

Prediction with the SVM using test point margins

Süreyya Özögür- Akyüz¹, Zakria Hussain², and John Shawe-Taylor²

¹ Department of Scientific Computing
Institute of Applied Mathematics
Middle East Technical University
06531, Ankara, Turkey
`sozogur@metu.edu.tr`

² Centre for Computational Statistics and Machine Learning
Department of Computer Science
University College, London
UK, WC1E 6BT
`{z.hussain,jst}@cs.ucl.ac.uk,`

Abstract. Support vector machines (SVMs) carry out binary classification by constructing a maximal margin hyperplane between the two classes of observed (training) examples and then classifying test points according to the half-spaces in which they reside (irrespective of the distances that may exist between the test examples and the hyperplane). Cross-validation involves finding the *one* SVM model together with its optimal parameters that minimizes the training error and has good generalization in the future. In contrast, in this paper we collect *all* of the models found in the model selection phase and make predictions according to the model whose hyperplane achieves the maximum separation from a test point. This directly corresponds to the L_∞ norm for choosing SVM models at the testing stage. Furthermore, we also investigate other more general techniques corresponding to different L_p norms and show how these methods allow us to avoid the complex and time-consuming paradigm of cross-validation. Experimental results demonstrate this advantage, showing significant decreases in computational time as well as competitive generalization error.

1 Introduction

Data mining is the process of analysing data to gather useful information or structure. Computationally it is a highly demanding area because of the large amount of experimental data in databases. Various applications of data mining are prevalent in areas such as medicine, finance, business and so forth. There are different types of data mining tools such as statistical analysis, probabilistic methods and learning theory.

For the majority of data mining tools, parameter selection is a critical question and attempts at determining the right model for data analysis and prediction. In this research, we develop a fast algorithm for model selection that

uses the benefit of all the models constructed during the parameter selection stage. We apply our model selection strategy to a *maximum margin* algorithm called the Support Vector Machines (SVMs), which is one of the most powerful methods in machine learning for solving binary classification problems. SVMs were invented by Vapnik [7] (and co-workers), with the idea to classify points by maximizing the distance between two classes [1].

In recent years, learning methods have become more desirable because of their reliability and effectiveness at solving real world problems. In real world situations, for instance in the engineering or biological sciences, conducting experiments can be costly and time consuming. In such situations, accurate predictive methods can be used to help overcome these difficulties in more efficient and cost effective ways. Furthermore, large amounts of information are freely accessible on the internet, containing large data sets. Therefore, it is important to understand and analyze these data sets to make them beneficial. Different methodologies have been developed to tackle learning, including supervised and unsupervised learning.

Supervised learning is a learning methodology where unseen data (test data) can be predicted with the help of observations. Algorithms learn functions based on training examples consisting of input-output pairs given to the learning system. These functions can subsequently be used to predict the output of test examples. Training sets are the main resource of supervised learning.

Let (\mathbf{x}, y) be an (input,output) pair where $\mathbf{x} \in \mathbb{R}^n$ and $y \in \{-1, 1\}$. We say that \mathbf{x} comes from some input domain X and similarly that y comes from some output domain Y . We define a training set that contains ℓ input-output pairs by $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{\ell}$. Given S and a set of functions \mathcal{F} we would like to find a candidate function $f \in \mathcal{F}$ such that

$$f : \text{input space} \mapsto \text{output domain}.$$

We refer to these candidate functions as *hypotheses* [2].

Binary classification is frequently performed using linear classification methods when the output domain Y consists of two labels $\{-1, 1\}$, and the function $f : X \rightarrow \{-1, 1\}$ maps examples to a two dimensional label space. A more rigorous definition follows.

Definition 1. Let f be a real-valued function defined on a subset $X \subseteq \mathbb{R}^n$,

$$f : X \longrightarrow \mathbb{R}.$$

Then, $\mathbf{x} = (x_1, x_2, \dots, x_n)^\top$ is assigned to the positive class if $f(\mathbf{x}) \geq 0$, otherwise it is assigned to the negative class.

In this study, a function $f(\mathbf{x})$ is affinely linear if it can be expressed as

$$\begin{aligned} f(\mathbf{x}) &= \langle \mathbf{w}, \mathbf{x} \rangle + b \\ &= \sum_{i=1}^n w_i x_i + b, \end{aligned}$$

where $\langle \cdot, \cdot \rangle$ denotes the scalar product and (\mathbf{w}, b) consists of the parameters that control the function and decision rule given by $\text{sgn}(f(\mathbf{x}))$. Here, $\mathbf{w} = (w_1, \dots, w_n)^\top \in \mathbb{R}^n$ is referred to as the *weight vector* and $b \in \mathbb{R}$ the *bias*.

In linear *binary classification*, the two classes are discriminated by a hyperplane defined by $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$. The distance of the hyperplane to some point $\mathbf{x} \in X$ is known as a *margin*.

Definition 2. We define the (functional) margin of the examples (\mathbf{x}_i, y_i) ($i = 1, 2, \dots, \ell$) with respect to a hyperplane (\mathbf{w}, b) to be the quantity

$$\gamma_i := y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \quad (i = 1, 2, \dots, \ell).$$

Note that if $\gamma_i > 0$, then correct classification is achieved.

Definition 3. The geometric margin is defined as the distance between each of the points nearest to the hyperplane and the hyperplane. The value of the geometric margin is ρ , where ρ is the functional margin [2].

For real world problems, data in the input space may not be linearly separable, in which case non-linear classifiers are needed. Multi-layer perceptrons were introduced to serve this purpose but suffered from slow training and local minima in weight (input) space. Kernel methods use a different approach that increases the flexibility of linear functions by applying a non-linear mapping ϕ from the input space into a higher dimensional vector space called the *feature space*. Then f and \mathbf{w} can be rewritten as:

$$f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b, \quad (1)$$

where the weight vector,

$$\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i y_i \phi(\mathbf{x}_i), \quad (2)$$

can be written as a linear combination of the training examples in feature space. In fact, we can compute f without the explicit feature vectors $\phi(\mathbf{x})$ if we have a direct method for computing $\kappa(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$, referred to as the *kernel function* [2].

Support vector machines choose the linear classifier that maximizes the geometrical margin over the training data. Since rescaling of (\mathbf{w}, b) does not change classification, we can enforce

$$y_i \cdot (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \quad (i = 1, \dots, \ell), \quad (3)$$

where the functional margin is 1 and the geometric margin is $\frac{1}{\|\mathbf{w}\|_2}$. Hence, to maximize the margin, it is necessary to minimize $\|\mathbf{w}\|_2^2$ (where $\|\cdot\|_2$ denotes the Euclidean norm).

Hard Margin Classifier Problem Given the above definition we have a convex optimization problem where we can look for the hard margin (optimal) classifier in the following form:

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \|\mathbf{w}\|_2^2 \\ \text{such that} \quad & y_i \cdot (\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle + b) \geq 1 \quad (i = 1, 2, \dots, \ell). \end{aligned}$$

In general, the *dual form* of the problem is preferred and it is found with the help of optimization theory. Using the *Lagrangian* technique and its partial derivatives, with the *Karush-Kuhn-Tucker (KKT) conditions*, the dual problem is given in the following form:

$$\begin{aligned} \max \quad & \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} y_i y_j \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \\ \text{Subject to} \quad & \sum_{i=1}^{\ell} y_i \alpha_i = 0, \\ & \alpha_i \geq 0 \quad (i = 1, 2, \dots, \ell). \end{aligned}$$

To solve complex classification problems (not perfectly linearly separable), it is not enough to apply hard margin maximal margin classifiers, since they will not be applicable to real world data. Therefore, slack variables (that allow errors) are introduced to permit the maximal margin criterion to be violated. This is known as a *soft margin classifier*.

Soft Margin Classifier Problem Given a vector $\boldsymbol{\xi} = (\xi_1, \dots, \xi_\ell)^\top$ known as a slack variable and adding it to the objective function results in the following soft margin optimization problem:

$$\begin{aligned} \min_{\boldsymbol{\xi}, \mathbf{w}, b} \quad & \|\mathbf{w}\|_2^2 + C \sum_i \xi_i \\ \text{Subject to} \quad & y_i \cdot (\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle + b) \geq 1 - \xi_i \quad (i = 1, 2, \dots, \ell). \end{aligned}$$

The use of different regularization parameters for positive and negative examples becomes important when there are significant imbalances between the number of positive and negative training examples.

The corresponding dual form can be constructed in the same way, by setting the gradient of the Lagrange function equal to zero and writing the KKT conditions. Hence, one is faced with the following dual optimization problem:

$$\begin{aligned} \max \quad & \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} y_i y_j \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j), \\ \text{Subject to} \quad & \sum_{i=1}^{\ell} y_i \alpha_i = 0, \\ & 0 \leq \alpha_i \leq C, \quad (i = 1, 2, \dots, \ell). \end{aligned}$$

The solution of this optimization problem yields a maximal margin hyperplane that we will refer to as the *Support Vector Machine (SVM)*.

All machine learning algorithms require a model selection phase. This consists of choosing the best parameters for a particular data set and using them in order

to make predictions. In the SVM (or ν -SVM) that uses a Gaussian kernel the number of parameters to tune is two – the C in the standard SVM (or the ν in the ν -SVM) and the kernel width parameter σ . Lets take the standard SVM and look a little closely at the model selection phase. Firstly, given some data set S the most common model selection technique is to use k -fold cross validation where $k > 0, k \in \mathbb{N}$. The idea is to split the data into k parts and then use $k - 1$ for training and the remaining for testing. The $k - 1$ folds are trained with various values of C and σ and tested on each test set. The set of values that give the smallest test error amongst all of the splits is used as the SVM model for the entire training set S .

In this paper we assume that the data sets consist of a small number of examples and applying cross-validation is costly as we will tend to use up a large proportion of points in the test set. We tackle this problem by using the full training set to construct all possible SVM models that can be defined using the list of parameter values. We classify a test point by checking to see which SVM hyperplane (from the full list of models) the test point is furthest from. Rather than re-training the SVM using the best C and best σ value we simply store all of the SVM models and make predictions using any one of them. This speeds up the computational time of model selection when compared to the cross-validation model selection regime described above. The intuition of choosing parameters from the test phase is based on the studies of [5] and [6]. In [5], biological data is classified according to the output values defined with confidence levels with different classifiers being determined for each protein sequence. The second motivation for the work comes from the theoretical work of [6] that gives generalization error bounds on test points given that they achieve a large separation from the hyperplane. This suggests that we can make predictions once we receive test points, from the hyperplanes already constructed and giving us a way of avoiding cross validation. The following section describes this methodology.

2 Methods

In this section, three different norms will be discussed for model selection at the testing phase. Given a set of functions $\{f_1(\mathbf{x}), \dots, f_\ell(\mathbf{x})\}$ output by the SVM with $\ell = |C| \times |\sigma|$ being the number of models that can be constructed from the set of parameter values $C = \{C_1, \dots\}$ and $\sigma = \{\sigma_1, \dots\}$ we can use some or a combination of them to make predictions. The first approach we propose uses the L_∞ norm for choosing which function to use. This is equivalent to evaluating the distance of a test point according to the function that achieves the largest (functional) margin. For example, assume we have three values for $C = \{C_1, C_2, C_3\}$ and two values for $\sigma = \{\sigma_1, \sigma_2\}$, respectively. Therefore, we have the following $\ell = 6$ SVM models together with their list of parameter values $\{C, \sigma\}$:

$$- f_1 = \text{SVM}_1: \{C_1, \sigma_1\}$$

- $f_2 = \text{SVM}_2: \{C_1, \sigma_2\}$
- $f_3 = \text{SVM}_3: \{C_2, \sigma_1\}$
- $f_4 = \text{SVM}_4: \{C_2, \sigma_2\}$
- $f_5 = \text{SVM}_5: \{C_3, \sigma_1\}$
- $f_6 = \text{SVM}_6: \{C_4, \sigma_4\}$

Now at evaluation we would compute the functions for all test points. For instance given a test example $\mathbf{x} \in X_{test}$, let us assume the following six functional values,

- $f_1(\mathbf{x}) = 1.67$
- $f_2(\mathbf{x}) = 0.89$
- $f_3(\mathbf{x}) = -0.32$
- $f_4(\mathbf{x}) = -0.05$
- $f_5(\mathbf{x}) = 1.1$
- $f_6(\mathbf{x}) = 1.8$

We assume here, without loss of generality, that the functions f compute the functional margins and not the geometrical margins (hence the reason that the example values we have presented are not bounded by 1 and -1). Finally we would predict the class of \mathbf{x} by looking for the maximum positive and the maximum negative value of all functions. This corresponds to f_6 and f_3 . However, the distance of the test example \mathbf{x} from the hyperplane is greater for the $f_6 = \text{SVM}_6$ function/model and therefore this example can be predicted as positive. Therefore, the L_∞ prediction function $F_\infty(\mathbf{x})$ given an example \mathbf{x} can be expressed in the following way,

$$F_\infty(\mathbf{x}) = \text{sgn} \left(\max\{f_i(\mathbf{x})\}_{i=1}^\ell + \min\{f_i(\mathbf{x})\}_{i=1}^\ell \right), \quad (4)$$

where $F = \{f_1, \dots, f_\ell\}$ is the set of all the functions that can be constructed from the list of parameter values.

The L_∞ norm approach is also illustrated in Fig. 1 on a real world data set. The figure gives the evaluations of 110 SVM models (10 C values and 11 σ values) for a particular test point. The plot on the left of Fig. 1 are the functional margin values for a test point given the 110 different parameter values and sorted in ascending order. We can see from the plot on the right that the maximum positive margin and minimum negative margin are the left most and right most bars (in red), respectively, and the sign of the sum of these two values will give us the prediction of the test point. This test point is classified positive.

The second approach we introduce is for the L_1 norm where the decision depends on the sign of the Riemann sum of all outputs evaluated for a test point. For example, looking at the example we gave above we can see that summing the positive function values $1.67 + 0.89 + 1.1 + 1.8$ for the set of functions $\{f_1, f_2, f_5, f_6\}$ and the negative function values $-0.32 + -0.05$ for $\{f_3, f_4\}$ will give an overall positive value of 5.46 and a negative value of -0.37 , respectively.

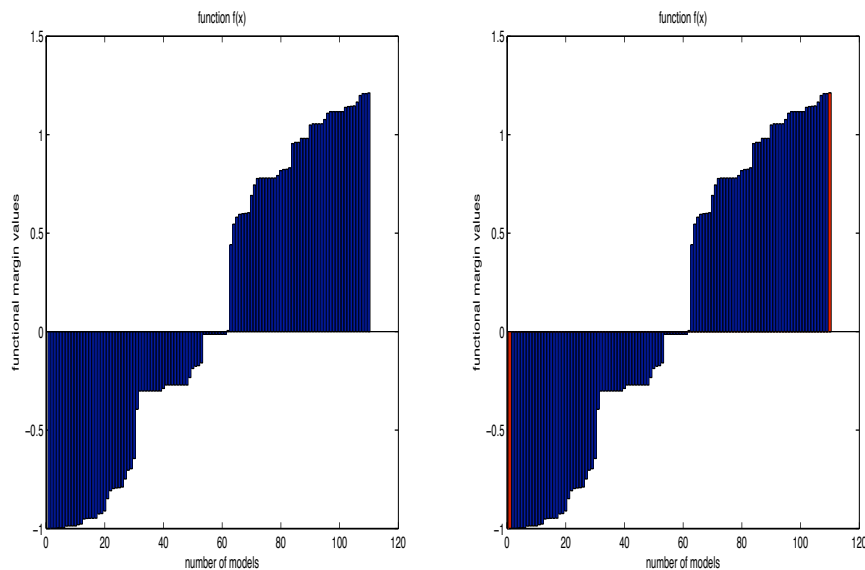


Fig. 1. Each stem corresponds to the functional margin value given for that particular SVM model f . Right hand side is the graph of L_∞ norm which predicts the example as +1 where actual class is +1

Clearly, the sign of the sum of these two values will result in a positive classification for the test point \mathbf{x} . Generalizing this example we can have the following L_1 norm prediction function $F_1(\mathbf{x})$ given a test example \mathbf{x} ,

$$F_1(\mathbf{x}) = \text{sgn} \left(\sum_{i=1}^{\ell} f_i(\mathbf{x}) \right). \quad (5)$$

This is illustrated on the left hand side of Figure 2. It is clear that the prediction function looks at the integrals of the two areas (indicated in red) above and below the threshold of 0. Essentially this equates to summing the above and below stems. In Fig. 2, it is clear that the summation will be positive since the area of the positive values (above 0) is bigger than the area of the negative values (below 0). This methodology corresponds to summing the weighted average of all the prediction functions with a uniform weighting of 1.

The final approach corresponds to the L_2 norm and is similar to the L_1 norm discussed above, but with a down-weighting if values are below 1 and an up-weighting if they are above 1. This means that we are giving a greater confidence to functions that predict functional values greater than 1 or -1 but less confidence to those that are closer to the threshold of 0. Another way of thinking about this approach is that it is equivalent to a weighted combination of functional margins with the absolute values of themselves. Once again, using the example

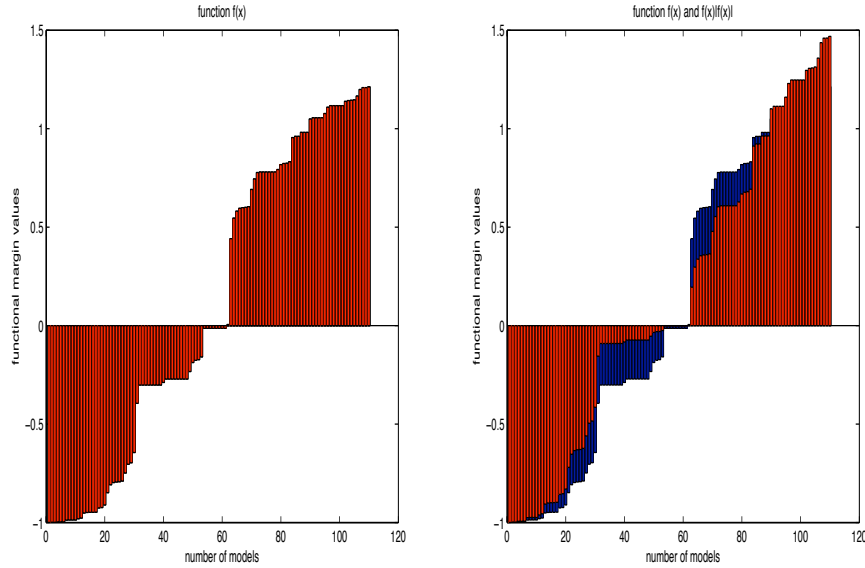


Fig. 2. Each stem corresponds to the functional margin value given for that particular SVM model f . The L_1 and L_2 norms both predict +1. The actual class of each example is +1.

from above, and weighting the summation of the positive function values with their absolute values gives $(1.67 \times 1.67) + (0.89 \times 0.89) + (1.1 \times 1.1) + (1.8 \times 1.8)$ for the positive functions $\{f_1, f_2, f_5, f_6\}$ and $(-0.32 \times 0.32) + (-0.05 \times 0.05)$ for the negative functions $\{f_3, f_4\}$, respectively. These two summations will give an overall positive value of 8.03 and a negative value of -0.1049 , respectively, and the summing of these two values will give a prediction of the test example as belonging to the positive class. Therefore, given a test example \mathbf{x} , we have the following L_2 norm prediction function $F_2(\mathbf{x})$,

$$F_2(\mathbf{x}) = \text{sgn} \left(\sum_{i=1}^{\ell} f_i(\mathbf{x}) |f_i(\mathbf{x})| \right). \quad (6)$$

The right hand side plot of Figure 2 represents the L_2 norm solution for the same test point predicted by the L_∞ norm in Fig. 1 and the L_1 norm method shown on the left hand side of Fig. 2. As you can see the blue region corresponds to the original values of the functions and the red stems are the down-weighted or up-weighted values of the 110 prediction functions. The L_2 norm corresponds to summing the weights of the red stems only. It can be seen that the values that are smaller than 1 are down-weighted (decreased) and those greater than 1 are up-weighted (increased). Clearly values that are close to 1 do not change significantly.

3 Data Set Description

In this study, we used the well known standard UCI machine learning repository (can be accessed via <http://archive.ics.uci.edu/ml/>). From the repository, we used the Votes, Glass, Haberman, Bupa, Credit, Pima, BreastW , Ionosphere, Australian Credit and the German Credit data sets. For the first seven data sets we removed examples containing unknown values and contradictory labels (this is why the Votes data set is considerably smaller than the one found at the UCI website). The number of examples, attributes and class distributions of all the data sets are given in Table 1.

Data set	# instances	# attributes	# pos	# neg
Voes	52	16	18	34
Glass	163	9	87	76
Haberman	294	3	219	75
Bupa	345	6	145	200
Credit	653	15	296	357
Pima	768	8	269	499
BreastW	683	9	239	444
Ionosphere	351	34	225	126
Australian	690	14	307	383
German	1000	20	300	700

Table 1. Data set description

4 Results

We call our methods the SVM- L_∞ , SVM- L_1 and SVM- L_2 which corresponds to using the L_∞ , L_1 and L_2 methods we had proposed in Section 2. We also test our methods against the SVM with cross-validation (CV), where we carry out 10-fold cross-validation and a further 10 folds to estimate the optimal C and σ values. Note that in the methods we propose we do not need to carry out this parameter tuning phase and hence achieve a 10 fold speed-up against the SVM with CV.

Table 2 presents the results and we report the standard deviation (STD) of the error over the 10-folds of cross-validation, the cumulative training and testing time (time) in seconds for all folds of CV, and the error as percentages (error %) and as numbers (error #) for the entire 10-fold cross-validation process.

The results of the SVM- L_p where $p = \infty, 2, 1$ shows a significant decrease in computational time when compared to the SVM with CV. For example, we can see that the German data set takes approximately 4368 seconds to train and test and that our methods take between 544 and 597 seconds for training

and testing purposes. This is approximately 8 times faster than using cross-validation. We can also see from Table 2 that the L_∞ method seems to capture better prediction models compared to the other two L_p norm methods, but all three methods compare favourably with respect to test error against the SVM with CV.

Finally, when comparing the three methods proposed it is clear that the most successful in terms of speed and accuracy is the L_∞ norm. This perhaps is less surprising when viewed from the theoretical motivation of this work, as [6] has proposed a bound that gives higher confidence of correct classification if the test point achieves a large separation from the hyperplane. This is exactly what the L_∞ norm method does.

5 Discussion and Future Work

We proposed a novel method for carrying out predictions with the SVM classifiers once they had been constructed using the entire list of regularization parameters (chosen by the user). We showed that we could apply the L_p norms to help pick these classifier(s). Moreover, we proposed the SVM- L_∞ , SVM- L_1 and SVM- L_2 strategies and discussed their attributes with a toy and real world example. We showed that the L_∞ method would choose a single classifier for prediction, the one that maximally maximized the distance of a test point from its hyperplane. The L_1 and L_2 norms were similar to each other and gave predictions using a (weighted) sum of the prediction functions constructed by each SVM function. Finally, in Section 4 we gave experimental results that elucidated the methods described in this paper.

The main purpose of the work proposed is to overcome situations where we have a very large or a very small number of examples. Having a small number of examples means that examples are scarce and using them for testing in a cross-validation split is costly. Also, having a large number of examples implies very long running times. We overcome these problems by avoiding the cross-validation process used for finding the best regularization parameters. Removing this CV dependency for finding parameters greatly improves training and testing time for the SVM algorithm.

A future research direction would be to use other methods for choosing the classifiers at testing. Perhaps, a convex combination of the functions would yield better generalization capabilities. Such a combination of functions could be weighted by a factor in the following way,

$$F(\mathbf{x}) = \operatorname{sgn} \left(\sum_{i=1}^{\ell} \beta_i f_i(\mathbf{x}) \right) \tag{7}$$

$$\text{s.t.} \quad \sum_i^{\ell} \beta_i = 1$$

where $\beta_i \in \mathbb{R}$.

Data Set	SVM with CV			SVM- L_∞			SVM- L_1			SVM- L_2		
	STD	time	err % err #	STD	time	err % err #	STD	time	err % err #	STD	time	err % err #
Votes	0.2115	11.54	8.33 5	0.0991	0.94	8.33 5	0.095	1.84	27 14	0.0979	0.9	19.17 10
Glass	0.1222	89.93	34.85 57	0.135	16.29	31.99 52	0.1125	13.28	39.27 64	0.1282	9.59	36.92 60
Haberman	0.0437	169.51	24.81 73	0.0363	23.18	25.17 74	0.0127	23.44	25.49 75	0.0133	13.62	25.83 76
Bupa	0.0648	354.46	28.95 100	0.0611	82.87	31.55 109	0.0089	80.59	42.02 145	0.0089	80.59	42.02 145
Credit	0.1916	1812.86	13.49 88	0.1439	370.21	18.09 118	0.0984	245.43	18.52 121	0.1036	199.31	18.22 119
Pima	0.038	1300.82	23.81 183	0.03	265.9	26.17 201	0.0101	183.89	34.64 266	0.0101	183.89	34.64 266
BreastW	0.0192	414.18	3.35 23	0.0361	111.26	4.81 33	0.0421	72.35	4.8 33	0.0363	105.88	3.78 26
Ionosphere	0.0568	172.21	6.21 22	0.0512	76.61	8.19 29	0.0694	54.91	14.16 50	0.0652	96.96	11.62 41
australian	0.0416	1868.16	0.14 102	0.0333	223.33	0.14 103	0.0195	225.98	0.17 119	0.0186	236.45	0.16 114
german	0.0454	4378.01	0.23 238	0.0378	544.93	0.26 266	0.0084	569.33	0.29 296	0.0097	596.79	0.2860 286

Table 2. L_∞ , L_1 and L_2 norm results against SVM with Cross-Validation

Finally, we believe that tighter margin based bounds would help to improve the selection of the SVM functions at testing. The bound proposed by [6] suggests the L_∞ method we proposed in this paper. However, from the results section it is clear that this does not always create smaller generalization error than the SVM with CV. Therefore, a future research direction is to use a tighter bounding principle for the margin based bound of [6], such as a PAC-Bayes analysis (due to [4], but also extended to margins by [3]). Therefore, we could use the bounds to indicate which classifiers to use at testing. We believe that a tighter estimate of the bounds would yield better generalization.

References

1. B.E. Boser, I.M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *In Fifth Annual Workshop on Computational Learning Theory*, ACM., pages 144–152, Pittsburgh, 1992. ACM.
2. N. Cristianini and J. Shawe-Taylor. *An introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.
3. J. Langford and J. Shawe-Taylor. PAC bayes and margins. In *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2003. MIT Press.
4. D.A. McAllester. Some pac-bayesian theorems. *Machine Learning*, 37(3):355–363, 1999.
5. S. Özöğür, J. Shawe-Taylor, G.-W. Weber, and Z.B. Ögel. Pattern analysis for the prediction of fungal pro-peptide cleavage sites. *to appear in special issue of Discrete Applied Mathematics on Networks in Computational Biology*, 2007.
6. J. Shawe-Taylor. Classification accuracy based on observed margin. *Algorithmica*, 22:157–172, 1998.
7. V. N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, 1998.