# Chapter 8

# Text Classification

# 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
- 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

- Text classification
  - a means to organize information
- 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
- Text classification
  - key technology in modern enterprises

# 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

# Machine Learning

- **Supervised learning**
  - training data provided as input
  - training data: classes for input documents
- **Unsupervised learning**
  - no training data is provided
  - Examples:
    - neural network models
    - independent component analysis
    - clustering
- **Semi-supervised learning**
  - small training data
  - combined with larger amount of unlabeled data

# The Text Classification Problem

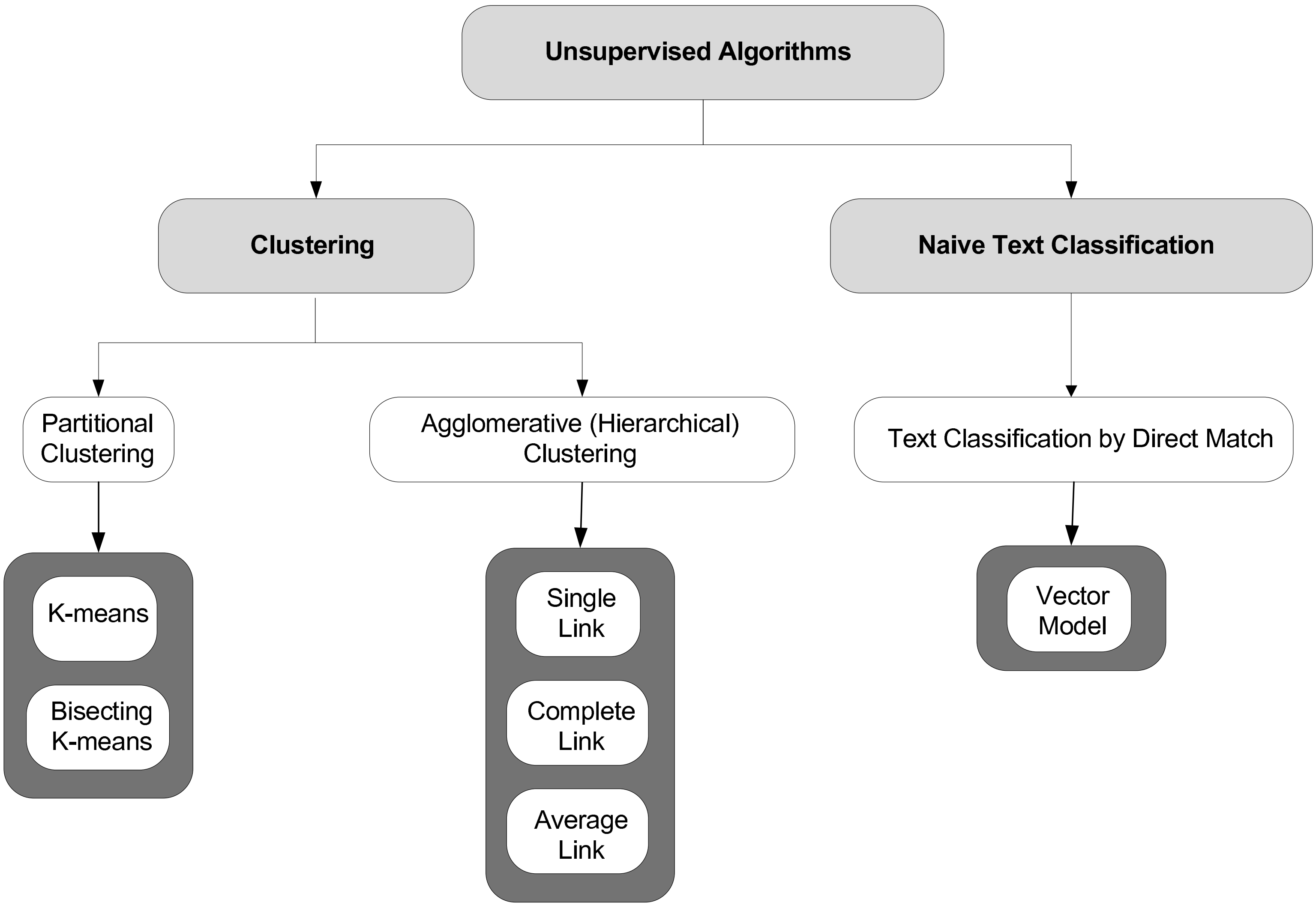
- A classifier can be formally defined
  - $\mathcal{D}$: a collection of documents
  - $\mathcal{C} = \{c_1, c_2, \ldots, c_L\}$: a set of $L$ classes with their respective labels
  - a text classifier is a binary function $\mathcal{F} : \mathcal{D} \times \mathcal{C} \rightarrow \{0, 1\}$, which assigns to each pair $[d_j, c_p]$, $d_j \in \mathcal{D}$ and $c_p \in \mathcal{C}$, a value of
    - 1, if $d_j$ is a member of class $c_p$
    - 0, if $d_j$ is *not a member of class* $c_p$
- 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 $\mathcal{F}$
  - defined as binary function of document-class pair $[d_j, c_p]$
  - can be modified to compute degree of membership of $d_j$ in $c_p$
    - documents as *candidates* for membership in class $c_p$
    - candidates sorted by decreasing values of $\mathcal{F}(d_j, c_p)$

# Text Classification Algorithms

- Unsupervised algorithms we discuss



# Text Classification Algorithms

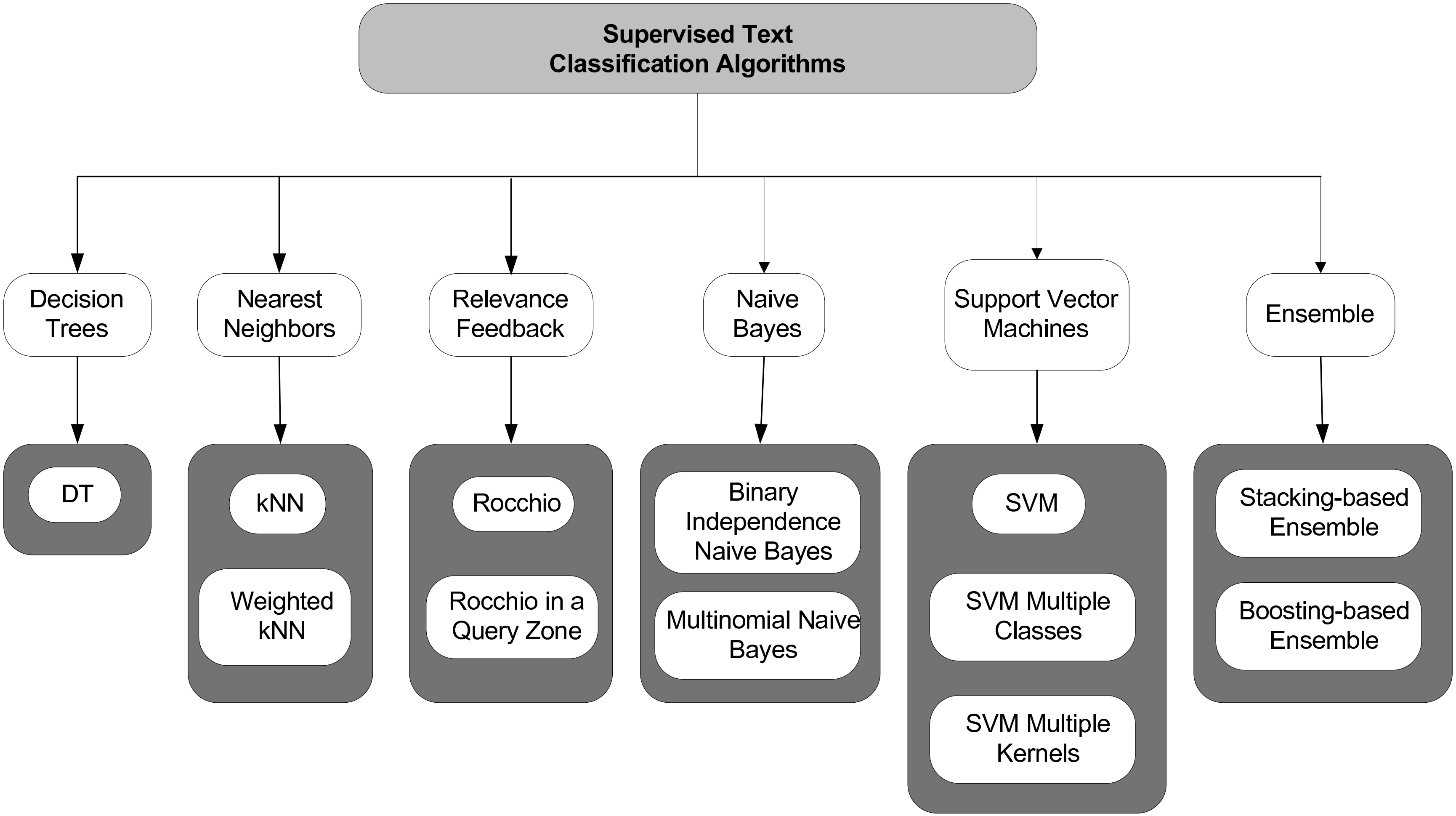
- **Supervised** algorithms depend on a **training set**
  - set of classes with examples of documents for each class
  - examples determined by human specialists
  - training set used to **learn** a classification function

# Text Classification Algorithms

- The larger the number of training examples, the better is the fine tuning of the classifier
  - **Overfitting:** classifier becomes specific to the training examples
- To evaluate the classifier
  - use a set of <u>unseen</u> objects
  - commonly referred to as **test set**

# Text Classification Algorithms

- Supervised classification algorithms we discuss



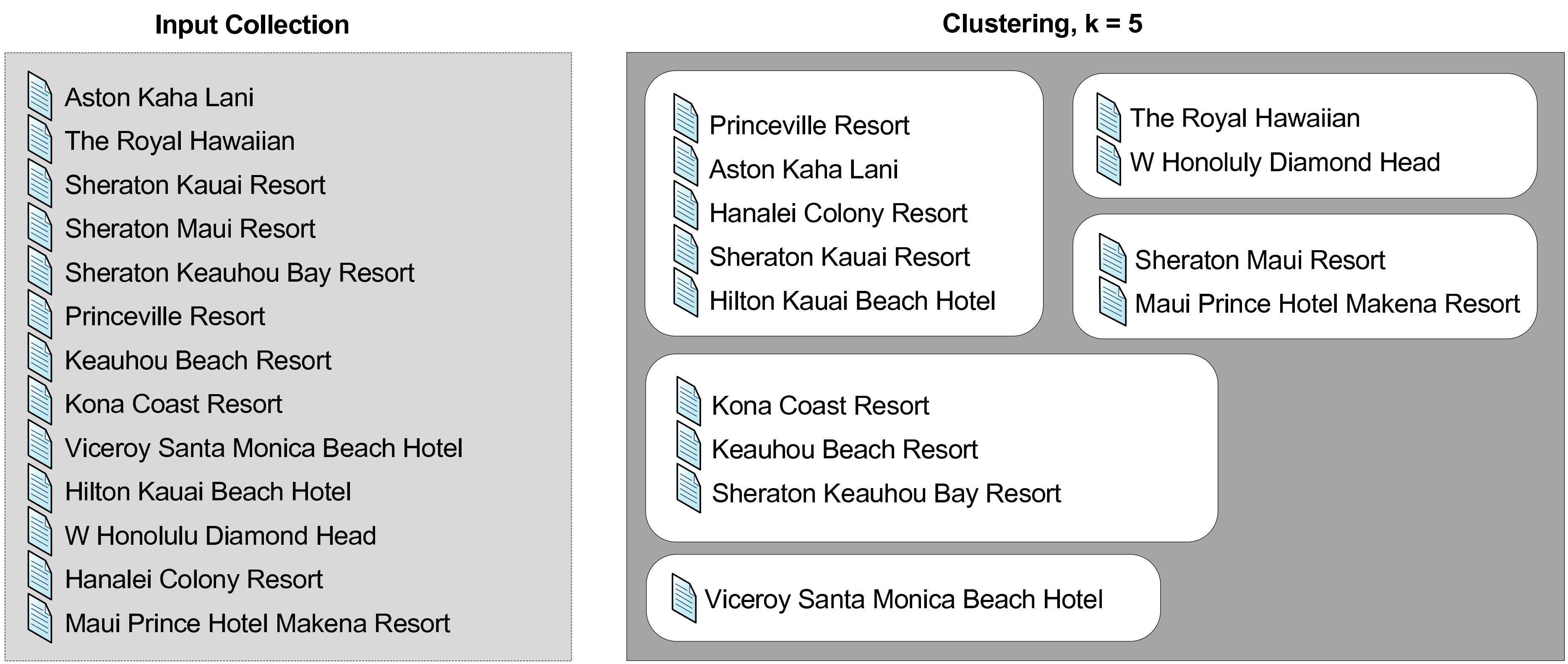
# Unsupervised Algorithms

# 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

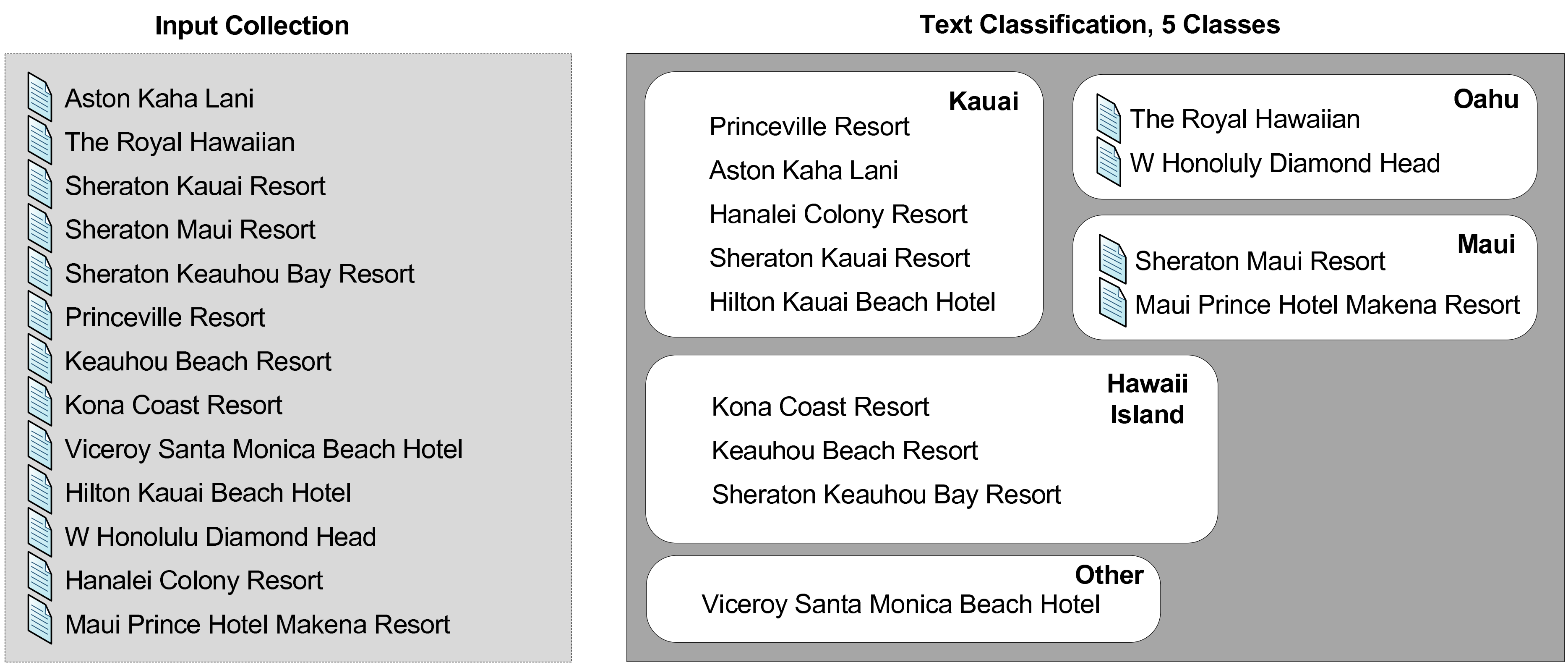
- Clustering of hotel Web pages in Hawaii




(a)

# Clustering

- To obtain classes, assign labels to clusters




(b)

# 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 $d_j$ be represented as vector $\vec{d_j}$

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

  where

  - $w_{i,j}$: weight of term $k_i$ in document $d_j$
  - $t$: size of the vocabulary

# K-means in Batch Mode

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

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

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

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

# K-means in Batch Mode

3. **Update Step.**
   - recompute centroids of each cluster $c_p$

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

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

# K-means Online

- Recompute centroids after classification of each individual doc

1. **Initial Step.**
   - select K documents randomly
   - use them as initial centroids
2. **Assignment Step.**
   For each document $d_j$ repeat
   - assign document $d_j$ to the cluster with closest centroid
   - recompute the centroid of that cluster to include $d_j$
3. **Final Step.** Repeat assignment step until no centroid changes.

- It is argued that online K-means works better than batch K-means

# Bisecting K-means

- Algorithm
  - build a hierarchy of clusters
  - at each step, branch into two clusters
- Apply K-means repeatedly, with K=2

1. **Initial Step.** assign all documents to a single cluster
2. **Split Step.**
   - select largest cluster
   - apply K-means to it, with $K = 2$
3. **Selection Step.**
   - if stop criteria satisfied (e.g., no cluster larger than pre-defined size), stop execution
   - go back to Split Step

# 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 $N \times N$ 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

- $dist(c_p, c_r)$: distance between two clusters $c_p$ and $c_r$
- $dist(d_j, d_l)$: distance between docs $d_j$ and $d_l$
- **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 $\mathcal{D}$ of documents
    - set $\mathcal{C} = \{c_1, c_2, \ldots, c_L\}$ of $L$ classes and their labels
  - Algorithm: associate one or more classes of $\mathcal{C}$ with each doc in $\mathcal{D}$
    - 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:
   - $\mathcal{D}$: collection of documents to classify
   - $\mathcal{C} = \{c_1, c_2, \ldots, c_L\}$: set of $L$ classes with their labels
2. Represent
   - each document $d_j$ by a weighted vector $\vec{d_j}$
   - each class $c_p$ by a weighted vector $\vec{c_p}$ (use the labels)
3. For each document $d_j \in \mathcal{D}$ do
   - retrieve classes $c_p \in \mathcal{C}$ whose labels contain terms of $d_j$
   - for each pair $[d_j, c_p]$ 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 $d_j$ classes $c_p$ with highest values of $sim(d_j, c_p)$

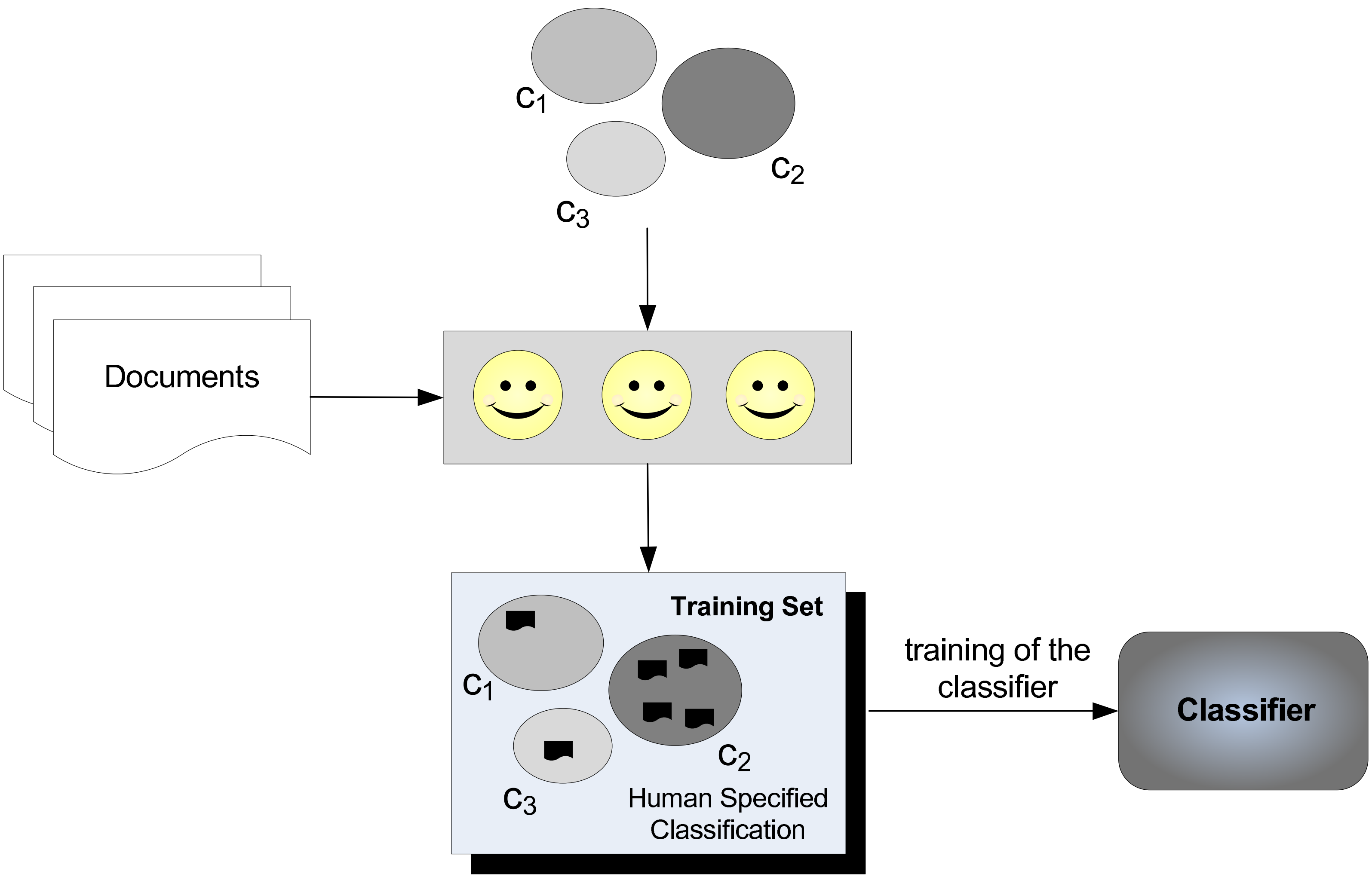
# Supervised Algorithms

# Supervised Algorithms

- Depend on a **training set**
  - $\mathcal{D}_t \subset \mathcal{D}$: subset of training documents
  - $\mathcal{T} : \mathcal{D}_t \times \mathcal{C} \rightarrow \{0, 1\}$: training set function
    Assigns to each pair $[d_j, c_p]$, $d_j \in \mathcal{D}_t$ and $c_p \in \mathcal{C}$ a value of
    - 1, if $d_j \in c_p$, according to judgement of human specialists
    - 0, if $d_j \notin c_p$, according to judgement of human specialists
  - Training set function $\mathcal{T}$ is used to fine tune the classifier

# Supervised Algorithms

- The training phase of a classifier

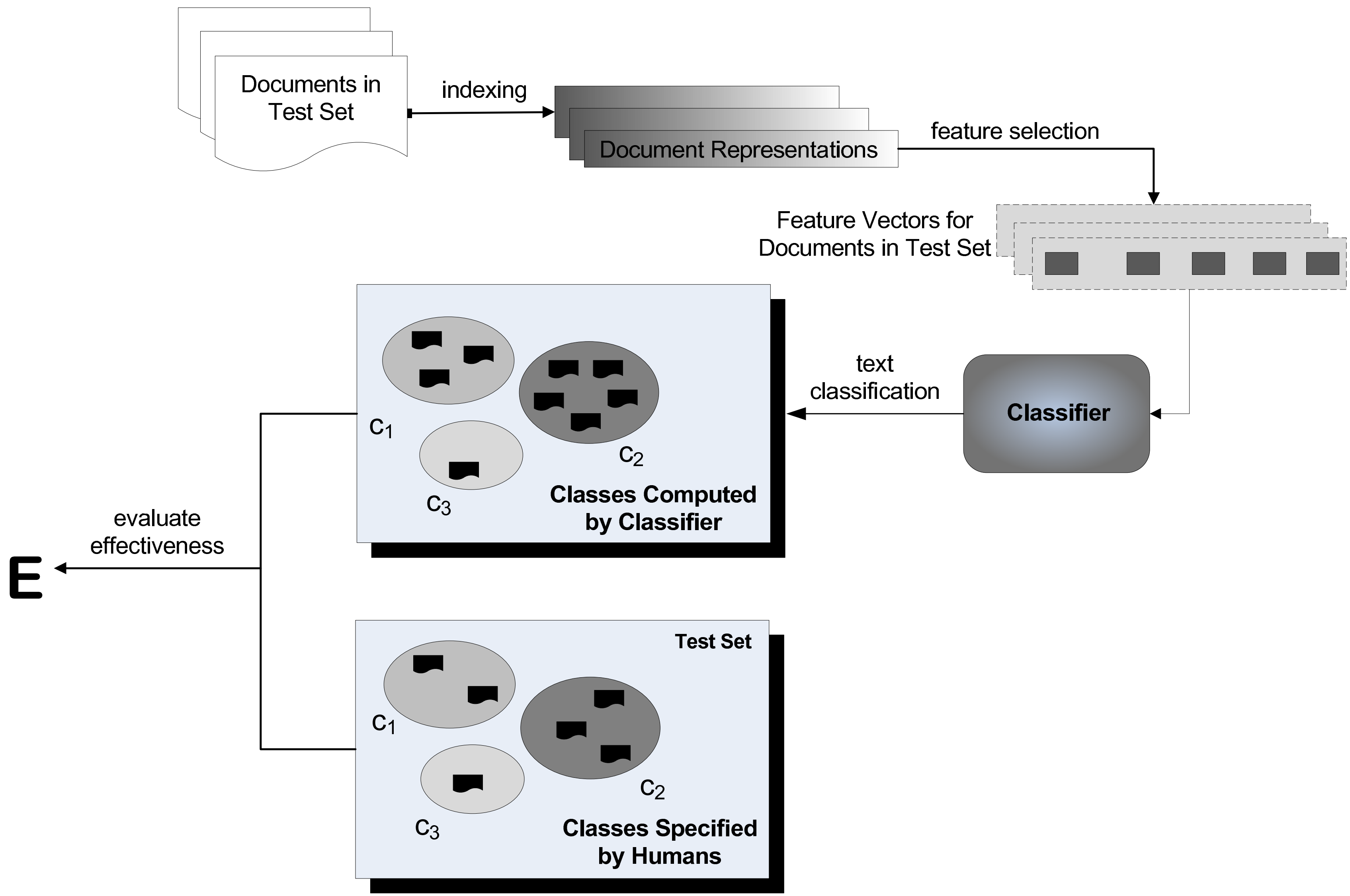


# Supervised Algorithms

- 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

# Supervised Algorithms

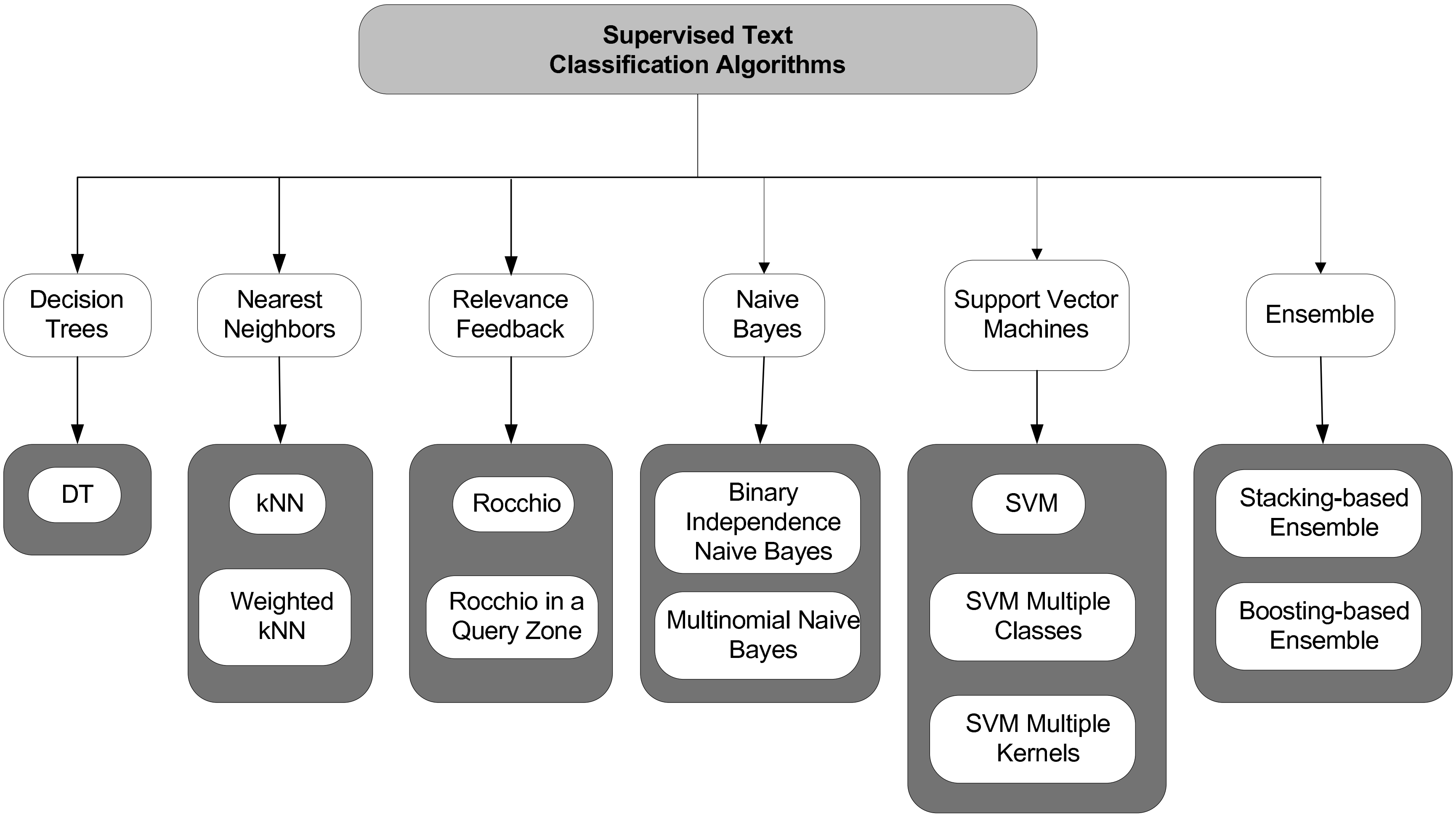
- Classification and evaluation processes



# Supervised Algorithms

- 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

# Supervised Algorithms



# 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

# Basic Technique

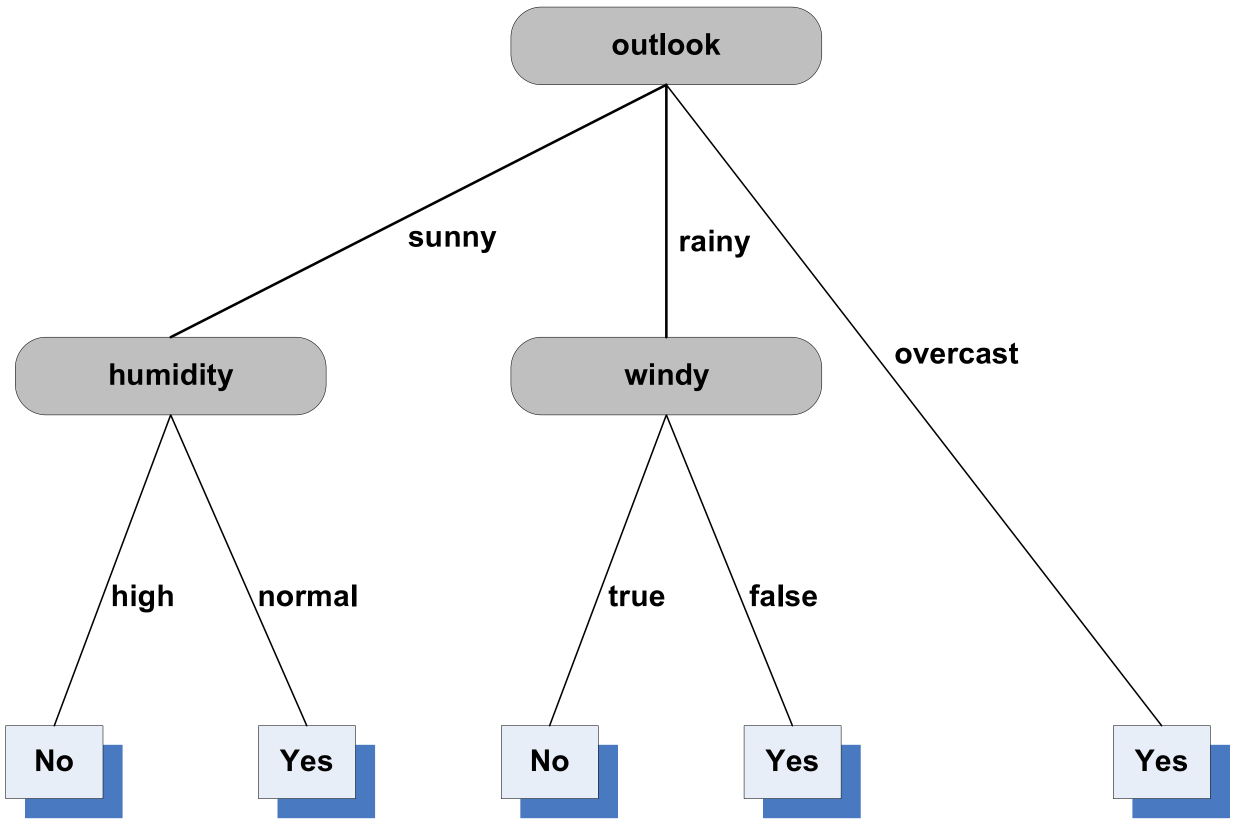
- Consider the small relational database below

| | 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

# Basic Technique

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



# Basic Technique

- Internal nodes $\rightarrow$ attribute names
- Edges $\rightarrow$ attribute values
- Traversal of DT $\rightarrow$ value for attribute “Play”.
- $(Outlook = sunny) \wedge (Humidity = high) \rightarrow (Play = no)$

| | Id | Play | Outlook | Temperature | Humidity | Windy |
|---|---|---|---|---|---|---|
| Test Instance | 11 | ? | sunny | cool | high | false |

# Basic Technique

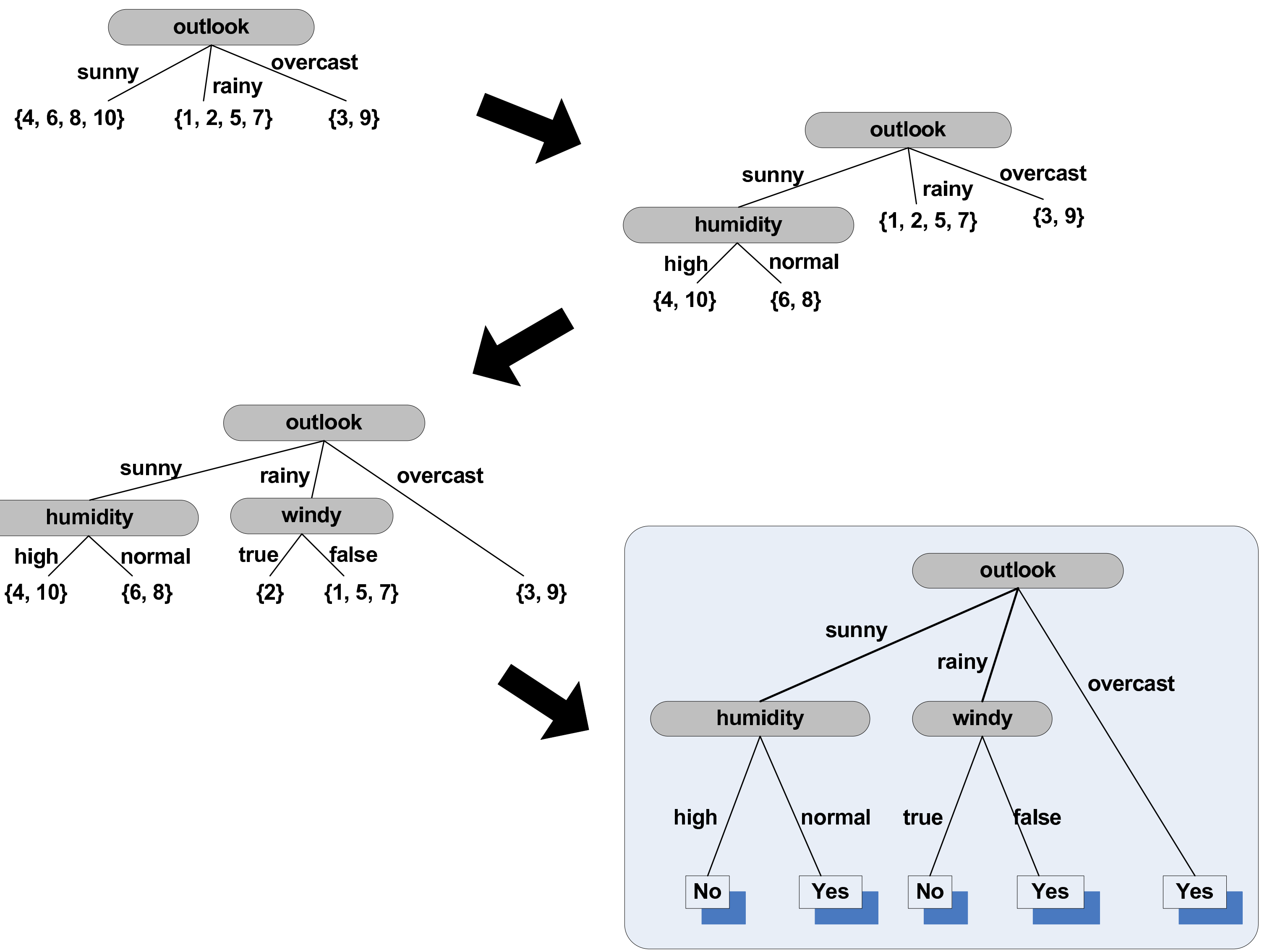
- Predictions based on seen instances
- New instance that violates seen patterns will lead to erroneous prediction
- Example database works as training set for building the decision tree

# The Splitting Process

- DT for a database can be built using recursive splitting strategy
- Goal: build DT for attribute Play
  - select one of the attributes, other than Play, as root
  - use attribute values to split tuples into subsets
  - for each subset of tuples, select a second splitting attribute
  - repeat

# The Splitting Process

- Step by step splitting process



# The Splitting Process

- Strongly affected by order of split attributes
  - depending on order, tree might become unbalanced
- Balanced or near-balanced trees are more efficient for predicting attribute values
- Rule of thumb: select attributes that reduce average path length

# 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

# 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 $edge(v_i, v_j) \in E$
    - $v_i$ is the father node
    - $v_j$ 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

# Classification of Documents

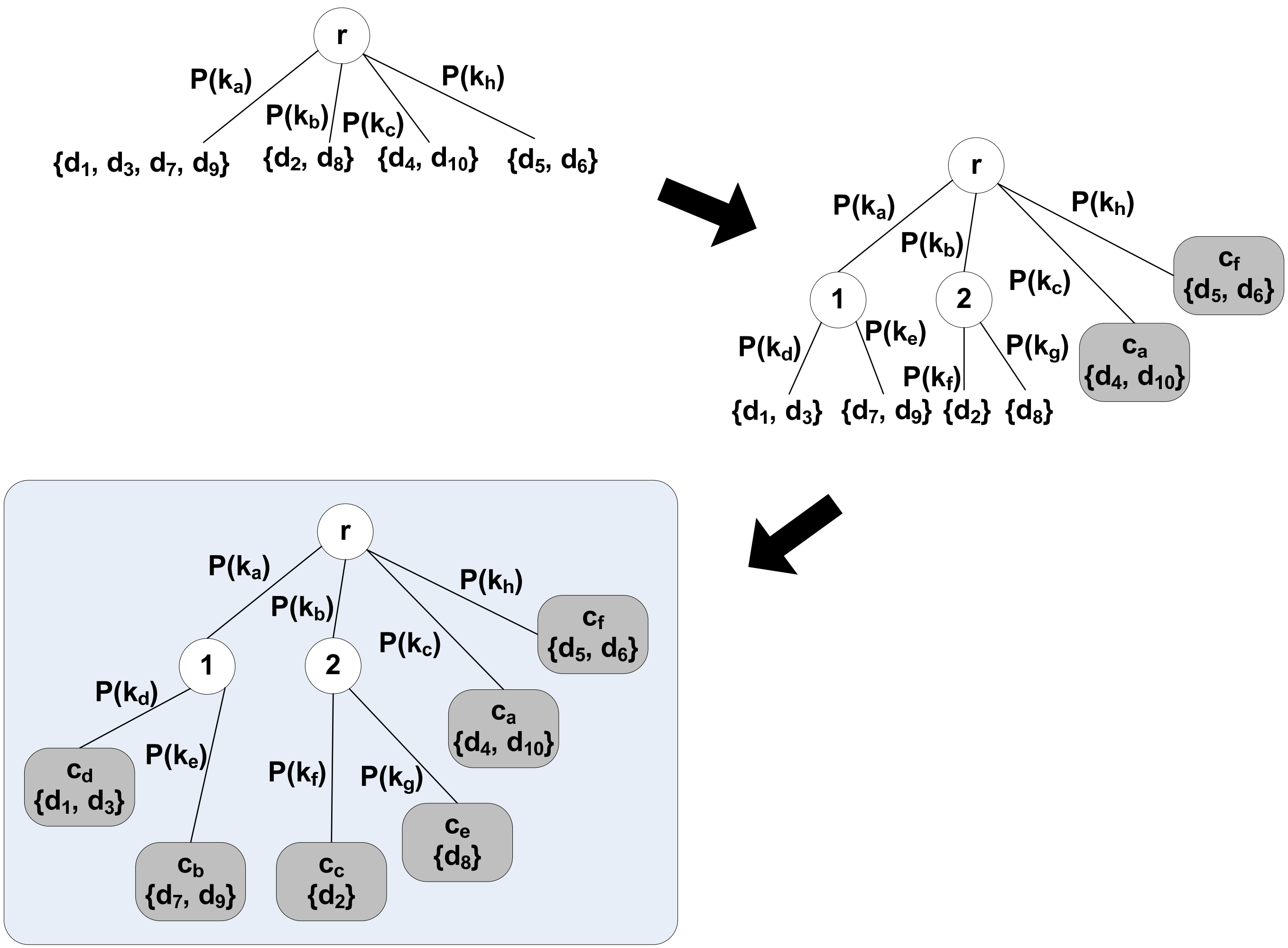
- Define
  - $K = \{k_1, k_2, \ldots, k_t\}$: set of index terms of a doc collection
  - $C$: set of all classes
  - $P$: set of logical predicates on the index terms
- $DT = (V, E; r; l_I, l_L, l_E)$: a six-tuple where
  - $(V; E; r)$: a tree whose root is $r$
  - $l_I : I \to K$: a function that associates with each internal node of the tree one or more index terms
  - $l_L : \overline{I} \to C$: a function that associates with each non-internal (leaf) node a class $c_p \in C$
  - $l_E : E \to P$: a function that associates with each edge of the tree a logical predicate from $P$

# Classification of Documents

- Decision tree model for class $c_p$ 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

# Classification of Documents

- Terms $k_a$, $k_b$, $k_c$, and $k_h$ have been selected for first split



# 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

# Classification of Documents

- 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

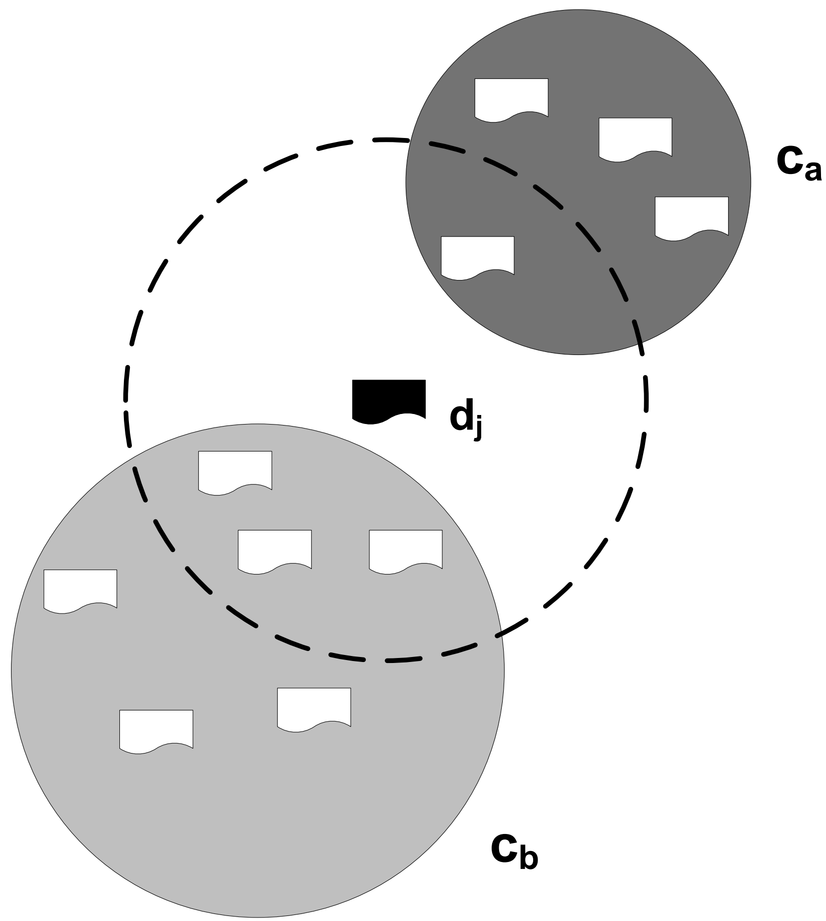
# The $k$NN Classifier

# The $k$NN Classifier

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

# The $k$NN Classifier

- An example of a 4-NN classification process



# Classification of Documents

- $k$NN: to each document-class pair $[d_j, c_p]$ 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

  - $N_k(d_j)$: set of the $k$ nearest neighbors of $d_j$ in training set
  - $similarity(d_j, d_t)$: cosine formula of Vector model (for instance)
  - $\mathcal{T}(d_t, c_p)$: training set function returns
    - $1$, if $d_t$ belongs to class $c_p$
    - $0$, otherwise

- Classifier assigns to $d_j$ class(es) $c_p$ with highest score(s)

# Classification of Documents

- Problem with $k$NN: 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 Rocchio Classifier

# The Rocchio Classifier

- Rocchio relevance feedback
  - modifies user query based on user feedback
  - produces new query that better approximates the interest of the user
  - can be adapted to text classification
- Interpret training set as feedback information
  - terms that belong to training docs of a given class $c_p$ are said to provide positive feedback
  - terms that belong to training docs outside class $c_p$ are said to provide negative feedback
- Feedback information summarized by a centroid vector
- New document classified by distance to centroid

# Basic Technique

- Each document $d_j$ represented as a weighted term vector $\vec{d_j}$

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

  - $w_{i,j}$: weight of term $k_i$ in document $d_j$
  - $t$: size of the vocabulary

# Classification of Documents

- Rochio classifier for a class $c_p$ is computed as a centroid given by

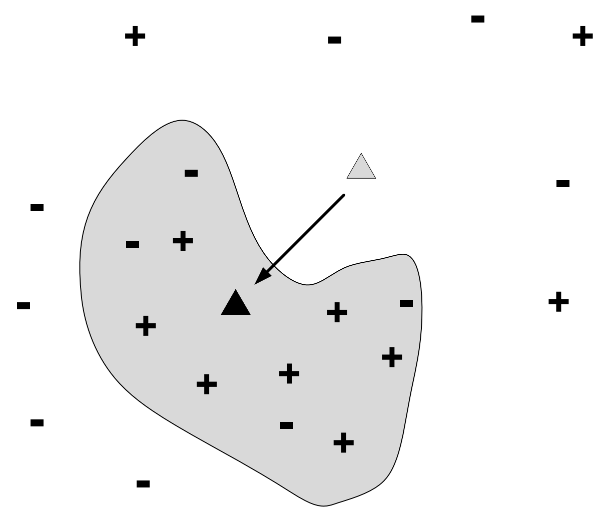
$$\vec{c}_p = \frac{\beta}{n_p} \sum_{d_j \in c_p} \vec{d}_j - \frac{\gamma}{N_t - n_p} \sum_{d_l \notin c_p} \vec{d}_l$$

  where

  - $n_p$: number of documents in class $c_p$
  - $N_t$: total number of documents in the training set
  - terms of training docs in class $c_p$: positive weights
  - terms of docs outside class $c_p$: negative weights

# Classification of Documents

- plus signs: terms of training docs in class $c_p$
- minus signs: terms of training docs outside class $c_p$



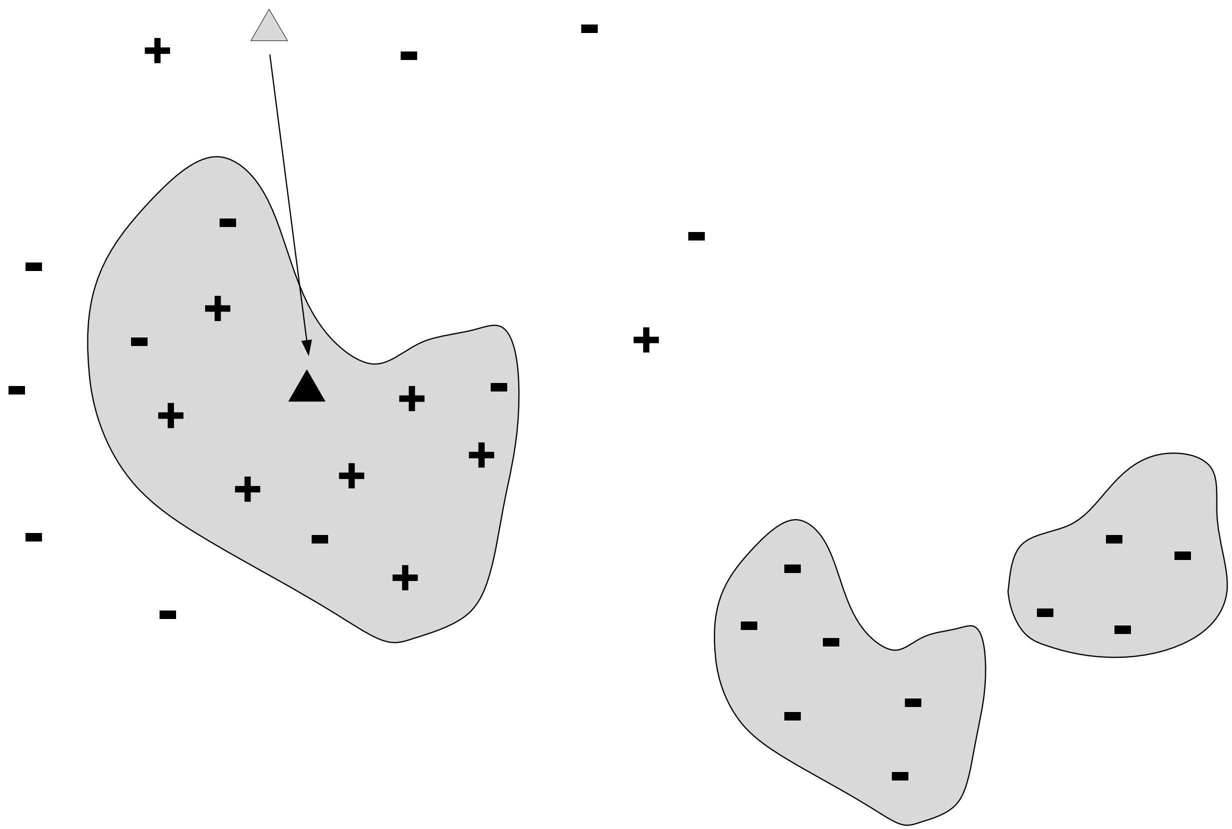
- Classifier assigns to each document-class $[d_j, c_p]$ a score

$$S(d_j, c_p) = |\vec{c}_p - \vec{d}_j|$$

- Classes with highest scores are assigned to $d_j$

# Rocchio in a Query Zone

- For specific domains, negative feedback might move the centroid away from the topic of interest



Distant negative feedback documents might shift the centroid away from the positive terms.

# Rocchio in a Query Zone

- To reduce this effect, decrease number of negative feedback docs
  - use only **most positive** docs among all docs that provide negative feedback
  - these are usually referred to as **near-positive documents**
- Near-positive documents are selected as follows
  - $\vec{c}_{p+}$ : centroid of the training documents that belong to class $c_p$
  - training docs outside $c_p$: measure their distances to $\vec{c}_{p+}$
  - smaller distances to centroid: near-positive documents

# The Probabilistic Naive Bayes Classifier

# Naive Bayes

- Probabilistic classifiers
  - assign to each document-class pair $[d_j, c_p]$ 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

# Naive Bayes Classifier

- For efficiency, simplify computation of $P(\vec{d}_j|c_p)$
  - most common simplification: independence of index terms
  - classifiers are called **Naive Bayes classifiers**
- Many variants of Naive Bayes classifiers
  - best known is based on the classic probabilistic model
  - doc $d_j$ represented by vector of binary weights

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

$$w_{i,j} = \begin{cases} 1 & \text{if term } k_i \text{ occurs in document } d_j \\ 0 & \text{otherwise} \end{cases}$$

# Naive Bayes Classifier

- To each pair $[d_j, c_p]$, the classifier assigns a score

$$S(d_j, c_p) = \frac{P(c_p|\vec{d_j})}{P(\overline{c}_p|\vec{d_j})}$$

  - $P(c_p|\vec{d_j})$: probability that document $d_j$ belongs to class $c_p$
  - $P(\overline{c}_p|\vec{d_j})$: probability that document $d_j$ does not belong to $c_p$
  - $P(c_p|\vec{d_j}) + P(\overline{c}_p|\vec{d_j}) = 1$

# Naive Bayes Classifier

- Applying Bayes, we obtain

$$S(d_j, c_p) \sim \frac{P(\vec{d_j}|c_p)}{P(\vec{d_j}|\overline{c}_p)}$$

- Independence assumption

$$\begin{aligned} P(\vec{d_j}|c_p) &= \prod_{k_i \in \vec{d_j}} P(k_i|c_p) \times \prod_{k_i \notin \vec{d_j}} P(\overline{k}_i|c_p) \\ P(\vec{d_j}|\overline{c}_p) &= \prod_{k_i \in \vec{d_j}} P(k_i|\overline{c}_p) \times \prod_{k_i \notin \vec{d_j}} P(\overline{k}_i|\overline{c}_p) \end{aligned}$$

# Naive Bayes Classifier

- Equation for the score $S(d_j, c_p)$

$$
\begin{aligned}
S(d_j, c_p) &\sim \sum_{k_i} w_{i,j} \left( \log \frac{p_{iP}}{1 - p_{iP}} + \log \frac{1 - q_{iP}}{q_{iP}} \right) \\
p_{iP} &= P(k_i | c_p) \\
q_{iP} &= P(k_i | \overline{c}_p)
\end{aligned}
$$

  - $p_{iP}$: probability that $k_i$ belongs to doc randomly selected from $c_p$
  - $q_{iP}$: probability that $k_i$ belongs to doc randomly selected from outside $c_p$

# Naive Bayes Classifier

- Estimate $p_{iP}$ and $q_{iP}$ from set $\mathcal{D}_t$ of training docs

$$p_{iP} = \frac{1 + \sum_{d_j | d_j \in \mathcal{D}_t \wedge k_i \in d_j} P(c_p|d_j)}{2 + \sum_{d_j \in \mathcal{D}_t} P(c_p|d_j)} = \frac{1 + n_{i,p}}{2 + n_p}$$

$$q_{iP} = \frac{1 + \sum_{d_j | d_j \in \mathcal{D}_t \wedge k_i \in d_j} P(\overline{c}_p|d_j)}{2 + \sum_{d_j \in \mathcal{D}_t} P(\overline{c}_p|d_j)} = \frac{1 + (n_i - n_{i,p})}{2 + (N_t - n_p)}$$

  - $n_{i,p}$, $n_i$, $n_p$, $N_t$: see probabilistic model
  - $P(c_p|d_j) \in \{0, 1\}$ and $P(\overline{c}_p|d_j) \in \{0, 1\}$: given by training set

- **Binary Independence Naive Bayes** classifier
  - assigns to each doc $d_j$ classes with higher $S(d_j, c_p)$ scores

# Multinomial Naive Bayes Classifier

- Naive Bayes classifier: term weights are binary
- Variant: consider term frequency inside docs
- To classify doc $d_j$ in class $c_p$

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

  - $P(\vec{d}_j)$: prior document probability
  - $P(c_p)$: prior class probability

$$P(c_p) = \frac{\sum_{d_j \in \mathcal{D}_t} P(c_p|d_j)}{N_t} = \frac{n_p}{N_t}$$

  - $P(c_p|d_j) \in \{0, 1\}$: given by training set of size $N_t$

# Multinomial Naive Bayes Classifier

- Prior document probability given by

$$P(\vec{d}_j) = \sum_{p=1}^{L} P_{prior}(\vec{d}_j|c_p) \times P(c_p)$$

where

$$P_{prior}(\vec{d}_j|c_p) = \prod_{k_i \in \vec{d}_j} P(k_i|c_p) \times \prod_{k_i \notin \vec{d}_j} [1 - P(k_i|c_p)]$$

$$P(k_i|c_p) = \frac{1 + \sum_{d_j|d_j \in \mathcal{D}_t \wedge k_i \in d_j} P(c_p|d_j)}{2 + \sum_{d_j \in \mathcal{D}_t} P(c_p|d_j)} = \frac{1 + n_{i,p}}{2 + n_p}$$

# Multinomial Naive Bayes Classifier

- These equations do not consider term frequencies
- To include term frequencies, modify $P(\vec{d_j}|c_p)$
  - consider that terms of doc $d_j \in c_p$ are drawn from known distribution
  - each single term draw
    - Bernoulli trial with probability of success given by $P(k_i|c_p)$
  - each term $k_i$ is drawn as many times as its doc frequency $f_{i,j}$

# Multinomial Naive Bayes Classifier

- Multinomial probabilistic term distribution

$$P(\vec{d}_j|c_p) = F_j! \times \prod_{k_i \in d_j} \frac{[P(k_i|c_p)]^{f_{i,j}}}{f_{i,j}!}$$

$$F_j = \sum_{k_i \in d_j} f_{i,j}$$

  - $F_j$: a measure of document length

- Term probabilities estimated from training set $\mathcal{D}_t$

$$P(k_i|c_p) = \frac{\sum_{d_j \in \mathcal{D}_t} f_{i,j} P(c_p|d_j)}{\sum_{\forall k_i} \sum_{d_j \in \mathcal{D}_t} f_{i,j} P(c_p|d_j)}$$

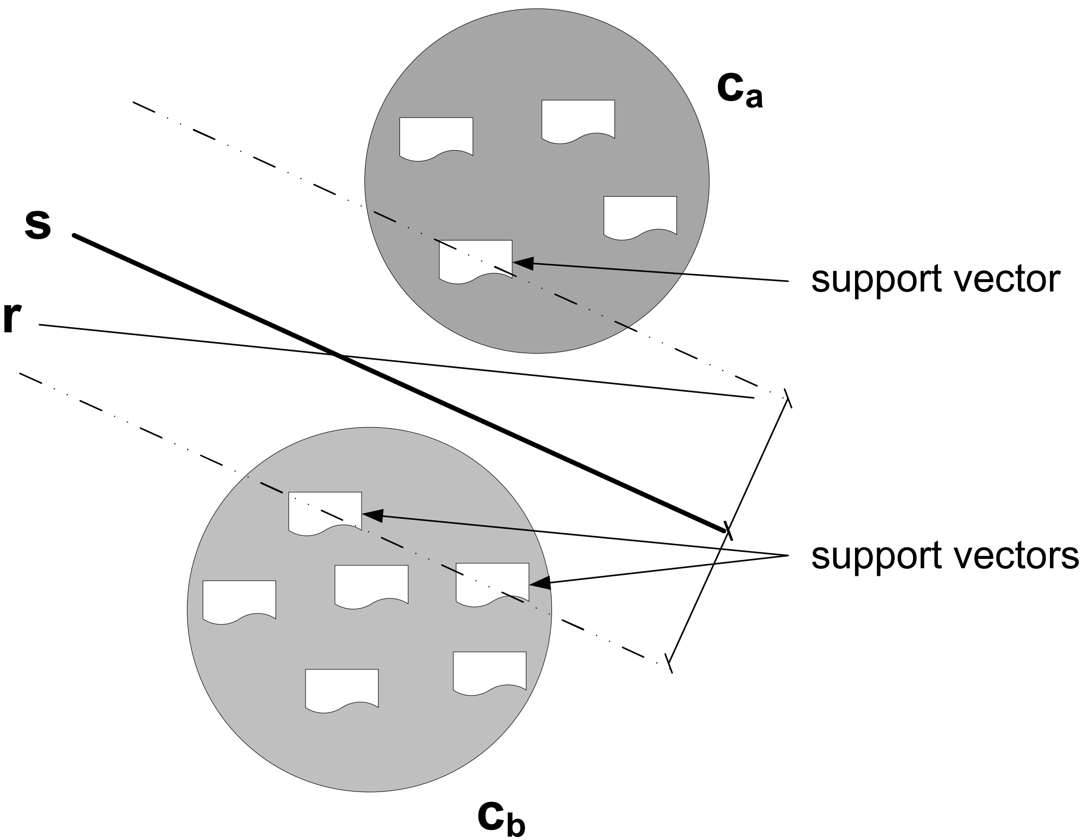
# The SVM Classifier

# SVM Basic Technique – Intuition

- **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

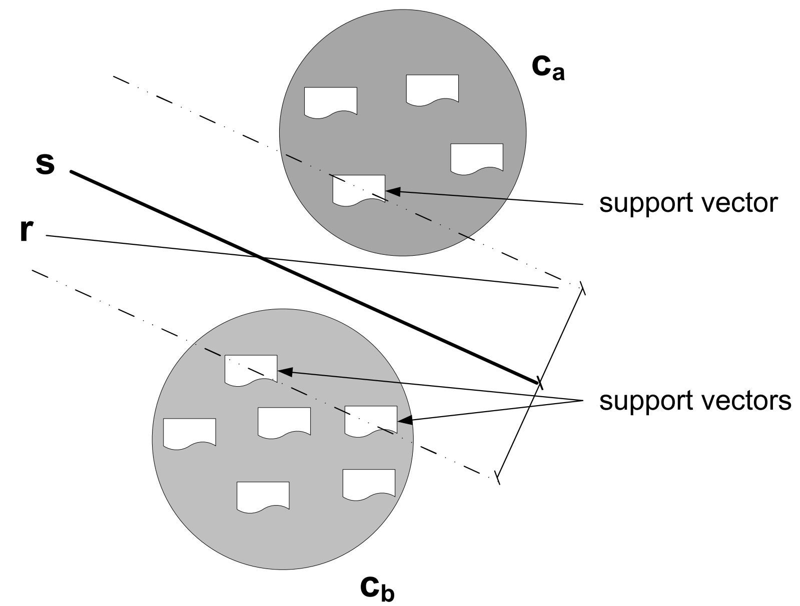
# SVM Basic Technique – Intuition

- Simple 2D example: training documents linearly separable



# SVM Basic Technique – Intuition

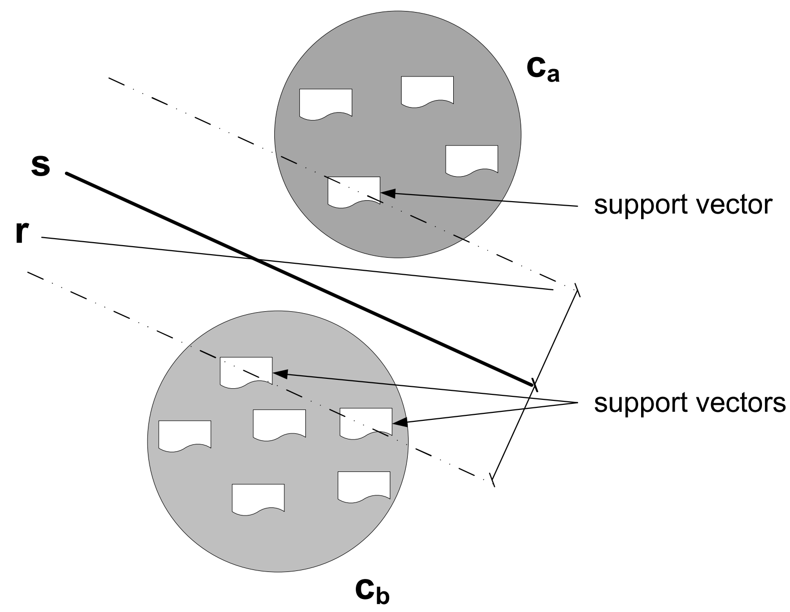
- **Line $s$—The Decision Hyperplane**
  - maximizes distances to closest docs of each class
  - it is the best separating hyperplane
- **Delimiting Hyperplanes**
  - parallel dashed lines that delimit region where to look for a solution



# SVM Basic Technique – Intuition

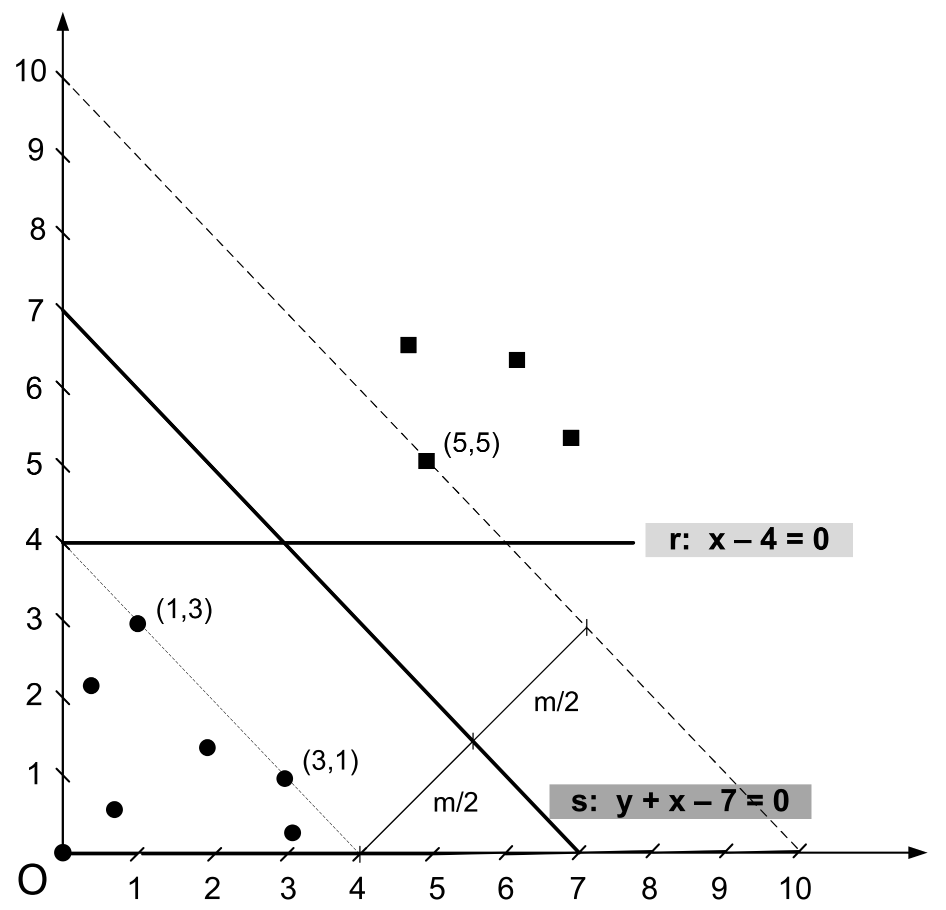
- Lines that cross the delimiting hyperplanes
  - candidates to be selected as the decision hyperplane
  - lines that are parallel to delimiting hyperplanes: best candidates

- **Support vectors:** documents that belong to, and define, the delimiting hyperplanes



# SVM Basic Technique – Intuition

- Our example in a 2-dimensional system of coordinates

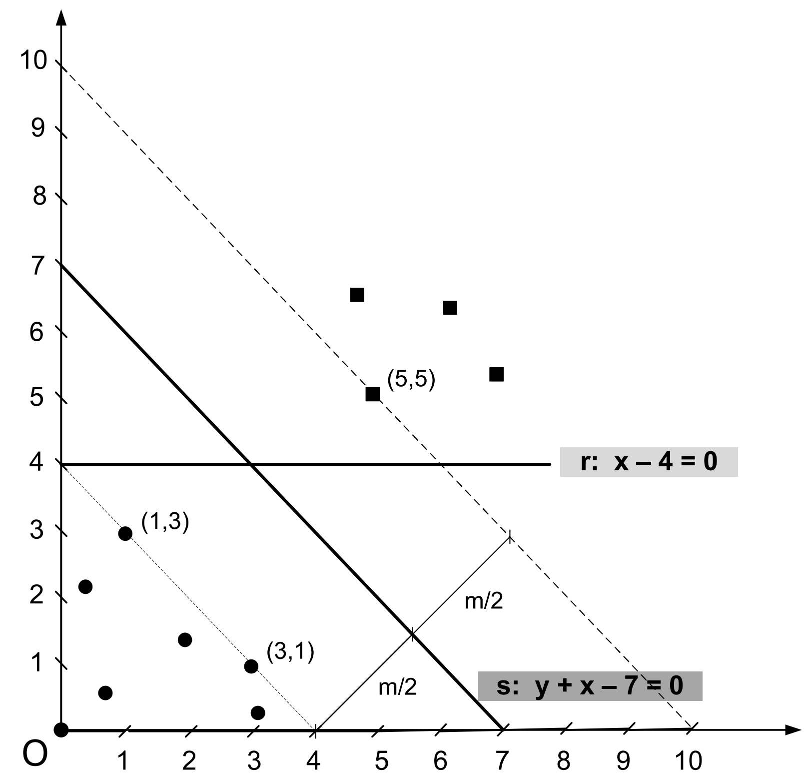


# SVM Basic Technique – Intuition

- Let,
  - $\mathcal{H}_w$: a hyperplane that separates docs in classes $c_a$ and $c_b$
  - $m_a$: distance of $\mathcal{H}_w$ to the closest document in class $c_a$
  - $m_b$: distance of $\mathcal{H}_w$ to the closest document in class $c_b$
  - $m_a + m_b$: **margin** $m$ of the SVM
- The **decision hyperplane** maximizes the margin $m$

# SVM Basic Technique – Intuition

- 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



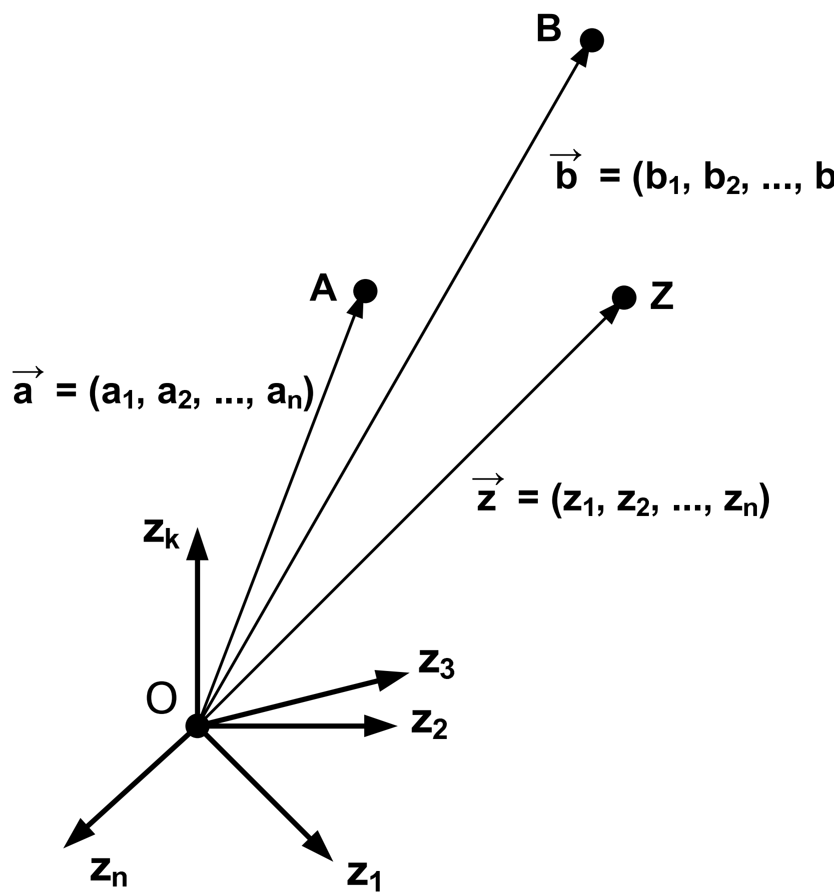
# Lines and Hyperplanes in the $\mathcal{R}^n$

- Let $\mathcal{R}^n$ refer to an $n$-dimensional space with origin in $\mathcal{O}$
- generic point $Z$ is represented as

$$\vec{z} = (z_1, z_2, \ldots, z_n)$$

$z_i$, $1 \leq i \leq n$, are real variables

- Similar notation to refer to specific fixed points such as A, B, H, P, and Q



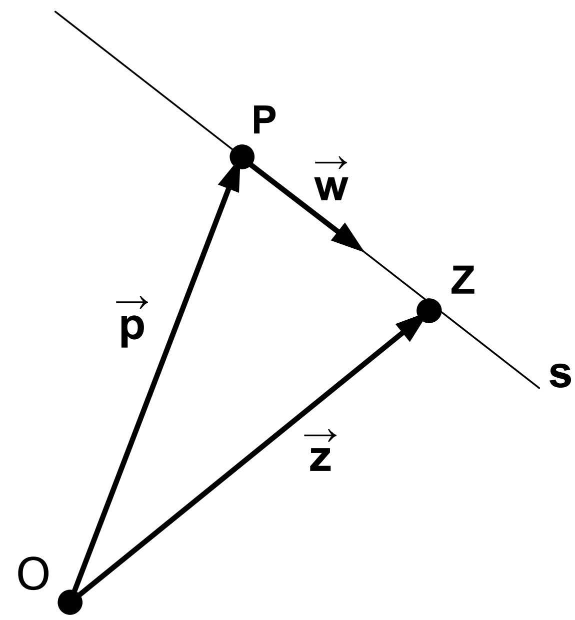
# Lines and Hyperplanes in the $\mathcal{R}^n$

- Line $s$ in the direction of a vector $\vec{w}$ that contains a given point $P$

- Parametric equation for this line

$$s : \vec{z} = t\vec{w} + \vec{p}$$

where $-\infty < t < +\infty$



# Lines and Hyperplanes in the $\mathcal{R}^n$

- Hyperplane $\mathcal{H}_w$ that contains a point $H$ and is perpendicular to a given vector $\vec{w}$

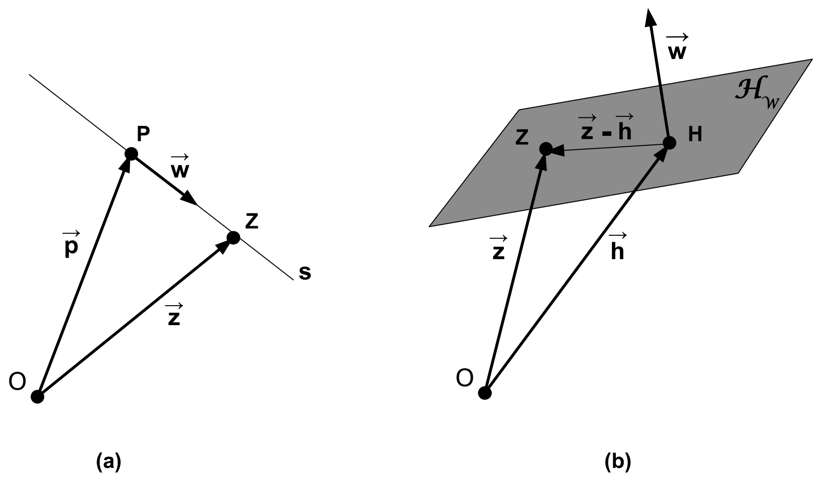
- Its normal equation is

$$\mathcal{H}_w : (\vec{z} - \vec{h})\vec{w} = 0$$

- Can be rewritten as

$$\mathcal{H}_w : \vec{z}\vec{w} + k = 0$$

where $\vec{w}$ and $k = -\vec{h}\vec{w}$ need to be determined

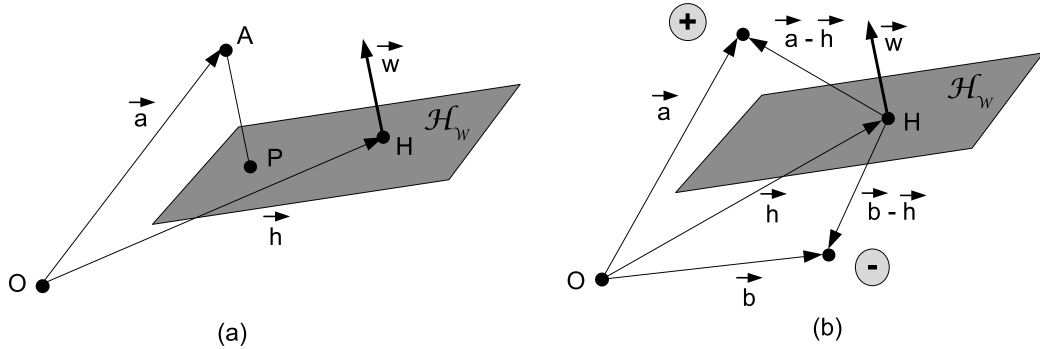


# Lines and Hyperplanes in the $\mathcal{R}^n$

- $P$: projection of point $A$ on hyperplane $\mathcal{H}_w$
- $\overline{AP}$: distance of point $A$ to hyperplane $\mathcal{H}_w$
- Parametric equation of line determined by $A$ and $P$

$$line(\overline{AP}) : \vec{z} = t\vec{w} + \vec{a}$$

where $-\infty < t < +\infty$



# Lines and Hyperplanes in the $\mathcal{R}^n$

- For point $P$ specifically

$$\vec{p} = t_p \vec{w} + \vec{a}$$

where $t_p$ is value of $t$ for point $P$

- Since $P \in \mathcal{H}_w$

$$(t_p \vec{w} + \vec{a})\vec{w} + k = 0$$

- Solving for $t_p$,

$$t_p = -\frac{\vec{a}\vec{w} + k}{|\vec{w}|^2}$$

where $|\vec{w}|$ is the vector norm

# Lines and Hyperplanes in the $\mathcal{R}^n$

- Substitute $t_p$ into Equation of point $P$

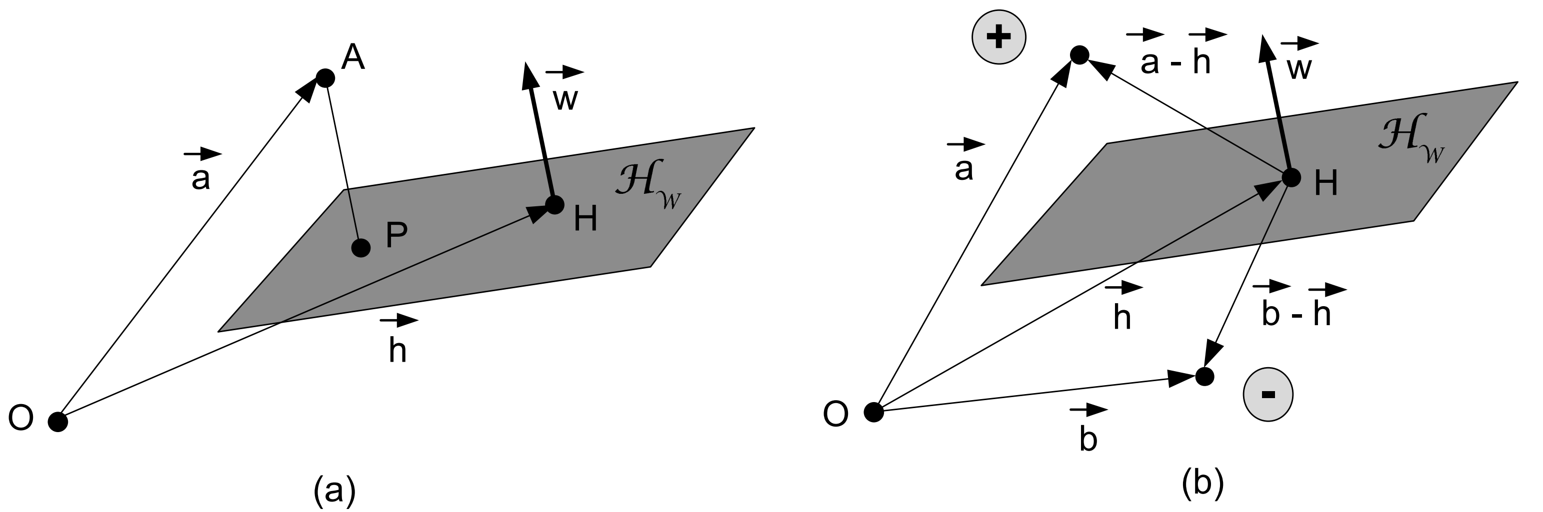
$$\vec{a} - \vec{p} = \frac{\vec{a}\vec{w} + k}{|\vec{w}|} \times \frac{\vec{w}}{|\vec{w}|}$$

- Since $\vec{w}/|\vec{w}|$ is a unit vector

$$\overline{AP} = |\vec{a} - \vec{p}| = \frac{\vec{a}\vec{w} + k}{|\vec{w}|}$$

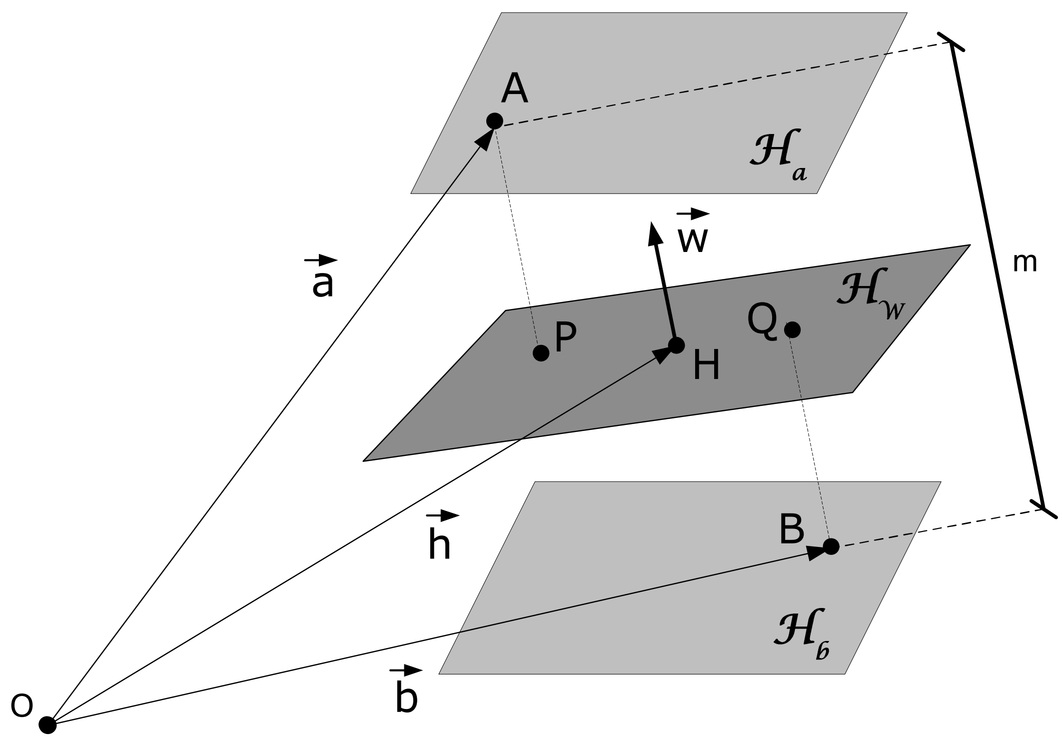
# Lines and Hyperplanes in the $\mathcal{R}^n$

- How signs vary with regard to a hyperplane $\mathcal{H}_w$
  - region above $\mathcal{H}_w$: points $\vec{z}$ that make $\vec{z}\vec{w} + k$ positive
  - region below $\mathcal{H}_w$: points $\vec{z}$ that make $\vec{z}\vec{w} + k$ negative



# SVM Technique – Formalization

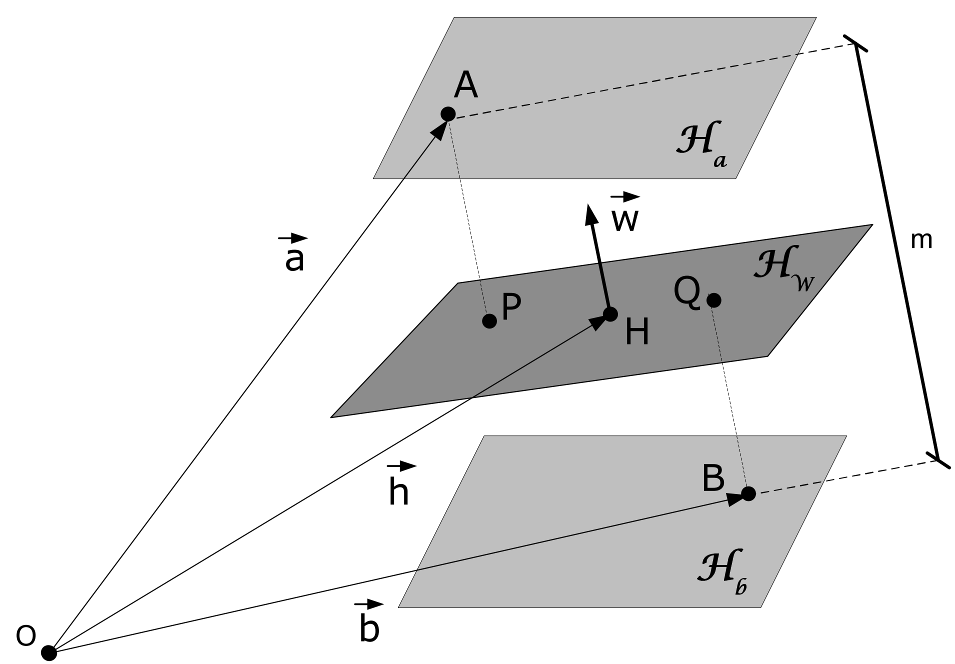
- **The SVM optimization problem:** given support vectors such as $\vec{a}$ and $\vec{b}$, find hyperplane $\mathcal{H}_w$ that maximizes margin $m$



# SVM Technique – Formalization

- $O$: origin of the coordinate system
  - point $A$: a doc from class $c_a$ (belongs to delimiting hyperplane $\mathcal{H}_a$)
  - point $B$: a doc from class $c_b$

- $\mathcal{H}_w$ is determined by a point $H$ (represented by $\vec{h}$) and by a perpendicular vector $\vec{w}$
  - neither $\vec{h}$ nor $\vec{w}$ are known a priori



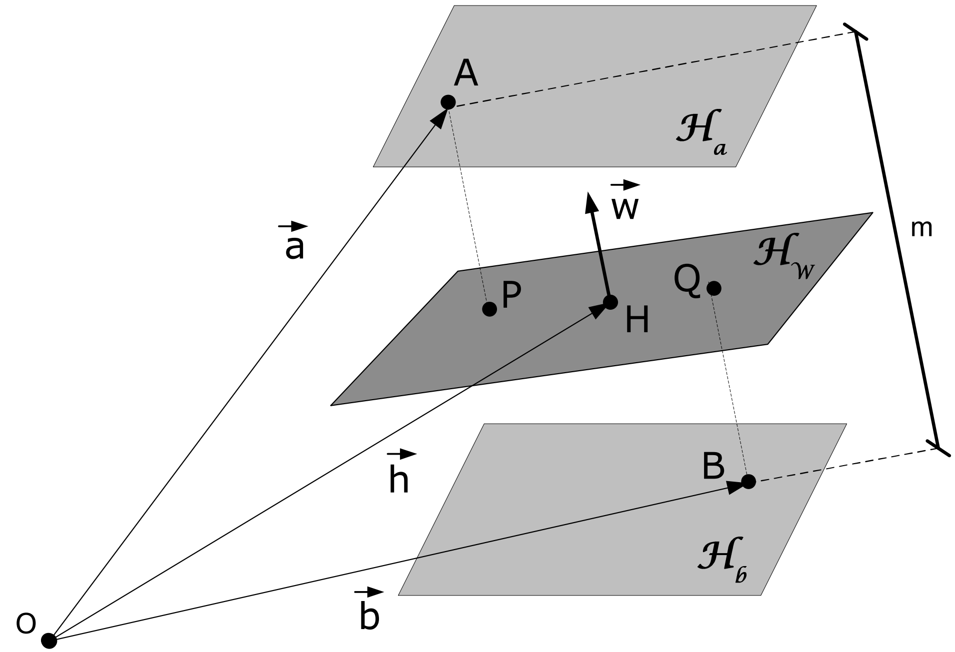
# SVM Technique – Formalization

- $P$: projection of point $A$ on hyperplane $\mathcal{H}_w$
- $\overline{AP}$: distance of point $A$ to hyperplane $\mathcal{H}_w$

$$\overline{AP} = \frac{\vec{a}\vec{w} + k}{|\vec{w}|}$$

- $\overline{BQ}$: distance of point $B$ hyperplane $\mathcal{H}_w$

$$\overline{BQ} = -\frac{\vec{b}\vec{w} + k}{|\vec{w}|}$$



# SVM Technique – Formalization

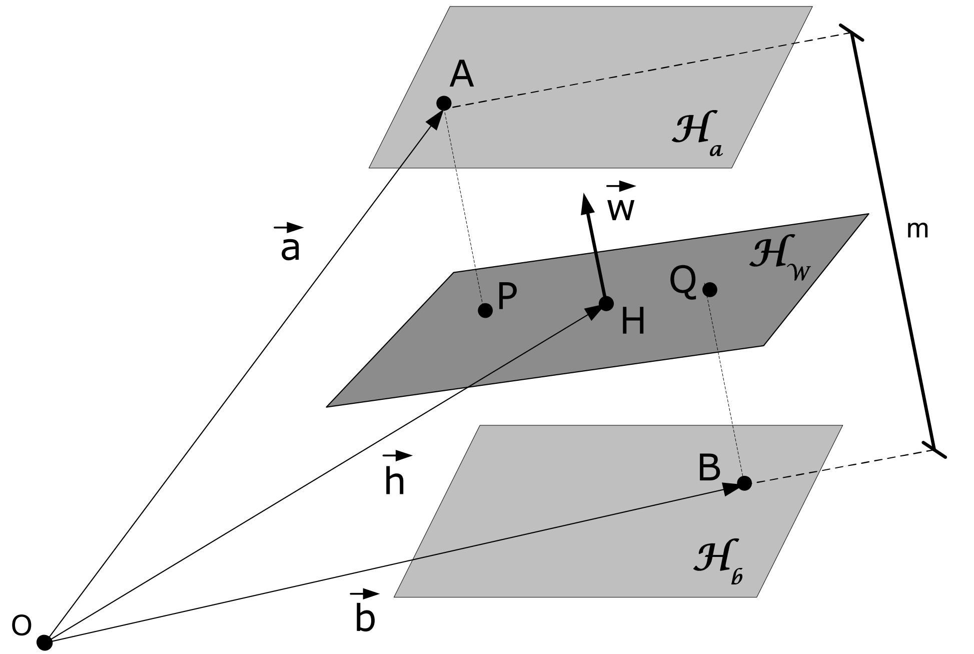
- Margin $m$ of the SVM

$$m = \overline{AP} + \overline{BQ}$$

  is independent of size of $\vec{w}$

- Vectors $\vec{w}$ of varying sizes maximize $m$
- Impose restrictions on $|\vec{w}|$

$$\begin{aligned}\vec{a}\vec{w} + k &= 1\\ \vec{b}\vec{w} + k &= -1\end{aligned}$$



# SVM Technique – Formalization

- Restrict solution to hyperplanes that split margin $m$ in the middle
- Under these conditions,

$$m = \frac{1}{|\vec{w}|} + \frac{1}{|\vec{w}|}$$

$$m = \frac{2}{|\vec{w}|}$$

# SVM Technique – Formalization

- Let,
  - $\mathcal{T} = \{\ldots, [c_j, \vec{z}_j], [c_{j+1}, \vec{z}_{j+1}], \ldots\}$: the training set
  - $c_j$: class associated with point $\vec{z}_j$ representing doc $d_j$
- Then,

**SVM Optimization Problem:**

maximize $m = 2/|\vec{w}|$
subject to

$$\vec{w}\vec{z}_j + b \geq +1 \text{ if } c_j = c_a$$
$$\vec{w}\vec{z}_j + b \leq -1 \text{ if } c_j = c_b$$

- **Support vectors:** vectors that make equation equal to either +1 or -1

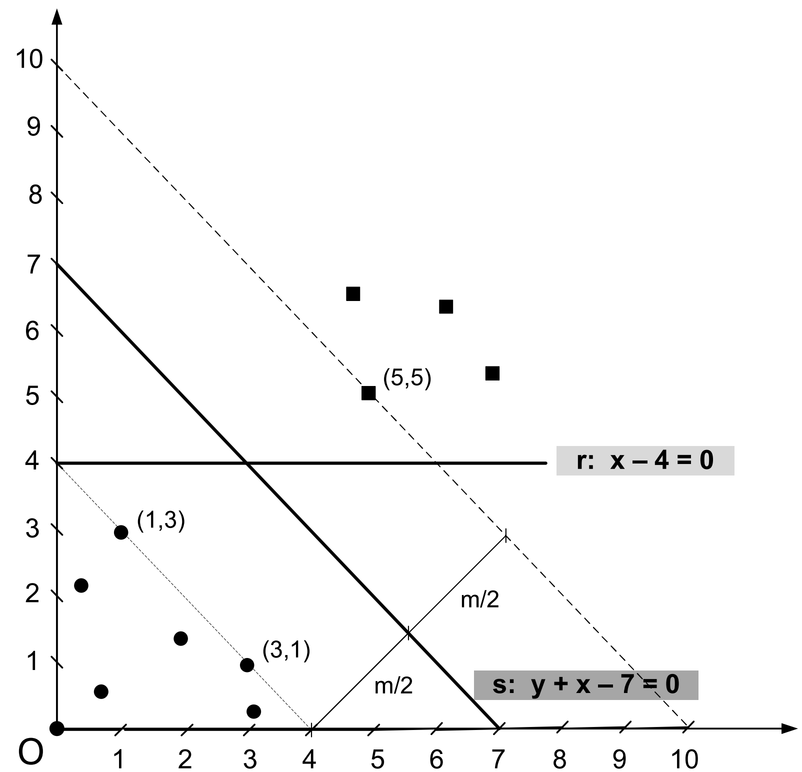
# SVM Technique – Formalization

- Let us consider again our simple example case

- Optimization problem:

  maximize $m = 2/|\vec{w}|$
  subject to

$$\vec{w} \cdot (5,5) + b = +1$$
$$\vec{w} \cdot (1,3) + b = -1$$



# SVM Technique – Formalization

- If we represent vector $\vec{w}$ as $(x, y)$ then $|\vec{w}| = \sqrt{x^2 + y^2}$
- $m = 3\sqrt{2}$: distance between delimiting hyperplanes
- Thus,

$$\begin{aligned} 3\sqrt{2} &= 2/\sqrt{x^2 + y^2} \\ 5x + 5y + b &= +1 \\ x + 3y + b &= -1 \end{aligned}$$

# SVM Technique – Formalization

- Maximum of $2/|\vec{w}|$
  - $b = -21/9$
  - $x = 1/3, y = 1/3$
  - equation of decision hyperplane

$$(1/3, 1/3) \cdot (x, y) + (-21/9) = 0$$

or

$$y + x - 7 = 0$$

# Classification of Documents

- Classification of doc $d_j$ (i.e., $\vec{z}_j$) decided by

$$f(\vec{z}_j) = sign(\vec{w}\vec{z}_j + b)$$

  - $f(\vec{z}_j) = " + "$ : $d_j$ belongs to class $c_a$
  - $f(\vec{z}_j) = " - "$ : $d_j$ belongs to class $c_b$

- SVM classifier might enforce margin to reduce errors
  - a new document $d_j$ is classified
    - in class $c_a$: only if $\vec{w}\vec{z}_j + b > 1$
    - in class $c_b$: only if $\vec{w}\vec{z}_j + b < -1$

# 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 $d_j$
  - run classification for each class
  - each class $c_p$ paired against all others
  - classes of $d_j$: those with largest margins

# SVM with Multiple Classes

- Another solution
  - consider binary classifier for each pair of classes $c_p$ and $c_q$
  - all training documents of one class: positive examples
  - all documents from the other class: negative examples

# Non-Linearly Separable Cases

- SVM has no solutions when there is no hyperplane that separates the data points into two disjoint sets
  - This condition is known as **non-linearly separable case**
- In this case, two viable solutions are
  - **soft margin approach:** allow classifier to make few mistakes
  - **kernel approach**: map original data into higher dimensional space (where mapped data is linearly separable)

# Soft Margin Approach

- Allow classifier to make a few mistakes

  maximize $m = \frac{2}{|\vec{w}|} + \gamma \sum_j e_j$
  subject to
  $\vec{w}\vec{z}_j + k \geq +1$ - $e_j$, if $c_j = c_a$
  $\vec{w}\vec{z}_j + k \leq -1$ + $e_j$, if $c_j = c_b$
  $\forall j$, $e_j \geq 0$

- Optimization is now trade-off between
  - margin width
  - amount of error
  - parameter $\gamma$ balances importance of these two factors

# Kernel Approach

- Compute max margin in transformed feature space

  minimize $m = \frac{1}{2} * |\vec{w}|^2$
  subject to
  $f(\vec{w}, \vec{z}_j) + k \geq +1$, if $c_j = c_a$
  $f(\vec{w}, \vec{z}_j) + k \leq -1$, if $c_j = c_b$

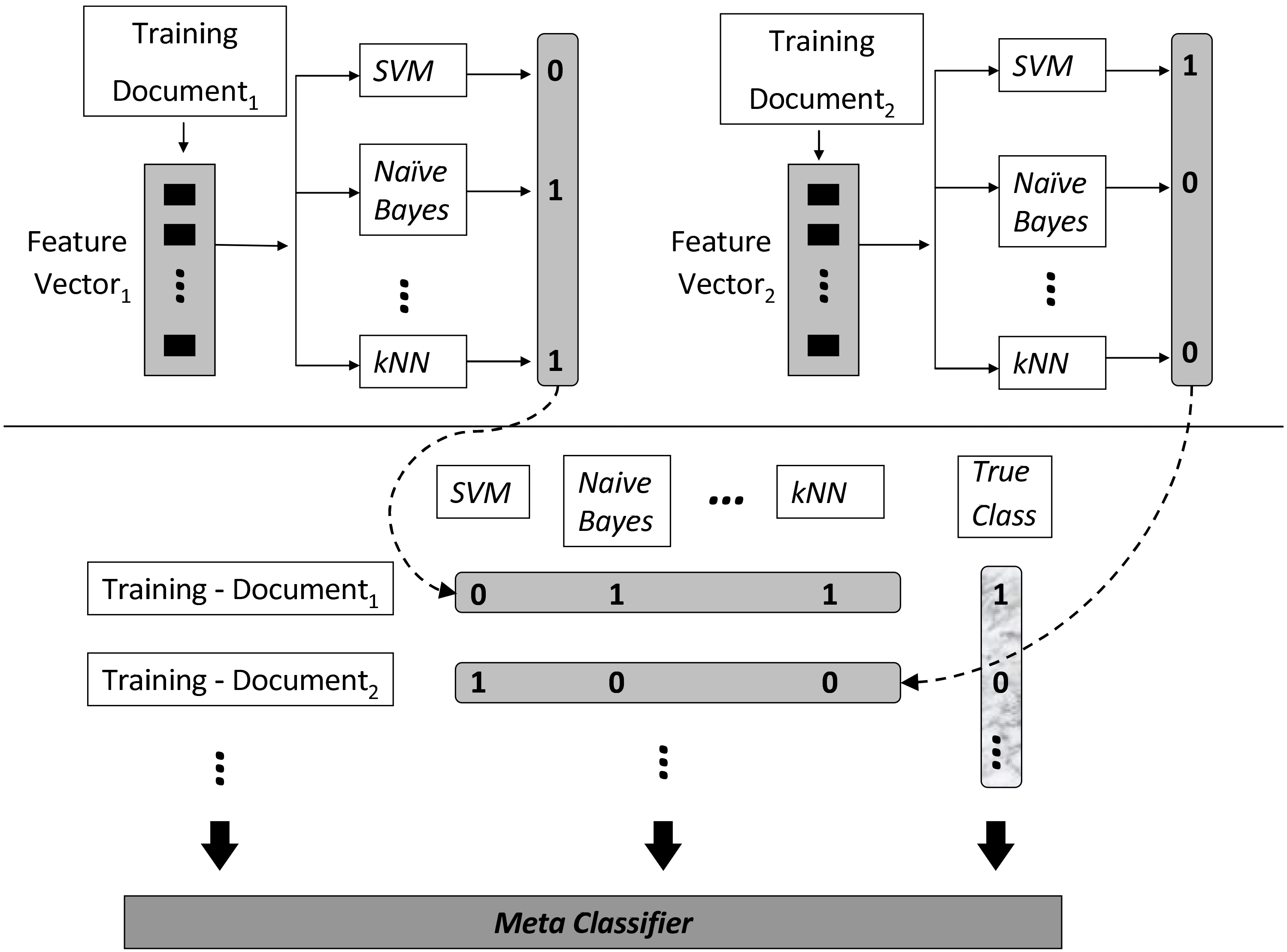
- Conventional SVM case
  - $f(\vec{w}, \vec{z}_j) = \vec{w}\vec{z}_j$, the kernel, is dot product of input vectors
- Transformed SVM case
  - the kernel is a modified map function
    - polynomial kernel: $f(\vec{w}, \vec{x}_j) = (\vec{w}\vec{x}_j + 1)^d$
    - radial basis function: $f(\vec{w}, \vec{x}_j) = exp(\lambda * |\vec{w}\vec{x}_j|^2)$, $\lambda > 0$
    - sigmoid: $f(\vec{w}, \vec{x}_j) = \tanh(\rho(\vec{w}\vec{x}_j) + c)$, for $\rho > 0$ and $c < 0$

# Ensemble Classifiers

# Ensemble Classifiers

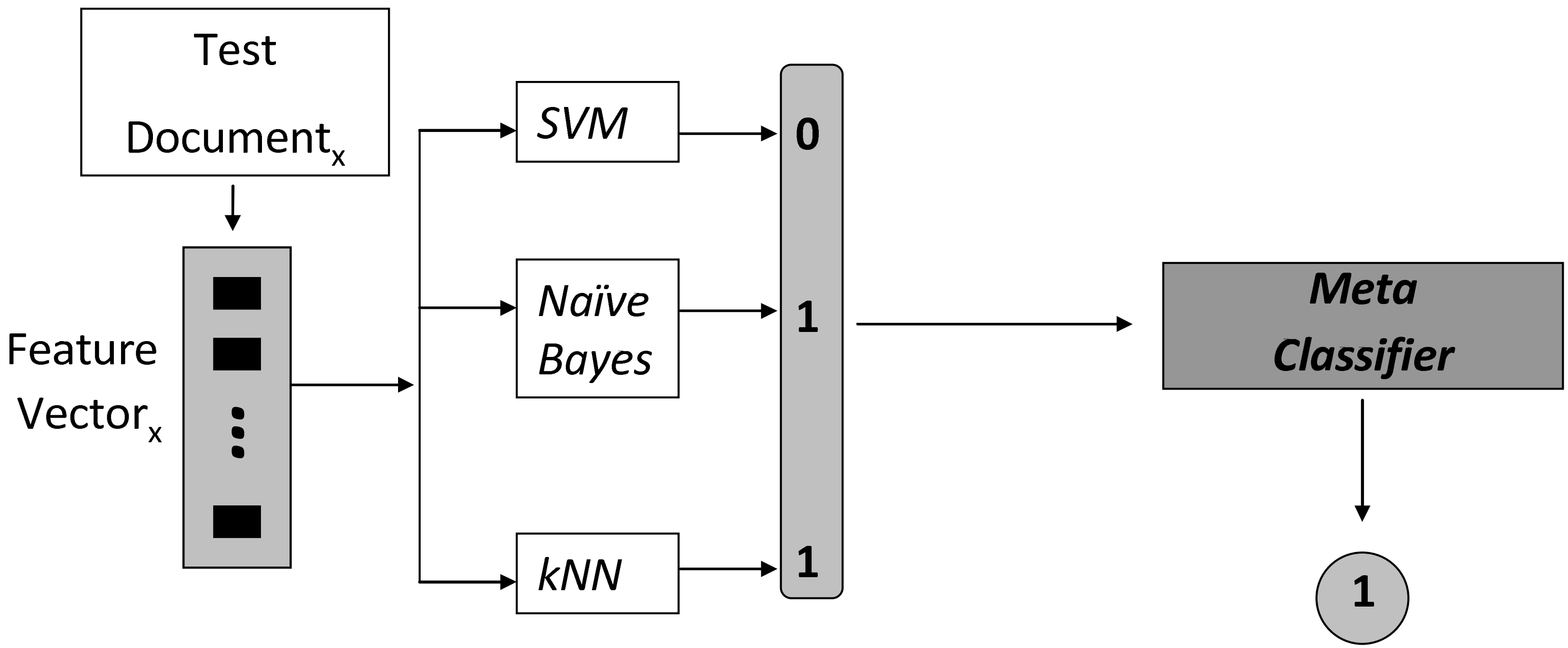
- Combine predictions of distinct classifiers to generate a new predictive score
- Ideally, results of higher precision than those yielded by constituent classifiers
- Two ensemble classification methods:
  - **stacking**
  - **boosting**

# Stacking-based Ensemble



# Stacking-based Classifiers

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



# Stacking-based Classifiers

- With each document-class pair $[d_j, c_p]$ in training set
  - associate predictions made by distinct classifiers
- Instead of predicting class of document $d_j$
  - predict the classifier that best predicts the class of $d_j$, 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

# Boosting-based Classifiers

- Variation of AdaBoost algorithm (Yoav Freund *et al*)

**AdaBoost**

let $\mathcal{T} : \mathcal{D}_t \times \mathcal{C}$ be the training set function;
let $N_t$ be the training set size and $M$ be the number of iterations;
initialize the weight $w_j$ of each document $d_j$ as $w_j = \frac{1}{N_t}$;
**for** $k = 1$ to $M$ {
  learn the classifier function $\mathcal{F}_k$ from the training set;
  estimate weighted error: $err_k = \sum_{d_j | d_j misclassified} w_j \; / \; \sum_{i=1}^{N_t} w_j$;
  compute a classifier weight: $\alpha_k = \frac{1}{2} \times \log\left(\frac{1-err_k}{err_k}\right)$;
  **for** all correctly classified examples $e_j$: $w_j \leftarrow w_j \times e^{-\alpha_k}$;
  **for** all incorrectly classified examples $e_j$: $w_j \leftarrow w_j \times e^{\alpha_k}$;
  normalize the weights $w_j$ so that they sum up to 1;
}

# 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
    - reduces **overfitting**

# Term-Class Incidence Table

- Feature selection
  - dependent on statistics on term occurrences inside docs and classes
- Let
  - $\mathcal{D}_t$: subset composed of all training documents
  - $N_t$: number of documents in $\mathcal{D}_t$
  - $t_i$: number of documents from $\mathcal{D}_t$ that contain term $k_i$
  - $\mathcal{C} = \{c_1, c_2, \ldots, c_L\}$: set of all $L$ classes
  - $\mathcal{T} : \mathcal{D}_t \times \mathcal{C} \rightarrow [0, 1]$: 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 | $n_p$ | $N_t - n_p$ | $N_t$ |

- $n_{i,p}$: # docs that contain $k_i$ and are classified in $c_p$
- $n_i - n_{i,p}$: # docs that contain $k_i$ but are not in class $c_p$
- $n_p$: total number of training docs in class $c_p$
- $n_p - n_{i,p}$: number of docs from $c_p$ that do not contain $k_i$

# 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 $k_i \not\in d_j$: $P(\overline{k}_i) = \frac{N_t - n_i}{N_t}$
  - Probability that $d_j \in c_p$: $P(c_p) = \frac{n_p}{N_t}$
  - Probability that $d_j \not\in c_p$: $P(\overline{c}_p) = \frac{N_t - n_p}{N_t}$
  - 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 $k_i \not\in d_j$ 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 $d_j \not\in c_p$: $P(k_i, \overline{c}_p) = \frac{n_i - n_{i,p}}{N_t}$
  - Probability that $k_i \not\in d_j$ and $d_j \not\in c_p$: $P(\overline{k}_i, \overline{c}_p) = \frac{N_t - n_i - (n_p - n_{i,p})}{N_t}$

# Feature Selection by Doc Frequency

- Let $K_{th}$ be a threshold on term document frequencies
- Feature Selection by Term Document Frequency
  - retain all terms $k_i$ for which $n_i \geq K_{th}$
  - 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 Tf-Idf Weights

- $w_{i,j}$: tf-idf weight associated with pair $[k_i, d_j]$
- $K_{th}$: threshold on tf-idf weights
- Feature Selection by TF-IDF Weights
  - retain all terms $k_i$ for which $w_{i,j} \geq K_{th}$
  - discard all others
  - recompute doc representations to consider only terms retained
- Experiments suggest that this feature selection allows reducing dimensionality of space by a factor of 10 with no loss in effectiveness

# Feature Selection by Mutual Informa

- **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

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

- $K_{th}$: threshold on entropy
- Feature Selection by Entropy
  - retain all terms $k_i$ for which $MI(k_i, C) \geq K_{th}$
  - discard all others
  - recompute doc representations to consider only terms retained

# Feature Selection: Information Gain

- Mutual information uses probabilities associated with the occurrence of terms in documents
- **Information Gain**
  - complementary metric
  - considers probabilities associated with absence of terms in docs
  - balances the effects of term/document occurrences with the effects of term/document absences

# Information Gain

- Information gain of term $k_i$ over set $\mathcal{C}$ of all classes

$$IG(k_i, \mathcal{C}) = H(\mathcal{C}) - H(\mathcal{C}|k_i) - H(\mathcal{C}|\neg k_i)$$

  - $H(\mathcal{C})$: entropy of set of classes $\mathcal{C}$
  - $H(\mathcal{C}|k_i)$: conditional entropies of $\mathcal{C}$ in the presence of term $k_i$
  - $H(\mathcal{C}|\neg k_i)$: conditional entropies of $\mathcal{C}$ in the absence of term $k_i$
  - $IG(k_i, \mathcal{C})$: amount of knowledge gained about $\mathcal{C}$ due to the fact that $k_i$ is known

# Information Gain

- Recalling the expression for entropy, we can write

$$
\begin{aligned}
IG(k_i, \mathcal{C}) &= -\sum_{p=1}^{L} P(c_p) \log P(c_p) \\
&\quad - \left( -\sum_{p=1}^{L} P(k_i, c_p) \log P(c_p|k_i) \right) \\
&\quad - \left( -\sum_{p=1}^{L} P(\overline{k}_i, c_p) \log P(c_p|\overline{k}_i) \right)
\end{aligned}
$$

# Information Gain

- Applying Bayes rule

$$IG(k_i, \mathcal{C}) = -\sum_{p=1}^{L} \Bigg( P(c_p) \log P(c_p) - P(k_i, c_p) \log \frac{P(k_i, c_p)}{P(k_i)} - P(\overline{k}_i, c_p) \log \frac{P(\overline{k}_i, c_p)}{P(\overline{k}_i)} \Bigg)$$

- Substituting previous probability definitions

$$IG(k_i, \mathcal{C}) = -\sum_{p=1}^{L} \left[ \frac{n_p}{N_t} \log\left(\frac{n_p}{N_t}\right) - \frac{n_{i,p}}{N_t} \log \frac{n_{i,p}}{n_i} - \frac{n_p - n_{i,p}}{N_t} \log \frac{n_p - n_{i,p}}{N_t - n_i} \right]$$

# Information Gain

- $K_{th}$: threshold on information gain
- <u>Feature Selection by Information Gain</u>
  - retain all terms $k_i$ for which $IG(k_i, \mathcal{C}) \geq K_{th}$
  - discard all others
  - recompute doc representations to consider only terms retained

# Feature Selection using Chi Square

- Statistical metric defined as

$$\chi^2(k_i, c_p) = \frac{N_t \left(P(k_i, c_p)P(\neg k_i, \neg c_p) - P(k_i, \neg c_p)P(\neg k_i, c_p)\right)^2}{P(k_i)\ P(\neg k_i)\ P(c_p)\ P(\neg c_p)}$$

  quantifies lack of independence between $k_i$ and $c_p$

- Using probabilities previously defined

$$\begin{aligned}
\chi^2(k_i, c_p) &= \frac{N_t\ \left(n_{i,p}\left(N_t - n_i - n_p + n_{i,p}\right) - \left(n_i - n_{i,p}\right)\left(n_p - n_{i,p}\right)\right)^2}{n_p\ (N_t - n_p)\ n_i\ (N_t - n_i)} \\
&= \frac{N_t\ \left(N_t n_{i,p} - n_p n_i\right)^2}{n_p n_i (N_t - n_p)(N_t - n_i)}
\end{aligned}$$

# Chi Square

- Compute either average or max chi square

$$\chi^2_{avg}(k_i) = \sum_{p=1}^{L} P(c_p)\, \chi^2(k_i, c_p)$$
$$\chi^2_{max}(k_i) = max_{p=1}^{L}\, \chi^2(k_i, c_p)$$

- $K_{th}$: threshold on chi square
- <u>Feature Selection by Chi Square</u>
  - retain all terms $k_i$ for which $\chi^2_{avg}(k_i) \geq K_{th}$
  - discard all others
  - recompute doc representations to consider only terms retained

# Evaluation Metrics

# Evaluation Metrics

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

# Contingency Table

- Let
  - $\mathcal{D}$: collection of documents
  - $\mathcal{D}_t$: subset composed of training documents
  - $N_t$: number of documents in $\mathcal{D}_t$
  - $\mathcal{C} = \{c_1, c_2, \ldots, c_L\}$: set of all $L$ classes
- Further let
  - $\mathcal{T} : \mathcal{D}_t \times \mathcal{C} \rightarrow [0, 1]$: training set function
  - $n_t$: number of docs from training set $\mathcal{D}_t$ in class $c_p$
  - $\mathcal{F} : \mathcal{D} \times \mathcal{C} \rightarrow [0, 1]$: text classifiier function
  - $n_f$: number of docs from training set assigned to class $c_p$ by the classifier

# 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 $c_p$
  - $n_t - n_{f,t}$: number of training docs in class $c_p$ that were miss-classified
  - The remaining quantities are calculated analogously

# Accuracy and Error

- Accuracy and error metrics, relative to a given class $c_p$

$$
\begin{aligned}
Acc(c_p) &= \frac{n_{f,t} + (N_t - n_f - n_t + n_{f,t})}{N_t} \\
Err(c_p) &= \frac{(n_f - n_{f,t}) + (n_t - n_{f,t})}{N_t} \\
Acc(c_p) + Err(c_p) &= 1
\end{aligned}
$$

- These metrics are commonly used for evaluating classifiers

# Accuracy and Error

- Accuracy and error have disadvantages
  - consider classification with only two categories $c_p$ and $c_r$
  - assume that out of 1,000 docs, 20 are in class $c_p$
  - a classifier that assumes all docs not in class $c_p$
    - 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 $c_p$

| | $\mathcal{T}(d_j, c_p) = 1$ | $\mathcal{T}(d_j, c_p) = 0$ | |
|---|---|---|---|
| $\mathcal{F}(d_j, c_p) = 1$ | 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

$$
\begin{aligned}
Acc(c_p) &= \frac{10 + 980}{1,000} = 99\% \\
Err(c_p) &= \frac{10 + 0}{1,000} = 1\%
\end{aligned}
$$

# Accuracy and Error

- This classifier is much better than one that guesses that all documents are not in class $c_p$
- 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 $c_p$

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

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

# 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

$$\begin{aligned} P(c_p) &= \frac{10}{10} = 100\% \\ R(c_p) &= \frac{10}{20} = 50\% \end{aligned}$$

# Precision and Recall

- Precision and recall
  - computed for every category in set $\mathcal{C}$
  - great number of values
    - makes tasks of comparing and evaluating algorithms more difficult
- Often convenient to combine precision and recall into a single quality measure
  - one of the most commonly used such metric: *F-measure*

# F-measure

- F-measure is defined as

$$F_\alpha(c_p) = \frac{(\alpha^2+1)P(c_p)R(c_p)}{\alpha^2 P(c_p) + R(c_p)}$$

  - $\alpha$: relative importance of precision and recall
  - when $\alpha = 0$, only precision is considered
  - when $\alpha = \infty$, only recall is considered
  - when $\alpha = 0.5$, recall is half as important as precision
  - when $\alpha = 1$, common metric called $F_1$-measure

$$F_1(c_p) = \frac{2P(c_p)R(c_p)}{P(c_p) + R(c_p)}$$

# F-measure

- Consider again the 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 |

- For this example, we write

$$F_1(c_p) = \frac{2 * 1 * 0.5}{1 + 0.5} \sim 67\%$$

# $F_1$ Macro and Micro Averages

- Also common to derive a unique $F_1$ value
  - average of $F_1$ across all individual categories
- Two main average functions
  - Micro-average $F1$, or $micF_1$
  - Macro-average $F_1$, or $macF_1$

# $F_1$ Macro and Micro Averages

- Macro-average $F_1$ across all categories

$$macF_1 = \frac{\sum_{p=1}^{|\mathcal{C}|} F_1(c_p)}{|\mathcal{C}|}$$

- Micro-average $F_1$ across all categories

$$\begin{aligned} micF_1 &= \frac{2PR}{P+R} \\ P &= \frac{\sum_{c_p \in \mathcal{C}} n_{f,t}}{\sum_{c_p \in \mathcal{C}} n_f} \\ R &= \frac{\sum_{c_p \in \mathcal{C}} n_{f,t}}{\sum_{c_p \in \mathcal{C}} n_t} \end{aligned}$$

# $F_1$ Macro and Micro Averages

- In micro-average $F_1$
  - every single document given the same importance
- In macro-average $F_1$
  - every single category is given the same importance
  - captures the ability of the classifier to perform well for many classes
- Whenever distribution of classes is skewed
  - both average metrics should be considered

# Cross-Validation

- Cross-validation
  - standard method to guarantee statistical validation of results
  - build $k$ different classifiers: $\Psi_1, \Psi_2, \ldots, \Psi_k$
  - for this, divide training set $\mathcal{D}_t$ into $k$ disjoint sets (folds) of sizes

$$N_{t1}, N_{t2}, \ldots, N_{tk}$$

  - classifier $\Psi_i$
    - training, or tuning, done on $\mathcal{D}_t$ minus the $i$th fold
    - testing done on the $i$th fold

# Cross-Validation

- Each classifier evaluated independently using precision-recall or $F_1$ figures
- Cross-validation done by computing average of the $k$ measures
- Most commonly adopted value of $k$ is 10
  - method is called **ten-fold cross-validation**

# Standard Collections

## Reuters-21578

- most widely used reference collection
- constituted of news articles from Reuters for the year 1987
- collection classified under several categories related to economics (e.g., acquisitions, earnings, etc)
- contains 9,603 documents for training and 3,299 for testing, with 90 categories co-occuring in both training and test
- class proportions range from 1,88% to 29,96% in the training set and from 1,7% to 32,95% in the testing set

# Standard Collections

- **Reuters: Volume 1 (RCV1) and Volume 2 (RCV2)**
  - RCV1
    - another collection of news stories released by Reuters
    - contains approximately 800,00 documents
    - documents organized in 103 topical categories
    - expected to substitute previous Reuters-21578 collection
  - RCV2
    - modified version of original collection, with some corrections

# Standard Collections

## OHSUMED

- another popular collection for text classification
- subset of Medline, containing medical documents (title or title + abstract)
- 23 classes corresponding to MesH diseases are used to index the documents

# Standard Collections

- **20 NewsGroups**
  - third most used collection
  - approximately 20,000 messages posted to Usenet newsgroups
  - partitioned (nearly) evenly across 20 different newsgroups
  - categories are the newsgroups themselves

# Standard Collections

- Other collections
  - WebKB hypertext collection
  - ACM-DL
    - a subset of the ACM Digital Library
  - samples of Web Directories such as Yahoo and ODP

# Organizing the Classes
# Taxonomies

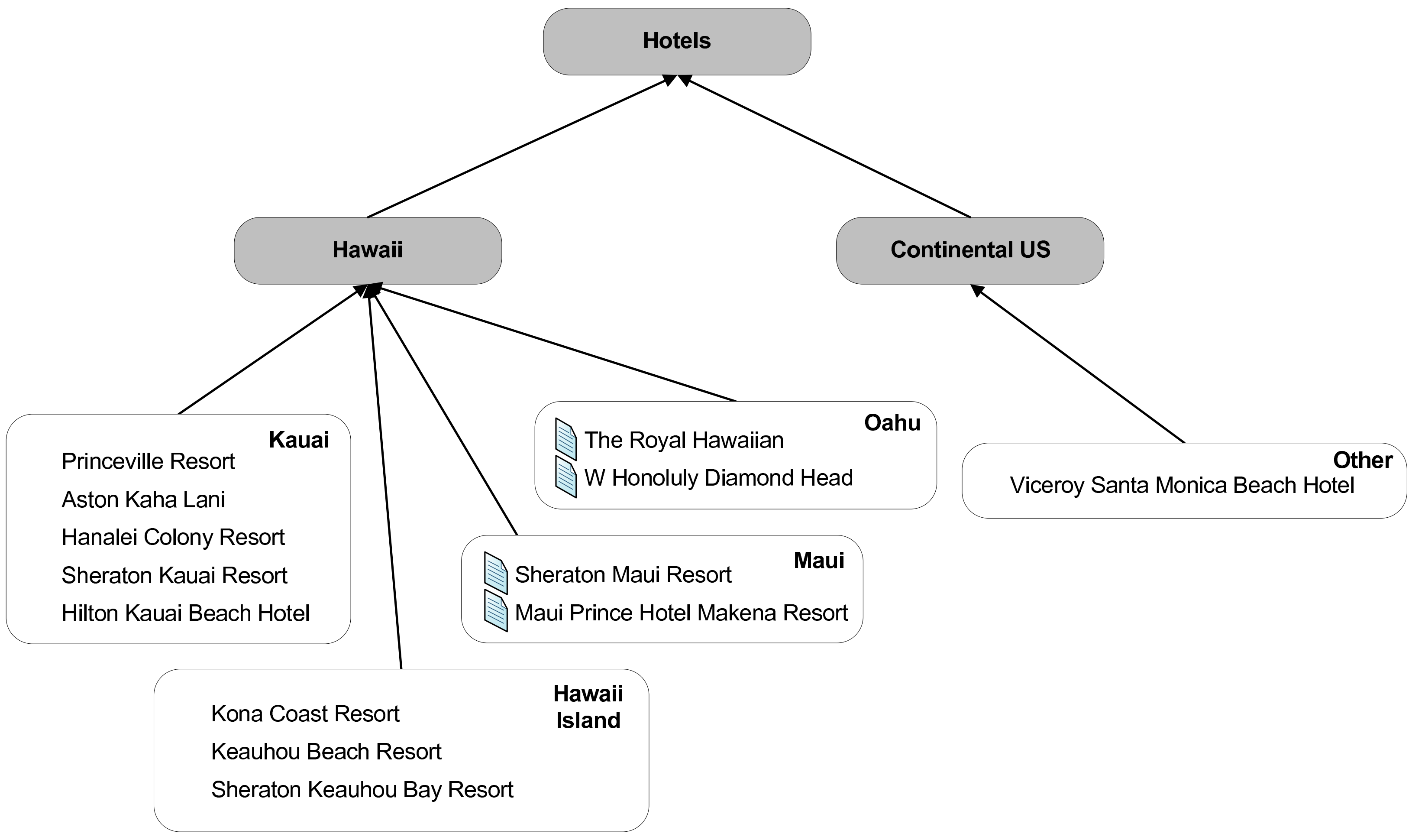
# Taxonomies

- Labels provide information on semantics of each class
- Lack of organization of classes restricts comprehension and reasoning
- Hierarchical organization of classes
  - most appealing to humans
  - hierarchies allow reasoning with more generic concepts
  - also provide for specialization, which allows breaking up a larger set of entities into subsets

# Taxonomies

- To organize classes hierarchically use
  - specialization
  - generalization
  - sibling relations
- Classes organized hierarchically compose a **taxonomy**
  - relations among classes can be used to fine tune the classifier
  - taxonomies make more sense when built for a specific domain of knowledge

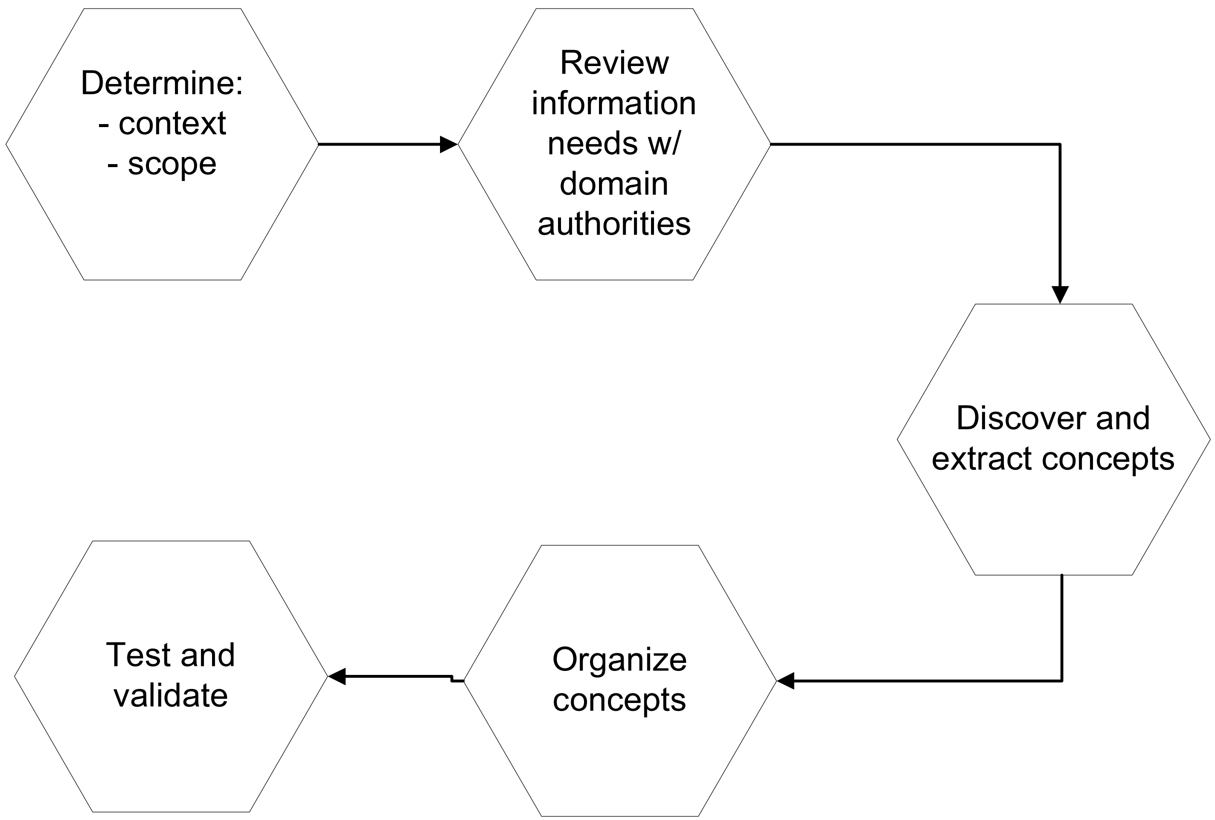
# Taxonomies



Hotels
Hawaii
Continental US
Kauai
Princeville Resort
Aston Kaha Lani
Hanalei Colony Resort
Sheraton Kauai Resort
Hilton Kauai Beach Hotel
Oahu
The Royal Hawaiian
W Honoluly Diamond Head
Maui
Sheraton Maui Resort
Maui Prince Hotel Makena Resort
Hawaii Island
Kona Coast Resort
Keauhou Beach Resort
Sheraton Keauhou Bay Resort
Other
Viceroy Santa Monica Beach Hotel

# Taxonomies

- Taxonomies are built **manually** or **semi-automatically**
- Process of building a taxonomy:



# Taxonomies

- Manual taxonomies tend to be of superior quality
  - better reflect the information needs of the users
- Automatic construction of taxonomies
  - needs more research and development
- Once a taxonomy has been built
  - documents can be classified according to its concepts
  - can be done manually or automatically
  - automatic classification is advanced enough to work well in practice