A Better Correlation of the SVM Kernel's Parameters

Daniel I. Morariu, Lucian N. Vintan

Abstract — **Support Vector Machine (SVM) is a powerful method of classification, based on kernels, working with large data sets. In almost all cases there are used two distinct parameters that can be modified for obtaining best results. One of these parameters is easy to infer but the second is usually used as being the number of features that are taken into consideration. This seems to be not a good idea for text classification because it was shown that for a great number of features the classification accuracy is quite poor. In this paper we propose a method to correlate those parameters in order to obtain better results and vary only one parameter. We also show that using this method leads in almost all cases to better results. We introduce two formulas to correlate the parameters for polynomial and Gaussian kernels.**

Keywords — **Learning with Kernels, Support Vector Machine, and Text Classification**

I. INTRODUCTION

Documents are typically represented as 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's vector. The native feature space consists of the unique terms that occur into the documents, which can be tens or hundreds of thousands of terms for even a moderate-sized text collection, being a major problem of text categorization.

As a method for text classification we use Support Vector Machine technique, which was proved as being efficient for nonlinear separable input data. This is a relatively recent learning method based on kernels [1],[2]. We use this method both in features selection step and in the classification step. Also we studied the influence of the input data representation on kernel parameters correlation.

In this paper we present a comparative study of different parameters for two types of kernels, polynomial and Gaussian, and different methods to correlate kernel's parameters. In almost all articles where the SVM method is used, researchers used two parameters to infer the kernel but usually they modified only one of them. For the second parameter it is not explicitly specified the modification rule. Sometimes this parameter is chosen as the number of features. For some kinds of applications, when the number of features is not so large, this can be a good idea; but for text classification, when the number of features can be a great, this can lead to powerless results.

The general process of classifying text data can be considered as having four steps. In the first step is done feature extraction from the text file (text mining). In the second step the features are selected. The third step is the learning step (training). The last is the evaluation step, where the classification process is evaluated.

In the first step, we have used text mining like an application of data mining techniques to extract the feature vectors that characterizes a document. Starting with a set of d documents and t terms (words belonging to the documents), we can model each document as a vector v in the t dimensional space $\mathbb{R}^t$ (a feature vector that characterizes the document into the set of documents).

In the second step we used SVM technique as a method of feature selection in order to reduce the features space dimension and to select the best features. In [3] SVM feature selection was proved to be the best one. For the input data we have used three types of representations: *Binary representation*, *Nominal representation,* and *Cornell SMART*.

In the next step (classification), we have also used Support Vector Machine. A great advantage of this technique is that it can use large input sets. We implemented this classification method for two types of kernels: *polynomial kernel* and *Gaussian kernel (Radial Basis Function - RBF)*. We tried to find a simplified form of the kernels, without reducing the performance, actually increasing it, using only one, more intuitive parameter.

For multi-class classification we chose the well-known method "one class versus the rest" [4]. Thus, considering M classes, we repeated two class classification for each topic (the category where the document is classified) obtaining M decision functions.

Section II contains prerequisites for the work that we present in this paper. In section III we present the frame and methodology used for our experiments. Section IV presents the main results of our experiments. Section V debates and concludes on the most important results obtained and proposes some further work.

II. SUPPORT VECTOR MACHINE

Support Vector Machine (SVM) is a classification technique based on the statistical learning theory [4],[5] that was applied with great success in many challenging non-linear classification problems and was used on large sets of data with big samples.

The purpose of the algorithm is to find a hyperplane that optimally splits the training set (a practical idea can be found in [6]). This technique is based on two class classification. There are some methods used for classification in more that two classes. Looking at the two dimensional problem we actually want to find a line that

D. Morariu is with the Faculty of Engineering, "Lucian Blaga" University of Sibiu, Romania (phone: 40/740/092202; e-mail: daniel.morariu@ulbsibiu.ro).

L. Vintan is with the Faculty of Engineering, "Lucian Blaga" University of Sibiu, Romania (phone: 40/745/927450; e-mail lucian.vintan@ublsibiu.ro).

"best" separates points in the positive class from the points in the negative class. The hyperplane is characterized by a decision function like $f(x) = \operatorname{sgn}(\langle \mathbf{w}, \mathbf{\Phi}(x)\rangle + b)$, 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, "*Φ(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. If **w** has unit length, then <**w,** *Φ(x)*> is the length of *Φ(x)* along the direction of **w**. Generally **w** will be scaled by ||**w**||. The training part of the algorithm needs to find the normal vector "**w**" that leads to the largest "*b*" of the hyperplane.

For training data which is non 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. This yields a non-linear decision boundary into 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 [4].

Everything was formulated in a dot product space. On the practical level, changes have to be made to perform the algorithm in a higher-dimensional feature space. Thus the new patterns $\Phi(x_i)$ aren't necessary to coincide with the input patterns. These patterns can equally well be the result of mapping the original input patterns x_i into a higher dimensional feature space using function Φ. Maximizing the target function and evaluating the decision function involve the computation of dot products $\langle \phi(x), \phi(x)\rangle$ into a higher 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 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 it is obtained the decision function having the following form:

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

where α_i represent the Lagrange multipliers and the samples x_i for which $\alpha_i > 0$ are called Support Vectors.

III. Experimental Framework

A. The used dataset

Our experiments are performed on the Reuters-2000 collection [7], which have 984Mb of newspapers articles in a compressed format. Collection includes a total of 806,791 documents, with news stories published by Reuters Press covering the period from 20.07.1996 through 19.07.1997. The articles have 9822391 paragraphs and contain 11522874 sentences and more than two hundred million words. Documents are pre-classified according to 3 categories: by the *Region* (366 regions) the article refers to, by *Industry Codes* (870 industry codes) and by *Topics* proposed by Reuters (126 topics, 23 of them contain no articles). Due to the huge dimension of the database we will present here results obtained using a subset of data. From all documents we selected the documents for which the industry code value is equal to "System software". We obtained 7083 files that are represented using 19038 features and 68 topics. We represent documents as vectors of words, applying a stop-word filter (from a standard set of 510 stop-words) and extracting the steam of the word. From these 68 topics we have eliminated those topics that are poorly or excessively represented. Thus we eliminated those topics that contain less than 1% documents from all 7083 documents in the entire set. We also eliminated topics that contain more than 99% samples from the entire set, as being excessively represented. The elimination was necessary because with these topics we have the risk to use only a single decision function for classifying all documents ignoring the rest of the decision functions. After doing so we obtained 24 different topics and 7053 documents that were splat randomly in training set (4702 samples) and evaluation set (2531 samples).

B. Used kernel types

The idea of the kernel is to compute the norm of the difference between two vectors in a higher dimensional space without representing those vectors in the new space. In practice we can see that by adding a scalar constant to the kernel we can get better classifying results. In this paper we tested a new idea to correlate this scalar with the dimension of the space where the data will be represented. We consider that those two parameters (the degree and the scalar) need to be correlated.

For the polynomial kernel we change only the polynomial degree and for the Gaussian kernel we change only the constant C according to the new following formulas (*x* and *x'* being the input vectors):

Polynomial $$k(x,x') = \left(2 \cdot d + \langle x \cdot x' \rangle\right)^d \qquad (2)$$

d being the only parameter that need to be modified and

Gaussian $$k(x,x') = \exp\left(-\frac{\|x - x'\|^2}{n \cdot C} \right) \qquad (3)$$

where *C* being the only parameter that need to be modified and *n* being the number of elements greater than 0 from the input vectors that is automatically computed.

C. Polynomial kernel parameter's correlation

Usually when learning with a polynomial kernel researchers use a kernel that looks like $(\langle \mathbf{x} \cdot \mathbf{x}' \rangle + b)^d$ where *d* and *b* are independent parameters. "*d*" is the degree of the kernel and it is used as a parameter that helps mapping the input data into a higher dimensional features space. This is why this parameter is intuitive. The second parameter "*b*", is not so easy to infer. In all studied articles, the researchers used it, but they don't present a method for selecting it. We notice that if this parameter was eliminated (i.e., chosen zero) the quality of results can be poor. It is logically that we need to correlate *d* and *b* parameters because the offset *b* needs to be modified as the dimension of the space modifies. Due to this, based on running laborious classification simulations presented in this paper, we suggest using "*b=2*d*" in our application.

D. Gaussian kernel parameter's correlation

Also for the Gaussian kernel we modified the standard kernel used by research community. Usually the kernel looks like $k(x,x') = \exp(-\|x - x'\| / C)$, where the parameter C is a number that represents the dimension of the training set (and usually this is a very big number for text classification). We introduce the parameter *n* that is multiplied by parameter *C* and is a value that represents the number of distinct features that occur into the current two input vectors (*x* and *x'*), having weights greater than 0. We keep the constant *C* that becomes a small number (usually obtain best results between 1 and 2). As far as we know, we are the first authors proposing a correlation between these two parameters for both polynomial and Gaussian kernels.

E. Representing the input data

Because there are many ways to define the feature-weight, we represent the input data in different formats, and we try to analyze their influence on classification accuracy. We take in consideration three formats for representing data [8].

Binary representation – in the input vector we 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.

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

$$TF(d,t) = \frac{n(d,t)}{\max_{\tau} n(d,\tau)}, \qquad (4)$$

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.

Cornell SMART representation – in the input vector we compute the value of the weight using the formula:

$$TF(d,t) = \begin{cases} 0 & if\, n(d,t) = 0 \\ 1 + \log(1 + \log(n(d,t))) & otherwise \end{cases} \qquad (5)$$

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 into the document and, greater then 1 if it exists. The scaled value creates a gap between the values for no word and the value of one appearance.

IV. Simulation Results

For each test we train the algorithm using the training files. After that we test the decision functions obtained using the test files. We classify the document into the class for which is obtained the greater absolute value. The result (class) is compared with the results (classes) that were proposed by Reuters. We obtain thus the accuracy of classification for all 24 classes. We'll present all tests using only 1309 features obtained from all 19038 features, using a feature selection method based also to Support Vector Machine [3].

A. Results for polynomial kernel

In order to improve the classification accuracy using polynomial kernel our idea was to correlate the bias of the kernel with the degree of the kernel (Section III.C). In this idea we develop tests for four degrees of the kernel, considering for each of them 16 distinct values of the bias and, respectively, for each input data representation. Thus for each degree of the kernel we vary the value of the bias from 0 to the number of the features.

In Figure 1 we present results obtained with polynomial kernel and Nominal data representation by varying the degree of the kernel and the bias. In "Our choice" entry we put only the values that were obtained using our formula that correlates the polynomial kernel's parameters. As it can be observed, using our correlation idea (formula 2) assures that in almost all cases there are obtained the best results. In this case, only for degree 4 the best value was obtained for bias equal with 2 and we obtain a value with 0.21% smaller than the best results (84.56% in comparison with the best obtained 84.77%).

Effective values of the accuracy obtained using Cornell Smart data representation, for each kernel degree and for each bias, are presented in Table 1. For each degree there are multiple bias values involving best results and our proposed formula assures to hit these values in almost all cases. Also an interesting observation is that for kernel degree = 1, we usually obtained the same classification accuracy for all bias values, with only 0.51% smaller that the best value. As it can be observed from Figure 1, with no bias we obtain the worst results. The same tests were developed also for Binary data representation and we have obtained usually the same results.

B. Results for Gaussian kernel

For the Gaussian kernel we modified the usually used constant *C* that represent the number of features, with a product between a small number (noted also by *C* in our formula 3) and a number that is automatically computed (Section III.D). We made tests with four *C* distinct values. For each of these values, we vary *n* from 1 to 1309 (total number of the features used). Because our proposed value for *n* is automatically computed, this number can not be specified by the command line, so that for each value of constant *C* we specified a value called "auto" (in Figure 2) that means the value automatically computed using our formula.

We made tests only for Binary and Cornell Smart representations of the input data. Into Gaussian kernel we fill in a parameter that represents the number of elements greater then zero (parameter "*n*" from equation 3). Nominal representation (equation 4) represents all weight values between 0 and 1. When parameter "*n*" is used, all the weights become very close to zero involving very poor

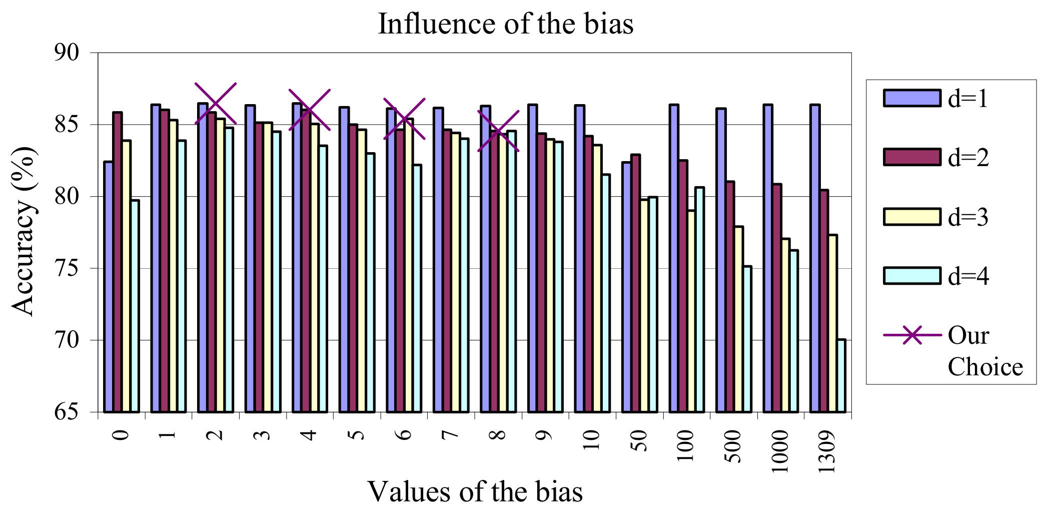

Figure 1 – Influence of bias for Nominal representation of the data

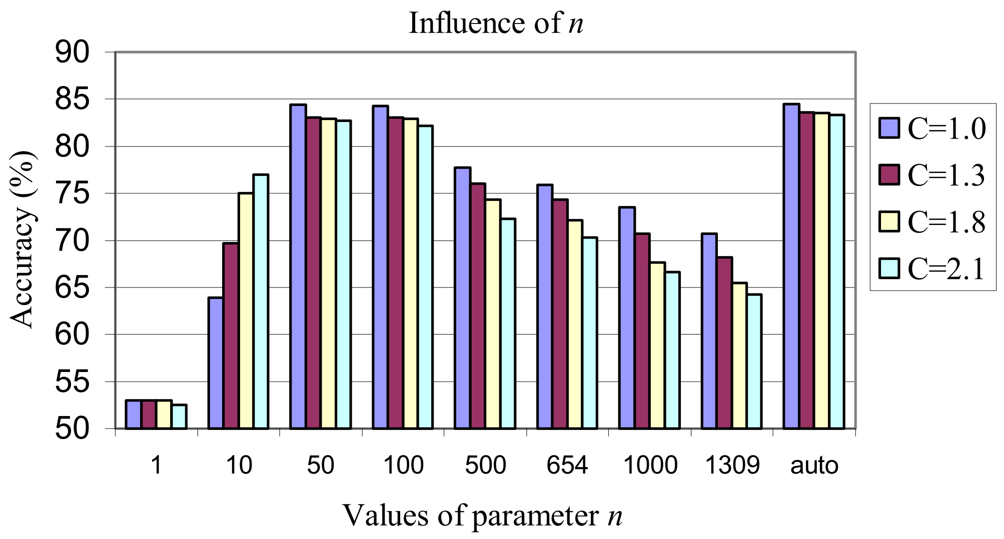

Figure 2 – Influence of value for "*n*" for Gaussian kernel and Binary data representation

classification accuracies (for example, due to its almost zero weight, a certain word really belonging to the document, might be considered as not belonging to that document). So we don't present here the results obtained using Nominal representation and Gaussian kernel.

In Figure 2 we present results obtained for Binary data representation. When we use our idea to compute *"n"*, we obtained the best results. Also better results were obtained when the value of *n* is between 50 and 100. This occurs because usually each document uses a small number of features (on average between 50 and 100) in comparison with features from entire set of documents. When *n* is equal with the total number of features (usually used into the literature) the accuracy decrees, in average for all testes, with more than 10% in comparison with using an automatically computed value for *n*. It can also be observed that when the value of parameter *n* increases the accuracy decrees substantially. The same tests were also made using Cornell Smart data representation, obtaining the same tendency and usually accuracy with 1% better than in Binary case.

In contrast with polynomial kernel, in Gaussian kernel case we obtained best results only with our proposed formula to compute the parameter "*n*".

C. *Kernel's influence*

In this section we present the influence of correlating kernel's parameters on classification accuracy using other implementation of SVM. In order to do this we make a short comparison between the results that we obtained with usually used implementation of SVM, called LibSvm [9], and our implemented application called UseSvm [10]. LibSvm uses "one versus the one" method for multiclass classification. Our developed UseSvm program uses "one versus the rest" method, as we already mentioned. Reuter's database, used in our tests, contains strongly overlapped data and in this case the first method usually obtains powerless results.

We have used also sets with 1309 features, obtained using SVM feature selection method. In order to fairly

Table 1: Influence of bias for CORNELL SMART data representation.

Bias	D=1	D=2	D=3	Choice
0	81.84	86.69	82.35	
1	81.84	86.64	83.37	
2	**82.22**	**86.81**	84.01	**82.22**
3	**82.22**	86.56	84.77	
4	**82.22**	**86.81**	65.12	**86.81**
5	82.01	86.47	85.54	
6	81.71	86.60	86.39	**86.39**
7	81.71	86.43	86.18	
8	82.09	86.43	**86.47**	
9	81.84	86.18	**86.47**	
10	81.80	85.96	86.26	
50	81.92	84.73	84.90	
100	**82.22**	83.71	82.82	
500	82.05	81.88	8.34	
1000	82.09	80.94	53.51	
1309	82.09	80.77	50.40	

compare LibSvm with our UseSvm, we eliminated, when possible, Reuters overlapped data (for working only with non-overlapped classes, as far as is possible; formally, $\forall i, j = \overline{1,13}, \mathbf{C}_i \cap \mathbf{C}_j = \varnothing \textit{ for each } i \neq j$). We choose for each sample only first class that was proposed by Reuters. We also eliminated classes that are poorly or excessively represented. We obtained only 13 classes randomly split in two sets and used for training and testing for both LibSvm and UseSvm. Results obtained by LibSvm are poor in comparison with the results of our application, because, despite our efforts, the data are however slightly overlapped. In Figure 3 and Figure 4 we present results obtained for the polynomial kernel and the Gaussian kernel with nominal data representation for both applications. We are using equivalent parameters for both applications. As LibSvm has more parameters than UseSvm, we have left on default value for the parameters that appear only in LibSvm.

As we already specified, for polynomial kernel our suggestion was to make "*b=2*d*" (see Section III.C). We present results using LibSvm with *b=0* (default value) respectively with *b=2*d* (specified explicitly by command line) comparing with our UseSvm program.

As it can be observed from Figure 3, our UseSvm program obtains far better results than the well-known LibSvm (with an average gain of 18.82% better). By comparing LibSvm with the default bias with LibSvm with modified bias (according to our formula 2), we noticed that the modified bias leads to better results (with an average gain of 24.26% better). The average gain is computed as average obtained by LibSvm with the default bias divided by the average obtained by LibSvm with modified bias. For degree 1 were obtained similar results because values of default bias and value computed using our formula are quite equal.

For the Gaussian kernel simulations, presented in Figure 4, our suggestion was to multiply the constant C with a parameter *n* (like we already explained in Section III.D). It is difficult to give this parameter from the LibSvm's command line because *n* is computed dynamically and LibSvm have only one parameter that can be modified, called *gamma*. More precisely, LibSvm uses $gamma = 1/n$ only when gamma is default (*n* means the average of number of attributes in the input data). For LibSvm we have used *gamma* as $1/C$. For *LibSvm+"gamma"* we considered "*gamma*" to be equal to $2/nC$, where *n* is the number of features. The single case of equivalence between these two programs (LibSvm and UseSvm) is obtained for the default value of *gamma* in LibSvm and respectively for *C*=1 in UseSvm. This case is presented separately as "def" in Figure 4.

As it can be observed, using our idea to modify the Gaussian kernel the results obtained using LibSvm are better in comparison with results obtained using LibSvm with standard kernel (with an average gain of 28.44%). Our UseSvm program obtains far better results than the well-known LibSvm (with an average gain of 25.57% better). For the default parameter of LibSvm our application also has obtained better results (76.88% in comparison with 69.97% for LibSvm).

V. Conclusions and further work

In this paper, we have proposed a new method to better correlate kernel's parameters. The method correlates the degree of the Polynomial kernel with the bias, respectively correlates the constant from the Gaussian kernel with a value that represents the number of distinct features that occurs into the currently used vectors and having weights greater than 0. In the polynomial kernel case there are more values for which we obtain the best results but our proposed formula assures to hit in almost all cases the best results without having to make more tests to find the good parameter for the bias. For Gaussian kernel our proposed formulas assures to obtain the best results. Also we showed that for text classification problems, using bias or parameter *C* equal to the number of features, as it is usually used in the literature, is not a good idea. Using our proposed formulas there is obtained in average with 3% better results for polynomial kernel and with 15% better results for Gaussian kernel.

Also we tested our idea using other SVM implementation, usually used into the literature and called LibSvm, and we obtain the same tendency. We obtained an average accuracy classification gain of 24.26% for polynomial kernel, respectively 28.44% for Gaussian kernel. As far as we know, we are the first authors proposing a correlation between these two parameters for both polynomial and Gaussian kernels.

Work is ongoing to classify larger text data sets (all Reuters database). In this idea we want to develop a pre-classification of all documents, obtaining fewer samples (using simple algorithms like Linear Vector Quantization or Self Organizing Maps). After that we'll use the obtained samples as entry vectors for the already developed features selection and classification methods.

Because almost all available data from a real world are

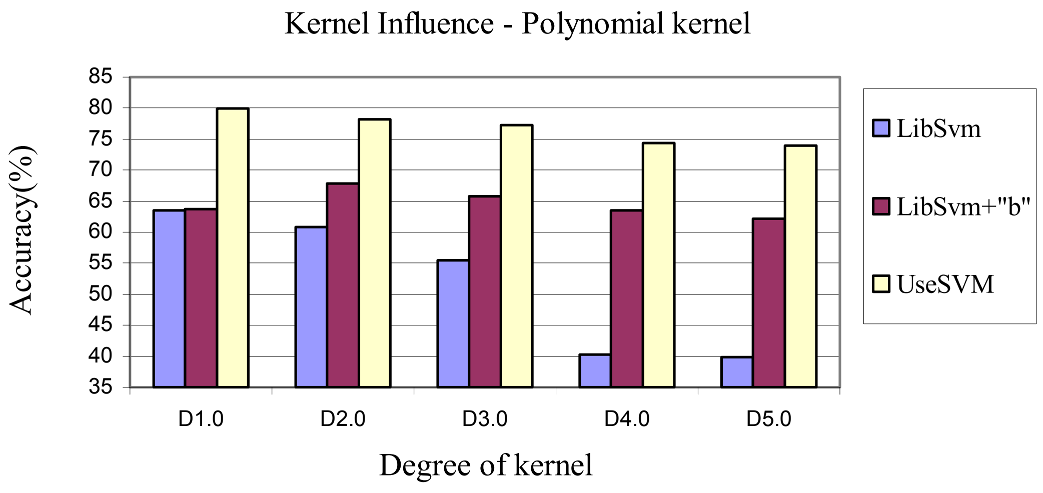

Figure 3 Influence of correlation between parameters from polynomial kernel

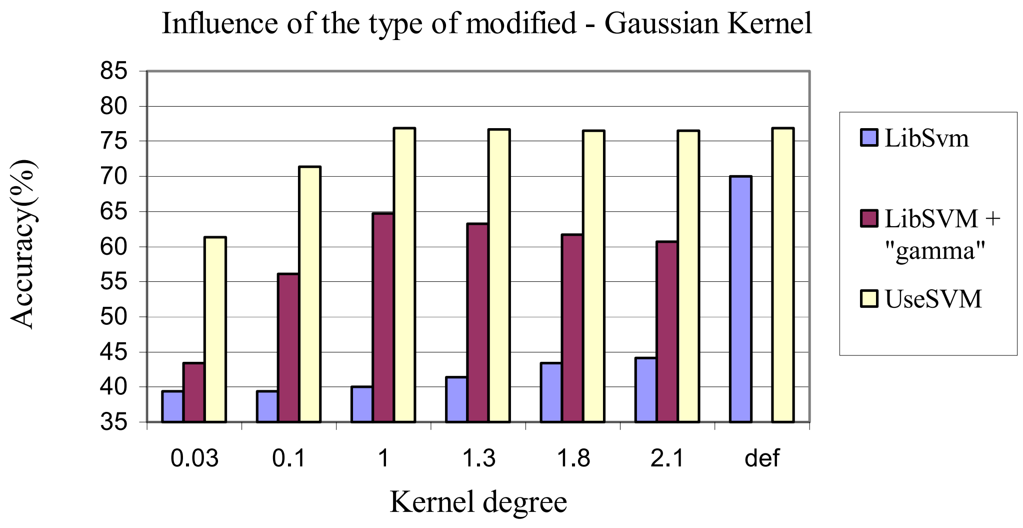

Figure 4 Influence of modified about Gaussian Kernel

in fact unlabeled data, we will try to combine classifying method with a clustering method, also based on SVM, in order to use labeled and unlabeled data into a hybrid classification algorithm. An interesting natural extension of our algorithm might be an adaptation for a Web mining application, in order to extract and categorized online news.

Acknowledgment

We would like to thank to SIEMENS AG, CT IC MUNCHEN, Germany, especially to Mr. Vice-President Dr. H. C. Hartmut RAFFLER, for his generous and various support, both professional and material, that he has provided in developing this work.

References

[1] Vapnik V., *The nature of Statistical learning Theory*. Springer, New York, 1995

[2] Platt J., *Fast 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

[3] Morariu D., Vintan L., Tresp V., *A comparative study of features selection methods for SVM Classification*, submitted to 17th European Conference on Machine Learning (ECML2006), Berlin, September 2006

[4] Schoslkopf B., Smola A., *Learning with Kernels, Support Vector Machines*, MIT Press, London, 2002

[5] Nello C., Swawe-Taylor J., *An introduction to Support Vector Machines*, Cambridge University Press, 2000

[6] 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)

[7] Reuters Corpus, *Volume 1, English Language, 1996-08-20 to 1997-08-19*. Available through http://about.reuters.com/ researchandstandards/corpus/. Relased in November 2000.

[8] Chakrabarti S., *Mining the Web- Discovering Knowledge from hypertext data*, Morgan Kaufmann Press, 2003

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

[10] Morariu D., *Classification and Clustering using Support Vector Machine*, 2nd Phd Report, University „Lucian Blaga“ of Sibiu, 2005, (Availabe at http://webspace.ulbsibiu.ro/daniel.morariu /html/Docs/Report2.pdf)