MODEL SELECTION FOR MULTI–CLASS SUPPORT VECTOR MACHINES

by

Evrim Itır Karaç

B.A. in Mathematics, Boğaziçi University, 2000

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2005

MODEL SELECTION FOR MULTI–CLASS SUPPORT VECTOR MACHINES

APPROVED BY:

Prof. Levent Akın (Thesis Supervisor)	.....
Prof. Ethem Alpaydın	.....
Assoc. Prof. Taner Bilgiç	.....

DATE OF APPROVAL: 17.06.2005

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor Prof. Levent Akın and Prof. Ethem Alpaydın for their guidance, patience and support throughout this study. Their belief in me made it possible what seemed impossible to me. I thank Assist. Prof. Taner Bilgiç for participating in my thesis jury and his invaluable comments.

I want to thank all my friends in the department, Ali Haydar Özer for his wisdom that enlightens me, Onur Dikmen for the Sommer Song of Itır, Aydın Ulaş for helping me regardless of whatever or whenever, Oya Aran for not leaving me alone in the backgammon games, Berk Gökberk for his blackboard busts, Rabun Koşar for providing a solution for every problem, Atay Özgövde for holding me the all-seeing mirror, Okan İrfanoğlu for listening to me and being there for me, Burak Turhan for the breakfasts at Erguvan, Kemal Kaplan for the Tophane nights and my roommates Arzucan Özgür and Damla Poslu for making the time I spent in the school full of fun. I am also grateful to my instructors and trainers, Arsun Artel and Furkan Kıraç for trying to grant me another degree in MT.

Finally, I am thankful to the interventionist God, for making me feel like I am whole again.

ABSTRACT

MODEL SELECTION FOR MULTI–CLASS SUPPORT VECTOR MACHINES

Support Vector Machines (SVMs) are widely used, powerful learning algorithms based on statistical learning theory. In SVM learning, the data are mapped to a high dimensional space via a non–linear kernel function and a maximal margin hyper–plane separating the positive and negative instances is found in this new space. Choosing the best kernel and generalization to multi–class cases are fundamental problems in SVM learning.

In literature, the best kernel function is chosen by using trial and error. We propose to use a cross–validation based model selection method to find the optimal kernel. Candidate classifiers, with different kernels are trained and hybrid models are built by selecting the best model using the 5×2 cross–validation F test. The performance of the proposed hybrid model is compared to the performances of classical SVMs in one vs. all (OVA) and pairwise classification schemes in terms of accuracy and complexity (as measured by the number of support vectors stored).

The basic two–class support vector machine needs to be extended to handle the multi–class problems and for this purpose, we incorporate our proposed hybrid models in one vs. all , pairwise and error–correcting output code (ECOC) schemes. We see that the proposed hybrid model together with ECOC finds accurate solutions for multi–class problems without significantly increasing complexity.

ÖZET

DESTEK VEKTOR MAKİNALARINDA MODEL SEÇİMİ

Destek Vektör Makinaları (DVM), temeli istatistiksel öğrenme teorisine dayanan, güçlü ve sık kullanılan öğrenme algoritmalarıdır. DVM'ler eğitilirken, data, doğrusal olmayan bir çekirdek fonksiyon kullanılarak yüksek boyutlu bir uzaya taşınır ve pozitif ve negatif örnekleri ayıran büyükçe marjlı bir üstün yüzey bulunur. En iyi çekirdek fonksiyonu seçmek ve ıkıden fazla sınıflı probleme genellemek, DVM'lerdeki temel problemlerdir.

Literatürde, en iyi çekirdek fonksiyon deneme yanılma ile seçilmektedir. Biz en iyi çekireği çapraz–geçerlemeye dayalı model seçimi ile bulmayı öneriyoruz. Aday sınıflandırıcılar değişik çekirdeklerle eğitilir ve 5×2 çapraz–geçerleme F testi kullanılarak en iyi model seçilerek melez modeller oluşturulur. Önerilen melez modelin performansı, bire karşı hepsi ve ikili sınıflandırma yöntemlerini kullanan DVM'lerle karşılaştırılmıştır. Karşılaştırmada doğruluk ve karmaşıklık (saklanan destek vektörlerinin sayısıyla orantılı) kriterleri kullanılmıştır.

İki sınıf problemi için tanımlanmış olan temel destek vektör makinasının ikiden fazla sınıf problemi için genişletilmesi gerekmektedir. Bunun için, önerdiğimiz melez modelleri bire karşı hepsi, ikili sınıflandırma ve hata düzelten çıktı kodları (HDÇK) kullanan yöntemler içinde kullandık. Önerdiğimiz melez modellerin HDKÇ ile kullanımı, çok sınıflı problemlerde karmaşıklığı belirgin bir şekilde artırmadan başarılı sonuçlar verdi.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

$\boldsymbol{x}$	Input vector
r	Target value
X	Training sample
C_i	i^{th} Class
d	Dimensionality of the input space
N	Number of training instances
k	Number of classes
l	Number of binary classifiers
$\boldsymbol{W}$	ECOC matrix
α	Lagrange multipliers
ξ	Slack variable
ρ	Margin
$\cdot$	Dot (Inner) product
$\|.\|$	L_2 Norm
cv	cross validation
ECOC	Error–Correcting Output Code
ERM	Empirical Risk Minimization
KKT	Karush–Kuhn–Tucker
OVA	One vs. All
SLT	Statistical Learning Theory
SRM	Structural Risk Minimization
SVM	Support Vector Machines
VC	Vapnik Chervonenkis

1. INTRODUCTION

In supervised learning problems, we are given a set of N examples $X = \{\boldsymbol{x}^t, r^t\}_{t=1}^N$ where $\boldsymbol{x}^t$ are d dimensional input vectors, called *training instances*, and r^t are the *target values*. In classification problems, targets, r^t, take discrete values from a finite set of classes, and in regression problems, they take real values. The learning task is to find a function which maps training instances to their target values. The class of functions in which this search is conducted is called the *hypothesis space*. Several learning algorithms exist which differ in the hypotheses space from which they choose the target function or in the way the search is conducted [1].

A special case of classification is *binary classification*, where there are only two classes, i.e. $r^t \in \{+1, -1\}$. The class whose instances have $r^t = +1$ is called the *positive class* and the other one is called the *negative class*. To discriminate these classes, a function that separates the input space into two parts such that each part contains instances from only one of the classes is sought. A hyper–plane, $g(\boldsymbol{x})$, in $\Re^n$ is the simplest function that can be used for this purpose. The points lying in the half–space where $g(\boldsymbol{x})$ is greater than zero are assigned to the positive class and the points on the other side are assigned to the negative class. Classifiers using hyper–planes to separate the classes are known as *linear discriminants*.

Support Vector Machines (SVMs) are linear discriminants developed by Vapnik [2]. In SVM learning, the input space is mapped to a high dimensional dot product space via a non–linear transformation. Then in this feature space, the hyper–plane which separates the instances of positive and negative classes with maximum margin is found by solving a quadratic optimization problem [3]. Here *margin*, denoted by ρ, is defined as the distance between the closest positive and negative examples along the direction perpendicular to the hyper–plane. The idea of maximizing the margin is motivated by strong theoretical arguments from Statistical Learning Theory (SLT) and results in good generalization performance. Another important feature of SVMs is that the complexity of the algorithm depends on the number of training instances, and not

on the input dimensionality. These two properties enable SVMs to overcome the curse of dimensionality and high computational complexity which are the disadvantages of working in high dimensional spaces.

One problem with SVM classification is that, they are originally designed for binary classification and its extension to multi–class problems is not straightforward. This problem is common to many classification algorithms and there are two main approaches to deal with it. The first approach is to extend the idea directly to the multi–class problems. This is the most natural and elegant way of extending the algorithm, but usually is more difficult. Another approach is using distributed output codes, in which the multi–class problem is decomposed into multiple binary problems and then the outputs of these classifiers are combined. There are many different methods for this decomposition and combination tasks [4]. The most widely used approaches are *one vs. all* (OVA) and *pairwise* classification schemes. In OVA classification scheme, one binary classifier is trained for each class to separate it from all other classes, and in pairwise classification scheme, one binary classifier is trained for every pair of classes. A more general approach is using *Error–Correcting Output Codes* (ECOCs), in which the rows of an ECOC matrix are used as codewords for the classes and the columns are used to determine the binary classifiers [5]. The classification of an example corresponds to the decoding of a binary string to one of the codewords.

Learning algorithms usually depend on hyper–parameters which control the size of the hypothesis space, and the type of functions in the hypothesis space. Several, so called, *model selection*, techniques exist for optimizing these hyper–parameters. In SVM learning, model selection problem deals with the selection of the mapping that is used to map the input space to a high dimensional space and the parameters of this mapping. Another hyper–parameter of SVM learning is the regularization parameter which controls the trade–off between the bias and the variance of the model [6].

In this thesis we study model selection problem for multi–class SVMs where binary classifiers are combined using one of the schemes described above. We propose a method based on cross–validation for choosing the best model type for each binary

classifier. The performance of this method is compared against other methods in which decision functions of the same type are used. Our performance criteria are generalization accuracy and model complexity, measured by the time/space complexity of the solution model.

The outline of the thesis is as follows: In Chapter 2 an overview of SVM classification will be given, and then in Chapter 3, current methods of extending SVMs to multi–class problems will be mentioned. In Chapter 4 the proposed method for model selection will be explained. The results of the experiments will be presented in Chapter 5. Finally, the conclusions drawn from this work will be given in Chapter 6.

2. SUPPORT VECTOR MACHINES

The elements of learning process are:

- a finite set of training examples produced by an unknown, fixed model,
- a set of functions, the hypothesis space,
- an algorithm which tries to approximate the unknown model by a function from the hypothesis space based on the training set.

Most of the learning algorithms choose the function which minimizes some loss function over the training set. This principle is known as *Empirical Risk Minimization* (ERM). The assumption here is that a function with low error on the training set will have low error on future examples [7]. But low empirical error does not always guarantee a good generalization performance on unseen data. That is why *Statistical Learning Theory* (SLT) studies the conditions under which the empirical error converges to the actual error. It provides a theory which helps us to understand generalization abilities of different class of functions [7].

The ability to learn any training data without errors is known as the *capacity* of a class of functions. Hence the capacity can be thought of as a degree of power, and as it increases the empirical error decreases. The *Vapnik Chervonenkis (VC) dimension* is a measure of the capacity of a class of functions, and is defined as the maximum number, h, of instances $\boldsymbol{x}^1, \boldsymbol{x}^2, ..., \boldsymbol{x}^h$, that can be separated into two classes for all possible 2^h labellings [2]. For example the VC dimension of a line in the plane is equal to three, since for all possible labellings of three points in the plane, a line can separate them, which is not the case for four points. It is shown that by controlling the VC dimension, a smaller bound on the actual error can be set. This can be seen from the following inequality [8]:

$$E_A \leq E_E + \sqrt{\frac{h[log(2N/h) + 1] - log(\eta/4)}{n}} \tag{2.1}$$

where η, $0 \leq \eta \leq 1$, is a chosen real number that determines the confidence level, E_A is the actual error and E_E is the empirical error. The second term on the right hand side in Equation 2.1 is called the *VC Confidence*. According to Equation 2.1, choosing the model which has a low training error together with a small VC confidence, will give the lowest upper bound on the actual risk. Since VC confidence is proportional to VC dimension, h, it can be kept small by keeping h small. For linear discriminants obtained by the SVM algorithm, the following bound holds for the VC dimension:

$$h \leq \frac{R^2}{\rho^2}, \tag{2.2}$$

where R is the radius of the smallest sphere containing the training data and ρ is the margin, i.e. the distance between the hyper–plane and the closest training vector [2]. Equation 2.2 shows that by maximizing the margin, the VC dimension of a classifier can be minimized, which results a smaller upper bound on the generalization error. Since the SVM algorithm finds the hyper–plane which separates the data with the maximum margin, the generalization error of the solution is minimum. Such a hyper–plane is called the *Maximal Margin Classifier*. In SVM learning, the problem of finding the maximal margin classifier is reduced to an optimization problem. The formulation of this optimization problem is given in Section 2.1.

The primary appeal of SVM is that it can be simply and elegantly applied to nonlinear discrimination. With only minor changes SVM methods can construct a wide class of two–class nonlinear discriminants by solving a single quadratic programming problem. The basic idea is to map the input nonlinearly to a higher dimensional space. Then in this space a linear discriminant which is nonlinear in the original space is constructed [9]. By using kernel functions, SVMs can effectively and efficiently construct many types of nonlinear discriminants including polynomials and radial basis functions. The extention of SVM algorithm to non–linear decision surfaces will be explained in Section 2.3.

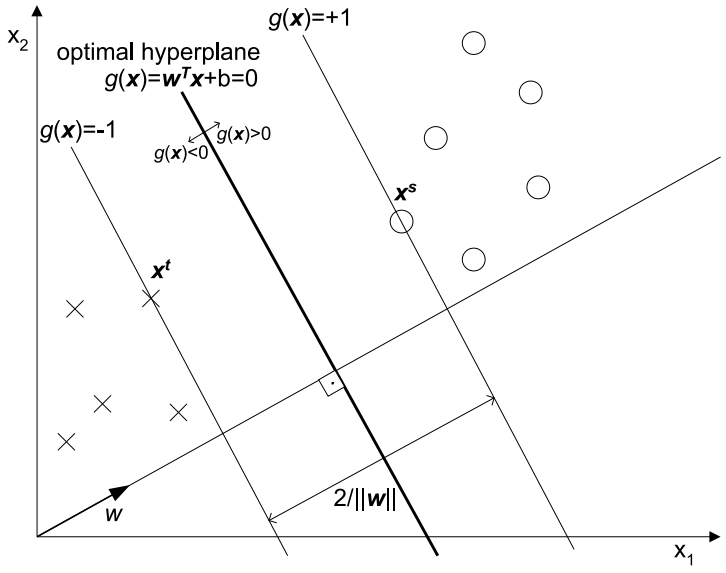


Figure 2.1. Maximal margin classifier

2.1. Maximal Margin Classifiers

This is the simplest model of SVM which is applicable only when the data are linearly separable. Let $X = \{\boldsymbol{x}^t, r^t\}_{t=1}^N$ be the training data where $r^t = +1$ for positive instances, and $r^t = -1$ for negative instances. If the data are linearly separable, then there exists a hyper–plane such that the positive instances lie on one side of the hyper–plane and the negative instances lie on the other side. The data in Figure 2.1 are separated by the hyper–plane $g(\boldsymbol{x}) = \boldsymbol{w}^T \cdot \boldsymbol{x} + b = 0$ where $\boldsymbol{w}$ is the normal vector and b is the threshold. This condition can be formulated as:

$$\begin{aligned} \boldsymbol{w}^T \cdot \boldsymbol{x}^t + b \geq 0, \quad \text{if } r^t = +1, \\ \boldsymbol{w}^T \cdot \boldsymbol{x}^t + b \leq 0, \quad \text{if } r^t = -1 \end{aligned}$$

The parametrization of the hyper–plane is unique up to the multiples of $\boldsymbol{w}$ and b, which means there are infinitely many $(\boldsymbol{w}, b)$ pairs satisfying these inequalities. To overcome this problem the hyper–plane is represented in the canonical form in which the instances closest to the hyper–plane has a distance of $1/\|\boldsymbol{w}\|$ [10]. Then the margin

becomes $\rho = 2/\|\boldsymbol{w}\|$ and maximizing the margin corresponds to minimizing $\|\boldsymbol{w}\|$.

$$\begin{aligned} \boldsymbol{w}^T \cdot \boldsymbol{x}^t + b \geq +1, \quad \text{if } r^t = +1, \\ \boldsymbol{w}^T \cdot \boldsymbol{x}^t + b \leq -1, \quad \text{if } r^t = -1 \end{aligned}$$

These inequalities can be written in a more compact form by taking r^t's into account.

$$r^t(\boldsymbol{w}^T \cdot \boldsymbol{x}^t - b) \geq 1, \quad \text{for } t = 1, ..., N \tag{2.3}$$

The problem of finding the maximal margin hyper–plane which satisfies constraints in Equation 2.3 can be reduced to the following optimization problem:

$$\begin{aligned} \text{minimize} \qquad & Z(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2, \\ \text{subject to} \qquad & r^t(\boldsymbol{w}^T \cdot \boldsymbol{x}^t + b) \geq 1, \quad \text{for } t = 1, ..., N \end{aligned} \tag{2.4}$$

This is a convex quadratic optimization problem since both the quadratic objective function $Z(\boldsymbol{w})$ and the linear constraints are convex [1]. The solution to this optimization problem is given at the saddle point of the Lagrange functional:

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{t=1}^{N} \alpha^t [r^t(\boldsymbol{w}^T \cdot \boldsymbol{x}^t - b) - 1] \tag{2.5}$$

where α^t are Lagrange multipliers. The Lagrangian has to be minimized with respect to the primal variables $\boldsymbol{w}$ and b, and maximized with respect to $\boldsymbol{\alpha}$ [8]. Consequently, the partial derivatives of Lagrangian with respect to $\boldsymbol{w}$ and b have to vanish.

$$\frac{\partial L(\boldsymbol{w}, b, \boldsymbol{\alpha})}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{t=1}^{N} r^t \alpha^t \boldsymbol{x}^t = \boldsymbol{0}, \tag{2.6}$$

$$\frac{\partial L(\boldsymbol{w}, b, \boldsymbol{\alpha})}{\partial b} = \sum_{t=1}^{N} r^t \alpha^t = 0 \tag{2.7}$$

Equations 2.6 and 2.7 show that $\boldsymbol{w}$ can be written as a linear combination of the training instances,

$$\boldsymbol{w} = \sum_{t=1}^{N} r^t \alpha^t \boldsymbol{x}^t, \tag{2.8}$$

and the coefficients α^t's have to satisfy

$$\sum_{t=1}^{N} r^t \alpha^t = 0, \quad \alpha^t \geq 0, \quad \text{for } t = 1, ..., N \tag{2.9}$$

The corresponding dual is found by substituting Equations 2.8 and 2.9 into the Lagrangian given in Equation 2.5:

$$W(\boldsymbol{\alpha}) = \sum_{t=1}^{N} \alpha^t - \frac{1}{2} \sum_{t=1}^{N} \sum_{s=1}^{N} \alpha^t \alpha^s r^t r^s \boldsymbol{x}^t \cdot \boldsymbol{x}^s \tag{2.10}$$

In the dual $\boldsymbol{w}$ and b are eliminated and $W(\boldsymbol{\alpha})$ has to be maximized only with respect to α^t's, subject to the equality and non–negativity constraints in Equation 2.9, which can be done using standard optimization procedures. Once α^t are obtained, $\boldsymbol{w}$ can be found using Equation 2.8. Although the summation in Equation 2.8 is over all of the training instances, it turns out that most of the α^t are equal to zero at optimality. This follows from the Karush–Kuhn–Tucker (KKT) conditions, which state that only the Lagrange multipliers that are non–zero at the saddle point correspond to constraints which are tight. Hence at optimality, the multiplication of Lagrange multipliers and corresponding constraints is equal to zero [11].

$$\alpha^t [r^t (\boldsymbol{w}^T \cdot \boldsymbol{x}^t - b) - 1] = 0, \quad \text{for } t = 1, ..., N \tag{2.11}$$

The instances $\boldsymbol{x}^t$ for which $\alpha^t > 0$ are called *support vectors* [2]. These instances lie exactly on the margin and we see from Equation 2.8 that the discriminant is written in terms of these support vectors. In Figure 2.3, $\boldsymbol{x}^t$ and $\boldsymbol{x}^s$ are support vectors. For all the other instances, α^t are zero and they do not appear in the expansion of $\boldsymbol{w}$.

The threshold b can be found by solving Equation 2.3 for the support vectors and then taking the average. Then the maximal margin hyper–plane is given by

$$g(\boldsymbol{x}) = \sum_{t=1}^{N} r^t \alpha^t \boldsymbol{x}^t \cdot \boldsymbol{x} + b, \tag{2.12}$$

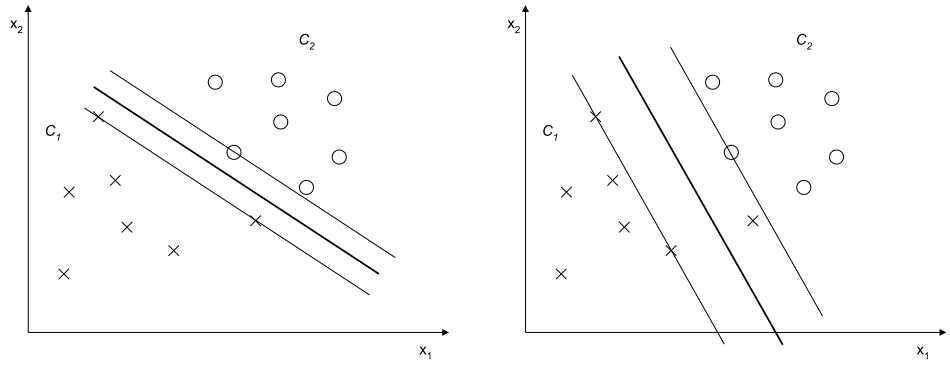
and the decision function is

$$g(\boldsymbol{x}) = sign(\sum_{t=1}^{N} r^t \alpha^t \boldsymbol{x}^t \cdot \boldsymbol{x} + b) \tag{2.13}$$

Any convex optimization problem has the property that every local solution is also global, which ensures that there is no local minima. But although the solution $\boldsymbol{w}$ is unique, the coefficients α^t need not to be, which means that there may be different set of support vectors for the same $\boldsymbol{w}$. In such a case, the solution with the least number of support vectors should be chosen since it requires minimum storage (space complexity) and fewer number of calculations in test phase (time complexity).

2.2. Soft Margin Classifiers

Real life problems are usually not as simple as the example in Figure 2.1 and the linear–separability assumption is not valid. If the data are not linearly separable, the optimization problem given in Equation 2.4 has no feasible solution. Even if the training data are linearly separable, one may not desire to find a hyper–plane which separates the data perfectly if the data are noisy. In such a case, the concept known as over–fitting occurs and the classifier found does not generalize well over the unseen data. An example of over–fitting is shown in Figure 2.2. The classifier on the right hand side has a larger margin and will generalize better than the one on the left, although its training error is larger. The concept of a maximal margin hyper–plane concept presented in Section 2.1 can be extended by relaxing the constraints given in Equation 2.3 and allowing misclassification of training instances [11] by introducing slack variables ξ^t for each training instance, measuring the deviation from the margin. An example of soft margin classifier is shown in Figure 2.3.



(a) (b)

Figure 2.2. Examples of two maximal margin classifiers with different margins. (b) has a larger margin and is preferred to (a)

The relaxed separation constraints are

$$r^t(\boldsymbol{w}^T \cdot \boldsymbol{x}^t - b) \geq 1 - \xi^t, \quad \text{for } t = 1, ..., N \tag{2.14}$$

In order to prevent ξ^t's from getting arbitrarily large, they are added to the objective function as a penalty. Hence the *soft margin hyper–plane* can be found by solving the following optimization problem.

$$\begin{aligned} \text{minimize} \quad & Z(\boldsymbol{w}, \xi) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{t=1}^{N} \xi^t, \\ \text{subject to} \quad & r^t(\boldsymbol{w}^T \cdot \boldsymbol{x}^t + b) \geq 1 - \xi^t, \quad \text{for } t = 1, ..., N \end{aligned} \tag{2.15}$$

where C is the regularization parameter which controls the trade–off between margin maximization and training error minimization. This parameter will be discussed in Chapter 4 in more detail.

According to Equation 2.14, a training instance $\boldsymbol{x}^t$ is misclassified only if the corresponding slack variable takes a value greater than one. In Figure 2.3 the training instance $\boldsymbol{x}^t$ is misclassified. The situation of $\boldsymbol{x}^t$ for different values of ξ is summarized in Figure 2.4.

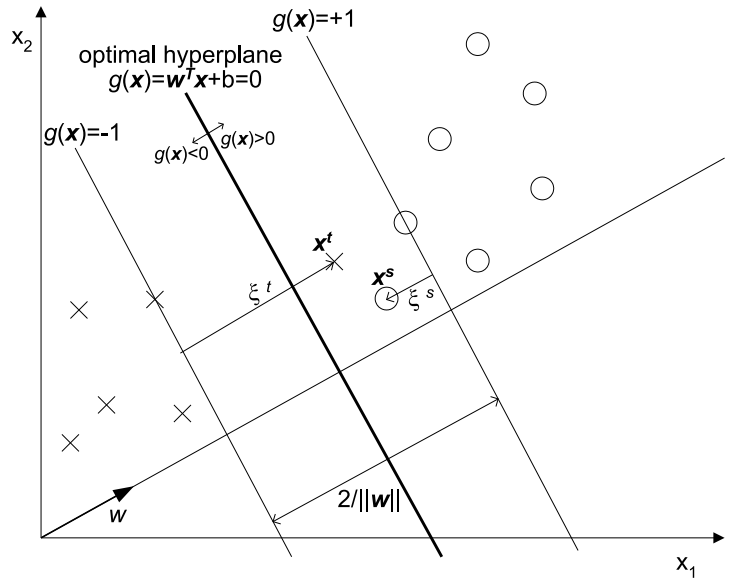


Figure 2.3. Soft margin classifier

$\xi^t < 0$	if $\boldsymbol{x}^t$ is classified correctly and lies outside the margin,
$\xi^t = 0$	if $\boldsymbol{x}^t$ is classified correctly and lies on the margin,
$0 < \xi^t < 1$	if $\boldsymbol{x}^t$ is classified correctly and lies within the margin,
$\xi^t > 1$	if $\boldsymbol{x}^t$ is misclassified

Figure 2.4. The situation of $\boldsymbol{x}^t$ for different values of ξ

The solution technique of the optimization problem in Equation 2.15 is almost equivalent to the technique explained in Section 2.1 and the form of the decision function is the same as in the maximal margin classifier.

2.3. Nonlinear Support Vector Machines

Both maximal margin classifiers and soft margin classifiers use linear decision surfaces to separate the data. To generalize SVMs to nonlinear decision surfaces, the data are first mapped to another vector space via a non–linear transformation, $\phi(\boldsymbol{x})$, and then the maximal margin or the soft margin hyper–plane is found using the original SVM algorithm in that new vector space. This new vector space is called the *feature space* and usually its dimensionality is much higher than the original input space. This

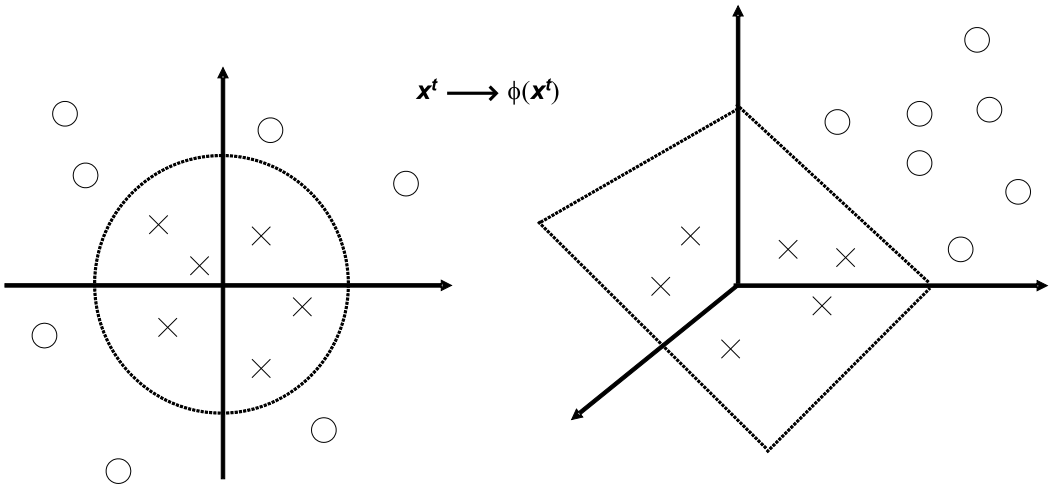


Figure 2.5. Mapping input space to a high dimensional space via a non–linear transformation

increases the separability of the data. This situation is illustrated in Figure 2.5. To find the decision surface in the feature space $\phi(X) = \{\phi(\boldsymbol{x}^t), r^t\}_{t=1}^N$ is used as the training set and the decision function is

$$g(\boldsymbol{x}) = sign\left[\sum_{t=1}^{N} r^t \alpha^t \phi(\boldsymbol{x}^t) \cdot \phi(\boldsymbol{x}) + b\right] \tag{2.16}$$

An important property of $g(\boldsymbol{x})$ is that it contains $\phi(\boldsymbol{x})$ and $\phi(\boldsymbol{x}^t)$ only in inner products. Therefore, the individual $\phi(\boldsymbol{x}^t)$ need not to be known for training an SVM; only their inner products are important. By computing $K(\boldsymbol{x}^t, \boldsymbol{x}) = \phi(\boldsymbol{x}^t) \cdot \phi(\boldsymbol{x})$ and replacing $\boldsymbol{x} \cdot \boldsymbol{x}^t$ by $K(\boldsymbol{x}^t, \boldsymbol{x})$ everywhere in the original algorithm, the maximal or soft margin hyper–plane in the feature space can be found without explicitly doing the transformation. The function $K(\boldsymbol{x}^t, \boldsymbol{x})$ is called a *kernel function*. Whether a given function corresponds to an inner product in some vector space can be checked by using Mercer's conditions [2]. This enables us to construct kernel functions directly without knowing the actual transformation $\phi(\boldsymbol{x})$.

For different classification tasks, different kernel functions may be used. To con-

struct polynomial decision surfaces of degree p, the following kernel function is used;

$$K(\boldsymbol{x}^t, \boldsymbol{x}) = [\boldsymbol{x}^t \cdot \boldsymbol{x} + 1]^p \tag{2.17}$$

When $\boldsymbol{x} \in \Re^2$ and a polynomial kernel of degree p is used, the dimension of the corresponding feature space is C_p^{d+p-1}. This number can get very large very quickly. But the complexity of the SVM algorithm depends on the number of training points and not on the dimensionality of the feature space. This is why SVMs can work in very high dimensional spaces efficiently without overfitting. Another type of frequently used kernel functions is the Gaussian kernel

$$K(\boldsymbol{x}^t, \boldsymbol{x}) = \exp\left[-\frac{\|\boldsymbol{x}^t - \boldsymbol{x}\|^2}{\sigma^2}\right], \tag{2.18}$$

where σ is the width of the Gaussian.

The flexibility that the kernel functions provide comes at a cost. The appropriate kernel type and kernel parameter has to be selected. This is usually done by comparing performances of classifiers using different types of kernels with different parameters on a separate validation set, and choosing the set of parameters which give the best result. This problem, known as model selection will be discussed in Chapter 4 in more detail.

3. MULTI–CLASS SUPPORT VECTOR MACHINES

The most basic and simple case of classification is binary classification where there are two classes and many classification algorithms are designed originally for this type of problems. If possible, they can also be generalized for the multi–class case. For some algorithms, like the $C4.5$ algorithm for training decision trees, this generalization can be done easily, since the concept used generalizes well to the multi–class case. But for most of the classification algorithms, including SVMs, this generalization is not trivial. For such algorithms, a multi–class problem is first decomposed into several binary classification problems and then a binary *base classifier* is trained for each subproblem. During testing, the example is given to all of the binary classifiers and their outputs are combined. Different methods have been proposed for the decomposition of a multi–class problem and the combination of the outputs of the base classifiers as described below.

3.1. Distributed Output Codes

3.1.1. One vs. All Classification

One vs. all (OVA) classification scheme is the simplest multi–class classification scheme where, for each class, a different binary classifier is trained to distinguish instances of that class from the instances of all other classes. For example, in Figure 3.1, the classifier given as $g_1(\boldsymbol{x})$ separates class C_1 from C_2 and C_3. In general, if there are k classes, k classifiers are trained. The size of each binary problem is equal to the size of the original problem, since the binary classifiers use training instances belonging to one class as positive examples and all the other instances as negative examples. Therefore, this approach is not very efficient when the base classifiers are SVMs, since the training of an SVM requires the solution of a quadratic optimization problem which has as many variables as there are training instances.

To classify a test example, all of the binary classifiers are run. If the test example

belongs to class C_i then the output of the classifier $g_i(\boldsymbol{x})$ is expected to be $+1$ and the outputs of all the other classifiers are expected to be -1. If this is the case then the test example is assigned to class C_i. But due to classification errors, more than one or none of the classifiers may output $+1$. In this case the test example can not be assigned to a class. In Figure 3.1, the shaded area is part of the unclassified region. To overcome this problem, instead of using indicator functions, real–valued functions are used as binary classifiers. In case of SVMs, using Equation 2.12, a test example is assigned to the class, for which the output of the corresponding binary classifier is the highest.

Although very simple and obvious, OVA is extremely powerful, producing results that are as accurate as more complex methods [12].

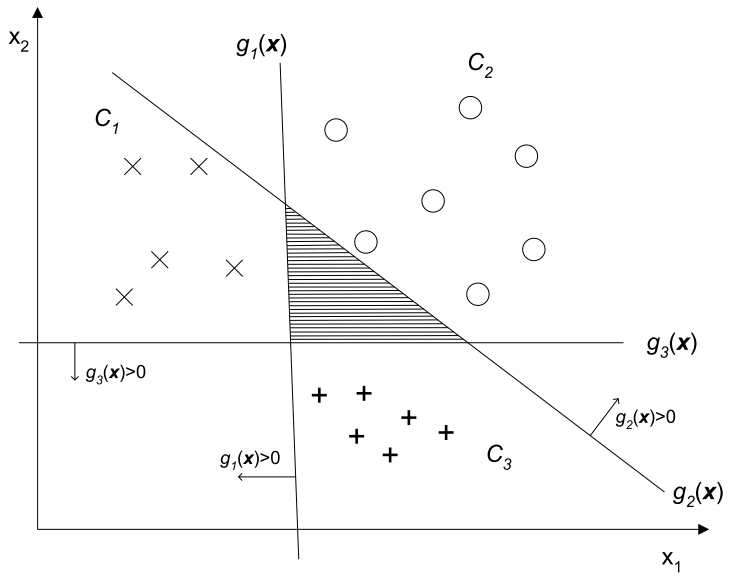


Figure 3.1. Example of OVA classification

3.1.2. Pairwise Classification

In pairwise classification, a binary classifier is trained for each pair of classes. For example in Figure 3.2, the classifier labeled as $g_{ij}(\boldsymbol{x})$ is separating C_i from C_j. For a k-class problem, $k(k-1)/2$ binary classifiers have to be trained. This number is larger than OVA classification which suggests an increase in storage requirement and training

times. But the sizes of individual binary problems are smaller than the original problem since they use only two of the classes and therefore, the total training times remain almost the same [13]. Because two–class problems are simpler, the number of support vectors per problem is also less and the total storage also remain approximately the same.

In the test phase, an example is given to all of the binary classifiers and the classification is done by voting. If an example is assigned to class C_i by classifier g_{ij}, then the vote for the i^{th} class is increased by one, otherwise the vote for the j^{th} class is increased by one. Then the example is assigned to the class with the maximum number of votes. When two classes have the same number of votes, the example cannot be classified; assigning the example to the class with smaller index, or to the class whose prior probability is higher are widely used ways of breaking ties [14].

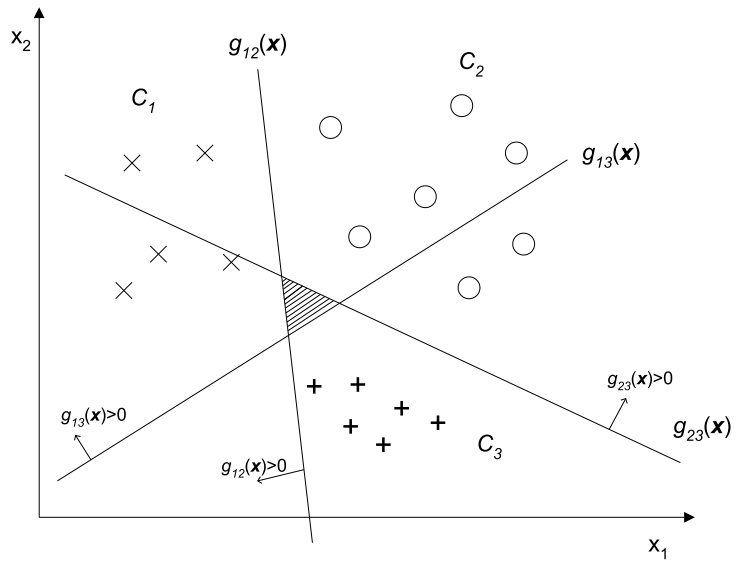


Figure 3.2. Example of pairwise classification

3.1.3. Error–Correcting Output Code Approaches

Error–Correcting Output Codes (ECOC) are originally proposed to increase the reliability of noisy communication channels over which binary signals are transmitted. The purpose of the code is to detect, and if possible, correct transmission errors [15].

This information theoretic method is extended to the framework of decomposition methods for multi–class classification problems to improve the generalization performance.

In ECOC scheme, a k by l matrix, $\boldsymbol{W}$, is constructed whose entries take values from the set $\{-1, +1\}$. Each row of this matrix, called a *codeword*, corresponds to a class. Then, F binary classifiers are trained, one for each bit position in these codewords. [5]. If $\boldsymbol{W}_{ij} = +1$, the examples of the i^{th} class are used as positive examples in the training of the j^{th} binary classifier, and as negative examples if $\boldsymbol{W}_{ij} = -1$. In this way, binary classification problems into which the multi–class problem is going to be decomposed are determined according to the columns of the ECOC matrix $\boldsymbol{W}$. During testing an example is given to all of these F binary classifiers, then the outputs of these binary classifiers are combined to obtain a binary string of length F. This output string is compared to each of the k codewords, and the new example is assigned to the class whose codeword is closest, according to some distance measure. A frequently used distance measure is the *Hamming distance*, which counts the number of bits that differ in two binary stings [16]. The process of mapping the output string to the nearest codeword is identical to the decoding step for ECOCs.

An example of an ECOC matrix for a ten–class problem is given in Table 3.1. For this ECOC matrix, the codeword for the first class is $(-1\ -1\ -1\ \ 1\ -1\ -1)$, and the first classifier is trained using the examples of the second, the fifth and the sixth classes as positive examples and the examples of all the other classes as negative examples. To classify a new example, all the six binary classifiers are evaluated to obtain a string of length six, such as $(1\ 1\ -1\ -1\ -1\ 1)$. Then the distance of this string to each of ten codewords is computed. This new example is assigned to the fifth class because the codeword of this class, $(1\ \ 1\ -1\ -1\ -1\ -1)$, is the nearest codeword [5].

A measure of quality of an error–correcting code is the minimum Hamming distance between any pair of codewords. This is because a single error done by one of the binary classifiers moves the output string one unit away from the true codeword. Hence, if the minimum Hamming distance between the pair of codewords is m, then

the code can correct up to $\lfloor \frac{m+1}{2} \rfloor$. This is known as *row separability*.

Another important concept when constructing ECOC matrices is the *column separability*. This is a result of the fact that the ECOC approach heavily relies on the errors produced by different binary classifiers being decorrelated. If there are two copies of the same binary classifier, then these two classifiers will make mistakes simultaneously, which will make it harder to detect and correct the error. Therefore, it is desired to have well separated columns as well as rows, which can be achieved by maximizing the minimum Hamming distance among the columns of the ECOC matrix. The Hamming distance between two binary strings is maximum when they are complements of each other. But in this case the classifiers corresponding to these columns will learn the same problem, just with the labels of the classes switched and again they will make the same mistakes. Therefore, for column separability, the minimum Hamming distance among the columns and their complements should be maximized. Existing algorithms and the proposed method for constructing ECOC matrices will be mentioned in Section 3.1.3.1 and Section 3.1.3.2.

Table 3.1. Example of an ECOC matrix for a ten–class problem

	CodeWords					
Classes	1	2	3	4	5	6
1	-1	-1	-1	1	-1	-1
2	1	-1	-1	-1	-1	-1
3	-1	1	1	-1	1	-1
4	-1	-1	-1	-1	1	-1
5	1	1	-1	-1	-1	-1
6	1	1	-1	-1	1	-1
7	-1	-1	1	1	-1	1
8	-1	-1	1	-1	-1	-1
9	-1	-1	-1	1	-1	-1
10	-1	-1	1	1	-1	-1

The OVA classification scheme and the pairwise classification scheme can be handled in the ECOC framework by choosing the ECOC matrices appropriately. The ECOC matrix for the OVA classification scheme is given in Table 3.2. The first column of the ECOC matrix, corresponds to the classifier, denoted by g_1, which separates the first class from the rest of the classes. To represent pairwise classification scheme, the ECOC approach needs to be extended by allowing zero entries in the ECOC matrix. An example of an ECOC matrix corresponding to the pairwise classification of a four–class problem is given in Table 3.3. In the example, the binary classifier denoted by g_{ij} separates the i^{th} class from the j^{th} class. $\boldsymbol{W}_{ij} = 0$ indicates that the examples of the i^{th} class are not used in the training of the j^{th} binary classifier. For example, in the training of the classifier g_{12}, the examples of the first class are used as positive examples and the examples of the second class are used as negative examples, the examples belonging to the rest of the classes are not used.

Table 3.2. ECOC matrix of the OVA classification scheme for a four–class problem

	Classifiers			
Classes	g_1	g_2	g_3	g_4
1	1	-1	-1	-1
2	-1	1	-1	-1
3	-1	-1	1	-1
4	-1	-1	-1	1

Table 3.3. ECOC matrix of the pairwise scheme for a four-class problem

	Classifiers					
Classes	g_{12}	g_{13}	g_{14}	g_{23}	g_{24}	g_{34}
1	1	1	1	0	0	0
2	-1	0	0	1	1	0
3	0	-1	0	-1	0	1
4	0	0	-1	0	-1	-1

3.1.3.1. Algorithms for Constructing ECOCs. Several algorithms for constructing good ECOC matrices for multi–class classification problems have been proposed in [5, 17, 18].

Usually different methods are used depending on the number of classes.

For a k-class problem, the number of possible columns is 3^k, since each one of the k entries can take a value from the set $\{-1, 0, +1\}$. But some of these columns do not correspond to binary classification problems; e.g. all zero or all one columns. In fact, all columns which do not contain at least one $+1$ and one -1 are useless. Therefore, the number of effective columns is $\frac{1}{2}(3^n - 2^{(n+1)} + 1)$. The factor $\frac{1}{2}$ is a result of not using the columns which are complements of each other. The ECOC matrix which contains all the $\frac{1}{2}(3^n - 2^{(n+1)} + 1)$ columns is called the *full code*. The size of the full code for different number of classes is given in Table 3.4. It can be seen that using the full code is not feasible even for a five–class problem. Therefore, a good subset of the full code has to be chosen. But this problem is a difficult problem too. In fact, it was shown that given a set of binary classifiers, the problem of finding a good output code is NP–complete [18].

Table 3.4. Size of the full code for different number of classes

# of Classes (k)	# of columns in the full code
2	2
3	6
4	25
5	90
6	301
7	966
8	3025
9	9330
10	28501

The first algorithm proposed in [5] formulates the problem of selecting a good subset of the full code as a propositional satisfiability problem by representing each column in the full code by a boolean variable. A solution is required to include exactly

l columns (the desired number of columns) while ensuring that the minimum Hamming distance between any pair of codewords is greater than by a chosen value.

Secondly, a random search algorithm is proposed. The algorithm starts with a random ECOC matrix and iteratively tries to improve it. To improve the code matrix, the pair of rows closest to each other and the pair of columns which have the largest Hamming distance are found. Then the four codeword bits where these rows and columns intersect are changed to improve row and column separability. When a local maximum is reached, two rows and two columns are chosen randomly and the algorithm tries to improve their separation. It was shown that this procedure is able to improve the minimum Hamming distance separation quite substantially.

3.1.3.2. Proposed Method for Constructing ECOCs. The proposed algorithm is an iterative algorithm which starts with an empty ECOC matrix and at each step selects the unused column in the full code which gives the best ECOC matrix according to some criteria. The main difference of this algorithm from the algorithms explained in the previous section is the criteria used in the selection of ECOC matrices. Instead of maximizing the minimum Hamming distance between each pair of codewords, it tries to maximize the sum of the total Hamming distance between the rows and the total Hamming distance between the columns. The formula of this evaluation function is given in Equation 3.1.

$$THD(\boldsymbol{W}) = \sum_{i=1}^{k}\sum_{j=1}^{k} HamDist(\boldsymbol{W}_i, \boldsymbol{W}_j) + \sum_{i=1}^{l}\sum_{j=1}^{l} HamDist(\boldsymbol{W}_i^T, \boldsymbol{W}_j^T), \qquad (3.1)$$

where $\boldsymbol{W}_i$ denotes the i^{th} row and $\boldsymbol{W}_i^T$ denotes the i^{th} column in the ECOC matrix.

When the number of columns is much larger than the number of rows, the second term in Equation 3.1 surpasses the first term and becomes more significant in the selection. To treat rows and columns equally, the first and second terms are divided by the number of classes and the number of columns respectively. In this way, the

normalized evaluation function in Equation 3.2 is obtained.

$$NHD(\boldsymbol{W}) = \frac{1}{k}\sum_{i=1}^{k}\sum_{j=1}^{k} HamDist(\boldsymbol{W}_i, \boldsymbol{W}_j) + \frac{1}{l}\sum_{i=1}^{l}\sum_{j=1}^{l} HamDist(\boldsymbol{W}_i^T, \boldsymbol{W}_j^T) \quad (3.2)$$

The pseudocode of the proposed algorithm is given in Figure 4.4.

```
Construct the full ECOC matrix, F
Initialize W to empty matrix
Add the first column of F to W
for i=2 to l
    for each unused column of F, f
        add f to W and calculate total column distance
        add f^T to W and calculate total column distance
        select among f and f^T, the one whose total column distance is larger
        calculate total Hamming distance of this candidate column
    select the candidate, f* which has the maximum total Hamming distance
    add f* to W
```

Figure 3.3. The pseudocode of the algorithm

3.2. Direct Methods

Direct methods are similar to OVA scheme in the sense that they find k functions such that each of them separates one class from the rest of the classes. But instead of solving one optimization problem for each one of the k classes, they solve a single optimization problem to find k functions simultaneously. Therefore, they are also known as single machine approaches.

In the OVA scheme, if an example, $\boldsymbol{x}^t$, belonging to class C_i is misclassified or lies within the margin , then the corresponding slack variable ξ^t takes a value greater than 1 and it is added to the objective function in Equation 2.4 as a penalty. In Figure 3.4, $\boldsymbol{x}^t$

is misclassified by the binary classifier $g_2(\boldsymbol{x})$. But this example will still be assigned to the class C_2 in the OVA scheme, since the closest classifier $g_2(\boldsymbol{x})$ will give the highest output (the distance to $\boldsymbol{w}_2$ is smallest). The single machine approach proposed by Vapnik [7] and Watkins [19] follow the idea that instead of paying a penalty for each machine separately based on whether each machine classifies an example correctly, a penalty is paid based on the relative values output by different machines.

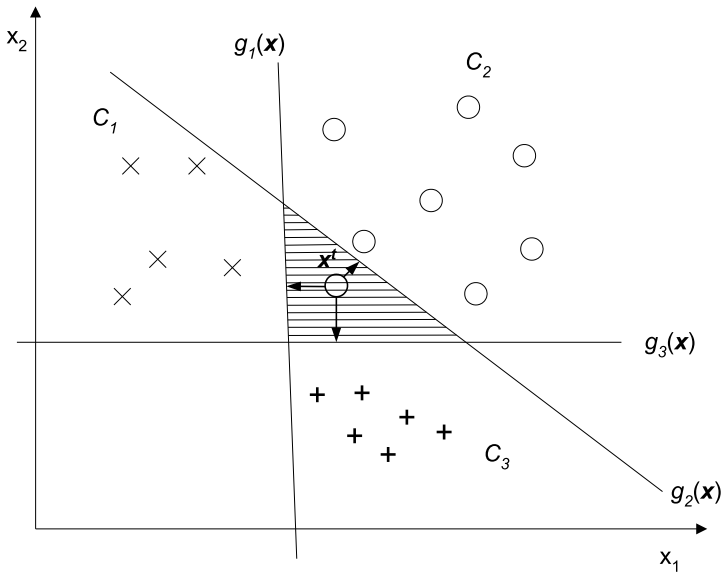


Figure 3.4. Single machine approach

These methods are generally complicated to implement and slow to train, while giving no better performance than a simple OVA Scheme [12].

4. MODEL SELECTION

The model selection problem in SVM classification corresponds to the problem of choosing a kernel function, the values of the kernel parameter, and the regularization parameter C, so as to minimize the expectation of the test error [6]. The type of the kernel function determines the feature space into which the training data is going to be mapped and the parameter C controls the trade–off between margin maximization and training–error minimization.

4.1. Choosing the Regularization Parameter

The regularization parameter C controls the trade–off between the complexity of the decision function and the number of training examples misclassified. As C increases, the number of training errors will decrease. In the algorithm, increasing C will increase the value of the penalty term in Equation 2.4 and to minimize the objective function a smaller margin will be chosen. After some point, the solution with the minimum number of training examples will be found and further increases in C will have no effect. This situation will usually result in a very small margin. Besides, the training time increases drastically for large values of C.

4.2. Choosing the Kernel Function

The kernel function chosen determines the type of the decision surface, hence has a very important effect on the performance. As mentioned in Section 2.3, the more frequently used kernel functions are the polynomial kernels and Gaussian kernels. After choosing a kernel function, a value for the kernel parameter has to be set. In the case of polynomial kernels, the parameter is the degree of the polynomial, whereas in the Gaussian kernels it is the width of the Gaussian function.

Although the Gaussian kernel is powerful enough to represent decision surfaces which can be generated using polynomial kernels, using an appropriate Gaussian is a

more difficult problem. For this reason, typically the starting point for model selection is the polynomial kernels where one starts with the simplest polynomial and the degree of the polynomial is increased until a satisfactory result is obtained.

To demonstrate the effect of the kernel function on the decision boundary produced, an artificial data set is created and SVMs using different type of kernels are trained on this data set. In Figure 4.1, the linear decision boundary produced by a standard SVM is given. In Figures 4.2 and 4.3, quadratic and cubic decision functions are found as a result of using polynomial kernels of degree two and three respectively. It can be seen that when using polynomial kernels, as the degree increases, the number of support vectors decreases. This is a result of the fact that higher degree polynomial kernels map the data to a higher dimensional feature space where separation of the data is better [8]. Hence, in that high–dimensional feature space a linear decision boundary with a small number of support vectors can be found.

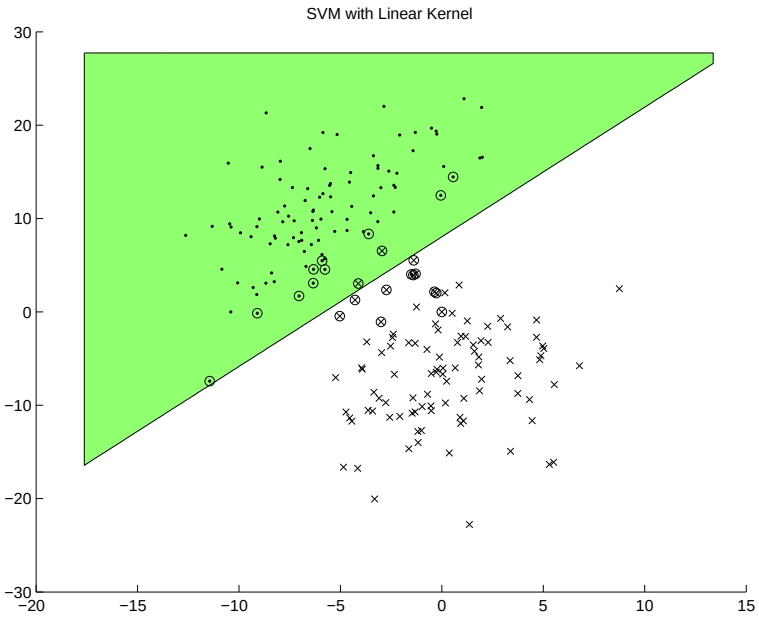


Figure 4.1. Example of a SVM with linear kernel

4.3. Proposed Method for Model Selection

In this study, we propose a model selection method, *MM–SVM* based on the 5×2 cv F test [1]. For a given binary classification problem, first, linear SVMs using

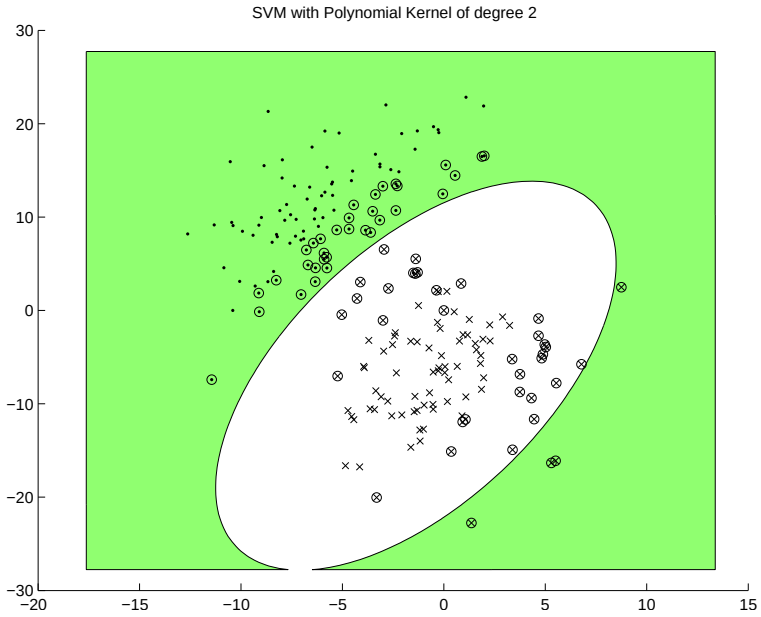


Figure 4.2. Example of a SVM with polynomial kernel of degree two

polynomial kernels of degrees two and three are trained. Next, each pair of decision functions is compared using the F test. If one of them has a higher accuracy on the validation set then it is preferred. If there is no significant difference then the simpler model is chosen.

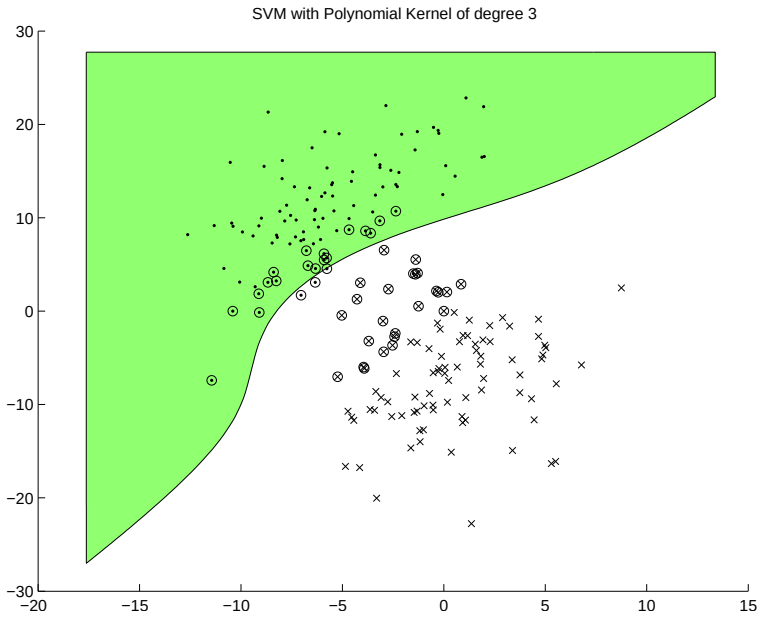


Figure 4.3. Example of a SVM with polynomial kernel of degree three

for each binary problem

find the best C value, C_{lin}, for linear kernels
find the best C value, C_{P2}, for second degree polynomial kernels
find the best C value, C_{P3}, for third degree polynomial kernels

train SVMs with linear kernel using C_{lin}
train SVMs with second degree polynomial kernel using C_{P2}
train SVMs with third degree polynomial kernel using C_{P3}

choose the best kernel type using 5×2 cv F test

Figure 4.4. The pseudocode of the MM-SVM algorithm

5. EXPERIMENTS AND RESULTS

We used the SVM–light [20] software tool to train the binary SVM classifiers. This was converted to a dynamically linked library and was used in Matlab 7.0.

5.1. Datasets

The datasets used in the experiments are obtained from UCI Repository of Machine Learning [21]. The number of attributes, classes and instances for each data set is given in Table 5.1. In the following sections the 3-letter abbreviations, given next to the dataset names, will be used.

Table 5.1. Datasets used for the experiments

Name	Number of attributes	Number of classes	Number of instances
Car (CAR)	6	4	1728
Dermatology (DRM)	34	6	366
Iris (IRS)	4	3	150
Optdigits (OPT)	64	10	3823
Pendigits (PEN)	16	10	7494
Pima (PIM)	8	2	768
Segment (SGM)	19	7	2100
Wine (WIN)	13	3	178
Yeast (YST)	8	10	1484
Zoo (ZOO)	16	7	101

The error rates of different learning algorithms on UCI datasets are presented in [22]. A summary of these results is given in Table 5.2. In the NMC algorithm, an example is assigned to the class with the closest mean vector. $C4.5$ is the archetypal decision tree method and LGC is Logistic Discrimination. A difference in the error rates

of Linear Discriminant Analysis (LDA) method and Nearest Neighbor (NN) algorithm for a dataset suggests that the data is not linearly separable. We focused on such datasets to be able to demonstrate the difference of the results for linear and nonlinear decision surfaces.

Table 5.2. Error rates of five classification algorithms

Dataset	NMC	LGC	C4.5	NN	LDA
CAR	47.77 ± 2.25	6.89 ± 0.77	15.55 ± 1.95	21.82 ± 1.36	31.64 ± 1.29
DRM	4.23 ± 1.30	2.73 ± 0.92	8.42 ± 3.55	6.00 ± 1.42	5.48 ± 1.66
IRS	13.59 ± 3.53	4.14 ± 1.89	6.29 ± 4.47	6.51 ± 2.38	2.55 ± 1.52
OPT	9.47 ± 0.57	4.34 ± 0.41	15.92 ± 1.27	2.94 ± 0.35	16.27 ± 0.82
PEN	15.70 ± 0.49	6.24 ± 0.40	7.45 ± 0.73	0.74 ± 0.13	27.58 ± 0.55
PIM	26.82 ± 1.88	23.35 ± 1.60	29.68 ± 4.24	30.37 ± 1.91	23.34 ± 1.68
SGM	15.66 ± 0.97	8.81 ± 0.75	8.00 ± 1.22	5.07 ± 0.66	14.63 ± 0.95
WIN	3.42 ± 1.62	2.41 ± 1.67	13.93 ± 6.08	5.40 ± 2.19	3.59 ± 1.89
YST	48.63 ± 1.75	41.10 ± 1.32	48.79 ± 4.63	49.19 ± 1.50	42.49 ± 6.31
ZOO	7.98 ± 4.14	5.88 ± 2.76	19.17 ± 9.23	7.70 ± 4.42	13.78 ± 4.72

The only 2-class dataset, *Pima*, is used in pilot experiments for model selection.

5.2. Results

In this study two sets of experiments are performed. In the first set, the performance of the proposed model selection algorithm, MM–SVM, is compared to the performances of multi–class SVMs which use the same type of kernel in all binary classifiers. For this comparison three types of models are considered: SVMs with linear kernels (L–SVM), SVMs with polynomial kernels of degree two (P2–SVM) and SVMs with polynomial kernels of degree three (P3–SVM). The performances of these classifiers are compared on five multi–class classification schemes: OVA classification scheme, pairwise classification scheme, and ECOC classification scheme with three different ECOC matrices. The results of these experiments are given in Section 5.2.1.

The second set of experiments are carried out to test the effectiveness of the proposed algorithm for constructing ECOC matrices. Three ECOC matrices with different number of binary classifiers are constructed. The ECOC matrix denoted by ECOC1 has as many binary classifiers as the number of classes. Hence, ECOC1 and OVA classification scheme have the same number of binary classifiers. In ECOC3, the number of binary classifiers is equal the number of binary classifiers in the pairwise classification scheme. And in ECOC2, the number of binary classifiers is equal to the average number of binary classifiers is OVA and pairwise classification schemes. The performance of the proposed algorithm for these three cases is compared to OVA classification scheme and pairwise classification scheme. The results of these experiments are given in Section 5.2.2.

The detailed results of the experiments are given in Appendix A.

5.2.1. Results Of Model Selection Experiments

In this section, the results of the experiments for the proposed model selection algorithm are presented. For different multi–class classification schemes, the average test error and the standard deviations in ten runs are given for L–SVM, P2–SVM, P3–SVM and MM–SVM on seven datasets.

The best classifier for each dataset is written in boldface. The best classifier is found by using the method for ordering and finding the best of $K > 2$ supervised learning algorithms proposed in [22]. In this method, a directed graph is formed where the vertices correspond to the supervised learning algorithms which are to be compared. The algorithms are given indices from $1, .., K$, as 1 being the the most preferred one according to some criteria, usually model complexity. In our study, the number of support vectors is used as the complexity measure, since during testing a summation over the support vectors is carried out. After algorithms are given indices, a directed edge from the i^{th} node to the j^{th} node is placed if the j^{th} algorithm is significantly better than the i^{th} algorithm according to a statistical test. If a vertex in this graph has no outgoing edges, this means that there is no other algorithm that has less ex-

pected error. Therefore, from the algorithms which have no outgoing edges, the one with the smallest index is selected as the best algorithm. In our experiments, 5×2 cv F test is used for pairwise comparisons of the algorithms. To construct the graph, first the classifiers are ordered in increasing number of support vectors, and then the one with the least number of support vectors is indexed by 1, the second one is indexed by 2, etc. Since a different MM–SVM is constructed for each dataset and multi–class classification scheme, this ordering is not the same for different cases. An example of the ordering of four algorithms is given in Figure 5.1. This graph corresponds to the results of PEN dataset using the ECOC classification scheme with ten binary classifiers (ECOC1). In this graph, the vertex with index 1 corresponds to P2–SVM, the vertex with index 2 corresponds to MM–SVM, the vertex with index 3 corresponds to P3–SVM and the vertex with index 4 corresponds to L–SVM. The vertices with no outgoing edges are MM–SVM and P3–SVM. But MM–SVM is chosen as the best classifier since it has a smaller index number.

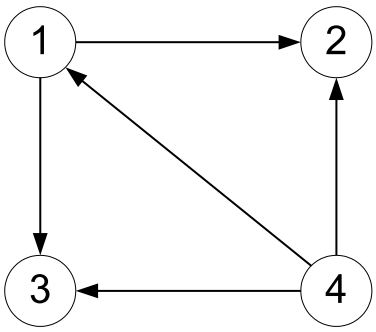


Figure 5.1. Ordering of four classifiers for the PEN dataset

The results for the OVA classification scheme is given in Table 5.3. For five datasets, the P3–SVM is the best classifier and for the remaining two datasets the P2–SVM is the best classifier. But these classifiers are significantly better than the MM–SVM only on CAR and OPT datasets, on the other datasets there is no significant difference. They are chosen as the best classifier because they have less number of support vectors. Moreover, in most of the datasets, the non–linear SVMs are significantly better than the L–SVM, which verifies that the chosen datasets are not linearly separable.

Table 5.3. Test errors in OVA classification scheme

	L–SVM	P2–SVM	P3–SVM	MM-SVM
CAR	20.03 ± 1.80	6.93 ± 0.62	**6.01 ± 0.60**	6.93 ± 0.62
DRM	2.52 ± 1.30	3.17 ± 1.51	**4.38 ± 2.27**	2.52 ± 1.30
OPT	4.00 ± 0.38	**1.73 ± 0.20**	1.52 ± 0.21	1.74 ± 0.20
PEN	5.76 ± 0.31	**0.65 ± 0.12**	0.55 ± 0.11	0.65 ± 0.13
SGM	10.52 ± 1.13	5.79 ± 0.63	**5.69 ± 0.61**	5.47 ± 0.57
YST	50.50 ± 3.80	41.06 ± 1.39	**41.09 ± 1.25**	41.98 ± 1.06
ZOO	5.37 ± 2.20	3.99 ± 1.93	**6.14 ± 1.12**	5.36 ± 2.88

The results for pairwise classification scheme are given in Table 5.4. The P3–SVM is chosen as the best algorithm in six of the seven datasets. But it is significantly better than the MM–SVM in only two of the datasets. The classifiers constructed by MM–SVM algorithm consists of SVMs with linear and second degree polynomial kernels. Nevertheless, they perform as well as P3–SVMs which have SVMs with third degree polynomial kernels in all of the base classifiers.

Table 5.4. Test errors in pairwise classification scheme

	L–SVM	P2–SVM	P3–SVM	MM–SVM
CAR	16.67 ± 0.96	6.50 ± 0.44	**6.55 ± 0.73**	6.57 ± 0.50
DRM	4.54 ± 1.89	**4.05 ± 1.70**	5.74 ± 1.77	4.54 ± 1.89
OPT	2.23 ± 0.26	1.62 ± 0.17	**1.38 ± 0.14**	2.16 ± 0.26
PEN	1.65 ± 0.12	0.60 ± 0.11	**0.56 ± 0.11**	0.88 ± 0.08
SGM	4.15 ± 0.41	3.75 ± 0.65	**4.30 ± 0.75**	4.15 ± 0.41
YST	41.43 ± 1.66	40.70 ± 0.96	**40.71 ± 1.52**	41.36 ± 1.63
ZOO	5.15 ± 2.91	4.56 ± 1.69	**5.37 ± 1.72**	5.15 ± 2.91

In Table 5.5, the results for the ECOC1 classification scheme are given. The P3–SVM performed significantly better than the other algorithms on CAR and PEN datasets. On five of the seven datasets, the P3–SVM is chosen as the best model. On the DRM and the PEN datasets, the MM–SVM is chosen as the best model.

Table 5.5. Test errors in ECOC1 scheme

	L–SVM	P2–SVM	P3–SVM	MM–SVM
CAR	21.52 ± 1.41	10.10 ± 0.55	**7.48 ± 0.77**	9.51 ± 0.49
DRM	3.28 ± 1.10	3.44 ± 1.19	5.85 ± 1.63	**3.28 ± 1.10**
OPT	14.58 ± 0.83	2.95 ± 0.37	**2.40 ± 0.26**	2.87 ± 0.38
PEN	25.07 ± 0.39	1.60 ± 0.14	0.96 ± 0.12	**0.97 ± 0.15**
SGM	20.52 ± 0.78	8.99 ± 0.71	**8.64 ± 0.73**	8.96 ± 0.53
YST	48.57 ± 1.32	43.30 ± 0.93	**42.96 ± 1.26**	43.53 ± 1.31
ZOO	7.55 ± 3.33	4.56 ± 1.86	**4.95 ± 2.36**	7.35 ± 3.19

The results for the ECOC2 case are given in Table 5.6. The situation is the similar to the ECOC1 case. P3–SVM is significantly better than all the other algorithms only on the CAR datasets. On five of the remaining six classes, there is no significant difference between the performances of P3–SVM and MM–SVM. But P3–SVM is chosen as the best algorithm because it has less number of support vectors.

Table 5.6. Test errors in ECOC2 scheme

	L–SVM	P2–SVM	P3–SVM	MM–SVM
CAR	21.78 ± 0.95	9.54 ± 0.52	**7.04 ± 0.66**	9.57 ± 0.51
DRM	2.68 ± 1.11	**2.79 ± 0.91**	4.65 ± 1.40	2.68 ± 1.11
OPT	9.99 ± 0.58	2.10 ± 0.32	**1.86 ± 0.35**	2.03 ± 0.26
PEN	15.72 ± 0.52	0.97 ± 0.13	0.70 ± 0.13	**0.72 ± 0.14**
SGM	17.94 ± 0.49	9.25 ± 0.73	**8.21 ± 0.65**	9.22 ± 0.67
YST	48.27 ± 1.11	42.23 ± 0.81	**41.84 ± 1.41**	42.76 ± 1.59
ZOO	6.74 ± 2.71	4.16 ± 2.16	**5.57 ± 1.37**	5.56 ± 2.71

In Table 5.7 the results for the ECOC3 scheme are given. In six of the seven datasets, P3–SVM is chosen as the best algorithm. But it is significantly better than the other algorithms only on the CAR dataset. On the other six datasets, it is chosen as the best algorithm because it has less number of support vectors. One important difference is that the classifiers constructed by MM–SVM consists of mostly SVMs with polynomial kernels of degree two. This results from the fact that the binary

problems here are more difficult than the binary problems in the pairwise scheme. In the pairwise scheme, each binary classifier separates two classes from each other, whereas in the ECOC3 scheme the binary classifiers separate two sets of classes from each other. This makes the binary problems of this scheme more difficult.

Table 5.7. Test errors in ECOC3 scheme

	L–SVM	P2–SVM	P3–SVM	MM–SVM
CAR	21.53 ± 1.04	8.47 ± 0.67	**6.62 ± 0.62**	8.59 ± 0.75
DRM	2.68 ± 1.69	**3.06 ± 1.52**	5.31 ± 1.95	2.68 ± 1.69
OPT	7.58 ± 0.39	1.83 ± 0.27	**1.66 ± 0.25**	1.75 ± 0.33
PEN	12.15 ± 0.40	0.82 ± 0.15	**0.60 ± 0.10**	0.65 ± 0.14
SGM	12.94 ± 0.53	8.60 ± 0.57	**6.88 ± 0.48**	7.00 ± 0.63
YST	46.20 ± 1.10	41.35 ± 0.79	**41.05 ± 0.95**	41.79 ± 0.93
ZOO	5.14 ± 1.90	4.37 ± 1.62	**4.77 ± 1.46**	4.76 ± 2.14

5.2.2. Results Of ECOC Experiments

In Table 5.8, the error rates for L–SVM are given for different multi–class classification schemes and datasets. It can be seen that the OVA scheme and the pairwise scheme has less error than the ECOC schemes for most of the datasets. This shows that the binary problems constructed by the proposed ECOC scheme are more difficult than the binary problems in the OVA and pairwise schemes. Therefore, they can not be separated by linear SVMs.

Table 5.8. Error rate of L–SVMs

	OVA	Pairwise	ECOC1 (k)	ECOC2 ($k(k-3)/4$)	ECOC3 ($k(k-1)/2$)
CAR	20.03 ± 1.80	16.67 ± 0.96	21.52 ± 1.41	21.78 ± 0.95	21.53 ± 1.04
DRM	2.52 ± 1.30	4.54 ± 1.90	3.28 ± 1.10	2.68 ± 1.11	2.68 ± 1.69
OPT	4.00 ± 0.38	2.23 ± 0.26	14.58 ± 0.83	9.99 ± 0.58	7.58 ± 0.39
PEN	5.76 ± 0.31	1.65 ± 0.12	25.07 ± 0.39	15.72 ± 0.52	12.15 ± 0.40
SGM	10.52 ± 1.13	4.15 ± 0.41	20.52 ± 0.78	17.94 ± 0.49	12.94 ± 0.53
YST	50.50 ± 3.80	41.43 ± 1.66	48.57 ± 1.32	48.27 ± 1.11	46.20 ± 1.10
ZOO	5.37 ± 2.20	5.15 ± 2.91	7.55 ± 3.33	6.74 ± 2.71	5.14 ± 1.90

The results for P2–SVM and P3–SVM are given in Table 5.9 and 5.10. Again the OVA and pairwise schemes performs better than the ECOC schemes for most of the datasets. This suggests that even using polynomial kernels of second and third degree does not separate the data of the underlying binary problems.

The results for the MM–SVM are given in Table 5.11. On most of the datasets the ECOC scheme performs better than the OVA and pairwise schemes. In particular, the ECOC3 scheme performs better the pairwise scheme. This suggests that the same number of binary classifiers are used, it is better to design the binary problems so that they use whole training set rather than some of the classes. Unfortunately, this increases the complexity, i.e. the number of support vectors.

Table 5.9. Error rate of P2–SVM

	OVA	Pairwise	ECOC1 (k)	ECOC2 ($k(k-3)/4$)	ECOC3 ($k(k-1)/2$)
CAR	6.93 ± 0.62	6.50 ± 0.44	10.10 ± 0.55	9.54 ± 0.52	8.47 ± 0.67
DRM	3.17 ± 1.51	4.05 ± 1.70	3.44 ± 1.19	2.79 ± 0.91	3.06 ± 1.52
OPT	1.73 ± 0.20	1.62 ± 0.17	2.95 ± 0.37	2.10 ± 0.32	1.83 ± 0.27
PEN	0.65 ± 0.12	0.60 ± 0.11	1.60 ± 0.14	0.97 ± 0.13	0.82 ± 0.15
SGM	5.79 ± 0.63	3.75 ± 0.65	8.99 ± 0.71	9.25 ± 0.73	8.60 ± 0.57
YST	41.06 ± 1.39	40.70 ± 0.96	43.30 ± 0.92	42.23 ± 0.81	41.35 ± 0.79
ZOO	3.99 ± 1.93	4.56 ± 1.69	4.56 ± 1.86	4.16 ± 2.16	4.37 ± 1.62

Table 5.10. Error rate of P3–SVM

	OVA	Pairwise	ECOC (k)	ECOC ($k(k-3)/4$)	ECOC ($k(k-1)/2$)
CAR	6.01 ± 0.60	6.55 ± 0.73	7.48 ± 0.77	7.04 ± 0.66	6.62 ± 0.62
DRM	4.38 ± 2.27	5.74 ± 1.77	5.85 ± 1.63	4.65 ± 1.40	5.31 ± 1.95
OPT	1.52 ± 0.21	1.38 ± 0.14	2.40 ± 0.26	1.86 ± 0.34	1.66 ± 0.25
PEN	0.55 ± 0.11	0.56 ± 0.10	0.96 ± 0.12	0.70 ± 0.13	0.60 ± 0.10
SGM	5.69 ± 0.61	4.30 ± 0.75	8.64 ± 0.73	8.21 ± 0.65	6.88 ± 0.48
YST	41.09 ± 1.24	40.71 ± 1.52	42.96 ± 1.26	41.84 ± 1.41	41.05 ± 0.95
ZOO	6.14 ± 1.12	5.37 ± 1.72	4.95 ± 2.36	5.57 ± 1.37	4.77 ± 1.46

The relationship between the test error and the different classification schemes can be seen in Figures 5.2–5.6 for different datasets. The conclusions drawn from the previous tables can be seen in these figures. On most of the datasets, the performance of the pairwise scheme is better than the other schemes for all of the algorithms considered. Another point is that the performance of the ECOC scheme increases as the number of binary classifiers is increased as expected.

Table 5.11. Error rate of the MM–SVM

	OVA	Pairwise	ECOC1 (k)	ECOC2 ($k(k-3)/4$)	ECOC3 ($k(k-1)/2$)
CAR	6.93 ± 0.62	6.57 ± 0.50	9.51 ± 0.49	9.57 ± 0.51	8.59 ± 0.75
DRM	2.52 ± 1.30	4.54 ± 1.89	3.28 ± 1.10	2.68 ± 1.11	2.68 ± 1.69
OPT	1.74 ± 0.20	2.16 ± 0.26	2.87 ± 0.38	2.03 ± 0.26	1.75 ± 0.33
PEN	0.65 ± 0.13	0.88 ± 0.08	0.97 ± 0.15	0.72 ± 0.14	0.65 ± 0.14
SGM	5.47 ± 0.57	4.15 ± 0.41	8.96 ± 0.53	9.22 ± 0.67	7.00 ± 0.63
YST	41.98 ± 1.06	41.36 ± 1.63	43.53 ± 1.31	42.76 ± 1.59	41.79 ± 0.93
ZOO	5.36 ± 2.88	5.15 ± 2.91	7.35 ± 3.19	5.56 ± 2.71	4.76 ± 2.14

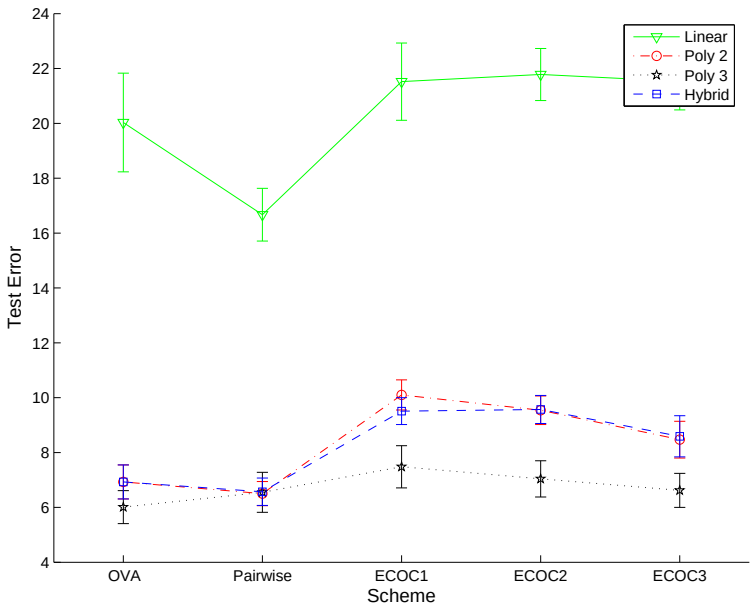


Figure 5.2. Test error vs. classification schemes on CAR data

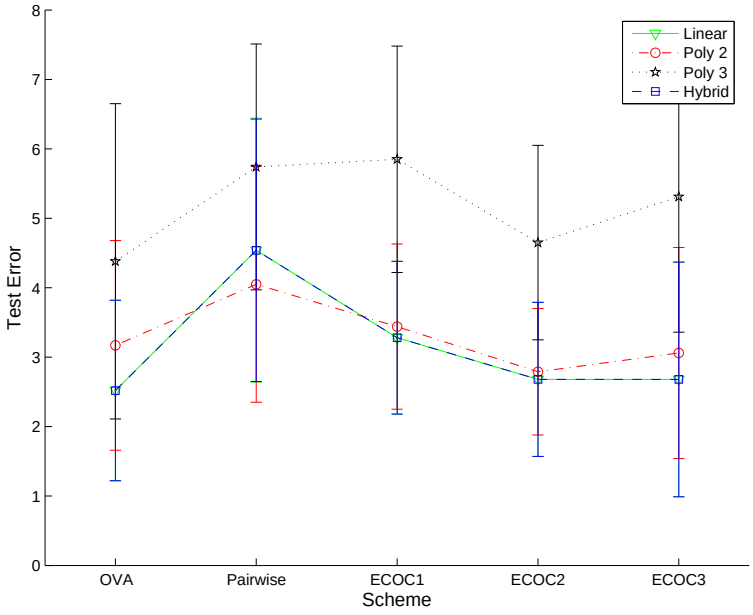


Figure 5.3. Test error vs. classification schemes on DRM data

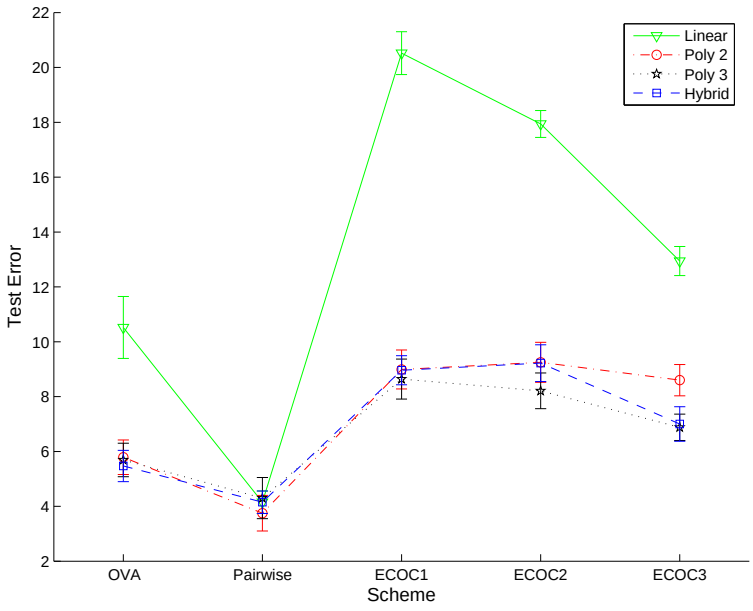


Figure 5.4. Test error vs. classification schemes on SGM data

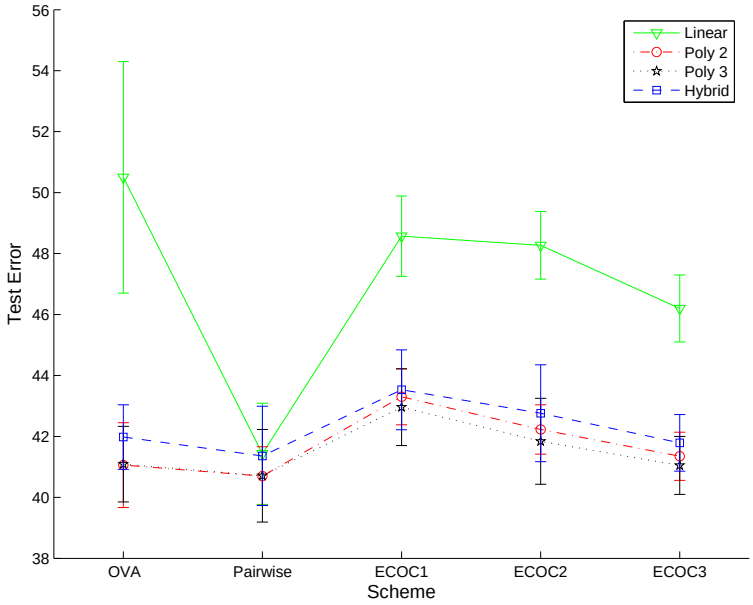


Figure 5.5. Test error vs. classification schemes on YST data

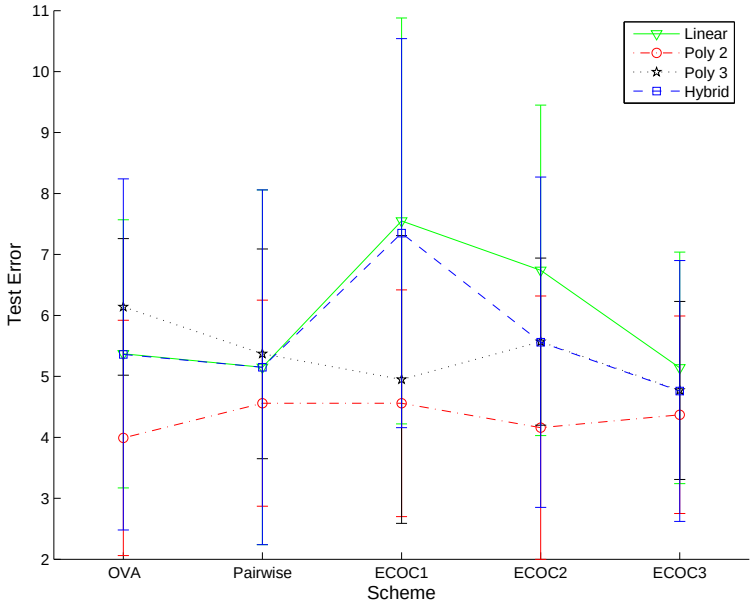


Figure 5.6. Test error vs. classification schemes on ZOO data

In the following tables number of support vectors for different classification schemes are given. The number of support vectors for L–SVM can be seen in Table 5.12. The smallest number of support vectors found for a dataset is indicated by boldface. For most of the cases, the OVA scheme has the least number of support vectors. It should be noted that the number of binary classifiers is not the same in all of the multi–class classification schemes. Therefore, when comparing two schemes in terms of the number of support vectors, the number of binary classifiers should be considered. It is interesting that in CAR and YST datasets, the number of support vectors in the pairwise scheme is less than the number of support vectors in the OVA scheme, although there are more binary classifiers in the pairwise scheme. This results from the fact that the binary classifiers share most of their support vectors.

Table 5.12. Number of support vectors for L–SVM

	OVA	Pairwise	ECOC1 (k)	ECOC2 ($k(k-3)/4$)	ECOC3 ($k(k-1)/2$)
CAR	392.80 ± 18.52	**293.60 ± 13.48**	390.40 ± 16.03	394.40 ± 15.98	397.90 ± 15.18
DRM	**88.70 ± 2.26**	106.40 ± 3.27	110.30 ± 2.21	114.90 ± 3.57	102.20 ± 3.05
OPT	**716.10 ± 14.64**	1185.60 ± 7.81	1491.90 ± 16.56	1693.20 ± 12.79	1763.90 ± 12.70
PEN	**1328.50 ± 25.15**	1530.70 ± 6.60	3547.00 ± 15.31	3676.20 ± 4.92	3722.40 ± 4.53
SGM	**596.70 ± 15.73**	654.30 ± 12.21	767.80 ± 11.84	860.30 ± 7.67	662.20 ± 10.35
YST	704.00 ± 7.47	**612.40 ± 12.75**	709.00 ± 6.93	716.60 ± 5.38	720.90 ± 4.43
ZOO	33.30 ± 2.58	34.40 ± 2.17	33.20 ± 2.35	36.10 ± 2.77	**29.70 ± 2.00**

The number of support vectors in P2–SVM are given in Table 5.13. The situation is similar to the L–SVM case. One difference is seen on DRM dataset. The pairwise scheme has the least number of support vectors instead of the OVA scheme.

Table 5.13. Number of support vectors for P2–SVM

	OVA	Pairwise	ECOC1 (k)	ECOC2 ($k(k-3)/4$)	ECOC3 ($k(k-1)/2$)
CAR	213.40 ± 37.36	**151.30 ± 7.20**	255.60 ± 23.31	259.30 ± 23.94	264.20 ± 23.27
DRM	82.10 ± 2.42	**74.10 ± 2.51**	92.90 ± 3.78	100.60 ± 2.95	83.00 ± 3.71
OPT	**590.70 ± 13.85**	1032.60 ± 11.64	798.10 ± 18.33	961.50 ± 15.64	1068.40 ± 16.39
PEN	**510.90 ± 15.22**	1744.00 ± 17.76	979.90 ± 13.27	1210.60 ± 22.96	1422.00 ± 30.20
SGM	**418.30 ± 12.69**	756.90 ± 7.72	501.20 ± 45.48	659.90 ± 26.71	419.20 ± 28.88
YST	669.30 ± 8.19	**613.90 ± 10.27**	676.90 ± 9.15	695.80 ± 6.99	702.80 ± 5.73
ZOO	31.10 ± 2.64	31.10 ± 2.56	32.20 ± 1.55	34.60 ± 2.37	**30.60 ± 1.96**

In Table 5.14, the number of support vectors for P3 –SVM are given. For P3–SVM, the pairwise scheme has the least number of support vectors for four of the seven datasets. This suggest that for more complex decision boundaries, i.e. for more complex kernel functions, the number of support vectors shared by binary classifiers increases.

Table 5.14. Number of support vectors for P3–SVM

	OVA	Pairwise	ECOC1 (k)	ECOC2 ($k(k-3)/4$)	ECOC3 ($k(k-1)/2$)
CAR	162.80 ± 6.96	**133.40 ± 6.47**	208.10 ± 10.40	212.90 ± 10.99	217.50 ± 11.23
DRM	65.30 ± 4.03	**56.00 ± 5.40**	75.20 ± 5.75	83.80 ± 6.34	62.50 ± 5.02
OPT	**625.30 ± 13.02**	848.50 ± 17.30	743.60 ± 13.56	886.90 ± 16.78	981.40 ± 19.84
PEN	**668.50 ± 18.54**	790.80 ± 21.18	1015.40 ± 36.00	1337.90 ± 45.34	1631.70 ± 53.93
SGM	394.80 ± 6.86	**243.40 ± 9.73**	473.60 ± 9.90	599.40 ± 12.95	386.50 ± 11.50
YST	657.60 ± 9.35	**598.30 ± 9.27**	673.20 ± 7.97	692.90 ± 7.31	700.90 ± 6.59
ZOO	30.50 ± 2.42	32.00 ± 2.49	30.50 ± 2.01	33.10 ± 1.60	**29.50 ± 1.96**

Finally, the number of support vectors for the MM–SVM are given in Table 5.15. The situation in this case is the same as the L–SVM.

Table 5.15. Number of support vectors for MM–SVM

	OVA	Pairwise	ECOC1 (k)	ECOC2 ($k(k-3)/4$)	ECOC3 ($k(k-1)/2$)
CAR	213.40 ± 37.36	**154.40 ± 6.98**	367.20 ± 13.12	369.30 ± 13.66	370.00 ± 13.77
DRM	**88.70 ± 2.26**	106.40 ± 3.27	110.30 ± 2.21	114.90 ± 3.57	102.20 ± 3.05
OPT	**588.60 ± 14.45**	1178.10 ± 8.29	795.00 ± 16.83	955.00 ± 15.81	1054.90 ± 15.60
PEN	**522.80 ± 14.05**	1505.10 ± 8.57	993.60 ± 35.39	1327.50 ± 43.37	1841.10 ± 47.69
SGM	**451.00 ± 10.17**	654.30 ± 12.21	680.00 ± 11.29	771.10 ± 11.86	475.60 ± 9.18
YST	686.80 ± 7.19	**612.30 ± 12.65**	679.40 ± 9.03	712.90 ± 5.70	719.40 ± 4.48
ZOO	33.60 ± 2.67	34.40 ± 2.17	33.80 ± 1.81	36.50 ± 2.72	**31.60 ± 1.90**

In the following figures the number of support vectors is plotted against the classification schemes.

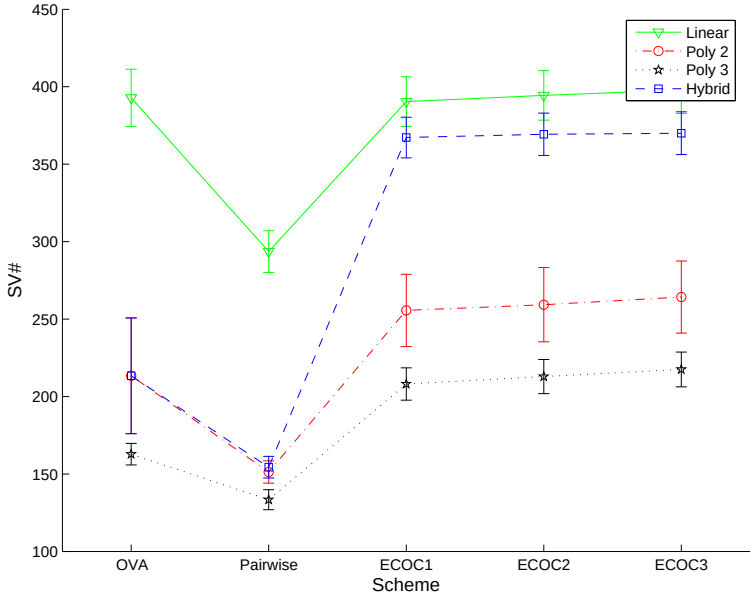


Figure 5.7. Number of support vectors vs. classification schemes on CAR data

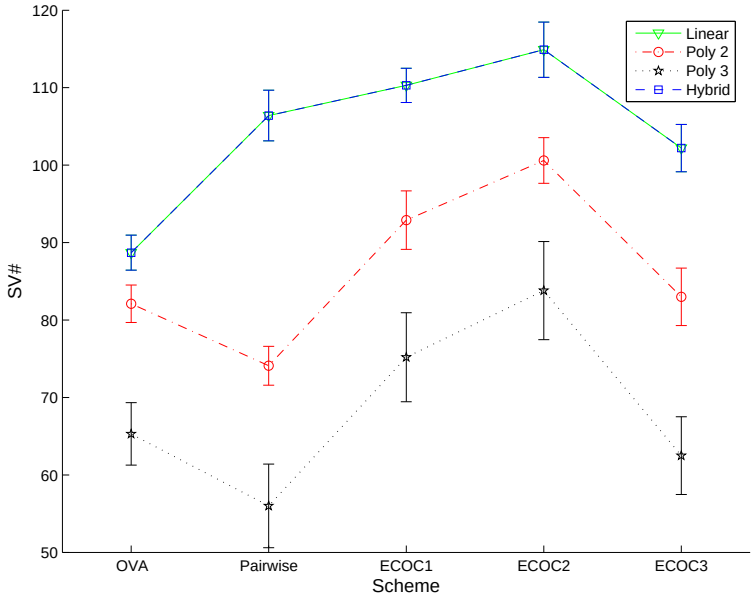


Figure 5.8. Number of support vectors vs. classification schemes on DRM data

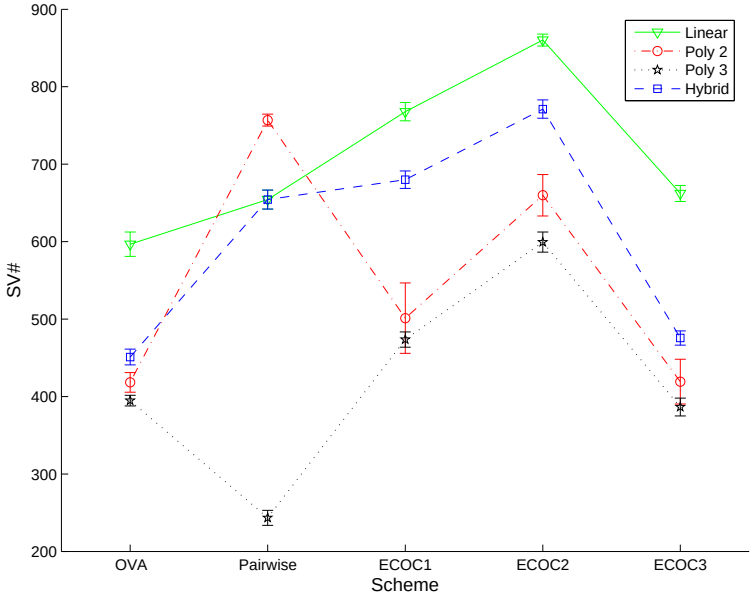


Figure 5.9. Number of support vectors vs. classification schemes on SGM data

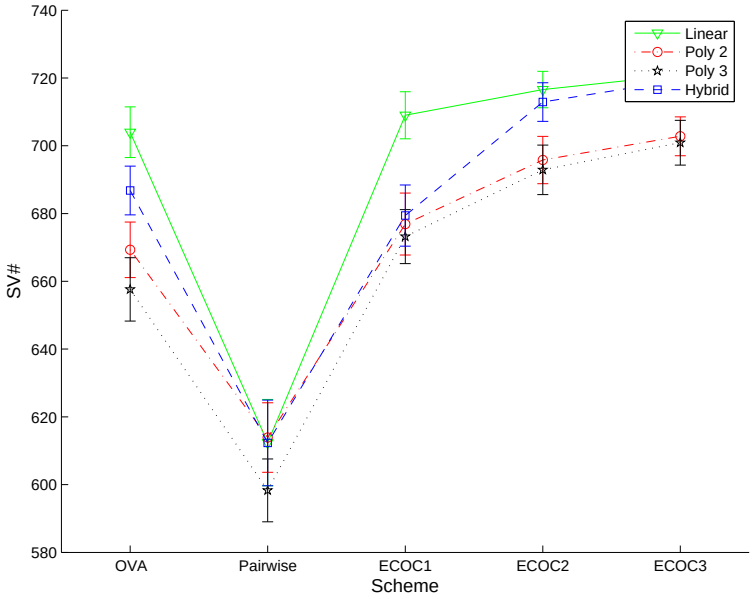


Figure 5.10. Number of support vectors vs. classification schemes on YST data

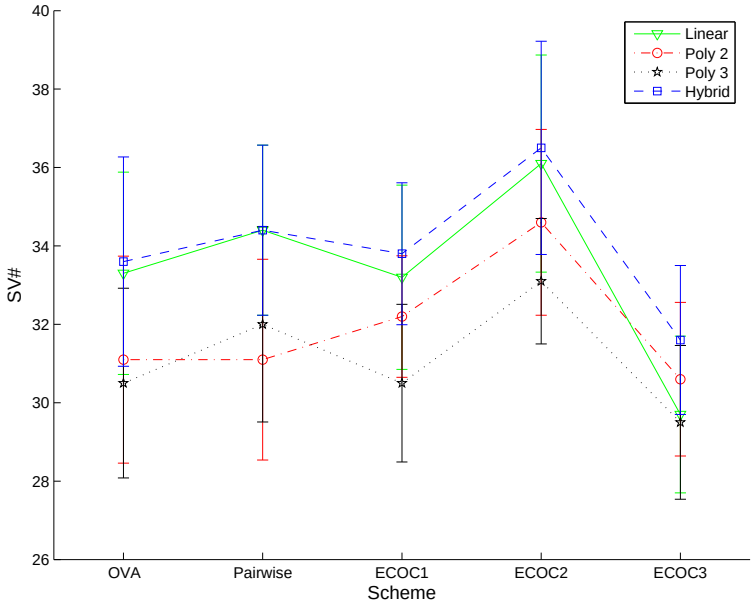


Figure 5.11. Number of support vectors vs. classification schemes on ZOO data

6. CONCLUSIONS

In this thesis, model selection problem for SVMs in multi–class classification problems is studied. The SVM algorithm is originally designed for binary problems. Several methods are proposed to extend them to multi–class classification problems. Commonly used methods are based on the idea of decomposing the problem into multiple binary classification problems and combining the outputs of these binary classifiers. As for the model selection, the same set of hyper–parameters are used for all of the binary classifiers. But since the difficulties of the individual binary classification problems are not the same, model selection should be done separately for each of them. In the proposed MM–SVM algorithm, the hyper–parameters of the underlying binary classifiers are optimized separately, and the resulting multi–class classifier is a combination of SVMs with different types of kernels and different values of the parameter C.

In the MM–SVM algorithm, first the value of the parameter C is selected for each binary classifier. This is done by training SVMs for ten different values of C and selecting the one which gives the best accuracy on a validation set. After C is selected for each binary classifier, the best kernel type is chosen. For kernel selection three types of kernels are considered; linear kernels, second degree polynomial kernels and third degree polynomial kernels. Again for each binary classifier, SVMs with these kernels are trained and the selection is done based on the result of 5×2 cv F test.

The performance of the MM–SVM is compared to classical SVMs in terms of accuracy and complexity. In most of the cases, MM–SVM performs as good as classical SVMs. We see that a hybrid model, which has linear and second degree polynomial binary classifiers can perform significantly better than the combination of all-linear binary classifier. Moreover, the hybrid model is simpler than the other one in the sense that it has fewer number of support vectors.

When a multi–class problem is decomposed into multiple binary problems, the difficulties of these binary classifiers will be different. A kernel type which performs well on difficult problems, will be too complex for the simpler problems and will overfit. On the other hand, a kernel type which is just good enough for the simplest binary problems will perform poorly on more difficult problems. Therefore, using different types of kernels for different binary classifiers in a multi–class problem is a good idea. The results of our experiments support this idea. But experiments also show that the kernel types considered do not constitute a rich set of functions. Therefore, a larger set of kernel types including polynomial kernels of higher degrees and gaussian kernels should be used for model selection of the binary classifiers.

For multi–class classification, an algorithm based on Hill Climbing is proposed for constructing ECOC matrices. The algorithm constructs the ECOC matrix iteratively by selecting the column from the full code which improves the row and column separability the most. The sum of the Hamming distances between each pair of rows and each pair of columns is used as a measure of the row and column separability.

The performance of the proposed algorithm is compared to classical OVA and pairwise classification schemes. ECOC matrices of three different sizes are constructed for each problem. The first one (ECOC1) has as many columns as number of classes. We see that the classical OVA scheme performs significantly better than ECOC1 scheme. But the training time for OVA scheme is much larger since the size of the underlying binary problems is equal to the original problem size. The pairwise classification scheme performs significantly better than all the other schemes in almost all of the cases. In pairwise classification scheme each binary classifier separates two classes from each other. Therefore, the binary problems into which the original multi–class problem is decomposed are the simplest binary problems. This again suggests that more complex kernel types should be used to train the binary classifiers.

As a conclusion, this study showed that the proposed hybrid model together with ECOC finds accurate solutions for multi–class problems without significantly increasing complexity. The accuracy of the MM–SVM can be increased further by using more

complex types of kernels.

APPENDIX A: DETAILED RESULTS OF EXPERIMENTS

Table A.1. Results on CAR dataset

CAR	One vs. All			
	error ± st. dev.	# of SVs	% SV	
L–SVM	20.03 ± 1.80	392.80 ± 18.52	45.46	
P2–SVM	6.93 ± 0.62	213.40± 37.36	24.70	> Lin
P3–SVM	6.01 ± 0.60	162.80 ± 6.96	18.84	> Lin, Poly2, H
MM–SVM (2 2 2 2)	6.93 ± 0.62	213.40 ± 37.36	24.70	> Lin
	Pairwise			
L–SVM	16.67 ± 0.96	293.60 ± 13.48	33.98	
P2–SVM	6.50 ± 0.44	151.30 ± 7.20	17.51	> Lin
P3–SVM	6.55 ± 0.73	133.40 ± 6.47	15.44	> Lin
MM–SVM (2 2 2 2 1 1)	6.57 ± 0.50	154.40 ± 6.98	17.87	> Lin
	ECOC1			
L–SVM	21.52 ± 1.41	390.40 ± 16.03	45.19	
P2–SVM	10.10 ± 0.55	255.60 ± 23.31	29.58	>Lin
P3–SVM	7.48 ± 0.77	208.10 ± 10.40	24.09	>Lin, Poly2, H
MM–SVM (3 2 2 1)	9.51 ± 0.49	367.20 ± 13.12	42.50	>Lin
	ECOC2			
L–SVM	21.78 ± 0.95	394.40 ± 15.98	45.65	
P2–SVM	9.54 ± 0.52	259.30 ± 23.94	30.01	>Lin
P3–SVM	7.04 ± 0.66	212.90 ± 10.99	24.64	>Lin, Poly2, H
MM–SVM (3 2 2 1 2)	9.57 ± 0.51	369.30 ± 13.66	42.74	>Lin
	ECOC3			
L–SVM	21.53 ± 1.04	397.90 ± 15.18	46.05	
P2–SVM	8.47 ± 0.67	264.20 ± 23.27	30.58	>Lin
P3–SVM	6.62 ± 0.62	217.50 ± 11.23	25.17	>Lin, Poly2, H
MM–SVM (3 2 2 1 2 2)	8.59 ± 0.75	370.00 ± 13.77	42.82	>Lin

Table A.2. Results on Dermatology dataset

DRM	One vs. All			
	error ± st. dev.	# of SVs	% SV	
L–SVM	2.52 ± 1.30	88.70 ± 2.26	47.18	
P2–SVM	3.17 ± 1.51	82.10 ± 2.42	43.67	
P3–SVM	4.38 ± 2.27	65.30 ± 4.03	34.73	
MM–SVM (1 1 1 ...)	2.52 ± 1.30	88.70 ± 2.26	47.18	
	Pairwise			
L–SVM	4.54 ± 1.89	106.40 ± 3.27	58.14	
P2–SVM	4.05 ± 1.70	74.10 ± 2.51	40.49	
P3–SVM	5.74 ± 1.77	56.00 ± 5.40	30.60	
MM–SVM (1 1 1 ...)	4.54 ± 1.89	106.40 ± 3.27	58.14	
	ECOC1			
L–SVM	3.28 ± 1.10	102.20 ± 3.05	55.85	
P2–SVM	3.44 ± 1.19	83.00 ± 3.71	45.36	>Poly3
P3–SVM	5.85 ± 1.63	62.50 ± 5.02	34.15	
MM–SVM (1 1 1 ...)	3.28 ± 1.10	102.20 ± 3.05	55.85	
	ECOC2			
L–SVM	2.68 ± 1.11	110.30 ± 2.21	60.27	
P2–SVM	2.79 ± 0.91	92.90 ± 3.78	50.77	>Poly3
P3–SVM	4.65 ± 1.40	75.20 ± 5.75	41.09	
MM–SVM (1 1 1 ...)	2.68 ± 1.11	110.30 ± 2.21	60.27	
	ECOC3			
L–SVM	2.68 ± 1.69	114.90 ± 3.57	62.79	
P2–SVM	3.06 ± 1.52	100.60 ± 2.95	54.97	>Poly3
P3–SVM	5.31 ± 1.95	83.80 ± 6.34	45.79	
MM–SVM (1 1 1 ...)	2.68 ± 1.69	114.90 ± 3.57	62.79	

Table A.3. Results on Iris dataset

IRS	One vs. All			
	error ± st. dev.	# of SVs	% SV	
L–SVM	10.80 ± 3.95	57.60 ± 1.78	0.77	
P2–SVM	3.87 ± 1.47	32.80 ± 2.57	0.44	
P3–SVM	3.33 ± 1.69	18.00 ± 1.83	0.24	> Lin
MM–SVM (1 2 1)	3.87 ± 1.33	41.10 ± 1.91	0.55	
	Pairwise			
L–SVM	3.07 ± 1.78	41.70 ± 1.89	0.56	
P2–SVM	2.93 ± 1.76	29.80 ± 2.62	0.40	>Poly3
P3–SVM	4.80 ± 1.69	23.20 ± 2.86	0.31	
MM–SVM (1 1 1)	3.07 ± 1.78	41.70 ± 1.89	0.56	
	Full ECOC			
L–SVM	2.93 ± 1.76	58.40 ± 1.90	77.87	
P2–SVM	2.93 ± 1.38	35.90 ± 3.45	47.87	
P3–SVM	4.00 ± 1.99	27.00 ± 2.16	36.00	
MM–SVM (1 1 2 1 1 1)	3.07 ± 1.78	44.20 ± 2.04	58.93	

Table A.4. Results on Optdigits dataset

OPT	One vs. All			
	error ± st. dev.	# of SVs	% SV	
L–SVM	4.00 ± 0.38	716.10 ± 14.64	37.45	
P2–SVM	1.73 ± 0.20	590.70 ± 13.85	30.89	> Lin
P3–SVM	1.52 ± 0.21	625.30 ± 13.02	32.70	> Lin, H
MM–SVM (1 2 2 3 2 2 1 1 2 2)	1.74 ± 0.20	588.60 ± 14.45	30.78	> Lin
	Pairwise			
L–SVM	2.23 ± 0.26	1185.60 ± 7.80	62.01	
P2–SVM	1.62 ± 0.17	1032.60 ± 11.60	54.01	> Lin, H
P3–SVM	1.38 ± 0.14	848.50 ± 17.30	44.38	> Lin, H
MM–SVM (Mixture 1 & 2)	2.16 ± 0.26	1178.10 ± 8.30	61.62	
	ECOC1			
L–SVM	14.58 ± 0.83	1491.90 ± 16.56	78.07	
P2–SVM	2.95 ± 0.37	798.10 ± 18.33	41.76	>Lin
P3–SVM	2.40 ± 0.26	743.60 ± 13.56	38.91	>Lin, Poly2, H
MM–SVM (Mixture 2 & 3)	2.87 ± 0.38	795.00 ± 16.83	41.60	>Lin
	ECOC2			
L–SVM	9.99 ± 0.58	1693.20 ± 12.79	88.60	
P2–SVM	2.10 ± 0.32	961.50 ± 15.64	50.31	>Lin
P3–SVM	1.86 ± 0.35	886.90 ± 16.78	46.41	>Lin
MM–SVM (Mixture 2 & 3)	2.03 ± 0.26	955.00 ± 15.81	49.97	>Lin
	ECOC3			
L–SVM	7.58 ± 0.39	1763.90 ± 12.70	92.28	
P2–SVM	1.83 ± 0.27	1068.40 ± 16.39	55.87	>Lin
P3–SVM	1.66 ± 0.25	981.40 ± 19.84	51.32	>Lin
MM–SVM (Mixture 2 & 3)	1.75 ± 0.33	1054.90 ± 15.60	55.13	>Lin

Table A.5. Results on Pendigits dataset

PEN	One vs. All			
	error ± st. dev.	# of SVs	% SV	
L–SVM	5.76 ± 0.31	1328.50 ± 25.10	0.35	
P2–SVM	0.65 ± 0.12	510.90 ± 15.20	0.14	> Lin
P3–SVM	0.55 ± 0.11	668.50 ± 18.50	0.18	> Lin
MM–SVM (Mixture 1 & 2)	0.65 ± 0.13	522.80 ± 14.10	0.14	> Lin
	Pairwise			
L–SVM	1.65 ± 0.12	1530.70 ± 6.60	0.41	
P2–SVM	0.60 ± 0.11	1744.00 ± 17.80	0.47	> Lin, H
P3–SVM	0.56 ± 0.11	790.80 ± 21.20	0.21	> Lin, H
MM–SVM (Mixture of 1 & 2)	0.88 ± 0.08	1505.10 ± 8.60	0.40	> Lin
	ECOC1			
L–SVM	25.07 ± 0.39	3547.00 ± 15.31	94.66	
P2–SVM	1.60 ± 0.14	979.90 ± 13.27	26.15	>Lin
P3–SVM	0.96 ± 0.12	1015.40 ± 36.00	27.10	>Lin, Poly2
MM–SVM (Mixture of 2 & 3)	0.97 ± 0.15	993.60 ± 35.39	26.52	>Lin, Poly2
	ECOC2			
L–SVM	15.72 ± 0.52	3676.20 ± 4.92	98.11	
P2–SVM	0.97 ± 0.13	1210.60 ± 22.96	32.31	>Lin
P3–SVM	0.70 ± 0.13	1337.90 ± 45.34	35.71	>Lin, Poly2
MM–SVM (Mixture of 2 & 3)	0.72 ± 0.14	1327.50 ± 43.37	35.43	>Lin, Poly2
	ECOC3			
L–SVM	12.15 ± 0.40	3722.40 ± 4.53	99.33	
P2–SVM	0.82 ± 0.15	1422.00 ± 30.20	37.95	>Lin
P3–SVM	0.60 ± 0.10	1631.70 ± 53.93	43.53	>Lin Poly2
MM–SVM (Mixture of 1, 2 & 3)	0.65 ± 0.14	1841.10 ± 47.69	49.13	>Lin, Poly2

Table A.6. Results on Segment dataset

SGM	One vs. All			
	error ± st. dev.	# of SVs	% SV	
L–SVM	10.52 ± 1.13	596.70 ± 15.73	0.57	
P2–SVM	5.79 ± 0.63	418.30 ± 12.69	0.40	> Lin
P3–SVM	5.69 ± 0.61	394.80 ± 6.86	0.38	> Lin
MM–SVM (1 1 3 2 2 2 1 1)	5.47 ± 0.57	451.00 ± 10.17	0.43	> Lin
	Pairwise			
L–SVM	4.15 ± 0.41	654.30 ± 12.21	0.62	
P2–SVM	3.75 ± 0.65	756.90 ± 7.72	0.72	
P3–SVM	4.30 ± 0.75	243.40 ± 9.73	0.23	
MM–SVM (1 1 1 ...)	4.15 ± 0.41	654.30 ± 12.21	0.62	
	ECOC1			
L–SVM	20.52 ± 0.78	662.20 ± 10.35	63.07	
P2–SVM	8.99 ± 0.71	419.20 ± 28.88	39.92	> Lin
P3–SVM	8.64 ± 0.73	386.50 ± 11.50	36.81	> Lin
MM–SVM (2 1 3 2 1 2 2)	8.96 ± 0.53	475.60 ± 9.18	45.30	> Lin
	ECOC2			
L–SVM	17.94 ± 0.49	767.80 ± 11.84	73.12	
P2–SVM	9.25 ± 0.73	501.20 ± 45.48	47.73	> Lin
P3–SVM	8.21 ± 0.65	473.60 ± 9.90	45.10	> Lin
MM–SVM (Mixture of 1, 2 & 3)	9.22 ± 0.67	680.00 ± 11.29	64.76	> Lin
	ECOC3			
L–SVM	12.94 ± 0.53	860.30 ± 7.67	81.93	
P2–SVM	8.60 ± 0.57	659.90 ± 26.71	62.85	>Lin
P3–SVM	6.88 ± 0.48	599.40 ± 12.95	57.09	>Lin, Poly2
MM–SVM (Mixture of 1, 2 & 3)	7.00 ± 0.63	771.10 ± 11.86	73.44	>Lin, Poly2

Table A.7. Results on Wine dataset

WIN	One vs. All			
	error ± st. dev.	# of SVs	% SV	
L–SVM	5.40 ± 2.05	54.40 ± 2.46	0.63	
P2–SVM	5.29 ± 2.00	47.50 ± 2.84	0.55	
P3–SVM	6.74 ± 3.66	78.00 ± 3.40	0.90	
MM–SVM (1 1 1)	5.40 ± 2.05	54.40 ± 2.46	0.63	
	Pairwise			
L–SVM	9.10 ± 2.86	32.50 ± 3.21	0.37	
P2–SVM	6.96 ± 2.56	28.90 ± 3.07	0.33	>Poly3
P3–SVM	10.66 ± 2.44	39.50 ± 5.08	0.45	
MM–SVM (1 1 1)	9.10 ± 2.86	32.50 ± 3.21	0.37	
	Full ECOC			
L–SVM	6.97 ± 3.14	54.50 ± 2.42	61.24	
P2–SVM	5.50 ± 3.00	47.70 ± 2.87	53.60	
P3–SVM	8.54 ± 3.55	80.20 ± 3.22	90.11	
MM–SVM (1 1 1 ...)	6.97 ± 3.14	54.50 ± 2.42	61.24	

Table A.8. Results on Yeast dataset

YST	One vs. All			
	error ± st. dev.	# of SVs	% SV	
L–SVM	50.50 ± 3.80	704.00 ± 7.47	0.95	
P2–SVM	41.06 ± 1.39	669.30 ± 8.19	0.90	
P3–SVM	41.09 ± 1.25	657.60 ± 9.35	0.89	
MM–SVM (2 1 2 1 1 1 2 1 1 1)	41.98 ± 1.06	686.80 ± 7.19	0.93	
	Pairwise			
L–SVM	41.43 ± 1.66	612.40 ± 12.75	0.83	
P2–SVM	40.70 ± 0.96	613.90 ± 10.27	0.83	
P3–SVM	40.71 ± 1.52	598.30 ± 9.27	0.81	
MM–SVM (26 = 2 others =1)	41.36 ± 1.63	612.30 ± 12.65	0.83	> Lin
	ECOC1			
L–SVM	48.57 ± 1.32	709.00 ± 6.93	95.55	
P2–SVM	43.30 ± 0.93	676.90 ± 9.15	91.23	> Lin
P3–SVM	42.96 ± 1.26	673.20 ± 7.97	90.73	> Lin
MM–SVM (Mixture of 1 & 2)	43.53 ± 1.31	679.40 ± 9.03	91.56	> Lin
	ECOC2			
L–SVM	48.27 ± 1.11	716.60 ± 5.38	96.58	
P2–SVM	42.23 ± 0.81	695.80 ± 6.99	93.77	> Lin
P3–SVM	41.84 ± 1.41	692.90 ± 7.31	93.38	> Lin
MM–SVM (Mixture of 1 & 2)	42.76 ± 1.59	712.90 ± 5.70	96.08	> Lin
	ECOC3			
L–SVM	46.20 ± 1.10	720.90 ± 4.43	97.16	
P2–SVM	41.35 ± 0.79	702.80 ± 5.73	94.72	> Lin
P3–SVM	41.05 ± 0.95	700.90 ± 6.59	94.46	> Lin
MM–SVM (Mixture of 1 & 2)	41.79 ± 0.93	719.40 ± 4.48	96.95	> Lin

Table A.9. Results on Zoo dataset

ZOO	One vs. All			
	error ± st. dev.	# of SVs	% SV	
L–SVM	5.37 ± 2.20	33.30 ± 2.58	0.67	
P2–SVM	3.99 ± 1.93	31.10 ± 2.64	0.62	
P3–SVM	6.14 ± 1.12	30.50 ± 2.42	0.61	
MM–SVM (1 1 2 1 1 1 1)	5.36 ± 2.88	33.60 ± 2.68	0.67	
	Pairwise			
L–SVM	5.15 ± 2.91	34.40 ± 2.17	0.69	
P2–SVM	4.56 ± 1.69	31.10 ± 2.56	0.62	
P3–SVM	5.37 ± 1.72	32.00 ± 2.49	0.64	
MM–SVM (1 1 1...)	5.15 ± 2.91	34.40 ± 2.17	0.69	
	ECOC1			
L–SVM	7.55 ± 3.33	29.70 ± 2.00	59.40	
P2–SVM	4.56 ± 1.86	30.60 ± 1.96	61.20	
P3–SVM	4.95 ± 2.36	29.50 ± 1.96	59.00	
MM–SVM (1 1 2 1 1 1 1)	7.35 ± 3.19	31.60 ± 1.90	63.20	
	ECOC2			
L–SVM	6.74 ± 2.71	33.20 ± 2.35	66.40	
P2–SVM	4.16 ± 2.16	32.20 ± 1.55	64.40	
P3–SVM	5.57 ± 1.37	30.50 ± 2.01	61.00	
MM–SVM (Mixture of 1 & 2)	5.56 ± 2.71	33.80 ± 1.81	67.60	
	ECOC3			
L–SVM	5.14 ± 1.90	36.10 ± 2.77	72.20	
P2–SVM	4.37 ± 1.62	34.60 ± 2.37	69.20	
P3–SVM	4.77 ± 1.46	33.10 ± 1.60	66.20	
MM–SVM (Mixture of 1 & 2)	4.76 ± 2.14	36.50 ± 2.72	73.00	

REFERENCES

1. Alpaydın, E., *Introduction to Machine Learning*, MIT Press, 2004.

2. Vapnik, V. N., *The Nature of Statistical Learning Theory*, Springer, 2000.

3. Cristiannini, N. and J. Shawe-Taylor, *An Introduction to Support Vector Machines*, Cambridge University Press, 2000.

4. Rifkin, R. and A. Klautou, "In Defense of One-Vs-All Classification", *Journal of Maachine Learning Research*, Vol. 5, pp. 101–141, 2004.

5. Dietterich, T. G. and G. Bakiri, "Solving Multi-class Learning Problems via Error-Correcting Output Codes", *Journal of Artificial Intelligence Research*, Vol. 2, pp. 263–186, 1995.

6. Chapelle, O., V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing Multiple Parameters for Support Vector Machines", *Machine Learning*, Vol. 46, No. 1-3, pp. 131–159, 2002.

7. Vapnik, V. N., *Statistical Learning Theory*, John Wiley and Sons, 1998.

8. Burges, C. J. C., "A Tutorial on Support Vector Machines for Pattern Recognition", *Data Mining and Knowledge Discovery*, Vol. 2, No. 2, pp. 121–167, 1998.

9. Bredensteiner, E. J. and K. Bennett, "Multicategory Classification by Support Vector Machines", *Computational Optimizations and Applications*, Vol. 12, pp. 53–79, 1999.

10. Schölkopf, B. and A. J. Smola, *Learning with Kernels*, MIT Press, 2002.

11. Cortes, C. and V. Vapnik, "Support Vector Networks", *Machine Learning*, Vol. 20, pp. 273 – 297, 1995.

12. Rifkin, R. M., *Everything Old Is New Again: A Fresh Look Historical Approaches In Machine Learning*, Ph.D. Thesis, The Sloan School of Management Science, 2002.

13. Allwein, E. R., R. E. Shapire, and Y. Singer, "Reducing Multi-class to Binary: A Unifying Approach for Margin Classifiers", *Journal of Machine Learning Research*, Vol. 1, pp. 113–141, 2000.

14. Hsu, C. and C. Lin, *A Comparison of Methods for Multi-class Support Vector Machines*, Technical Report 2001.19, Department of Computer Science and Information Engineering, National Taiwan University, 2001.

15. Lin, S. and D. J. Costello, *Error Control Coding*, Prentice Hall, 1983.

16. Passerini, A., M. Pontil, and P. Frasconi, "New Results on Error-Correcting Output Codes of Kernel Machines", *IEEE Transactions on Neural Networks*, Vol. 15, No. 1, pp. 45–53, 2004.

17. Masulli, F. and G. Valentini, "Effectiveness of Error-Correcting Output Coding Methods in Ensemble and Monolithic Learning Machines", *Pattern Analysis and Applications*, Vol. 6, No. 4, pp. 285–300, 2003.

18. Crammer, K. and Y. Singer, "On the Learnability and Design of Output Codes for Multi–Class Problems", *Computational Learing Theory*, pp. 35–46, 2000.

19. Weston, J. and C. Watkins, *Multi-class Support Vector Machines*, Technical Report CSD-TR-98-04, Department of Computer Science, University of London, 1998.

20. Joachims, T., *Advances in Kernel Methods - Support Vector Learning*, MIT Press, 1999.

21. Blake, C. and C. Merz, "UCI Repository of Machine Learning Databases", `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 1998.

22. Yıldız, O. T., *Tuning Model Completing Using Cross-Validation in Supervised Learning*, Ph.D. Thesis, Boğaziçi University, 2005.