CLASS DOCUMENT FREQUENCY AS A LEARNED FEATURE FOR TEXT CATEGORIZATION

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE
UNIVERSITY OF HAWAI'I IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

MAY 2008

By
Anand Sharma

Thesis Committee:

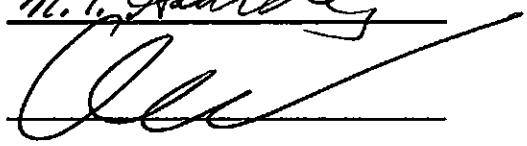
Anthony Kuh, Chairperson
N. T. Gaarder
Yingfei Dong

4288

We certify that we have read this thesis and that, in our opinion, it is satisfactory in scope and quality as a thesis for the degree of Master of Science in Electrical Engineering.

THESIS COMMITTEE

Chairperson



Acknowledgements

I am grateful to my advisor Prof. Anthony Kuh, University of Hawai'i at Manoa, for his guidance throughout my research work and writing of this thesis. I am also thankful to my committee members Prof. N. T. Gaarder and Prof. Yingfei Dong.

Contents

List of Tables

List of Figures

ABSTRACT

With the increase in online information, which are mostly in text document form, there is a need to organize them so that management and retrieval by search engine become easier. Manual organization of these documents is very difficult and prone to error. Machine learning algorithms can be used for classification and then organization because they are quick, relatively more accurate and less costly. However, documents need to have feature representations that are suitable for training machine learning algorithms for document classification.

Machine learning algorithms for document classification use different types of word weightings as features for representation of documents. In our findings we find the class document frequency, dfc, of a word is the most important feature in document classification. Machine learning algorithms trained with dfc of words show similar performance in terms of correct classification of test documents when compared to more complicated features. The importance of dfc is further verified when simple algorithm $Algd_1$ developed solely on the basis of dfc shows performance that compares closely with that of $Algt_1$ and other more complex machine learning algorithms. The importance of high dfc is verified when $Algd_2$ performs comparably with $Algt_2$ and other complex algorithms. This also implies that term frequency does not contribute much to the classification of documents compared to class document frequency. We also find improved performance when the link information of documents in a class is used along with the word attributes of the document. The contribution of term frequency of link and class document frequency of link are are similar in their classification performance. This shows the importance of class document frequency as the learned feature that learning algorithms use for effective text categorization. We compared the algorithms for showing the importance of dfc on the Reuters-21578 text categorization test classification set, Cora data set and Citeseer data set.

Chapter 1

Introduction

With the tremendous increase of online information [17][18], most of which are in text document form, management and retrieval of these documents by internet search engines become unimaginable without good document classification algorithms. Document classification can be used for spam filtering [27] because most of the spams have text content. Document classification along with link analysis can be used to extract social network [28] that can help people with similar interest connect and communicate.

Document classification is the process of assigning text documents to one or more predefined categories based on their contents. The classification has to be learned from the available past data set based upon which the decision about the class of new data is made. Classification can be from a supervised or unsupervised learning algorithm depending upon whether or not class labels are available.

Classification from unsupervised learning algorithms occurs without proper class label. Examples of unsupervised learning algorithms include K-means clustering [9] which finds clusters into which the data set is partitioned. The clustering is based on the similarity of data with the cluster centers. Another algorithm includes Principle component analysis [9] which works by decomposing the given data set into a smaller set of uncorrelated data. Independent component analysis [9] also decomposes the data into smaller set of data but under the constraint that the reduced set of data are independent of one another. Gaussian mixture [9] estimates the number of effective components required to be combined linearly with different weights to reproduce the data. Then there are graphical methods [9] that try

to estimate the relationship among the data samples in terms of conditional dependencies among them.

In supervised machine learning the class label of different data samples are already available and the label of a new test sample is determined based on the already labeled data samples. There are different supervised machine learning algorithms: rule based machine learning, statistical machine learning and ensemble based machine learning. Decision tree learning [20] is rule based learning. Naive Bayes [5] and support vector machines [6] are some examples of statistical machine learning. Bagging and boosting [21][22] are examples of ensemble learning. Ensemble learning algorithms are meta-learning algorithms that are used to strengthen other weak learning algorithms by combining different algorithms together.

Supervised machine learning algorithms learn patterns from features in documents. Different types of features can be used to represent a document as a vector of features. The feature could mean the presence of a single word, a linguistic phrase or a complicated syntax template. According to the study conducted by Lewis [4] on the effects of syntactic phrases in text categorization it was found that a Naive Bayesian classifier trained only with single word as feature did better than the one trained with noun phrases. Using syntactic phrases as features did not contribute to the improvement on rule-based classifiers [3]. It also did not show much improvement on Naive Bayesian and SVM classifiers [11]. Information retrieval research suggests that words work well as representation units and that their ordering in a document can be ignored for classification tasks [11][12]. A document therefore can be considered as a bag of words and defined as a vector of features with each component corresponding to a word in the document. Since the features themselves cannot be directly used by the machine learning algorithms it is necessary to quantify or assign weights to the features. It is important that the quantified value corresponding to the feature be able to capture as much information as possible regarding the feature distribution in the document, in the document class and in all the classes. There are different approaches to assigning weights to these features, [1][2][3][4][15][19]. Papers [2] and [26] assess performance of machine learning algorithms on different types of features and come to different conclusions as to the best feature suitable for document classification. In this paper, a binary 1 for presence and binary 0 for absence of word is the feature value

assigned to represent a document vector so that the sum of these features across all the documents in a class constitute dfc. Class document frequency, dfc, is used as the feature for performance assessment of different machine learning algorithms.

The machine learning algorithms used for text categorization are supervised statistical machine learning algorithms. The performance of Bayesian learning algorithm and support vector machines is assessed on class document frequency. Based on this performance an argument is made that dfc is the feature that the machine learning algorithms inherently learn that helps in effective classification of the text documents into different classes.

We developed $Algd_1$ based directly on dfc and its performance is found to be comparable to that of $Algt_1$ based on class term frequency and more complex machine learning algorithms. We also developed another algorithm $Algd_2$ in which words with high class document frequency are made to have more contribution than those with low class document frequency towards classification of test documents. It is found that this algorithm also performs well compared to $Algt_2$ based on class term frequency with high value and other algorithms. This confirms the importance of high class document frequency. Based on this, it can be inferred that dfc is an important discriminator for document classification whether using Bayesian learning algorithm or support vector machines. Both of these methods extract information about dfc during training which helps in the effective classification of documents. It is also shown in this thesis that link information of text documents, such as conference papers with bibliographic citations, when used along with the word features of documents enhance the classification performance of algorithms.

In this thesis we developed mathematical representation for document, class and link of document. We found modification of Naive Bayes and Probabilistic Term Frequency Inverse Document Frequency algorithms by using class document frequency as feature for representation of documents in a class in place of term frequency as feature for representation of documents. We showed the importance of class document frequency with KNN and SVM algorithms also. We developed algorithms $Algd_1$, $Algd_2$, $Algt_1$ and $Algt_2$ to show the importance of class document frequency. We showed that class document frequency of words along with class document frequency of link of document further enhance the classification performance of SVM. We conducted several experiments using algorithms

mentioned above on Reuters 21578, Cora and Citeseer data sets respectively and the results of these experiments showed the importance of class document frequency.

This thesis is organized as follows. Chapter 2 discusses the feature representation of documents and deals with mathematical representation of documents, of class and of links of documents. Chapter 3 discusses different machine learning algorithms and how to use the class document frequency to train those machine learning algorithms. Chapter 4 discusses the data sets used for simulations, different performance measures for classification, and the results obtained. Chapter 5 summarizes the thesis and discusses further research directions.

Chapter 2

Feature Representation

There have been different researches on text document classification seeking to find the features [1][2][3][4] that would act as effective discriminators contributing to correct classification of documents to their respective classes. Although different features such as phrases, complicated syntax template and individual words have been considered as features for representation of documents, it has been found that words act as reliable descriptors of documents in terms of classification performance. Also the order of occurrence of words in documents does not much contribute to the classification performance and so documents can be considered as bags of words.

In text document categorization in our research, words are considered as features for representation of documents ignoring their order of occurrence in the documents. Machine learning algorithms for text categorization require numerical representation of these features for learning and categorization. There are different ways of assigning numerical values to these features. This section discusses a formal mathematical model for these representations.

Let $\mathcal{W} = \{w_1, w_2, \ldots\ w_N\}$ be the set of N words constituting the vocabulary set. Let $\mathcal{D} = \{d_1, d_2, \ldots\ d_P\}$ be the set of P documents such that

$$d_p = \{w_1(p)\ w_2(p) \ldots\ w_{|d(p)|}(p)\},\ w_m(p) \in \mathcal{W},\ 1 \leq m \leq |d(p)| \tag{2.0.1}$$

where $|d(p)|$ is the cardinality of d_p. Each document belongs to a particular class. A class can be represented as a set of documents. Let $\mathcal{C} = \{c_1, c_2, \ldots\ c_J\}$ be the set of classes such that

$$c_j = \{d_1(j)\ d_2(j) \ldots\ d_{|c_j|}(j)\},\ d_p(j) \in \mathcal{D},\ 1 \leq p \leq |c_j| \tag{2.0.2}$$

where $|c_j|$ is the cardinality of c_j.

Documents can be represented as N dimensional vectors of features with numerical values assigned to them. A document $d_p, 1 \leq p \leq P$ can be represented as a binary vector,

$$xb(p) = [1 \ 0 \ 0 \ \ldots\ldots \ 1]^T, \tag{2.0.3}$$

of binary values if the features are assigned with binary values i.e.

$$xb_i(p) = I(w_i \in d_p), \ 1 \leq i \leq N, \ 1 \leq p \leq P \tag{2.0.4}$$

where T is the transpose of the vector. The indicator function $I(x) = 1$ if x is true else $I(x) = 0$. If the first element of the vector $xb(p)$ is 1 this means that the first word w_1 from the vocabulary set $\mathcal{W}$ is present in the document. If the second element of vector xb is 0 this means that the second word w_2 from the vocabulary set $\mathcal{W}$ is absent in the document. A document, d_p, can be represented as a vector,

$$xt(p) = [tf_1(p) \ tf_2(p) \ \ldots\ldots \ tf_N(p)]^T, \tag{2.0.5}$$

of term frequency if the features are assigned with term frequency values i.e.

$$xt_i(p) = tf_i(p) = \sum_{l=1}^{|d(p)|} I(w_l(p) = w_i), \ \ 1 \leq i \leq N. \tag{2.0.6}$$

Term frequency, $tf_i(p) \in [0, tf_{max}]$, of a word in a document is the number of times it occurs in the document. tf_{max} is the the maximum number of times that a word can occur in a document. If the first element of the vector $xt(p)$ is $tf_1(p)$ this means that the document, d_p, contains the first word w_1 from the vocabulary set $\mathcal{W}$ and there are tf_1 number of such words in the document. Similarly the document, d_p contains tf_2 number of second word w_2 from the vocabulary set $\mathcal{W}$. A document, $d_p, \ 1 \leq p \leq P$, is represented as a vector,

$$xd(p) = [tfidf_1(p) \ tfidf_2(p) \ \ldots\ldots \ tfidf_N(p)]^T, \tag{2.0.7}$$

of term frequency inverse document frequency if the features are assigned with term frequency inverse document frequency values. Term frequency inverse document frequency, $tfidf_i(p) \in [0, tfidf_{max}]$, is the product of $tf_i(p)$ and inverse document frequency, idf_i, of

the word $w_i \in \mathcal{W}$. $tfidf_{max}$ is the maximum value that any $tfidf_i(p)$ can take. Inverse document frequency, idf_i, of a word is given by

$$idf_i = log(\frac{P}{df_i}) \tag{2.0.8}$$

Document frequency, df_i, is the number of the documents containing the word w_i i.e.

$$df_i = \sum_{p=1}^{P} I(w_i \in d_p),\ 1 \leq i \leq N,\ 1 \leq p \leq P. \tag{2.0.9}$$

Therefore [5]

$$xd_i(p) = tfidf_i(p) = tf_i(p) * idf_i. \tag{2.0.10}$$

A class c_j can also be represented as a vector, $xcd(j)$, of class document frequency, $dfc_i(j) \in [0, |c_j|]$,

$$xcd(j) = [dfc_1(j)\ dfc_2(j) \ldots\ dfc_N(j)]^T, j \in [1, J] \tag{2.0.11}$$

such that

$$xcd_i(j) = dfc_i(j) = \sum_{p=1}^{|c_j|} I(w_i \in d_p(j)),\ 1 \leq i \leq N. \tag{2.0.12}$$

Here the first element of vector $xcd(j)$ is $dfc_1(j)$ meaning that the class c_j contains the first word $w_1 \in \mathcal{W}$ and dfc_1 number of documents in the class contain the word w_1 and similarly for the other elements.

Class c_j can be represented as a vector,

$$xct(j) = [tfc_1(j)\ \ tfc_2(j)\ \ \ldots\ldots\ \ tfc_N(j)]^T, j \in [1, J], \tag{2.0.13}$$

of class term frequency, $tfc_i(j) \in [0, tfc_{max}]$. Here tfc_{max} is the maximum number of times that any word $w_i \in \mathcal{W}$ can occur in class c_j and class term frequency is given by

$$xct_i(j) = tfc_i(j) = \sum_{p=1}^{|c_j|} tf_i(p),\ 1 \leq i \leq N. \tag{2.0.14}$$

The first element of vector xct_j is $tfc_1(j)$ meaning that the class c_j contains the first word $w_i \in \mathcal{W}$ and there are total of tfc_1 number of the first word in the class c_j and similarly for the other elements of the vector $xct(j)$.

In case the documents are linked, such as conference papers with bibliographic citations, link pattern can be captured by introducing link features. Link features of a document $d \in \mathcal{D}$ can be modeled as the class distribution of the documents $d_p(j)$ the given document cites to and is cited by. The link features corresponding to the class distribution of documents that the given document cites to is given by

$$l(d)_{cite} = [l_O(c_1, d)\ l_O(c_2, d) \ldots \ldots l_O(c_J, d)]^T \tag{2.0.15}$$

where

$$l_O(c_j, d) = \sum_{p=1}^{|c_j|} I(d \rightarrow d_p(j)). \tag{2.0.16}$$

$I(d \rightarrow d_p(j)) = 1$ if the given document d cites the document $d_p(j)$ otherwise $I(d \rightarrow d_p(j)) = 0$. The link features corresponding to the class distribution of documents that the given document is cited by is given by

$$l(d)_{cited} = [l_I(c_1, d)\ l_I(c_2, d) \ldots \ldots l_I(c_J, d)]^T \tag{2.0.17}$$

where

$$l_I(c_j, d) = \sum_{p=1}^{|c_j|} I(d \leftarrow d_p(j)). \tag{2.0.18}$$

$I(d \leftarrow d_p(j)) = 1$ if the given document d is cited by the document $d_p(j)$ otherwise $I(d \leftarrow d_p(j)) = 0$.

This feature representation corresponds to the term frequency of the link in that it provides with the number of class documents that the given document cites to or is cited by. The link features can also be modeled in terms of presence or absence of class of documents that the given document cites to or is cited by. The binary link features defined above for the class of documents that the given document cites to is given by

$$lb(d)_{cite} = [lb_O(c_1, d)\ lb_O(c_2, d) \ldots \ldots lb_O(c_J, d)]^T \tag{2.0.19}$$

where

$$lb_O(c_j, d) = I(l_O(c_j, d) > 0),\ \ I(x > 0) = 1\ \ if\ \ x > 0\ \ else\ \ I(x > 0) = 0. \tag{2.0.20}$$

The binary link features corresponding to the class of documents that the given document is cited by is given by

$$lb(d)_{cited} = [lb_I(c_1, d)\ lb_I(c_2, d) \ldots \ldots lb_I(c_J, d)]^T \tag{2.0.21}$$

where

$$lb_I(c_j, d) = I(l_I(c_j, d) > 0), \;\; I(x > 0) = 1 \;\; if \;\; x > 0 \;\; else \;\; I(x > 0) = 0. \quad (2.0.22)$$

This feature representation corresponds to the class document frequency of the link in that it provides with the number of documents in a class that cite to or are cited by documents from certain class.

Chapter 3

Machine Learning Algorithms

3.1 Bayesian Learning

For a statistical experiment ζ which has S as the set of possible outcomes with $\{c_1, c_2 \ldots\ldots c_J\}$ as the partition of S. Let $Pr(d)$, $d \subseteq S$ be the probability distribution defined on all events in S. Then for the event c_j and d in S,

$$Pr(c_j|d) = \frac{Pr(d|c_j)Pr(c_j)}{Pr(d)} \quad j = 1, 2, \ldots J. \tag{3.1.1}$$

$Pr(d) > 0$, $Pr(c_j)$ is the prior probability, $Pr(d|c_j)$ is the likelihood probability and $Pr(c_j|d)$ is the posterior probability [25]. Also from the law of total probability

$$Pr(d) = \sum_{j=1}^{J} Pr(d|c_j)Pr(c_j). \tag{3.1.2}$$

Here the denominator is the normalizing factor. Then

$$Pr(c_j|d) \propto Pr(d|c_j)Pr(c_j) \quad j = 1, 2, \ldots J \tag{3.1.3}$$

or

$$Pr(c_j|d) \propto L(c_j|d)Pr(c_j) \quad j = 1, 2, \ldots J \tag{3.1.4}$$

where

$$L(c_j|d) \propto Pr(d|c_j) \tag{3.1.5}$$

is the likelihood function. Therefore the posterior probability is proportional to the product of likelihood function and the prior. The likelihood funtion is based upon the observed data

whereas the prior is based upon the previous knowledge about the probability distribution. This is how Bayesian learning uses probability to measure the uncertainty about posterior predictions.

Bayesian learning is important in that it helps measure the uncertainty of predictions, suggests ways to adapt to the characteristics of the data such as smoothness, degree of relevancy etc. Since the posterior predicted distribution depends upon the prior distribution, the proper prior distribution representing uncertainty as to the relevancy of available data, smoothness function and noise level need to be considered very carefully.

3.1.1 Naive Bayes (NB)

This algorithm assumes a probabilistic model for the generation of text and makes a simplifying assumption of word independence in documents [29]. In this model there are total of J classes and each class is represented as vector, $xct(j)$, of class term frequency $tfc_i(j)$. The test document d' which is a set of words $d' = \{w'_1, w'_2 \ldots w'_{|d'|}\}$ is represented as vector $xt(d')$ of term frequency $tf_i(d')$. $|d'|$ is the cardinality of d'. The decision, $H_{NB}(d')$, on a test document d' as the class it belongs to is given by

$$H_{NB}(d') = \underset{c_j \in C}{^{argmax}} Pr(c_j|d') \tag{3.1.6}$$

where the function $\underset{x}{^{argmax}} f(x)$ returns that value of x for which $f(x)$ has maximum value. Using Bayes formula, $Pr(c_j|d')$ can be written as

$$Pr(c_j|d') = \frac{Pr(d'|c_j)Pr(c_j)}{Pr(d')} \tag{3.1.7}$$

or

$$Pr(c_j|d') = \frac{Pr(d'|c_j)Pr(c_j)}{\sum_{c' \in C} Pr(d'|c')Pr(c')}. \tag{3.1.8}$$

Here, $Pr(c_j)$ is the prior probability of occurrence of a document belonging to class c_j and $Pr(d'|c_j)$ is the likelihood function that gives probability of observing the document d' given it belongs to the class c_j and is given by

$$Pr(d'|c_j) = Pr(\{w_1(d'), w_2(d'), \ldots\ldots, w_{(|d'|)}(d')\}|c_j). \tag{3.1.9}$$

Finding the joint probability in the above equation is computationally expensive and so to reduce the computational cost words in d' are assumed to occur independently of one another so that

$$Pr(d'|c_j) = Pr(\{w_1(d'), w_2(d'), \ldots\ldots, w_{(|d'|)}(d')\}|c_j) = \prod_{i=1}^{|d'|} Pr(w_i(d')|c_j). \quad (3.1.10)$$

Substitution of (3.1.10) into (3.1.8) gives

$$Pr(c_j|d') = \frac{Pr(c_j)\prod_{i=1}^{|d'|} Pr(w_i(d')|c_j)}{\sum_{c'\in\mathcal{C}} Pr(c')\prod_{i=1}^{|d'|} Pr(w_i(d')|c')} \quad (3.1.11)$$

Finally, the decision of Naive Bayes algorithm is given by

$$H_{NB}(d') = \underset{c_j\in\mathcal{C}}{argmax} \frac{Pr(c_j)\prod_{w_i(d')\in\mathcal{W}} Pr(w_i(d')|c_j)^{tf_i(d')}}{\sum_{c'\in\mathcal{C}} Pr(c')\prod_{w_i(d')\in\mathcal{W}} Pr(w_i(d')|c')^{tf_i(d')}}. \quad (3.1.12)$$

$Pr(c_j)$ is estimated as

$$\widehat{Pr}(c_j) = \frac{|c_j|}{P}, \quad (3.1.13)$$

where $|c_j|$ is the number of documents belonging to the class c_j and P is the total number of documents. $Pr(w_i(d')|c_j)$ is estimated as

$$\widehat{Pr}(w_i(d')|c_j) = \frac{1 + tfc_i(j)}{N + \sum_{i=1}^{N} tfc_i(j)}, \quad (3.1.14)$$

where $tfc_i(j)$ is the number of times word w_i occurs in class c_j and $tf_i(d')$ is the total number of times word $w_i(d')$ occurs in the test document d'.

The model has to be modified when the class is represented as vector of class document frequency $dfc_i(j)$. The test document d' is now represented as vector $xb(d')$ of binary values. The decision of modified Naive Bayes, $H_{ModNB}(d')$ on the test document d' as the class it belongs to is given by

$$H_{ModNB}(d') = \underset{c_j\in\mathcal{C}}{argmax} \frac{Pr(c_j)\prod_{w_i(d')\in\mathcal{W}} Pr'(w_i(d')|c_j)}{\sum_{c'\in\mathcal{C}} Pr(c')\prod_{w_i(d')\in\mathcal{W}} Pr'(w_i(d')|c')}. \quad (3.1.15)$$

$Pr'(w_i(d')|c_j)$ is estimated as

$$\widehat{Pr'}(w_i(d')|c_j) = \frac{1 + dfc_i(j)}{N + \sum_{i=1}^{N} dfc_i(j)}, \quad (3.1.16)$$

where $dfc_i(j)$ is the number of documents in class c_j that contain the word $w_i(d')$. Since only the presence or absence of words in a document is being considered, the term frequency $tf_i(d')$ becomes irrelevant.

3.1.2 Probabilistic Term Frequency Inverse Document Frequency (PrTFIDF)

The motivation for this algorithm is Rocchio relevance feedback algorithm using term frequency inverse document frequency (TFIDF). This algorithm considers generative model of the text documents and shows how it is related to the Rocchio algorithm. However, in our research this algorithm has been considered to assess the importance of class document frequency in terms of performance of this algorithm.

This algorithm considers words as descriptors of documents and assigns different probabilities to different words being considered as descriptors of a document [5]. In PrTFIDF, the words are considered as belonging to a document of certain class unlike NB in which words are are considered as belonging to a class without regard to the document they are coming from.

In this model there are a total of J classes and each class is represented as vector of class term frequency. The test document d' is represented as vector, $xt(d')$, of term frequency, $tf_i(d')$. The decision, $H_{PrTFIDF}(d')$, on a test document d' as the class it belongs to is given by

$$H_{PrTFIDF}(d') = \underset{c_j \in C}{argmax}\ Pr(c_j|d'). \tag{3.1.17}$$

$$Pr(c_j|d') = \sum_{w_i(d') \in \mathcal{W}} Pr(c_j, w_i(d')|d'), \tag{3.1.18}$$

$$Pr(c_j|d') = \sum_{w_i(d') \in \mathcal{W}} Pr(c_j|w_i(d'), d') Pr(w_i(d')|d'), \tag{3.1.19}$$

where the probability of word $w_i(d')$ being a descriptor of the document d' is given by $Pr(w_i(d')|d')$. Using the Bayes formula,

$$Pr(c_j|w_i(d'), d') = \frac{Pr(d'|c_j, w_i(d'))\ Pr(c_j|w_i(d'))}{Pr(d'|w_i(d'))}. \tag{3.1.20}$$

Then

$$Pr(c_j|d') = \sum_{w_i(d') \in \mathcal{W}} \frac{Pr(d'|c_j, w_i(d'))Pr(c_j|w_i(d'))Pr(w_i(d')|d')}{Pr(d'|w_i(d'))}. \quad (3.1.21)$$

Here assumption is made that

$$Pr(d'|w_i(d')) = Pr(d'|c_j, w_i(d')). \quad (3.1.22)$$

Then

$$Pr(c_j|d') = \sum_{w_i(d') \in \mathcal{W}} Pr(c_j|w_i(d'))Pr(w_i(d')|d'). \quad (3.1.23)$$

Using the Bayes formula

$$Pr(c_j|w_i(d')) = \frac{Pr(w_i(d')|c_j)Pr(c_j)}{\sum_{c' \in \mathcal{C}} Pr(w_i(d')|c')Pr(c')}. \quad (3.1.24)$$

Then

$$H_{PrTFIDF}(d') = \underset{c_j \in \mathcal{C}}{argmax} \sum_{w_i(d') \in \mathcal{W}} \frac{Pr(c_j)\,\mathrm{Pr}(w_i(d')|c_j)}{\sum_{c' \in \mathcal{C}} Pr(c')Pr(w_i(d')|c')} Pr(w_i(d')|d'). \quad (3.1.25)$$

Here $Pr(c_j)$ is estimated by

$$\widehat{Pr}(c_j) = \frac{|c_j|}{P} \quad (3.1.26)$$

where $|c_j|$ is the number of documents belonging to the class c_j and P is the total number of documents. $Pr(w_i(d')|c_j)$ is estimated by

$$\widehat{Pr}(w_i(d')|c_j) = \frac{1}{|c_j|} \sum_{d_p \in c_j} \widehat{Pr}(w_i(d')|d_p) \quad (3.1.27)$$

where

$$\widehat{Pr}(w_i(d')|d_p) = \frac{tf_i(p)}{\sum_{i=1}^{N} tf_i(p)}. \quad (3.1.28)$$

Here $tf_i(p)$ is the number of times word $w_i(d')$ occurs in document d_p.

The model has to be modified when the class is represented as vector of class document frequency. Then the test document d' is represented as vector, $xb(d')$, of binary values. The decision of modified PrTFIDF, $H_{ModPrDF}(d')$, on the test document as the class it belongs to is given by

$$H_{ModPrDF}(d') = \underset{c_j \in \mathcal{C}}{argmax} \sum_{w_i(d') \in \mathcal{W}} \frac{Pr(c_j)\,\mathrm{Pr}'(w_i(d')|c_j)}{\sum_{c' \in \mathcal{C}} Pr(c')Pr'(w_i(d')|c')}. \quad (3.1.29)$$

$Pr(c_j)$ is estimated as before. $Pr'(w_i(d')|c_j)$ is estimated as

$$\widehat{Pr'}(w_i(d')|c_j) = \frac{dfc_i(j)}{\sum_{i=1}^{N} dfc_i(j)}. \tag{3.1.30}$$

Here $dfc_i(j)$ is the number of documents in the class c_j containing the word $w_i(d')$ that occurs in document d'. Also, since only the presence of word in the document is being considered, $Pr(w_i(d')|d')$ is replaced with indicator fucntion, $I(w_i(d') \in d')$, representing d' in terms of presence or absence of words $w_i(d')$.

3.2 Support Vector Machines (SVM)

SVM is a maximum margin classifier that maximizes the minimum distance of the decision hyperplane from the positive and negative examples [6][9][10][30]. In this model there are 2 classes, class 1 constituting positive examples labeled as $t_p = +1$ and class 2 constituting negative examples labeled as $t_p = -1$. The training document $d_p \in \mathcal{D}$ is represented as vector, $x(p)$ of some features of that document and the test document d' is represented as vector $x(d')$ of similar features of corresponding test document. The decision, $H_{SVM}(d')$, on a test document d' as the class it belongs to is given by

$$H_{SVM}(d') = sign(w^T x(d') + b). \tag{3.2.1}$$

If the decision is $+1$ the test document belongs to class 1 otherwise it belongs to class 2. SVM is based on the Structural Risk Minimization principle.

3.2.1 Structural Risk Minimization

While minimizing risk during machine learning it is desired to learn parameters w such that the expected value of risk,

$$R(w) = \int L(t, f(x, w))dF(x, t) \tag{3.2.2}$$

is minimized. Here t is the correct output or label associated with the vector x, $F(x, t)$ is the probability measure defined on $\mathcal{X} \times \mathcal{T}$ and $L(t, f(x, w))$ is the loss function. Loss

functions are different depending upon the type of learning being done. For regression learning a risk function is the quadratic loss function

$$R(w) = \int (t - f(x, w))^2 dF(x, t), \tag{3.2.3}$$

for classification learning a risk function is probability of error

$$R(w) = \int I(f(x, w) \neq t) dF(x, t). \tag{3.2.4}$$

Since the probability measure $F(x, t)$ is unknown, risk is minimized according to the inductive principle and so the empirical risk for regression learning is

$$R_{emp}(w) = \frac{1}{K} \sum_{k=1}^{K} (t_k - f(x_k, w))^2, \tag{3.2.5}$$

and the empirical risk for classification learning is

$$R_{emp}(w) = \frac{1}{K} \sum_{k=1}^{K} I(f(x_k, w) \neq t_k). \tag{3.2.6}$$

Here empirical risk for classification will be considered as we are dealing with classification problem. Empirical risk is a function of training error which is the number of error made by the learning algorithm on training samples. Generalization error is defined as the number of errors made by the learning algorithm on unseen test examples. Assumption is made that both the training and test examples come from the same probability distribution. Then Structural risk minimization (SRM), which is an inductive principle, simultaneously minimizes the empirical risk and capacity or Vapnik Chervonenkis (VC) dimension of the learning algorithms [23][30]. So if a set of approximating functions

$$S_l = \{f(x, w); w \in \mathcal{W}_l\}, \quad l = 1, 2, \ldots, L \tag{3.2.7}$$

is given by the nested structure

$$S_1 \subset S_2 \subset S_3 \ldots \subset S_l \ldots \tag{3.2.8}$$

with the increasing VC dimension given by

$$h_1 \leq h_2 \leq h_3 \ldots \leq h_l \ldots \tag{3.2.9}$$

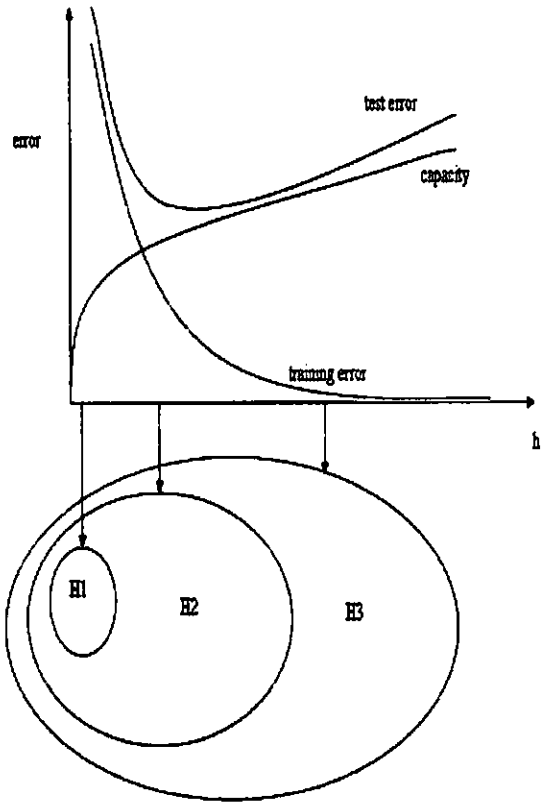


Figure 3.1: Structural risk minimization

where h_l is the capacity or the complexity of the l^{th} approximating function. Then SRM provides a formal way of finding a model such that it has optimal complexity and minimum empirical risk. This is as shown in the figure above. According to this principle the acutal risk is given by

$$R(w) \leq R_{emp}(w) + \sqrt{\frac{h \log(\frac{2P}{h} + 1) - \log \frac{\eta}{4}}{P}} \tag{3.2.10}$$

where P is the number of training samples for which $R_{emp}(w)$ is determined, h is the capacity of the learning algorithm. The inequality holds with confidence of at least $1 - \eta$. The actual risk is bounded by sum of the empirical risk and confidence interval. As shown in the figure the training error decreases monotonically for a fixed number of samples as the VC dimension and hence the confidence interval increases. The generalization error goes through a minimum in this process. Before the generalization error reaches the minimum, the capacity of approximating function is too small to capture all the details of training samples and after the generalization error reaches the minimum the capacity of the approximating function is large than required to capture the details of training samples. This shows a tradeoff between the training error and the complexity of approximating function. Therefore there is a need for regularization of the capacity of the approximation function. The minimum of generalization error corresponds to the condition in which the capacity is optimally regularized in the sense that it has enough complexity in order to capture the details of the training samples to ensure minimum possible empirical risk and yet is simple enough to generalize on the unseen test samples to ensure minimum generalization error.

The hyperplanes corresponding to SVM that separate the data samples with maximum margin fulfil this requirement in that they minimize the empirical error and at the same time have optimal complexity by having only a few support vectors on the margin.

3.2.2 SVM with No Class Overlap

For the case in which the data are separable

$$w^T x(p) + b \geq 1, \; if \; t_p = 1 \tag{3.2.11}$$

$$w^T x(p) + b \leq -1, \; if \; t_p = -1 \tag{3.2.12}$$

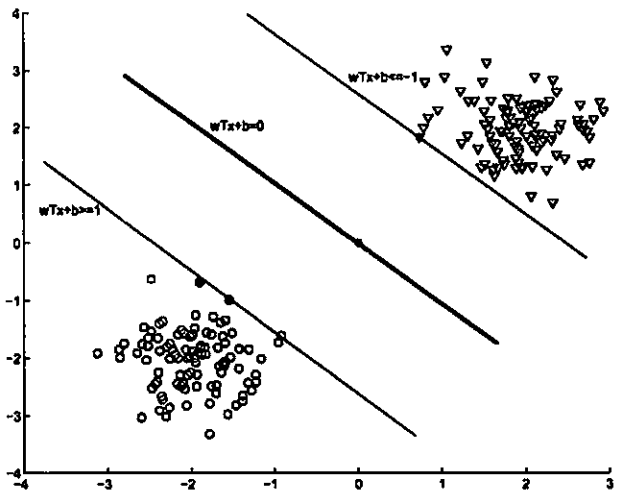


Figure 3.2: Maximum margin classifier without class overlap

These can be combined into one condition

$$t_p(w^T x(p) + b) \geq 1 \tag{3.2.13}$$

Here the equalities are satisfied for the examples lying on the hyperplane on the margin. All the other examples lie on the other side of the hyperplane. It can be shown that the distance between hyperplanes or the margin is

$$\frac{2}{||w||}, \tag{3.2.14}$$

which means that maximizing the margin is the same as minimizing $||w||$. This is equivalent to minimizing $||w||^2$ which is equivalent to solving a quadratic programming (QP) problem and is given by

$$\underset{w,b}{argmin} \frac{1}{2}||w||^2 \tag{3.2.15}$$

$$subject\ to\ t_p(w^T x(p) + b) \geq 1,\ \ t_p \in \{+1, -1\}. \tag{3.2.16}$$

This constrained optimization problem can be solved by introducing Lagrange multipliers $\alpha_p \geq 0$, with one multiplier α_p for each of the constraints

$$t_p(w^T x(p) + b) \geq 1,\ for\ p \in \{1 \ldots P\} \tag{3.2.17}$$

giving the Lagrangian function

$$L(w, b, \alpha) = \frac{1}{2}||w^2|| - \sum_{p=1}^{P} \alpha_p\{t_p(w^T x(p) + b) - 1\}. \tag{3.2.18}$$

Minimizing the Lagrangian with respect to w and b gives

$$w = \sum_{p=1}^{P} \alpha_p t_p x(p) \tag{3.2.19}$$

and

$$\sum_{p=1}^{P} \alpha_p t_p = 0. \tag{3.2.20}$$

The duel representation of this problem is to maximize the function

$$\tilde{L}(\alpha) = \sum_{p=1}^{P} \alpha_p - \frac{1}{2} \sum_{p=1}^{P} \sum_{p1=1}^{P} \alpha_p \alpha_{p1} t_p t_{p1} x_p x_{p1} \tag{3.2.21}$$

$$subject\ to\ \ \alpha_p \geq 0,\ p = 1, \ldots\ P,\ \sum_{p=1}^{P} \alpha_p t_p = 0. \tag{3.2.22}$$

Here the non zero Lagrangian multiplier α_p corresponds to those document vectors that are on the margin and are called Support Vectors represented by $xs(p)$. Once w has been found, b can be found using $b = t_p - w^T xs(p)$.

3.2.3 SVM with Class Overlap

For the case in which the data are not entirely separable

$$w^T x(p) + b \geq 1 - \xi_p,\ if\ t_p = 1 \tag{3.2.23}$$

$$w^T x(p) + b \leq -(1 - \xi_p),\ if\ t_p = -1 \tag{3.2.24}$$

These can be combined into one condition

$$t_p(w^T x(p) + b) \geq 1 - \xi_p. \tag{3.2.25}$$

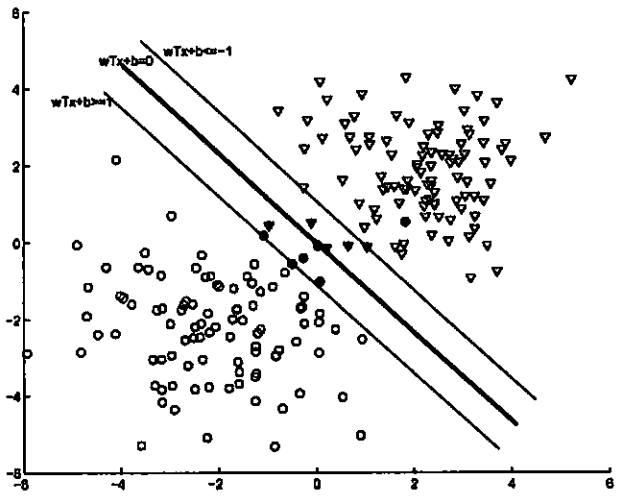


Figure 3.3: Maximum margin classifier with class overlap

Here the the slack variables $\xi_p = 0$ for those samples that are on or inside the correct margin, $0 < \xi_p \leq 1$ for those samples that lie inside the margin but on the correct side of the decision boundary and $\xi_p > 1$ for those samples that lie on the wrong side of the decision boundary. This way the hard margin has been transformed into soft margin through slack variables by allowing some examples to be misclassified. So the optimization problem now is that of maximizing the margin but by penalizing the outliers and so the corresponding QP is given by

$$\underset{w,b}{arg\ min}\ \frac{1}{2}||w^2|| + C\sum_{p=1}^{P}\xi_p,\ \xi_p \geq 0 \tag{3.2.26}$$

$$subject\ to\ t_p(w^T x(p) + b) \geq 1 - \xi_p,\ \ t_p \in \{+1, -1\}. \tag{3.2.27}$$

This constrained optimization problem can be solved by introducing Lagrange multipliers $\alpha_p \geq 0$, with one multiplier α_p for each of the constraints

$$t_p(w^T x(p) + b) \geq 1 - \xi_p,\ for\ p \in \{1 \ldots\ P\} \tag{3.2.28}$$

giving the Lagrangian function

$$L(w, b, \alpha) = \frac{1}{2}||w^2|| + C\sum_{p=1}^{P}\xi_p - \sum_{p=1}^{P}\alpha_p\{t_p(w^T x(p) + b) - 1 + \xi_p\} - \sum_{p=1}^{P}\mu_p\xi_p. \tag{3.2.29}$$

Minimizing the Lagrangian with respect to w, b and ξ_p gives

$$w = \sum_{p=1}^{P} \alpha_p t_p x(p), \tag{3.2.30}$$

$$\sum_{p=1}^{P} \alpha_p t_p = 0 \tag{3.2.31}$$

and

$$\alpha_p = C - \mu_p. \tag{3.2.32}$$

The duel representation of this problem is to maximize the function

$$\tilde{L}(\alpha) = \sum_{p=1}^{P} \alpha_p - \frac{1}{2} \sum_{p=1}^{P} \sum_{p1=1}^{P} \alpha_p \alpha_{p1} t_p t_{p1} x_p x_{p1} \tag{3.2.33}$$

$$subject\ to\ 0 \leq \alpha_p \leq C,\ p = 1, \ldots\ P,\ \sum_{p=1}^{P} \alpha_p t_p = 0. \tag{3.2.34}$$

Here the non zero Lagrangian multiplier $0 < \alpha_p < C$ corresponds to those document vectors that are on the margin and are called Support Vectors represented by $xs(p)$. Once w has been found, b can be found using $b = t_p - w^T xs(p)$.

3.2.4 Kernel for SVM

Kernel function [32] measures the similarity among the data samples and is represented as $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$. When the data samples are not linearly separable then kernels can be incorporated into SVM by transforming the data samples x into high dimensional feature space $\phi(x)$ to make the data samples linearly separable in the high dimensional feature space. Although the data samples are transformed into a high dimensional feature space $\phi(x)$ yet because of the kernel functions the computation is done in the low dimensional feature space x without having to know the transformation $\phi(x)$. This significantly reduces the computational cost for training. kernel function can be easily incorporated in (3.2.33) as

$$\tilde{L}(\alpha) = \sum_{p=1}^{P} \alpha_p - \frac{1}{2} \sum_{p=1}^{P} \sum_{p1=1}^{P} \alpha_p \alpha_{p1} t_p t_{p1} K(x_p, x_{p1}) \tag{3.2.35}$$

$$subject\ to\ 0 \leq \alpha_p \leq C,\ p = 1, \ldots\ P,\ \sum_{p=1}^{P} \alpha_p t_p = 0. \tag{3.2.36}$$

Some of the important kernel functions are:

$$Linear\ kernel\ function\ K(x, y) = x^T y, \tag{3.2.37}$$

$$Polynomial\ kernel\ function\ K(x, y) = (x^T y + 1)^p, \tag{3.2.38}$$

$$Gaussian\ kernel\ function\ K(x, y) = exp(-\frac{1}{\sigma^2}||x - y||^2), \tag{3.2.39}$$

$$Sigmoidal\ kernel\ function\ \tanh(ax^T y - b)\ for\ some\ a\ and\ b. \tag{3.2.40}$$

For training SVM, training documents, $d_p \in \mathcal{D}$, are represented as vectors, $xt(p)$, of term frequency, $tf_i(p)$, and vectors, $xb(p)$, of binary values, $xb_i(p)$, respectively. When SVM is used to classify documents that have link information also [16], vectors corresponding to the documents are augmented with link information by concatenating the link vector with the document vector. When the term frequency of links as in (2.0.15) and (2.0.17) are considered as features to form the link vector, the augmented training document vector is given by

$$xbc(p) = [xb(p);\ \ l(d_p)_{cite};\ \ l(d_p)_{cited}]. \tag{3.2.41}$$

When the class document frequency of links as in (2.0.19) and (2.0.21) are considerd as features for the link vector, the augmented training document vector is given by

$$xbb(p) = [xb(p);\ \ lb(d_p)_{cite};\ \ lb(d_p)_{cited}] \tag{3.2.42}$$

which is then used to train the SVM. The corresponding augmented test document vectors for the above two cases are respectively given by

$$xbc(d') = [xb(d');\ l(d')_{cite};\ l(d')_{cited}] \tag{3.2.43}$$

$$xbb(d') = [xb(d');\ lb(d')_{cite};\ lb(d')_{cited}] \tag{3.2.44}$$

which are used to test the performance of SVM. Linear kernel has been used for training of SVM [33] because the data are sparse.

3.3 K Nearest Neighbor (KNN)

In this algorithm, K nearest documents in training set to the test document is found and the class with the maximum number of the nearest neighbor documents is assigned to the test document [8]. Here each document $d_p \in \mathcal{D}$ is represented as vector, $xt(p)$, of term frequency, $tf_i(p)$. There are total of J classes. The test document d' is represented as vector, $xt(d')$, of term frequency, $tf_i(d')$. The decision, $H_{KNN}(d')$, on a test document as the class it belongs to is given by

$$H_{KNN}(d') = \underset{c_j \in \mathcal{C}}{argmax} \sum_{xt(p) \in KNN} I(xt(p) \in c_j) \tag{3.3.1}$$

where KNN are the K nearest neighbor documents to the test document. The distance between the test document and the training documents in different classes is computed as

$$||xt(d') - xt(p)|| = \sqrt{\sum_{i=1}^{N} (tf_i(d') - tf_i(p))^2}. \tag{3.3.2}$$

The distances are sorted and K minimum distances are chosen which form the K nearest neighbors of the test document.

For the algorithm to be able to learn class document frequency the training document is represented as a vector, $xb(p)$, of binary values and the test document is represented as vector, $xb(d')$. The distance between the test document and training documents in different classes is then computed as

$$|xb(d') - xb(p)| = \sum_{i=1}^{N} I(xb_i(d') \neq xb_i(p)). \tag{3.3.3}$$

Again, the distances are sorted and K minimum distances are chosen which form the K nearest neighbors of the test document. The decision, $H_{KNN}(d')$, on the test document d' as the class it belongs to is now given by

$$H_{KNN}(d') = \underset{c_j \in C}{argmax} \sum_{xb(p) \in KNN} I(xb(p) \in c_j). \tag{3.3.4}$$

3.4 Algorithms Using Class Document Frequency(dfc)

In this section we discuss two simple algorithms just using dfc.

3.4.1 Classification Based on dfc

The algorithm, $Algd_1$, examines class document frequency vector, $xcd(j)$. There are J classes so there are J vectors, $xcd(j)$. A test document is represented as vector $xb(d')$ of binary values. When a new document d' from the test collection is considered for classification, words from class c_1 are checked for their presence or absence in the test document. A vector of 1's and 0's is formed with 1 indicating the presence and 0 indicating the absence of the word in the test document. Let $xb(d'_1) = [1\ 0\ 0\ \ldots\ldots\ 1]^T$, be the vector corresponding to the test document for the first class c_1. This means that the first word from c_1 is present in the test document where as the second and the third words from c_1 are absent in the test document. The same process is repeated for all the classes $c_j, 1 \leq j \leq J$ and corresponding vectors $xb(d'_j)$ are determined. Once all the vectors $xb(d'_j)$ have been determined, inner product of each of these vectors is taken with the corresponding class vector $xcd(j)$ which is given by

$$cls_j = < xb(d'_j), xcd(j) > \tag{3.4.1}$$

where $< a, b >$ is the inner product of the vectors a and b. The decision, $H_{Algd_1}(d')$, of the algorithm as the class to which the test document belongs is given by,

$$H_{Algd_1}(d') = \underset{j \in J}{argmax}\ cls_j. \tag{3.4.2}$$

3.4.2 Classification Based on dfc with High Value

The algorithm, $Algd_2$, examines class document frequency, $dfc_i(j)$, of words and uses some heuristics so that $dfc_i(j)$ of words with high value contributes more to the classification than $dfc_i(j)$ of words with low value in a document. The reason for doing this is to assess whether words with high $dfc_i(j)$ contributes dominantly towards the classification performance of an algorithm.

This algorithm assumes vector representation, $xcd(j)$ of class c_j. A test document is represented as vector $xb(d')$. There are J classes so there are J vectors, $xcd(j)$. Each of these vectors is normalized to get a normalized vector $nxcd(j) = \frac{xcd(j)}{max_i(xcd_i(j))}$, where $max_i(xcd_i(j))$ which is the maximum value occurring in the vector, $xcd(j)$, divides all the elements of that vector to give the vector $nxcd(j)$. The elements of vector, $nxcd(j)$ whose value is less than 0.1 is removed meaning that those words that occur less than ten percent of the documents are removed from the vector. A new vector $nxcd(j0.1)$ with $N1$ elements is formed in which the element with least value is 0.1. This process is applied to vectors of all classes. The vector is now represented as

$$nxcd(j0.1) = [dfc'_1(j)\ \ dfc'_2(j)\ \ \ldots\ \ dfc'_{N1}(j)]^T \tag{3.4.3}$$

where $dfc'_m(j), 1 \leq m \leq N1$ is a normalized class document frequency. For each these vectors, $nxcd(j0.1)$, fraction of elements are considered as contributing significantly to the classification of documents. In this paper, vector elements with value of at least 0.5 are considered as contributing 50% to the classification of training documents. All the elements of a vector $nxcd(j0.1)$ with value at least 0.5 are considered to form a new vector $nxcd(j0.5)$ with $N2$ elements in which the element having the least value is 0.5. Value of 0.5 is considered because it gives the best classification result for the documents considered. It is worth noting that the size of this vector is very small compared to the size of $nxcd(j)$ which is a large vector. This vector is represented as

$$nxcd(j0.5) = [dfc'_1(j)\ \ dfc'_2(j)\ \ \ldots\ \ dfc'_{N2}(j)]^T. \tag{3.4.4}$$

To solve this problem, an unknown α_j has to be found such that the condition

$$\frac{\sum_{m=1}^{N2}(exp(\alpha_j\ nxcd_m(j0.5)))}{\sum_{m=1}^{N1}(exp(\alpha_j\ nxcd_m(j0.1)))} = 0.5 \tag{3.4.5}$$

is fulfilled. The value of α_j is found by assuming some initial value and then iterating until (3.4.5) is approximately satisfied. Then for each class the weight vector, $w_{j0.1}$, of $dfc'_m(j) \geq 0.1$ is determined as follows

$$w_{j0.1} = exp(\alpha_j \, nxcd(j0.1)). \tag{3.4.6}$$

Now that the weight vectors, $w_{j0.1}$, for different classes have been determined, test documents can be operated upon by this algorithm so that classification performance of the algorithm can be assessed.

When a document d' from the test collection is considered for classification, words from class c_1 with $dfc'_m(j) \leq 0.1$ are checked for their presence or absence in the test document. A vector of 1's and 0's is formed with 1 indicating the presence and 0 indicating the absence of the word in the test document. Let

$$xb(d'_1) = [1 \;\; 0 \;\; 0 \;\; \ldots\ldots \;\; 1]^T, \tag{3.4.7}$$

be the vector corresponding to the test document for the class c_1. This means that the first word corresponding to vector $nxcd(j0.1), j = 1$, is present in the test document where as the second and the third words corresponding to $nxcd(j0.1), j = 1$, are absent in the test document. The same process is repeated for all the classes $c_j, 1 \leq j \leq J$ and corresponding vectors $x(d'_j)$ are determined. Inner product of each of these vectors $xb(d'_j)$ is taken with the weight vector $w_{j0.1}$ calculated for the corresponding class as in (3.4.6)

$$cls_j = < xb(d'_j), w_{j0.1} > . \tag{3.4.8}$$

The decision on the class to which the given test document, d', belongs is given by

$$H_{Algd_2}(d') = \overset{argmax}{j \in J} \; cls_j. \tag{3.4.9}$$

3.5 Algorithms Using Class Term Frequency(tfc)

In this section we discuss the same algorithms but using tfc.

3.5.1 Classification Based on tfc

The algorithm, $Algt_1$, examines class term frequency vector, $xct(j)$. There are J classes so there are J vectors, $xct(j)$. A test document is represented as vector $xt(d')$ of term frequency. When a document d' from test collection is considered for classification, words from class c_1 are checked for their presence or absence in the test document. Then a vector of term frequency is formed which is given by $xt(d'_1) = [tf_1(d'_1)\ 0\ 0\ \ldots\ldots\ tf_N(d'_1)]^T$ and which corresponds to the test document for the first class c_1. This means that the first word corresponding to vector $xct(1)$ is present in the test document where as the second and the third words corresponding to vector $xct(1)$ are absent in the test document. The same process is repeated for all the classes $c_j, 1 \leq j \leq J$ and the corresponding vectors, $xt(d'_j)$, are determined. Once all such vectors have been determined, inner product of each of these vectors is taken with the corresponding class vectors $xct(j)$ which is given by

$$cls_j =< xt(d'_j), xct(j) > . \tag{3.5.1}$$

The decision, $H_{Algt_1}(d')$, of the algorithm as the class to which the test document belongs is given by

$$H_{Algt_1}(d') = \stackrel{argmax}{j \in J}\ cls_j. \tag{3.5.2}$$

3.5.2 Classification Based on tfc with High Value

The algorithm, $Algt_2$, examines class term frequency, $tfc_i(j)$, of words and uses some heuristics so that $tfc_i(j)$ of words with high value contributes more to the classification than $tfc_i(j)$ of words with low value in a document. The reason for doing this is to assess whether words with high $tfc_i(j)$ contributes dominantly towards the classification performance of an algorithm when compared with words with high $dfc_i(j)$.
This algorithm assumes vector representation, $xct(j)$ of class c_j. A test document, d', is represented as vector $xt(d')$. There are J classes so there are J vectors, $xct(j)$. Each

of these vectors is normalized to get a normalized vector $nxct(j) = \frac{xct(j)}{max_i(xct_i(j))}$, where $max_i(xct_i(j))$ which is the maximum value occurring in the vector, $xct(j)$, divides all the elements of that vector to give the vector $nxct(j)$. The elements of vector, $nxct(j)$, whose value is less than 0.1 is removed. A new vector, $nxct(j0.1)$, with $N1$ elements is formed in which the element with least value is 0.1. This process is applied to vectors of all classes. The vector is now represented as

$$nxct(j0.1) = [tfc'_1(j)\ \ tfc'_2(j)\ \ \ldots\ tfc'_{N1}(j)]^T \tag{3.5.3}$$

where $tfc'_m(j), 1 \leq m \leq N1$ is a normalized class term frequency. For each of these vectors, $nxct(j0.1)$, fraction of elements are considered as contributing significantly to the classification of documents. In this paper, vector elements with value of at least 0.5 are considered as contributing 50% to the classification of training documents. All the elements of a vector $nxct(j0.1)$ with value of at least 0.5 are considered to form a new vector $nxct(j0.5)$ with $N2$ elements in which the element having the least value is 0.5. Value of 0.5 is considered because it gives the best classification result for the documents considered. This vector is represented as

$$nxct(j0.5) = [tfc'_1(j)\ \ tfc'_2(j)\ \ \ldots\ tfc'_{N2}(j)]^T. \tag{3.5.4}$$

To solve this problem, an unknown α_j has to be found such that the condition below is fulfilled.

$$\frac{\sum_{m=1}^{N2}(exp(\alpha_j\ nxct_m(j0.5)))}{\sum_{m=1}^{N1}(exp(\alpha_j\ nxct_m(j0.1)))} = 0.5. \tag{3.5.5}$$

The value of α_j is found by assuming some initial value and then iterating until (3.5.5) is approximately satisfied. Then for each class the weight vector, $w_{j0.1}$, of $tfc'_m(j) \geq 0.1$ is determined as

$$w_{j0.1} = exp(\alpha_j\ nxct(j0.1)). \tag{3.5.6}$$

Now that the weight vectors, $w_{j0.1}$, for different classes have been determined, test documents can be operated upon by this algorithm so that classification performance of the algorithm can be assessed.

When a document d' from the test collection is considered for classification, words from class c_1 with $tfc'_m(j) \leq 0.1$ are checked for their presence or absence in the test document.

A vector of term frequency is formed with $tf_i(d'_1)$ indicating the presence and 0 indicating the absence of the word in the test document. Let

$$xt(d'_1) = [tf_i(d'_1)\ \ 0\ \ 0\ \ \ldots\ldots\ tf_N(d'_1)]^T, \tag{3.5.7}$$

be the vector corresponding to the test document for the first class c_1. This means that the first word corresponding to the vector $nxct(j0.1), j = 1$, is present in the test document where as the second and the third words corresponding to the vector $nxct(j0.1), j = 1$, are absent in the test document. The same process is repeated for all the classes $c_j, 1 \leq j \leq J$ and corresponding vectors, $xt(d'_j)$, are determined. Inner product of each of these vectors $xt(d'_j)$ is taken with the weight vector $w_{j0.1}$ calculated for the corresponding class as in (3.5.6)

$$cls_j =< xt(d'_j), w_{j0.1} > . \tag{3.5.8}$$

The decision on the class to which the given test document belongs is given by

$$H_{Algt_2}(d') = \overset{argmax}{j \in J}\ cls_j. \tag{3.5.9}$$

Chapter 4

Simulation Results

4.1 Data Set

Several experiments have been performed to assess the importance of class document frequency in document classification. The document sets considered are from Reuters-21578 text categorization test collection [13], Cora data set [14] and Citeseer data set [14]. Preprocessing for Reuters 21578 has been done by us to obtain the features described in Chapter 2. Data from Cora dataset and Citeseer dataset have already been processed. Performances of the algorithms NB, PrTFIDF, KNN, SVM, Algd_1, Algd_2, Algt_1, Algt_2 on the features described in chapter 2 have been assessed using these three different data sets.

4.1.1 Reuters-21578

The data set considered is from the Reuters-21578 text categorization test collection . In this collection documents are marked up with SGML (Standard generalized mark up language) tags, and a corresponding SGML DTD (Document Type Definition) is produced, so that the boundaries of important sections of documents are unambiguous. There are multiple categories with the categories overlapping each other and most plausible feature/example matrices being large and sparse. The Reuters-21578 collection consists of 22 files. Each of the first 21 files (reut2-000.sgm through reut2-020.sgm) contains 1000 documents and the last (reut2-021.sgm) contains 578 documents and hence the name Reuters

21578. Each of the 22 files begins with a document type declaration line:

$$<!DOCTYPE\ lewis\ SYSTEM\ "lewis.dtd" > .$$

Following the document type declaration line are individual Reuters articles marked up with SGML tags, as described below.

Each article starts with an "open tag" of the form

$$< REUTERSTOPICS =??LEWISSPLIT =??CGISPLIT =??$$

$$OLDID =??NEWID =?? >,$$

where the ?? are filled in an appropriate fashion. Each article ends with a "close tag" of the form

$$< /REUTERS > .$$

In all cases the

$$< REUTERS > \ and \ < /REUTERS >$$

tags are the only items on their line. Each REUTERS tag contains explicit specifications of the values of five attributes:

$$TOPICS,\ LEWISSPLIT,\ CGISPLIT,\ OLDID,\ NEWID.$$

These attributes are meant to identify documents and groups of documents and are used to define training set splits. The attribute $TOPICS$ gives information about whether or not the document had topic. The attribute $LEWISSPLIT$ gives information about whether or not the document had been used in training or test set in the old Reuters collection. The attribute $CGISSPLIT$ gives information about whether or not the document was in training set or test set for the experiments reported in HAYES89. $OLDID$ gives information about the identification number the document has in old collection and $NEWID$ gives information about the identification number the document has in the Reuters 21578.

Just as the $< REUTERS > < /REUTERS >$ tag serves to delimit documents within a file, other tags are used to delimit elements within a document. $< DATE > < /DATE >$ encloses date and time of the document, $< MKNOTE > < /MKNOTE >$ notes on certain hand corrections that were done to the old Reuters collection. $< TOPICS >$

< /TOPICS > encloses the list of TOPICS categories, if any, for the document. If TOPICS categories are present, each will be delimited by the tag < D >< /D >. < PLACES >< /PLACES > encloses the list of places, < PEOPLE >< /PEOPLE > encloses the list of people, and similarly for the tags, < ORGS >< /ORGS >, < COMPANIES >< /COMPANIES >, < UNKNOWN >< /UNKNOWN >. < TEXT >< /TEXT > delimits all the textual material of each story and < AUTHOR > < /AUTHOR > gives information about the author of the story. < DATELINE >< /DATELINE > gives information about the location of the story, < TITLE >< /TITLE > encloses the title of the story and finally < BODY >< /BODY > encloses the main text of the story.

Reuters 21578 has five different sets of content related categories. The category EXCHANGES has 39 subcategories. The category ORGS has 56 subcategories. The category PEOPLE has 267 subcategories. The category PlACES has 175 subcategories. The category TOPICS has 135 subcategories. The TOPICS subcategories have been used most frequently in previous research with Reuters 21578. The ten TOPICS subcategories used in this thesis are: earn, acq, money-fx, grain, crude, trade, interest, ship, corn and wheat.

A small sample of the Reuters 21578 collection is shown below.

```
<!DOCTYPE lewis SYSTEM "lewis.dtd">
<REUTERS TOPICS="NO" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET"
OLDID="16321" NEWID="1001">
<DATE> 3-MAR-1987 09:18:21.26</DATE>
<TOPICS></TOPICS>
<PLACES><D>usa</D><D>ussr</D></PLACES>
<PEOPLE></PEOPLE>
<ORGS></ORGS>
<EXCHANGES></EXCHANGES>
<COMPANIES></COMPANIES>
<UNKNOWN>
&#5;&#5;&#5;G T
```

&#22;&#22;&#1;f0288&#31;reute
d f BC-SANDOZ-PLANS-WEEDKILL 03-03 0095</UNKNOWN>
<TEXT>&#2;
<TITLE>SANDOZ PLANS WEEDKILLER JOINT VENTURE IN USSR</TITLE>
<DATELINE> BASLE, March 3 - </DATELINE><BODY>Sandoz AG said it planned a joint venture to produce herbicides in the Soviet Union. The company said it had signed a letter of intent with the Soviet Ministry of Fertiliser Production to form the first foreign joint venture the ministry had undertaken since the Soviet Union allowed Western firms to enter into joint ventures two months ago. The ministry and Sandoz will each have a 50 pct stake, but a company spokeswoman was unable to give details of the size of investment or planned output. Reuter
&#3;<BODY></TEXT>
</REUTERS>

Frequent words that do not contribute much towards text classification such as 'a', 'the', 'has', 'have', 'is', 'was', 'are', 'were', 'had' etc are removed from the text corpus. There are many morphologically related words with common root. Stemming algorithm [31] can be used to reduce these words to their common base. However, the benefits are not very obvious and so word stemming is not done for this data set. The documents in this data set contain hundred and ten words on average. The plot given below shows the distribution of the words in different classes in terms of class document frequency. This shows that the data vectors are sparse.

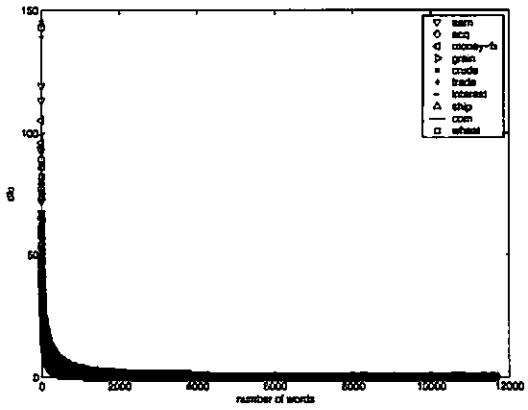


Figure 4.1: Class document frequency of words for different classes

4.1.2 Cora Data Set

The Cora data set consists of machine learning papers. These papers are classified into one of the following seven classes

1.Case Based

2.Genetic Algorithms

3.Neural Networks

4.Probabilistic Methods

5.Reinforcement Learning

6.Rule Learning

7.Theory

There are 2708 papers in the whole corpus. After stemming and removing stop-words, vocabulary size is of 1433 unique words. The documents in this data set contain ninety words on average. The data set contains files related to word attributes and link attributes. The files related to word attributes are represented in the following format:

$< paper\ identification\ tag > + < word\ attributes > + < class\ label >$.

The first entry in each line contains the unique string identification of the paper followed by binary values indicating whether each word in the vocabulary is present (indicated by 1) or absent (indicated by 0) in the paper. Finally, the last entry in the line contains the class label of the paper. The file related to link attributes is represented as citation graph of the

corpus. Each line describes a link in the following format:

$< Identification\ tag\ of\ cited\ paper >< Identification\ tag\ of\ citing\ paper >$.

Each line contains two paper identification tags. The first entry is the identification tag of the paper being cited and the second entry is that of paper which contains the citation. The direction of the link is from right to left. If a line is represented by "paper1 paper2" then the link is "$paper2 \Rightarrow paper1$" meaning "paper2 cites paper1".

The plot below shows the distribution of words in different classes in terms of class document frequency. This shows that the data vectors are sparse.

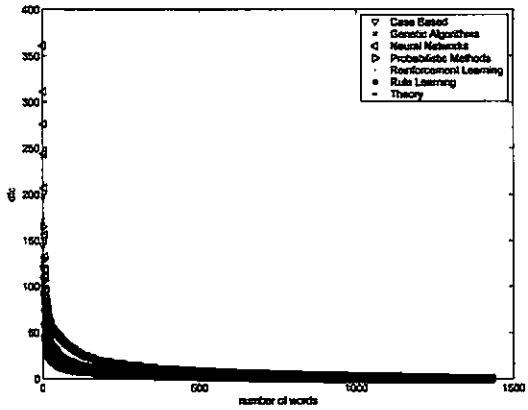


Figure 4.2: Class document frequency of words for different classes

4.1.3 Citeseer Data Set

Citeseer data set contains papers classified into one of the following six classes:

1. Agents
2. Artificial Intelligence
3. Data Base
4. Information Retrieval
5. Machine Learning
6. Human Computer Interface

The papers have been selected in such a way that in the final set every paper cites or is cited by at least one other paper. There are 3312 papers in the whole data set. After

stemming and removing stop words, the vocabulary has 3703 unique words. All words with document frequency less than 10 have been removed. The documents in this data set contain 120 words on average. The data set contains files related to word attributes and link attributes. The files related to word attributes are represented in the following format:
$< paper\ identification\ tag > + < word\ attributes > + < class\ label >$.
The first entry in each line contains the unique string ID of the paper followed by binary values indicating whether each word in the vocabulary is present (indicated by 1) or absent (indicated by 0) in the paper. Finally, the last entry in the line contains the class label of the paper. The file related to link attributes is represented as citation graph of the data set. Each line describes a link in the following format:
$< Identification\ tag\ of\ cited\ paper >< Identification\ tag\ of\ citing\ paper >$.
Each line contains two paper identification tags. The first entry is the identification tag of paper being cited and the second entry is that of paper which contains the citation. The direction of the link is from right to left. If a line is represented by "paper1 paper2" then the link is "$paper2 \Rightarrow paper1$" meaning "paper2 cites paper1".

The plot below shows the distribution of words in different classes in terms of class document frequency. This shows that the data vectors are sparse.

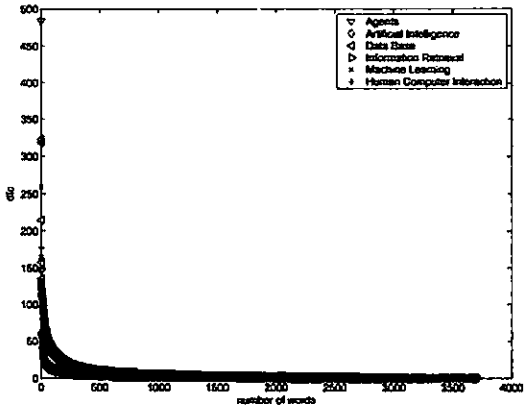


Figure 4.3: Class document frequency of words for different classes

4.2 Performance Evaluation

There are different performance evaluation measures used in text categorization [7][24] based on number of correctly and incorrectly classified documents. In this thesis accuracy(Ac), f-measure($F1$) and break-even point(BEP) have been used as performance measures for text categorization algorithms.

Let true positives, tp_j, with respect to the class c_j, be the number of the correctly classified documents belonging to class c_j. Let false positives, fp_j with respect to class c_j be the number of documents falsely classified as belonging to the class c_j. Let true negatives, tn_j, with respect to the class c_j be the number of documents correctly classified as not belonging to the class c_j. Let false negatives, fn_j, with respect to the class c_j be the number of documents falsely classified as not belonging to the class c_j. Then accuracy, Ac, of the classifier with respect to the class c_j is the ratio of number of documents correctly classified as belonging and not belonging to the class c_j and the total number of documents provided for classification, which is given by

$$Ac = \frac{tp_j + tn_j}{tp_j + tn_j + fp_j + fn_j}. \tag{4.2.1}$$

Precision of the classifier with respect to the class c_j is the ratio of number of correctly classified documents belonging to the class c_j to the total number of documents classified as belonging to c_j and is given by

$$p_j = \frac{tp_j}{tp_j + fp_j}. \tag{4.2.2}$$

Recall of the classifier with respect to the class c_j is the ratio of number of correctly classified documents belonging to class c_j to the total number of documents actually in the class c_j and is given by

$$r_j = \frac{tp_j}{tp_j + fn_j}. \tag{4.2.3}$$

There are different ways of combining these measures for classes into a single global measure. In this thesis micro averaging and macro averaging are used as a global measures.

In micro averaging each class proportionally contributes to the global measure and so the corresponding precision and recall are respectively given by,

$$p_{mic-avg} = \frac{\sum_{j=1}^{J} tp_j}{\sum_{j=1}^{J}(tp_j + fp_j)} \tag{4.2.4}$$

$$r_{mic-avg} = \frac{\sum_{j=1}^{J} tp_j}{\sum_{j=1}^{J}(tp_j + fn_j)}. \tag{4.2.5}$$

In macro averaging each class contributes equally to the global measure and so the corresponding precision and recall are respectively given by,

$$p_{mac-avg} = \frac{\sum_{j=1}^{J} p_j}{J} \tag{4.2.6}$$

$$r_{mac-avg} = \frac{\sum_{j=1}^{J} r_j}{J}. \tag{4.2.7}$$

f-measure($F1$), is the harmonic mean of the precision and recall and is given by

$$F1 = \frac{2\,p\,r}{p + r}. \tag{4.2.8}$$

Break-even point(BEP) is that value for which precision equals recall and is given by

$$BEP = \frac{p + r}{2}. \tag{4.2.9}$$

Here

$$[p, r] \in \{[p_j, r_j], [p_{mic-avg}, r_{mic-avg}], [p_{mac-avg}, r_{mac-avg}]\}. \tag{4.2.10}$$

4.3 Results

Reuters 21578						
Category	NB	PrTFIDF	KNN	SVM	Algt_1	Algt_2
earn	68.54	95.72	84.33	89	96.6	94.64
acq	82.7	96.7	92	88	94.53	86.84
money-fx	86	90	88.3	93	89.7	89
grain	87	89.4	87.5	97	88.2	88
crude	85	94	91.3	90	93.8	93.6
trade	77	87	88	91.3	80.1	84.8
interest	72.7	92.5	86.4	93	90	87.2
ship	87	93.3	92	94.9	93.7	85.6
corn	88	92	90.8	97	91.4	95.5
wheat	92.67	92	89.4	96	93	93.3

Table 4.1: *Ac* of algorithms with term frequency as features for representation of documents

Reuters 21578						
Category	ModNB	ModPrDF	KNN	SVM	Algd_1	Algd_2
earn	77.12	95.44	96	88.47	95.35	95
acq	81	96.6	96.3	87.4	95.2	93.3
money-fx	84	91	88.6	92.7	89.7	88
grain	86	88.8	87.9	98.7	88.8	87
crude	84	95.5	93	90	95	93
trade	84	85.7	82.1	93.5	73	93.1
interest	71	93.8	89.4	94	92.9	88
ship	84.1	95.3	92.3	96	94	94.5
corn	93.3	92.7	90.5	97.6	94.1	96
wheat	92.4	92	92.6	96.6	93.6	91

Table 4.2: *Ac* of algorithms with class document frequency as features for representation of documents

Reuters 21578

Category	NB	PrTFIDF	KNN	SVM	Algt_1	Algt_2
earn	26.25	75.86	53.82	88.89	80.75	75
acq	8.45	84.74	57.78	88.12	79.57	53.89
money-fx	14.74	49.12	43.48	65.85	24.52	25.15
grain	0	10.81	12.6	30.51	15.19	18.71
crude	11	76.03	47.9	79.42	77.17	75.09
trade	13.77	56.65	54.31	83.19	49.44	48.5
interest	6.45	50.29	54.05	68.12	60.58	50.17
ship	1.34	66.06	48.92	67	26.8	18.09
corn	1.4	31.82	22.22	45	31.94	31.58
wheat	6.67	24.56	34.72	46.43	43.08	46.9

Table 4.3: $F1$ of algorithms with term frequency as features for representation of documents

Reuters 21578

Category	ModNB	ModPrDF	KNN	SVM	Algd_1	Algd_2
earn	37.07	78.88	80.24	89.58	78.57	80
acq	17.49	86.84	83.77	91.86	80.56	74.36
money-fx	18.02	51.21	39.05	65.06	28.22	42.91
grain	8.09	2.9	23.02	17.83	26.29	18.78
crude	18.5	78.75	73.97	81.53	77.30	77.66
trade	13.02	56.91	54.60	86.67	42.48	70.85
interest	14.95	68.72	63.28	65.45	64.32	32.54
ship	2.25	70.97	30.61	72.82	31.68	45.76
corn	9.41	37.24	36.36	51.40	33.66	28.13
wheat	10.31	39.42	32.99	46.55	46.72	54.55

Table 4.4: $F1$ of algorithms with class document frequency as features for representation of documents

Reuters 21578

Category	NB	PrTFIDF	KNN	SVM	Algt_1	Algt_2
earn	30.44	77.59	60.65	89.40	82.36	78.73
acq	10.24	85.1	58.5	88.47	80.11	54.39
money-fx	17.72	50.53	43.81	66.29	52.33	39.65
grain	0	31.56	18.81	41.47	30.53	28.15
crude	14.87	76.08	49.85	79.97	77.24	75.16
trade	14.71	64.39	55.36	83.41	64.86	53.94
interest	7.65	51.71	57.01	68.94	64.56	54.85
ship	1.35	71.91	49.85	70.09	46.36	18.81
corn	1.2	35.8	23.01	46.88	37.17	32.69
wheat	8.31	24.94	35.16	48.66	43.08	47.4

Table 4.5: BEP of algorithms with term frequency as features for representation of documents

Reuters 21578

Category	ModNB	ModPrDF	KNN	SVM	Algd_1	Algd_2
earn	39.84	82.28	83.5	89.74	81.79	82.14
acq	17.88	86.85	83.96	91.9	80.75	74.44
money-fx	18.83	56.08	40.41	65.36	48.24	43.23
grain	11.8	34.07	28.51	35.66	37.27	24.77
crude	20.77	78.96	74.53	81.70	77.7	80.79
trade	13.07	67.35	66.65	86.96	63.08	72.68
interest	19.17	69.32	66.46	65.8	64.87	32.54
ship	2.27	72.38	48.97	75.54	48.1	52.40
corn	9.44	43.5	40.91	55.94	34.07	33.68
wheat	11.66	39.52	36.74	49.43	46.84	65.51

Table 4.6: BEP of algorithms with class document frequency as features for representation of documents

Reuters 21578						
Measure	NB	PrTFIDF	KNN	SVM	Algt_1	Algt_2
$F1_{mic-avg}$	11.89	59	48.3	73	54	57
$BEP_{mic-avg}$	11.89	59	48.3	73	54	57
$F1_{mac-avg}$	9	52.6	45	66.3	48.4	52.8
$BEP_{mac-avg}$	10.5	57	47.5	68.4	56.2	54.3

Table 4.7: Global performance measure of algorithms for term frequency as representation of documents

Reuters 21578						
Measure	ModNB	ModPrDF	KNN	SVM	Algd_1	Algd_2
$F1_{mic-avg}$	17.9	63	56	72	57.5	62
$BEP_{mic-avg}$	17.9	63	56	72	57.5	62
$F1_{mac-avg}$	15	57.2	51.7	68.1	51	55.7
$BEP_{mac-avg}$	16.5	63	56.1	70.6	57.2	60.5

Table 4.8: Global performance measure of algorithms for class document frequency as representation of documents

Cora Data Set

Category	SVM (words)	SVM (words+tf_link)	SVM (words+dfc_link)
Case_Based	93.27	95.80	95.92
Genetic_Algorithms	94.88	97.66	97.70
Neural_Networks	85.30	92	92.09
Probabilistic_Methods	91.24	96.18	95.73
Reinforcement_Learning	95.51	97.32	97.21
Rule_Learning	95.54	97.33	97.36
Theory	90.72	94.06	93.91

Table 4.9: Ac of SVM with dfc, $dfc + tf\_link$ and $dfc + dfc\_link$

Cora Data Set

Category	SVM (words)	SVM (words+tf_link)	SVM (words+dfc_link)
Case_Based	71.70	79.47	80.11
Genetic_Algorithms	87.22	92.66	92.83
Neural_Networks	81.22	88.27	86.95
Probabilistic_Methods	77.82	87.95	86.64
Reinforcement_Learning	75.38	83.03	82.15
Rule_Learning	70.88	81.35	81.48
Theory	67.04	76.01	75.92

Table 4.10: $F1$ of SVM with dfc, $dfc + tf\_link$ and $dfc + dfc\_link$

Cora Data Set

Category	SVM (words)	SVM (words+tf_link)	SVM (words+dfc_link)
Case_Based	68	80.02	80.60
Genetic_Algorithms	83.35	92.72	92.92
Neural_Networks	76.88	88.28	86.96
Probabilistic_Methods	72.88	87.97	86.66
Reinforcement_Learning	70.64	83.07	82.26
Rule_Learning	65.37	81.66	81.77
Theory	62.82	76.05	75.93

Table 4.11: BEP of SVM with dfc, $dfc + tf\_link$ and $dfc + dfc\_link$

Cora Data Set

Measure	SVM (words)	SVM (words+tf_link)	SVM (words+dfc_link)
$F1_{mic-avg}$	73	86.5	86.3
$BEP_{mic-avg}$	73	86.5	86.3
$F1_{mac-avg}$	71.2	85.1	85.3
$BEP_{mac-avg}$	71.4	85.3	85.5

Table 4.12: Global performance measure of SVM with dfc, $dfc + tf\_link$ and $dfc + dfc\_link$

Citeseer Data Set

Category	SVM (words)	SVM (words+tf_link)	SVM (words+dfc_link)
Agents	89.15	90.81	91.15
Artificial Intelligence	90.24	91.02	91.29
Data Base	86.83	89.30	90.06
Information Retrieval	85.34	87.34	87.76
Machine Learning	85.34	86.55	87.10
Human Computer Interaction	89.12	92.35	92.75

Table 4.13: Ac of SVM with dfc, $dfc + tf\_link$ and $dfc + dfc\_link$

Citeseer Data Set

Category	SVM (words)	SVM (words+tf_link)	SVM (words+dfc_link)
Agents	70.69	75.08	75.91
Artificial Intelligence	28.58	33.37	34.60
Data Base	69.09	74.86	76.48
Information Retrieval	65.12	68.69	70.09
Machine Learning	57.39	61.87	63.45
Human Computer Interaction	63.02	74.67	76.08

Table 4.14: $F1$ of SVM with dfc, $dfc + tf\_link$ and $dfc + dfc\_link$

Citeseer Data Set

Category	SVM (words)	SVM (words+tf_link)	SVM (words+dfc_link)
Agents	71.13	75.64	76.42
Artificial Intelligence	29.06	34.38	35.48
Data Base	69.12	74.93	76.56
Information Retrieval	65.86	69.57	70.86
Machine Learning	58.28	62.48	64.13
Human Computer Interface	63.28	74.84	76.18

Table 4.15: BEP of SVM with dfc, $dfc + tf\_link$ and $dfc + dfc\_link$

Citeseer Data Set

Measure	SVM (words)	SVM (words+tf_link)	SVM (words+dfc_link)
$F1_{mic-avg}$	65.6	72.6	73.3
$BEP_{mic-avg}$	65.6	72.6	73.3
$F1_{mac-avg}$	60.3	66.7	67.4
$BEP_{mac-avg}$	61	67.7	68.2

Table 4.16: Global performance measure of SVM with dfc, $dfc + tf\_link$ and $dfc + dfc\_link$

Reuters 21578

Algorithms	NB	PrTFIDF	KNN	SVM	Algt_1	Algt_2
Time (secs)	530	1580	2200	1360	1211	413

Table 4.17: Computational time of algorithms with term frequency of words

Reuters 21578

Algorithms	ModNB	ModPrDF	KNN	SVM	Algd_1	Algd_2
Time (secs)	580	1400	991	1200	1211	423

Table 4.18: Computational time of algorithms with class document frequency of words

Cora Data Set

SVM	words	words+tf_link	words+dfc_link
Time (secs)	164	509	483

Table 4.19: Computational time of SVM with dfc, $dfc + tf\_link$ and $dfc + dfc\_link$

Citeseer Data Set

SVM	words	words+tf_link	words+dfc_link
Time (secs)	310	1338	1145

Table 4.20: Computational time of SVM with dfc, $dfc + tf\_link$ and $dfc + dfc\_link$

4.4 Discussion

The performances of different algorithms trained with term frequency and that of different algorithms trained with class document frequency are comparable as shown by the results. The performance in terms of accuracy for class document frequency shown by table 4.2 and for term frequency as shown by table 4.1 are comparable. The performances in terms of f-measure and break even point for class document frequency shown by table 4.4 and 4.6 and for term frequency shown by table 4.3 and 4.5 respectively are also comparable for all the categories. The global f-measure and global break even point for class document frequency are nearly the same as for term frequency as shown by tables 4.8 and 4.7 respectively. All these show that class document frequency is an important learned feature for learning algorithms. Also these tables show that algorithms, $Algd_1$ and $Algt_1$ perform comparably and algorithms $Algd_2$ and $Algt_2$ also perform comparably. This means that term frequency does not add much to the performance compared to the class document frequency. $Algd_2$ shows that even if words with low class document frequency are removed the performance is not much affected emphasizing the importance of high class document frequency.

The performances of SVM in terms of Ac, $F1$, BEP, $F1_{micro-avg}$, $BEP_{micro-avg}$, $F1_{macro-avg}$, and $BEP_{micro-avg}$ as shown by tables 4.9, 4.10, 4.11 and 4.12 respectively for Cora data set with class document frequency of words and link information in the document is better than that with class document frequency of words in the document only. The same is the case for Citeseer data set as shown by the tables 4.13, 4.14, 4.15 and 4.16 respectively. These tables clearly show that performances with term frequency of link and class document frequency of link are comparable towards the enhancement in categorization. This shows that class document frequency of words along with the class document frequency of the link further enhance the performance of learning algorithms.

The computational time of the algorithms as shown by tables 4.17 and 4.18 for term frequency and class document frequency respectively are comparable. Also these tables show that the computational time of $Algd_1$ and $Algd_2$ are less compared to the other algorithms. The algorithms $Algd_1$ and $Algd_2$ are simple in that they are based on inner product for classification of documents and so are computationally cheap compared to

other more complicated algorithms. Also the computational time of SVM on Cora data set and Citeseer data set for term frequency of link and class document frequency of link are comparable as shown by the tables 4.19 and 4.20 respectively. The algorithms have not been optimized to reduce the computation time depending on the features used.

Chapter 5

Conclusion

5.1 Summary

Machine learning algorithms for text categorization require some form of numerical feature representation of documents. These numerical features can be used to train the algorithms so that they can be used for classification of unknown documents. In our study of text classification literature we found a lack of proper mathematical representation for feature representation of documents. So chapter two discusses the mathematical feature representation of documents. In this representation, words are considered as features for representation of documents. In this model document is considered as vector of word attributes and class is also considered as vector of word attributes. The document with link is represented as vector of word attributes concatenated with vector of link attributes. Performances of algorithms have been assessed on term frequency and class document frequency of words and links in documents.

To assess the Performance of Naive Bayes on class document frequency we modified the Naive Bayes algorithm to obtain Modified Naive Bayes algorithm. We found that the performance of Modified Naive Bayes algorithm using class document frequency is comparable and to a certain extent better than that of Naive Bayes algorithm using term frequency. We also modified the Probabilistic Term Frequency Inverse Document Frequency algorithm using term frequency to Modified Probabilistic Document Frequency algorithm using class document frequency. We found comparable performances of these algorithms.

The performances of Support Vector Machines and K-Nearest-Neighbor algorithm are comparable for term frequency and class document frequency.

To further assess the importance of class document frequency, we developed two simple algorithms. $Algd_1$ is based directly upon the class document frequency of words whereas $Algd_2$ is based on class document frequency of words with high value. The performances of these algorithms are comparable to other machine learning algorithms indicating the importance of class document frequency and emphasizing the importance of words with high class document frequency. Similar to the above two algorithms, we developed the algorithms $Algt_1$ and $Algt_2$ based on the term frequency of the documents. The performances of $Algd_1$ and $Algt_1$ are comparable and the performances of $Algd_2$ and $Algt_2$ are also comparable indicating that term frequency does not contribute much to the classification of documents compared to class document frequency. The performance evaluations were done on Reuters 21578 data set.

For the linked document, the document vector of word attributes is augmented with link vector of link attributes. Term frequency of link and class document frequency of link are the two attributes used for the construction of link vectors. SVM was used for performance evaluation of these features on Cora data set and Citeseer data set respectively. It was found that the classification performance based on combination of word attributes and link attributes for document representation was better than the one based only on word attributes for document representation. More importantly, it was found that the class document frequency of link and term frequency of link have comparable contribution towards classification performance indicating that term frequency of link does not add much to the classification performance compared to class document frequency of the link. This verifies the importance of class document frequency.

5.2 Importance of Class Document Frequency

It is clear from the results that the learning algorithms we considered perform comparably whether they are trained with term frequency or class document frequency. The reason for this is, term frequency implicitly gives information on class document frequency as a word present in different documents during the training of machine learning

algorithms. This is the reason that the performance of SVM does not degrade when the documents are represented just as binary vector because then, the only feature that SVM is learning is the class document frequency. The class document frequency is the underlying important feature that machine learning algorithms learn and is used as discriminator for document categorization.

We also showed that two very simple classifiers, $Algd1$ and $Algd2$ based on dfc alone perform similarly compared to $Algt1$ and $Algt2$. Although not the best, yet the developed algorithm has performance comparable to that of other complicated learning algorithms.

When the class document frequency of the link is incorporated with that of the words there is further enhancement in the classification performance of the algorithm. This verifies that class document frequency is an important feature and should be the underlying learned feature during the training of machine learning algorithms for text categorization.

5.3 Further Research Directions

We only considered three data sets for performance evaluation of machine learning algorithms on the considered features. It would be important to know whether the results obtained on these data set are similar with that obtained with other data sets also. We used equal number of documents from each class for training and classification. Study could be done on performance changes of algorithms as the number of documents in different classes are made different. All the data sets considered by us are in English. Study on whether the features considered and results obtained are similar with data sets in other languages would provide further insight into the generality of these approaches.

It would be important to explore whether we can find practical applications for algorithms developed since they are simple and computationally less costly. It would be important to know whether the simple algorithms could be further developed so as to make them more competitive compared to powerful machine learning algorithms like SVM. The algorithms have not been optimized to reduce the computational time. It would be worth exploring whether optimization of algorithms would reduce the computational time of algorithms for class document frequency as compared to that for term frequency.

Finally, empirical results obtained from our simulations on the data sets show that class document frequency is an important learned feature. Analytical results verifying the empirical results would strengthen our claim and so is an important direction for further research.

Bibliography

[1] Gerald Salton, Christopher Buckley. "Term-weighting approaches in automatic text retrieval," *IPM*, 1988.

[2] Man Lan, Sam-Yuan Sung, Hwee-Boon Low, Chew-Lim Tan. "A comparative study on term weighting schemes for text categorization," *IJCNN*, 2005.

[3] Sam Scott, Stan Matwin. "Feature engineering for text classification," *ICML*, 1999.

[4] David D. Lewis. "Feature selection and feature extraction for text categorization," *Proceedings of Speech and Natural Language*, 1992.

[5] T. Joachims. "A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization," *ICML*, 1997.

[6] T. Joachims. "Text categorization with Support Vector Machines: learning with many relevant features," *ECML*, 1998.

[7] Yiming Yang, Xin Liu. "A re-examination of text categorization methods," *RDIR*, 1999.

[8] Eui-Hong(Sam) Han, George Karypis, Vipin Kumar. "Text categorization using weight adjusted k-Nearest Neighbor classification," *Lecture Notes in Computer Science*, 2001

[9] Christopher M. Bishop. *Pattern recognition and machine learning*, Springer, 2006.

[10] Thorsten Joachims. "Making large-scale SVM learning practical," *AKM*, 1999.

[11] Susan Dumais, John Platt, David Heckerman, Mehran Sahami. "Inductive learning algorithms for text categorization," *Proceedings of CIKM*, 1998.

[12] David D. Lewis. "An evaluation of phrasal and clustered representations on a text categorization task." *Proceedings of SIGIR*, 1992.

[13] http://www.daviddlewis.com/resources/testcollections/reuters21578/

[14] http://www.cs.umd.edu/projects/linqs/projects/lbc/index.html

[15] Chuong B. Do, Andrew Y. Ng. "Transfer learning for text classification," *NIPS*, 2005.

[16] Qing Lu, Lise Getoor. "Link-based text classification," *IJCAI Workshop on Text Mining and Link Analysis*, 2003.

[17] Steve Lawrence and C. Lee Giles. "Accessibility of information on the web," *Nature*, 1999.

[18] Kevin Chen-Chuan Chang, Junghoo Cho. "Accessing the Web: from search to integration," *ICMOD*, 2006.

[19] Xiao-Bing Xue and Zhi-Hua Zhou. "Distributional features for text categorization," *ECML*, 2006.

[20] J.R. Quinlan. "Induction of decision trees," *ML*, 1986.

[21] Robert E. Schapire, Yoram Singer. "BoosTexter: a boosting-based system for text categorization," *ML*, 2000.

[22] Robert E. Schapire. "Theoretical views of boosting and applications," *Proceedings of Algorithmic Learning Theory*, 1999.

[23] Vladimir N. Vapnik. *The nature of statistical learning theory*, Springer, 1995.

[24] Y. Yang. "An evaluation of statistical approaches to text categorization," *Technical Report CMU-CS-97-127, Carnegie Mellon University*, 1997.

[25] Thomas Leonard, John S. J. Hsu. *Bayesian methods: an analysis for statisticians and interdisciplinary researchers* , Cambridge University Press, 1999.

[26] Ciya Liao, Shamim Alpha, Paul Dixon. "Feature preparation in text categorization," *Oracle Corporation*, 2003.

[27] Gregory L. Wittel and Shyhtsun Felix Wu. "On attacking statistical spam filters," *CEAS*, 2004.

[28] Aron Culotta, Ron Bekkerman and Andrew McCallum. "Extracting social networks and contact information from email and the Web," *CEAS*, 2004.

[29] David D. Lewis. "Naive (Bayes) at forty: The independence assumption in information retrieval," *Conference proceedings of European Conference on Machine Learning*, 1998.

[30] Simon Haykin. *Neural Networks*, Prentice Hall, Inc, 1999.

[31] Hongyan Jing and Evelyne Tzoukermann. "Information retrieval based on context distance and morphology," *Research and Development in Information Retrieval*, 1999.

[32] Nello Cristianini. "Support Vector and Kernel Machines," *ICML*, 2001.

[33] http://webspace.ship.edu/thbrig/mexsvm/download.html