„Lucian Blaga" University of Sibiu
"Hermann Oberth" Engineering Faculty
Computer Science Department

Classification and clustering using SVM

2nd PhD Report

Thesis Title: "Data Mining for Unstructured Data"

Author:
Daniel MORARIU, MSc

PhD Supervisor:
Professor Lucian N. VINTAN

SIBIU, 2005

Contents

1 Introduction

While more and more textual information is available online, effective retrieval is difficult without good indexing and summarization of documents content. Document categorization is one solution to this problem and is the task of classifying natural language documents into a set of predefined categories. A growing number of classification methods and machine learning techniques have been applied in recent years.

Documents are typically represented as sparse vectors of the features space. Each word in the vocabulary represents a dimension of the feature space. The number of occurrences of a word in a document represents the value of the corresponding component in the document characteristic vector. This higher dimensionality of the feature space is a major problem of text categorization. The native feature space consists of the unique terms that occur in the documents, which can be tens or hundreds of thousands of terms for even a moderate-sized text collection. Much time and memory is needed for training a classifier on a large collection of documents. This is why we tray using various methods for reducing the feature space and the response time. As we will see the results are better when we work with a lower dimension of the space. As the space grows the accuracy of the classifier doesn't grow significantly, actually it decreases when we work with a higher dimensional feature space.

This report is a comparative study of feature selection methods (Information Gain, Mutual Information and Support Vector Machine) and types of input data representation in statistical learning of text categorization. Also I will present a technique used with great success in the last years in classification problems for nonlinear separable input data. I will present the application for processing documents and creating the vector of features. I will continue by presenting the application implemented for the classification and clustering parts using techniques based on support vectors and kernels.

I have used Text Mining like an application of data mining techniques to extract the signature of each document (the features vector). Starting with a set of d documents and t terms (words belonging to documents), we can model each document as a vector v in the t dimensional space $\Re^t$. In the classification phase, I have used Support Vector Machine that is a powerful technique for nonlinear separable input sets. A great advantage of this technique is that it can use large input sets. Thus we can easily test the number of features influence on the classification and clustering accuracy. I implemented this classification for two types of kernels: polynomial kernel and Gaussian kernel (Radial Basis Function - RBF). I will present results for two class classification and for multi-class classification using SVM technique. For two class classification I took into account only documents in one class versus the rest of the documents from the set. For multi-class classification I repeated two class classification for each topic (the category where the document is classified) obtaining more decision functions. I have also modified this technique so that it can be used as a method of features selection in the text mining step. I will present a graphical visualization of classification and clustering results using this method.

I will use different types of kernels and different types of input data representation trying to find the best parameters for better classification accuracy. I tried to find a simplified form of the kernels, without reducing the performance, actually increasing it, using more intuitive parameters.

Input data are represented in different formats, and I analyzed the influence of those representations on kernel type. I have used three types of representation. *Binary format* where the attributes are represented using values "0" or "1" ("0" if the word doesn't occur in the document

and "1" if it occurs, without being interested in the number of occurrences). The second format is *Nominal format* where the attributes store the number of occurrences of the word in the frequency vector, normalized with normal norm. The last format used is *Connell SMART system* where the attribute store the number of occurrences of the word in the frequency vector, normalized by another formula.

Section 2 describes the term selection method and the details of constructing the training and testing datasets that are used in the report. I will give a brief overview of feature selection methods in the context of proposed training strategy. In Section 3 I will describe the classifier and clustering algorithm based on Support Vector Machine technique. Also I will present details of implementation for those algorithms. In Section 4 we illustrate how to use the application to improve accuracy of classifiers and parameters of the application. I will give a description of the experiments in which we compare the effectiveness of the presented methods and I will present the results of the experiment. In the final section, I will present concluding remarks and future work.

Acknowledgments

Besides my parents there are a lot of people that deserve my gratitude. I can not mention them all here but I want to thank all of them. I would like to thank a few people, some of them were my teachers, that had guided me in this project. First of all I would like to express my sincere gratitude to my PhD supervisor Professor Lucian VINȚAN for his responsible scientific coordination, for providing stimulating discussions focused on my PhD work and for all his valuable support. I would also like to thank the ones that guided me from the beginning of my PhD studies: Prof. Ioana MOISIL, Prof. Boldur BĂRBAT, prof. Daniel VOLOVICI, Dr. Dorin SIMA and Dr. Macarie BREAZU for their valuable generous professional support.

I would also like to thank SIEMENS AG, CT IC MUNCHEN, Germany, especially Vice-President Dr. h. c. mat. Hartmut RAFFLER, for his very useful professional suggestions and for the financial support that he have provided. I want to thank to my tutor from SIEMENS, Dr. Volker TRESP, Senior Principal Research Scientist in Neural Computation, for the scientific support provided and for his valuable guidance in this wide interesting domain of research. I also want to thank Dr. Kai Yu for his useful information in the development of my ideas. Last but not least, I want to thank all those who supported me in the preparation of this technical report.

2 Feature Extraction

In fact a substantial fraction of the available information is stored in text or document database which consist of a large collection of documents from various sources such as news articles, research papers, books, web pages, etc. Data stored in text format is considered semi-structured data in that they are neither completely unstructured nor completely structured, because a document may however contain a few structured fields such as title, authors, publication, data, etc. Unfortunately those fields usually are not filled in, the majority of people don't loose time to complete these fields. Some researchers suggest that the information needs to be organized during the creation time, using some planning rules. This is pointless because most of people will not respect them. This is considered a characteristic of unordered egalitarianism of Internet [Der00]. Any attempt to apply the same organizing rules will determine the users to leave. The result for the time being is that most information needs to be organized after it was generated and searching and organizing tools need to work together.

Typically, only a small fraction of many available documents will be relevant to a given individual user. Without knowing what could be in the document, it is difficult to formulate effective queries for analyzing and extracting useful information from the data. Users need tools to compare different documents, rank the importance and relevance of the documents. Thus text data mining has become increasingly important. Text mining goes one step beyond keyword-based and similarity-based information retrieval and discovers knowledge from semi-structured text data using methods such as keyword-based association and document classification. In this context traditional information retrieval techniques become inadequate for the increasingly vast amounts of text data. Information retrieval is concerned with the organizing and retrieval of information from a large number of text-based documents. Most information retrieval systems accept *keyword-based* and/or *similarity-based* retrieval [Cro99]. Those ideas were presented in detailed in my first PhD technical report in section 2.2 [Mor05]. I only want to give a brief theoretical background as a support for the text mining step presentation. The text mining step is the first step of my application.

In keyword-based information retrieval system, a document is represented by a string, which can be identified by a set of keywords. Similarity-based retrieval system finds similar documents based on a set of common keywords. The output of such retrieval should be based on the *degree of relevance*, where relevance is measured based on the closeness of the keywords in the document, actually the relative frequency of the keyword.

In modern information retrieval systems, keywords for document representation are automatically extracted from the document. Normally, this implies removing high frequency words (stopwords), stripping the suffixes, and detecting equivalent stems. After determining the document representation in this manner, each term is assigned a weight. In the literature this method of document representation is called "bag-of-words" [Cha00].

Let's consider a set of d documents and a set of t terms for modeling information retrieval. We can model each of the documents as a vector v in the t dimensional space. The i-th coordinate of v (v_i) is a number that measures the association of the i-th term with respect to the given document: it is generally defined as 0 if the document does not contain the term, and nonzero otherwise. There are many ways to define the term-weighing for the nonzero entries in such a vector [Mit97]. For example, it can be simply defined $v_i = 1$ as long as the i-th term occurs in the document, or let v_i be the term frequency, or normalized term frequency. Term frequency is the number of occurrences of the i-th term in the document. Normalized term frequency is term frequency divided by the total

number of occurrences of all terms in the document. I have tested various term-weights in my application. Once the terms representing the documents and their weights are determined, we can form a document-term matrix for the entire set of documents. This matrix can be used to compute the pair-wise dependences of terms. A reasonable measure to compute those dependences is *odds ration* [Mla99], *information gain*, *mutual information* [Mla99], or *support vector machine*[Mla02]. There are some problems in determining term dependences based on document representation [Bha00]:

- *Too many terms in description.* The total number of distinct terms is quite large even for a small collection. A large ratio of these terms, when intellectually evaluated, seems irrelevant for the description of the document.
- *Inability to expand query with good quality terms.* It is well known that good quality terms are those that are more prevalent in relevant documents. Dependence base on all documents may not help in adding good quality terms to the query.
- *Mismatch between query and document terms.* Usually users' vocabulary differs from that of authors or indexes. This leads in some query terms not being assigned to any of the documents. Such terms will not form a node in the dependence tree construct by the earlier studies.

2.1 The Dataset

For experiments I used the Reuters-2000 collection [Reu2000], which includes a total of 806791 documents, with all news stories published by Reuters Press covering the period from 20th August 1996 through 19th August 1997. Each news is stored as an XML file (`xml version="1.0" encoding="UTF-8")`. Files are grouped by date in 365 zip files. All that files are grouped according to three criteria. According to the industry criterion the articles are grouped in 870 categories. According to the region the article referrers to there are 366 categories and according to topics there are 160 distinct topics. The definition of the metadata element of Reuters Experimental NewML allows the attachment of systematically organized information about the news' summary. The metadata contains information about date when the article was published, language of the article, title, place, etc. The title is marked using title markups like <title> </title>. Then the text of the article follows marked by <text> and </text>. After the content of the article is a structure that is based on a scheme for "Reuters Web" that are in the following format:

```
<metadata>
<codes class="bip:countries:1.0">…</code>
<codes class="bip:topics:1.0">…</code>
<codes class="bip:industry:1.0">…</code>
<!ENTITY % dc.elements

"(dc.title |

dc.creator.name |
dc.creator.title |
dc.creator.location |
dc.creator.location.city |
dc.creator.location.sublocation |
dc.creator.location.stateOrProvince |
dc.creator.location.country.code |
dc.creator.location.country.name |
dc.creator.phone |
dc.creator.email |
dc.creator.program |

dc.date.created |
dc.date.lastModified |
```

```
dc.date.converted |

dc.publisher |
dc.publisher.provider |
dc.publisher.contact.name  |
dc.publisher.contact.email  |
dc.publisher.contact.phone  |
dc.publisher.contact.title  |
dc.publisher.location |
dc.publisher.graphic  |

dc.coverage.start      |
dc.coverage.end        |
dc.coverage.period     |

dc.relation.obsoletes  |
dc.relation.includes   |
dc.relation.references |

dc.date.published |
dc.date.live |
dc.date.statuschanges |
dc.date.expires |

dc.source |
dc.source.contact   |
dc.source.provider  |
dc.source.location  |
dc.source.graphic        |
dc.source.date.published |
dc.source.identifier  |

dc.contributor.editor.name |
dc.contributor.captionwriter.name )">
</metadata>
</newsitem>
```

In the entries above there is information included on region, topic and industry. This information is included using codes. There are separate files where it can be found the correspondence between the codes and the complete name. In my application I used only the name of the news stories, the content of the news, topic proposed by Reuters for classifying and industry. Thus from each file I extract the words from the title and the content of the news and I create a vector that characterizes that document.

A text retrieval system often associates a stop list with a set of documents. A stop list is a set of words that are deemed "irrelevant" for a set of documents [Jia01] [Ian00]. In my application I have used a general stopwords list from the package *ir.jar* [IR] from Texas University. I wanted to use a general list in order to eliminate the non-relevant words. In the stopword list there are 509 different words included which are considered to be irrelevant for the content of the text.

For each word that remained after the elimination of stopwords I have extracted the root of the word (stemming). If after stemming I obtained the root with dimension smaller or equal to 2 (the word obtained has only two or one characters) and the original dimension of the word was 2 or 1 I eliminate those words. If the original dimension was greater than three I will kept the word in the original format without stemming. Up to this moment I didn't deal with the case in which two different words have the same root after stemming. Afterwards I counted the occurrences of every remaining root. Thus I have created an array for each document from Reuters with the remaining roots (called later tokens). This array is the individual characterization of each document.

For example a vector that characterizes small news is represented in the following format:

"24 : singapor:3 pore:1 middl:4 distil:4 stock:4 highest:2 apr:1 weekli:1 april:2 trade:1 develop:1 board:1 statist:1 show:1 thursdai:1 week:2 end:2 august:1 regist:1 on:2 barrel:2 newsroom:1 - c21:1 ccat:1 m14:1 m143:1 mcat:1"

Number 24 represent the number of the indexed document. A root and the number of its occurrences are separated by a ":"(in document can occur words in different context but have same steam). The "-" symbol at the end of the line introduces Reuters classification topics.

This representation of documents is called in the literature bag-of-words approach. For training and testing data in two classes I build a subset of data here referred to as Subset-c152. From all 806791 documents I select those documents that are grouped by Reuters in "System Software" (I330202) as industry code. After this selection I obtained only 7083 documents. In the resulting set there are 63 different topics for classifying according to Reuters. For multiclass classification I used all 63 topics. For binary classification I chose topic "c152" that means "Comment /Forecasts" according to Reuters codes. I grouped those 7083 articles in training set and testing set randomly assuring that the training set is smaller than the testing set.

The algorithm that I will present is based on the fact that data is grouped only in two classes. This is in fact a binary classification algorithm where the labels can take only two values. Therefore in this phase I took in consideration only one class of documents and documents that are not belonging to that class are considered to be in another class. In the multiclass classification I take in consideration all topics and I learn separately for each topic using SVM "one versus the rest" technique. I chose samples that have a specific topic versus the rest of samples for each topic.

Afterwards I have created a large frequency vector with all unique tokens and the number of occurrences of each token in all documents (in order to further apply the SVM method). I use a vector with all tokens to memorize each document in the set. If a token appears in the document then I will store the number of occurrences in the new vector otherwise I will store "0" for that token (sparse vector [sparse]) . By doing so all vectors have the same size and the tokens are arranged in same order so that all data available is organized in the same way. For the Subset-c152 I have obtained 19038 different tokens.

2.2 Data Reduction

In the learning phase data needs to be stored in the memory in order to compute on it and so the learning time is considerably bigger. Due to the size of the token vector the accuracy of learning (as we will further see in this report) is diminished. I have applied some techniques to reduce the dimension of this large frequency vector. For doing so I have used the *Information Gain* [Yan97], Mutual Information [Cov91] and Support Vector Machine [NEL00][Dou04]. There is another type of methods used for features induction that automatically create a nonlinear combination of existing features and additional input features to improve classification accuracy like the method proposed by [Jim04]. All methods use the elimination of tokens which occurs less than the preordain threshold.

2.2.1 Entropy and Information Gain

Entropy and information gain are functions of the probability distribution that underlie the process of communications. The entropy is a measure of uncertainty of a random variable. Let X be a discrete random variable with alphabet S and probability mass function $p(x) = \Pr\{X = x\}, x \in S$. We denote the probability mass function by *p(x)* rather than $p_X(x)$ for convenience. More information about Entropy and Information Gain and a complete example I have presented in my first PhD report in section 2.1.1.1 [Mor05]. There I have used only two values for the attribute (attributes could take only values 0 and 1). Here I want to present another aspect in which the attributes can take more than just two values (the attributes can take any of the values in their domain).

Note that the entropy is a function of the distribution of X. It does not depend on the actual values taken by random variable X, but only on the probabilities. Thus if $X \sim p(x)$, then the expected value of the random variable g(X) is written

$$E_p g(X) = \sum_{x \in S} g(x) p(x) \qquad (2.1)$$

or more simply as Eg(x) when the probability mass function is understood from the context.

The entropy of X can also be interpreted as the expected value of $\log \frac{1}{p(X)}$, where X is drawn according to probability mass function *p(x)*. Thus

$$E(X) = E_p \log \frac{1}{p(X)} \qquad (2.2)$$

As seen this definition of entropy is related to definition of entropy in thermodynamics. It is possible to derive the definition of entropy axiomatically by defining certain properties that the entropy of a random variable must satisfy. The concept of entropy in information theory is closely connected with the concept of entropy in statistical mechanics. If we draw a sequence of n independent and identically distribution random variables, it can be shown that the probability of a "typical" sequence is about $2^{-nE(X)}$ and that there are about $2^{nE(X)}$ such "typical" sequences. This property (known as the asymptotic equation property) is the basis of many of the proofs in information theory.

In [For04] the author justified that Information Gain failed to produce good results on an industrial text classification problem. The author says that for a large class of features scoring methods suffers a pitfall: they can be blinded by a surplus of strongly predictive features for some classes, while largely ignoring features needed to discriminate difficult classes.

2.2.2 Mutual Information

Entropy is uncertainty of a single random variable. We can define conditional entropy, which is the entropy of a random variable, given another random variable. The reduction in uncertainty due to another random variable is called the mutual information. For two random variables X and Y this reduction is:

$$I(X;Y) = Entropy(X) - Entropy(X|Y) = \sum_{x,y} p(x,y) * \log \frac{p(x,y)}{p(x) * p(y)} \qquad (2.3)$$

The mutual information I(X,Y) is a measure of the dependence between the two random variables. It is symmetric in X and Y and always non-negative. Thus the mutual information I(X;Y) is the reduction in the uncertainty of X due to the knowledge of Y. According to the formula we can observe that the mutual information for a random variable with itself is the entropy of the random variable. This is the reason that the entropy is sometimes referred to as self-information. Between entropy and mutual information there is the following relationship presented in a Venn diagram (Figure 2.1). Notice that the mutual information I(X;Y) corresponds to the intersection of the information in X with the information in Y.

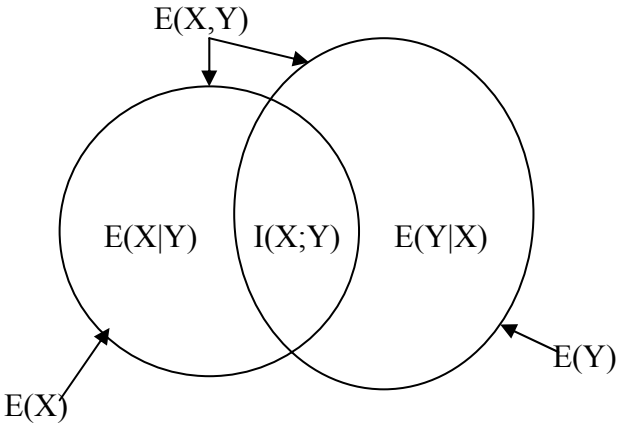


Figure 2.1. Relationship between entropy and mutual information

In my application I use another method for feature selection using the concept of Support Vector Machine that is presented in detail in the next chapter. For this I have used the linear kernel and I have taken in consideration only those attributes that have a weight greater than a certain threshold.

2.3 Training/Testing Files Structure

It is useful to eliminate the words that occur in all or many documents from the set because these words can't characterize the documents, they are like a stopword for this sets. I have considered that this new array characterizes all documents. After that I have modified the arrays of each document in order to have the same dimension as the larger frequency vector. In the new array there are entries equal to zero if the word doesn't occur in the document. We consider this to be the signature of each document in the documents set.

In the first phase the developed application produces the text mining on Reuter's files chosen as above. The large frequency vectors are stored in four different files after creation. Two of these files contain the date necessary for binary training and testing classification. The other two files contain data necessary for multiclass training and testing classification. The files structure format is presented below. This format is almost identical with the format presented by Ian in [Ian00] and used by Weaka application that can be found at [Weka]. The files have a part containing attributes, a part containing topics and a part containing data. In the attributes part there are specified all attributes (tokens) that characterize the frequency vectors. The attributes are specified using the letterhead *"@attribute*" followed by the name of the attribute. In the topic part there are specified all the topics for this set according to Reuter's classification. The topics are specified using the letterhead "@topics" followed by the name of the topic and the number of samples that contain that topic. The data section is marked using "@data" and contains all large frequency vectors followed by ":" and all topics for that entry, according to Reuter's classification. For the output

files where we have all the topics (for multi-class classification) the large frequency vectors are ordered by their occurrence in Reuter's files. For the output files where we have only one topic (for classification one versus the rest) the large frequency vectors are ordered by topic. Initially we put the large frequency vectors that belong to the class and after that we put the large frequency vectors that don't belong to the class.

The structure presented below is obtained as a result of text mining on Reuter's database. In this step I have used some classes taken from the package *ir.jar* [IR] from Texas University. In order to test the influence of feature selection on classification accuracy I used several levels of threshold for Information Gain, Mutual Information and SVM feature selection. I will present later the exact value of threshold and number of resulting attributes. The training and testing file need to have the same structure. This means that we need to have the same number of attributes and topics. Both attributes and topics have to be in the same order both in the training and testing file. As follows I will present a sample structure.

```
@attribute singapor
@attribute april
@attribute develop
@attribute week
@attribute stock
@attribute newsroom
@attribute apr
@attribute board
@attribute august
@attribute barrel
@attribute show
@attribute statist
@attribute thursdai
@attribute weekli

@topic c151 1
@topic c151 -1


@data
3,2,3,2,0,0,0,3,1,0,0,8,0,1:1
1,5,1,5,0,1,0,9,0,0,0,2,0,3:1
4,1,1,3,0,4,0,0,1,0,0,0,0,1:1
2,9,7,2,0,2,0,6,0,0,0,11,0,3:1
3,2,17,3,0,14,0,0,13,0,0,0,0,1:1
0,0,9,0,5,2,5,0,0,1,1,6,8,0:-1
3,0,2,0,1,2,0,2,0,5,1,0,2,0:-1
1,0,2,5,3,0,3,9,7,2,0,0,1,0:-1
2,0,0,4,1,0,6,0,0,2,0,0,1,0:-1
4,0,2,1,1,0,6,2,1,1,0,0,4,0:-1
3,0,1,3,2,0,2,0,0,4,0,0,1,0:-1
1,0,0,2,2,0,1,7,1,2,0,0,4,0:-1
```

3 Support Vector Machine in Classification/Clustering Problems

In this chapter I will present some theoretical aspects referring to Support Vector Machine technique and some aspects referring to the implementation of this technique for classifying and clustering documents. Classifying using SVM is a supervised learning technique that uses a labeled dataset for training and tries to find a decision function that classifies best the training data. This technique is based on classifying only in two classes. There are some methods used for classification in more that two classes. At the end of this chapter I will present some changes that need to be made in the classification algorithm, so that it can work with unlabeled documents (clustering).

3.1 SVM Technique for Binary Classification

Support Vector Machine (SVM) is a classification technique based on the statistical learning theory[SCK02], [NEL00]. It was successfully used a few centuries old mathematical discoveries to solve optimization problems on large sets as far as the numbers of articles and their characteristics (features) are concerned.

The purpose of the algorithm is to find a hyperplane (in a *n*-dimensional space the hyperplane is a space with the dimension *n-1*) that splits optimally the training set (a practical idea can be found in [Chi03]). Actually the algorithm consists in determining the parameters that determine the general equation of the plane. Looking at the two dimensional problem we actually want to find a line that "best" separates points in the positive class (points that are in the class) from points in the negative class (the remaining points). The hyperplane is characterized by a decision function like $f(x) = sgn(\langle w, \Phi(x) \rangle + b)$, where **w** is the weight vector orthogonal to the hyperplane, "*b*" is a scalar that represents the margin of the hyperplane, "*x*" is the current sample tested, "$\Phi(x)$" is a function that transforms the input data into a higher dimensional feature space and $\langle \cdot, \cdot \rangle$ representing the dot product. *Sgn* is the signum function that returns 1 if the value is greater than to 0 and -1 otherwise. If **w** has unit length, then $\langle \mathbf{w}, \Phi(x) \rangle$ is the length of $\Phi(x)$ along the direction of **w**. Generally **w** will be scaled by $\|\mathbf{w}\|$. In the training part the algorithm need to find the normal vector "**w**" that leads to the largest "*b*" of the hyperplane. For better understanding let's consider that we have data separated into two classes (circles and squares) as in Figure 3.1. The problem that we want to solve consists in finding the optimal line that separates those two classes.
The problem seems very easy to solve but we have to keep in mind that the optimal classification line should classify correctly all the elements generated by the same given distribution. There are a lot of hyperplanes that meet the classification requirements but the algorithm tries to determine the optimum.

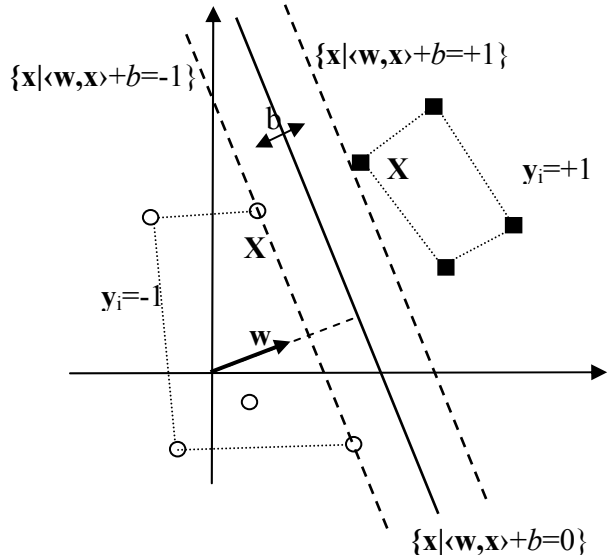


Figure 3.1 - The optimal hyperplane with normal vector **w** and offset *b*

Let $x,y \in \mathbb{R}^n$ where $x=(x_1, x_2, \ldots, x_n)$ and $y=(y_1, y_2, \ldots, y_n)$. We can define the dot product of two vectors as the real value which measures the area determined by the two vectors. We will denote this further on as <x,y> and we will compute it:

$$< x , y > = x_1 * y_1 + x_2 * y_2 + \ldots + x_n * y_n \qquad (3.1)$$

We say that *x* is orthogonal on *y* if their dot product equals to 0, that is if:

$$< x , y > = 0 \qquad (3.2)$$

if *w* has the norm equal to 1, then $<w, \Phi(x)>$ is equal to the length of $\Phi(x)$ along the direction of *w*. Generally *w* will be scaled by $\| w \|$ in order to obtain a unit vector. By $\| w \|$ we mean the norm of *w*, defined as $\| w \| = <w , w>$ and also called Euclid's norm. All through our presentation we will use normalized vectors.

This learning algorithm can be performed in a dot product space and for data which is separable by hyperplane, by constructing *f* from empirical data. It is based on two facts. First, among all the hyperplanes separating the data, there is a unique optimal hyperplane, distinguished by the maximum margin of separation between any training point and the hyperplane. Second, the capacity of the hyperplane to separate the classes decreases with the increasing of the margin. These are two antagonistic features which transform the solution of this problem into an exercise of compromises.

$$\underset{w \in H, b \in \Re}{\text{maximize}} \min \left\{ \left\| \mathbf{x} - \mathbf{x_i} \right\| \middle| \mathbf{x} \in \mathcal{H}, \left\langle \mathbf{w}, \mathbf{x} \right\rangle + b = 0, i = 1, \ldots, m \right\} \qquad (3.3)$$

For training data which is not separable by a hyperplane in the input space the idea of SVM is to map the training data into a higher-dimensional feature space via Φ, and construct a separating hyperplane with the maximum margin there. This yields a non-linear decision boundary in the input space. By the use of a kernel function $\langle \mathbf{w}, \phi(x) \rangle$ it is possible to compute the separating hyperplane without explicitly carrying out the map into the feature space [SCK02].

In order to find the optimal hyperplane, distinguished by the maximum margin, we need to solve the following objective function:

$$\underset{\mathbf{w}\in H, b\in\Re}{\text{minimize}}\ \tau(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2, \qquad (3.4)$$

subject to $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$ for all i=1,...,m

The constraints ensure that $f(x_i)$ will be +1 for y_i=+1 and -1 for y_i=-1. This problem is computationally attractive because it can be constructed by solving a quadratic programming problem for which there are efficient algorithms. Function τ is called objective function (the first feature) with inequality constrains. Together, they form a so-called primal optimization problem. Lets see why we need to minimize the length of **w**. If ||**w**|| would be 1, then the left side of the restriction would equal the distance between x_i and the hyperplane. Generally we need to divide $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$ by ||**w**|| in order to change this into a distance. From now on if we can meet the restriction for all $i=1,..,m$ with a minimum length **w** then the total margin will be maximal.

Problems like this one are the subject of optimization theory and are in general difficult to solve because they imply great computational costs. To solve this type of problems it is more convenient to deal with the dual problem, which according to mathematical demonstrations leads to the same results. In order to introduce the dual problem we have to introduce the Lagrange multipliers $\alpha_i \geq 0$ and the Lagrangian [SCK02] which lead to the so-called dual optimization problem:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i (y_i (\langle \mathbf{x_i}, w \rangle + b) - 1) \qquad (3.5)$$

with Lagrange multipliers $\alpha_i \geq 0$. The Lagrangian L must be maximized with respect to the dual variables α_i, and minimized with respect to the primal variables **w** and b (in other words, a saddle point has to be found). Note that the restrictions are embedded in the second part of Lagrangian and need not be applied separately.

We will try to give an intuitive solution to the restricted optimization problem. If the restriction is violated then $y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle + b) - 1 < 0$, in which case L can be increased by increasing the corresponding α_i. At the same time **w** and b need to be modified if L diminishes. In order that $\alpha_i(y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle + b) - 1)$ doesn't become an arbitrary large negative number the changes in **w** and b will ensure that for a separable problem the conditions will finally be met. Similarly, for all the restrictions that don't meet the equality (that is for which $y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle + b) - 1 > 0$) the corresponding α_i needs to be 0. For these α_i the value of L is maximized. The second complementary restriction in the optimization theory is given by Karush-Kuhn-Tucker (also known as the KKT restrictions). At the saddle point the partial derivates of L with respect to the primal variables need to be 0:

$$\mathbf{w} = \sum_{i=0}^{m} \alpha_i y_i x_i \ , \quad \text{and} \quad \sum_{i=0}^{m} \alpha_i y_i = 0 \qquad (3.6)$$

The solution vector thus has an expansion in terms of a subset of training samples, namely those samples with non zero α_i, called Support Vectors. Note that although the solution **w** is unique (due to the strict convexity of primal optimization problem), the coefficients α_i, need not be. According to the Karush-Kuhn-Tucker (KKT) theorem, only the Lagrange multipliers α_i that are non-zero at the saddle point, correspond to constraints which are precisely met. Formally, for all i=1,…,m, we have

$$\alpha_i [y_i (\langle \mathbf{x_i}, \mathbf{w} \rangle + b) - 1] = 0 \text{ for all i=1,...,m} \qquad (3.7)$$

The patterns x_i for which $\alpha_i > 0$ are called **Support Vectors**. This terminology is related to the corresponding terms in the theory of convex sets, related to convex optimization. According to the

KKT condition they lie exactly on the margin. All remaining training samples are irrelevant. By eliminating the primal variables **w** and *b* in the Lagrangian we arrive to the so-called dual optimization problem, witch is the problem that one usually solves in practice.

$$\underset{\alpha\in\Re^m}{\text{maximize}} W(\alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j \left\langle x_i, x_j \right\rangle \qquad (3.8)$$

this is called the target function

$$\text{subject to } \alpha_i \geq 0 \text{ for all i=1 ,..., m and } \sum_{i=1}^{m}\alpha_i y_i = 0 \qquad (3.9)$$

Thus the hyperplane can be written in the dual optimization problem as:

$$f(x) = \text{sgn}\left(\sum_{i=1}^{m} y_i\alpha_i \left\langle x_i, x\right\rangle + b\right) \qquad (3.10)$$

where *b* is computed using KKT conditions.

The optimization problem structure is very similar to the one that occurs in the mechanical Lagrange formulation. In solving the dual problem it is frequently when only a subset of restriction becomes active. For example, if we want to keep a ball in a box then it will usually roll in a corner. The restrictions corresponding to the walls that are not touched by the ball are irrelevant and can be eliminated.

In practice, the separating hyperplane may not exist, for instance if the classes are overlapped. To take in consideration the samples that can possibly violate the restrictions we can introduce the slack variables $\xi_i \geq 0$ for all $i = 1,...,m$.

Using slack variables the restriction become $y_i (\langle w, \mathbf{x_i} \rangle + b) \geq 1 - \zeta$ for all $i=1,...,m$. The optimum hyperplane can now be found both by controlling the classification capacity ($\|\mathbf{w}\|$) and by controlling the sum of slack variables $\Sigma_i\zeta_i$. We notice that the second part provides a superior boundary to the number of training errors. This is called soft margin classification. The objective function becomes then:

$$\tau(\mathbf{w},\xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i \qquad (3.11)$$

with the same restrictions, where the constant C > 0 determines the exchange between maximizing the margin and minimizing the number of training errors. If we rewrite this in terms of Lagrange multipliers, we get the same maximization problem with an extra restriction:

$$0 \leq \alpha_i \leq C \text{ for all } i=1,...,m \text{ and } \sum_{i=1}^{m}\alpha_i y_i = 0 \qquad (3.12)$$

Everything was formulated in a dot product space. We think of this space as the feature space. On the practical level, changes have to be made to perform the algorithm in a higher-dimensional feature space. The patterns x_i thus need not coincide with the input patterns. They can equally well be the result of mapping the original input patterns x_i into a higher dimensional feature space using Φ function. Maximizing the target function and evaluating the decision function, than requests the computation of dot products $\langle \phi(x), \phi(x) \rangle$ in a high dimensional space. These expensive calculations are reduced significantly by using a positive definite kernel *k*, such

that $k(x,x') = \langle x, x' \rangle$. This substitution, which is referred sometimes as the kernel trick is used to extend the hyperplane classification to nonlinear Support Vector Machines. The kernel trick can be applied since all feature vectors only occur in dot products. The weight vectors than becomes an expression in the feature space, and therefore Φ will be the function through which we represent the input vector in the new space. Thus we obtain the decision function:

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i k(x, x_i) + b \right) \quad (3.13)$$

The main advantage of this algorithm is that it doesn't require transposing all data into a higher dimensional space. So there is no expensive calculus as in neural networks. We also get a smaller dimensional set of testing data as in the training phase we will only consider the support vectors which are usually few. Another advantage of this algorithm is that it allows the usage of training data with as many features as needed without increasing the processing time exponentially. This is not true for neural networks. For instance the back propagation algorithm has troubles dealing with a lot of features. The only problem of the support vector algorithm is the resulting number of support vectors. As the number increases the response time increases linearly too.

3.2 Multiclass Classification

Most real life problems require classification to more than two classes. There are several methods for dealing with multiple classes that use binary classification. One of these methods consists in classifying "one versus the rest" in which elements that belong to a class are differentiated from the others. In this case we calculate an optimal hyperplane that separates each class from the rest of the elements. As the output of the algorithm we will choose the maximum value obtained from all decision functions.

To get a classification in M classes, we construct a set of binary classifiers where each train separates one class from the rest. After that we combine them by doing the multi-class classification according to the maximal output before applying the *sgn* function; that is, by taking

$$\arg\max_{j=1,..,M} g^j(x), \quad where\ g^j(x) = \sum_{i=1}^{m} y_i \alpha_i^j k(x, x_i) + b^j \quad (3.14)$$

and

$$f^j(x) = \operatorname{sgn}(g^j(x)) \quad (3.15)$$

This problem has a linear complexity as for M classes we compute M hyperplanes. Another method for multiclass classification consists in classifying the pairs. In this method we choose two classes and we compute a hyperplane for them. We do this for each pair of classes in the training set. For M classes we compute (M-1)*M/2 hyperplanes. This method has a polynomial complexity. The advantage of this method is that usually the resulting hyperplane has a smaller dimension (less support vectors).

3.3 Clustering using Support Vector Machine

Further on we will designate by classification the process of supervised learning on labeled data and we will designate by clustering the process of unsupervised learning on unlabeled data. The algorithm above only uses labeled training data. Vapnik presents in [VAP01] an alteration of the classical algorithm in which are used unlabeled training data. Here finding the hyperplane becomes finding a maximum dimensional sphere of minimum cost that groups the most resembling data (presented also in [Jai00]. This approach will be presented as follows. In [SCK02] we can find a different clustering algorithm based mostly on probabilities.

For document clustering we will use the terms defined above and we will mention the necessary changes for running the algorithm on unlabeled data.

There are more types of usually used kernels that can be used in the decision function. The most frequently used are the linear kernel, the polynomial kernel, the Gaussian kernel and the sigmoid kernel. We can choose the kernel according to the type of data that we are using. The linear and the polynomial kernel run best when the data is well separated. The Gaussian and the sigmoid kernel work best when data is overlapped but the number of support vectors also increases.

For clustering the training data will be mapped in a higher dimensional feature space using the Gaussian kernel. In this space we will try to find the smallest sphere that includes the image of the mapped data. This is possible as data is generated by a given distribution and when they are mapped in a higher dimensional feature space they will group in a cluster. After computing the dimensions of the sphere this will be remapped in the original space. The boundary of the sphere will be transformed in one or more boundaries that will contain the classified data. The resulting boundaries can be considered as margins of the clusters in the input space. Points belonging to the same cluster will have the same boundary. As the width parameter of the Gaussian kernel decreases the number of unconnected boundaries increases. When the width parameters increases there will be overlapping clusters. We will use the following form for the Gaussian kernel:

$$\Phi(x,x') = e^{-\frac{\|x-x'\|^2}{2\cdot\sigma^2}} \qquad (3.16)$$

Note this that if σ^2 decreases the exponent increases in absolute value and so the value of the kernel will tend to 0. This will map the data into a smaller dimensional space instead of a higher dimensional space. Mathematically speaking the algorithm try to find "the valleys" of the generating distribution of the training data.

If there are outliers the algorithm above will determine a sphere of very high costs. To avoid excessive costs and misclassification there is a soft margin version of the algorithm similar to the one in section 3.1. The soft margin algorithm uses a constant so that distant points (high cost points) won't be included in the sphere. In other words we choose a limit of the cost to introduce an element in the sphere.

We will present in detail the clustering algorithm. We will also use some mathematical aspects to justify some of the assertions. Let $X \in \Re^n$ be the input space, $\{x_i\} \subseteq X$ the set of N samples to be classified, R the radius of the searched sphere and the Gaussian kernel presented in (3.16). We will consider a as being the center of the sphere. Given this the problem becomes:

$$\| \Phi(x_j) - a \|^2 \le R, \forall j = 1...N \qquad (3.17)$$

and by replaced the norm with the corresponding dot product we get:

$$< \Phi(x_j) - a, \Phi(x_j) - a > \leq R, \forall j = 1...N \quad (3.18)$$

Equation (3.17) represents the primal optimization problem for clustering.

For the soft margin case, we will use the slack variables ξ_i and formulas (3.17) and (3.18) become:

$$\| \Phi(x_j) - a \|^2 \leq R + \xi_j, \forall j = 1...N \quad (3.19)$$

and respectively:

$$< \Phi(x_j) - a, \Phi(x_j) - a > \leq R + \xi_j, \forall j = 1...N \quad (3.20)$$

The equations above are to equivalent forms of the same primal optimization problem using soft margin. The primal optimization problem is very difficult if not impossible to solve in practice. This is why we will introduce the Lagrangian and the dual optimization problem. As above the solution of the dual problem is the same to the solution of the primal problem.

$$L = R^2 - \sum_j \left(R^2 + \xi_i - \left\| \Phi(x_j - a) \right\|^2 \right) \cdot \beta_j - \sum_j \xi_j \mu_j + C \sum_j \xi_j \quad (3.21)$$

where $\beta_j \geq 0, \mu_j \geq 0$ are the Lagrange multipliers, C is a constant and $C \sum \xi_j$ is the penalty term. In this case we have two Lagrange multipliers as there are more restrictions to be met.

Considering the partial derivates of L with respect to *R*, *a* and ξ_j (primal variables) equal to 0 we get:

$$\frac{\partial L}{\partial R} = 2R - 2\sum_j R\beta_j = 2R\left(1 - \sum_j \beta_j \right) = 0 \Rightarrow \sum_j \beta_j = 1 \quad (3.22)$$

$$\frac{\partial L}{\partial a} = -\sum_j \beta_j \Phi(x_j) + a = 0 \Rightarrow a = \sum_j \beta_j \Phi(x_j) \quad (3.23)$$

$$\frac{\partial L}{\partial \xi_j} = -\beta_j - \mu_j + C = 0 \Rightarrow \beta_j = C - \mu_j \quad (3.24)$$

Then the KKT restrictions are:

$$\xi_j \cdot \mu_j = 0 \quad (3.25)$$

$$\left(R^2 + \xi_j - \left\| \Phi(x_j) - a \right\|^2 \right) \cdot \beta_j = 0 \quad (3.26)$$

Analyzing the problem and the restriction above we notice the following cases:

$\xi_i > 0$ and $\beta_j > 0 \Rightarrow \left\| \Phi(x_i) - a \right\|^2 > R$, which means that $\Phi(x_i)$ lies outside the sphere

$\mu_i = 0 \Rightarrow \beta_j = C \Rightarrow \left\| \Phi(x_i) - a \right\|^2 > R$, which means that x_i is a bounded support vector - BSV

$\xi_i = 0 \Rightarrow \left\| \Phi(x_i) - a \right\|^2 \leq R$, which means that these elements belong to the interior of the sphere and they will be classified when we remap them into the original space.

$\xi_i = 0$ and $0 < \beta_i < C \Rightarrow \left\| \Phi(x_i) - a \right\|^2 = R$, which means that the points x_i lie on the sphere. These points are actually the support vectors that will determine the clusters and that will then be used to classify new elements.

Thus the support vectors (SV) will lie exactly on the sphere and the bounded support vectors (BSV) will lie outside the sphere. We can easily see that when $C \geq 1$ there are no BSV, so all the points in the input space will be classified (the hard margin case).

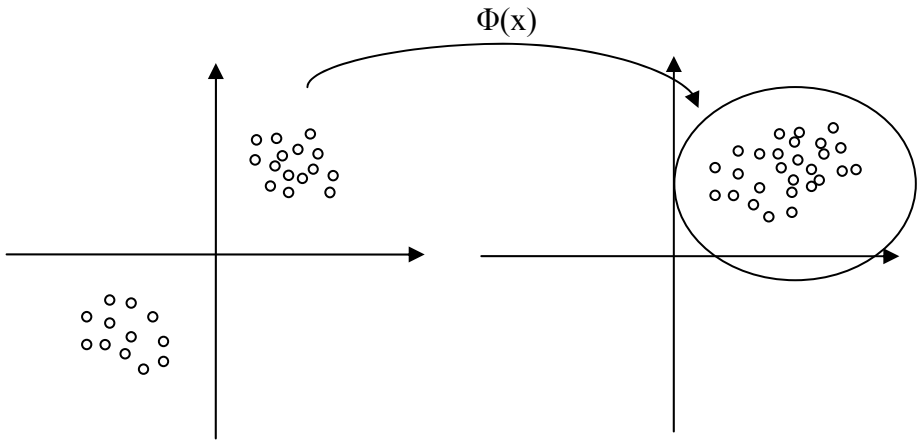


Figure 3.2– Mapping the unlabeled data from the input space into a higher dimensional feature space and determining the sphere that includes all points (hard margin)

In Figure 3.2 we presented the geometrical interpretation of the first part of the optimization problem: mapping the unlabeled data from the input space into a higher dimensional feature space via the Gaussian kernel function Φ. Thus we can determine the sphere that contains all the points to be classified. We considered the hard margin case. We will now present the mapping of the sphere in the input space to see the way we create the boundaries.

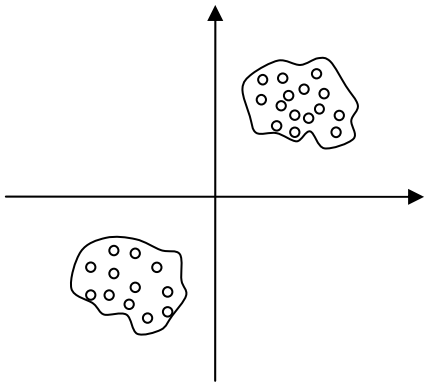


Figure 3.3 – Grouping data into classes

In Figure 3.3 we present the way data is grouped after mapping the boundary of the sphere in Figure 3.2 back in the input space.

This is very difficult to implement as an algorithm. For an easier implementation we will eliminate R, a and β_j so that the Lagrangian becomes a Wolfe in the dual form. The Wolfe obtained so is an optimization problem in β_j. We will now replace in (3.21) formulas (3.23) and (3.24) and we will have:

$$W = R^2 - \sum_j \left(R^2 + \xi_j - \left\| \Phi(x_j) - \sum_i \beta_i \Phi(x_i) \right\|^2 \right) \cdot \beta_j - \sum_j \xi_j (C - \beta_j) + C \sum_j \xi_j \Rightarrow$$

$$W = R^2 - \sum_j R^2 \cdot \beta_j - \sum_j \xi_j \cdot \beta_j + \sum_j \left\| \Phi(x_j) - \sum_i \beta_i \Phi(x_i) \right\|^2 \cdot \beta_j - \sum_j \xi_j C + \sum_j \xi_j \beta_j + C \sum_j \xi_j \Rightarrow$$

$$W = R^2 \left(1 - \sum_j \beta_j \right) + \sum_j \left\| \Phi(x_j) - \sum_i \beta_i \Phi(x_i) \right\|^2 \cdot \beta_j \Rightarrow$$

$$W = \sum_j \left\| \Phi(x_j) - \sum_i \beta_i \Phi(x_i) \right\|^2 \cdot \beta_j \Rightarrow$$

$$W = \sum_j < \Phi(x_j) - \sum_i \beta_i \Phi(x_i), \Phi(x_j) - \sum_i \beta_i \Phi(x_i) > \cdot \beta_j \Rightarrow$$

$$W = \sum_j \left[< \Phi(x_j), \Phi(x_j) > - < \Phi(x_j), \sum_i \beta_i \Phi(x_i) > - < \sum_i \beta_i \Phi(x_i), \Phi(x_j) > + < \sum_i \beta_i \Phi(x_i), \sum_i \beta_i \Phi(x_i) > \right] \cdot$$

$$\cdot \beta_j \Rightarrow$$

$$W = \sum_j \left[< \Phi(x_j), \Phi(x_j) > + < \sum_i \beta_i \Phi(x_i), \sum_i \beta_i \Phi(x_i) > \right] \cdot \beta_j \Rightarrow$$

$$W = \sum_j < \Phi(x_j), \Phi(x_j) > \cdot \beta_j - \sum_{i,j} \beta_i \beta_j \Phi(x_i) \Phi(x_j) \qquad (3.27)$$

The restrictions for μ_j will be replaced by:

$$0 \le \beta_j \le C, \quad j = 1 \ldots N \qquad (3.28)$$

Using the support vector technique for representing the dot product $\langle \Phi(x_i), \Phi(x_j) \rangle$ by a Gaussian kernel:

$$K(x_i, x_j) = e^{-q \cdot \|x_i - x_j\|^2} \qquad (3.29)$$

with the width parameter $q = \frac{1}{\sigma}$.

In these conditions the Lagrangian becomes:

$$W = \sum_j K(x_j, x_j) \cdot \beta_j - \sum_{i,j} \beta_i \beta_j K(x_i, x_j) \qquad (3.30)$$

We will denote the distance for each point *x* the distance from each image in the feature space to the center of the sphere as:

$$R^2 = \|\Phi(x) - a\|^2 \qquad (3.31)$$

and by using (3.23) this becomes:

$$R^2 = \left\| \Phi(x) - \sum_j \beta_j \Phi(x_j) \right\|^2 \Rightarrow$$

$$R^2 = < \Phi(x) - \sum_j \beta_j \Phi(x_j), \Phi(x) - \sum_j \beta_j \Phi(x_j) > \Rightarrow$$

$$R^2 = < \Phi(x), \Phi(x) > - < \sum_j \beta_j \Phi(x_j), \Phi(x) > - < \Phi(x), \sum_j \beta_j \Phi(x_j) > + < \sum_j \beta_j \Phi(x_j), \sum_j \beta_j \Phi(x_j) > \Rightarrow$$

$$R^2 = <\Phi(x),\Phi(x)> - \sum_j \beta_j <\Phi(x_j),\Phi(x)> - \sum_j \beta_j <\Phi(x),\Phi(x_j)> + \sum_{i,j} \beta_i \beta_j <\Phi(x_j),\Phi(x_j)> \Rightarrow$$

$$R^2(x) = K(x,x) - 2\sum_j \beta_j K(x_j,x) + \sum_{i,j} \beta_i \beta_j K(x_i,x_j) \quad (3.32)$$

The radius will now be:

$$R = \{R(x_i) | x_i \in SV\} \quad (3.33)$$

which is:

$$R = \{R(x_i) | x_i cu \xi_i = 0 \text{ and } 0 < \beta_i < C\}$$

The boundaries that include points belonging to the same cluster are defined by the following set:

$$\{x | R(x) = R\} \quad (3.34)$$

Another issue to be discussed is assigning the cluster, which is how do we decided what points are in what class. Geometrically speaking this means determining the entering of the boundaries. When two points are in different clusters the line that unites them is transpose outside of the sphere in the higher space. This kind of line will contain a point y for which $R(y) > R$. In order to determine the existence of points y it is necessary to consider a number of test points to check if it goes outside the sphere or not. Let M be the number of test points and so the number of points being considered for testing if the line belongs or not to the sphere. The n-th test point will be:

$$y_n = \frac{n(x_i + x_j)}{M}, \quad n = \overline{1, M-1} \quad (3.35)$$

The results of these tests will be stored in an adjacency matrix A for each pair of points. The form of adjacency matrix will be:

$$A_{ij} = \begin{cases} 1, \forall y \in x_i x_j, \text{ with } R(y) \le R \\ 0, \text{ otherwise} \end{cases} \quad (3.36)$$

The cluster will be defined by the connected components of graphs induced by matrix A.

When we use the soft margin and thus have BSV, these will not be classified in any of the classes by this procedure. They can be assigned to the closest class or they can remain unclassified.

3.4 SMO - Sequential Minimal Optimization

This represents one of methods that implement a problem of quadratic programming introduced by Platt [Pla99]. The strategy of SMO is to break up the constraints into the smallest optimization groups possible. Note that it is not possible to modify variables α_i individually without modifying the sum of constraints (the KKT conditions). We therefore generate a generic convex constrained optimization problem for pairs of variables. Thus, we consider the optimization over two variables α_i and α_j with all other variables considered fixed, optimizing the target function with respect to them. Here i and j are sample i and sample j from the training set. The exposition proceeds as follows: first we solve the generic optimization problem in two variables and subsequently we determine the value of the placeholders of the generic problem, we adjust b properly and determine how pattern can be selected to ensure speedy convergence.

Thus the problem presented above implies solving a linearly analytical optimization problem in two variables. The only problem now is the order that the training samples are chosen in for speedy convergence. By using two elements we have the advantage of dealing with the linear decision function which is easier to solve than a quadratic programming optimization problem. Another advantage of SMO algorithm is that not all training data must be simultaneously in the memory, but only the samples for which we optimize the decision function. This fact allows the algorithm to work with very high dimension samples.

We begin with a generic convex constrained optimization problem in two variables. Using the shorthand $K_{ij} := k(x_i,x_j)$ the quadratic problem becomes:

$$\begin{aligned} &\underset{\alpha_i,\alpha_j}{\text{minimize}} \frac{1}{2}\left[\alpha_i^2 K_{ii} + \alpha_j^2 K_{jj} + 2\alpha_i\alpha_j K_{ij}\right] + c_i\alpha_i + c_j\alpha_j \\ &\text{subject to} \quad s\alpha_i + \alpha_j = \gamma \\ &\qquad\qquad 0 \le \alpha_i \le C_i \quad \text{and} \quad 0 \le \alpha_j \le C_j \end{aligned} \tag{3.37}$$

where α_i and α_j are the Lagrange multipliers for those two samples, K is the kernel and s is +1 or -1. Here, c_i, c_j, $\gamma \in \mathrm{R}$, and $K \in \Re^{2\times 2}$ are chosen suitably to take the effect of the *m-2* variables that are kept fixed. The constants C_i represent pattern dependent regularization parameters. To obtain the minimum we use the restriction $\alpha_i + \alpha_j = \gamma$. This formula allows us to reduce actual optimization problem to a optimization problem in α_i by eliminating α_j. For shorthand we use the following notation $\chi := K_{ii} + K_{jj} - 2K_{ij}$

In the implementation of SMO I take the value for the constants C_i equal to 1 because I have chosen hard margin in primal form of SVM. Constants c_i, c_j, and γ are defined thus:

$$c_i := \sum_{l\neq i,j}^{m} \alpha_l K_{il},\ c_j := \sum_{l\neq i,j}^{m} \alpha_l K_{jl} \ \textit{and}\ \gamma = 1 - \sum_{l\neq i,j}^{m} \alpha_l \tag{3.38}$$

As the initial condition I have considered all weight vectors equal to zero and the Lagrange multipliers α for all samples equal to zero. The dimension of the weight vector is equal to the number of attributes and the dimension of **α** vector is equal to the number of samples. To solve the minimization problem in the objective function I have created five sets that split the input data, according to KKT condition, as follows:

$$\begin{aligned} &I_0 = \{i | \alpha_i \in (0, C_i)\} \quad I_{+,0} = \{i | \alpha_i = 0, y_i = +1\} \quad I_{+,C} = \{i | \alpha_i = C_i, y_i = +1\} \\ &\qquad\qquad\qquad\quad I_{-,0} = \{i | \alpha_i = 0, y_i = -1\} \quad I_{-,C} = \{i | \alpha_i = C_i, y_i = -1\} \end{aligned} \tag{3.39}$$

From now on we will use the following notations:

- m_I0 - all the training samples for which the Lagrange multipliers **α** are between 0 and C
- m_I0_p - all the samples for which **α** = 0 and positive topic
- m_I0_n - all the samples for which **α** = 0 and negative topic
- m_IC_p - all the samples for which **α** = C and positive topic
- m_IC_n - all the samples for which **α** = C and negative topic

I have also defined:

$$m\_bUp := \min_{i\in m\_I0\cup m\_I0\_p\cup m\_IC\_n} f(x_i) - y_i \tag{3.40}$$

$$m\_bLow := \min_{i \in m\_I0 \cup m\_I0\_n \cup m\_IC\_p} f(x_i) - y_i$$

where m_bUp and m_bLow are the superior and the inferior limits for the margin of the hyperplane. As an improved estimation of b I have used the formula $m\_b = \frac{(m\_bUp + m\_bLow)}{2}$. The stopping criteria is no longer that the KKT conditions are smaller than some tolerance *Tol*, but that m_bLow ≤ m_bUp, holds with some tolerance $m\_bLow \leq m\_bUp - Tol$.

In the training part, the samples with $\boldsymbol{\alpha} \neq 0$ are considered support vectors and are relevant in the objective function. They are the only ones that need to be kept from the training set. For this, I have created a set called "m_supportVectors" that stores all the samples that have $\boldsymbol{\alpha} \neq 0$. In the application I have used a member "m_data" that keeps all the training samples from the file. Each sample has a specified position in "m_data". This position is the same in all the sets used to make the search easier. If the sets don't contain the sample on the specified position, the entry for that position contains a *null* value.

Considering the above I have put all the samples in the two sets, m_I0_p and m_I0_n, because for the beginning I have considered that $\boldsymbol{\alpha}$ and the weight vector equal to zero. Then I examine all the samples, renewing all the sets. If during this process there is no change we can consider that the training is over, otherwise I repeat the process again as presented below.

In minimization problem, in order to find the minimum, we use $\alpha_i+\alpha_j=\gamma$. Modifying α_i depends on the difference between the values $f(x_i)$ and $f(x_j)$. There are some algorithms that can be used for choosing the index i and j, thus the larger the difference among them, the bigger is the distance to the hyperplane. I have chosen the two loops approach to maximize the objective function. The outer loop iterates over all patterns violating the KKT conditions, or possibly over those where the threshold condition of the m_b is violated. Usually we first loop over those with Lagrange multipliers neither on the upper nor the lower boundary. Once all of these are satisfied we loop over all patterns violating the KKT conditions, to ensure self consistency on the entire dataset. This solves the problem of choosing the index i.

It is sometimes useful, especially when dealing with noise data, to iterate over the complete KKT violating dataset before complete self consistence on the subset has been achieved. Otherwise considerable computational resources are spent making subsets. The trick is to perform a sweep through the data once only less than, say 10% of the non bound variables change.

Now to select j: to make a large step towards the minimum, one looks for large steps in α_i. Since it is computationally expensive to compute $K_{ii} + K_{jj} - 2sK_{ij}$ for all possible pairs (i,j) one chooses an heuristic to maximize the change in the Lagrange multipliers α_i and thus maximize the absolute value of numerator in expressions for $\tilde{\alpha}_i$ (new alpha). This means that we are looking for patterns with large difference in their relative errors $f(x_i) - y_i$ and $f(x_j) - y_j$. The index j corresponding to the maximum absolute value is chosen for this purpose.

If this heuristic happens to fail (if little progress is made by this choice) all other indices j are looked at in the following way (a second choice):

- All indices j corresponding to non-bound examples are looked at, searching for an example to make progress on.
- In the case that the first heuristic was unsuccessful, all other sample are analyzed until an example is found where progress can be made.
- If both previous steps fail, SMO precedes to the next index i.

In the problem of computing the offset b from the decision function another stopping criterion occurs. This new criterion occurs because, unfortunately, during the training, not all Lagrange multipliers will be optimal, since, if they were, we would already have obtained the solution. Hence, obtaining *b* by exploiting the KKT conditions is not accurate (because the margin must be exactly 1 for Lagrange multipliers for which the box constraints are inactive). Using the limit of the margin presented in (3.40) and since the KKT condition has to hold for a solution we can check that this corresponds to $m\_bUp \geq 0 \geq m\_bLow$. For m_*I0* I have already used this fact in the decision function. After that using KKT conditions I have renewed the thresholds "m_bUp" and "m_bLow" (superior and inferior limit for margin of the hyperplane). I have also chosen an interval from which to choose the second sample which I renew by renewing "m_iUp" and "m_iLow". According to the class of the first sample I choose the second training sample as follows. If the first sample is in the set with positive topic I choose the second sample from the set with negative topic. After choosing the samples for training I check if there is a real progress in the algorithm. The real benefit comes from the fact that we may use m_iLow and m_iUp to choose patterns to focus on. The large contribution to the discrepancy between m_iUp and m_iLow stems from those pairs of patterns (i, j) for which the discrepancy:

$$discrepancy\ (i,j) := (f(x_i)-y_i) - (f(x_j) - y_j) \text{ where } \begin{cases} i \in I_0 \cup I_{-,0} \cup I_{+,C} \\ j \in I_0 \cup I_{+,0} \cup I_{-,C} \end{cases} \quad (3.41)$$

is the largest. In order to have a real benefit the discrepancy needs to have a high value.
For the first sample I have computed the superior and inferior limit of α using:

$$\alpha_i = \begin{cases} L & \zeta > 0 \\ H & otherwise \end{cases}, \quad (3.42)$$

where $L = \begin{cases} \max(0, s^{-1}(\gamma - C_j)) & if\ s > 0 \\ \max(0, s^{-1}\gamma) & otherwise \end{cases}$ and $H = \begin{cases} \min(C_i, s^{-1}\gamma) & if\ s > 0 \\ \max(C_i, s^{-1}(\gamma - C_j)) & otherwise \end{cases}$

and $\begin{aligned} \zeta &:= sc_j - c_i + \gamma s K_{jj} - \gamma K_{ij} \\ \chi &:= K_{ii} + K_{jj} - 2sK_{ij} \end{aligned}$, (3.43)

After that I can compute the new α using the formula:

$$\overline{\alpha} = \begin{cases} \alpha_i^{old} + \chi^{-1}\delta & if\ \chi > 0 \\ -\infty & if\ \chi = 0\ and\ \delta > 0 \\ \infty & if\ \chi = 0\ and\ \delta < 0 \end{cases}, \quad (3.44)$$

where $\delta := y_i((f(x_j) - y_j) - (f(x_i) - y_i))$, and $\overline{\alpha}$ is the value of the new alpha (the solution without restriction).
After that I have computed the new α for first sample and I checked and forced it to hold the KKT conditions (to be between 0 and C). After that I have computed the new α for the second sample using the formula $\alpha_j = s(\alpha_i^{old} - \alpha_i) - \alpha_j^{old}$. I have modified all the sets using the new α. Thus for the first sample if the new α is greater than zero it is included in the "m_supportVector" set, otherwise it is eliminated if it is in set. If α is between 0 and C and the topic is positive this sample is included in "m_I0_p" otherwise it is eliminated from the set if it is, etc. We repeat that with the

second sample. After that I have updated the weight vector for all the attributes using the formula:

$$w = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

Using the new objective function we can compute the classifier error for all samples, the inferior and the superior limit of the margin of the hyperplane *b,* and the inferior and superior margin of the domain for choosing sample "m_iUp" and "m_iLow".

To speed up the computing process, I have used in my program a large matrix called "m_store" in which I have stored the outputs for the kernels computed between two samples. The next time when I need the value, I will take it from the matrix. I have put the value in this matrix after I have calculated the value of the kernel. I use the trick that the value of kernel for two variables is constant in report with evolution of variables from algorithm.

There is a different approach of the algorithm for linear kernel and others kernels. In the case of the linear kernel the objective of the algorithm is to compute the weight from the initial decision function taking into consideration the primal optimization problem "*f(x) = sgn(<*w*, Φ(x)> + b)"*. The linear kernel is a particular case of the polynomial kernel, when the degree is 1. We take into consideration only the weight because with this type of kernel, the data is kept in the original space. In almost all cases this type of kernel, presented by some researchers separately, obtains poor results in the classification. Also it is used especially when the algorithm based on support vectors is used for feature selection because it produces a separate weight for each attribute. Other type of kernels used, like the polynomial kernel with the degree greater then 1 and the Gaussian kernel, use decision function $f(x) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i k(x_i, x) + b\right)$ that comes form dual optimization problem (where I compute only the Lagrange multipliers **α.**

3.5 Probabilistic Outputs for SVM

The standard SVM doesn't have an output to measure the confidence of the classification. Platt in [Pla99_2] presents a method for transforming the standard SVM output into a posterior probability output. Posterior probabilities are also required when a classifier is making a small part of an overall decision, and the classification outputs must be combined for the overall decision. Thus the author presents a method for extracting probabilities P(class / input) from the output of SVM. These can be used for post processing classification. This method leaves the SVM error function unchanged. For classification in multiple classes, the class is chosen based on maximal posterior probability over all classes. The author follows the idea of [Wah99] for producing probabilistic outputs from a kernel machine. Support Vector Machine produce an uncalibrated value that is not a probability, and if signum function is applied it obtain only two values (-1 and 1).

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i k(x_i, x) + b\right) \qquad (3.45)$$

that lies in a Reproducing Kernel Hilbert Space. Training a SVM minimizes an error function that penalizes an approximation of the training misclassification rate plus a penalty term. Minimizing the error function is also minimizing a bound on the test misclassification rate, which is also a desirable goal. An additional advantage of this error function is that minimizing will produce a sparse machine where only a subset of possible kernels is used in the final machine.

$$P(class|input) = P(y = 1|x) = p(x) = \frac{1}{1 + \exp(-f(x))} \qquad (3.46)$$

where $f(x)$ is the output function for SVM. Instead of estimating the class-conditional densities $p(f \mid y)$, the authors suggest using a parametric model to fit the posterior probability $P(y=1|f)$ directly. The parameters of the model are adapted to give the best probability outputs. The authors analyse the empirical data and see that the density are very far away from Gaussian. There are discontinuities in the derivates of both densities and positive margin $f=1$ and the negative margin $f=-1$. These discontinuities occur because the cost function also has discontinuities at the margins. The class-conditional densities between the margins are apparently exponential and the authors suggest the usage of a parametric form of the sigmoid:

$$P(y=1|f)=\frac{1}{1+\exp(Af(x)+B)} \qquad (3.47)$$

This sigmoid model is equivalent to assuming that the output of the SVM is proportional to the logarithmical probabilities of positive samples. This sigmoid model has two parameters which are trained independently in an extra post processing step. They are trained using the regularized binomial likelihood. As long as $A < 0$, the monotony of (3.45) is assured. The parameters A and B from (3.47) are found using maximum likelihood estimation from the training set (f_i, y_i). The complete algorithm for computing the parameters A and B is presented in [Pla99_2]. Also this method was used by Kai in [Kai03] using only a single parameter.

4 Experimental Research

4.1 Background Work

4.1.1 Experimental Data Sets and Feature Selection

In this section, I will describe the process of extracting useful data that I will use and the developed applications. I have used for experiments the Reuters-2000 collection [Reu2000], which includes a total of 806791 documents, with news stories. The Reuters collection is commonly used in text categorization research. Due to the huge dimension of the database I have chosen only a subset of data that I will work upon referred to as Subset-c152. Thus form all 806791 documents I have selected those documents that are grouped by Reuters in "System Software" (I330202) by industry. After this classification there are 7083 documents left to work on. In the resulting documents there are 63 different topics for classifying, according to Reuters. For binary classification I chose the topic "c152" that means "Comment /Forecasts", according to Reuters codes. I have chosen topic "c152" as only about 30% of the data belongs to that class so the algorithm can actually learn. Those 7083 articles are divided randomly into a training set of 4722 documents and a testing set of 2361 documents (samples). Most researchers take in consideration only the title and the abstract of the article or only the first 200 words from each piece of news. In my evaluation I take into consideration the entire news and the title of the news in order to create the characteristic vector. I have used the bag-of-words approach for representing the documents as presented in section 2.

Text categorization is the problem of automatically assigning predefined categories to free text documents. The major problem is the high dimensionality of the feature space. Automatic feature selection methods include the removal of non-informative terms according to corps statistics and the construction of new features which combine lower level feature into higher level orthogonal dimensions. In feature selection the focus consists in an aggressive dimensionality reduction.

In Subset-c152 we have 19038 distinctive attributes that represent in fact the root of the words form the documents. A training algorithm with a vector of this dimension is time consuming and in almost all cases the accuracy of the learning is low due to the noise in data. For feature selection I have evaluated three methods including term selection based on document frequency. The first method that I used is Information Gain with different thresholds. By varying the threshold I have obtained a number of features varying from 41 for the threshold equal to 0.1 to 7999 for the threshold equal to 0.001. The method was presented in section 2.2.1. Another method used is Support Vector Machine algorithm with linear kernel like in [Mla02]. For this I use the threshold between 0.1 and 0.001 obtaining a number of features between 427 and 12936. As we will see a relatively small number of features make the classifier work better than with many features. The same conclusions were drawn in [Gab04].

For the text mining step I have created a java package "ReadReuters" that uses few classes from the package ir.jar [IR] in order to extract the data (the root of the words and the number of appearances) from the documents. This package generates the four files presented in section 2.1.

For the learning and testing step I have implemented a Java application using the algorithm presented in [SCK02] based on a support vector technique. For this algorithm I have added the idea proposed by Platt in [Pla99_2] for the probabilistic output of SVM presented in section 3.5 of this report. I have tested this algorithm using the Subset-c152 for classifying in two classes and I have tested different kernels and degrees of kernels for the SVM algorithm. Also I have analyzed

different methods of feature selection like Information Gain and feature selection using Support Vector Machine. When using feature selection with different thresholds there can be cases when all the attributes (features) into a vector are zero (the document doesn't contain the selected attributes) and in this case we eliminate that vector. This kind of documents won't be represented by any vector so their value will be lost. Thus we can have a smaller number of samples in the training or the testing set. The number of documents presented above is obtained for the minimum threshold.

4.1.2 Application's Parameters

All applications run and accept parameters from command line. We have done this as we are more interested in the results of the learning process instead in a graphical interface. All learning results will be stored in a file. Even though we have created a simple graphical interface to see the way the classes look like for the two dimensional model.

The application for the data mining step has the following command line:
java supportVector.ReadReuters -t <folder name> [-T <threshold value>] [-f <feature selection method>]

Where the options are:

- *"-t <folder name>"* – the name of the folder where the Reuters files are located (or other files in Reuters format). This option is compulsory.
- *"-T <threshold value> "*- value of the threshold, the number of occurrences of the word in all documents. If this option is omitted default value is 0.
- "*[-f <feature selection method>]"* – the method used for feature selection. You can choose from Information Gain, Average entropy, Mutual information or no method of feature selection.

If the compulsory options is omitted or it is called using option "-?" the application returns a message with all these options.

The learning application has the general command line:

SupportVector.UseSVM -t <training file> -v <testing file> [-k <kernel types>] [-c <degree or constant >] [-d <type of attributes>] [-f <SVM feature selection>]

Where the options are:

- *"-t <training file>*" - where training file is the name of the file containing the training set - this option is compulsory
- *"-v <testing file>*" - where testing file is the name of the file containing the testing set - this option is compulsory
- *"-k <kernel's types>* "- type of kernel. If we want to test using polynomial kernel we need to write "-k POL", for radial basis function kernel we need to write "-k RBF". If this option is omitted the default value is polynomial kernel.
- *"-c <degree or constant>*" – For polynomial kernel it represents the degree of the polynomial and for the RBF kernel it represents the constant of the denominator. Any double value is admitted. If this option is omitted the default value is 1.
- *"-d <type of attributes>*" – type of attribute representation in the training and testing sets. Possible values are "BIN" for binary representation, "NOM" for nominal representation, "SMART" for Connell Smart representation. The default value is BIN.

- "*-f <SVM feature selection>"* – it is used when we want to use the algorithm as a feature selection algorithm. The double value represents the value of the threshold.

If the compulsory option is omitted or if the application is called using option "-?" the application returns a message with all the options above.

The java applet has all the parameters of the learning application besides the training and testing files as the data is taken from the interface. The training data is considered to be the points given by the user and testing data are all the points of the applet.

4.1.3 Types of Kernels Used

The purpose of the kernel is to transpose the training data from the input space in a higher dimensional feature space and to separate the data in this new space. The idea of the kernel is to compute the norm of the difference between two vectors (the cosine of the angle between the two vectors) in a higher dimensional space without representing those vectors in the new space (kernel trick). In order to use the kernel trick we need to express the kernel in terms of dot products. In practice we can see that by adding a scalar constant to the kernel we can get better classifying and clustering results. In this report I tested a new idea to correlate this new scalar with the dimension of the space where the data will be represented because I consider that those two parameters (the degree and the scalar) need to be correlated.

I will present the results for different kernels and for different parameters of each kernel. For the polynomial kernel I change the degree of the polynomial and for the Gaussian kernel I change the constant C according to the following formulas:

- polynomial $k(x,x') = \left(2 \cdot d + \langle x \cdot x' \rangle\right)^d$ - d being the parameter to be modified
- Gaussian (radial basis function RBF) $k(x,x') = \exp\left(-\frac{\|x-x'\|^2}{n \cdot C}\right)$ - C being the parameter to be modified and n being the number of elements greater that 0 from the input vectors.

x and x' being the input vectors.

For the linear kernel I used the polynomial kernel with degree 1. Linear kernel was also used for feature selection.

4.1.4 LibSvm

Almost all researchers present their results using an implementation of support vector machine technique called "LibSVM" available at [LibSvm]. LibSvm is a simple, easy-to-use, and efficient software for SVM classification and regression. In almost all cases I try to present results of classification of my application versus the results obtained for the same set using LibSvm. LibSvm is a command line application and allows the user to select different algorithms for SVM, different types of kernels and different parameters for each kernel. LibSvm implemented five type of SVM (C-SVM, nu-SVM, one-class SVM, epsilon-SVR and nu-SVR) and has 4 types of kernels (linear, polynomial, Gaussian and sigmoid). The forms of the kernels that we used are:

- polynomial $k(x,x') = \left(gamma \cdot \langle x \cdot x' \rangle + coef0\right)^d$, where *gamma, d* and *coef0* are variables.
- radial basis function (RBF) $k(x,x') = \exp\left(-gamma * \|x-x'\|^2\right)$, where *gamma* is variable.

The default value of *gamma* is 1/k, k is the number of attributes, the default value of *d* is 3, and the default value of *coef0* is 0.

I will use for comparisons only "C-SVM" and "one-class SVM" with different parameters. I will compare with C-SVM for supervised classification and with "one-class SVM" for unsupervised case (clustering).

4.2 Graphical Interpretation

The developed java applet offers the possibility to see a graphical visualization of the results of the algorithm into a bidimensional space. The applet offers functions like: saving and loading data training files containing points belonging to one or more classes. My application can work up to 7 classes but I will present results only for three classes as the LibSvm application can only work up to 3 classes. I will present results for different type of kernels and different degrees for my application and in comparison with LibSvm using the same characteristics. I have called my application "UseSvm_toy".

4.2.1 SVM Classification

I have tested the graphical interface for two types of kernels: polynomial and Gaussian. I will present now the results for the polynomial kernel. In order to have an equivalent comparison I have chosen *gamma*=1 and *coef0* = 2*d in LibSvm.

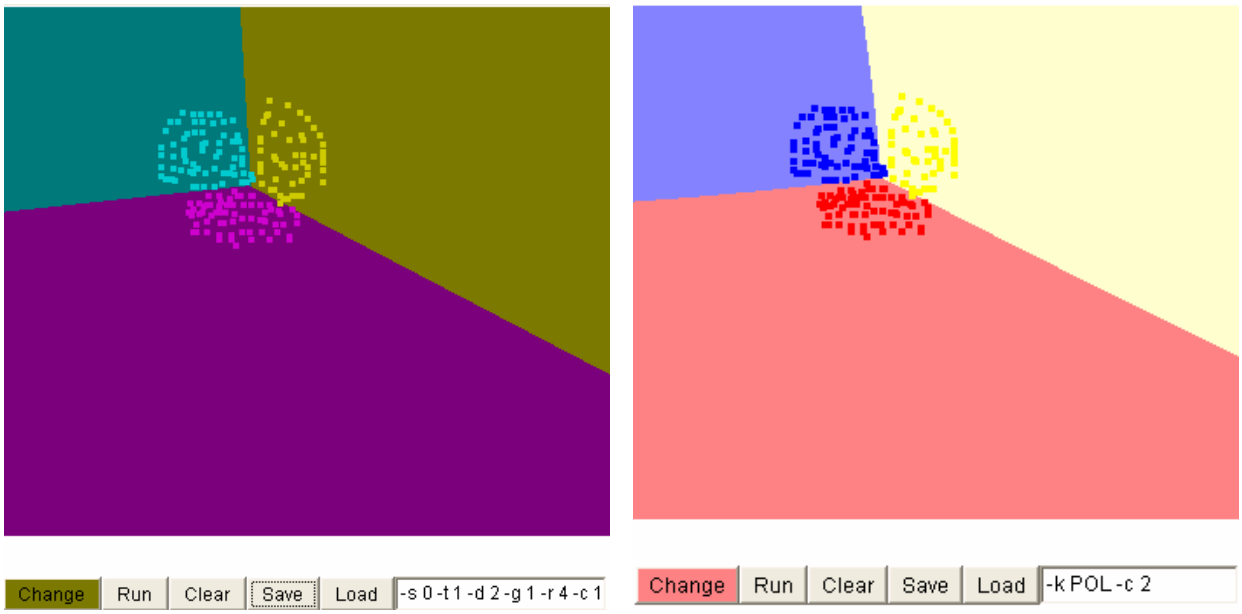


Figure 4.1- Polynomial kernel with degree 2. Left - LibSvm; Right - UseSVM_toy

For the degree 2 LibSvm has 3 errors in the testing part and UseSVM_toy has 4 errors. The number of support vectors (*sv*) for LibSVM (27) is smaller the number of *sv* for UseSVM_toy (40).

As follows I will present the results for the degree equal to 4. For this type of kernel LibSVM and UseSVM_toy work better. They also work better for degrees greater then 4.

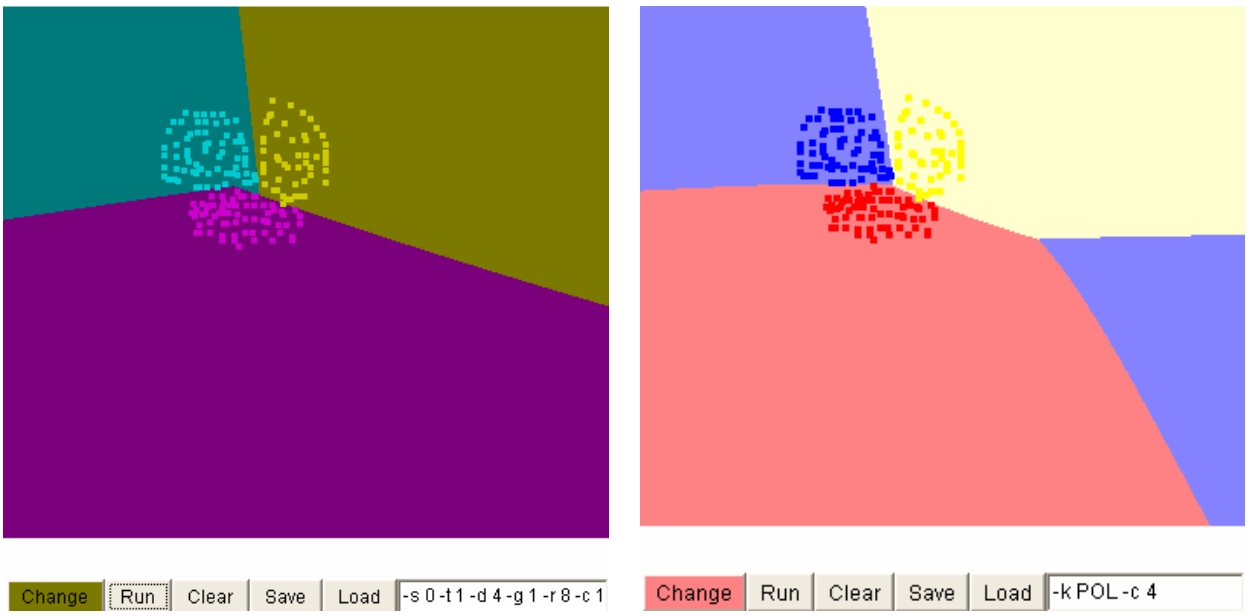
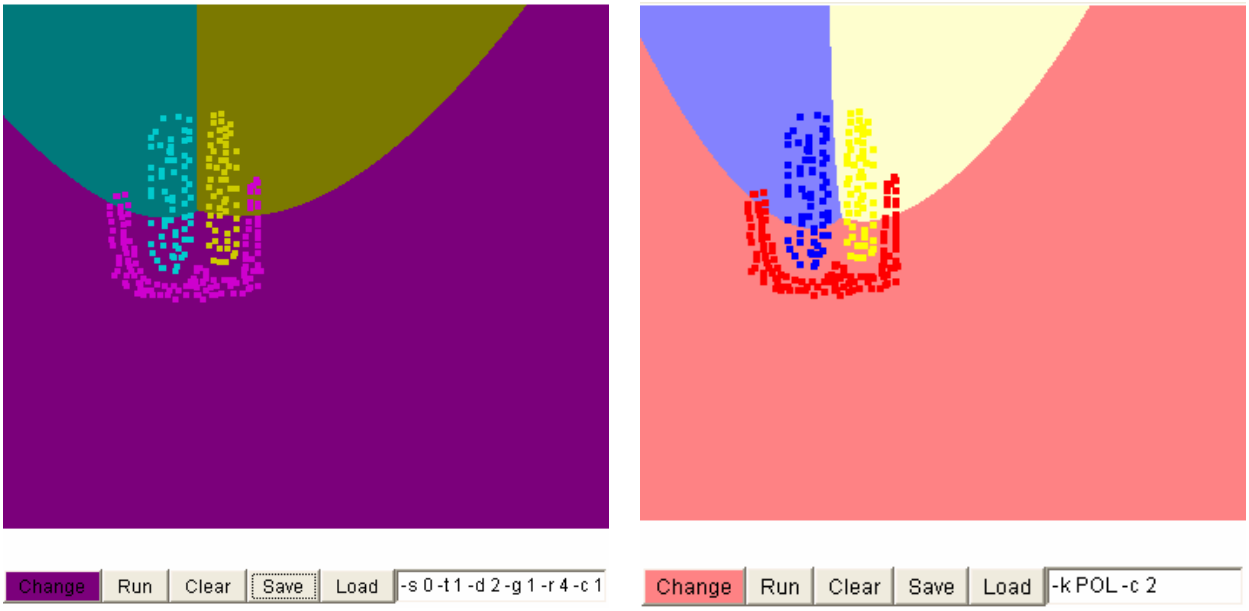


Figure 4.2 - Polynomial kernel with degree 4.

In the case of polynomial kernel if we have mixed data (the classes are very close to each other) or we have a clump of data bounded by other data, the results are very poor for small degrees of the kernel. We present these cases in Figure 4.3. We need to use a degree greater then five to obtain good results with the LibSvm. For UseSVM_toy the result are good for degree greater or equal with 4.

Using the default parameters of LibSvm leads to poor results. If we use the degree 2 it can still find 3 classes but for higher degrees it groups all the data into a single class. I have noticed that even though many researchers use no bias, the learning process is improved if we use a value of the bias equal to twice the value of the degree. It is logical that the bias and the kernel degree are interconnected as changing one leads to a shift in the points. Those parameters I think need to be connected because when changing the representing space the bias needs to be modified in the new space.



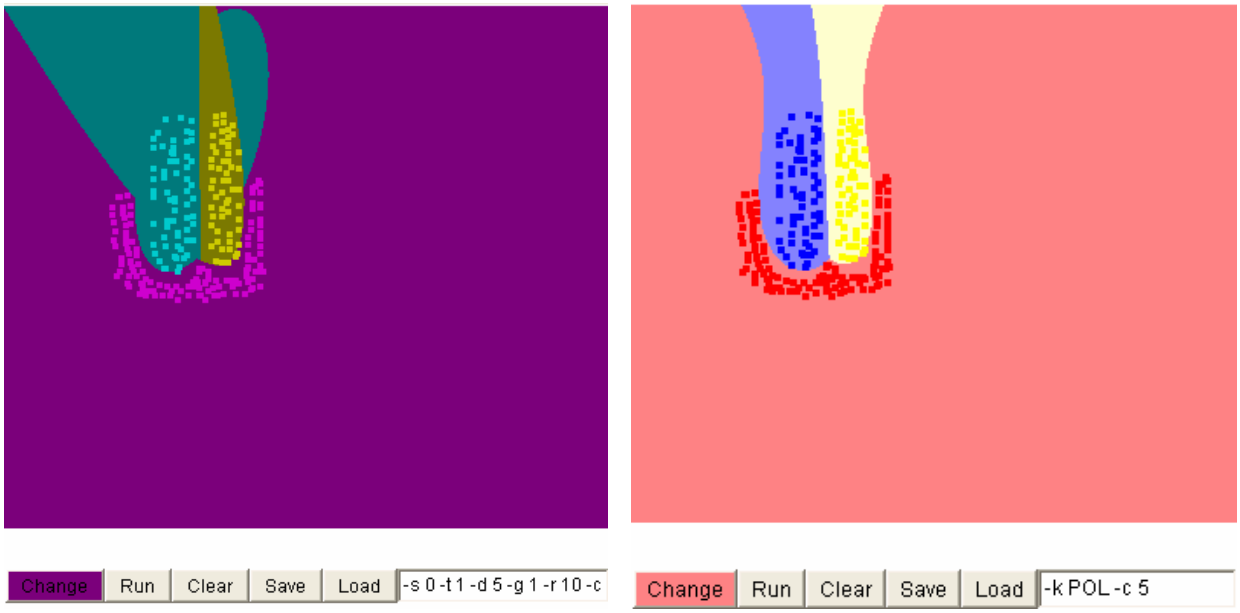
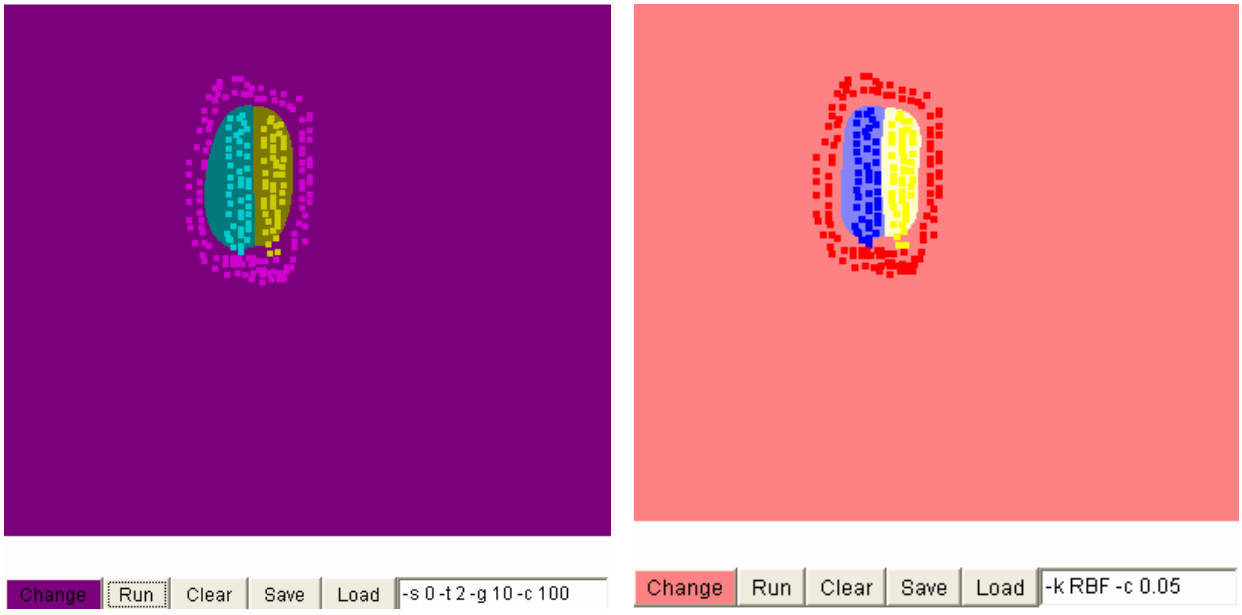


Figure 4.3 - Polynomial kernel used for mixed data

The main advantage of the polynomial kernel is that it has a smaller number of support vectors. As a consequence we have a reduced time for testing. In order that mixed data is classified correctly we need a greater kernel degree which implies a longer time for training. In almost all cases we do not know how does the data look like, thus we need to use a greater value of the degree to obtain better results or we need to try to find the optimal degree.

I will present now the visual results for the Gaussian kernel varying of the variable *C*. The Gaussian kernel is known to work better than the polynomial kernel on both separated data and overlapped data. This is why I will present only the results for overlapped data. One of the disadvantages of the Gaussian kernel is that the number of support vectors is greater than in the case of the polynomial kernel. In order to establish an equivalence between the two application we need to use $gamma = \frac{1}{nC}$, where *n* needs to be 2.



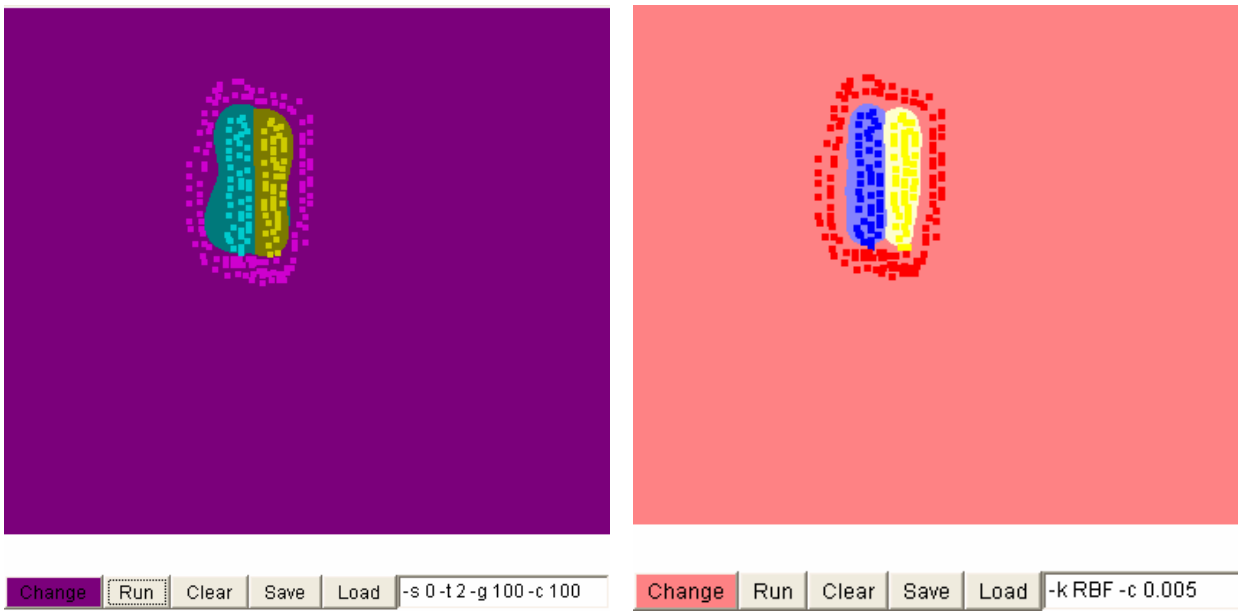
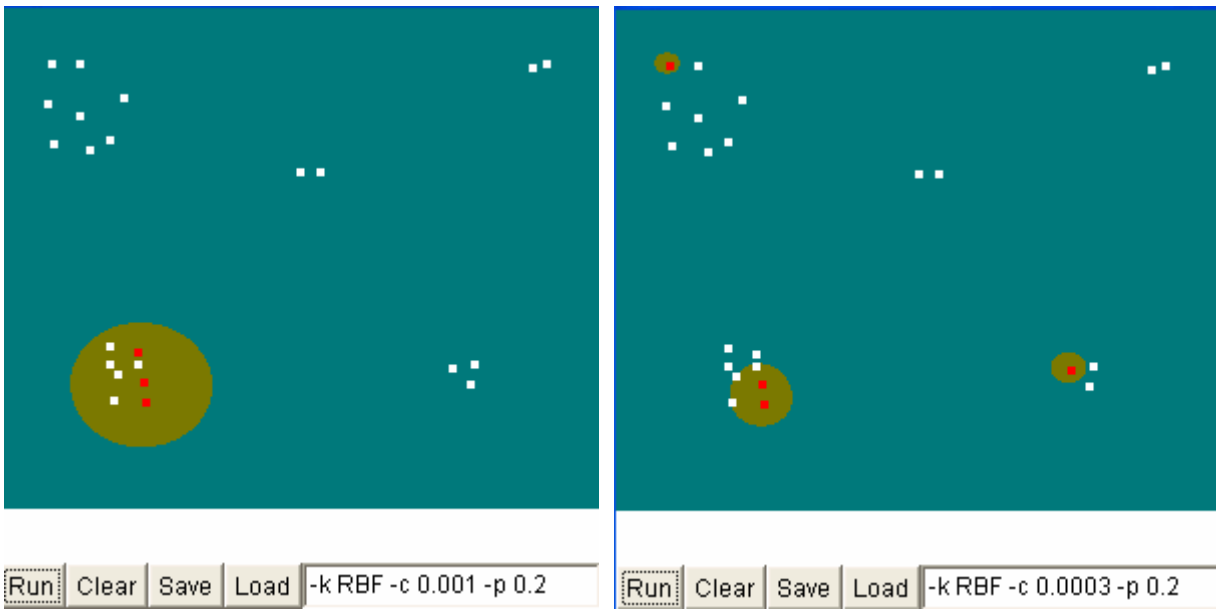


Figure 4.4 - RBF kernel with degree 0.05 and 0.005

4.2.2 One - Class SVM

In this section I will present some visual results obtained using clustering based on Support Vector Machine methods. The mathematical part of this method was presented in section 3.3. For clustering I will present results only for the Gaussian kernel because it obtains the best results in comparison with other kernels and is constantly used in the literature for clustering. The parameters that need to be modified for this application are C – the constant for Gaussian kernel and υ the percentage of data that are initially chosen. In the beginning I will present results using a java applet that offers the possibility of a graphical visualization of the results. The results obtained using testing files will be presented in section 4.4.5.

I will present now the influence of constant C on the number of classes.



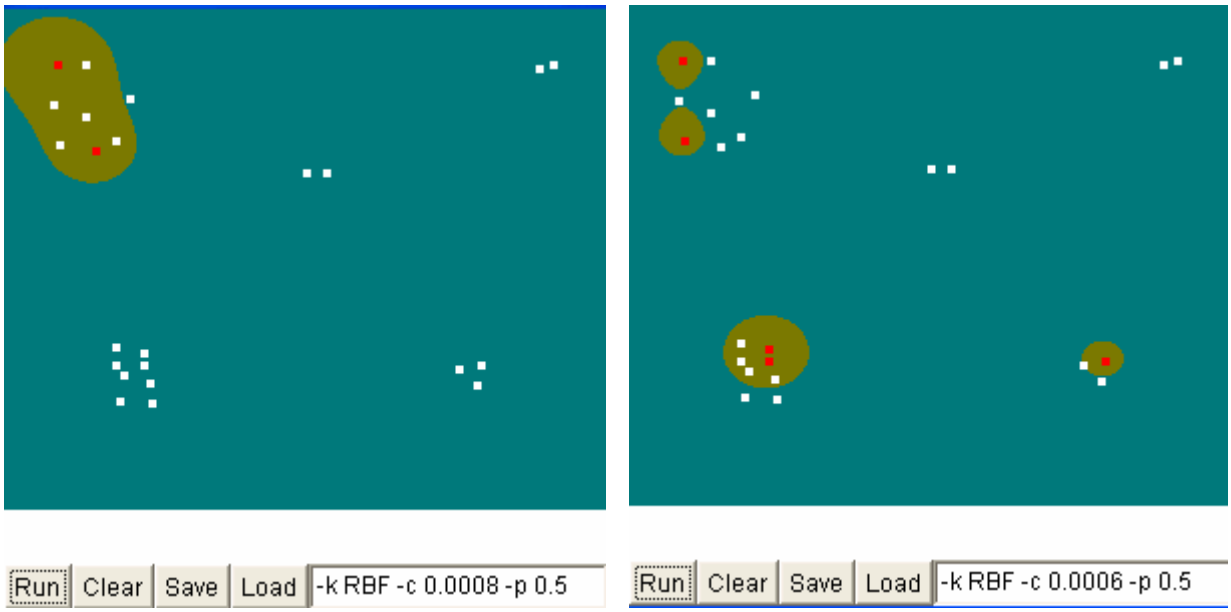
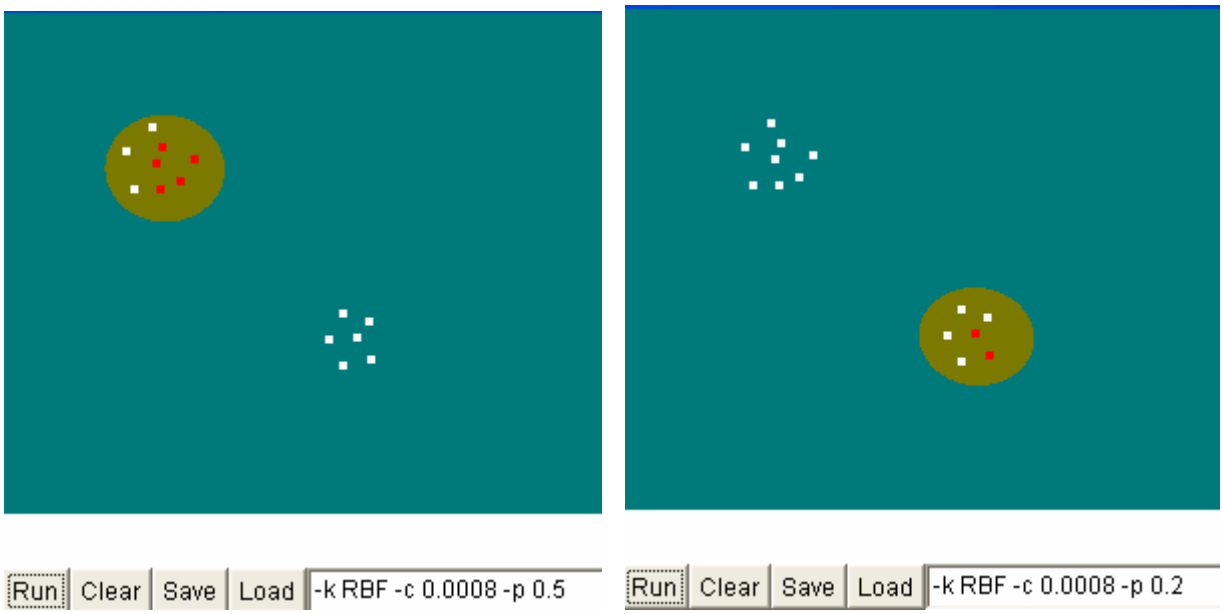


Figure 4.5 – Influence of constant *C* on the number of clusters

By modifying the constant *C* we actually modify the accuracy, a small *C* leads to smaller clusters and more disconnected clusters. As *C* decreases, the dimension of the features space where the data is mapped increases. As *C* decreases the border of the clusters is more accurate (it fits the data better).

As follows I will present the influence of the initially chosen data (I will consider percentages of data) on the results obtained in clustering.



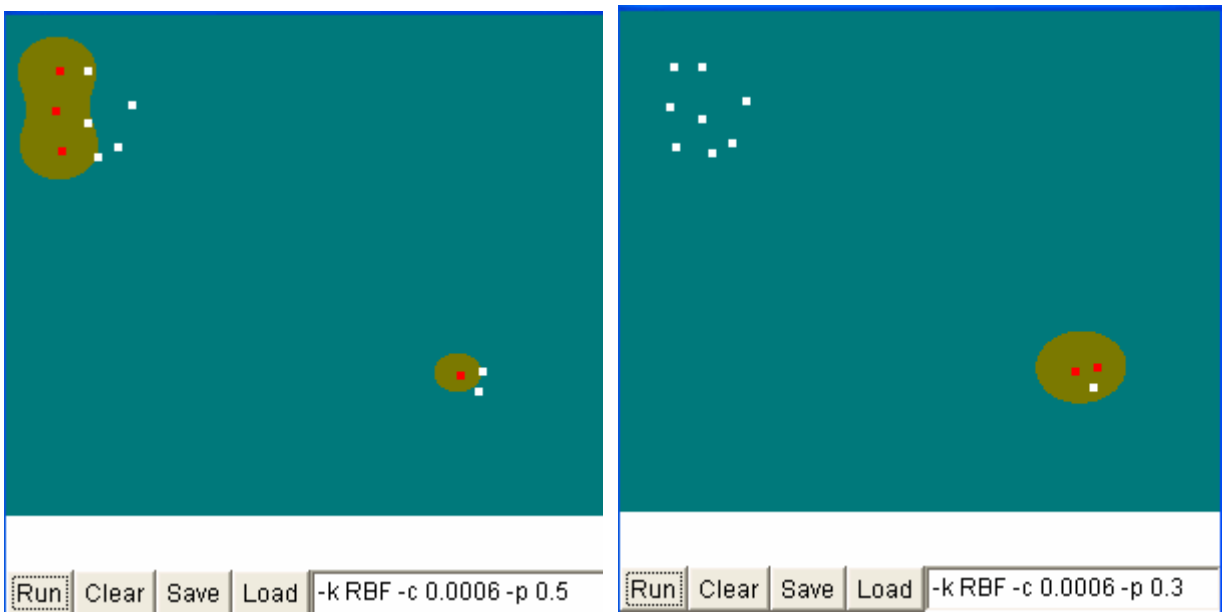


Figure 4.6 – Influence of initial percentage of data υ on the accuracy of learning

We can see that the percentage of initially chosen data has a substantial influence on the evolution of the algorithm for same value of constant *C*. When we have a large number of initial data (great value of υ) we have one great class or more and smaller classes. When we have a small value of υ we have a smaller dimensional class and a small number of classified data. I have also noticed that for value of υ greater than 0.5 the algorithm classified all data in the same cluster.

4.3 Feature Subset Selection Using SVM

In [Mla02], Mladenic et. all present a method for selecting features based on linear support vector machine. The authors compare more traditional feature selection methods, such as odds ration and Information Gain, in achieving the desired tradeoff between the vector sparseness and the classification performance. The result indicates that at the same level of sparseness, feature selection based on normal SVM yields better classification performances. First the authors train the linear SVM on a subset of training data and retain only those features that correspond to highly weighted components (in the absolute value sense) of the resulting hyperplane that separates positive and negative samples. The reduced feature space is then used to train a classifier over a large training set because more documents now fit into the same amount of memory. This idea was also presented in [Dou04]. In [Jeb00][Jeb04] are explained the advantages of using the same methods in the features selection step and in the learning step.

Following this idea I have used my algorithm of SVM with linear kernel in the step of feature selection. Thus the text mining step becomes a learning step that trains over all the features (attributes, in my case 19038) and it tries to find the hyperplane that splits the positive and negative samples. The resulting weight vector in the decision function has the same dimension as the feature space. Using this weight vector we select only the features with an absolute value of the weight that exceeds a specified threshold. The resulting set has a smaller dimension and it is then splat randomly in the training and testing set. The resulting sets are then used in the learning step of the algorithm.

For the threshold I use different values, ranging from 0.1 to 0.001. Thus for a threshold equal to 0.001 I have obtained 12936 attributes. This number of attributes is reduced to 427 for a threshold

equal to 0.1. The results in comparison with other classical features selection are presented further on. In [Jur03] the authors goes further on to grammatical semantic parsing using support vector machine adding different weights to words appearing in different parts of the phrase.

4.4 Classifying using Support Vector Machine. Implementation Aspects and Results

Because there are many ways to define the term-weight, I represent the input data in different formats in my application, and I try to analyze their influence. I take in consideration three formats for representing the data [Cha03].

a) **Binary representation** – in the input vector I store "0" if the word doesn't occur in the document and "1" if it occurs without considering the number of occurrences. The weight can only be 0 or 1.

b) **Nominal representation** – in the input vector we compute the value of the weight using the formula:

$$TF(d,t) = \frac{n(d,t)}{\max_\tau n(d,\tau)}, \qquad (4.1)$$

where *n(d, t)* is the number of times that term *t* occurs in document *d*, and the denominator represents the value of term that occurs the most in document *d*, and *TF(d,t)* is the term frequency. The weight can take values between 0 and 1.

c) **Connell SMART representation** – in the input vector we compute the value of the weight using the formula:

$$TF(d,t) = \begin{cases} 0 & if\ n(d,t) = 0 \\ 1 + \log(1 + \log(n(d,t)) & otherwise \end{cases} \qquad (4.2)$$

where *n(d,t)* represent number of times term *t* occurs in document *d*. In this case the weight can take value 0 if the word does not exist in the document and between 1 and 2 if it exists. The value of the weight is bordered by 2 for reasonable numbers of appearances of a word in a document (less then 10^9 if the logarithm used is based 10).

The ideas of separating these representations are that we can either measure if the word appears or not in the document or we can measure how often the word appears in a document by scaling the number of appearances or we can have a scaled value if the word appears in the document and 0 if it doesn't. In the last case the scaled value creates a gap between the values for no word and the value of one appearance.

Because the above formulae represent a local normalization (only for the document and not for the entire set of documents) we can also make a global normalization which introduces a measure over all the documents from the set (not all axes from the vector space are equally important). We can compute this global norm by using the IDF (Inverse Document Frequency):

$IDF(t) = \log\frac{1+|D|}{|D_t|}$, where D is the document collection and D_t is the set of documents that contain term *t*. If $|D_t| << |D|$ the term *t* will enjoy a large IDF scale factor and vice versa.

TF and IDF are combined into the complete vector-space model in an obvious way: the coordinate of document *d* in axis *t* is given by:

$$d_t = TF(d,t)IDF(t)$$

The results obtained using a global normalization (Connell SMART system) were very bad and I shall not presented them in this report.

For binary classification I chose one class and I considered all documents that were classified by Reuters in that class as the positive class and the rest of the documents (that are classified in other classes) will be considered to be in the negative class. For multiclass classification I took separately each class versus the rest and I obtained for each class a classifier that is, in fact, a hyperplane that separate a positive and negative documents for the class.

4.4.1 Binary Classification

These results are obtained on training and testing when the dimension of the set equals to 7083 documents and 19038 features taken from *Reuters2000* [Reu2000]. For all the results I have used a tolerance (that occurs in the stopping criteria, Section 3.1) equal to $1.0*e^{-6}$. In the case of binary classification I have selected class "c152" that contains 2096 documents in the positive class and the rest (4987) will be in other classes, according to Reuter's classification. I have chosen this class because its offers equilibrium between positive class and negative class. There are other classes that have more samples (for instance "ccat" has 7074 positive samples and "c15" has 3645 positive samples from a total of 7082) but in this case the classifier doesn't learn. For good learning we need to have a balance between positive samples and negative samples, maybe the number of negative samples has to be a little larger then the number of positive samples. These samples are splat randomly into training and testing files. Because we want to use the files concurrently we have generated them using the same distribution this leading to the same number of features and in the same order in both the training and testing files.

In the following charts we will present the results as percentages of correctly classified samples (classification accuracy). We will present now the influence of various feature selection methods on various types of representation and on various kernels (polynomial and Gaussian) in learning.

4.4.1.1 Polynomial Kernel

We will now analyze the influence of data representation and kernel degree, for different thresholds of the Information Gain.

- Information Gain, threshold equal to 0.08, 63 features

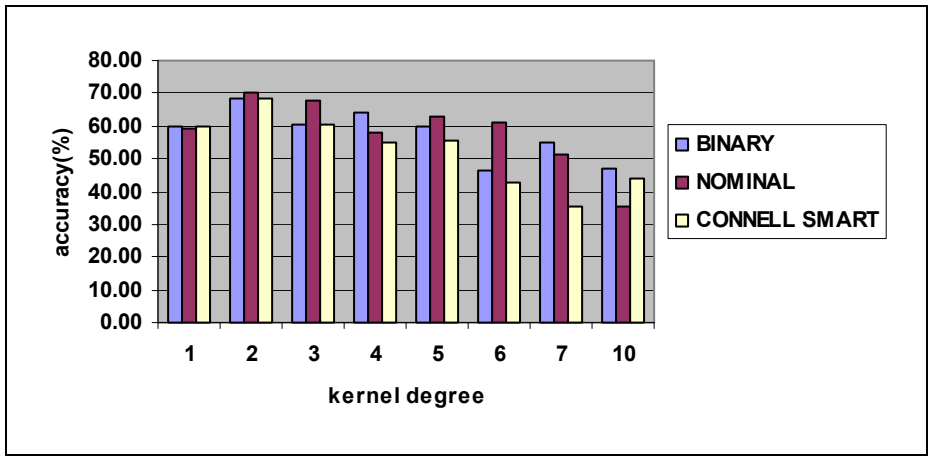


Figure 4.7 – Classification accuracy for a set with 63 features and polynomial kernel

- Information Gain, threshold equal to 0.01, 1309 features

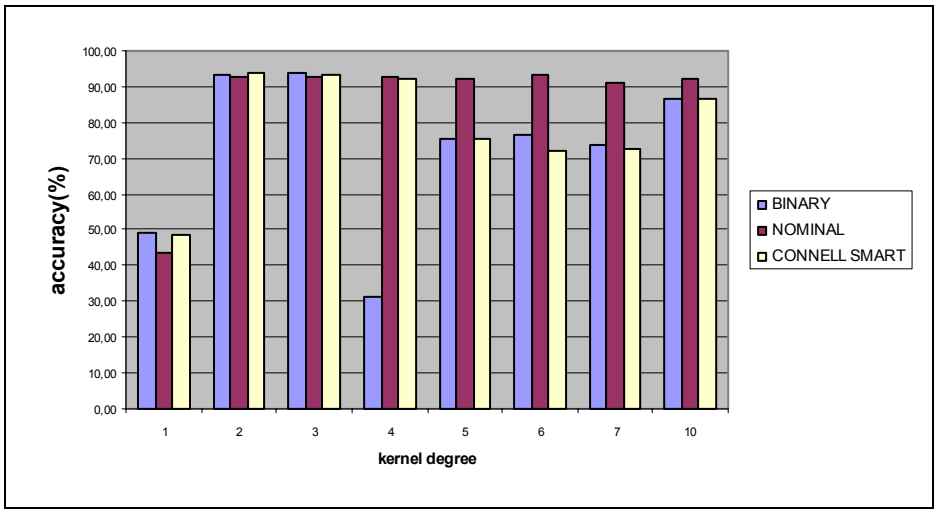


Figure 4.8 – Classification accuracy for different kernel degrees and 1309 features

- classification accuracy for 2388 attributes resulting for Information Gain with the threshold equal to 0.005

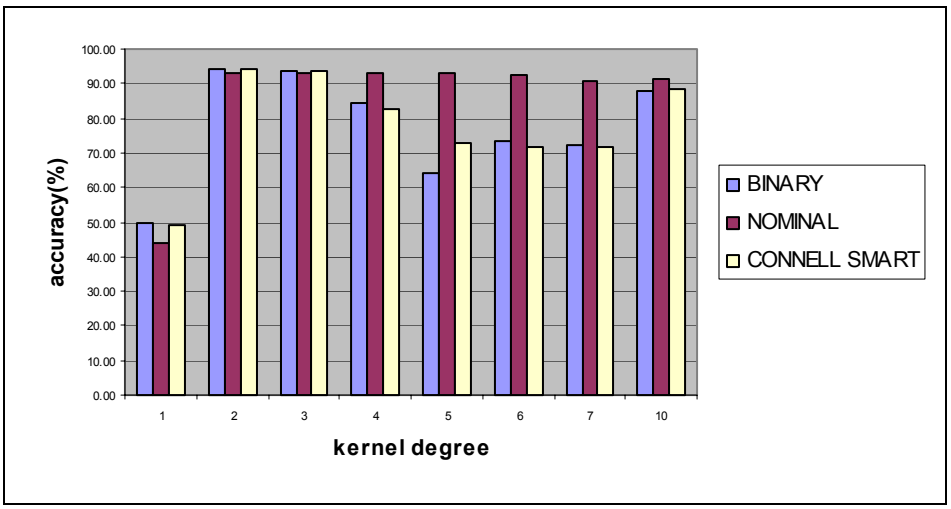


Figure 4.9 – Classification accuracy for 2488 features

- Information Gain, threshold equal to 0.001, 7999 features

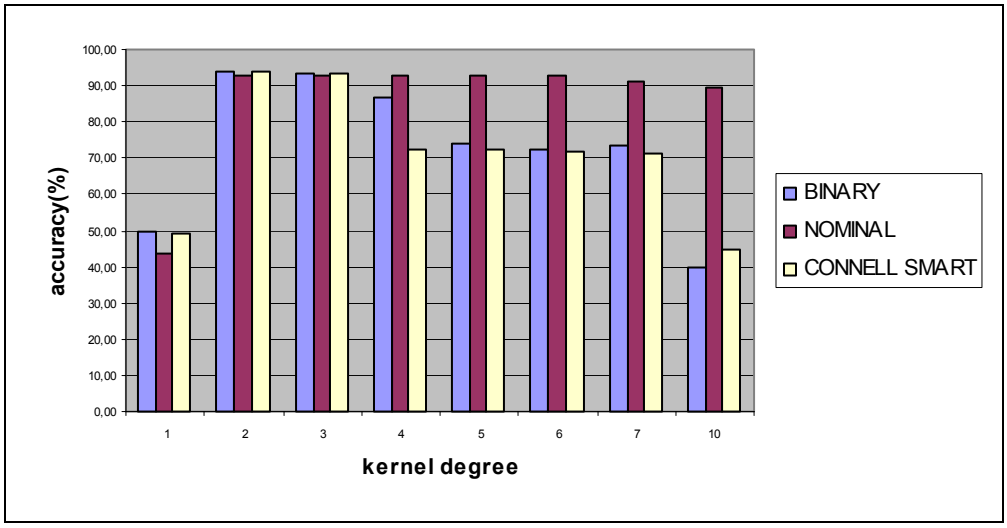


Figure 4.10 – Accuracy of classification for different kernel degrees and 7999 features

For the Binary representation of the attributes (used constantly in the literature) the results are not so good. Better results are obtained for Nominal representation and usually have the same value for different kernel degrees. The Binary representation obtains its best results for small degrees. We can also see that the type of representation is irrelevant for small values of the degree. For Nominal representation the results are pretty good even for higher values of the degree of the kernel. We can see from the charts that the best results are obtained for the nominal representation with degrees between 4 and 7.

Bellow I want to present the correlation between the number of features and kernel degree for accuracy of classification for nominal representation of the data.

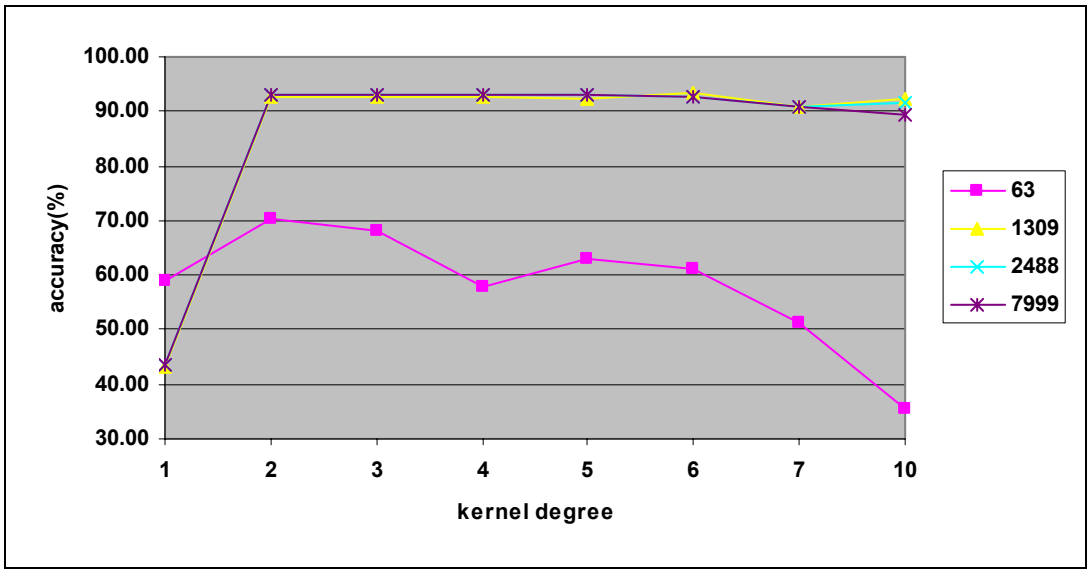


Figure 4.11 – The features number influence on classification accuracy for the polynomial kernel

Analyzing now the number of selected features for the Nominal representation we can see that if we choose less then 1% of the data (63 features) the learning results are usually poor. If we choose more features, 6% of the data (1309 features), the learning results increase significantly but if we

increase the number of features to more than 10% of the data (2488 features or 7999 features) the results don't increase so much or they even drop. As the number of features grows the learning time increases exponentially. Even though when we have a smaller number of features the data set is smaller, the results are not influenced as they are presented in percentages. The number of reduced vectors is in this case insignificant.

Because we are also interested in the number of support vectors resulted after training we have presented them in Table 4.1. As we said before, we can have situations when we have a smaller number of samples in the training or testing sets because there are samples with all the feature values zero. We will present the percentage of support vectors from the training set.

d – kernel's degree / feature representation	#attrib	1	2	3	4	5	6	7	10
# of SVs- Binary	63	40.13	64.78	66.54	27.23	46.54	71.62	56.95	55.19
	1309	50.80	31.00	36.21	57.22	65.57	63.55	68.36	33.27
	2488	50.47	33.33	37.72	57.52	66.31	75.29	77.23	33.12
	7999	35.77	41.74	61.88	77.64	69.21	81.87	10.95	35.77
	Average	44,29	42,71	50,59	54,90	61,91	73,08	53,37	39,34
# of SVs Nominal	63	38.96	62.65	67.93	82.03	16.62	11.95	83.99	64.08
	1309	56.93	23.44	23.91	24.63	25.39	37.15	33.35	32.23
	2488	56.99	25.10	25.82	25.98	26.49	28.65	35.35	37.17
	7999	56.69	26.83	28.06	28.27	29.14	41.38	36.19	34.05
	Average	52,39	34,51	36,43	40,23	**24,41**	**29,78**	47,22	41,88
# of SVs CONNELL SMART	63	40.24	63.32	62.41	14.41	7.78	49.72	68.27	49.65
	1309	50.85	30.56	34.86	33.14	59.89	71.92	76.05	28.97
	2488	50.61	32.97	37.10	55.12	62.73	75.67	76.68	30.71
	7999	50.44	35.28	41.17	59.28	79.82	81.81	82.32	17.85
	Average	48,04	40,53	43,89	40,49	52,56	69,78	75,83	31,80

Table 4.1 – Percentage of support vectors for the polynomial kernel

In the shaded lines from Table 4.1 we have stored the average of support vectors percentage resulting for different number of attributes. We can see from the charts (Figure 4.7 to Figure 4.10) that the best results are obtained for the nominal representation with degrees between 4 and 7. Comparing those results with the number of support vectors we can see that the best choice would be degrees 5 or 6 because it has the smallest number of support vectors and a high accuracy. Thus for the polynomial kernel the best results are obtained using the nominal representation of the data, and choosing only 6% from initial number of attributes and kernel degree 5.

4.4.1.2 Gaussian Kernel (Radial Bases Function – RBF)

For this type of kernel we also tested the influence of feature representation, influence of kernel degree on the accuracy of prediction, and also the number of resulting support vectors.

- Classification accuracy for 63 attributes resulting from Information Gain with the threshold equal to 0.08

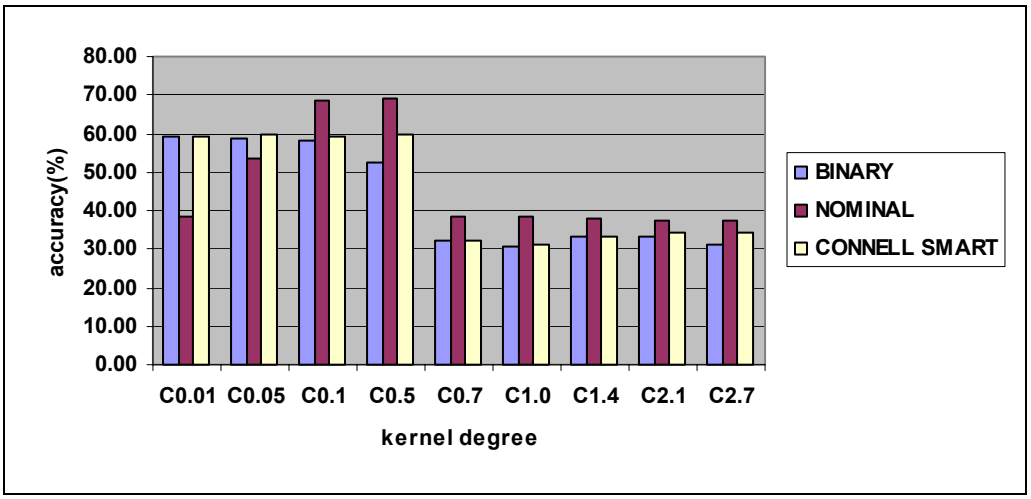


Figure 4.12 – Classification accuracy for different values of constant *C*

- Classification accuracy for 1309 attributes resulting from Information Gain with the threshold equal to 0.01

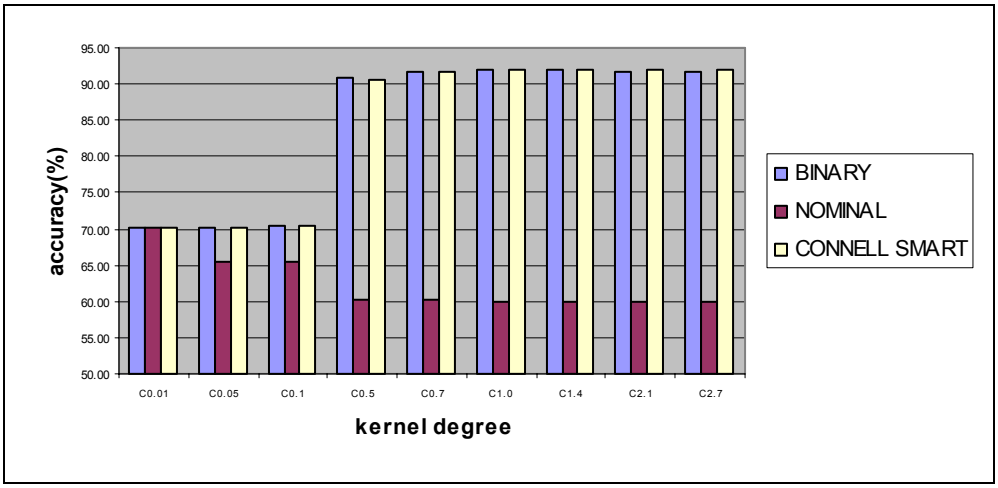


Figure 4.13 – Classification accuracy for a set with 1309 features

- Classification accuracy for 2488 attributes resulting from Information Gain with the threshold equal to 0.001

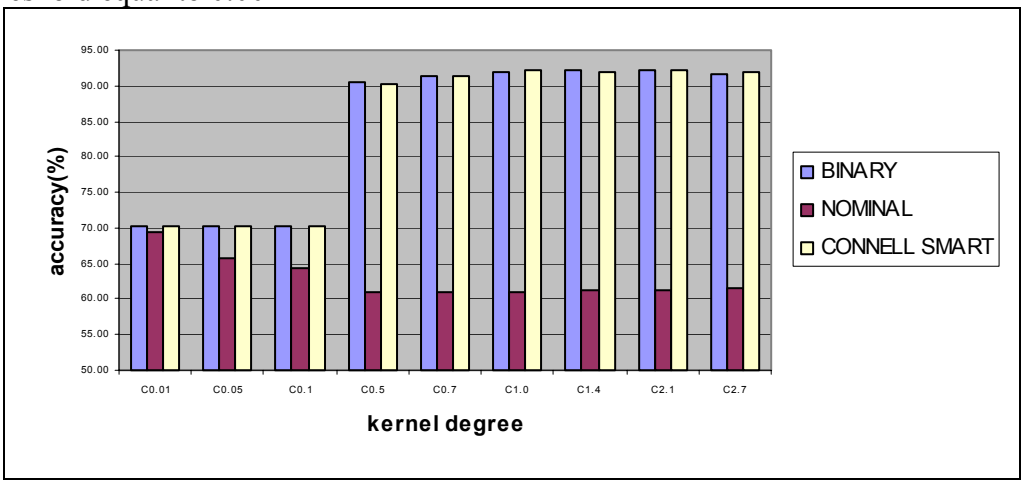


Figure 4.14 – Accuracy of classification for IG threshold 0.005 and Gaussian kernel

- Classification accuracy for 7999 attributes resulting from Information Gain with the threshold equal to 0.001

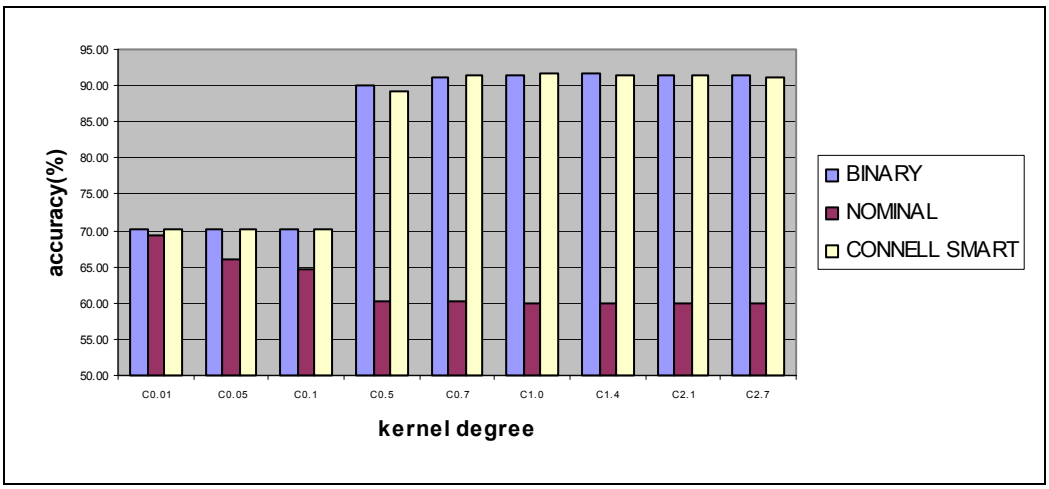


Figure 4.15 – Accuracy of classification for 7999 features

For a small number of features learning with the Gaussian kernel is also poor. The accuracy of the classification increases significantly for the width parameter *C* larger than 0.5. The Connell Smart representation obtains in this case slightly better results then the Binary representation. Comparing the results of the Gaussian kernel with the results of the polynomial kernel we notice that the Nominal representation works worse in the first case. This can be worse because it works with very small numbers. In the Nominal representation we have values in the domain [0,1] and for the Gaussian kernel this value is divided by the number of features making this value even smaller and thus reducing the importance of the value.

A better result was obtained for a value of coefficient C between 0.5 and 2.7 and Connell Smart representation of the attributes.

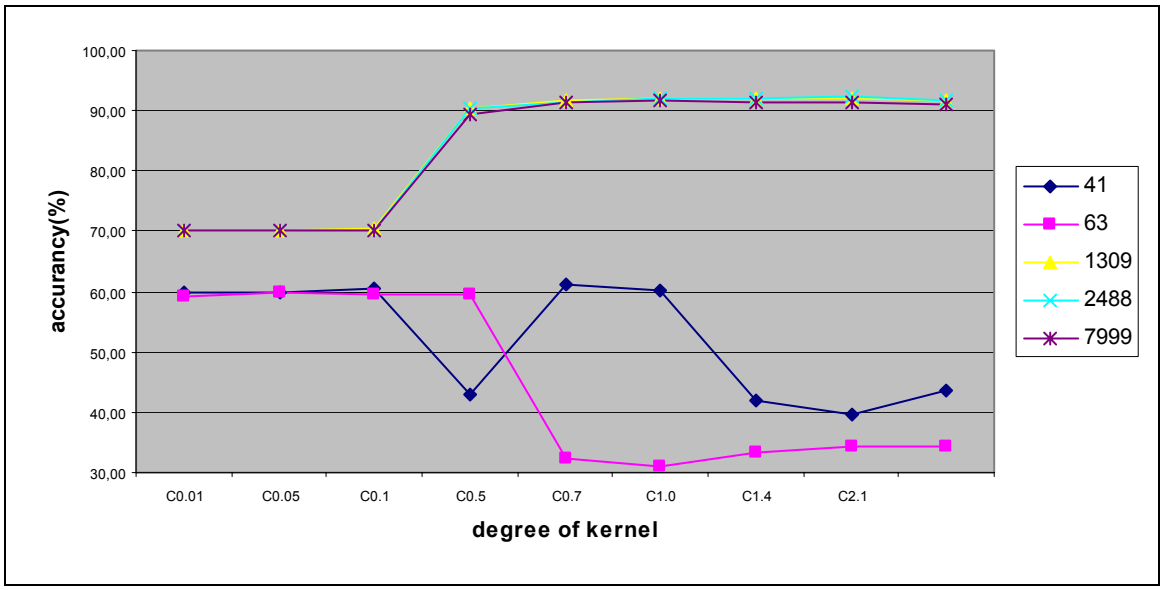


Figure 4.16 – The number of features influence on the classification accuracy for Gaussian kernel with different value of constant *C*

We can see the same tendency as in the polynomial kernel as far as the number of features involved in the learning is considered. We notice that for a small percentage of data considered (less then 1%) the learning accuracy is poor. We also notice that for a higher percentage of data (more then 10%) the learning accuracy starts to decrease. The best results were obtained for a percentage of chosen data between 6% (1309) and 10% (2488).

We will present now the resulting percentage of support vectors for the Gaussian kernel:

Coefficient C / Feature representation	#attrib	0,01	0,05	0,1	0,5	0,7	1,0	1,4	2,1	2,7
# of SVs-Binary	63	96.22	96.20	96.28	71.03	62.04	58.02	41.50	38.27	18.19
	1309	99.98	100.00	100.00	71.52	56.08	47.04	42.12	39.50	38.73
	2488	99.79	99.68	99.72	72.26	56.29	51.04	41.42	43.58	39.92
	7999	100.00	99.98	99.92	78.76	63.30	53.07	46.89	43.60	43.03
	average	99,00	98,97	98,98	73,39	59,43	52,29	42,98	41,24	**34,97**
# of SVs Nominal	63	95.57	73.55	53.04	42.39	41.20	41.13	41.42	41.92	43.11
	1309	90.17	39.47	37.84	40.32	40.64	40.89	41.34	41.87	42.25
	2488	81.85	39.09	38.23	36.32	35.75	34.75	34.63	34.18	33.84
	7999	72.96	38.52	38.65	40.79	40.96	41.36	41.59	42.12	42.59
	average	85,14	47,66	41,94	39,96	39,64	39,53	39,75	40,02	40,45
# of SVs Connell Smart	63	96.13	93.94	95.13	74.66	62.99	59.13	32.42	24.75	24.68
	1309	99.98	100.00	99.94	75.12	57.75	47.25	41.91	38.16	37.63
	2488	99.83	99.77	99.68	75.26	59.45	48.01	40.77	47.08	45.13
	7999	99.98	100.00	99.94	81.72	64.53	52.90	45.45	41.11	40.60
	average	98,98	98,43	98,67	76,69	61,18	51,82	40,14	37,78	**37,01**

Table 4.2 – Percentage of support vectors for the Gaussian kernel

Considering also the number of support vectors we think that the best choice for learning is C=2.7. We can also see that the number of support vectors for the Gaussian kernel is higher than the number of support vectors for the polynomial kernel in most cases.

4.4.2 Feature Subset Selection. A Comparative Approach

In this section I will present results obtained for two methods of feature selection Information Gain and Support Vector Machine. A similar comparison was presented by Mladenic in [Mla99] and the general idea was presented in section 4.3. The authors use the application SvmLight by Thoresten Joachims for feature selection and for classifying data [SvmLight] that is optimized for working with linear kernel. My application has the possibility of working with linear kernel but as it can be observed from the above graphics the results with linear kernel were not so good. This can occur because my application was not optimized for the linear kernel, as a different decision function was used for this type of kernel. I have only optimized the decision function for the other kernels. However I use my application with the linear kernel in the first step of features selection like it was presented by Mladenic in his article. Than I use the application in the classification step with the polynomial and the Gaussian kernel and with several values of kernel degrees. Thus I try to find out if the SVM feature selection method is comparable with the Information Gain method. For tests I have used the same data set “Subset-c152” and I have used the same three methods of data representation.

For features selection using Support Vector Machine I used different thresholds trying to obtain the same numbers of features as when I used Information Gain. I will present results comparatively for Information Gain (IG) and SVM features selection. For features selection using SVM I used the absolute value of weights obtained after training to compare it with the threshold. For the same threshold as in the IG I have obtained a different numbers of features. For example for threshold 0.08 I obtained 538 features for SVM and 63 for IG. Because there is a great difference between the numbers of features I use different thresholds to obtain an equivalent number of features. Thus for threshold equal with 0.1 I have obtained 427 features with SVM method and for threshold equal with 0.025 I have obtained 447 features. In that follow I will compare those values for all type of input data representation.

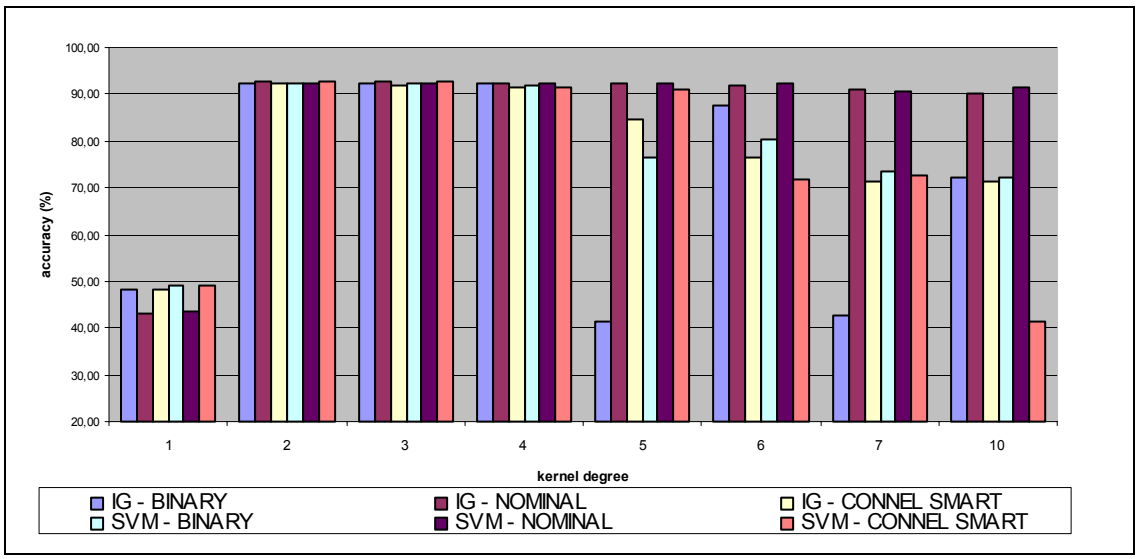


Figure 4.17 – Comparison between IG and SVM, approximately 427 features, polynomial kernel

It can be observed from the chart that in almost all cases the results obtained with SVM features selection method are a little greater than results obtained when using Information Gain method.

I will also present results for the Gaussian kernel for all types of data representation.

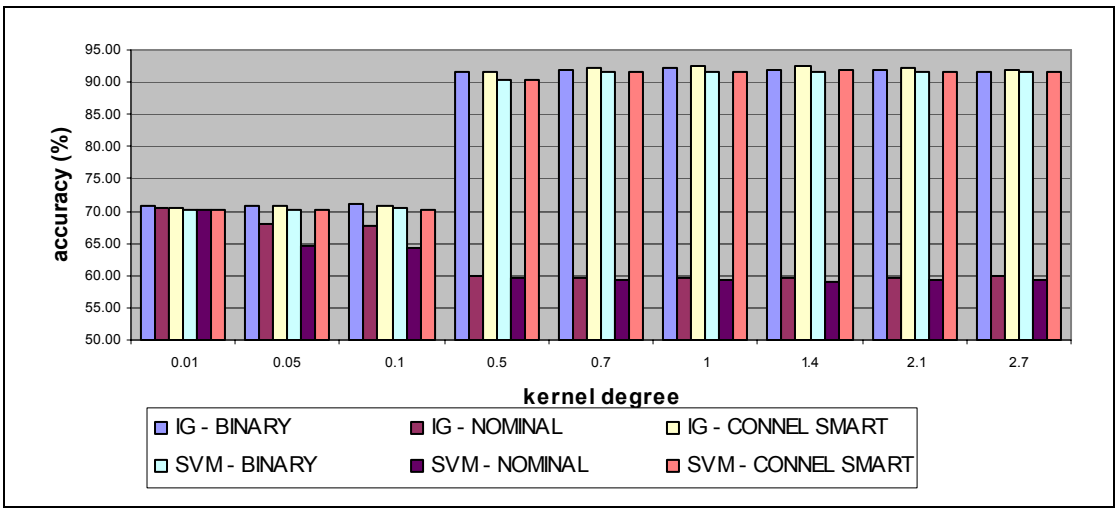


Figure 4.18 - Comparison between IG and SVM, approximately 427 features, Gaussian kernel

Another comparable case with more features is obtained when using a threshold equal to 0.01 for features selection using SVM and for a threshold equal to 0.005 for Information Gain. In the first case I have obtained 2493 features while in the other one I have obtained 2488 features. As follows I will compare those values for all type of the input data representation.

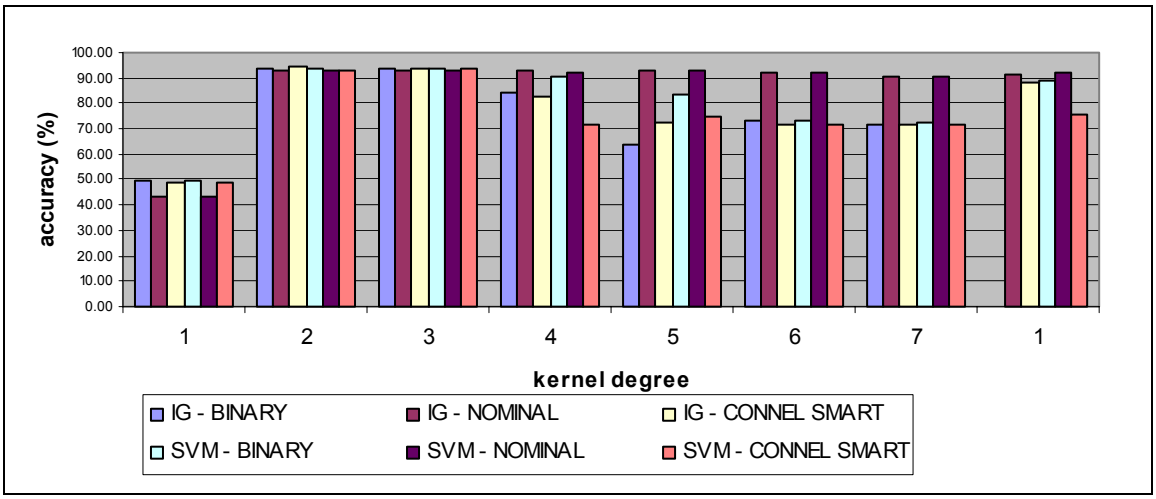


Figure 4.19 - Comparison between IG and SVM, approximately 2488 features, polynomial kernel

It can be observed from the chart that in almost all cases the results obtained with SVM features selection method are equivalent to the results obtained when using Information Gain method.
We will now present the results obtained for the Gaussian kernel and all types of input data representation.

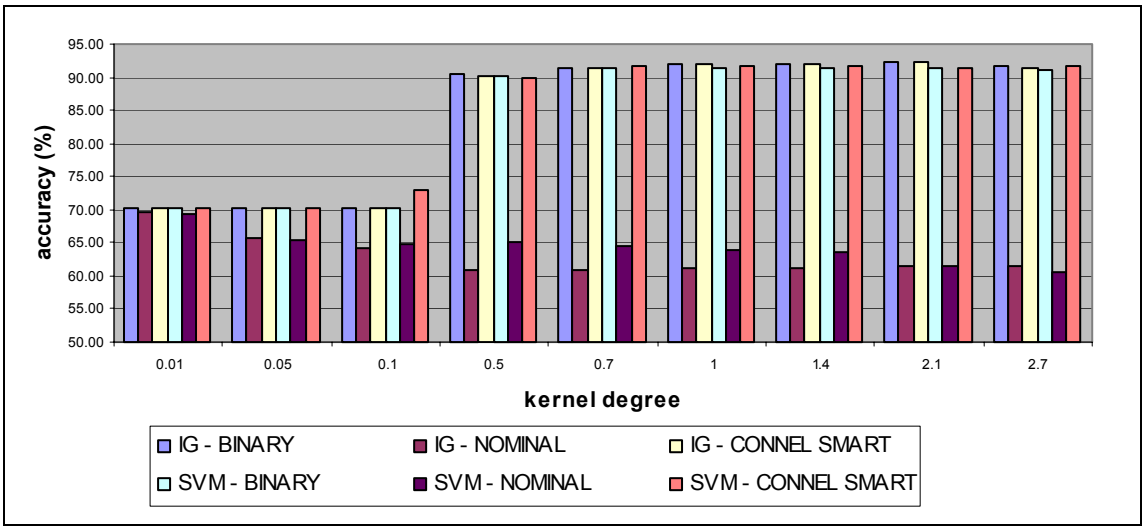


Figure 4.20 - Comparison between IG and SVM, approximately 2488 features, Gaussian kernel

As follow I will present the influence of vector dimension on the classification accuracy. For the following tests I have used all data sets (those obtained using Information Gain and those obtained using SVM features selection). I will also present the results for different kernel degrees for polynomial and Gaussian kernel. For polynomial kernel I choose to present the results for nominal representation only because in almost all cases it obtained better results. For the Gaussian kernel I choose to present the results for Connell SMART representation because it obtains better results than for binary or nominal representation.

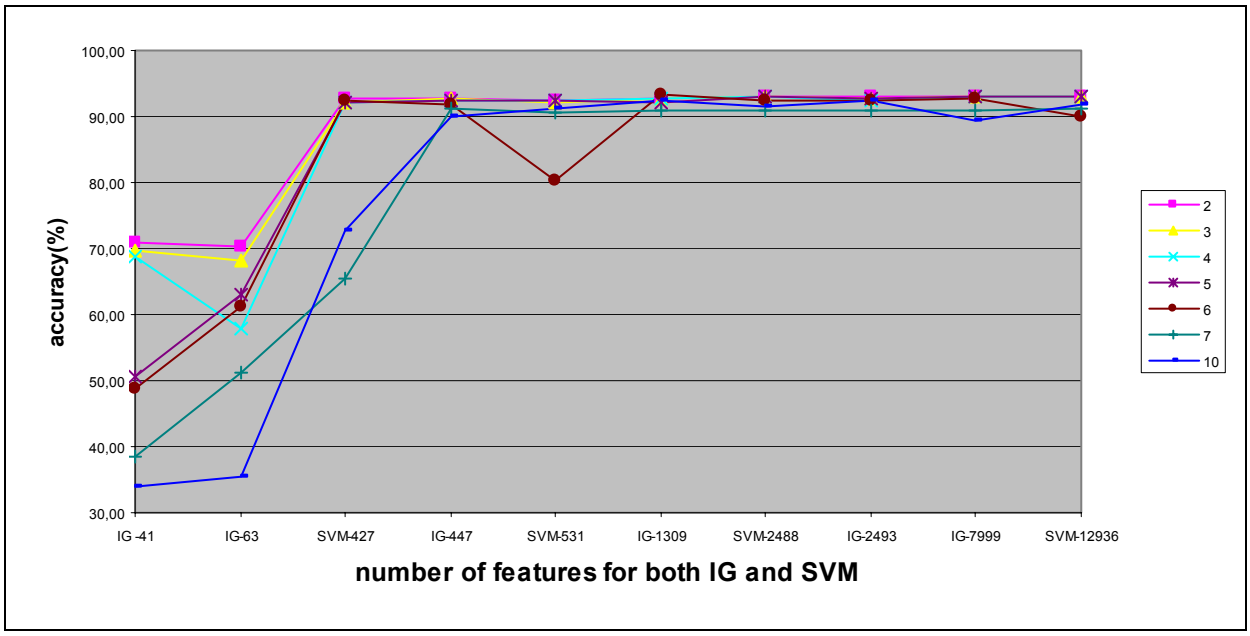


Figure 4.21 - Influence of vector size for both IG and SVM for polynomial kernel

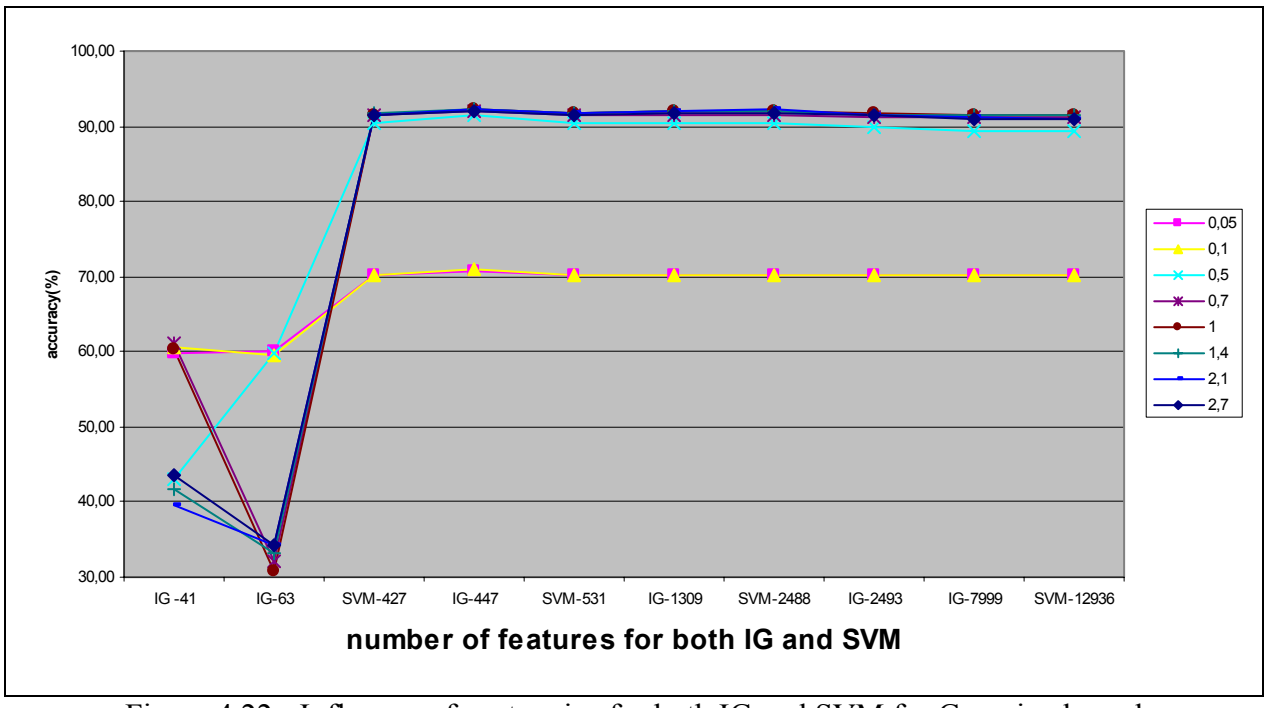


Figure 4.22 - Influence of vector size for both IG and SVM for Gaussian kernel

As it can be seen there is no big difference between these two types of feature selection. The number of features is very important. As it can be seen for a small numbers of features (41 and 63 features) the results are better for a small value of the kernel degree, and for a large enough number of features, between 427 (4% from all features) and 2488 features (10% from all features) in our case, the results are not strongly influenced by the kernel degree in both cases (polynomial and Gaussian kernel). For the number of chosen features greater then 10% from the initial set the result decrease and the time of training and testing increases. The results presented in [Mla02] on data sets created using SVM features selection method are better than the results obtained on sets created using Information Gain. This could be because Mladenic uses an optimized version of SVM for the linear kernel and his results were only presented for the linear kernel.

4.4.3 LibSvm versus UseSvm

In this section I want to present a short comparison between the results that I obtained with LibSvm and my application – UseSvm. I have used four distinct sets for these tests. The sets were obtained using SVM feature selection. I will present results for the polynomial kernel and the Gaussian kernel. I will use equivalent parameters for both applications. As LibSvm has more parameters than UseSvm I have left on default value the parameters that appear only in LibSvm.

For the polynomial kernel I have obtained the following results by varying the degree of the kernel and the number of features. The numbers following the name of the application represent the number of features used for testing:

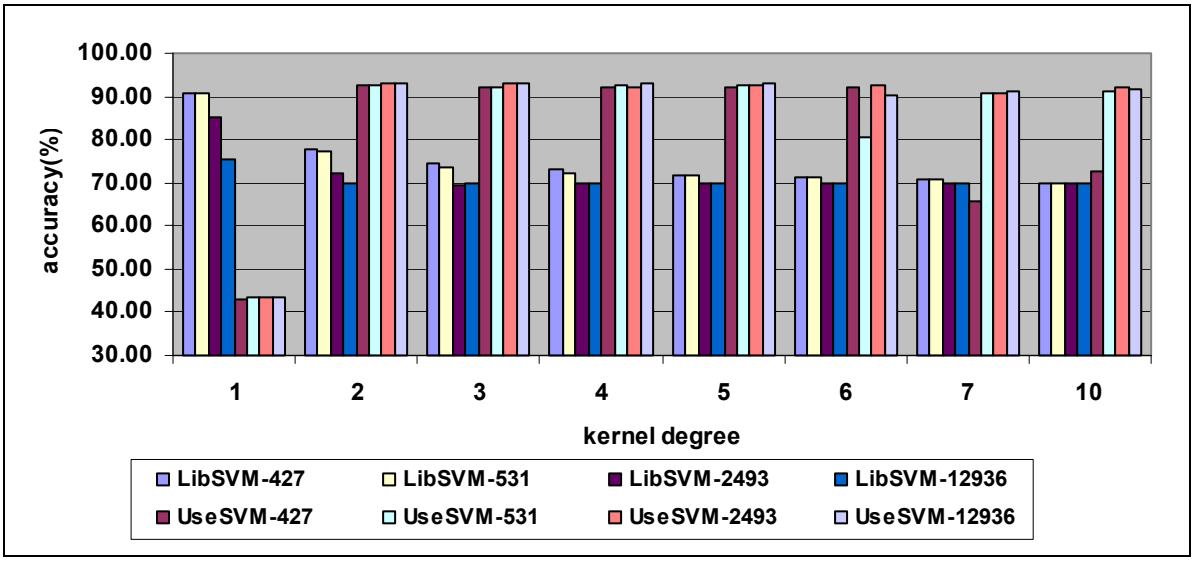


Figure 4.23 – UseSvm versus LibSvm, polynomial kernel with different degrees

We can notice that the results obtained by UseSvm are better than the results obtained by LibSvm (there is only one difference, for degree 1). LibSvm learns a little bit better for a small number of attributes but a lot worse then UseSvm. We notice little change in the accuracy of learning for LibSvm as we change the degree of the kernel (except for linear kernel).

Usually when learning with a polynomial kernel researchers use a kernel that looks like $\left(\langle \mathbf{x} \cdot \mathbf{x}' \rangle + b\right)^d$ where *d* and *b* are variable parameters. "*d"* is the degree of the kernel and in learning with kernels it is used as a parameter that helps us map the input data into a higher dimensional space. This is why this parameter is intuitive. The second parameter "*b"*, is not so easy to infer. If it is not specified the results are usually poor. In all studied articles, the researchers used them, but they don't present a method for selection. I notice that if this parameter was avoided the quality of results can be poor. It is logical that there is an influence between parameters *d* and *b* because the offset needs to be modified as the dimension of the space modifies. Because of this I suggest using "*b=2*d"* in my application and in testing I obtained good results. In the results presented above I have used the default value for *coef0* (0) for LibSvm with the polynomial kernel. So the poor results obtained in comparison with UseSvm results are justifiable.

Below I want to present results obtained when using equivalent values (same degree and same bias) of the kernel with both LibSvm and UseSvm in comparison with the default values for the kernel in LibSvm (where we only modify the degree of the kernel). I will present the results obtained for each of the three testing application in separate graphics for each chosen number of

features. In order to obtain equivalent values for those two applications I will specify for LibSvm the value of the kernel degree and the value of "coef0" that will be twice the kernel degree (specified in the command line).

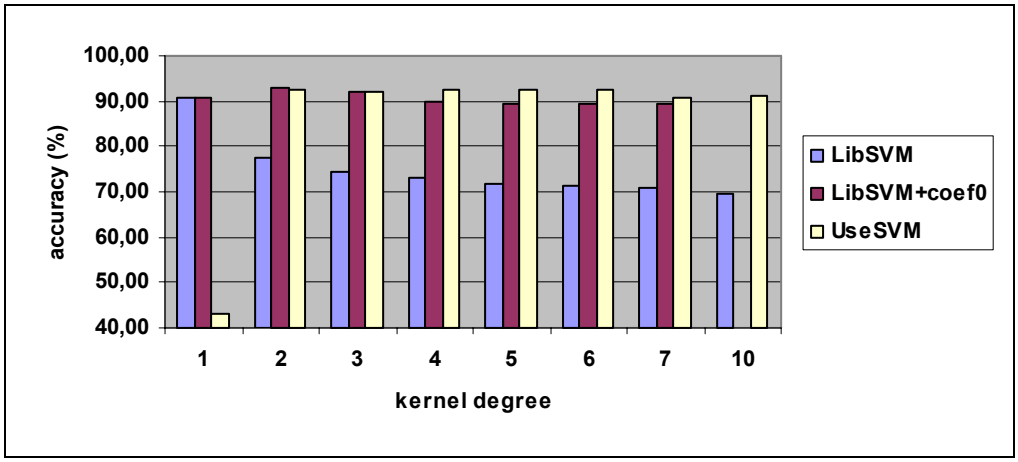


Figure 4.24 - UseSvm versus LibSvm and LibSvm with coef0, polynomial kernel, 427 features

We notice that generally UseSvm obtains the best results.

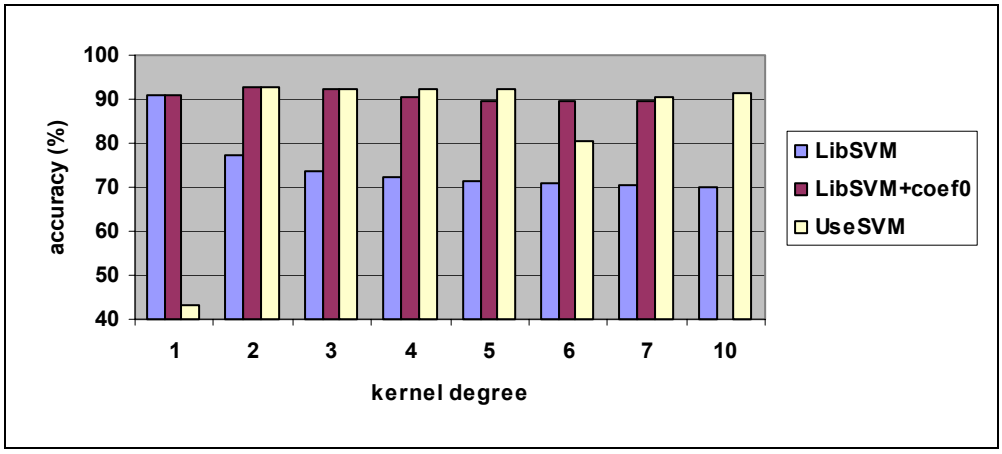


Figure 4.25 - UseSvm versus LibSvm and LibSvm with modified coefficient, polynomial kernel (531 features)

In the two charts above there is no value for LibSvm with *coef0* for the degree 10 as in testing it has exceeded time limits and there were no results (more than 4 hours using a computer with P IV at 3.2Gh). I notice that LibSVM with a small number of features and greater kernel degree doesn't work (I tested with degree equal with 10 and 15). For a greater number of features this problem didn't occur any more.

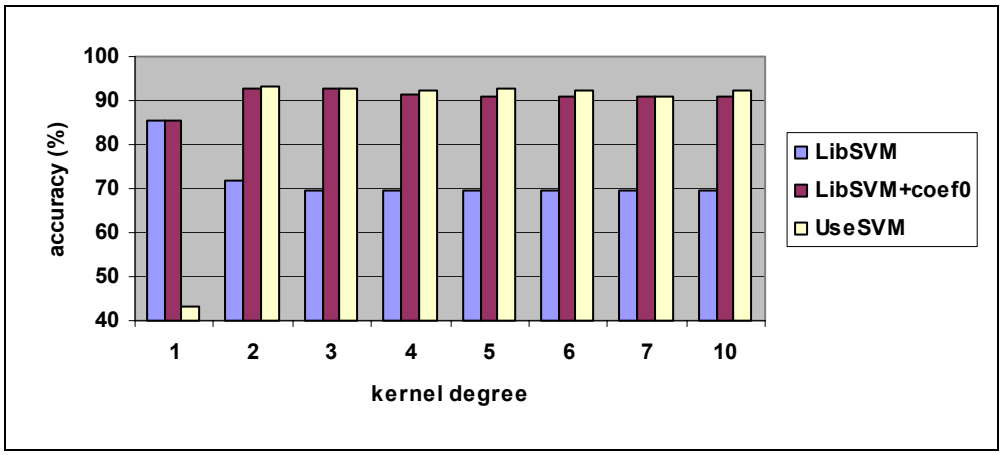


Figure 4.26 - UseSvm versus LibSvm and LibSvm with modified coefficient, polynomial kernel (2493 features)

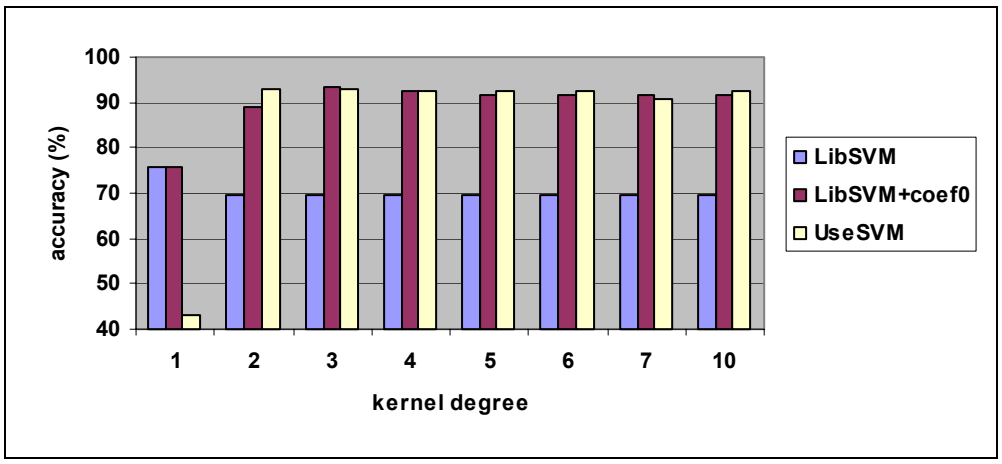


Figure 4.27 - UseSvm versus LibSvm and LibSvm with modified coefficient, polynomial kernel (12936 features)

We have obtained the best results for degrees varying from 3 to 7. No matter how many features are used we see that in almost all cases the accuracy of learning is greater for UseSvm than for the others. As for the degree equal to one UseSvm uses a different decision function we also notice the poor results.

For the Gaussian kernel we obtain the following results. There is a problem in creating the equivalence between the parameters of UseSvm and LibSvm as we have the following relationship $gamma = \frac{1}{nC}$, where *gamma* and *nC* vary. For this type of kernel in UseSvm this parameter "*n*" represents the number of vector attributes greater then zero. It is difficult to correlate this parameter from the command line. LibSvm only uses $gamma = \frac{1}{n}$ as default. So I just correlate parameter "*gamma*" in LibSVM with parameter "*C*" from UseSVM. The only case of equivalence between the two programs is obtained for default value of *gamma* in LibSvm and for *C*=1 in UseSvm. This case is presented separately as "def". Also in this case I vary the number of features. The numbers following the name of the application represent the number of features used for testing:

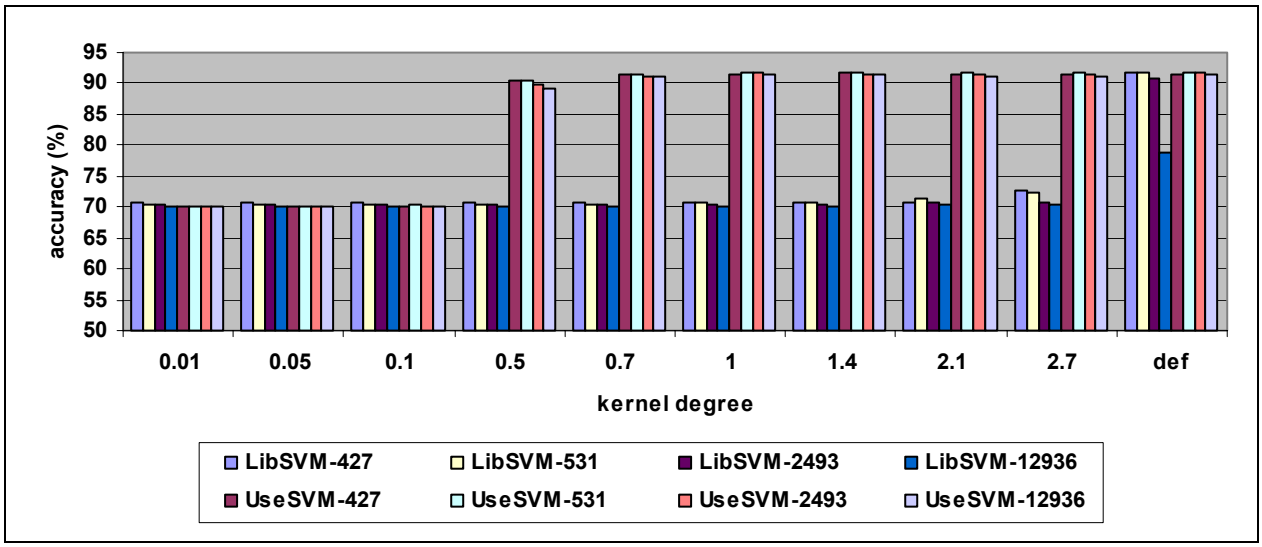


Figure 4.28 – UseSvm versus LibSvm Gaussian kernel with different value of *C*-Gaussian kernel

We can observe that generally UseSvm obtains better results than LibSvm. For the default value LibSvm obtains slightly better results than UseSvm. The best results have been obtained for C greater than 0.5.

Bellow I will present results obtained when trying to use an equivalent value of the kernel between UseSvm and LibSvm. For UseSvm parameter *n* represents the number of attributes greater then zero (can vary between 1 and total number of attributes). This parameter I will represent for LibSvm as the total number of attributes at all times. With this parameter LibSvm will obtain better results than LibSvm without this parameter.

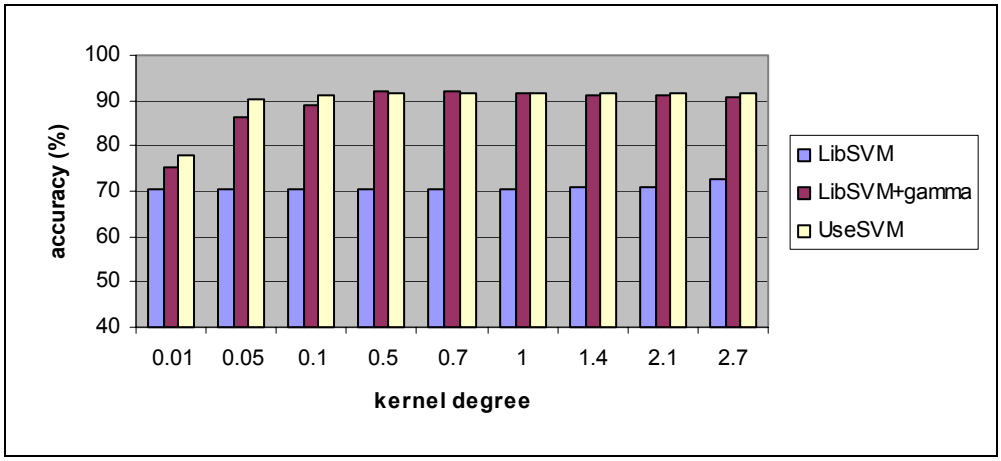


Figure 4.29 - UseSvm versus LibSvm and LibSvm with modified coefficient (RBF, 427 features)

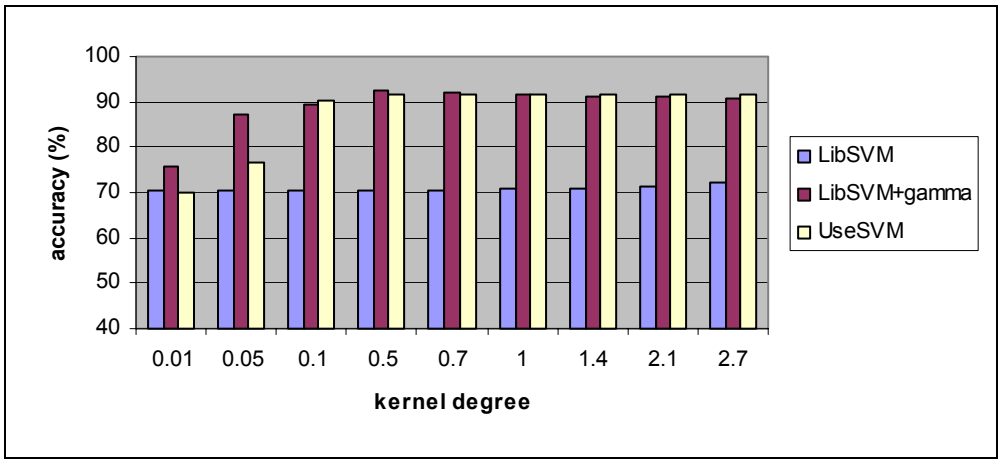


Figure 4.30 - UseSvm versus LibSvm and LibSvm with modified coefficient (RBF, 531 features)

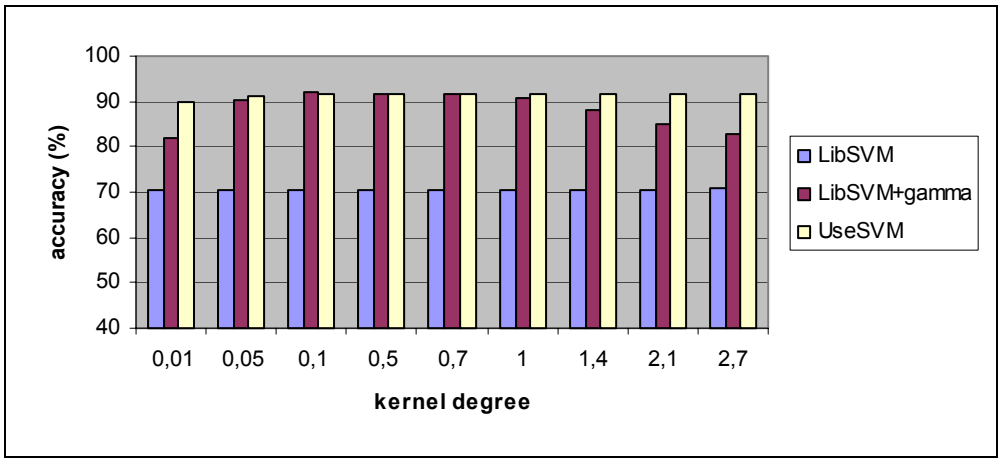


Figure 4.31 - UseSvm versus LibSvm and LibSvm with modified coefficient (RBF, 2493 features)

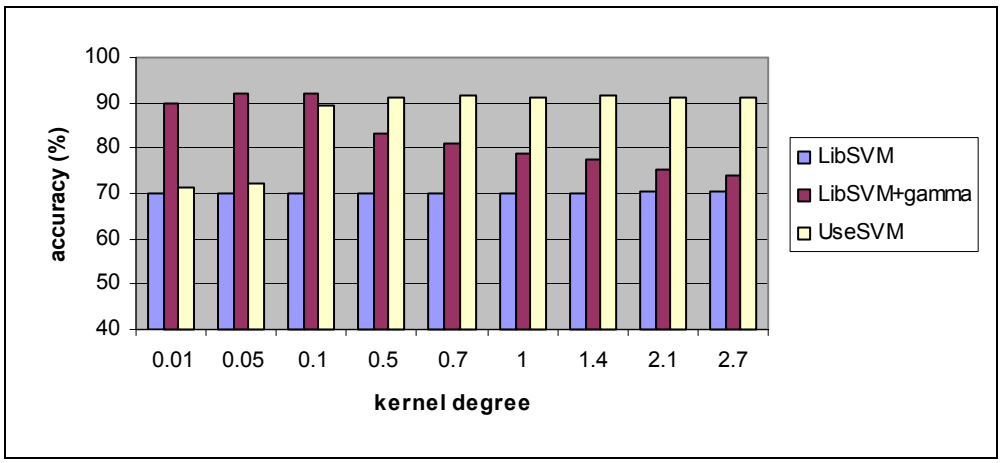


Figure 4.32 - UseSvm versus LibSvm and LibSvm with modified coefficient (RBF, 12936 features)

We can observe that UseSvm and LibSvm with modified *gamma* coefficient obtain the same results. Also we can observe that the domain for parameter *C* in which better results are obtained is larger for UseSvm, especially when we use a greater number of features. This might occur because for LibSvm we divide all the time the value of *C* by the total number of features and for UseSvm is

constantly modified. When the number of features increases the difference becomes significant because there are more features equal to zero.

4.4.4 Multi-class Classification. Quantitative Aspects

The next results are obtained on sets created using Information Gain as feature selection method. For all results I have used a tolerance (that occurs in stopping criterion, Section 3.4) equal to 0.000001. In the case of multi-class classification I tested all possible combinations "one versus the rest" using the method presented in section 3.2. Subset-c152 has 68 distinct topics. From these topics I have eliminated those topics that are poorly or excessively represented. Thus I eliminated those topics that contain less than 100 documents from all 7083 documents in the entire set. I also eliminated the topic "ccat" because it contains 7074 samples from the entire set as it is excessively represented. The elimination was necessary because with these topics we have the risk of classifying all documents into a class using only one decision function while ignoring the others. After doing so I obtained 24 different topics and I compute 24 different decision functions.

The training set and the testing set are obtained in the same manner as for binary classification. In comparison with the dataset used for binary classification in multiclass classification I use a different dataset that has the same numbers of samples as the previous dataset but it has 24 topics (not only one topic like in binary classification) and the samples are not grouped by topic. In the training part for each topic I learned a decision function. In the testing phase each sample is tested with each decision function and is classified in the class for which the value of the decision function is the greatest in absolute value. The results thus obtained are compared with the results proposed by Reuters. Reuters usually classifies a document in more than one class. If the result obtained by us belongs to the set of topics designated by Reuters than we have a correct classification. The training set contains 4722 samples and the testing set has 2361 samples. I will also present the results for polynomial kernel and Gaussian kernel and for three different types of data representation.

For some number of features I tested multiclass classification using all 68 topics and in almost all cases I obtained an accuracy of 99.98% (we have cases in which only one sample wasn't classified correctly). This can occur if there are more elements into a class (for example "ccat" that occurs in 7074 from 7083). In this case the decision function for this topic will always return the greatest value and the other decision function have no effect in multiclass classification. In this case it is easier to use a static predictor that can have 99.88% accuracy.

a) Results for the polynomial kernel when varying the number of features:

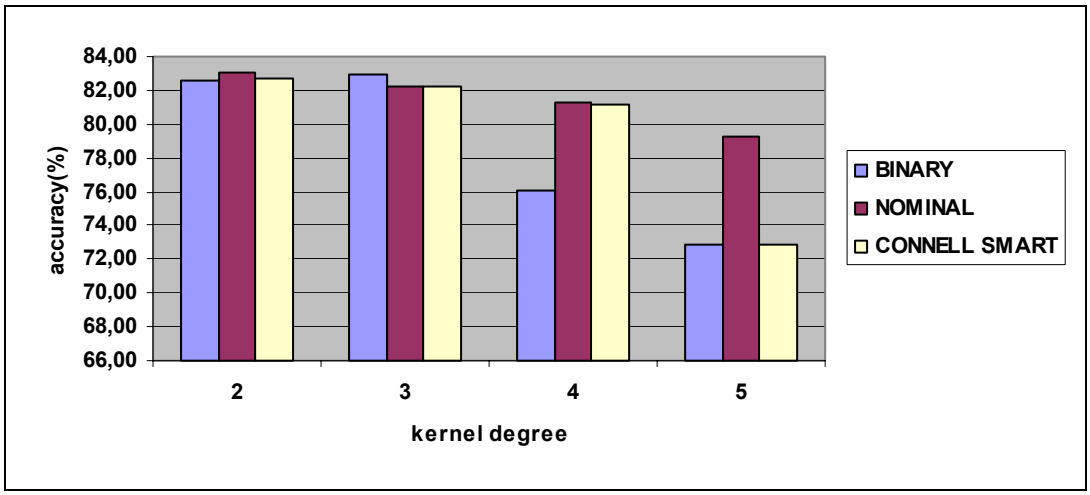


Figure 4.33 – Multiclass classification accuracy for 475 features and 24 classes

As it can be observed, the general tendency in the two class classification is maintained in multi-class classification. For a small number of features, the learning is better for a small value of kernel degree.

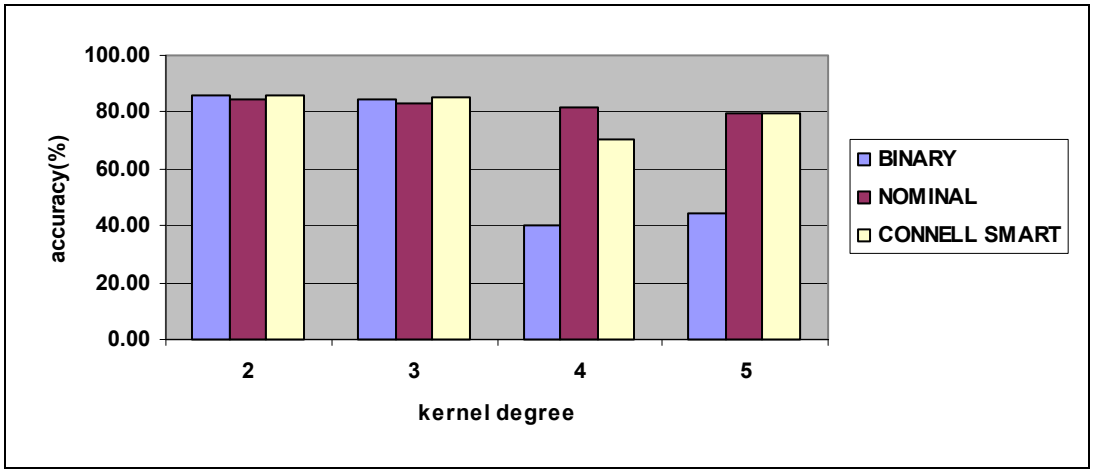


Figure 4.34 – Multiclass classification accuracy for 1309 features and 24 classes

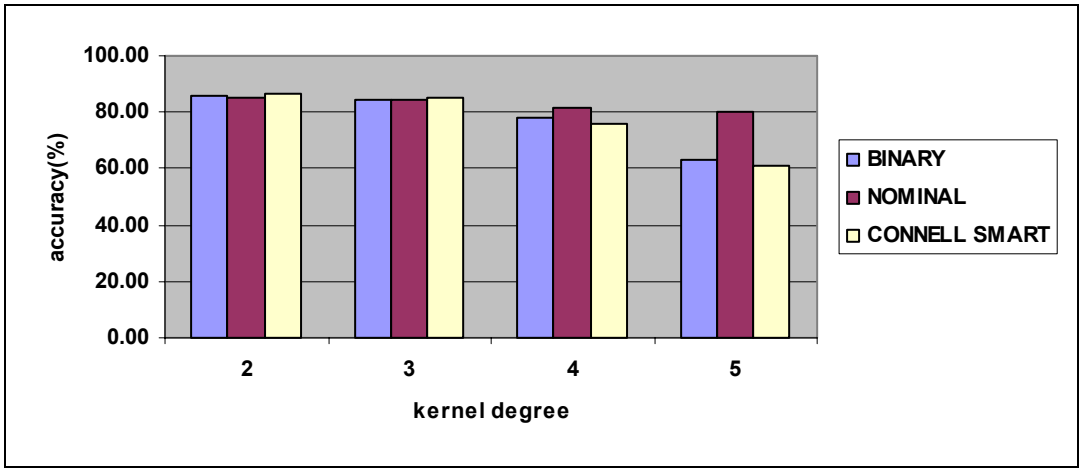


Figure 4.35 – Multiclass classification accuracy for 2488 features and 24 classes

Generally for polynomial kernel the training time is great (one test for one type of representation of input data and one value for kernel degree takes about 1 hour, using a P IV processor at 3,2Gh and 1Gb memory). Generally the results have the same distribution as the results obtained for binary classification but the results are poorer than the results obtained for binary classification with about 7%. Another interesting observation is that we obtain better results for a small value of the kernel degree in comparison with the binary classification.

b) Results for Gaussian kernel (RBF) for different number of features:

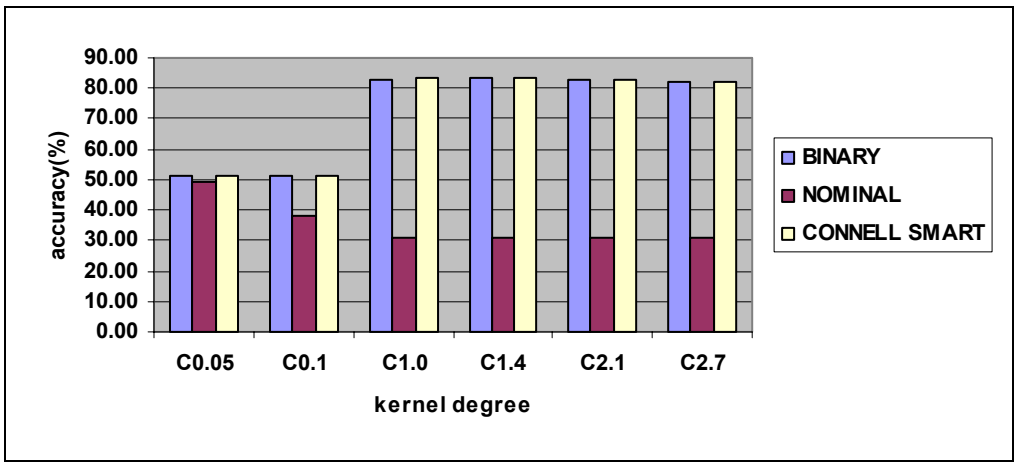


Figure 4.36– Multiclass classification accuracy for 475 features, 24 classes

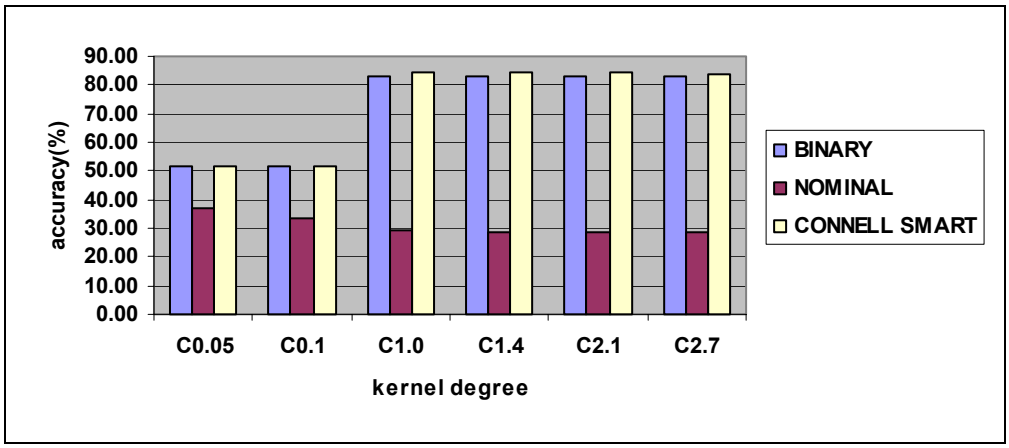


Figure 4.37– Multiclass classification accuracy for 1309 features, 24 classes

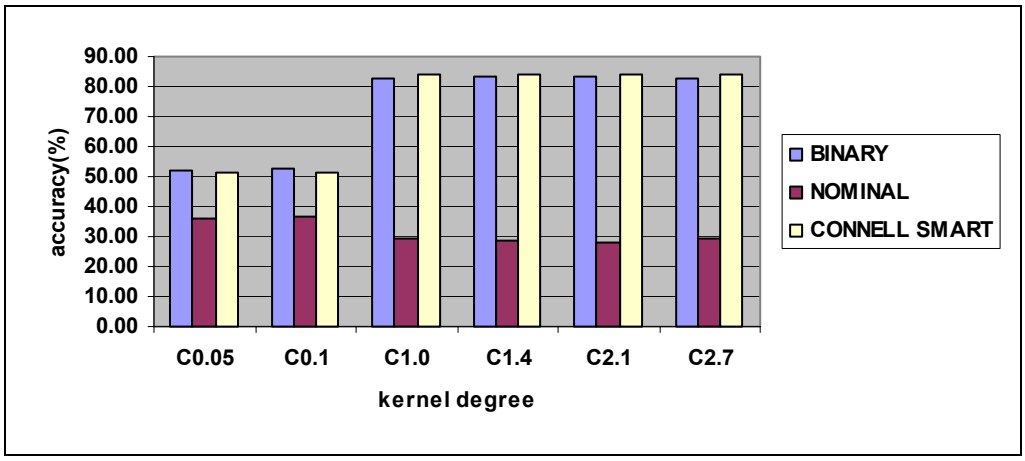


Figure 4.38 – Multiclass classification accuracy for 2488 features, 24 classes

For this type of kernel the learning time was substantially greater than for polynomial kernel (one test for one type of representation of input data and one value of the kernel degree exceeded 8 hours, using a P IV processor at 3.2Gh and 1G memory). And we have the same distribution as for the binary classification. Also the results are poorer that when using a single decision function. As for the polynomial kernel, in the Gaussian kernel the constant C decreases with the increase in accuracy, while in the binary classification constant C and the accuracy have the same tendency. The best results (87.71%) are obtained for C=1.4 in comparison to the binary classification where the best results were obtained for C=2.7.

4.4.5 Clustering using SVM. Quantitative Aspects

In this section I will present the results obtained using clustering algorithm based on support vector machine. The implemented clustering algorithm was presented in section 3.3. As the results obtained using the Gaussian kernel are better then the results obtained using the polynomial kernel and because usually the Gaussian kernel is used in clustering I will present only the results obtained with this kernel and the nominal representation of the initial data. The training and testing sets are obtained using different levels of threshold for Information Gain on "Subset–c152". There are two parameters that have a great influence on the accuracy of clustering. Those parameters are constant C in the Gaussian kernel and parameter υ that represents the percentage of data that are initially chosen as support vectors (have α parameter greater than 0). For those parameters I will initialize the Lagrange multipliers α by $\frac{1}{\upsilon m}$ where m is the number of features. I will also use two data sets, like for classification, one for training that is used to develop the hyperplane and one for testing that is used to compute the accuracy of clustering. All results are presented in percentage of samples correctly classified. From tests we notice that for a greater value of parameter υ (over 0.5 that means 50% of data are initially taken into consideration in the first step) the algorithm doesn't work so good (usually it classifies all documents into the same class). Thus I will present results for values of υ smaller than 0.5.

At first I will present the influence of parameter υ and constant C on the accuracy of prediction for training and testing sets with a 2111 number of features using nominal representation.

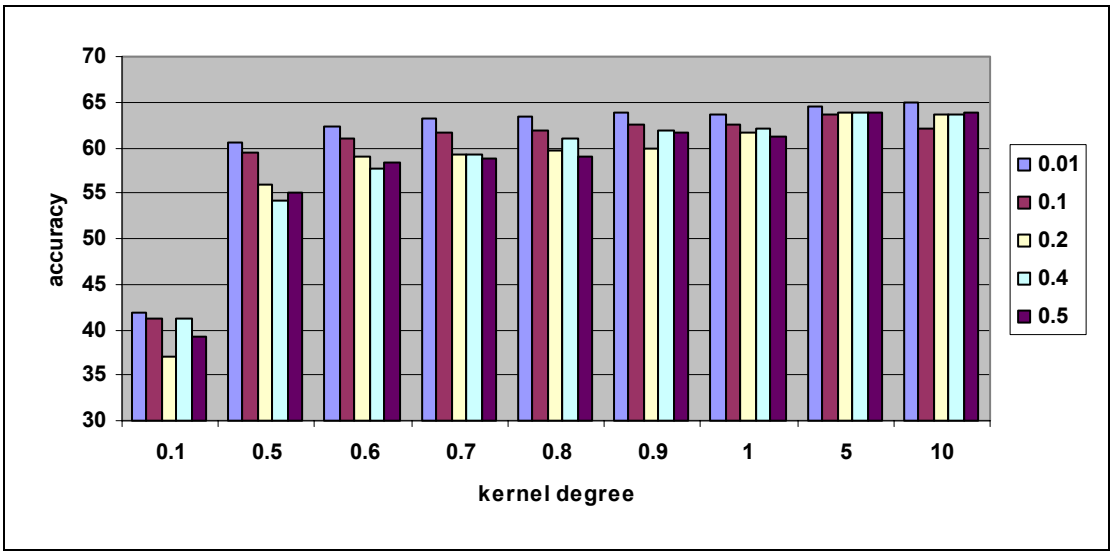


Figure 4.39 - Influence of initially chosen data (υ)

As it can be observed, the classification accuracy increases when we use a greater value of constant C and when we use a smaller number of initially chosen data. Best results were obtained for C=10 and υ =0.01 (64.99%).

I also notice that when I increase the initial number of samples taken into consideration the number of support vectors increases. This can be seen in the next table.

C / υ	0,1	0,5	0,6	0,7	0,8	0,9	1,0	5,0	10,0
0,01	27	26	26	28	28	26	25	29	29
0,1	250	250	250	250	256	256	256	264	250
0,2	499	502	501	510	508	507	507	499	499
0,4	1007	1001	997	1110	998	997	999	1009	997
0,5	1247	1246	1248	1248	1246	1246	1248	1255	1247

Table 4.3 – Variation of number of support vectors for a nominal representation of data

Below I present the influence of the number of selected features for the nominal representation of data. In this case I chose sets resulting using Information Gain method as feature selection. We use sets with 41 features, 63 features, 1309 features and 2111 features and analyze the influence of the number of features on clustering accuracy. I will present the results for C=5.0.

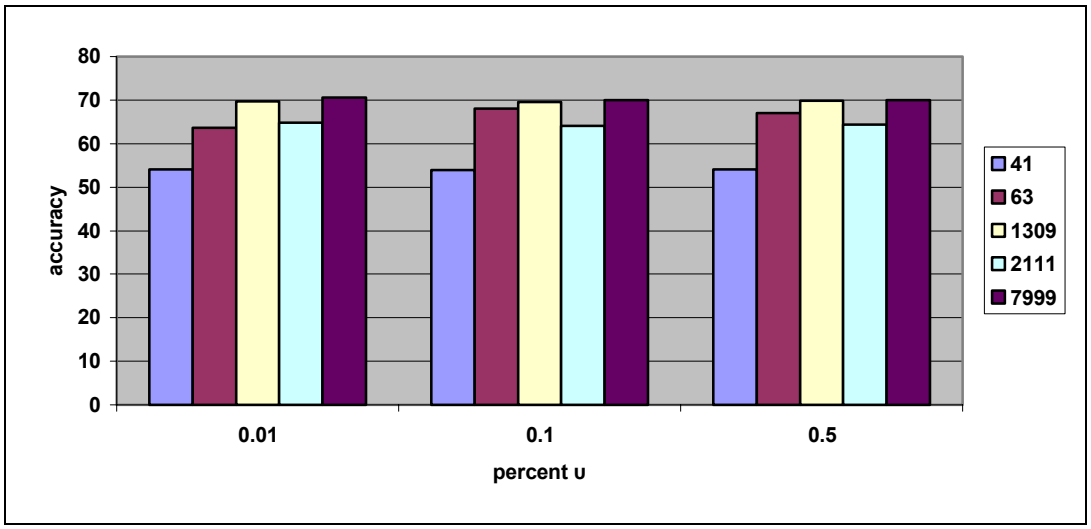


Figure 4.40 – Influence of vector size for C=5

As we can observe the best accuracies are obtained when we have a medium number of features (63 and 1309), the results decrease when we increase the number of features. We can also notice that the best results are obtained for a percentage of features similar to the one in classification (6% -10%). The accuracy decreases from 69.84% for 1309 features to 64.38% for 2111 features.

From Figure 4.39 we can observe that when we have the smaller number of initially chosen data the value of *C* doesn't mater so much as the accuracy varies only from 60.62% to 64.99%. When the value of initially chosen data increases the coefficient *C* has a greater influence on the accuracy which varies from 55.11% to 63.88%.

As a conclusion of this first tests we see that the best parameters is *C=10* for a small parameter υ (0.01). Because we are also interested in the response time we present here percentage of support vectors resulted after learning. We will present the values in percentages as the number of documents varies.

#features / υ	41	63	1309	2111
0,01	0,6%	0,6%	0,7%	0,6%
0,1	0,5%	0,5%	0,5%	0,5%
0,5	25,2%	25,1%	25,1%	25,1%

Table 4.4 – Percentage of support vectors

As we can observe the percentage of support vectors is small in comparison with the number of support vectors used for classification. Considering the best results obtained and the smallest number of support vectors the best choice is 1309 features and 1% of initially chosen data. Even if the best values was obtained for υ =0.5 (69.84%) this uses 25% of data as support vectors. For υ = 0.01 the accuracy of classification decreased only to 69.63% using only 0.7% of data as support vectors.

5 Conclusions and Further Work

The general process of classifying and clustering text data can be considered as having four steps. In the first step we process the data using text mining. The second step is the step in which we select the features. The third step is the learning step in which data processed above is used as input data in the algorithm. The last step is the testing step when we see how good the learning is. When the learning accuracy is unsatisfactory the process is reloaded. This report approaches all steps presented above and the problems that occurred in each step.

In this report I explored the problem of textual data representation. The current methods used to solve this problem have been exposed, with a special focus on statistical approaches. For this I used representation using bag-of-words for documents, keeping only the root of the word and eliminating the stopwords.

In the features selection step I compare two methods. One method frequently used in the literature Information Gain with a method presented recently by Mladenic in [Mla02] based on new technique using support vectors. As I presented in the report with the SVM method I obtained better results (92,29% for polynomial kernel and 91,78% for Gaussian kernel) than those obtained using Information Gain (91,44% for polynomial kernel and 91,44% for Gaussian kernel). This result is not in contradiction with the results reported on text learning problems, for example by [Mla02]. However the results obtained are not so good as in the specified article because, I think that my implementation of this type of algorithm is not so optimized (linear kernel). This can be seen in section 4.4.3 where I compared my application with a usually used application of SVM (LibSvm). Another problem that can make feature selection with SVM be not so good is that in the features selection step I used all classes (63) and as I presented in section 4.4.4 this can mislead us. As we can see from the results it is not necessary to use more features in the learning process. Actually using more features can decrease the accuracy of learning.

In this report I presented separately the classification and clustering process. For these processes I used a method that was used with great success in the last years in solving nonlinear separable problems. This method is based in fact on learning with kernels and support vectors. Using this method I compared three types of data representation (binary, nominal and Connell Smart representation) for two types of kernels (Polynomial and Gaussian). Considering both points of view I notice that for binary classification:

- the best results for polynomial kernel were obtained when using nominal representation of data for kernel degree 6 (93.22%);
- the best results for Gaussian kernel were obtained using Connell Smart representation for the value of constant C= 2.7 (92.29%);
- polynomial kernel usually has a smaller number of support vectors (29.78% for degree 6 for polynomial kernel versus 37.01% for C=2.7 for Gaussian kernel) and for this type of data we obtain better results with polynomial kernel;
- it can be observed that for polynomial kernel binary representation usually obtains better results for a small kernel degree in comparison with nominal representation that has good results for all kernel degrees.

Another important issue in classifying documents is the number of features that are used to represent the set of documents. A greater number of features selected can make the algorithm run slowly and the accuracy decreases. We can say that:

- by choosing a small number of features from the initial set, in our case less then 3% (41, 63, 427) the accuracy of learning is powerless;

- by choosing a value between 3% and 10% (2488) the accuracy has a spectacular improvement from 33,56% to 93,10% for polynomial kernel and from 30,91% to 92.08% for Gaussian kernel;
- by choosing a number of features greater than 10% the accuracy has little improvement (only 0.5% better in few cases) or it usually decrees, but it certainly increases the time needed for processing the data.

I also compared my application with a frequently used application in the literature (LibSvm) and I obtained pretty good results especially when for LibSvm I specified from command line parameters to do the kernels identically for both application (92,59% for UseSvm and 90.94% for LibSvm for polynomial kernel). I notice that the power is in modeling the kernel. Thus running LibSvm with all command line parameters specified default the difference is great and the accuracy increases from 69.66% to 90.94% for polynomial kernel and from 70.65% to 88.07% for Gaussian kernel. These specific forms of the kernels allow in almost all cases maximum value of classification accuracy. This study about correlation between kernel coefficients (degree of kernel and bias) is not presented in this report.

In multiclass classification we notice that we have the same tendency as for binary classification. Usually for multiclass classification better results were obtained for a smaller value of kernel degree. For the polynomial kernel the best results were obtained for degree 2 (86.36%) and for Gaussian kernel for C=1.4 (84,71%). Better results for multiclass were also obtained for nominal representation and polynomial kernel and for Connell Smart representation and Gaussian kernel like in the binary classification. An interesting conclusion is that for multi-class classification the results are less powerful than the results obtained for binary classification although we expected to have better results.

Regarding the results obtained in clustering in comparison with classification the accuracy decreases considerably. The accuracy of clustering is influenced by the constant C of the Gaussian kernel and the percentage of initially chosen data υ. Even though the number of tests ran for clustering is smaller then the number of tests ran for we notice that the best results are obtained when we choose a smaller percent (6% of features) of the data and they decrease when we choose more data. The number of support vectors resulted after clustering training is a lot smaller than the number of support vectors resulted after classification training. The best parameters for the Gaussian kernel are C=10 and υ=0.01 if we tuck into consideration the best results (69.63%) and the small numbers of support vectors (0.7%).

Because almost all available data are in fact unlabeled data, the tendency in the last years is to develop the clustering application rather than to develop the classification algorithm. For using the classification algorithm almost all data needs to be labeled by human effort consequently. By using classification the results are higher that the results obtained with clustering, as it can be seen in this report. An interesting onset was presented by [Xu02]. The authors use classification and clustering algorithms at the same time in order to use both labeled and unlabeled data and so substantially reduce the number of documents that need to be labeled by human.

In further experiments I will try to combine this two methods implemented thus I can use labeled and unlabeled data into a hybrid classification algorithm. An idea is to change the primal step from clustering, when we chose an initial value of data and initialize the Lagrange multipliers $\alpha_{i,}$ into the classification step. In this step a small number of labeled data are presented to a classification algorithm to obtain the α_i coefficient that are used as initial values for clustering. Those coefficients are then used into a clustering algorithm using more unlabeled data.

A major issue that occurs in all classification and clustering algorithms is that they are reluctant to fitting in real spaces. For instance they have a problem when they have to deal with new documents for which none of the features are in the previous feature set (the product between the new features set and the previous feature set is the empty set). There definitely are methods of updating the algorithm. The newly obtained algorithm will somehow classify the documents by creating a merge between the feature sets. This problem occurs as the training set can not contain the roots of all existing words. Feature selection methods choose only interesting features. As we see in this report the algorithm obtains better results when using fewer features. Thus training on few features increases the number of documents that can not be then classified. As further development I will test on families of words and use as features only a representative of each family. By doing so the number of features will be significantly reduced and thus we can increase the number of files that can be classified further on. This method could increase the classification accuracy as it is used as a feature selection method. In order to achieve this we could use the [WordNet] database which contains the families of words for the English language.

In further experiments I plan to include more datasets, remove infrequent features and improve the existing methods of extracting features. Another problem for extracting features is the synonym problem and the polysemy problem that can reduce the size of the vector space. Maybe we could also find association rules between words and eliminate those words that occur together.

6 References

[Der00] Dertouzos, Michael – *What will be: How the New World of Information Will Change Our Lives*, Technical Publisher, Bucharest, 2000 (Translation in Romanian by I. Filip et al)

[Cro99] *Introduction to Data Mining and Knowledge Discovery*, Tow Crows Corporation, a short introduction to a new book ,1999 (Available at http://www.twocrows.com/booklet.htm)

[Mor05] Morariu Daniel – *Web information retrieval,* first PhD Technical Report, February 2005

[Cha00] Soumen Chakrabarti.- *Data mining for hypertext: A tutorial survey*, Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining, ACM Explorations 1(2), pages. 1-11,2000

[Mit97] Tom Mitchell – *Machine Learning*, McGraw Hill Publishers, 1997

[Mla99] D. Mladenic and M. Grobelnik – *Feature selection for unbalanced class distribution and naïve bayes*, In Proceedings of the 16th International Conference on Machine Learning ICML, p.258-267,1999

[Mla99] Mladenic Dunja, *Feature Subset Selection in Text Learning*, Proceedings of the 10th European Conference on Machine Leraning (ECML-98), pages 95-100, 1998

[Mla02] Mladenic Dunja, Brank J., Grobelnik M., Milic-Frayling N., *Feature Selection Using Support Vector Machines* The 27th Annual International ACM SIGIR Conference (SIGIR2004), pp 234-241, 2004

[Bha00] Bhatia Sanjiv, *Selection of Search Terms Based on User Profile*, ACM Explorations, pages. 224-233, 1998

[Reu2000] Misha Wolf and Charles Wicksteed- http://www.reuters.com/researchandstandards/corpus/ accessed in June 2005

[Jia01] Jiawei Han, Micheline Kamber - *Data Mining. Concepts and techniques*, Morgan Kaufmann Press, 2001

[Ian00] Ian H. Witten, Eibe Frank – *Data Mining, Practical Machine Learning Tools and Techniques with Java implementation*, Morgan Kaufmann Press, 2000

[IR] http://www.cs.utexas.edu/users/mooney/ir-course/ Information Retrieval java Application, accessed in September 2004

[Sparse] www.cs.utk.edu/~dongarra/etemplates/node372.html - representing implementation of sparse matrixes and vectors

[Yan97] Y. Yang, J.O. Pedersan – *A Comparative Study on Feature Selection in Text Categorization*, Proceedings of ICML, 14th International Conference of Machine Learning, pages 412-420, 1997

[Cov91] Thomas M. Cover, Thomas A. Joy – *Elements of Information Theory*, Jhon Wiley & Sons Interscience Publication, 1991

[Nel00] Nello Cristianini, John Shawe- Taylor – *An Introduction to Support Vector Machines and other kernel-based learning methods*, Cambridge University Press, 2000

[Dou04] Douglas Hardin, Ioannis Tsamardinos, Constantin Aliferis – *A Theoretical Characterization of Linear SVM-Based Feature Selection*, Proceedings of the 21st International Conference on Machine Learning, Banff, Canada,2004

[Jim04] Romng Jim, Huan Liu – *Robust Feature Induction for Support Vector Machine*, Proceedings of the 21st International Conference on Machine Learning, Banff, Canada, 2004

[For04] Forman George - *A Pitfall and Solution in Multi-Class Feature Selection for Text Classification,* Proceedings of the 21st International Conference on Machine Learning, Banff, Canada, 2004

[Weka] www.cs.waikato.ac.nz/~ml/weka/

[Sck02] Bernhard Schoslkopf, Alexander Smola – *Learning with Kernels, Support Vector Machines*, MIT Press, London, 2002

[Chi03] Chih-Wei Hsu, Chih-Chang Chang and Chih-Jen Lin – *A Practical Guide to Support Vector Classification*, Department of Computer Science and Information Engineering National Taiwan University, 2003(Available at http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide)

[Vap01] Vladimir Vapnik, Asa Ben-Hur, David Horn, Hava T. Sieglmann, *Support Vector Clustering*, Journal of Machine Learning Research 2, pages 125-137, 2001

[Jai00] Jaideep Srivastava, Robert Cooley,Mukund Deshpande, Pang Ning Tan, Web - Usage *Mining: Discovery and Applications of Usage Patterns from Web Data*, Technical Report, Department of Computer Science and Engineering University of Minnesota, 2000 (Available at http://maya.cs.depaul.edu/~classes/ect584/papers/srivastava.pdf)

[Pla99] J. Platt. *First training of support vector machines using sequential minimal optimization*. In B. Scholkopf, C.J.C. Burges, and A. J. Smola, editors, Advances in Kernel Methods – Support Vector Learning, pages 185-208, Cambridge, MA, 1999, MIT Press.

[Pla99_2] J. Platt – *Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods*, In Advances in Large Margin Classifiers, A. Smola, P. Bartlett, B. Scholkopf, D. Schuurmans, eds., MIT Press, Cambridge, pages 61-74, 1999

[Wah99] G. Wahba – *Support Vector Machine, reproducing kernel Hilbert space and the randomized GACV*. In B. Scholkopf, C. J. C. Burgers, and A. J. Smol;a, editors, Advances in Kernel Methods – Support Vector Learning, pages 69-88, Cambridge, MA, 1999, MIT Press.

[Kai03] Kai Yu, Anton Schwaighofer, Volker Tresp, Wei-Ying Ma, HongJing Zhang - *Collaborative Ensemble Learning: Combining Collaborative and Content Based Information Filtering via Hierarchical Bayes*, Proceeding of the 19th Conference, Uncertainty in Artificial Intelligence, pages 616-623, 2003

[Gab04] Gabrilovich E., Markovitch S. – *Text Categorization with Many Redundant Features Using Aggressive Feature Selection to Make SVM Competitive with C4.5*, Proceedings of the 21st International Conference on Machine Learning, Banff, Canada,2004

[LibSvm] http://www.csie.ntu.edu.tw/~cjlin/libsvm

[Jeb00] Jebara Tony and Jaakkola T. – *Feature selection and dualities in maximum entropy discrimination*, In Uncertainty in Arti cal Intelligence 16, 2000

[Jeb04] Jebara Tony – *Multi Task Feature and Kernel Selection for SVMs*, Proceedings of the 21st International Conference on Machine Learning, Banff, Canada, 2004

[Jur03] Jurafsky D, Pradhan S, Ward W., Hacioglu K., Martin J. – Shallow Semantic Parsing using Support Vector Machines, 2003

[Cha03] Soumen Chakrabarti – *Mining the Web- Discovering Knowledge from hypertext data*, Morgan Kaufmann Press, 2003

[SvmLight] Thorsten Joachims – *Making large-scale support vector machine learning practical*, In B. Scholkopf, C. J. C. Burges, A.J. Smola (Eds): *Advances in kernel methods: Support vector learning*, MIT Press, 1999, pp. 169-184

[Xu02] Zhao Xu, Kai Yu, Volker Tresp, Xiaowei Xu, Jizhi Wang – *Representative Sampling for Text Classification using Support Vector Machines*, 2002

[WordNet] http:\\www.cogsci.princeton.edu\wn2.0 – online lexical system