign an unique identifier to each document
  - does not allow findings documents on a **subject or topic**
- To allow searching documents on a subject or topic
  - group documents by common topics
  - name these groups with meaningful labels
  - each labeled group is call a **class**

Introduction

- **Text classification**
  - process of associating documents with classes
  - if classes are referred to as **categories**
    - process is called **text categorization**
  - we consider classification and categorization the same process
  - a means to organize information
- Related problem: partition docs into subsets, no labels
  - since each subset has no label, it is not a class
  - instead, each subset is called a **cluster**
  - the partitioning process is called **clustering**
    - we consider clustering as a simpler variant of text classification

Introduction

- Consider a large engineering company
  - thousands of documents are produced
  - if properly organized, they can be used for business decisions
  - to organize large document collection, text classification is used

Machine Learning

- Machine Learning
  - algorithms that learn patterns in the data
  - patterns learned allow making predictions relative to new data
  - learning algorithms use training data and can be of three types
    - supervised learning
    - unsupervised learning
    - semi-supervised learning

The Text Classification Problem

- A classifier can be formally defined
  - D: a collection of documents
  - C = {c1, c2, . . . , cL}: a set of L classes with their respective labels
  - a text classifier is a binary function $F : D \times C \rightarrow \{0, 1\}$, which assigns to each pair [dj , cp], $dj \in D$ and $cp \in C$, a value of
    - 1, if dj is a member of class cp
    - 0, if dj is *not a member of class cp*
- Broad definition, admits supervised and unsupervised algorithms
- For high accuracy, use **supervised algorithm**
  - multi-label: one or more labels are assigned to each document
  - single-label: a single class is assigned to each document

The Text Classification Problem

- Classification function F
  - defined as binary function of document-class pair [dj , cp]
  - can be modified to compute degree of membership of dj in cp
    - documents as *candidates* for membership in class cp
    - candidates sorted by decreasing values of F(dj , cp)

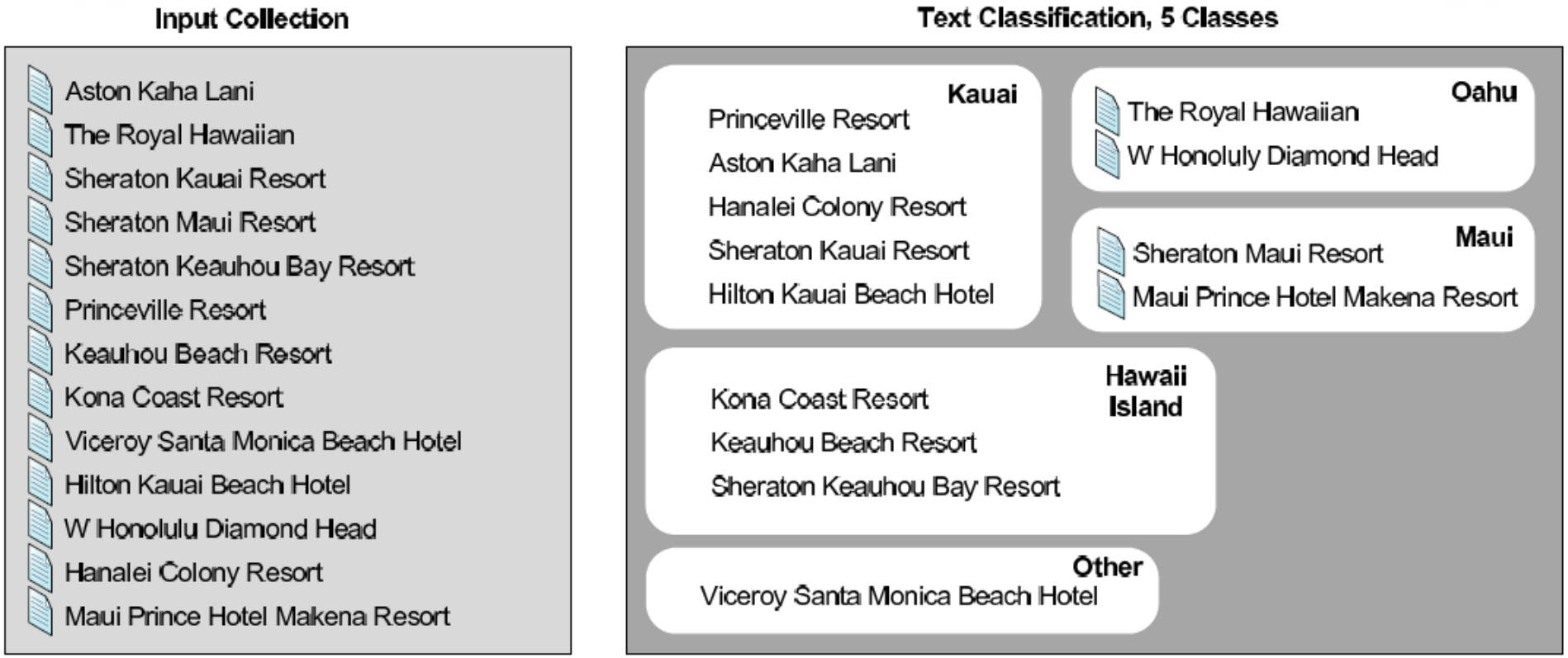
3.5.2 Unsupervised Algorithms

Clustering

- Input data
  - set of documents to classify
  - not even class labels are provided
- Task of the classifier
  - separate documents into subsets (clusters) automatically
  - separating procedure is called **clustering**

Clustering

- Clustering of hotel Web pages in Hawaii
- To obtain classes, assign labels to clusters



Clustering

- **Class labels** can be generated automatically
  - but are different from labels specified by humans
  - usually, of much lower quality
  - thus, solving the whole classification problem with no human intervention is hard
- If class labels are provided, clustering is more effective

K-means Clustering

- **Input:** number K of clusters to be generated
- Each cluster represented by its documents **centroid**
- K-Means algorithm:
  - partition docs among the K clusters
    - each document assigned to cluster with closest centroid
  - recompute centroids
  - repeat process until centroids do not change

K-means in Batch Mode

- **Batch mode**: all documents classified before recomputing centroids
- Let document dj be represented as vector $\vec{d}_j$

$$\vec{d}_j = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$$

where

- wi,j : weight of term ki in document dj
- t: size of the vocabulary

K-means in Batch Mode

- 1. **Initial step.**
  - select K docs randomly as centroids (of the K clusters)

$$\vec{\Delta}_p = \vec{d}_j$$

- 2. **Assignment Step.**
  - assign each document to cluster with closest centroid
  - distance function computed as inverse of the similarity
  - similarity between dj and cp, use cosine formula

$$sim(d_j, c_p) = \frac{\vec{\Delta}_p \bullet \vec{d}_j}{|\vec{\Delta}_p| \times |\vec{d}_j|}$$

K-means in Batch Mode

- **Update Step.**
  - recompute centroids of each cluster cp

$$\vec{\Delta}_p = \frac{1}{size(c_p)} \sum_{\vec{d}_j \in c_p} \vec{d}_j$$

- **Final Step.**
  - repeat **assignment** and **update** steps until no centroid changes

Hierarchical Clustering

- Goal: to create a hierarchy of clusters by either
  - decomposing a large cluster into smaller ones, or
  - agglomerating previously defined clusters into larger ones

Hierarchical Clustering

- General hierarchical clustering algorithm
  - 1. Input
    - a set of N documents to be clustered
    - an N × N similarity (distance) matrix
  - 2. Assign each document to its own cluster
    - N clusters are produced, containing one document each
  - 3. Find the two closest clusters
    - merge them into a single cluster
    - number of clusters reduced to N − 1
  - 4. Recompute distances between new cluster and each old cluster
  - 5. Repeat steps 3 and 4 until one single cluster of size N is produced

Hierarchical Clustering

- Step 4 introduces notion of similarity or distance between two clusters
- Method used for computing **cluster distances defines** three variants of the algorithm
  - *single-link*
  - *complete-link*
  - *average-link*

Hierarchical Clustering

- dist(cp, cr): distance between two clusters cp and cr
- dist(dj , dl): distance between docs dj and dl
- **Single-Link Algorithm**

$$dist(c_p, c_r) = \min_{\forall\ d_j \in c_p, d_l \in c_r} dist(d_j, d_l)$$

- **Complete-Link Algorithm**

$$dist(c_p, c_r) = \max_{\forall\ d_j \in c_p, d_l \in c_r} dist(d_j, d_l)$$

- **Average-Link Algorithm**

$$dist(c_p, c_r) = \frac{1}{n_p * n_r} \sum_{d_j \in c_p} \sum_{d_l \in c_r} dist(d_j, d_l)$$

Naive Text Classification

- Classes and their labels are given as input
  - no training examples
- **Naive Classification**
  - Input:
    - collection D of documents
    - set C = {c1, c2, . . . , cL} of L classes and their labels
  - Algorithm: associate one or more classes of C with each doc in D
    - match document terms to class labels
    - permit partial matches
    - improve coverage by defining alternative class labels i.e., **synonyms**

Naive Text Classification

- **Text Classification by Direct Match**
  - 1. Input:
    - D: collection of documents to classify
    - C = {c1, c2, . . . , cL}: set of L classes with their labels
  - 2. Represent
    - each document dj by a weighted vector $\vec{d}_j$
    - each class cp by a weighted vector $\vec{c}_p$ (use the labels)
  - 3. For each document dj $\in$ D do
    - retrieve classes cp $\in$ C whose labels contain terms of dj
    - for each pair [dj , cp] retrieved, compute vector ranking as

$$sim(d_j, c_p) = \frac{\vec{d}_j \bullet \vec{c}_p}{|\vec{d}_j| \times |\vec{c}_p|}$$

    - associate dj classes cp with highest values of sim(dj , cp)

3.5.3 Supervised Algorithms

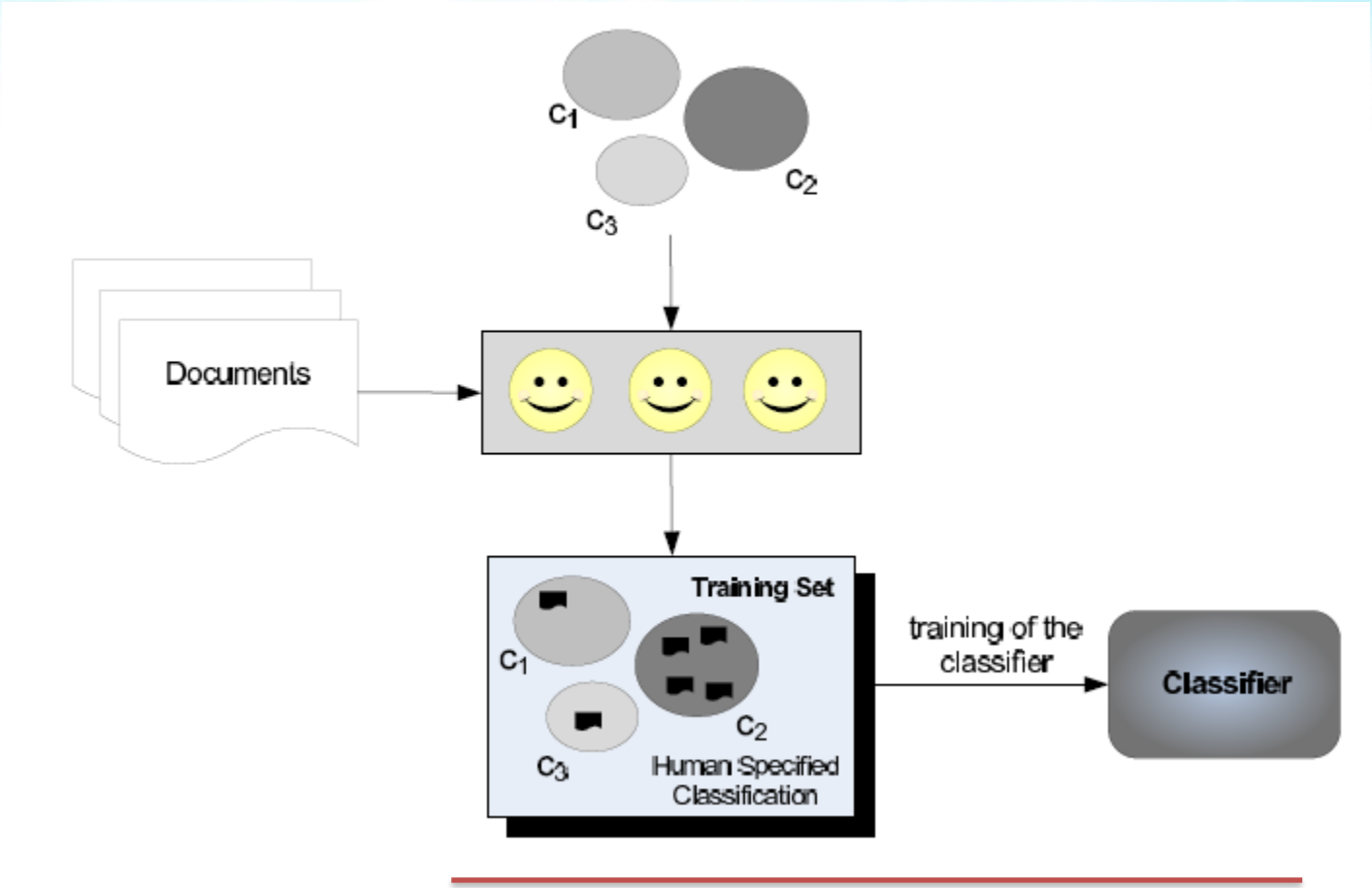
Supervised Algorithms

- Depend on a **training set**
  - $Dt \subset D$: subset of training documents
  - $T : Dt \times C \rightarrow \{0, 1\}$: training set function

    Assigns to each pair [dj , cp], $dj \in Dt$ and $cp \in C$ a value of
    - 1, if $dj \in cp$, according to judgement of human specialists
    - 0, if $dj \notin cp$, according to judgement of human specialists
  - Training set function T is used to fine tune the classifier

Supervised Algorithms

- The training phase of a classifier



Supervised Algorithms

- To evaluate the classifier, use a **test set**
  - subset of docs with no intersection with training set
  - classes to documents determined by human specialists
- **Evaluation** is done in a two steps process
  - use classifier to assign classes to documents in test set
  - compare classes assigned by classifier with those specified by human specialists
- Once classifier has been trained and validated
  - can be used to classify new and unseen documents
  - if classifier is well tuned, classification is highly effective

Decision Trees

- Training set used to build **classification rules**
  - organized as paths in a tree
  - tree paths used to classify documents outside training set
  - rules, amenable to human interpretation, facilitate interpretation of results

Decision Trees: Basic Technique

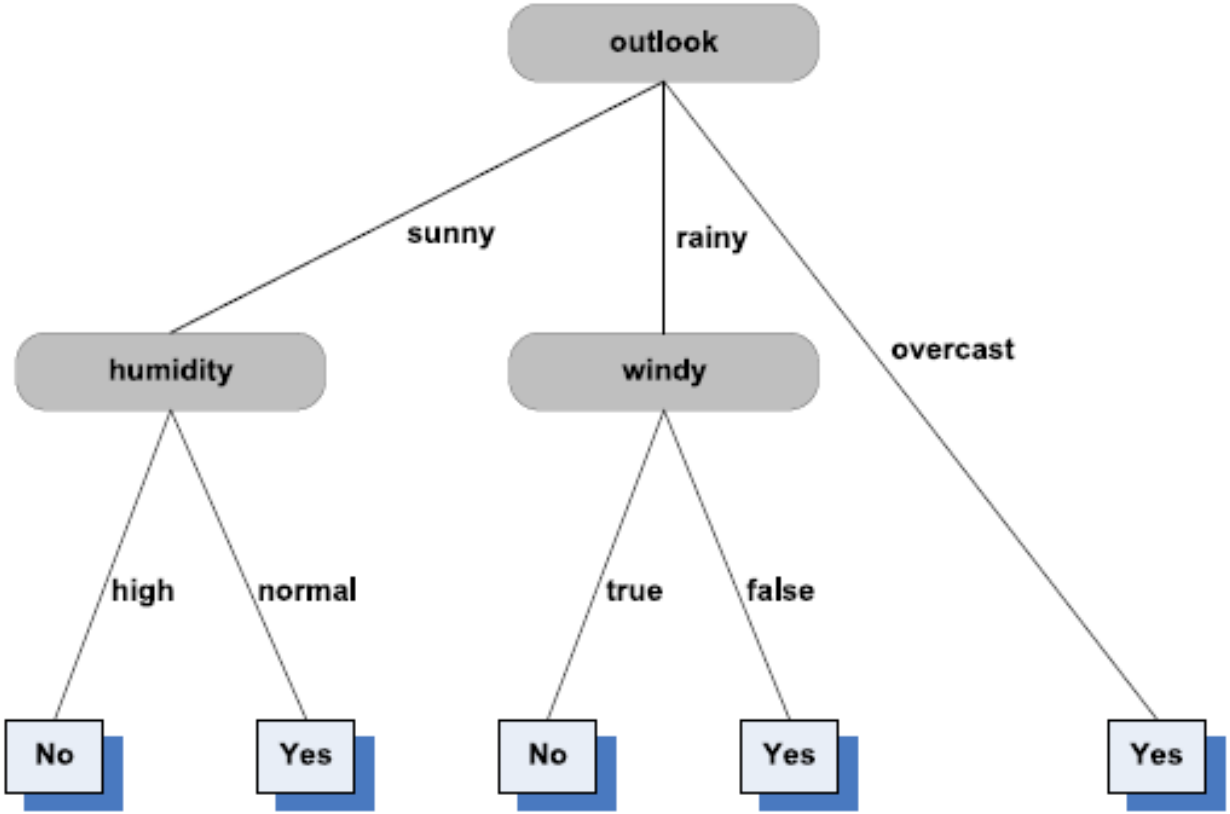
- Consider the small relational database below

	Id	Play	Outlook	Temperature	Humidity	Windy
Training set	1	yes	rainy	cool	normal	false
	2	no	rainy	cool	normal	true
	3	yes	overcast	hot	high	false
	4	no	sunny	mild	high	false
	5	yes	rainy	cool	normal	false
	6	yes	sunny	cool	normal	false
	7	yes	rainy	cool	normal	false
	8	yes	sunny	hot	normal	false
	9	yes	overcast	mild	high	true
	10	no	sunny	mild	high	true
Test Instance	11	?	sunny	cool	high	false

- Decision Tree (DT) allows predicting values of a given attribute

Decision Trees: Basic Technique

- DT to predict values of attribute Play
  - Given: Outlook, Humidity, Windy



Decision Trees: Classification of Documents

- For document classification
  - with each internal node associate an index term
  - with each leave associate a document class
  - with the edges associate binary predicates that indicate presence/absence of index term

Decision Trees: Classification of Documents

- V: a set of nodes
- Tree T = (V,E, r): an acyclic graph on V where
  - $E \subseteq V \times V$ is the set of edges
  - Let edge(vi, vj) $\in$ E
    - vi is the father node
    - vj is the child node
  - r $\in$ V is called the root of T
  - *I*: set of all internal nodes
  - $\overline{I}$: set of all leaf nodes

Decision Trees: Classification of Documents

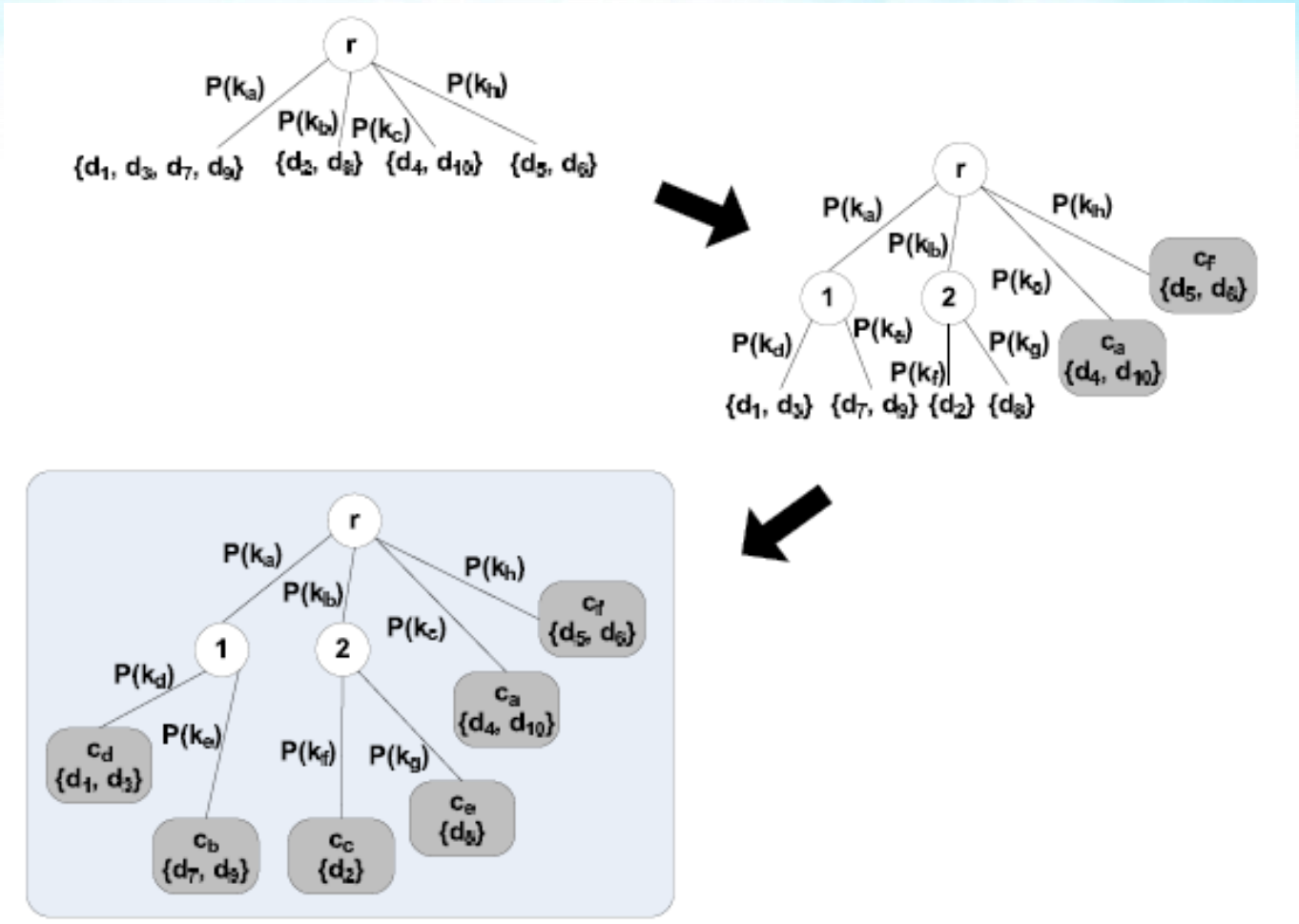
- Define
  - K = {k1, k2, . . . , kt}: set of index terms of a doc collection
  - C: set of all classes
  - P: set of logical predicates on the index terms
- DT = (V,E; r; lI , lL, lE): a six-tuple where
  - (V ;E; r): a tree whose root is r
  - lI : I → K: a function that associates with each internal node of the tree one or more index terms
  - lL : $\overline{I}$ → C: a function that associates with each non-internal (leaf) node a class cp ∈ C
  - lE : E → P: a function that associates with each edge of the tree a logical predicate from P

Decision Trees: Classification of Documents

- Decision tree model for class cp can be built using a recursive splitting strategy
  - first step: associate all documents with the root
  - second step: select index terms that provide a good separation of the documents
  - third step: repeat until tree complete

Decision Trees: Classification of Documents

- Terms ka, kb, kc, and kh have been selected for first split



Decision Trees: Classification of Documents

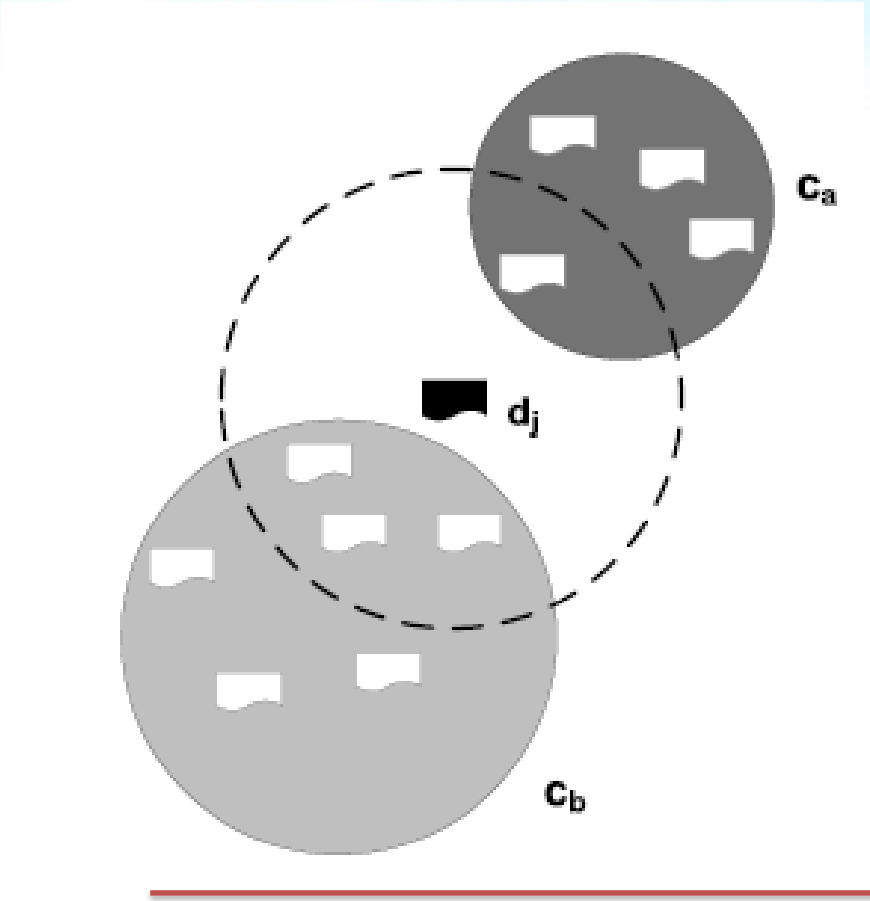
- To select splitting terms use
  - information gain or entropy
- Selection of terms with high information gain tends to
  - increase number of branches at a given level, and
  - reduce number of documents in each resultant subset
  - yield smaller and less complex decision trees
- Problem: missing or unknown values
  - appear when document to be classified does not contain some terms used to build the DT
  - not clear which branch of the tree should be traversed
- Solution:
  - delay construction of tree until new document is presented for classification
  - build tree based on features presented in this document, avoiding the problem

The kNN Classifier

- kNN (k-nearest neighbor): **on-demand or lazy classifier**
  - lazy classifiers do not build a classification model a priori
  - classification done when new document dj is presented
  - based on the classes of the k nearest neighbors of dj
    - determine the k nearest neighbors of dj in a training set
    - use the classes of these neighbors to determine a class for dj

The kNN Classifier

- An example of a 4-NN classification process



The kNN Classifier

- kNN: to each document-class pair [dj , cp] assign a score

$$S_{d_j,c_p} = \sum_{d_t \in N_k(d_j)} similarity(d_j, d_t) \times \mathcal{T}(d_t, c_p)$$

where

- Nk(dj): set of the k nearest neighbors of dj in training set
- similarity(dj , dt): cosine formula of Vector model (for instance)
- T (dt, cp): training set function returns
  - 1, if dt belongs to class cp
  - 0, otherwise

- Classifier assigns to dj class(es) cp with highest score(s)

The kNN Classifier

- Problem with kNN: performance
  - classifier has to compute distances between document to be classified and all training documents
  - another issue is how to choose the “best” value for k

The Naive Bayes Classifier

- Probabilistic classifiers
  - assign to each document-class pair [dj , cp] a probability $P(c_p|\vec{d}_j)$

$$P(c_p|\vec{d}_j) = \frac{P(c_p) \times P(\vec{d}_j|c_p)}{P(\vec{d}_j)}$$

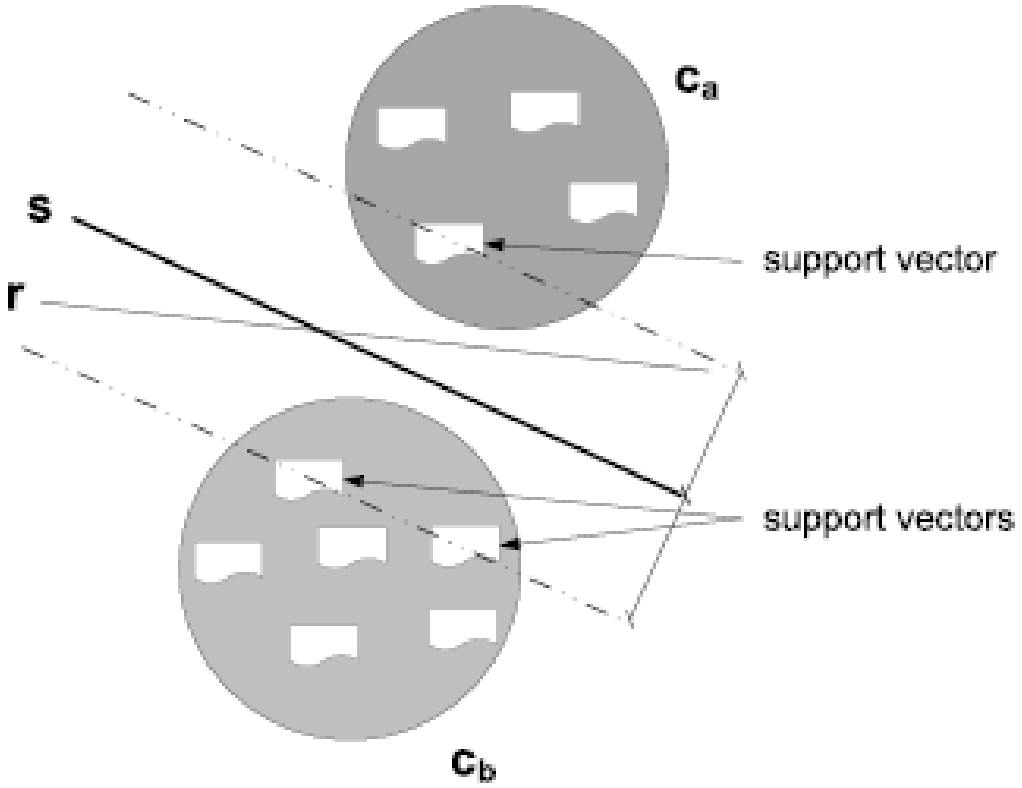
  - $P(\vec{d}_j)$: probability that randomly selected doc is $\vec{d}_j$
  - P(c_p): probability that randomly selected doc is in class c_p
- assign to new and unseen docs classes with highest probability estimates

SVM Classifier : SVM Basic Technique

- **Support Vector Machines (SVMs)**
  - a vector space method for binary classification problems
  - documents represented in t-dimensional space
  - find a decision surface **(**hyperplane**)** that best separate documents of two classes
  - new document classified by its position relative to hyperplane

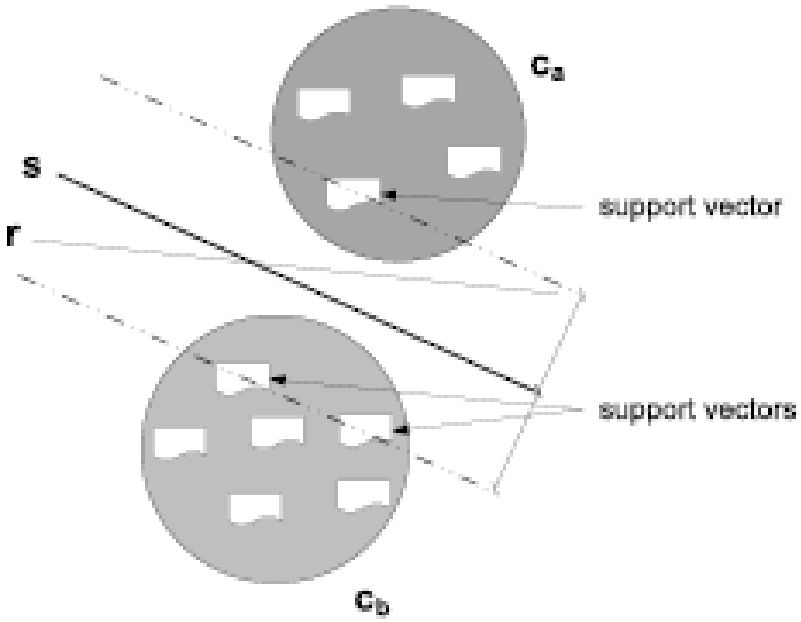
SVM Classifier : SVM Basic Technique

- Simple 2D example: training documents linearly separable



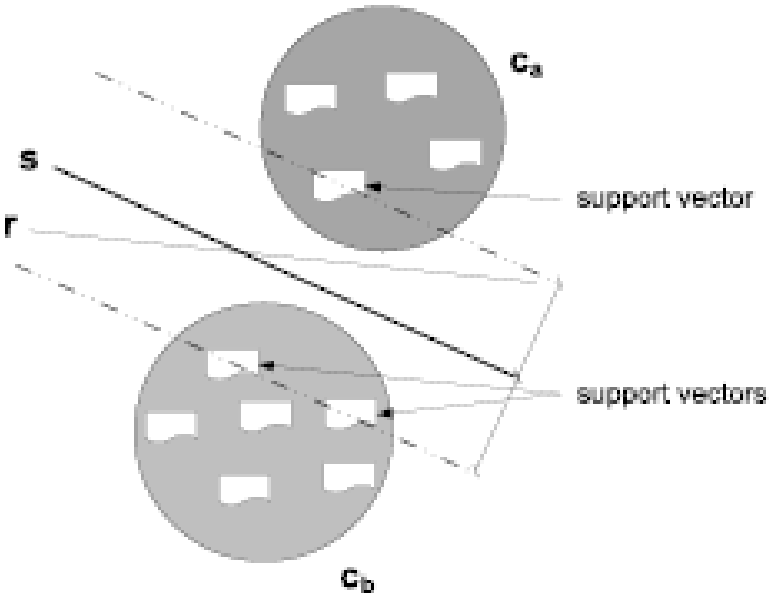
SVM Classifier : SVM Basic Technique

- **Line s—The Decision Hyperplane**
  - maximizes distances to closest docs of each class
  - it is the best separating hyperplane
- **Delimiting Hyperplanes**
  - parallel dashed lines that delimit region where to look for a solution



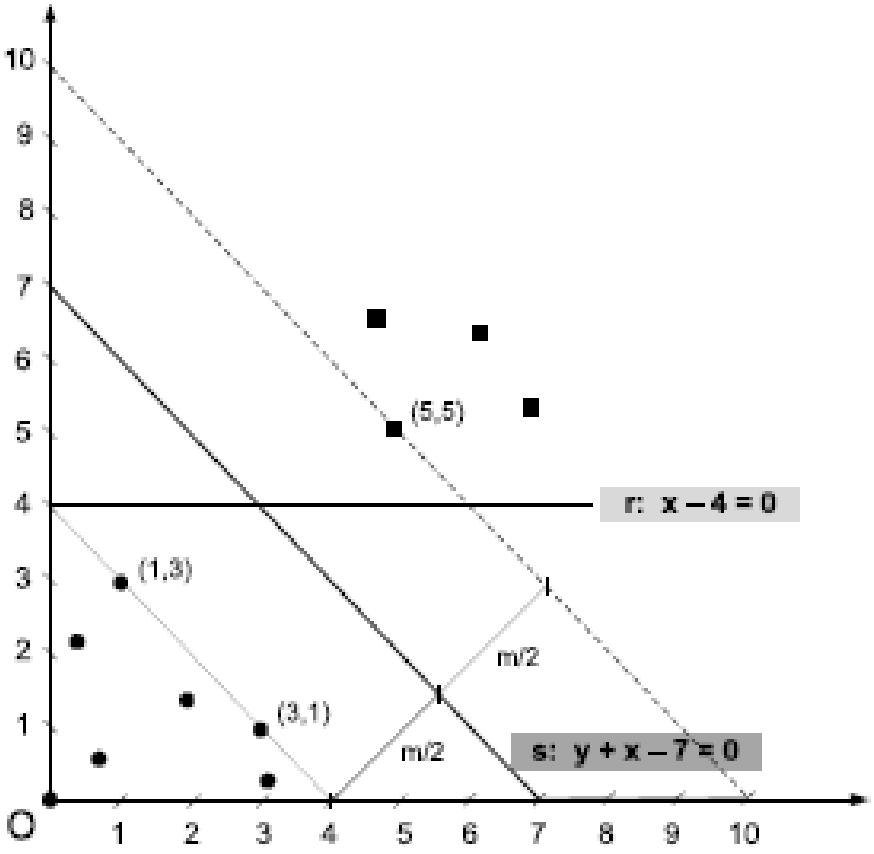
SVM Classifier : SVM Basic Technique

- Lines that cross the delimiting hyperplanes
  - candidates to be selected as the decision hyperplane
  - lines that are parallel to delimiting hyperplanes: best candidates
- **Support vectors:** documents that belong to, and define, the delimiting hyperplanes



SVM Classifier : SVM Basic Technique

- Our example in a 2-dimensional system of coordinates



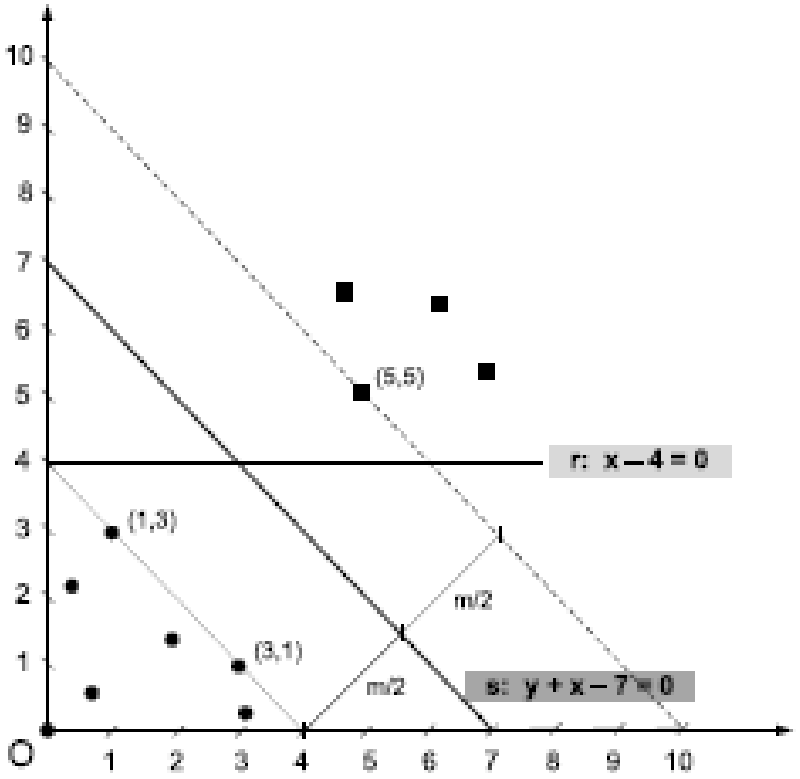
SVM Classifier : SVM Basic Technique

- Let,
  - Hw: a hyperplane that separates docs in classes ca and cb
  - ma: distance of Hw to the closest document in class ca
  - mb: distance of Hw to the closest document in class cb
  - ma + mb: **margin** m of the SVM
- The **decision hyperplane** maximizes the margin m

SVM Classifier : SVM Basic Technique

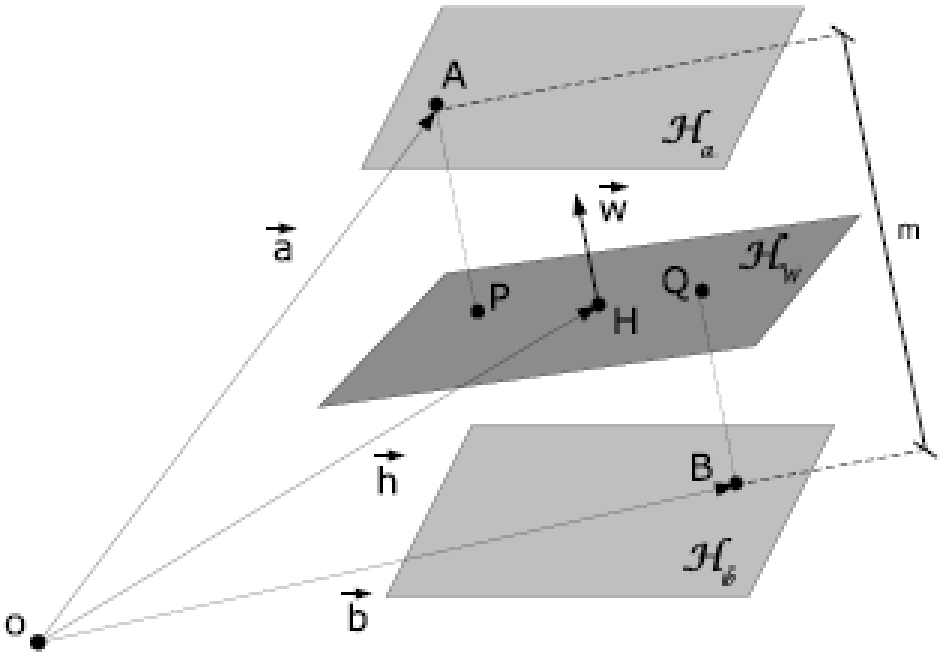
- Hyperplane r : x − 4 = 0 separates docs in two sets
  - its distances to closest docs in either class is 1
  - thus, its margin m is 2

- Hyperplane s : y +x−7 = 0 has margin equal to $3\sqrt{2}$
  - maximum for this case
  - s is the decision hyperplane



SVM Technique – Formalization

- **The SVM optimization problem:** given support vectors such as $\vec{a}$ and $\vec{b}$, find hyperplane Hw that maximizes margin m



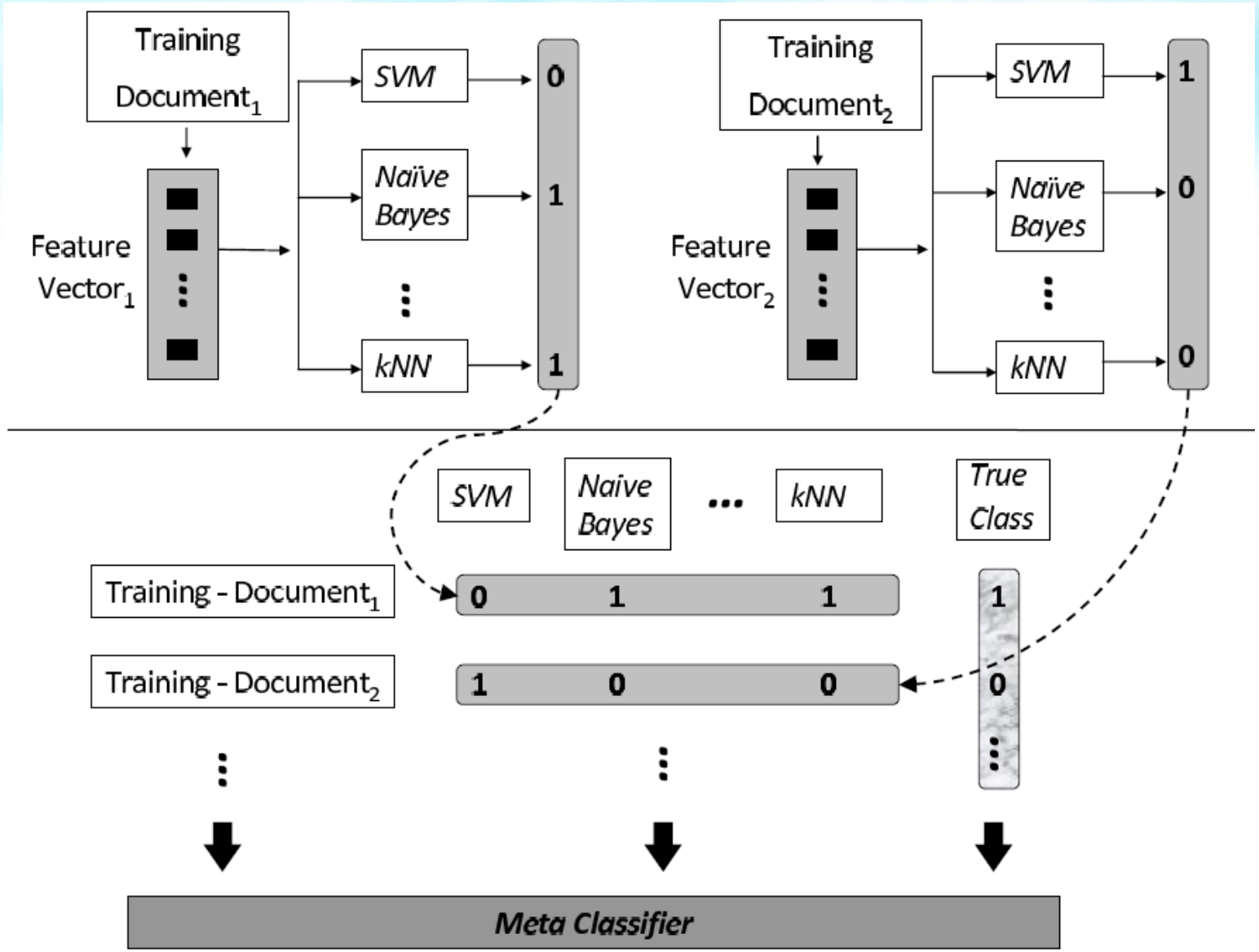
SVM with Multiple Classes

- SVMs can only take binary decisions
  - a document belongs or not to a given class
- With multiple classes
  - reduce the multi-class problem to binary classification
  - natural way: one binary classification problem per class
- To classify a new document dj
  - run classification for each class
  - each class cp paired against all others
  - classes of dj : those with largest margins

Ensemble Classifiers

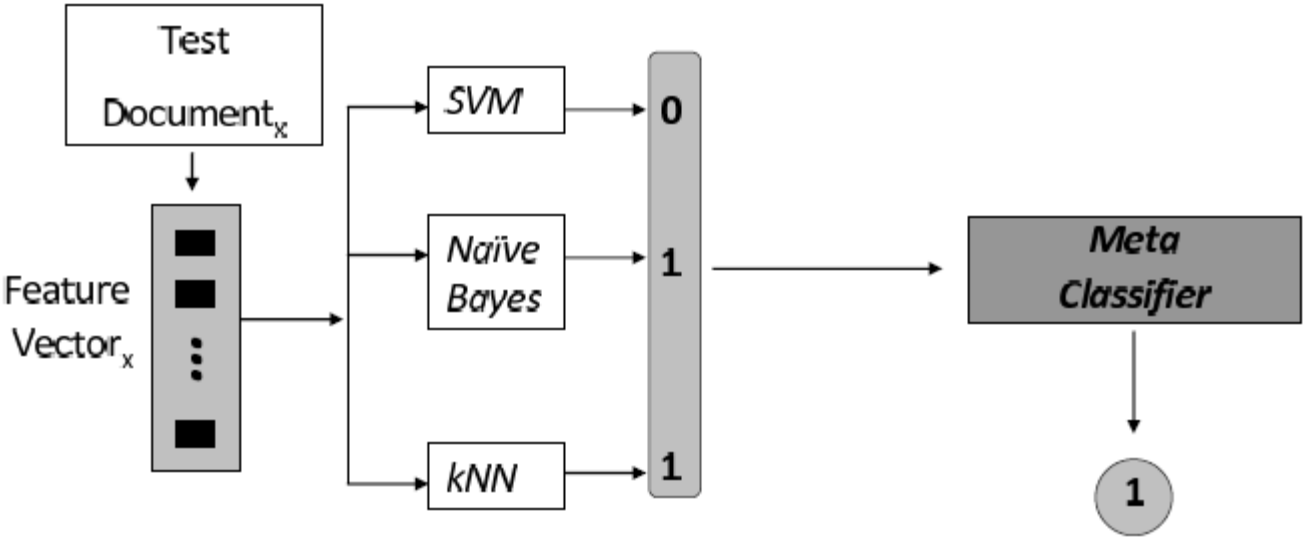
- Combine predictions of distinct classifiers to generate a new predictive score
- Ideally, results of higher precision than those yielded by constituent classifiers
- Two ensemble classification methods:
  - **stacking**
  - **boosting**

Stacking-based Ensemble



Stacking-based Classifiers

- **Stacking method**: learn function that combines predictions of individual classifiers



Stacking-based Classifiers

- With each document-class pair [dj , cp] in training set
  - associate predictions made by distinct classifiers
- Instead of predicting class of document dj
  - predict the classifier that best predicts the class of dj , or
  - combine predictions of base classifiers to produce better results
- Advantage: errors of a base classifier can be counter-balanced by hits of others

Boosting-based Classifiers

- Boosting: classifiers to be combined are generated by several iterations of a **same learning technique**
- Focus: missclassified training documents
- At each interaction
  - each document in training set is given a weight
  - weights of incorrectly classified documents are increased at each round
- After n rounds
  - outputs of trained classifiers are combined in a weighted sum
  - weights are the error estimates of each classifier

3.5.4 Feature Selection or Dimensionality Reduction

Feature Selection

- Large feature space
  - might render document classifiers impractical
- Classic solution
  - select a subset of all features to represent the documents
  - called **feature selection**
    - reduces dimensionality of the documents representation

Term-Class Incidence Table

- Feature selection
  - dependent on statistics on term occurrences inside docs and classes
- Let
  - Dt: subset composed of all training documents
  - Nt: number of documents in Dt
  - ti: number of documents from Dt that contain term ki
  - C = {c1, c2, . . . , cL}: set of all L classes
  - T : Dt $\times$ C $\rightarrow$ [0, 1]: a training set function

Term-Class Incidence Table

- Term-class incidence table

Case	Docs in c_p	Docs not in c_p	Total
Docs that contain k_i	$n_{i,p}$	$n_i - n_{i,p}$	n_i
Docs that do not contain k_i	$n_p - n_{i,p}$	$N_t - n_i - (n_p - n_{i,p})$	$N_t - n_i$
All docs	n_p	$N_t - n_p$	N_t

- ni,p: # docs that contain ki and are classified in cp
- ni − ni,p: # docs that contain ki but are not in class cp
- np: total number of training docs in class cp
- np − ni,p: number of docs from cp that do not contain ki

Term-Class Incidence Table

- Given term-class incidence table above, define

Probability that $k_i \in d_j$: $P(k_i) = \frac{n_i}{N_t}$

Probability that $k_i \notin d_j$: $P(\overline{k}_i) = \frac{N_t - n_i}{N_t}$

Probability that $d_j \in c_p$: $P(c_p) = \frac{n_p}{N_t}$

Probability that $d_j \notin c_p$: $P(\overline{c}_p) = \frac{N_t - n_p}{N_t}$

Probability that $k_i \in d_j$ and $d_j \in c_p$: $P(k_i, c_p) = \frac{n_{i,p}}{N_t}$

Probability that $k_i \notin d_j$ and $d_j \in c_p$: $P(\overline{k}_i, c_p) = \frac{n_p - n_{i,p}}{N_t}$

Probability that $k_i \in d_j$ and $d_j \notin c_p$: $P(k_i, \overline{c}_p) = \frac{n_i - n_{i,p}}{N_t}$

Probability that $k_i \notin d_j$ and $d_j \notin c_p$: $P(\overline{k}_i, \overline{c}_p) = \frac{N_t - n_i - (n_p - n_{i,p})}{N_t}$

Feature Selection by Doc Frequency

- Let Kth be a threshold on term document frequencies
- Feature Selection by Term Document Frequency
  - retain all terms ki for which ni $\geq$ Kth
  - discard all others
  - recompute doc representations to consider only terms retained
- Even if simple, method allows reducing dimensionality of space with basically no loss in effectiveness

Feature Selection by Mutual Information

- **Mutual information**
  - relative entropy between distributions of two random variables
- If variables are independent, mutual information is zero
  - knowledge of one of the variables does not allow inferring anything about the other variable

Mutual Information

- Mutual information across all classes

$$I(k_i, c_p) = \log \frac{P(k_i, c_p)}{P(k_i)P(c_p)} = \log \frac{\frac{n_{i,p}}{N_t}}{\frac{n_i}{N_t} \times \frac{n_p}{N_t}}$$

- That is,

$$\begin{aligned} MI(k_i, C) &= \sum_{p=1}^{L} P(c_p)\ I(k_i, c_p) \\ &= \sum_{p=1}^{L} \frac{n_p}{N_t} \log \frac{\frac{n_{i,p}}{N_t}}{\frac{n_i}{N_t} \times \frac{n_p}{N_t}} \end{aligned}$$

Mutual Information

- Alternative: maximum term information over all classes

$$I_{max}(k_i, C) = max_{p=1}^{L} I(k_i, c_p)$$

$$= max_{p=1}^{L} \log \frac{\frac{n_{i,p}}{N_t}}{\frac{n_i}{N_t} \times \frac{n_p}{N_t}}$$

- Kth: threshold on entropy
- Feature Selection by Entropy
  - retain all terms ki for which MI(ki,C) ≥ Kth
  - discard all others
  - recompute doc representations to consider only terms retained

3.5.5 Evaluation Metrics

Evaluation Metrics

- Evaluation
  - important for any text classification method
  - key step to validate a newly proposed classification method

Contingency Table

- Let
  - D: collection of documents
  - Dt: subset composed of training documents
  - Nt: number of documents in Dt
  - C = {c1, c2, . . . , cL}: set of all L classes
- Further let
  - T : Dt × C → [0, 1]: training set function
  - nt: number of docs from training set Dt assigned to class cp by the classifier T
  - F : D × C → [0, 1]: text classifiier function
  - nf: number of docs from training set Dt assigned to class cp by the classifier F

Contingency Table

- Apply classifier to all documents in training set
- Contingency table is given by

Case	$\mathcal{T}(d_j, c_p) = 1$	$\mathcal{T}(d_j, c_p) = 0$	*Total*
$\mathcal{F}(d_j, c_p) = 1$	$n_{f,t}$	$n_f - n_{f,t}$	n_f
$\mathcal{F}(d_j, c_p) = 0$	$n_t - n_{f,t}$	$N_t - n_f - n_t + n_{f,t}$	$N_t - n_f$
All docs	n_t	$N_t - n_t$	N_t

- $n_{f,t}$: number of docs that both the training and classifier functions assigned to class cp
- $n_t - n_{f,t}$: number of training docs in class cp that were miss-classified
- The remaining quantities are calculated analogously

Accuracy and Error

- Accuracy and error metrics, relative to a given class cp

$$Acc(c_p) = \frac{n_{f,t} + (N_t - n_f - n_t + n_{f,t})}{N_t}$$

$$Err(c_p) = \frac{(n_f - n_{f,t}) + (n_t - n_{f,t})}{N_t}$$

$$Acc(c_p) + Err(c_p) = 1$$

- These metrics are commonly used for evaluating classifiers

Accuracy and Error

- Accuracy and error have disadvantages
  - consider classification with only two categories cp and cr
  - assume that out of 1,000 docs, 20 are in class cp
  - a classifier that assumes all docs not in class cp
    - accuracy = 98%
    - error = 2%

    which erroneously suggests a very good classifier

Accuracy and Error

- Consider now a second classifier that correctly predicts 50% of the documents in cp

	$\mathcal{T}(d_j, c_p) = 1$	$\mathcal{T}(d_j, c_p) = 0$	
$\mathcal{F}(d_j, c_p) = 1$	10	0	10
$\mathcal{F}(d_j, c_p) = 0$	10	980	990
all docs	20	980	1,000

- In this case, accuracy and error are given by

$$Acc(c_p) = \frac{10 + 980}{1,000} = 99\%$$

$$Err(c_p) = \frac{10 + 0}{1,000} = 1\%$$

Accuracy and Error

- This classifier is much better than one that guesses that all documents are not in class cp
- However, its accuracy is just 1% better, it increased from 98% to 99%
- This suggests that the two classifiers are almost equivalent, which is not the case.

Precision and Recall

- Variants of precision and recall metrics in IR
- Precision P and recall R relative to a class cp

$$P(c_p) = \frac{n_{f,t}}{n_f} \qquad R(c_p) = \frac{n_{f,t}}{n_t}$$

- Precision is the fraction of all docs assigned to class cp by the classifier that really belong to class cp
- Recall is the fraction of all docs that belong to class cp that were correctly assigned to class cp

Precision and Recall

- Consider again the classifier illustrated below

	$\mathcal{T}(d_j, c_p) = 1$	$\mathcal{T}(d_j, c_p) = 0$	
$\mathcal{F}(d_j, c_p) = 1$	10	0	10
$\mathcal{F}(d_j, c_p) = 0$	10	980	990
all docs	20	980	1,000

- Precision and recall figures are given by

$$P(c_p) = \frac{10}{10} = 100\%$$

$$R(c_p) = \frac{10}{20} = 50\%$$