Chapter 5

Text Classification

Introduction

- Since ancient times, librarians had to deal with the issue of storing documents for later retrieval and reading
- As time passed, the size of the collection grew and the problem hardened
- To minorate the problem, librarians started **labeling the documents**
- A very first approach for labeling documents is to assign a single identifier to each document

Introduction

- However, this not solve the more generic problem of finding documents on a specific subject or topic
- In this case, the natural solution is to
  - group documents by common topics, and
  - name these groups with meaningful labels
- Each labeled group is call a **class**, that is a set of documents that can be described by the group label
- The process of inserting the documents into the classes is commonly referred to as **text classification**

The Need to Organize Information

- Every day we receive documents through postal mail that we feel compelled to store for later reference
  - These are utility bills, bank account reports, tax filings
- As the volume of itens increases, there is a natural need to organize them somehow
- For this, we usually separate the itens into files, label them in a meaningful way, and store them in a cabinet
- The process of separating documents into files is a **text classification procedure**

The Need to Organize Information

- Text classification provides a means to organize information which allows better understanding and interpretation of the data
- To illustrate, consider a large company that produce thousands of documents related to its business
- These documents constitute a valuable asset to support the business decision making process
- To organize the corporate information we need a **knowledge organization system**
- Such systems allow defining
  - a set of classes, organizing them hierarchically
  - classifying the documents of the collection into these classes automatically

A Characterization of Algorithms

A Characterization of Algorithms

- A classification algorithm can be fundamentally of two types, as we now discuss:
  - unsupervised
  - supervised

A Characterization of Algorithms

Unsupervised Algorithms

Unsupervised Algorithms

- Unsupervised algorithm: no examples of documents that belong to each class is given as input
- Consider, for instance, that the only input data are the documents in the collection
- In this case, the task of the classifier is to separate the documents in groups in fully automatic fashion
- This procedure is frequently referred to as **clustering**

Unsupervised Algorithms

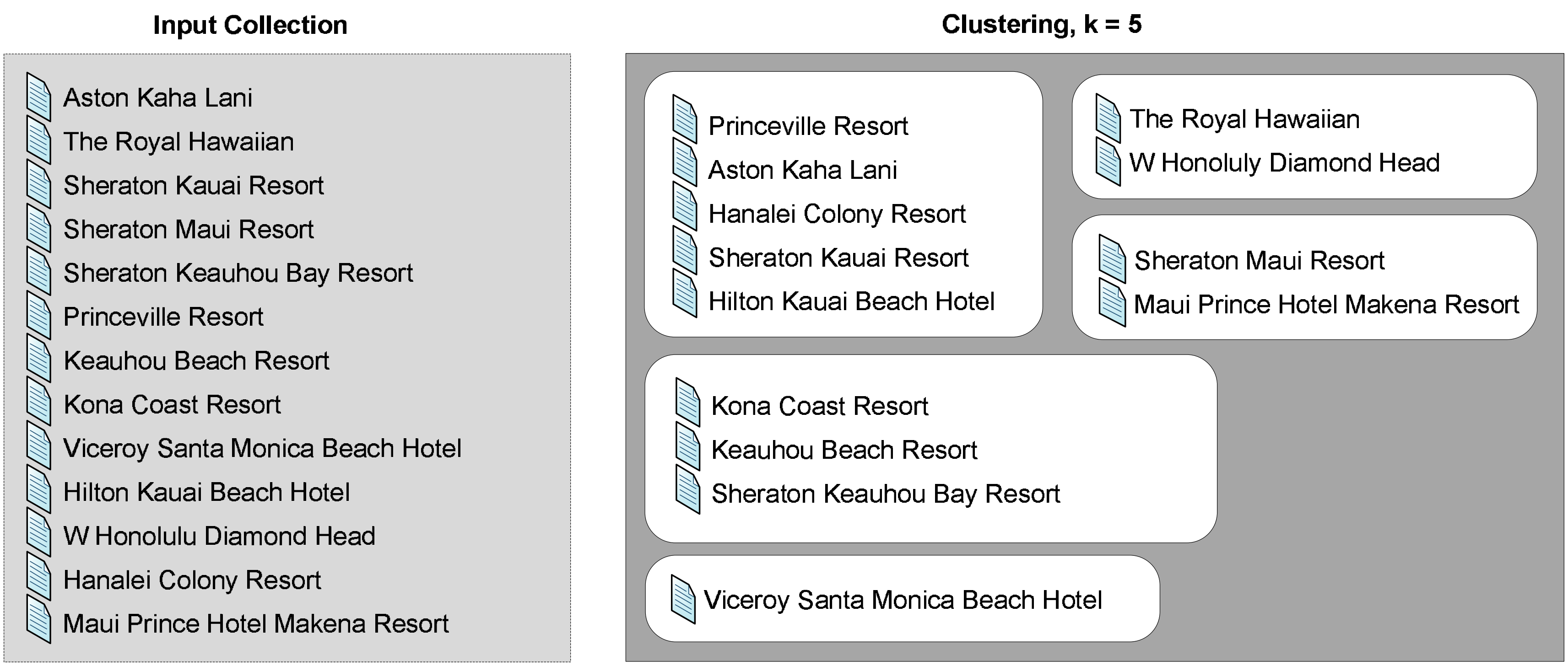
- A second form of algorithm is to specify the classes without any information on training examples
- The classifier matches the text of the documents to the class labels to guide the classification process
- The classifier can use a ranking function to define rewards and penalties on the classification of a doc
- However, matching of text documents to the few terms in a class label might yield poor results

Unsupervised Algorithms

- Unsupervised classification is useful, for instance, to gain initial insight into the data
- It might also be used to gain insight on the number of classes of a set of docs
- In its most general form, the unsupervised text classification problem can be defined as follows
  - Given a collection of N documents, separate them automatically into clusters
  - Following, generate labels for each cluster in fully automatic fashion

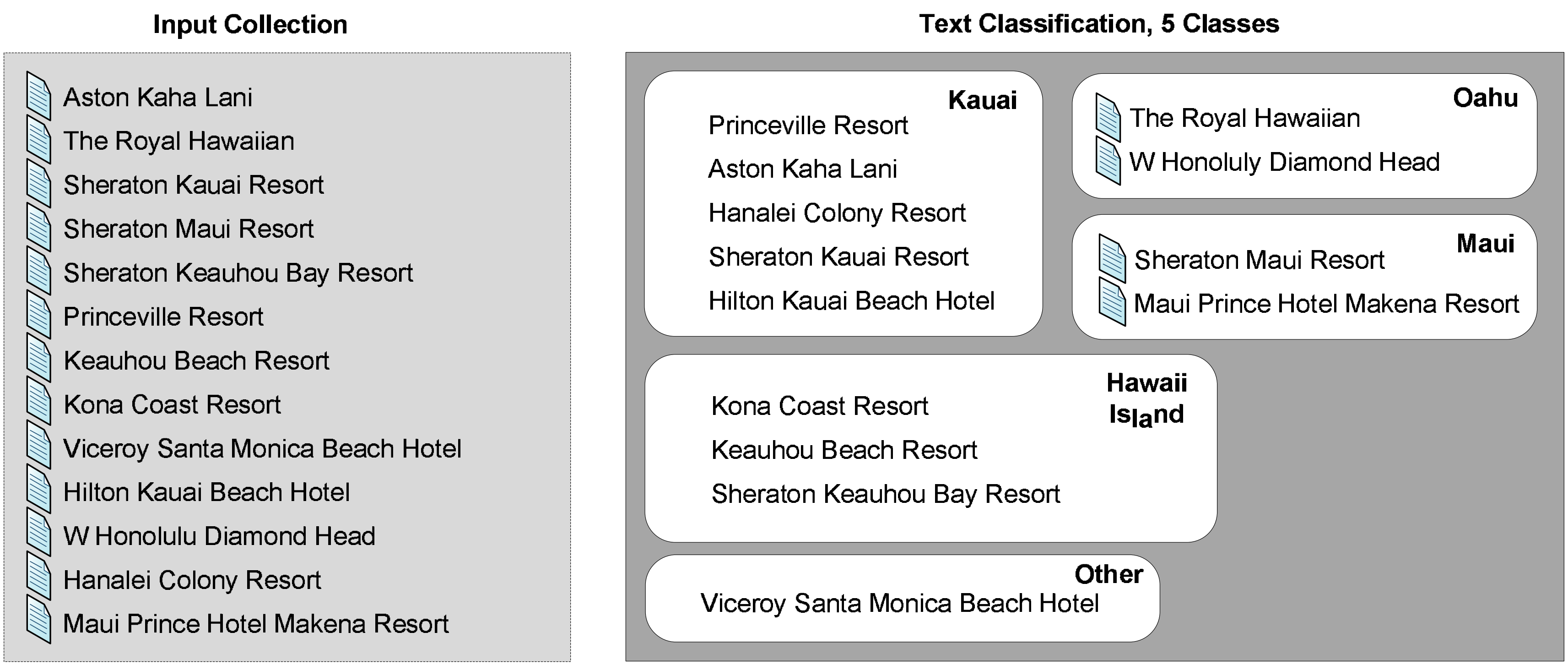
Unsupervised Algorithms

- Unsupervised text classification applied to Web pages of hotels in Hawaii: **clustering**



Unsupervised Algorithms

- Unsupervised text classification applied to Web pages of hotels in Hawaii: **assignment of classes to hotels**
- Each class is composed of hotels located in a same island



Unsupervised Algorithms

- While this clustering is naturally appealing to humans, it is difficult to be generated by an automatic procedure
- The reason is that the hotel Web pages contain many terms in common
- Without human understanding, it is very hard to establish that terms describe the hotel locations
- Thus, most likely, an automatic clustering algorithm will not generate the clusters shown

Unsupervised Algorithms

- Even if the right clusters had been produced, we would still have to label these clusters
- Labels generated automatically tend to be very distinct from labels specified by humans
- Thus, solving the whole classification problem with no provision of any form of human knowledge is hard
- This continues to be an excessively complex task, particularly because of the labelling of the clusters
- Unsupervised classification is more effective when class labels have been provided

A Characterization of Algorithms

Supervised Algorithms

Supervised Algorithms

- An algorithm is said to be **supervised** when it uses human provided information as input data
- In the standard case, a set of classes and examples of documents for each class are provided
- The examples are determined by human specialists and constitute what we call the **training set**
- This training set is used to **learn** a classification function
- Once this function has been learnt, it is used to classify new unseen documents

Training Examples

- For instance, one can specify 4 classes with an average number of 100 documents in each of them
- These documents are known to belong to the classes, as determined by human experts
- The classifier can then determine, for instance, the terms that occur most frequently in each class
- These terms can be used to produce an initial description of that class
- An analogous process can be repeated in the case the examples are provided for clusters

Training Examples

- Larger the number of training examples, usually better is the fine tunning of the classifier
- Once the fine tunning is completed, the classifier can be applied to an **unseen set of objects**
- This set objects is usually referred to as the **test set**

A Characterization of Algorithms

Organizing the Classes – Taxonomies

Taxonomies

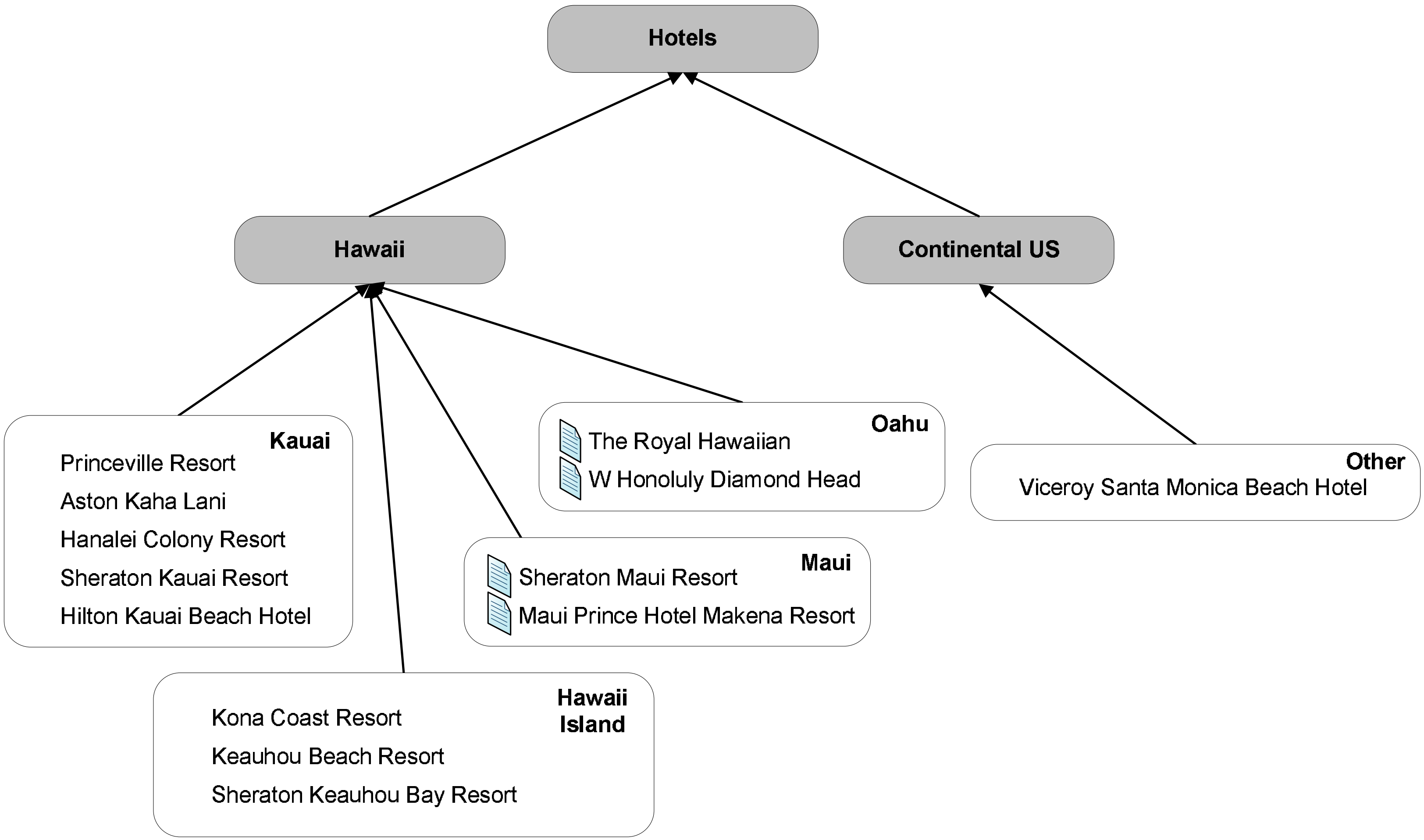
- Labeling provides information on the semantics of each class
- However, no organization of the classes is provided
- Lack of organization of the classes imposes restrictions to comprehension and reasoning
- Among all sorts of organization, the most appealing one is the **hierarchical organization**
- Hierarchies allow us to reason in terms of more generic concepts
- They also provide for specialization which allows breaking up a larger set of entities into subsets

Taxonomies

- We can organize the classes hierarchically using specialization, generalization, and sibling relations
- Classes organized hierarchically in this fashion compose a **taxonomy**
- In this case, the relations among the classes can be used to further fine tune the classifier
- Taxonomies make more sense when built for a specific domain of knowledge

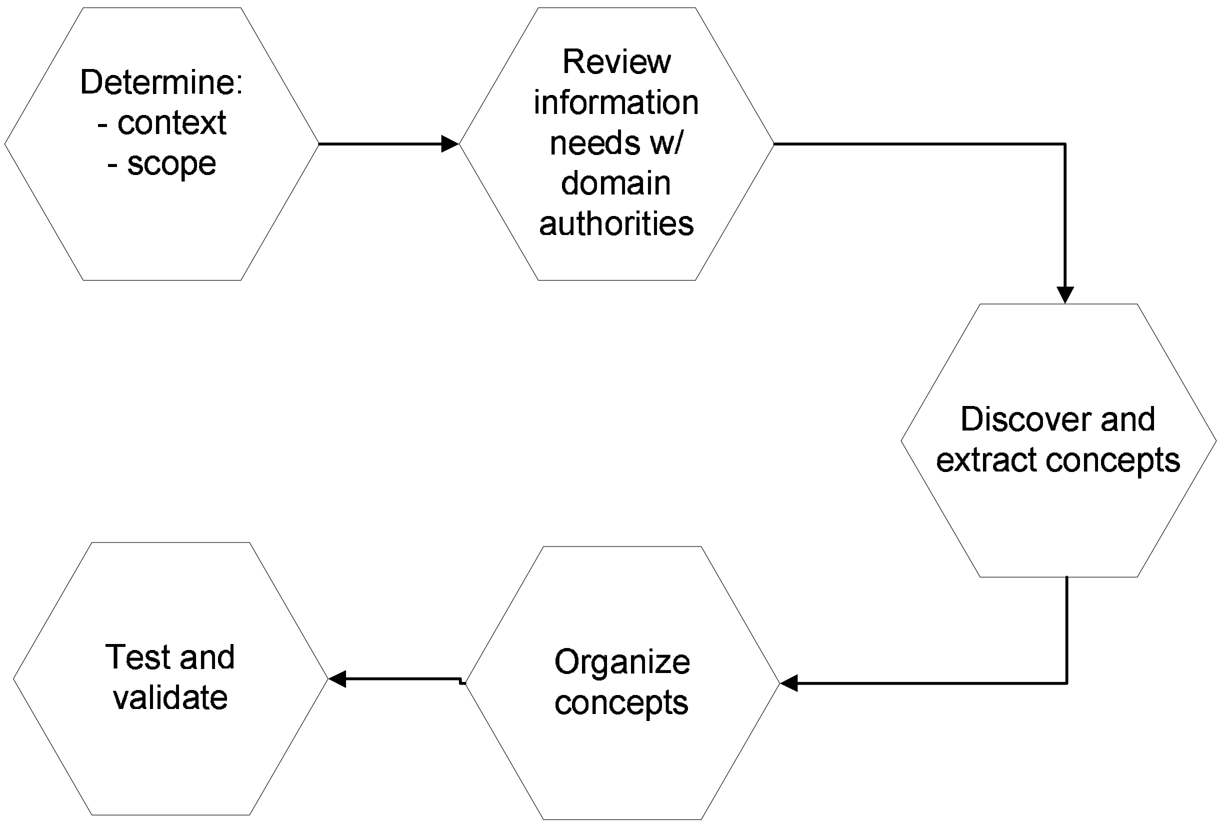
Taxonomies

- Organization of Web pages of hotels in Hawaii in a geo-referenced taxonomy



Taxonomies

- Usually, taxonomies are built **manually** or **semi-automatically** using complex procedures
- The process of building a taxonomy:



Taxonomies

- Manual taxonomies tend to be of superior quality, and better reflect the information needs of the users
- The automatic construction of taxonomies is one area of technology that needs more research and development
- Once a taxonomy has been built, the documents of the collection can be classified according to its concepts
- This can be accomplished manually or automatically
- Automatic classification algorithms are advanced enough to work well in practice

The Text Classification Problem

The Text Classification Problem

- The text classification problem, in its more general form, can be stated as follows
  - Given (i) a collection $\mathcal{D}$ of documents, and (ii) a set $\mathcal{C}$ of classes with their respective labels,
  - Determine a classification method to automatically assign documents to one or more classes
- The method itself is frequently referred to as **the classifier**

The Text Classification Problem

- **Fundamental intuition**: the words in a document should be matched against the labels of each class
- This matching function can take many different forms, but always guides the classification procedure
- The problem can be made more sophisticated by adding two extra components to the input
  - First, the classes might be organized hierarchically composing a taxonomy
  - Second, for each class we might have a number of example documents pre-classified by humans (training set)

The Text Classification Problem

- In a more restrict sense, the problem can be defined as follows
  - Given (i) a collection $\mathcal{D}$ of documents, and (ii) a set $\mathcal{C} = \{c_1, c_2, \ldots, c_L\}$ of L classes with their respective labels,
  - A text classifier is a binary function $\mathcal{F} : \mathcal{D} \times \mathcal{C} \rightarrow \{0, 1\}$
- That is, $\mathcal{F}$ is a function that assigns a value of 0 or 1 to each pair $[d_j, c_p]$, such that $d_j \in \mathcal{D}$ and $c_p \in \mathcal{C}$
- If the value assigned is 1, we say that the document d_j is a member of class c_p
- If the value assigned is 0, we say that the document d_j is not a member of class c_p

The Text Classification Problem

- If no restrictions are posed to the classifier, two or more classes might be assigned to a single document
- In this case, we say that the classifier is of type **multi-label**
- A different problem results if we restrict the classifier to assigning a single class to each document
- In this case, we say that the classifier is of type **single-label**

The Text Classification Problem

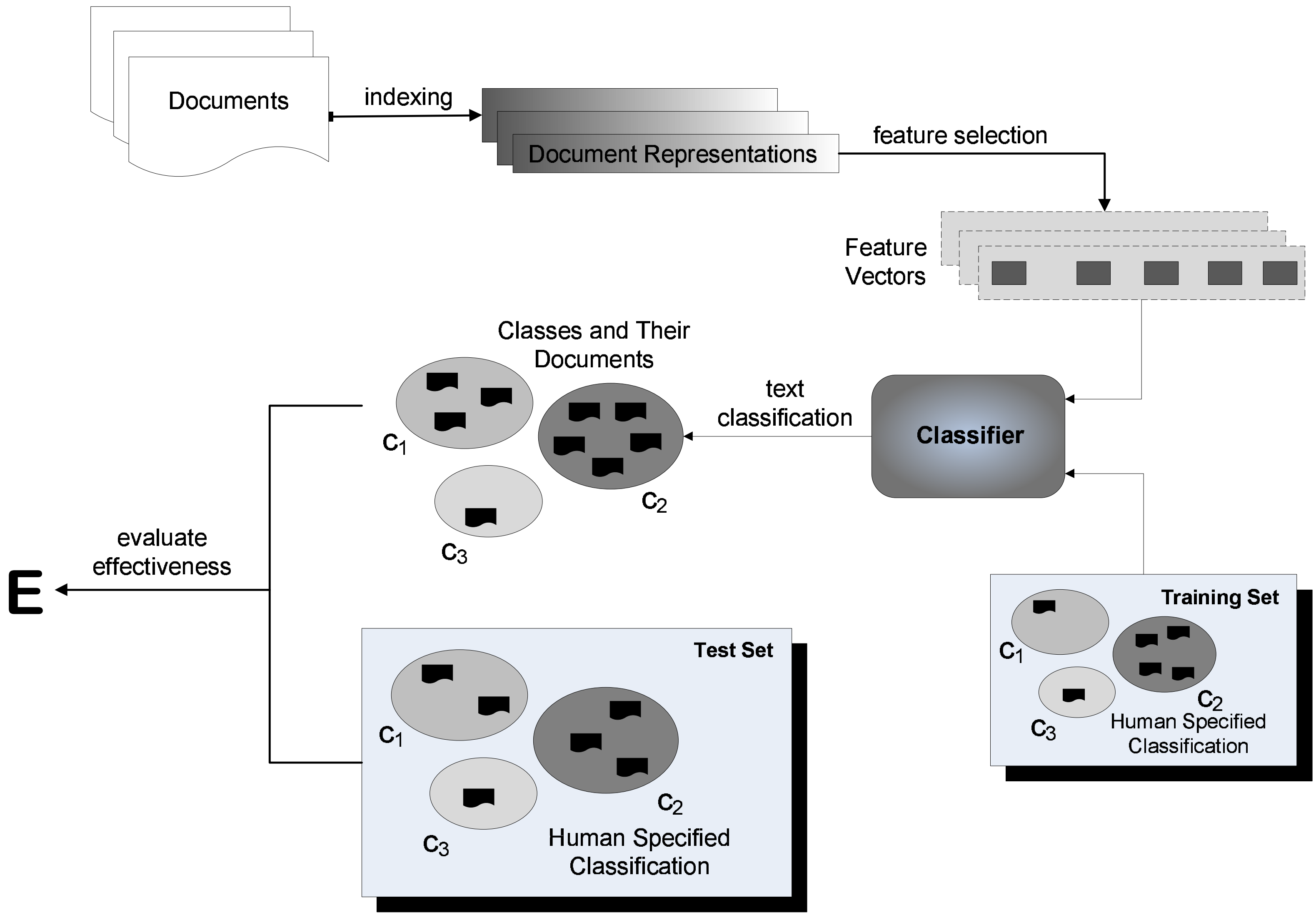
- The above definition of our classification function $\mathcal{F}$ is binary
- However, the function $\mathcal{F}$ might be built to compute a degree of membership of document d_j in the class c_p
- Then, we say that there are a number of documents that are candidates to be members of class c_p
- Further, we can present them to a user ordered by the decreasing values of $\mathcal{F}(d_j, c_p)$
- We also can use this rank to enforce a decision on whether a document belongs to a class or not

The Text Classification Problem

- To improve its effectiveness, our classifier might take advantage of a **training set**, as follows
  - Given a sub-collection $\mathcal{D}_t \subset \mathcal{D}$ of training documents,
  - A training set function $\mathcal{T} : \mathcal{D}_t \times \mathcal{C} \rightarrow \{0, 1\}$ assigns a value of 0 or 1 to each pair $[d_j, c_p]$, $d_j \in \mathcal{D}_t$ and $c_p \in \mathcal{C}$, according to the judgement of human specialists
- The training set function $\mathcal{T}$ is used to fine tune the classifier
- Any evaluation of the classifier must be done considering only documents outside the training set

The Text Classification Problem

- The text classification process



The Text Classification Problem

- Given a collection of documents, we first index them producing document representations
- In the full text logical view, a representation of a document d_j is the set of all its terms (or words)
- Each term of the document representation is considered as a separate variable or **feature**
- We can select a subset of the terms to represent the documents
- This is done through a process of **feature selection**, which reduces the document representations

The Text Classification Problem

- To improve its effectiveness, the classifier might also go through a process of fine tunning its internal parameters
- This is accomplished using a training set
- Once its parameters have been fine tunned, the classifier is used to classify the documents in a test set
- The final phase of the process is to evaluate the effectiveness of the classification
- The evaluation of a classifier is done by comparing results with a classification produced by humans

Classification Algorithms

Classification Algorithms

- In here, we discuss distinct approaches for performing document classification
- We do cover state-of-the art approaches that deliver top performance
- For each algorithm, we present first the basic technique used as its foundation

Classification Algorithms

Decision Trees

Decision Trees

- A **decision tree (DT)** is a supervised classification method
- It uses a training set to build classification rules organized as paths in a tree
- These tree paths can then be used to classify documents outside the training set
- One of the advantages of the approach is that the rules in the tree are amenable to human interpretation
- This data structure facilitates the interpretation of the results of the classification process

Basic Technique

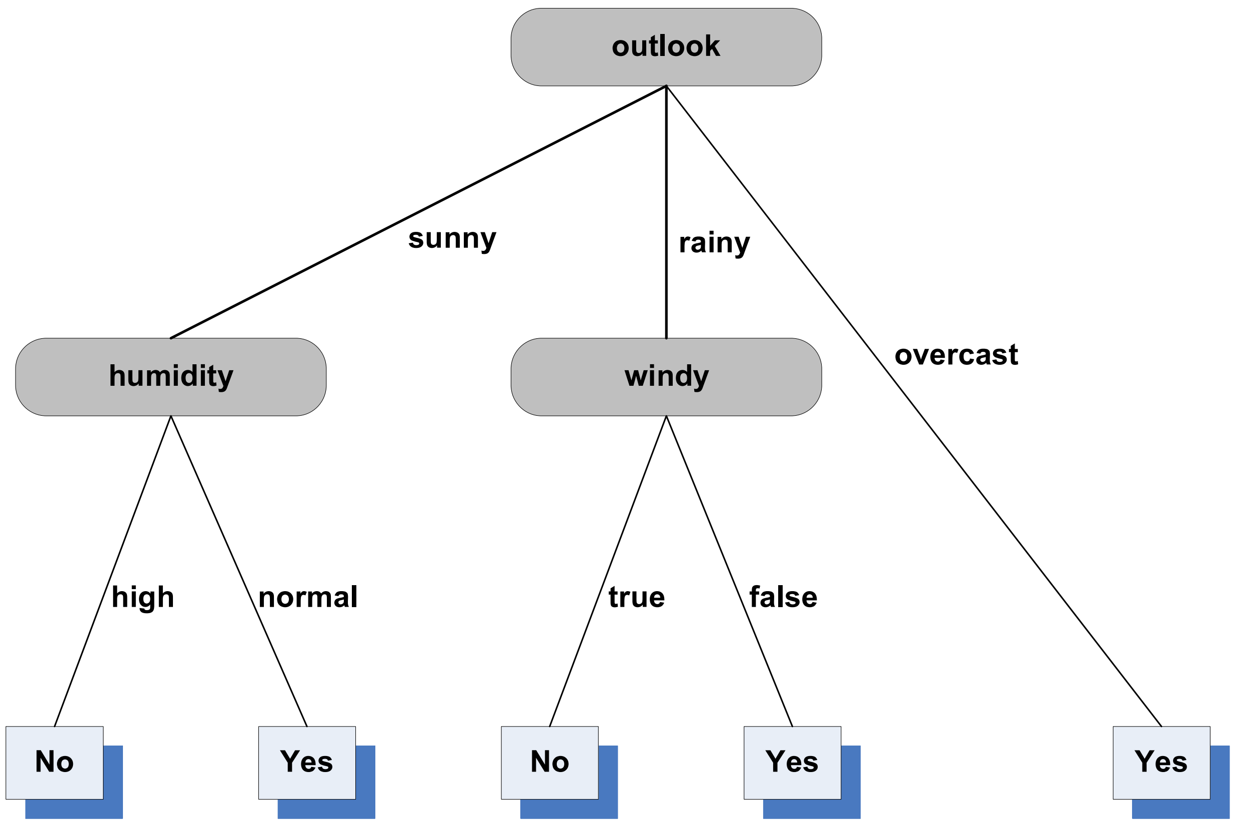
- Consider the small relational database in Table below

	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

- A decision tree for this database is a data structure that allows predicting the values of a given attribute

Basic Technique

- To illustrate, the DT below allows predicting the values of the attribute Play, given that we know the values for attributes like Outlook, Humidity, and Windy



Basic Technique

- The internal nodes are associated with attribute names and the edges are associated with attribute values
- A recursive traversal of the DT allows deciding the most appropriate value for the attribute “Play”.
- In the case of tuple 11, the decision would be “not to play” based on the rule induced by the path $(Outlook = sunny) \wedge (Humidity = high)$

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

Basic Technique

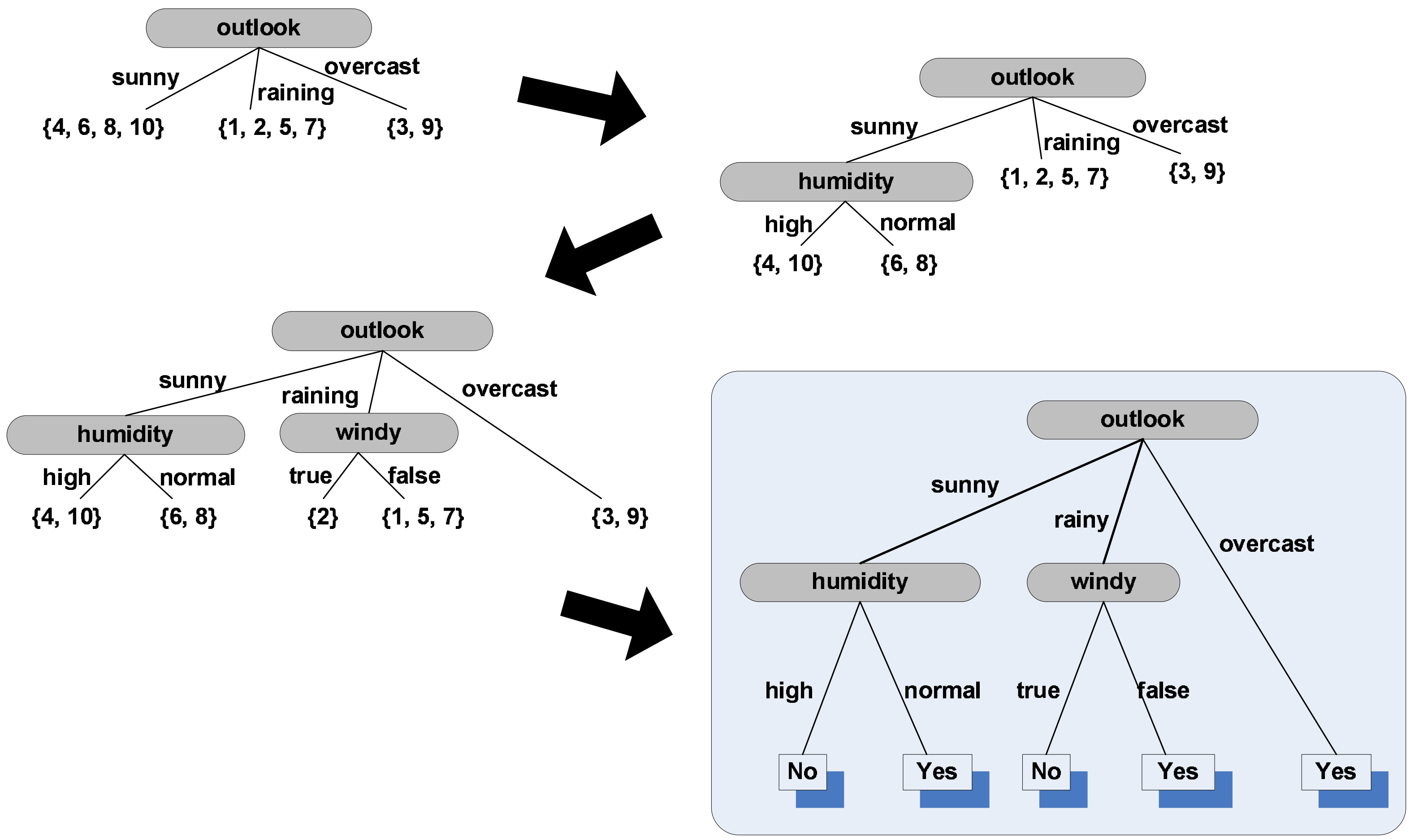
- Notice that our predictions are based on the instances that we have seen in the example database
- A new instance that violates these predictions will lead to an erroneous prediction
- We say that the example database works as a training set for building the decision tree

The Splitting Process

- Given a training set, a DT model for the database can be built using a recursive splitting strategy
- Consider that the objective is to build a decision tree for predicting the values of attribute Play
- The first step is to select one of the attributes, other than Play, to be the root of the decision tree
- The corresponding attribute values are then used to split the tuples in the database into subsets
- For each subset of tuples, a second splitting attribute is selected and the process is repeated

The Splitting Process

- Figure below shows an step by step example of this splitting process



The Splitting Process

- Notice that the splitting process is strongly affected by the order with which the split attributes are selected
- Depending of this ordering, the tree might become unbalanced
- This is one of the key challenges while devising a splitting strategy
- Usually, balanced or near-balanced trees work better and more efficient for predicting attribute values
- Thus, a common rule of thumb is to select attributes that reduce the average path length of the leaves

Classification of Documents

- For document classification, with each internal node in the tree we associate an index term
- With each leave in the tree we associate a document class
- Further, with the edges we associate binary predicates that indicate the presence/absence of an index term
- One alternative is to verify the weight of the term in the document to decide which path to take in the the tree
- This recursive traversal is conducted for each document to be classified
- Once a leaf is reached the class node associated with it becomes the document class

Classification of Documents

- Let V be a set of nodes
- A tree $T = (V, E, r)$ is an acyclic graph on V where
  - $E \subseteq V \times V$ is the set of edges and
  - $r \in V$ is called the root of T
- Given an edge (v_i, v_j), v_i is considered the father node and v_j is the child node
- We designate by I the set of all internal nodes and by $\overline{I}$ the set of all leaf nodes

Classification of Documents

- Given a tree T, we can associate information on documents and their classes with the tree
- By doing so, we create a decision tree for document classification, as follows
- Let
  - $K = \{k_1, k_2, \ldots, k_t\}$ be the set of index terms of a doc collection
  - C be the set of all classes, as before
  - P be a set of logical predicates on the index terms

Classification of Documents

- Further, let
  - A decision tree $DT = (V, E; r; l_I, l_L, l_E)$ is a six-tuple where $(V; E; r)$ is a tree whose root is r
  - $l_I : I \rightarrow K$ is a function that associates with each internal node of the tree one or more index terms
  - $l_L : \overline{I} \rightarrow C$ is a function that associates with each non-internal (leaf) node a class $c_p \in C$
  - $l_E : E \rightarrow P$ is a function that associates with each edge of the tree a logical predicate from P

Classification of Documents

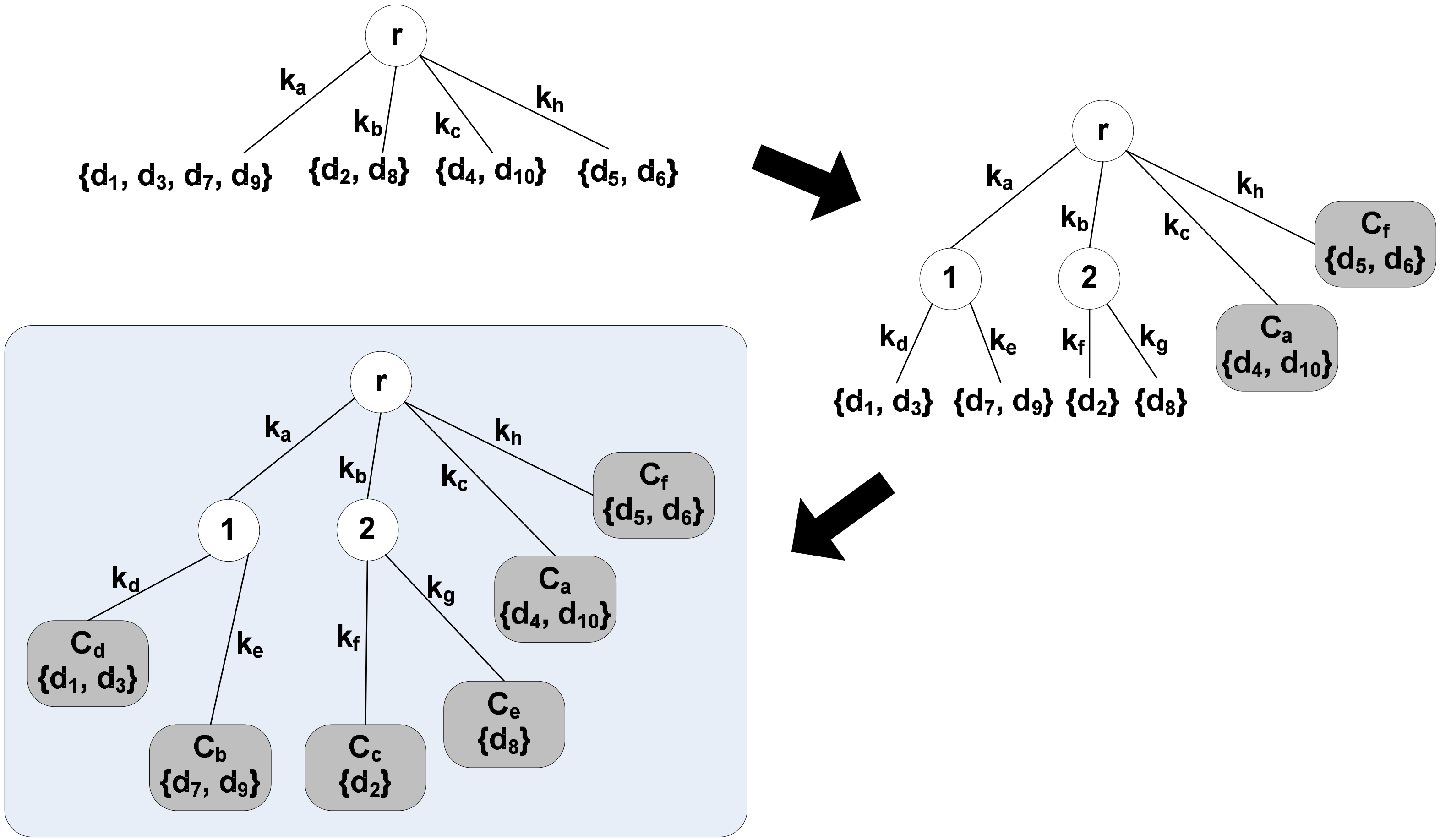
- Given a training set, a decision tree model for a class c_p can be built using a recursive splitting strategy
- The **first step** is to associate all documents to the root
- The **second step** is to select a number of index terms that provide a good separation of the documents
- To illustrate, consider that terms k_a, k_b, k_c, and k_h have been selected for this first split
- Then, the documents in the root are separated into 4 subsets

Classification of Documents

- The edge connecting a subset to the root is labelled with the respective index term
- Notice that a same document might appear in more than one subset
- Following, new splitting terms are selected for each doc subset and the process is repeated recursively
- At each branch, the recursion is stopped whenever all documents of a subset belong to a same class

Classification of Documents

- Splitting process for inducing a decision tree for a collection of documents



Classification of Documents

- The crucial point of this process is the procedure for selecting splitting terms
- While distinct procedures can be used, information gain and entropy are the most commonly used ones
- Selection of terms with high information gain tends
  - to increase the number of branches at a given level, and
  - to reduce the number of documents in each resultant subset
- This tends to yield smaller and less complex decision trees

Classification of Documents

- Decision trees have some inherent problems such as missing or unknown values
- These appear when the document to be classified does not contain some terms used to build the DT
- In this case, it is not clear which branch of the tree should be traversed
- One approach to deal with the problem is to delay the construction of the tree until a new document is presented for classification
- The tree is then built based on the features presented in the document, therefore avoiding the problem

Classification Algorithms

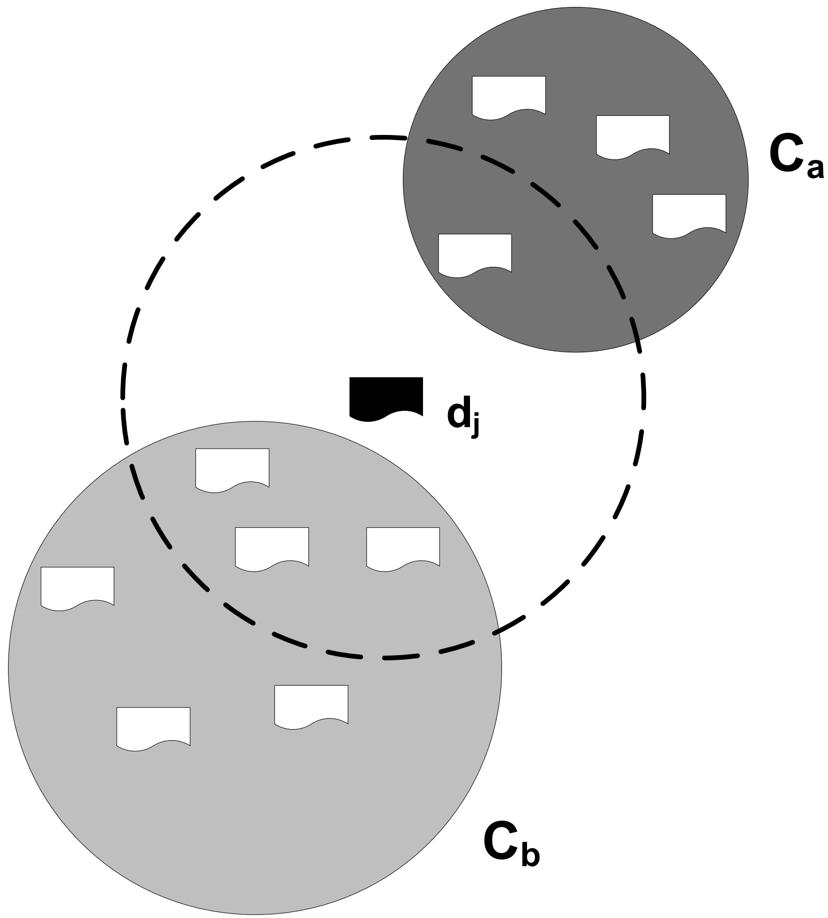
The kNN Classifier

The kNN Classifier

- The kNN (k-nearest neighbor) classifier is a type of on-demand (or lazy) classifier
- **Lazy classifiers** do not build a classification model a priori
- The classification is performed at the moment a new document d_j is given to the classifier
- It is based on the classes of the k nearest neighbors of d_j, computed using a distance function
- This is accomplished as follows:
  - 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 kNN Classifier

- An example of a 4-NN classification process



Classification of Documents

- In the kNN algorithm, with each document-class pair $[d_j, c_p]$ we assign a score S_{d_j,c_p}, given by:

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

  where

  - $N_k(d_j)$ is the set of the k nearest neighbors of d_j in the training set
  - $\mathcal{T}(d_t, c_p)$, the training set function, returns 1 if d_t belongs to class c_p, and 0 otherwise

- The classifier assigns to document d_j the class(es) c_p with the highest score(s)

Classification of Documents

- The cosine measure between the two document vectors is commonly used as the similarity function
- One problem with kNN is performance
- To determine the nearest documents, the classifier has to compute distances between the document to be classified and *all* training documents
- Another issue is how to choose the "best" value for k

Classification Algorithms

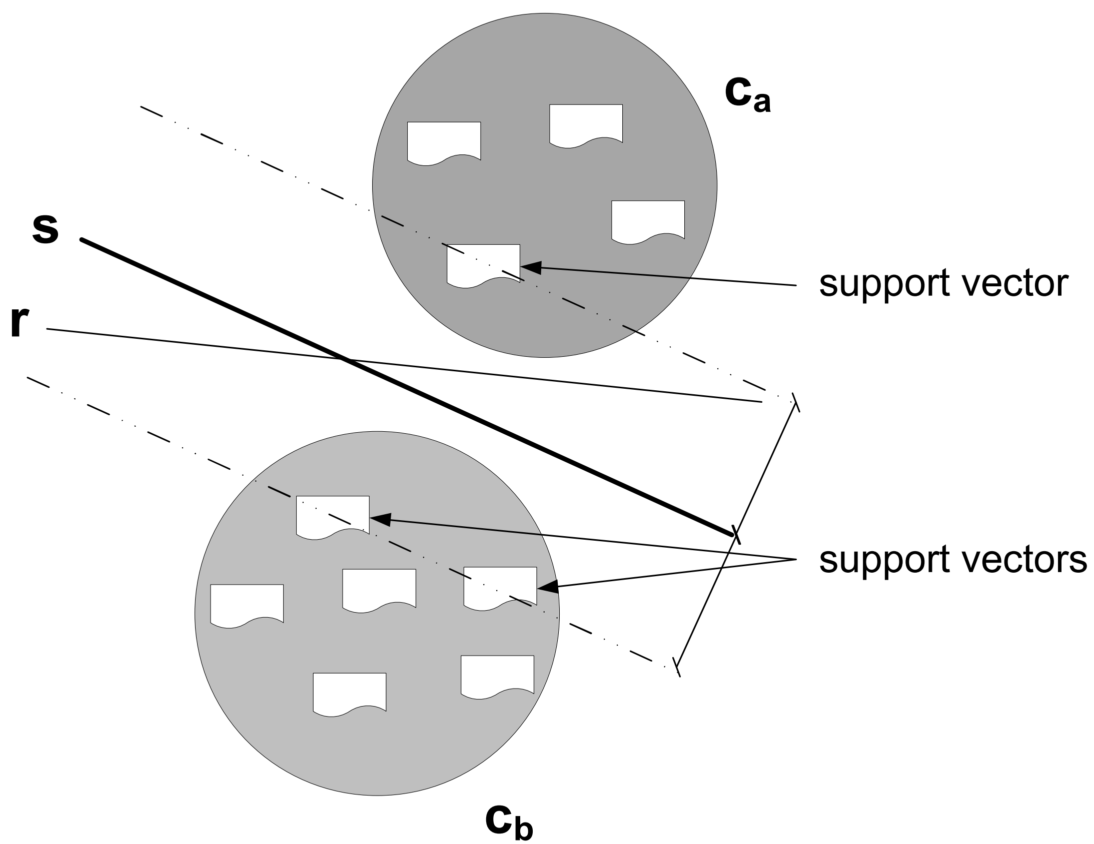
The SVM Classifier

SVM Basic Technique – Intuition

- **Support Vector Machines (SVMs)** constitute a vector space method for binary classification problems
- Consider a set of documents represented in a t-dimensional space
- The idea is to find a **decision surface (hyperplane)** that best separate the elements of two classes
- Then, a new document d_j can be classified by computing its position relative to the hyperplane

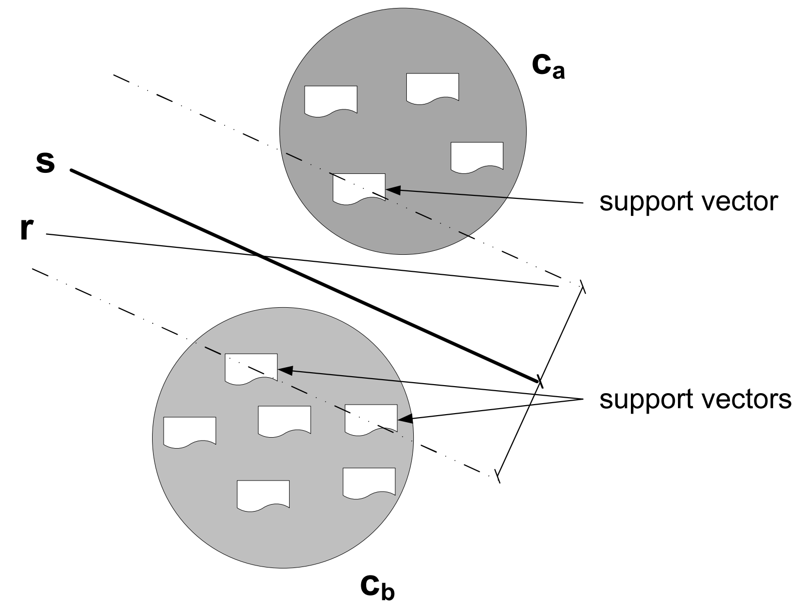
SVM Basic Technique – Intuition

- Consider a simple 2D example whose training data points are linearly separable, as illustrated below



SVM Basic Technique – Intuition

- Line s **maximizes** the distances to the closest documents of each class
- Then, s constitutes the best separating hyperplane, that we refer to the **decision hyperplane**
- In Figure, the parallel dashed lines delimit the region where to look for a solution
- We refer to them as the **delimiting hyperplanes**

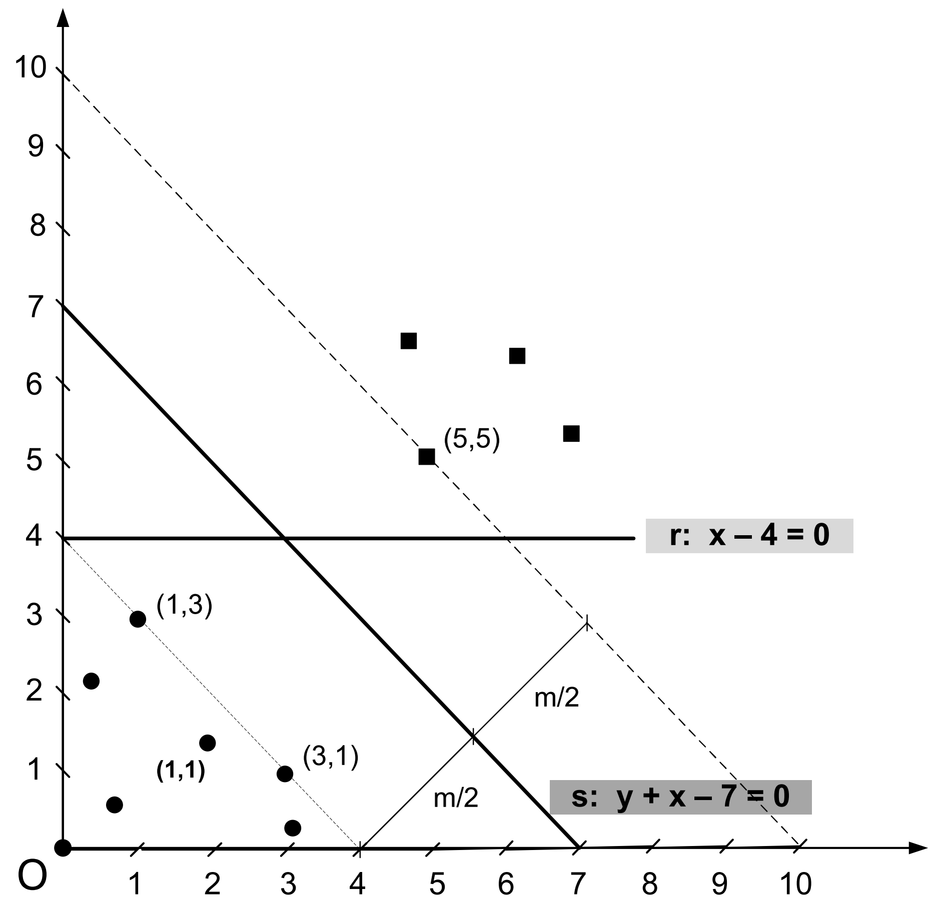


SVM Basic Technique – Intuition

- Lines that cross the delimiting hyperplanes are candidates to be selected as the decision hyperplane
- Lines that are parallel to the this space are the best candidates
- Documents that belong to the delimiting hyperplanes are called **support vectors**

SVM Basic Technique – Intuition

- Figure below illustrates our example in a 2-dimensional system of coordinates

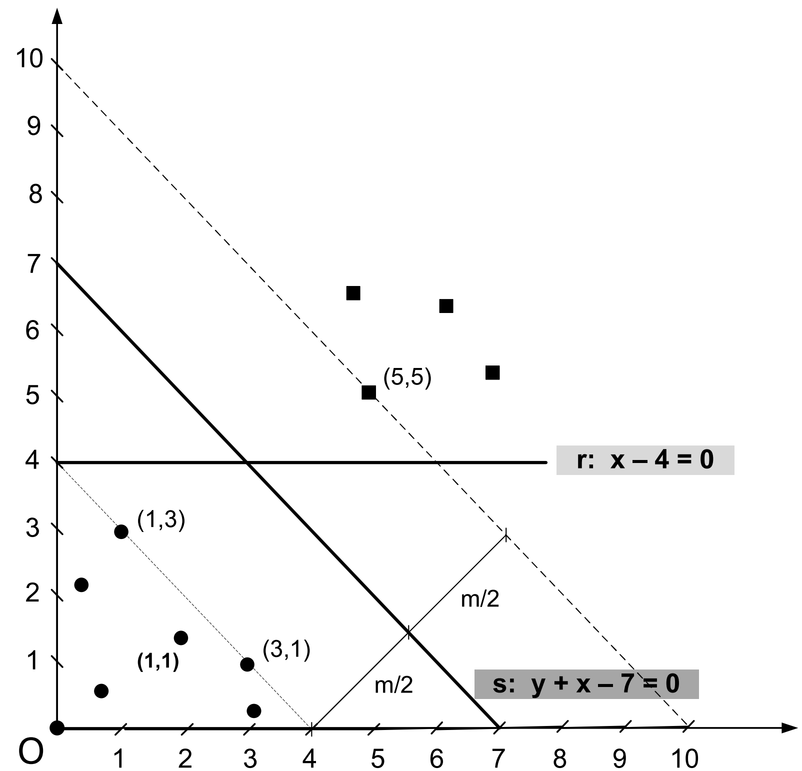


SVM Basic Technique – Intuition

- The SVM optimization problem can be stated as follows
- Let $\mathcal{H}_w$ be a hyperplane that separates all docs in class c_a from all docs in class c_b
- Further, Let
  - m_a be the distance of $\mathcal{H}_w$ to the closest document in class c_a
  - m_b be the distance of $\mathcal{H}_w$ to the closest document in class c_b
  - $m_a + m_b$, called the distance m, is the **margin** of the SVM
- The decision hyperplane $\mathcal{H}_w$ maximizes the margin m

SVM Basic Technique – Intuition

- In our example, the hyperplane $r : x - 4 = 0$ separates the documents in the two sets
- It has distances to the closest documents in either class, points (1,3) and (5,5), equal to 1
  - Thus, its margin m is 2
- The hyperplane $s : y + x - 7 = 0$ provides a margin equal to $3\sqrt{2}$, which is maximum for this case
- Then, the hyperplane s is the decision hyperplane



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

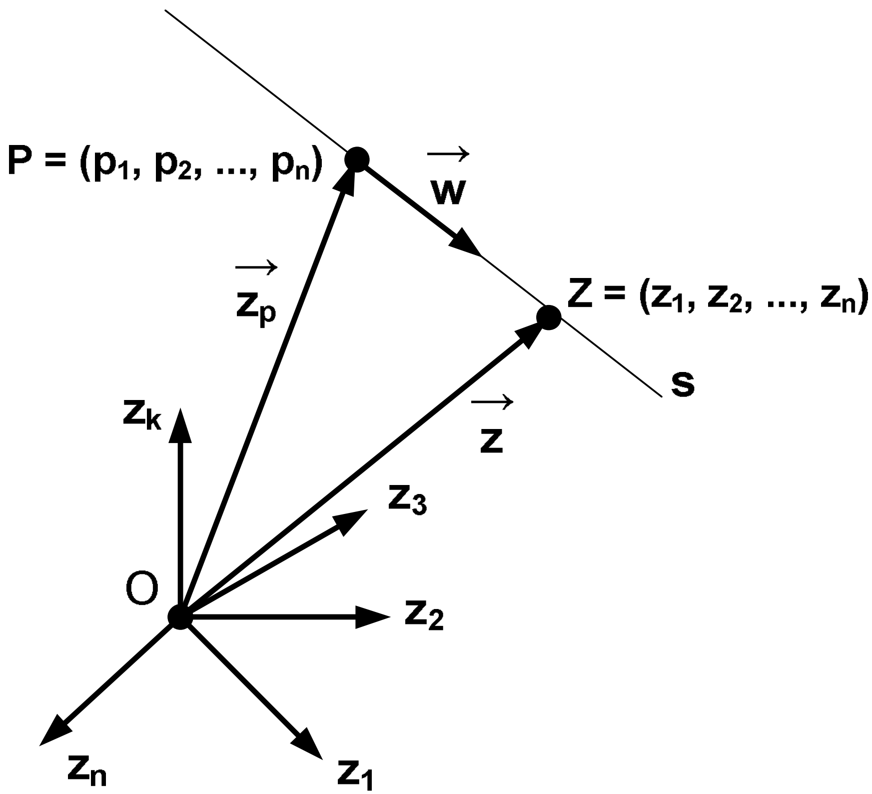
- Figure below illustrates a line s in the direction of a vector $\vec{w}$ that contains a given point P (or $\vec{z}_p$)

- The parametric equation for the line s can be written as

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

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

- As the parameter t varies from $-\infty$ to $+\infty$, the point $\vec{z}$ traverses the line



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

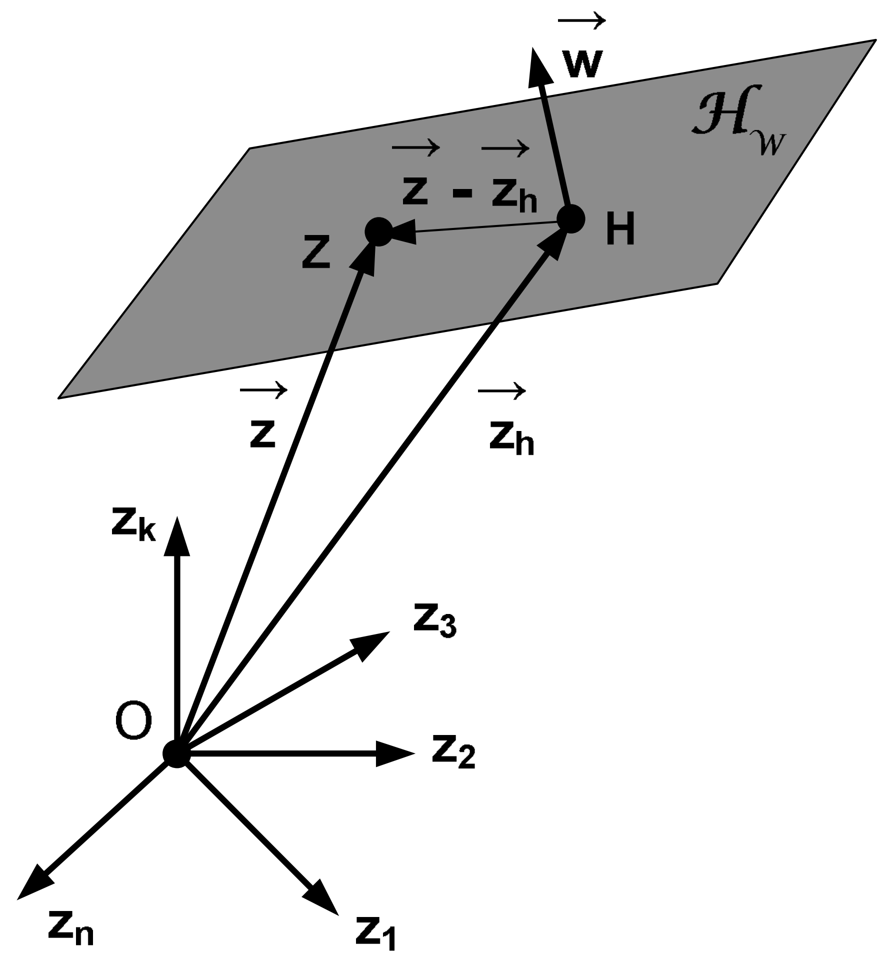
- Figure below illustrates a hyperplane $\mathcal{H}_w$ that contains a point H and is perpendicular to the vector $\vec{w}$

- The normal equation for this hyperplane is

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

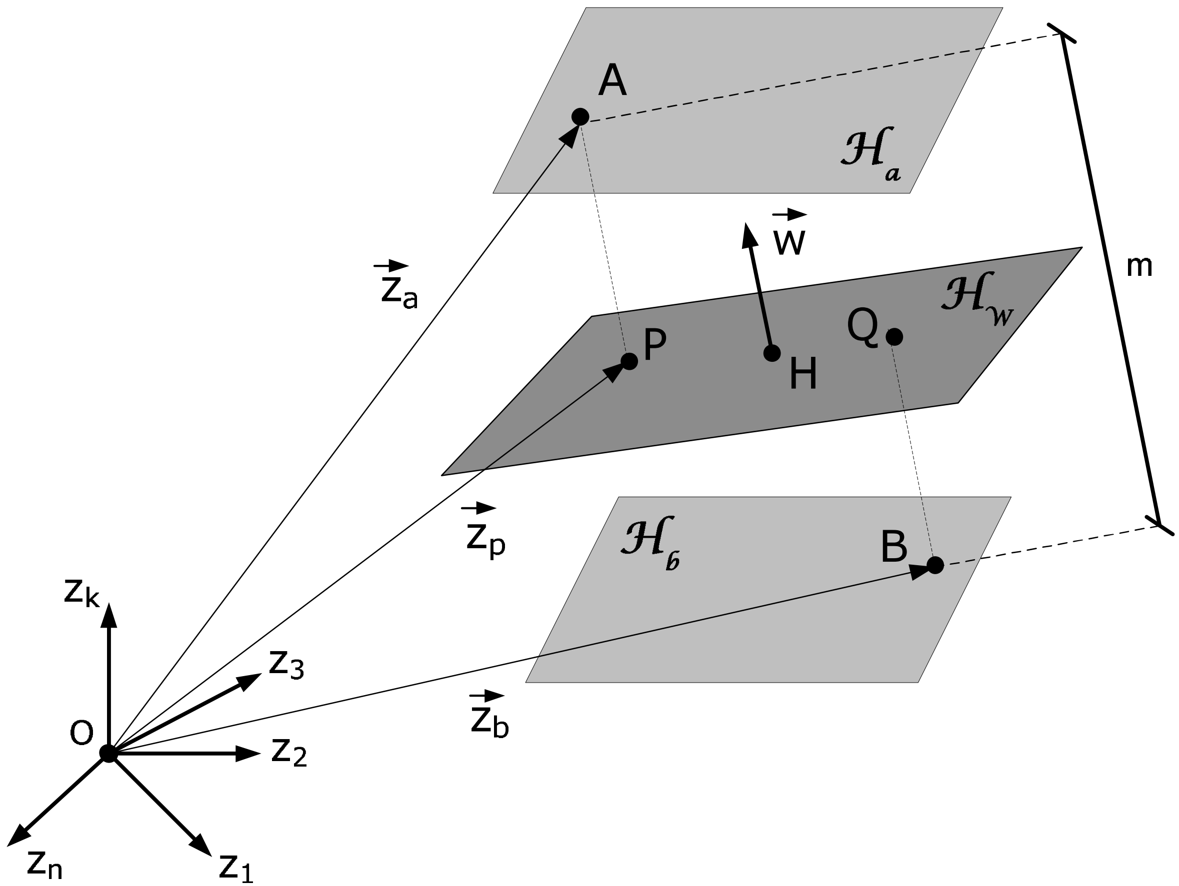
  because the vectors $\vec{z} - \vec{z}_h$ and $\vec{w}$ are perpendicular

- The points $\vec{z}$ that satisfy this equation belong to $\mathcal{H}_w$



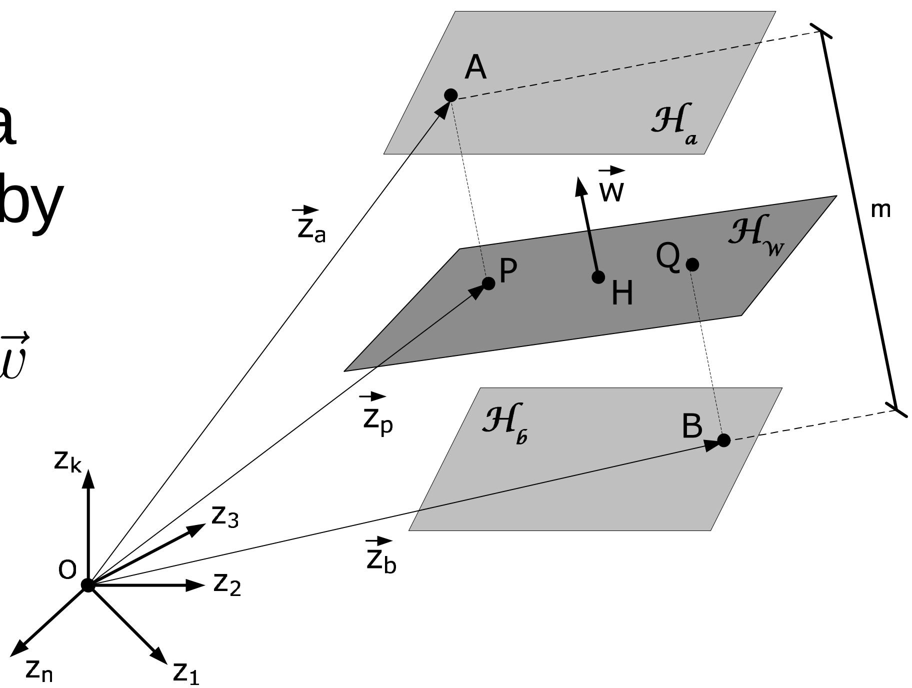
SVM Technique – Formalization

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



SVM Technique – Formalization

- The origin of the coordinate system is point O
- The point A represent a doc from the class c_a and the point B a doc from the class c_b
- These points belong to the delimiting hyperplanes $\mathcal{H}_a$ and $\mathcal{H}_b$
- $\mathcal{H}_w$ is determined by a point H (represented by $\vec{z}_h$) and by a perpendicular vector $\vec{w}$
- Neither $\vec{z}_h$ nor $\vec{w}$ are known a priori



SVM Technique – Formalization

- The normal equation for hyperplane $\mathcal{H}_w$ is then

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

- This equation can be rewritten as

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

  where $\vec{w}$ and $b = -\vec{z}_h\vec{w}$ need to be determined

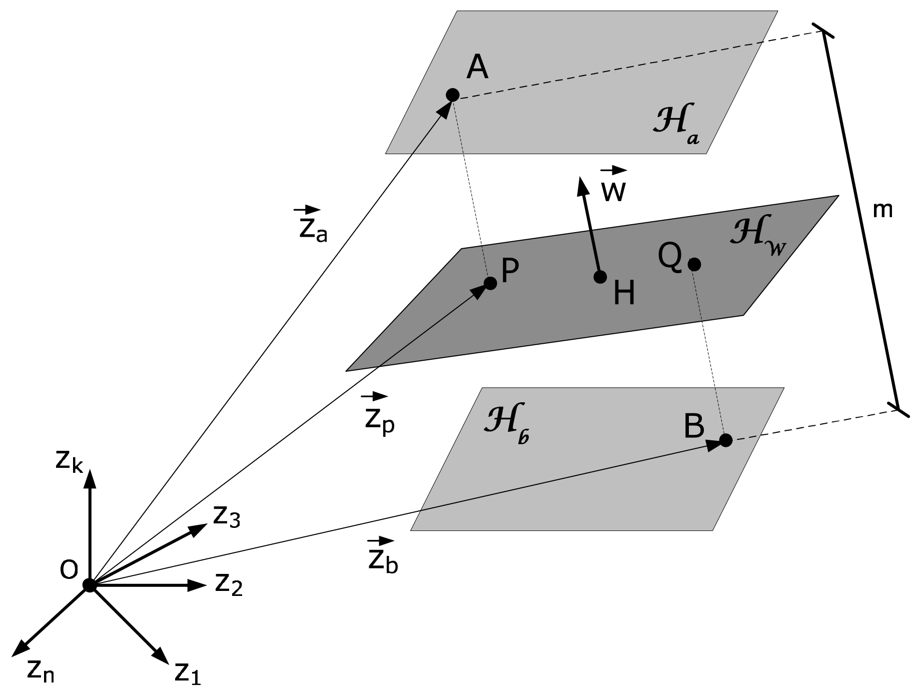
- The second form of the equation makes explicit that b is not a vectorial quantity
- This equation will be used interchangeably with the normal equation to refer to the hyperplane

SVM Technique – Formalization

- Let P be the projection of point A on $\mathcal{H}_w$
- The line determined by points A and P is necessarily in the direction of vector $\vec{w}$
- Then, its parametric equation is

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

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



SVM Technique – Formalization

- For the point P specifically, we have $\vec{z}_p = t_p\vec{w} + \vec{z}_a$, where t_p is the value of t for point P
- Since $P \in \mathcal{H}_w$, we obtain

$$(t_p\vec{w} + \vec{z}_a)\vec{w} + b = 0$$

- Solving for t_p,

$$t_p = -\frac{\vec{z}_a\vec{w} + b}{|\vec{w}|^2}$$

- By substituting t_p back into Equation $\vec{z}_p = t_p\vec{w} + \vec{z}_a$,

$$\vec{z}_a - \vec{z}_p = \frac{\vec{z}_a\vec{w} + b}{|\vec{w}|} \times \frac{\vec{w}}{|\vec{w}|}$$

SVM Technique – Formalization

- Since $\vec{w}/|\vec{w}|$ is a unit vector in the direction of $\vec{w}$, we have

$$\overline{AP} = |\vec{z}_a - \vec{z}_p| = \frac{\vec{z}_a \vec{w} + b}{|\vec{w}|}$$

which is the distance of point A to hyperplane $\mathcal{H}_w$

- This distance is normalized by $|\vec{w}|$ and, as a result, does not depend on the size of $\vec{w}$

SVM Technique – Formalization

- Consider the hyperplane $\mathcal{H}_b$, which is opposite to $\mathcal{H}_a$ relatively to $\mathcal{H}_w$
- We can write analogous equations for $\mathcal{H}_b$, although these equations should all be relative to $-\vec{w}$:

$$\begin{aligned} \mathcal{H}_w : & \quad (\vec{z}_h - \vec{z})(-\vec{w}) = 0 \\ line(\overline{BQ}) : & \quad \vec{z} = -t\vec{w} + \vec{z}_b \end{aligned}$$

where Q is the projection of point B on $\mathcal{H}_w$

SVM Technique – Formalization

- These two equations lead to

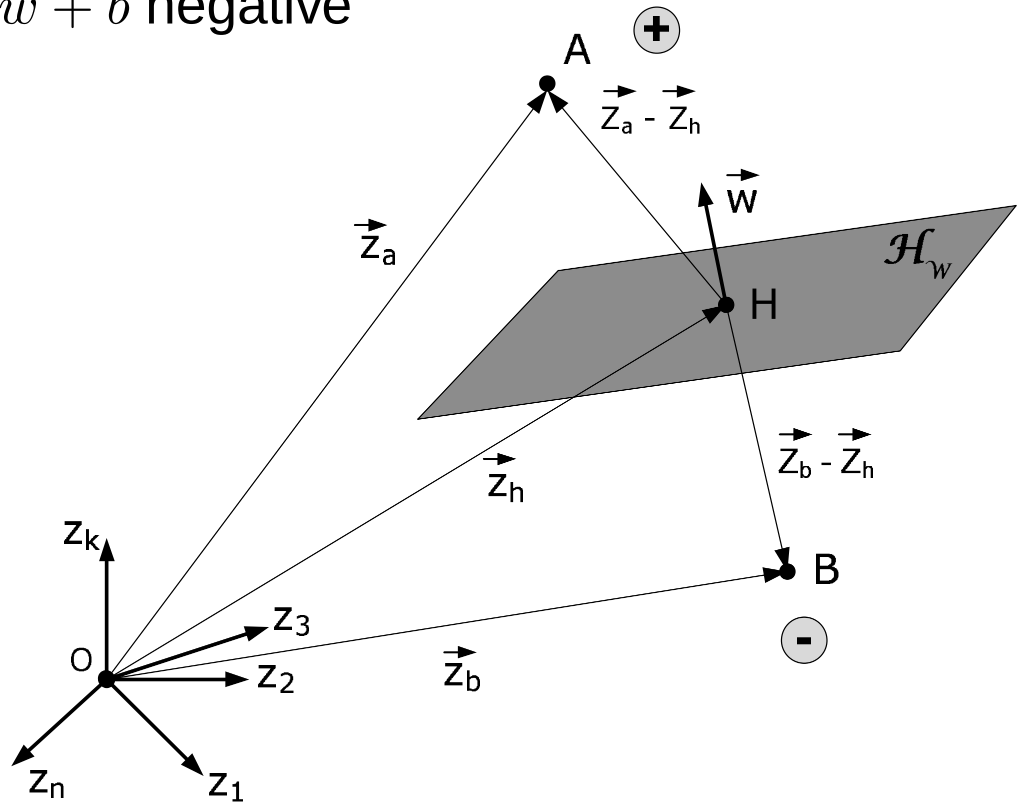
$$\overline{BQ} = |\vec{z}_b - \vec{z}_q| = -\frac{\vec{z}_b\vec{w} + b}{|\vec{w}|}$$

  which is analogous to the expression for the segment $\overline{AP}$

- Notice that the expression $\vec{z}_b\vec{w} + b$ is necessarily negative
- This is because $\vec{z}_b$ is in the opposite region of $\vec{z}_a$, relatively to $\mathcal{H}_w$

SVM Technique – Formalization

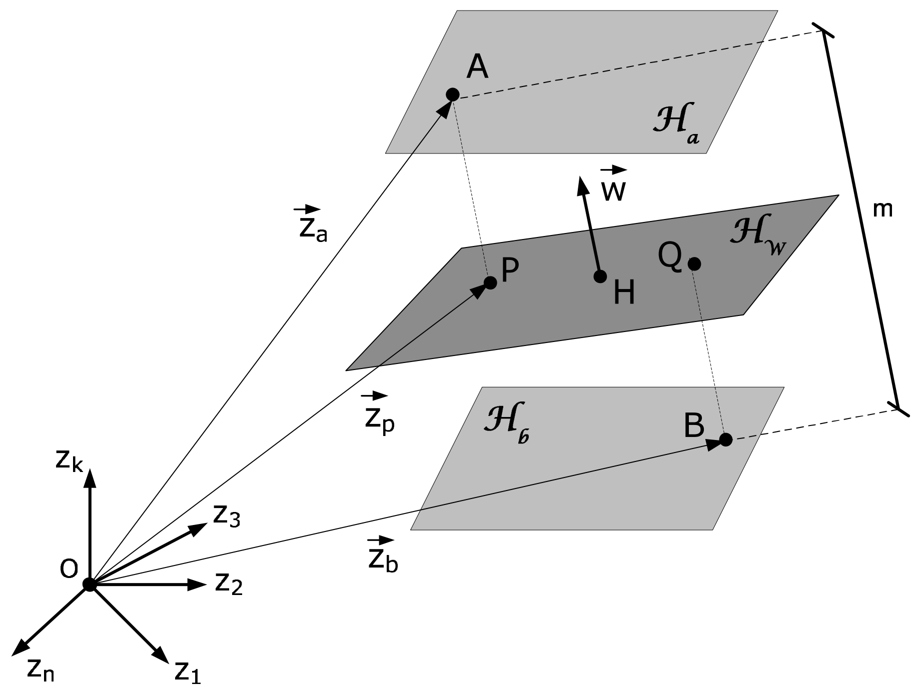
- Signs in the SVM system:
  - The region above $\mathcal{H}_w$ is composed of points $\vec{z}$ that make $\vec{z}\vec{w} + b$ positive
  - The region below the hyperplane is composed of points that make $\vec{z}\vec{w} + b$ negative



SVM Technique – Formalization

- The margin m of the SVM is given by $m = \overline{AP} + \overline{BQ}$ and is independent of the size of $\vec{w}$
- That is, there are vectors $\vec{w}$ of varying sizes that maximize m
- To impose restrictions on $|\vec{w}|$, we can set

$$\begin{aligned} \vec{z}_a \vec{w} + b &= 1 \\ \vec{z}_b \vec{w} + b &= -1 \end{aligned}$$



SVM Technique – Formalization

- Notice that this restricts the solution to hyperplanes that split the margin m in the middle
- Under these conditions, the expression for the margin m becomes

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

- For a point C that is located farther away from the $\mathcal{H}_w$ than a support vector, it must be either

$$\vec{z}_c\vec{w} + b > 1 \; or \; \vec{z}_c\vec{w} + b < -1$$

depending on whether the point is in the positive or in the negative region

SVM Technique – Formalization

- Let $\mathcal{T} = \{\ldots, [c_j, \vec{z}_j], [c_{j+1}, \vec{z}_{j+1}], \ldots\}$, be the training set
- c_j is the class (either c_a or c_b) associated with point $\vec{z}_j$, i.e., with a document d_j
- Then,

SVM Optimization Problem:

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

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

- The vectors that make the equation equal to either 1 or -1 are the support vectors

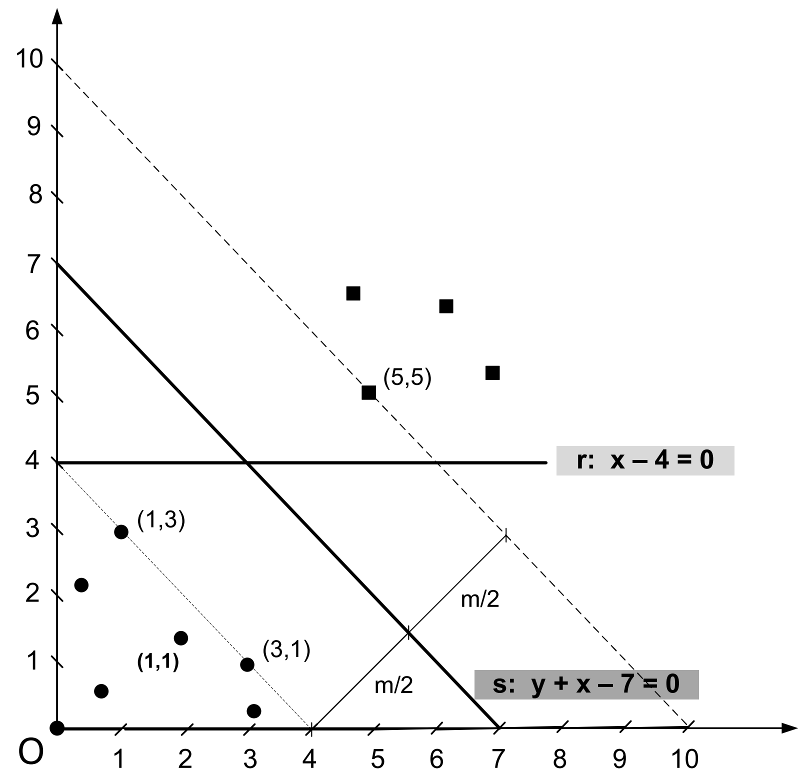
SVM Technique – Formalization

- Let us consider again the simple example in Figure below

- For that case, the optimization problem can be especified as:

  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 $|\vec{w}| = \sqrt{x^2 + y^2}$
- The parameter m stands for the distance between the delimiting hyperplanes and is equal to $3\sqrt{2}$
- Thus,

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

which yields $b = -16/9$ or $b = -21/9$

SVM Technique – Formalization

- The value that maximizes $2/|\vec{w}|$ is $b = -21/9$, which yields $x = 1/3, y = 1/3$
- Thus, the equation of the decision hyperplane is given by

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

  or

  $$y + x - 7 = 0$$

Classification of Documents

- The classification of a doc d_j, represented as a vector $\vec{z}_j$, is achieved by applying the decision function

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

- If the sign is positive then the document d_j belongs to class c_a; otherwise, it belongs to class c_b
- The SVM classifier might also enforce the margin to reduce classification errors
- In this case a new document d_j
  - is classified in class c_a only if $\vec{w}\vec{z}_j + b > 1$, and
  - is classified in class c_b only if $\vec{w}\vec{z}_j + b < -1$

Classification of Documents

- SVMs are directly applicable when documents are represented by weighted multidimensional vectors
- Moreover, we can apply a feature selection to select the set of terms that are relevant to the classification task
- Finally, SVMs can only take binary decisions: a document belongs or not to a given class
- With multiple classes, a different classifier needs to be learned for each class

Feature Selection or Dimensionality Reduction

Feature Selection

- Metrics for feature selection are a key component of the classification process
- To produce feature vectors, we need to select a subset of the features (terms) to represent the documents
- Feature selection contributes to
  - Reduce the dimensionality of the documents representation
  - Reduce **overfitting**

Term-Class Incidence Table

- Before proceeding, it is useful to define a term-class incidence table
- Let $\mathcal{D}_t$ be the subset composed of all training documents
- Further, let
  - N_t be the number of documents in $\mathcal{D}_t$
  - t_i be the number of documents from $\mathcal{D}_t$ that contain the term k_i
  - $\mathcal{C} = \{c_1, c_2, \ldots, c_L\}$ be the set of all L classes
- Assume that a training set function $\mathcal{T} : \mathcal{D}_t \times \mathcal{C} \rightarrow [0, 1]$ has been specified

Term-Class Incidence Table

- The term-class incidence table is given by

	docs in c_p	docs not in c_p	
docs that contain k_i	$c_{i,p}$	$t_i - c_{i,p}$	t_i
docs that do not contain k_i	$P - c_{i,p}$	$N_t - t_i - (P - c_{i,p})$	$N_t - t_i$
all docs	P	$N_t - P$	N_t

- The number of documents that contain term k_i and are classified in class c_p is given by $c_{i,p}$
- The number of documents that contain term k_i but are not in class c_p is given by $t_i - c_{i,p}$
- P is the total number of training documents in class c_p

Term-Class Incidence Table

- $P - c_{i,p}$ is the number of documents from c_p that do not contain term k_i
- The remaining quantities are calculated analogously
- The term-class incidence table contains the sizes of various subsets that constitute the doc training set

Feature Selection or Dimensionality Reduction

Term Document Frequency

Term Document Frequency

- Here, the idea is to select the terms whose document frequency exceeds a frequency threshold FT
- That is, all terms k_i for which $t_i \geq FT$ are retained, all other terms are discarded
- Even if simple, selection of higher document frequency terms allows considerably reducing the dimensionality of the space with no loss in effectiveness

Feature Selection or Dimensionality Reduction

Tf-Idf Weights

Tf-Idf Weights

- A term selection procedure that retains the terms of higher tf-idf weights in each document d_j
- Some experiments suggest that this feature selection allows reducing the dimensionality of the space by a factor of 10 with no loss in effectiveness
- Further, a reduction of dimensionality by a factor of 100 leads to just a small loss in effectiveness

Feature Selection or Dimensionality Reduction

Mutual Information

Mutual Information

- **Mutual information** is a measure of the relative entropy between the distributions of two random variables
- If those variables are independent, we say that their mutual information is zero
  - In this case, knowledge of one of the variables does not allow infering anything about the other variable

Mutual Information

- Mutual information is expressed as

$$I(k_i, c_p) = \log \frac{P(k_i, c_p)}{P(k_i)P(c_p)}$$

where all probabilities are interpreted relatively to a doc d_j randomly selected from the set of training docs:

  - $P(k_i, c_p)$: probability that $k_i \in d_j$ and $d_j \in c_p$
  - $P(k_i)$: probability that $k_i \in d_j$
  - $P(c_p)$: probability that $d_j \in c_p$

Mutual Information

- Using the term-class incidence table, we can specify these probabilities as follows:

$$P(k_i, c_p) = \frac{c_{i,p}}{N_t} \qquad P(k_i) = \frac{t_i}{N_t} \qquad P(c_p) = \frac{P}{N_t}$$

- Our information metric relating the term k_i to the class c_p then becomes

$$I(k_i, c_p) = \log \frac{\frac{c_{i,p}}{N_t}}{\frac{t_i}{N_t} \times \frac{P}{N_t}} = \log \frac{c_{i,p} \times N_t}{t_i \times P}$$

Mutual Information

- To apply feature selection to term k_i, we compute its average mutual information across all classes as follows

$$
\begin{aligned}
MI_{avg}(k_i) &= \sum_{p=1}^{L} P(c_p)\, I(k_i, c_p) \\
&= \sum_{p=1}^{L} \frac{P}{N_t} \log \frac{c_{i,p} \times N_t}{t_i \times P}
\end{aligned}
$$

- An alternative is to use the maximum term information over all classes, as follows:

$$
\begin{aligned}
I_{max}(k_i) &= max_{p=1}^{L}\, I(k_i, c_p) \\
&= max_{p=1}^{L}\, \log \frac{c_{i,p} \times N_t}{t_i \times P}
\end{aligned}
$$

Mutual Information

- The feature selection process, in this case, works as follows:
  - terms with values of $MI_{avg}(k_i)$, or of $I_{max}(k_i)$, above a given threshold are retained;
  - the remaining terms are discarded.

Feature Selection or Dimensionality Reduction

Information Gain

Information Gain

- Mutual information uses the probabilities associated with the occurrence of terms in documents
- **Information Gain** is a complementary metric, that also considers the probabilities associated with the absence of terms in documents
- It balances the effects of term/document occurrences with the effects of term/document absences

Information Gain

- The information gain $IG(k_i, \mathcal{C})$ of term k_i over the set $\mathcal{C}$ of all classes is defined as follows

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

  where
  - $H(\mathcal{C})$ is the entropy of the set of classes $\mathcal{C}$
  - $H(\mathcal{C}|k_i)$ and $H(\mathcal{C}|\neg k_i)$ are the conditional entropies of $\mathcal{C}$ in the presence and in the absence of term k_i

- In information theory terms, $IG(k_i, \mathcal{C})$ is a measure of the amount of knowledge gained about $\mathcal{C}$ due to the fact that k_i is known

Information Gain

- Using the term-class incidence table defined previously, and recalling the expression for entropy, we can write,

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

Information Gain

- Applying Bayes rule, this can rewritten as

$$IG(k_i, \mathcal{C}) = -\sum_{p=1}^{L} \left[P(c_p) \log P(c_p) - P(k_i, c_p) \log \frac{P(k_i, c_p)}{P(k_i)} - P(\neg k_i, c_p) \log \frac{P(\neg k_i, c_p)}{P(\neg k_i)} \right]$$

where all probabilities are interpreted relatively to a doc d_j randomly selected from the training set

Information Gain

- That is,
  - $P(\neg k_i, c_p)$: probability that $k_i \notin d_j$ and $d_j \in c_p$
  - $P(\neg k_i)$: probability that $k_i \notin d_j$

- Using the term-class incidence table defined previously, we can specify these probabilities as follows:

$$
\begin{aligned}
P(\neg k_i, c_p) &= \frac{P - c_{i,p}}{N_t} \\
P(\neg k_i) &= \frac{N_t - t_i}{N_t}
\end{aligned}
$$

Information Gain

- Based on the previous Equation, we can write

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

- The feature selection process works as follows:
  - terms with values of $IG(k_i, \mathcal{C})$ above a given threshold are retained
  - the remaining terms are discarded

Evaluation Metrics

Evaluation Metrics

- Evaluation is a very important step in the development of any text classification method
- Without proper evaluation, there is no way to determine how good is a newly proposed text classifier
- That is, evaluation is a key step to *validate* a newly proposed classification method
- Here, we describe some of the most used metrics to assess the quality of single label text classifiers
- We start defining a contingency table, that will be to describe the most used evaluation metrics

Contingency Table

- Let
  - $\mathcal{D}$ be a collection of documents
  - $\mathcal{D}_t$ be the subset composed of training documents
  - N_t be the number of documents in $\mathcal{D}_t$
  - $\mathcal{C} = \{c_1, c_2, \ldots, c_L\}$ be the set of all L classes
- Assume that have been specified
  - A training set function $\mathcal{T} : \mathcal{D}_t \times \mathcal{C} \rightarrow [0, 1]$
  - A text classifier function $\mathcal{F} : \mathcal{D} \times \mathcal{C} \rightarrow [0, 1]$
- Further, let
  - P_t be the number of docs from the training set $\mathcal{D}_t$ in class c_p
  - P_f be the number of docs from the training set assigned to class c_p by the classifier

Contingency Table

- Consider the application of the classifier to all documents in the training set
- The contingency table is given by

	$\mathcal{T}(d_j, c_p) = 1$	$\mathcal{T}(d_j, c_p) = 0$	
$\mathcal{F}(d_j, c_p) = 1$	$c_{f,t}$	$P_f - c_{f,t}$	P_f
$\mathcal{F}(d_j, c_p) = 0$	$P_t - c_{f,t}$	$N_t - P_f - P_t + c_{f,t}$	$N_t - P_f$
all docs	P_t	$N_t - P_t$	N_t

  - where $c_{f,t}$ is the number of docs that both the training and classifier functions assigned to class c_p
  - The number of training docs in class c_p that were miss-classified by the classifier is given by $P_t - c_{f,t}$
  - The remaining quantities are calculated analogously

Evaluation Metrics

Contingency Table

Accuracy and Error

- The accuracy and error metrics are defined relative to a given class c_p, as follows

$$
\begin{aligned}
Acc(c_p) &= \frac{c_{f,t} + (N_t - P_f - P_t + c_{f,t})}{N_t} \\
Err(c_p) &= \frac{(P_f - c_{f,t}) + (P_t - c_{f,t})}{N_t}
\end{aligned}
$$

- Notice that necessarily

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

Accuracy and Error

- Accuracy and error, despite their common use, have disadvantages
- Consider, for instance, a binary classification problem with only two categories c_p and c_r
- Assume that out of 1,000 documents there are only 20 in class c_p
- Then, a classifier that assumes that all docs are not in class c_p has an accuracy of 98% and an error of just 2%
- These values suggest that we have a very good classifier, but this is not the case

Accuracy and Error

- Consider now a second classifier that correctly predicts 50% of the documents in c_p, as 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

- 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.

Evaluation Metrics

Precision and Recall

Precision and Recall

- Precision and recall in text classification are variants of the precision and recall metrics in IR
- Precision P and recall R are computed relative to a given class c_p, as follows

$$P(c_p) = \frac{c_{f,t}}{P_f} \qquad R(c_p) = \frac{c_{f,t}}{P_t}$$

- 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 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

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

- That is, the classifier has precision of 100% and recall of 50% for class c_p

Precision and Recall

- Precision and recall defined in this way are computed for every category in set $\mathcal{C}$
- This yields a great number of values, making the tasks of comparing and evaluating algorithms more difficult
- It is often convenient to combine precision and recall into a single quality measure
- One of the most commonly used such measures is the *F-measure*, which we discuss now

Evaluation Metrics

F-measure

F-measure

- The F-measure combines precision and recall values
- It allows for the assignment of different weights to each of these measures
- It is defined as follows:

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

  where α defines the 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, and so on

F-measure

- The most used form of the *F-measure* is obtained by assigning equal weights to precision and recall
- This is accomplished by making $\alpha = 1$, and is is called the 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

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

Evaluation Metrics

F-measure

F_1 Macro and Micro Averages

- It is also common to derive a unique F_1 value for a classifier
- This is accomplished computing the average of F_1 values across all individual categories
- There are two average functions that are considered in the literature:
  - Micro-average $F1$, or $micF_1$
  - Macro-average F_1, or $macF_1$

F_1 Macro and Micro Averages

- Macro-average F_1 is computed as

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

- Thus, macro-average F_1 simply averages F_1 across all categories

F_1 Macro and Micro Averages

- To compute micro-average F_1 we consider recall and precision figures over all categories, as follows

$$
\begin{aligned}
P &= \frac{\sum_{c_p \in \mathcal{C}} c_{f,t}}{\sum_{c_p \in \mathcal{C}} P_f} \\
R &= \frac{\sum_{c_p \in \mathcal{C}} c_{f,t}}{\sum_{c_p \in \mathcal{C}} P_t}
\end{aligned}
$$

- Then, micro-average F_1 can be computed by

$$
micF_1 = \frac{2PR}{P+R}
$$

F_1 Macro and Micro Averages

- In micro-average F_1, every single document is given the same importance
- In macro-average F_1, every single category is given the same importance
- Macro-average F_1 captures better the ability of the classifier to perform well for many classes
- This becomes important whenever the distribution of classes is very skewed
- In this case, both average metrics should be considered

Evaluation Metrics

Cross-Validation

Cross-Validation

- Cross-validation has become a standard method to guarantee the statistical validation of the results
- It consists of building k different classifiers: $\Psi_1, \Psi_2, \ldots,$ Ψ_k
- The classifiers are built by dividing the training set $\mathcal{D}_t$ into k disjoint sets or folds of sizes

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

Cross-Validation

- The classifier Ψ_i uses the training set minus the ith fold for tunning and run its test on the ith fold
- Each classifier is evaluated independently using precision-recall or F_1 figures
- The cross-validation is done by computing the average of the k measures
- The most commonly adopted value of k is 10, in which case the method is called **ten-fold cross-validation**

Evaluation Metrics

Standard Collections

Standard Collections

- Several standard benchmark collections are available for experimenting classification techniques
- In the immediately following, we present some of the most used benchmark collections

Standard Collections

- **Reuters-21578**
  - It is the most widely used collection in the classification experiments
  - It is constituted of news articles from Reuters for the 1987 year
  - The collection is classified under several categories related to economics (e.g., acquisitions, earnings, etc)
  - It 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 Corpus Volume 1 (RCV1) and Volume 2 (RCV2)**
  - The RCV1 is another collection of news stories recently released by Reuters
  - It contains approximately 800,00 documents organized in 103 topical categories
  - It is expected to substitute the previous Reuters-21578 collection in text classification experiments
  - RCV2 is a modified version of the original released collection, in which some corrections were made

Standard Collections

OHSUMED

- OHSUMED is another popular collection used for testing text classification algorithms
- It is a subset of the Medline digital library, containing medical documents (title or title + abstract)
- There are 23 classes corresponding to MesH diseases used to index the documents

Standard Collections

- **20 NewsGroups**
  - A third largely used collection is 20 Newsgroups
  - This is a collection of approximately 20,000 messages posted to Usenet newsgroups, partitioned (nearly) evenly across 20 different newsgroups
  - The categories are the newsgroups themselves

Standard Collections

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