VERSITA

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **12**, No 4

Sofia • 2012

Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2012-0032

Pattern Synthesis Using Multiple Kernel Learning for Efficient SVM Classification

Hari Seetha[1], *R. Saravanan*[2], *M. Narasimha Murty*[3]

[1]*School of Computing Science and Engineering, VIT University, Vellore-632014*
[2]*School of Information Technology and Engineering, VIT University, Vellore-632 014*
[3]*Department of Computer Science and Automation, IISc, Bangalore-12*
Email: hariseetha@gmail.com

Abstract: *Support Vector Machines* (*SVMs*) *have gained prominence because of their high generalization ability for a wide range of applications. However, the size of the training data that it requires to achieve a commendable performance becomes extremely large with increasing dimensionality using RBF and polynomial kernels. Synthesizing new training patterns curbs this effect. In this paper, we propose a novel multiple kernel learning approach to generate a synthetic training set which is larger than the original training set. This method is evaluated on seven of the benchmark datasets and experimental studies showed that SVM classifier trained with synthetic patterns has demonstrated superior performance over the traditional SVM classifier.*

Keywords: *SVM classifier; curse of dimensionality, synthetic patterns; multiple kernel learning.*

1. Introduction

In most of the real world data sets, the dimensionality of the data exceeds the number of training patterns. It is generally recommended that the ratio of training set size to the dimensionality be large [1]. Earlier studies reported that the number of training samples per class should be at least 5-10 times the dimensionality of the

data ([1, 2]). Duda et al. [3] mentioned that the demand for a large number of samples increases exponentially with the dimensionality of feature space. This results in the *curse of dimensionality*.

SVM classifier lacks perfectness in case of real life data sets where the size of the data is generally lower than that of dimensionality, though the available literature confirms its prominent performance using only linear SVMs. Hastie et al. [4] discussed that whether using linear or nonlinear kernels, SVMs are not immune to the curse of dimensionality. The reasons could be insufficient training data and noise in the training data. In order to demonstrate that kernel based pattern recognition is not entirely robust against high dimensional input spaces; Silverman [5] reported the difficulty of kernel estimation in high dimensions as shown in Table 1.

Table 1. Dimensionality vs. sample size

Dimensionality	Required sample size
1	4
2	19
5	786
7	10 700
10	842 000

Typically, SVM performs classification using linear, polynomial and RBF (Gaussian) kernels. All of them use inner products. The most popular kernel used for classification is Gaussian kernel $k(x_1, x_2) = e^{-(\|x_1 - x_2\|)^2/2\sigma^2}$. The square of the Euclidean distance $(\|x_1 - x_2\|)^2$ affects the Gaussian kernel. Beyer et al. [6] illustrated that the maximally distant point and minimally distant point converge which is a problem with Euclidean distance in high dimensionality. In [7] is shown that the linear kernel is a special case of Gaussian kernel. Further, the relationship between Gaussian and linear kernel can be given as follows:

$$e^{-(\|x_1 - x_2\|)^2/2\sigma^2} = 1 - \frac{\left(\|x_1 - x_2\|\right)^2}{2\sigma^2} \text{ (neglecting higher order terms) =}$$

$$= 1 - \frac{1}{2\sigma^2}\left(\left(x_1 - x_2\right)^t \left(x_1 - x_2\right)\right) =$$

$$= 1 - \frac{1}{2\sigma^2}\left(x_1^2 + x_2^2 - x_2^t x_1 - x_1^t x_2\right) =$$

$$= 1 - \frac{1}{2\sigma^2}\left(2 - 2\left(x_1 . x_2\right)\right)$$

(∵ $\|x_1^2\| = \|x_2^2\| = 1$, as the datasets are generally normalized to have unit length).

Fillipone et al. [8] explained that the linear kernel leads to the computation of the Euclidean norm in the input space. Evangelista et al. [9] showed that increasing dimensionality degrades the performance of the linear, Gaussian and polynomial kernels and also demonstrated that each variable (feature) added affects the overall behaviour of the kernel. Hastie et al. [4] discussed that if the dimensionality is large and the class separation occurred only in the linear subspace,

spanned by the first two features then the polynomial kernel would suffer from having many dimensions to search over.

Synthetic pattern generation is a novel approach to overcome the curse of dimensionality. Very few studies were reported in literature regarding artificial pattern generation. Viswanath et al. [10, 11] proposed a pattern synthesis approach for efficient nearest neighbor classification. Agrawal et al. [12] applied prototyping as an intermediate step in the synthetic pattern generation technique to reduce classification time of K nearest neighbour classifier.

It is evident from the literature that almost no effort has been made to generate synthetic patterns for improving the performance of SVM classifier; although it is widely believed that achieving a given classification accuracy needs a large training set when the dimensionality of the data is high. But such a study would be helpful in the classification of real world data because getting real world large datasets is difficult. Hence, the main objective of this investigation is to simulate smoothed training patterns using Multiple Kernel Learning (MKL) approach, such that the size of the new training set is larger than that of the original training set, and thereby it improves the classification performance of SVM on high dimensional data. In MKL approach several kernels are synthesized into a single kernel while classical kernel-based algorithms are based on a single kernel. Although MKL has recently been a topic of interest ([13, 14]), it was not earlier applied (as far as authors knowledge goes) to generate synthetic patterns.

This paper is organized as follows: Section 2 describes the proposed method with an example, Section 3 explains the block diagram of the proposed system used to simulate new training patterns, Section 4 discusses the feature separation and Section 5 explains the bootstrapping technique. Experimental studies are shown in Section 6 with conclusions in Section 7.

2. Notations and description of the method proposed

Let us suppose that the data under consideration has *n* features $F = (f_1, f_2, ..., f_n)$. Each of the samples in the data belongs to one of the classes given by $C = (C_1, C_2, ..., C_i)$. The data is divided into training and testing sets, such that the training set is independent on the testing set. The *m*-th training sample of class C_i is represented by $X_{mi} = (x_{mi_1}, x_{mi_2}, ..., x_{mi_n})$ where x_{mi_1} is the value of the training sample X_{mi} for feature f_1, x_{mi_2} is the value of the training sample X_{mi} for feature f_2, and x_{mi_n} is the value of the training sample X_{mi} for feature f_n. If Ω_1 is the set of the training samples of class C_1, Ω_2 is the set of the training samples of class C_2, and Ω_i is the set of the training samples of class C_i, then $\Omega = \Omega_1 \cup \Omega_2 \cup ... \cup \Omega_i$ is the set of all training samples. For each class of data, the set of n features F is separated into p blocks $B = \{B_1, B_2, ..., B_p\} \ni B_q \subseteq F$

for $q = 1, 2, ..., p,$ and $\bigcup_{q=1}^{p} B_q = F$, as well as $B_q \cap B_r = \phi$ for $q \neq r \ \forall q, \forall r.$ Thus each training pattern of each class is partitioned into p sub-patterns. Let X_{mi}^{p} represents the sub pattern of *m*-th training sample X_{mi}^{p} of class C_i that belongs to block B_p. Let $X_{1i}^{p}, X_{2i}^{p}, ..., X_{ri}^{p}$ be its r nearest neighbours in the block B_p of class C_i. Then $X_{mi}^{bp} = \frac{1}{r}\sum_{h=1}^{r} X_{hi}^{p}$ is the artificial bootstrap pattern generated for X_{mi}^{bp} [1]. This process is repeated for each training sub pattern of the block B_p without selecting it more than once. Applying one class of a SVM classifier on bootstrapped samples of B_p of class C_i, the support vectors SV_i^p of block B_p of class C_i are determined. This procedure is repeated for every block of each class. Thus a single kernel function, i.e. either linear, RBF or polynomial kernel is applied commonly on each classwise feature partition. Firstly, a linear kernel is applied commonly on all classwise blocks, then RBF and later the polynomial separately. The Cartesian product $\Omega_i^{'} = \left\{\mathrm{SV}_i^1 \times \mathrm{SV}_i^2 \times ... \times \mathrm{SV}_i^p\right\}$ is the new synthetic training set generated for class C_i. This procedure is repeated for each class generating $\Omega_1', \Omega_2', ..., \Omega_i'$ new training patterns for each class. In this way a novel approach of multiple kernel learning is used for generating synthetic patterns.

Example. To illustrate the proposed method, let us accept that the dataset has six training patterns, with five features, represented by the set of features $F = \left(f_1, f_2, f_3, f_4, f_5\right)$ and each of the training pattern belongs to any one of the classes having class labels C_1 and C_2. Let the set of training samples of class C_1 be $\Omega_1 = \left\{\left(a_1, a_2, a_3, a_4, a_5\right), \left(d_1, d_2, d_3, d_4, d_5\right), \left(e_1, e_2, e_3, e_4, e_5\right)\right\}$ and let the set of training samples of class C_2 be

$$\Omega_2 = \left\{\left(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\right), \left(\beta_1, \beta_2, \beta_3, \beta_4, \beta_5\right), \left(\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5\right)\right\}.$$

Then the original training set is

$$\Omega = \Omega_1 \cup \Omega_2 = \left\{\begin{matrix}\left(a_1, a_2, a_3, a_4, a_5\right), \left(d_1, d_2, d_3, d_4, d_5\right), \left(e_1, e_2, e_3, e_4, e_5\right), \\ \left(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\right), \left(\beta_1, \beta_2, \beta_3, \beta_4, \beta_5\right), \left(\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5\right)\end{matrix}\right\}.$$

Let $B = \left\{B_1, B_2\right\}$ be the partition of the features *F*, such that

$$B_1 = \{f_1, f_3, f_4\} \text{ and } B_2 = \{f_2, f_5\}.$$

Then, $\Omega_1^1 = \left\{\left(a_1, a_3, a_4\right), \left(d_1, d_3, d_4\right), \left(e_1, e_3, e_4\right)\right\}$ represents the sub-patterns of block B_1 of class C_1. $\Omega_1^2 = \left\{\left(a_2, a_5\right), \left(d_2, d_5\right), \left(e_2, e_5\right)\right\}$ represents the sub-patterns of block B_2 of class C_1, and

$$\Omega_2^1 = \left\{(\alpha_1, \alpha_3, \alpha_4), (\beta_1, \beta_3, \beta_4), (\gamma_1, \gamma_3, \gamma_4)\right\},$$
$$\Omega_2^2 = \left\{(\alpha_2, \alpha_5), (\beta_2, \beta_5), (\gamma_2, \gamma_5)\right\}$$

represent the sub-patterns of block B_1 and B_2 of class C_2 respectively. Let $\Omega_1^{b1} = \left\{\left(a_1^b, a_3^b, a_4^b\right), \left(d_1^b, d_3^b, d_4^b\right), \left(e_1^b, e_3^b, e_4^b\right)\right\}$ represent the bootstrapped sub-patterns of block B_1 of class C_1. Let

$$\Omega_1^{b2} = \left\{\left(a_2^b, a_5^b\right), \left(d_2^b, d_5^b\right), \left(e_2^b, e_5^b\right)\right\}$$

represent the bootstrap sub-patterns of B_2 of class C_1. Similarly,

$$\Omega_2^{b1} = \left\{\left(\alpha_1^b, \alpha_3^b, \alpha_4^b\right), \left(\beta_1^b, \beta_3^b, \beta_4^b\right), \left(\gamma_1^b, \gamma_3^b, \gamma_4^b\right)\right\},$$
$$\Omega_2^{b2} = \left\{\left(\alpha_2^b, \alpha_5^b\right), \left(\beta_2^b, \beta_5^b\right), \left(\gamma_2^b, \gamma_5^b\right)\right\}$$

represent the bootstrap sub-patterns of block B_1 and B_2 of class C_2 respectively. Let $\mathrm{SV}_1^1 = \left\{\left(a_1^b, a_3^b, a_4^b\right), \left(d_1^b, d_3^b, d_4^b\right)\right\}$ be the support vectors obtained by applying one class of a SVM classifier to block B_1 of class C_1 using any one of the kernels, i.e. linear, RBF or polynomial. Similarly, $\mathrm{SV}_1^2 = \left\{\left(a_2^b, a_5^b\right), \left(e_2^b, e_5^b\right)\right\}$ be the support vectors obtained from block B_2 of class C_1. In the same way, $\mathrm{SV}_2^1 = \left\{\left(\alpha_1^b, \alpha_3^b, \alpha_4^b\right), \left(\beta_1^b, \beta_3^b, \beta_4^b\right)\right\}$ and $\mathrm{SV}_2^2 = \left\{\left(\alpha_2^b, \alpha_5^b\right), \left(\beta_2^b, \beta_5^b\right)\right\}$ be the support vectors obtained from block B_1 and B_2 of class C_2 respectively. Then the synthetic training set for class C_1 is generated by performing the Cartesian product $\Omega_1' = \mathrm{SV}_1^1 \times \mathrm{SV}_1^2$ and rearranging the features in the original order of features. The new simulated set of the training patterns for class C_1 is

$$\Omega_1' = \left\{\left(a_1^b,\ a_2^b,\ a_3^b,\ a_4^b,\ a_5^b\right),\ \left(a_1^b, e_2^b, a_3^b, a_4^b, e_5^b\right),\right.$$
$$\left.\left(d_1^b,\ a_2^b,\ d_3^b,\ d_4^b,\ a_5^b\right),\ \left(d_1^b, e_2^b, d_3^b, d_4^b, e_5^b\right)\right\}.$$

Similarly, the new training set generated for class C_2 is $\Omega_2' = \mathrm{SV}_2^1 \times \mathrm{SV}_2^2$, i.e.,

$$\Omega_2' = \left\{\left(\alpha_1^b, \alpha_2^b, \alpha_3^b, \alpha_4^b, \alpha_5^b\right),\ \left(\alpha_1^b, \beta_2^b, \alpha_3^b, \alpha_4^b, \beta_5^b\right),\right.$$
$$\left.\left(\beta_1^b, \alpha_2^b, \beta_3^b, \beta_4^b, \alpha_5^b\right),\ \left(\beta_1^b, \beta_2^b, \beta_3^b, \beta_4^b, \beta_5^b\right)\right\}.$$

The synthetic training set generated is given by

$$\Omega' = \Omega'_1 \cup \Omega'_2 = \Big\{\Big(a_1^b, a_2^b, a_3^b, a_4^b, a_5^b\Big), \Big(a_1^b, e_2^b, a_3^b, a_4^b, e_5^b\Big),$$
$$\Big(d_1^b, a_2^b, d_3^b, d_4^b, a_5^b\Big), \Big(d_1^b, e_2^b, d_3^b, d_4^b, e_5^b\Big), \Big(\alpha_1^b, \alpha_2^b, \alpha_3^b, \alpha_4^b, \alpha_5^b\Big),$$
$$\Big(\alpha_1^b, \beta_2^b, \alpha_3^b, \alpha_4^b, \beta_5^b\Big), \Big(\beta_1^b, \alpha_2^b, \beta_3^b, \beta_4^b, \alpha_5^b\Big), \Big(\beta_1^b, \beta_2^b, \beta_3^b, \beta_4^b, \beta_5^b\Big)\Big\}.$$

The synthetic training set Ω´ having eight patterns is larger in size than the original training set Ω, having six patterns. In this way the training set size can be increased by multiple kernel learning.

3. Proposed system

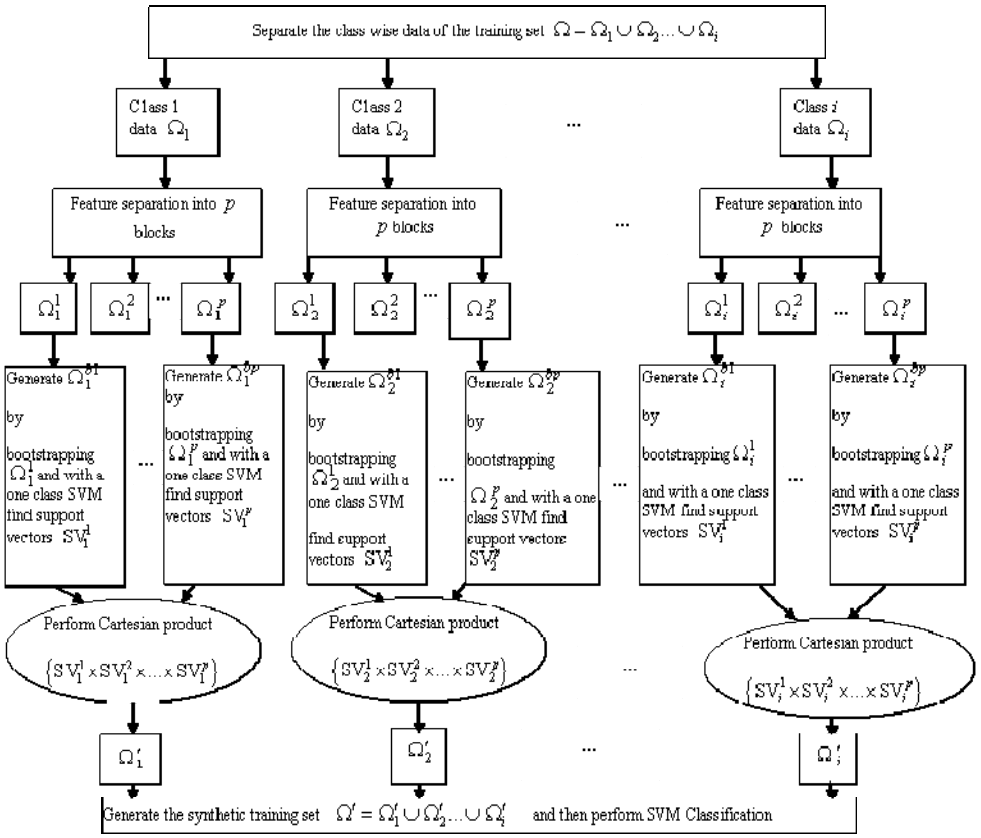


Fig. 1. Generating synthetic patterns using multiple kernel learning. The proposed system

The proposed system is shown in Fig. 1. The features of the class wise partitions of the training set are separated into p blocks where p =2, 3, and 4, using the correlation based feature separation method explained in Section 3. The class wise data is represented as Ω_1, Ω_2, …, Ω_i corresponding to class labels C_1, C_2, …, C_i respectively and each of them is partitioned into p blocks denoted by $\Omega_1^1, \Omega_1^2, ..., \Omega_1^p, \Omega_2^1, \Omega_2^2, ..., \Omega_2^p, ..., \Omega_i^1, \Omega_i^2, ..., \Omega_i^p$ respectively. Bootstrapping, suggested by Hamamoto et al. [1] is applied on each of these blocks. Thus

each of these blocks now contains bootstrapped data given by $\Omega_1^{b1}, \Omega_1^{b2}, ..., \Omega_1^{bp}, \Omega_2^{b1}, \Omega_2^{b2}, ..., \Omega_2^{bp}, ..., \Omega_i^{b1}, \Omega_i^{b2}, ..., \Omega_i^{bp}$. Support vectors are generated from each of these blocks with one class of SVM classifier $\mathrm{SV}_1^1, \mathrm{SV}_1^2, ..., \mathrm{SV}_1^p, \mathrm{SV}_2^1, \mathrm{SV}_2^2, ..., \mathrm{SV}_2^p, ..., \mathrm{SV}_i^1, \mathrm{SV}_i^2, ..., \mathrm{SV}_i^p$. Thus, a single kernel, i.e., either linear, RBF or polynomial kernel is applied commonly on each of these blocks. Then the Cartesian products of the support vectors of all the class wise blocks generate a new data set for each class, i.e.,

$$\Omega_1' = \left\{\mathrm{SV}_1^1 \times \mathrm{SV}_1^2 \times ... \times \mathrm{SV}_1^p\right\},\ \Omega_2' = \left\{\mathrm{SV}_2^1 \times \mathrm{SV}_2^2 \times ... \times \mathrm{SV}_2^p\right\}, ...,$$

$$\Omega_i' = \left\{\mathrm{SV}_i^1 \times \mathrm{SV}_i^2 \times ... \times \mathrm{SV}_i^p\right\}.$$

The class wise simulated patterns are then used to generate a larger training set represented by $\Omega' = \Omega_1' \cup \Omega_2' \cup ... \cup \Omega_i'$. This synthetic training set is used for the final SVM classification with the same kernel function that is used on each of the blocks. Thus a novel multiple kernel learning approach is applied to generate synthetic patterns.

4. Feature separation method

In this paper we used the partitioning method suggested by Viswanath et al. [10] for efficient nearest neighbour classification, in order to separate the features of each class of the training data into uncorrelated blocks. This method is based on pair-wise correlation between the features and therefore is suitable for data, having numerical feature values only. The objective of this method is to find blocks of features in such a way that the average correlation between the features within a block is high and that between features of different blocks is low. Since this objective is a computationally demanding one, a greedy method which can find only a locally optimal partition was suggested by Viswanath et al. [10].

5. Bootstrapping

The bootstrapping method that we employed in this paper is different from the ordinary bootstrapping in the manner in which the bootstrap samples are generated. The ordinary bootstrapping is a method of resampling the given data and has been a successful method for error estimation [15-18]. The bootstrapping method that creates (not selects) new training samples was proposed by Hamamoto et al. [1] that acts as a smoother of the distribution of the training samples and was successfully applied in the design of 1NN classifier, particularly in high dimensional spaces. Further, Hamamoto et al. [1] generated bootstrap samples by combining the training data locally and illustrated that the NNC (Nearest Neighbour Classifier) based on bootstrap patterns performed better than that of K-NNC (K-nearest-neighbor classifier) based on the original data [18].

In the present work, we applied the bootstrapping method suggested by Hamamoto et al. [1] to each block as shown by the following algorithm.

Algorithm 1. Generating bootstrapped sub-patterns

Input: $X_w^j = \left\{ X_{1w}^j, X_{2w}^j, ..., X_{Nw}^j \right\}$, the original set of sub-patterns from block B_j of class C_w.

Step 1. Select a block B_j of class C_w and initialize, where $X_w^{bj} = \phi$, where X_w^{bj} represents the set of bootstrapped sub-patterns of block B_j of class C_w.

Step 2. Set m =1.

Step 3. Select m-th sub-pattern X_{mw}^j from block B_j of class C_w.

Step 4. Find the r nearest neighbour sub-patterns $X_{1w}^j, X_{2w}^j, ..., X_{rw}^j$ of X_{mw}^j in block B_j of class C_w using Euclidean distance.

Step 5. Determine m-th bootstrapped sub-pattern $X_{mw}^{bj} = \frac{1}{r} \sum_{h=1}^{r} X_{hw}^j$.

Step 6. $X_w^{bj} = X_w^{bj} \cup \left\{ X_{mw}^{bj} \right\}$.

Step 7. Repeat Steps 3-5 for $m = 2, ..., N$.

Step 8. Output the synthetic set $X_w^{bp} = \left\{ X_{1w}^{bj}, X_{2w}^{bj}, ..., X_{Nw}^{bj} \right\}$ of bootstrapped sub-patterns generated for block B_j of class C_w.

Step 9. Repeat Steps 1-7 for $j = 1, 2, ..., p$.

Step 10. Repeat Steps 1-8 for $w = 1, 2, ..., i$.

In Step 3 the sub-patterns from block B_j are selected so that no sub-pattern is chosen more than once. Thus a synthetic set of bootstrap sub-patterns is generated for each of the blocks belonging to every class. The bootstrapping technique has the ability to remove outliers which therefore reduces the variability in the data, as well as removes noise. This in turn increases the distance between two close patterns belonging to different classes and thereby improves the generalization performance of the classifier [18].

6. Experimental study

The proposed system is implemented with seven of the benchmark datasets viz., Thyroid, Ionosphere, Glass, Wine, Breast Cancer and Sonar obtained from UCI machine learning repository [19]. OCR data set was also used by Viswanath et al. [10]. The characteristics of these datasets, i.e., the number of features, the number of the training patterns, the number of the testing patterns and the number of the classes are shown in the Table 2. (It is noted that in Glass data there is no data corresponding to class label 4). For Thyroid and OCR datasets the training and testing set are separately available. For all the other datasets, approximately the first 60% of the data of each class is used for training and the remaining data of each class is utilized for testing. The features of all these datasets have numerical values.

Except OCR, the features of Thyroid, Glass, Wine, Breast Cancer and Sonar datasets are normalized to zero mean and unit variance.

Table 2. Characteristics of datasets used

Data Set	Number of features	Number of training patterns	Number of testing patterns	Number of classes
Thyroid	21	3772	3428	3
Ionosphere	34	216	135	2
Wine	13	108	70	3
Glass	9	130	84	6
Breast Cancer	30	342	227	2
Sonar	60	125	83	2
OCR	192	300	3333	10

The experiments are performed as follows:

Scheme 1. Generating synthetic patterns based on the proposed system using a linear kernel and performing SVM classification using the linear kernel finally.

Scheme 2. Synthesizing new patterns applying the proposed approach using RBF kernel and performing SVM classification using RBF kernel.

Scheme 3. Producing artificial patterns using the proposed system with a polynomial kernel and finally performing SVM classification using the polynomial kernel.

In all these schemes, initially each dataset is partitioned classwise. The classwise partition of each dataset is then divided into p blocks using the algorithm for the correlation based feature partitioning discussed in Section 3. Each block consists of features that are better correlated with each other than the features in different blocks. Each block of data is bootstrapped. xperiments are performed varying the number of blocks, i.e., p =2, 3 and 4 only because earlier studies [10] showed that increasing the number of blocks does not improve the performance. The experiments are implemented in MATLAB, and LIBSVM is used both as one class of a SVM classifier on the blocks of features and also for the final SVM classification using a synthetic training set [20].

The same C parameter value was used for SVM classification on the original data and for the final SVM classification using a synthetic training set in case of a linear, RBF and polynomial kernel respectively. This value of C was chosen to be a default value (i.e., C=1) for all the data sets using a linear kernel. In case of RBF and Polynomial for all the data sets except OCR, this value of C was chosen to be a default value (and the other parameters, such as γ in case of RBF and degree in case of a polynomial were also chosen to have default values of LIBSVM tool as shown in Table 15 Appendix). For OCR data C = 0.5 in case of RBF and C = 0.03125 in case of a polynomial kernel are used. These values are respectively determined by varying C, and noting the CA% (classification accuracy) of the proposed system, as well as CA% of the original data and fixing C to the value where the CA% of the proposed system was higher than the CA% of the original data.

In *Scheme 1*, varying ν parameter of one class of SVM classifier (with other parameters of one class of a linear SVM classifier being default values, as given by LIBSVM as shown in Table 15 in Appendix) and the number of nearest neighbours (k) for bootstrapping, appropriate number of support vectors are selected from each block for each class of data and then Cartesian product is performed such that the new training data is generated for that class. For the value of C (used on the original data and for the final SVM classification), the ν_{cb} parameter values for each block b of each class c is fixed at those values for which the CA% of the proposed method is higher than the CA% of the original data. These values are shown in Tables 6-8 of the Appendix respectively. The number of the nearest neighbours r_m for which the maximum CA% is obtained, using the method proposed, is also noted and shown in Tables 3-5 respectively.

Table 3. CA% obtained by applying proposed system with a linear kernel

Data Set	On original data CA%	On applying proposed system		
		Number of partitions (p)	r_m	CA%
Thyroid	93.0572	2	21	97.287
		3	37	97.4037
		4	53	97.3454
Ionosphere	91.1111	2	66	91.8519
		3	27	91.8519
		4	44	91.8519
Wine	97.2222	2	7	100
		3	11	98.611
		4	12	98.611
Glass	57.1429	2	5	72.619
		3	4	72.619
		4	5	71.4286
Breast Cancer	96.4758	2	54	98.2379
		3	64	97.7974
		4	84	97.3568
Sonar	62.6506	2	4	72.2892
		3	9	74.6988
		4	30	81.9277
OCR	81.4881	2	6	82.6283
		3	6	82.6883
		4	28	70.4770

The same procedure is followed for RBF and polynomial kernels in *Scheme 2* and *Scheme 3* respectively. For Thyroid data using RBF, γ parameter values, for $p = 2$ and $p = 3$ blocks (for each block using one class of a SVM classifier) chosen different from the default values, as shown in Tables 9-10 whereas for other data sets the γ parameter (for each block using one class of a SVM classifier) were chosen to have default values (as given in LIBSVM tool). For a polynomial kernel, except the nu (ν_{cp}) parameter, all others were chosen to have default values of LIBSVM tool in case of one class of a SVM classifier as shown in Tables 12-15 in the Appendix. The experimental results of *Scheme 2* and *Scheme 3* are shown in Tables 4-5 respectively.

Table 4. CA% obtained by applying the proposed system with RBF kernel

Data set	On original data CA%	On applying proposed system		
		Number of partitions (p)	r_m	CA%
Thyroid	94.895	2	6	97.4329
		3	34	96.0035
		4	50	95.4492
Ionosphere	93.3333	2	12	96.2963
		3	2	94.8148
		4	2	94.0741
Wine	98.6111	2	4	100
		3	24	100
		4	12	100
Glass	66.6667	2	3	78.5714
		3	3	72.619
		4	3	72.619
Breast Cancer	96.4758	2	4	98.6784
		3	13	97.7974
		4	34	96.9163
Sonar	49.3976	2	24	74.6988
		3	3	74.6988
		4	37	84.3373
OCR(C=0.5)	76.9277	2	2	84.0684
		3	2	84.1884
		4	2	75.6076

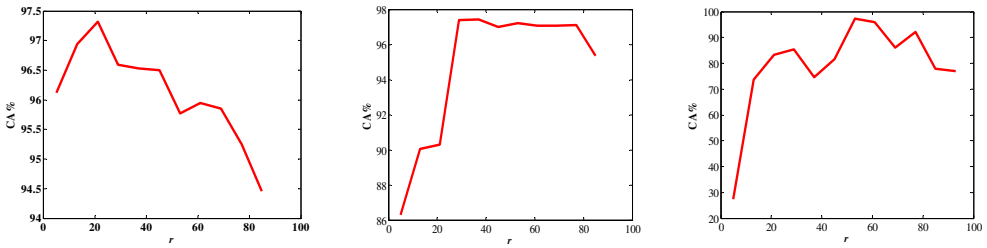
Table 5. CA% obtained by applying the proposed system with a polynomial kernel

Data set	On original data CA%	On applying proposed system		
		Number of partitions (p)	r_m	CA%
Thyroid	93.7573	2	78	93.9615
		3	50	94.049
		4	63	93.9032
Ionosphere	64.4444	2	5	91.8519
		3	15	91.1111
		4	2	77.037
Wine	91.6667	2	6	95.8333
		3	4	98.6111
		4	9	94.4444
Glass	51.1905	2	4	72.6190
		3	2	71.4286
		4	5	71.4286
Breast Cancer	91.63	2	54	98.2379
		3	30	97.3568
		4	40	96.0352
Sonar	46.988	2	32	75.9036
		3	7	75.9036
		4	16	80.7229
OCR	77.0777	2	2	79.8080
		3	2	79.5380
		4	26	69.0669

From Tables 3-5 it can be summarized that RBF kernel showed better performance for all the datasets. Generally, the linear kernel is preferred as it performs well when the number of features is large when compared to the size of the data, but the experimental results showed that RBF kernel showed good performance on using the proposed system. This may be because of the sufficiently available training patterns. The disadvantage of a linear kernel is that it performs poorly in case of noisy data. In the proposed system the noise is removed by bootstrapping and hence, it showed better performance using the proposed system as shown in Table 3. Hard margin classifier is easily affected by noise. Although soft margin SVM classifiers were introduced to overcome this difficulty, the set of support vectors may have noisy patterns. The preprocessing that is applied in the proposed method, i.e bootstrapping, reduces the impact of such noisy patterns.

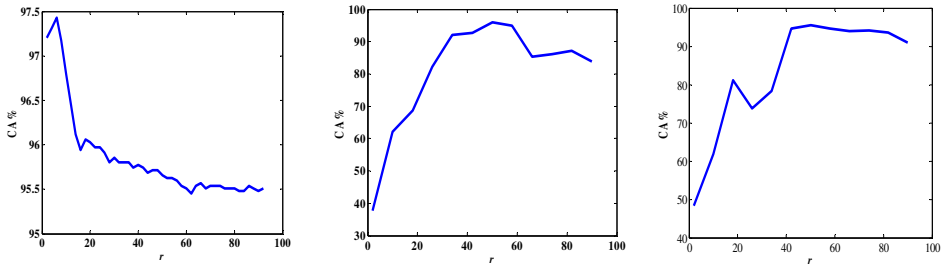
For Breast Cancer data using all three kernels the CA% decreased with increasing the number of blocks. This may be due to overlearning, as the size of the training data increases with increase in the number of blocks. An almost similar observation could be made on Glass data using all three kernels, Wine, Ionosphere & OCR data using a polynomial kernel, Thyroid &Ionosphere data using RBF kernel. For Thyroid data using a linear kernel, OCR data using RBF and linear kernels, the maximum CA% using the proposed system, it was obtained for p =3 blocks. This shows that if insufficient training data (for p = 2) is used then the output will not be a true representative of the input and if the size of the training data is more (for p = 4) then it causes overfitting. For Sonar data using all three kernels the highest CA% is obtained for p = 4 blocks. This may be due to the requirement for a larger number of training patterns.

Figs 2-4 have been plotted to study the effect of bootstrapping for different number of blocks used for pattern synthesis, on the classification performance of the SVM classifier using linear, RBF and polynomial kernels respectively. Fig. 2 shows the influence of the number of the nearest neighbours (r) chosen for bootstrapping, on CA% of a SVM classifier using the linear kernel for Thyroid data, for p = 2, 3 and 4. Similarly, Figs 3 and 4 display the variation in CA% of the SVM classifier with a varying number of the nearest neighbors used for bootstrapping, for p =2, 3 and 4, using RBF and a polynomial kernel for the Thyroid data respectively.



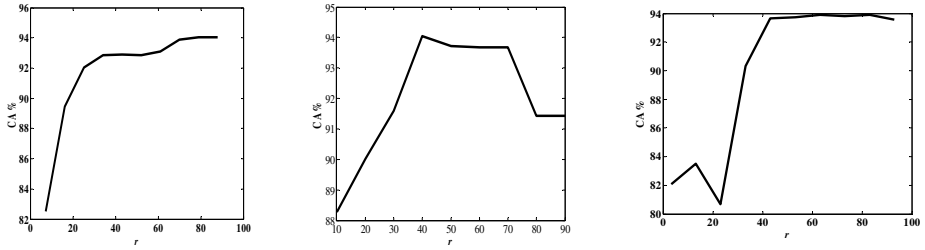
p=2 p=3 p=4

Fig. 2. CA% vs r using a linear kernel for Thyroid data



p=2 *p*=3 *p*=4

Fig. 3. CA% vs r using RBF kernel for Thyroid data



p=2 *p*=3 *p*=4

Fig. 4. CA% vs r using a polynomial kernel for Thyroid data

From Figs 2-4 it is clear that as the number of the nearest neighbours (r) increases, the CA% first increases, reaches maximum at r_m and then decreases. This is explained by the different number of blocks (p =2, 3 and 4) using a linear, RBF and polynomial kernels respectively. A similar observation was made even in case of other data sets. This is because if the number of the nearest neighbors is less, then smoothing is less, causing overfitting and increasing the number of the nearest neighbors causes excessive smoothing leading to underfitting of the data (see [21, 22]).

7. Conclusions

In the present work a novel method to synthesize training patterns is proposed based on multiple kernel learning approach to subdue the effects of high dimensionality on classifying small samples of data with a SVM classifier. This method increases the size of the training samples to vanquish the effect of 'Curse of dimensionality'. Experimental studies are performed on seven standard datasets viz., Thyroid, Ionosphere, Glass, Wine, Breast Cancer, Sonar and OCR data, using linear, RBF and polynomial kernels separately. The main findings are summarized below:

- Experimental results showed that the SVM classifier, trained using synthetic patterns outperformed the conventional SVM classifier trained on original data and hence it can be concluded that the synthetic pattern generation improves the generalization performance of the SVM classifier.
- Experimental observations demonstrated that synthetic pattern generation reduced the effect of the curse of dimensionality that occurs when the

dimensionality is larger than the size of the data and hence, the CA% obtained by a SVM classifier using the proposed system was better than the CA% obtained by the conventional SVM classifier.

- The size of the training set can be increased by increasing the number of blocks of features, but it is shown experimentally that it may not increase the performance of the classifier always, which may be due to the increase in the deviation from the original training set.
- The proposed method is suitable for the datasets having high dimensionality, but not very high dimensionality, as the computational time and the memory resources for finding the correlation (used for partitioning the features) between the features of the data increases with dimensionality.
- The experimental results were in good agreement with the results reported by Viswanath et al. [10, 11] on pattern synthesis for nearest neighbour classification.
- The figures showed the variation of CA% with variation in the number of the nearest neighbors and demonstrated the profound effect of smoothing of the training patterns on the performance of the SVM classifier. These results were in good agreement with the report made by Hamamoto et al. [1], that bootstrapping technique removes noise by smoothing training patterns, particularly in high dimensional spaces.

Synthetic pattern generation suggested in this paper is helpful, because it is costly to get large real world patterns. Our future work will be directed to overcome the limitation of the proposed method (that is increase in the training time of the SVM classifier due to increase in the size of the training set) by using greedy methods, instead of Cartesian product, to generate synthetic patterns .

Acknowledgements: The authors gratefully acknowledge Dr. P. Viswanath (Dean (R & D), Dept. of CSE, RGMCET, Nandyal, A. P., India) for giving OCR data.

References

1. Hamamoto, Y., S. Uchimura, S. Tomita. A Bootstrap Technique for Nearest Neighbor Classifier Design. – IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. **19**, 1997, No 1, 73-79.
2. Jain, A., B. Chandrasekharan. Dimensionality and Sample Size Considerations in Pattern Recognition Practice. – In: P. Krishnaiah, L. Kanal, Eds. Handbook of Statistics. Vol. **2**. North Holland, 1982, 835-855.
3. Duda, R. O., P. E. Hart, D. G. Stork. Pattern Classification. John Wiley & Sons, Inc., 2005.
4. Hastie, T., R. Tibshirani, J. Friedman. The Elements of Statistical Learning. Second Edition. Springer Series in Statistics, 2009.
5. Silverman, B. W. Density Estimation for Statistics and Data Analysis. London, Chapman &Hall, 1986.
6. Beyer, K. S., J. Goldstein, R. Ramakrishnan, U. Shaft. When is "Nearest Neighbor" Meaningful? –In: Proc. of 7th International Conference on Database Theory, ICDT'99, London, UK, 1999. Springer Verlag, 217-235.
7. Sathiya, K. S., Chih-Jen Lin. Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel. – Neural Computation, Vol. **15**, 2003, No 7, 1667-1689.

8. Fillipone, M., F. Camastra, F. Masulli, S. Revatta. A Survey of Kernel and Spectral Methods for Clustering. – Pattern Recognition,Vol. **41**, 2008, 176-190.
9. Evangelista, P. F., M. J. Embrechts, B. K. Szymanski. Taming the Curse of Dimensionality in Kernels and Novelty Detection, Applied Soft Computing Technologies: The Challenge of Complexity. A. Abraham, B. Baets, M. Koppen, B. Nickolay, Eds. Berlin, Springer Verlag, 2006.
10. Viswanath, P., M. N. Murty, S. Bhatnagar. Partition Based Pattern Synthesis Technique with Efficient Algorithms for Nearest Neighbor Classification. – Pattern Recognition Letters, Vol. **27**, 2006, No 14, 1714-1724.
11. Viswanath, P., M. N. Murty, S. Bhatnagar. Fusion of Multiple Approximate Nearest Neighbor Classifiers for Fast and Efficient Classification. – Information Fusion, Vol. **5**, 2004, 239-250.
12. Agrawal, M., N. Gupta, R. Shreelekshmi, M. N. Murty. Efficient Pattern Synthesis for Nearest Neighbor Classifier. – Pattern Recognition, Vol. **38**, 2005, No 11, 2200-2203.
13. Lanckriet, G., N. Cristianini, P. Bartlett, L. El Ghaoui, M. Jordan. Learning the Kernel Matrix with Semi-Definite Programming. – Journal of Machine Learning Research, Vol. **5**, 2004.
14. Sonnenburg, S., G. Rätsch, C. Schäfer, B. Schölkopf. Large Scale Multiple Kernel Learning. – Journal of Machine Learning Research, Vol. **7**, 2006.
15. Jain, A. K., R. C. Dubes, C.-C. Chen. Bootstrap Techniques for Error Estimation. – IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. **9**, 1987, 628-633.
16. Chernick, M. C., V. K. Murthy, C. D. Nealy. Application of Bootstrap and Other Resampling Techniques: Evaluation of Classifier Performance. – Pattern Recognition Letters, Vol. **3**, 1985, 167-178.
17. Weiss, S. M. Small Sample Error Rate Estimation for k-NN Classifiers. – IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. **13**, 1991, 285-289.
18. Saradhi, V. V., M. N. Murty. Bootstrapping for Efficient Handwritten Digit Recognition. – Pattern Recognition, Vol. **34**, 2001, No 5, 1047-1056.
19. Murphy, P. M. UCI Repository of Machine Learning Databases. Department of Information and Computer Science. University of California, Irvine, CA, 1994. **http://www.ics.uci.edu/mlearn/MLRepository.html**
20. Chang, C.-C., C.-J. Lin. LIBSVM: A Library for Support Vector Machines. 2001. **http://www.csie.ntu.edu.tw/~cjlin/libsvm**.
21. Seetha, H., M. N. Murty, R. Saravanan. On Improving the Generalization of SVM Classifier. – In K. R. Venugopal, L. M. Patnaik, Eds., ICIP'2011, CCIS 157, 2011, 11-20.
22. Seetha, H., M. N. Murty, R. Saravanan. A Note on the Effect of Bootstrapping and Clustering on the Generalization Performance. – International Journal of Information Processing, Vol. **5**, 2011, No 4,19-34.

Appendix

Table 6. Parameter values chosen for one class of a SVM classifier in case of two partitions using a linear kernel

Dataset	Parameter values
Thyroid	v_{11}=0.12, v_{12}=0.1, v_{21}=0.028, v_{22}=0.07, v_{31}=0.050, v_{32}=0.4
Ionosphere	v_{11}=0.08, v_{12}=0.003, v_{21}=0.3, v_{22}=0.4
Breast cancer	v_{11}=0.3, v_{12}=0.3, v_{21}=0.4, v_{22}=0.4
Sonar	v_{11}=0.3, v_{12}=0.7, v_{21}=0.009, v_{22}=0.1
Wine	v_{11}=0.2, v_{12}=0.1, v_{21}=0.2, v_{22}=0.2, v_{31}=0.1, v_{32}=0.4
Glass	v_{11}=0.2, v_{12}=0.2, v_{21}=0.1, v_{22}=0.4, v_{31}=0.1, v_{32}=0.3, v_{51}=0.2, v_{52}=0.3, v_{61}=0.3, v_{62}=0.5, v_{71}=0.2, v_{72}=0.2
OCR	v_{11}= v_{21}= v_{31}= v_{41}= v_{51}= v_{61}= v_{71}= v_{81}= v_{91}=0.9, v_{12}= v_{22}= v_{32}= v_{42}= v_{52}= v_{62}= v_{72}= v_{82}= v_{92}=0.9

Table 7. Parameter values chosen for one class of a SVM classifier in case of three partitions using a linear kernel

Dataset	Parameter values
Thyroid	v_{11}=0.1, v_{12}=0.1, v_{13}=0.1, v_{21}=0.06, v_{22}=0.06, v_{23}=0.06, v_{31}=0.003, v_{32}=5×10^{-6}, v_{33}=0.002
Ionosphere	v_{11}=0.1, v_{12}=0.01, v_{13}=0.1, v_{21}=0.01, v_{22}=0.2, v_{23}=0.2
Breast cancer	v_{11}=0.3, v_{12}=0.1, v_{13}=0.001, v_{21}=0.04, v_{22}=0.01, v_{23}=0.01
Sonar	v_{11}=1×10^{-5}, v_{12}=1.5×10^{-5}, v_{13}=0.01, v_{21}=1.5×10^{-5}, v_{22}=1×10^{-5}, v_{23}=1.4×10^{-5}
Wine	v_{11}=0.1, v_{12}=0.1, v_{13}=0.1, v_{21}=0.002, v_{22}=0.06, v_{23}=0.06, v_{31}=0.1, v_{32}=0.1, v_{33}=0.4
Glass	v_{11}=0.1, v_{12}=0.15, v_{13}=0.15, v_{21}=0.3, v_{22}=0.1, v_{23}=0.08, v_{31}=0.1, v_{32}=0.02, v_{33}=0.01, v_{51}=0.01, v_{52}=0.02, v_{53}=0.5, v_{61}=0.01, v_{62}=0.35, v_{63}=0.1, v_{71}=0.2, v_{72}=0.01, v_{73}=0.01
OCR	v_{11}= v_{21}= v_{31}= v_{41}= v_{51}= v_{61}= v_{71}= v_{81}= v_{91}=0.4, v_{12}= v_{22}= v_{32}= v_{42}= v_{52}= v_{62}= v_{72}= v_{82}= v_{92}=0.6, v_{13}= v_{23}= v_{33}= v_{43}= v_{53}= v_{63}= v_{73}= v_{83}= v_{93}=0.9

Table 8. Parameter values chosen for one class of a SVM classifier in case of four partitions using a linear kernel

Dataset	Parameter values
Thyroid	v_{11}=0.05, v_{12}=0.05, v_{13}=0.05, v_{14}=0.05, v_{21}=0.03, v_{22}=0.02, v_{23}=0.02, v_{24}=0.02, v_{31}=9×10^{-6}, v_{32}=9×10^{-6}, v_{33}=9×10^{-6}, v_{34}=9×10^{-6}
Ionosphere	v_{11}=0.001, v_{12}=0.01, v_{13}=0.01, v_{14}=0.00003, v_{21}=0.01, v_{22}=0.01, v_{23}=0.01, v_{24}=0.01
Breast cancer	v_{11}=0.2, v_{12}=0.01, v_{13}=1×10^{-5}, v_{14}=0.001, v_{21}=3×10^{-4}, v_{22}=0.001, v_{23}=0.01, v_{24}=0.01
Sonar	v_{11}=1×10^{-4}, v_{12}=1×10^{-5}, v_{13}=1×10^{-5}, v_{14}=0.001, v_{21}=3×10^{-6}, v_{22}=1×10^{-5}, v_{23}=0.004, v_{24}=0.004
Wine	v_{11}=0.1, v_{12}=0.2, v_{13}=0.01, v_{14}=0.01, v_{21}=0.002, v_{22}=0.002, v_{23}=0.001, v_{24}=0.0001, v_{31}=0.1, v_{32}=0.01, v_{33}=0.01, v_{34}=0.01
Glass	v_{11}=0.001, v_{12}=0.003, v_{13}=0.1, v_{14}=0.1, v_{21}=0.4, v_{22}=0.1, v_{23}=0.01, v_{24}=0.001, v_{31}=0.01, v_{32}=0.001, v_{33}=0.001, v_{34}=0.0001, v_{51}=0.001, v_{52}=0.01, v_{53}=0.01, v_{54}=0.01, v_{61}=0.01, v_{62}=0.01, v_{63}=0.3, v_{64}=0.3, v_{71}=0.02, v_{72}=0.02, v_{73}=0.01, v_{74}=0.01
OCR	v_{11}= v_{21}= v_{31}= v_{41}= v_{51}= v_{61}= v_{71}= v_{81}= v_{91}=0.25, v_{12}= v_{22}= v_{32}= v_{42}= v_{52}= v_{62}= v_{72}= v_{82}= v_{92}=0.125, v_{13}= v_{23}= v_{33}= v_{43}= v_{53}= v_{63}= v_{73}= v_{83}= v_{93}=0.125, v_{14}= v_{24}= v_{34}= v_{44}= v_{54}= v_{64}= v_{74}= v_{84}= v_{94}=0.5

Table 9. Parameter values chosen for one class of a SVM classifier in case of two partitions using RBF kernel

Dataset	Parameter values
Thyroid	v_{11}=0.01, v_{12}=0.01, v_{21}=0.01, v_{22}=0.03, v_{31}=0.0001, v_{32}=0.0009, γ_{11}=0.9, γ_{12}=0.9, γ_{21}=0.4, γ_{22}=0.4, γ_{31}=0.9, γ_{32}=0.9
Ionosphere	v_{11}=0.1, v_{12}=0.1, v_{21}=0.3, v_{22}=0.7
Breast cancer	v_{11}=0.4, v_{12}=0.1, v_{21}=0.2, v_{22}=0.2
Sonar	v_{11}=0.1, v_{12}=0.2, v_{21}=0.1, v_{22}=0.3
Wine	v_{11}=0.1, v_{12}=0.3, v_{21}=0.1, v_{22}=0.2, v_{31}=0.02, v_{32}=0.05
Glass	v_{11}=0.2, v_{12}=0.2, v_{21}=0.3, v_{22}=0, V_{31}=0.1, v_{32}=0.1, v_{51}=0.2, v_{52}=0.2, v_{61}=0.3, v_{62}=0.3, v_{71}=0.2, v_{72}=0.1
OCR	v_{11}= v_{21}= v_{31}= v_{41}= v_{51}= v_{61}= v_{71}= v_{81}= v_{91}=0.9, v_{12}= v_{22}= v_{32}= v_{42}= v_{52}= v_{62}= v_{72}= v_{82}= v_{92}=0.99

Table 10. Parameter values chosen for one class of a SVM classifier in case of three partitions using RBF kernel

Dataset	Parameter values
Thyroid	v_{11}=0.0001, v_{12}=0.0003, v_{13}=0.0004, v_{21}=1.5×10^{-4}, v_{22}=0.0001, v_{23}=0.03, v_{31}=1×10^{-5}, v_{32}=1.5×10^{-5}, v_{33}=5×10^{-5}, γ_{11}=0.9, γ_{12}=0.9, γ_{13}=0.9, γ_{21}=0.9, γ_{22}=0.9, γ_{23}=0.9, γ_{31}=0.9, γ_{32}=0.9, γ_{33}=0.9
Ionosphere	v_{11}=0.01, v_{12}=0.2, v_{13}=0.2, v_{21}= v_{22}= v_{23}=0.1
Breast cancer	v_{11}=3×10^{-4}, v_{12}=0.2, v_{13}=0.1, v_{21}=0.002, v_{22}=0.0015, v_{23}=0.0015
Sonar	v_{11}=0.7, v_{12}=1.5×10^{-4}, v_{13}= 0.9, v_{21}=0.004, v_{22}=0.001, v_{23} =0.9
Wine	v_{11}=0.01, v_{12}=0.01, v_{13}=0.001, v_{21}=1×10^{-4}, v_{22}= 0.001, v_{23}=0.001, v_{31}=0.001, v_{32}=0.001, v_{33}=0.00001
Glass	v_{11}=0.35, v_{12}=0.13, v_{13}=0.11, v_{21}=0.2, v_{22}=0.02, v_{23}=0.0008, v_{31}=0.01, v_{32}=0.01, v_{33}=0.01, v_{51}=0.03, v_{52}=0.02, v_{53}=0.001, v_{61}=0.03, v_{62}=0.03, v_{63}=0.03, v_{71}= v_{72}= v_{73}=0.001
OCR	v_{11}= v_{21}= v_{31}= v_{41}= v_{51}= v_{61}= v_{71}= v_{81}= v_{91}=0.4, v_{12}== v_{22}= v_{32}= v_{42}= v_{52}= v_{62}= v_{72}= v_{82}= v_{92}=0.7, v_{13}= v_{23}= v_{33}= v_{43}= v_{53}= v_{63}= v_{73}= v_{83}= v_{93}=0.9

Table 11. Parameter values chosen for one class SVM classifier in case of four partitions using RBF Kernel

Dataset	Parameter values
Thyroid	v_{11}=1.5×10^{-5}, v_{12}=1.5×10^{-4}, v_{13}=2×10^{-4}, v_{14}=0.002, v_{21}=3×10^{-5}, v_{22}=0.025, v_{23}=2×10^{-4}, v_{24}=0.001, v_{31}=1×10^{-4}, v_{32}=5×10^{-6}, v_{33}=3×10^{-6}, v_{34}=1×10^{-4}
Ionosphere	v_{11}=0.001, v_{12}=0.01, v_{13}=1×10^{-5}, v_{14}=0.0002, v_{21}=0.004, v_{22}=0.001, v_{23}=0.001, v_{24}=0.0001
Breast cancer	v_{11}=1×10^{-4}, v_{12}=1×10^{-5}, v_{13}=1×10^{-4}, v_{14}=1×10^{-4}, v_{21}=3×10^{-5}, v_{22}=1×10^{-4}, v_{23}=1×10^{-5}, v_{24}=1×10^{-5}
Sonar	v_{11}=0.001, v_{12}=0.001, v_{13}=0.016, v_{14}=0.38, v_{21}=4×10^{-6}, v_{22}=0.015, v_{23}=0.1, v_{24}=0.85
Wine	v_{11}=0.0001, v_{12}=0.0001, v_{13}=0.0001, v_{14}=0.0001, v_{21}=0.001, v_{22}=0.001, v_{23}=0.00001, v_{24}=0.00001, v_{31}=0.001, v_{32}=0.001, v_{33}=0.0001, v_{34}=0.0001
Glass	v_{11}=0.01, v_{12}=0.0001, v_{13}=0.0001, v_{14}=0.0001, v_{21}=0.0001, v_{22}=0.0001, v_{23}=0.001, v_{24}=0.01, v_{31}=0.01, v_{32}=0.001, v_{33}=0.001, v_{34}=0.0001, v_{51}=0.001, v_{52}=0.01, v_{53}=0.01, v_{54}=0.01,v_{61}=0.001, v_{62}=0.001, v_{63}=0.001, v_{64}=0.001, v_{71}=0.001, v_{72}=0.001, v_{73}=0.001, v_{74}=0.001
OCR	v_{11}= v_{21}= v_{31}= v_{41}= v_{51}= v_{61}= v_{71}= v_{81}= v_{91}=2×10^{-10}, v_{12}= v_{22}= v_{32}= v_{42}= v_{52}= v_{62}= v_{72}= v_{82}= v_{92}=2×10^{-10}, v_{13}= v_{23}= v_{33}= v_{43}= v_{53}= v_{63}= v_{73}= v_{83}= v_{93}=2×10^{-10}, v_{14}= v_{24}= v_{34}= v_{44}= v_{54}= v_{64}= v_{74}= v_{84}= v_{94}=2×10^{-10}

Table 12. Parameter values chosen for one class SVM classifier in case of two partitions using Polynomial Kernel

Dataset	Parameter values
Thyroid	v_{11}=0.1, v_{12}=0.08, v_{21}=0.03, v_{22}=0.03, v_{31}=0.028, v_{32}=0.02
Ionosphere	v_{11}=0.4, v_{12}=0.1, v_{21}=0.1, v_{22}=0.2
Breast cancer	v_{11}=0.3, v_{12}=0.2, v_{21}=0.2, v_{22}=0.4
Sonar	v_{11}=0.3, v_{12}=0.45, v_{21}=0.5, v_{22}=0.5
Wine	v_{11}=0.2, v_{12}=0.09, v_{21}=0.35, v_{22}=0.05, v_{31}=0.5, v_{32}=0.03
Glass	v_{11}=0.2, v_{12}=0.2, v_{21}=0.3, v_{22}=0.001, v_{31}=0.01, v_{32}=0.01, v_{51}=0.3, v_{52}=0.2, v_{61}=0.3, v_{62}=0.3, v_{71}=0.2, v_{72}=0.2
OCR	v_{11}= v_{21}= v_{31}= v_{41}= v_{51}= v_{61}= v_{71}= v_{81}= v_{91}=0.9, v_{12}= v_{22}= v_{32}= v_{42}= v_{52}= v_{62}= v_{72}= v_{82}= v_{92}=0.99

Table 13. Parameter values chosen for one class SVM classifier in case of three partitions using Polynomial Kernel

Dataset	Parameter values
Thyroid	v_{11}=0.2, v_{12}=0.04, v_{13}=0.0005, v_{21}=0.03, v_{22}=0.008, v_{23}=0.0004, v_{31}=1×10^{-5}, v_{32}=2×10^{-5}, v_{33}=2×10^{-5}
Ionosphere	v_{11}=0.01, v_{12}=0.13, v_{13}=0.2, v_{21}=0.1, v_{22}=0.1, v_{23}=0.15 (neg)
Breast cancer	v_{11}=0.2, v_{12}=0.02, v_{13}=0.04, v_{21}=0.3, v_{22}=0.02, v_{23}=0.01 (pos)
Sonar	v_{11}=0.04, v_{12}=0.1, v_{13}= 0.1, v_{21}=0.003, v_{22}=0.01, v_{23} =0.01(pos)
Wine	v_{11}= 0.15, v_{12}=0.1, v_{13}=0.1, v_{21}=0.04, v_{22}= 0.01, v_{23}=0.001, v_{31}=0.1, v_{32}=0.1, v_{33}=0.2
Glass	v_{11}=0.55, v_{12}=0.01, v_{13}=0.01, v_{21}=0.55, v_{22}=0.022, v_{23}=0.01, v_{31}=0.1, v_{32}=0.001, v_{33}=0.001, v_{51}=0.1, v_{52}=0.2, v_{53}=0.001, v_{61}=0.4, v_{62}=0.3, v_{63}=0.01, v_{71}=0.3, v_{72}=0.1, v_{73}=0.1
OCR	v_{11}= v_{21}= v_{31}= v_{41}= v_{51}= v_{61}= v_{71}= v_{81}= v_{91}=0.6, v_{12}= v_{22}= v_{32}= v_{42}= v_{52}= v_{62}= v_{72}= v_{82}= v_{92}=0.74, v_{13}= v_{23}= v_{33}= v_{43}= v_{53}= v_{63}= v_{73}= v_{83}= v_{93}=0.9

Table 14. Parameter values chosen for one class SVM classifier in case of four partitions using Polynomial Kernel

Dataset	Parameter values
Thyroid	v_{11}=0.002, v_{12}=0.018, v_{13}=0.011, v_{14}=0.01, v_{21}=0.03, v_{22}=0.002, v_{23}=0.002, v_{24}=0.001, v_{31}=5×10^{-4}, v_{32}=1.5×10^{-5}, v_{33}=1.5×10^{-5}, v_{34}=1×10^{-5}
Ionosphere	v_{11}=0.01, v_{12}=0.035, v_{13}=0.01, v_{14}=0.05, v_{21}=0.03, v_{22}=0.03, v_{23}=0.03, v_{24}=0.03
Breast cancer	v_{11}=0.12, v_{12}=0.012, v_{13}=0.014, v_{14}=0.014, v_{21}=0.032, v_{22}=0.025, v_{23}=0.025, v_{24}=0.016
Sonar	v_{11}=0.02, v_{12}=0.01, v_{13}=0.02, v_{14}=0.1, v_{21}=1×10^{-5}, v_{22}=0.001, v_{23}=0.015, v_{24}=0.52
Wine	v_{11}=0.1, v_{12}=0.1, v_{13}=0.1, v_{14}=0.1, v_{21}=0.01, v_{22}=0.01, v_{23}=0.1, v_{24}=0.2, v_{31}=0.1, v_{32}=0.1, v_{33}=0.1, v_{34}=0.3
Glass	v_{11}=0.3, v_{12}=0.001, v_{13}=0.001, v_{14}=0.001, v_{21}=0.23, v_{22}=0.02, v_{23}=0.02, v_{24}=0.02, v_{31}=0.1, v_{32}=0.1, v_{33}=0.01, v_{34}=0.02, v_{51}=0.1, v_{52}=0.1, v_{53}=0.01, v_{54}=0.02, v_{61}=0.01, v_{62}=0.01, v_{63}=0.01, v_{64}=0.2, v_{71}=0.001, v_{72}=0.001, v_{73}=0.001, v_{74}=0.001
OCR	v_{11}= v_{21}= v_{31}= v_{41}= v_{51}= v_{61}= v_{71}= v_{81}= v_{91}= –4, v_{12}= v_{22}= v_{32}= v_{42}= v_{52}= v_{62}= v_{72}= v_{82}= v_{92}= –4, v_{13}= v_{23}= v_{33}= v_{43}= v_{53}= v_{63}= v_{73}= v_{83}= v_{93}= –3, v_{14}= v_{24}= v_{34}= v_{44}= v_{54}= v_{64}= v_{74}= v_{84}= v_{94}= –2

Table 15. Default parameter values chosen by LIBSVM

Parameter	Default value
C	1
γ	1/(number of features)
Degree (d) (for polynomial kernel only)	3
r coef 0 (for polynomial kernel only)	0