LETTER Communicated by Joseph Yue Wang

Arbitrary Norm Support Vector Machines

Kaizhu Huang
kzhuang@cse.cuhk.edu.hk
Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, Hong Kong

Danian Zheng
danianzheng@cn.fujitsu.com
Information Technology Laboratory, Fujitsu Research and Development Center, Beijing 100025, China

Irwin King
king@cse.cuhk.edu.hk
Michael R. Lyu
lyu@cse.cuhk.edu.hk
Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, Hong Kong

Support vector machines (SVM) are state-of-the-art classifiers. Typically L_2-norm or L_1-norm is adopted as a regularization term in SVMs, while other norm-based SVMs, for example, the L_0-norm SVM or even the L_∞-norm SVM, are rarely seen in the literature. The major reason is that L_0-norm describes a discontinuous and nonconvex term, leading to a combinatorially NP-hard optimization problem. In this letter, motivated by Bayesian learning, we propose a novel framework that can implement arbitrary norm-based SVMs in polynomial time. One significant feature of this framework is that only a sequence of sequential minimal optimization problems needs to be solved, thus making it practical in many real applications. The proposed framework is important in the sense that Bayesian priors can be efficiently plugged into most learning methods without knowing the explicit form. Hence, this builds a connection between Bayesian learning and the kernel machines. We derive the theoretical framework, demonstrate how our approach works on the L_0-norm SVM as a typical example, and perform a series of experiments to validate its advantages. Experimental results on nine benchmark data sets are very encouraging. The implemented L_0-norm is competitive with or even better than the standard L_2-norm SVM in terms of accuracy but with a reduced number of support vectors, −9.46% of the number on average. When compared with another sparse model, the relevance vector machine, our proposed algorithm also demonstrates better sparse properties with a training speed over seven times faster.

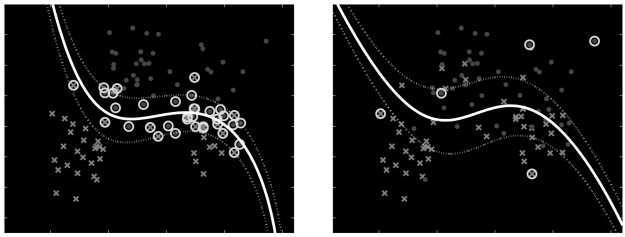
1 Introduction

As the state-of-the-art learning algorithms, support vector machines (SVM) (Vapnik, 2000) have been widely studied and applied in machine learning, pattern recognition, and computer vision. The standard SVM usually adopts a term of L_2-norm or L_1-norm to control the structure complexity, while other norms (e.g., L_0-norm), are rarely seen in the literature. The L_0-norm, defined as the number of nonzero elements in a vector, is discontinuous and nonconvex, resulting in an NP-hard optimization in general. However, it enjoys a significant feature: it is an ideal approach for enforcing sparsity without losing classification performance. This is especially useful for SVMs, since the standard SVM often generates too many support vectors (SV). Because the prediction speed is exclusively dependent on the number of SVs, having too many SVs leads to a very slow classification speed. Figure 1a illustrates the decision boundary of the L_2-norm SVM and its associated SVs (the samples circled by $\circ$'s). According to SVM theory, the misclassified data samples and the ones located within the margins are support vectors. This generates many irrelevant SVs, as observed in Figure 1a. Moreover, since the prediction function is a linear combination of the kernel functions involving only SVs, more SVs lead to more calculation time when classifying a new data point. The slow classification speed is one of the shortcomings for the L_2-norm SVM.

In this letter, inspired by the notion of Bayesian learning, we propose a novel framework that can achieve arbitrary norm SVMs (especially the L_0-norm SVM) in polynomial time. By defining hierarchical priors, we prove an asymptotic equivalence between Bayesian learning and L_p-norm ($p = 0, 1$) optimization. We then extend the theory to arbitrary norm SVMs. Although our framework starts from Bayesian learning, it goes beyond it because no hierarchy priors need to be defined explicitly as required by Bayesian learning. Hence, this provides a convenient way for incorporating Bayesian concepts into most kernel machines (Vapnik, 2000; Huang, Yang, King, Lyu, & Chan, 2004; Huang, Yang, King, & Lyu, 2008).

One significant feature of the proposed framework is that the optimization involved can be implemented by solving a sequence of sequential minimal optimization (SMO) problems (Platt, 1998; Keerthi, Shevade, Bhattacharyya, & Murthy, 2001). Hence, both the space complexity and the time complexity are small, making the framework highly practical in many applications.

As a typical and important application, we demonstrate how our approach works for the L_0-norm SVM, which is generally regarded as an NP-hard problem. Theoretical and empirical evidence show that the proposed L_0-norm SVM is competitive with or even better than the standard L_2-norm SVM in terms of accuracy, but with significantly fewer support vectors. In other words, the L_0-norm SVM attains a much sparser classifier with performance maintained or slightly increased. Figure 1b illustrates



(a) L_2-norm SVM (b) Proposed L_0-norm SVM

Figure 1: An illustration of the L_2-norm SVM and the proposed L_0-norm SVM. $\times$'s and $\bullet$'s represents two types of data. Samples circled by $\circ$'s are the support vectors. The solid lines represent the decision boundaries (when a polynomial kernel is exploited), and the two dashed lines in each figure show the upper and lower margins ($f(\mathbf{x}) = \pm 1$).

one example of our proposed L_0-norm SVM. As observed in the figure, the boundary given by the L_0-norm SVM is almost the same as the one given by the L_2-norm SVM, but with many fewer SVs. Hence, this greatly reduces the calculation time involved in the predication. In the literature, the relevance vector machine (RVM; Tipping, 2000, 2001) has been proposed as an extremely sparse model for both regression and classification. As we show in the experiments, the proposed L_0-norm model achieves comparable sparseness with RVM, but with a training speed seven times faster on average.

The rest of the letter is organized as follows. In the next section, we present our notations as well as the baseline model, the L_2-norm SVM. In section 3, we introduce the framework of the arbitrary norm support vector machines. Bayesian motivations, theoretical derivations, and an implementation of the L_0-norm SVM are introduced in this section. In section 4, we present experimental results to validate the effectiveness of our framework. In section 5, we provide a literature review and describe the connections of our work with other approaches. Finally, we set out the conclusion and discuss future work in section 6.

2 Notations

We first provide the notations used in the letter. We then briefly introduce the L_2-norm SVM as the baseline algorithm.

Suppose we are given empirical data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ with the input pattern $\mathbf{x}_i \in \mathcal{X}$ and the output label $y_i \in \{\pm 1\}$.[1] Kernel methods such as SVMs try

[1] In this letter, we consider only binary classification. Multiclass classification can be easily approached by using a one versus one or one versus others strategy.

to find a linear hyperplane $f(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}) + b$ to separate the positive from the negative examples as robustly as possible, where $\Phi(\cdot) : \mathcal{X} \to \mathcal{F}$ denotes a nonlinear mapping from the input space $\mathcal{X}$ into a higher-dimensional feature space $\mathcal{F}$, the weight vector $\mathbf{w} \in \mathcal{F}$, and the bias term $b \in \mathbb{R}$. To construct this optimal hyperplane, the L_2-norm SVM usually needs to solve the following primal problem:

$$\begin{aligned} &\min \ \frac{1}{2}\mathbf{w}^T\mathbf{w} + C_\xi \mathbf{1}^T \boldsymbol{\xi} \\ &\text{s.t.} \ \ y_i(\mathbf{w}^T \Phi(\mathbf{x}_i) + b) - 1 + \xi_i \geq 0, \xi_i \geq 0, \forall i, \end{aligned} \tag{2.1}$$

where ξ_i are slack variables, $\mathbf{1}$ is a vector with all elements as 1, $\boldsymbol{\xi}$ is a vector with elements ξ_i, and C_ξ is a trade-off constant between the margin $\frac{1}{2}\mathbf{w}^T\mathbf{w}$ and the empirical error. In fact, this primal optimization problem can be equivalently transformed into the dual problem:

$$\begin{aligned} &\min \ \frac{1}{2}\boldsymbol{\alpha}^T \mathbf{H}\boldsymbol{\alpha} - \mathbf{1}^T\boldsymbol{\alpha} \\ &\text{s.t.} \ \ \mathbf{y}^T\boldsymbol{\alpha} = 0, \quad 0 \leq \alpha_i \leq C_\xi, \forall i, \end{aligned} \tag{2.2}$$

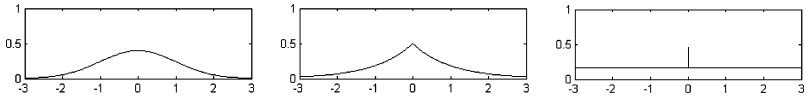
where $\mathbf{H}$ denotes a symmetric matrix with elements $h_{ij} = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$, $k(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$ defines a kernel function, $\boldsymbol{\alpha}$ describes a vector with elements α_i, and $\mathbf{y}$ is a vector with elements y_i. The decision surface then takes the form

$$f(\mathbf{x}) = \sum_{i=1}^{l} y_i \alpha_i k(\mathbf{x}, \mathbf{x}_i) + b. \tag{2.3}$$

Because the coefficients α_i are Lagrange multipliers, many of them will be equal to zero in the final solution. The patterns $\mathbf{x}_i$ associated with those nonzero coefficients α_i are so-called support vectors. The time taken for equation 2.3 to predict a class label of a new pattern is proportional to the number of support vectors. All training patterns lying in the margin zone $f(\mathbf{x}) \pm 1$, and any patterns outside the margin zone but wrongly classified, are support vectors. Hence, the support vectors of SVM are often redundant, especially for a large, nonseparable training set. This will also be observed in the example shown in section 4.1. To speed up the classification process, it is necessary to reduce the number of support vectors. A concrete implementation of our novel framework, presented in the next section, will conquer this problem effectively.

3 Proposed Framework

In this section, we present the framework of our arbitrary norm support vector machines. We start from the Bayesian learning approach, which aims



(a) Gaussian prior (b) Laplacian prior (c) Pulse prior

Figure 2: The prior assumptions $p(\alpha_i)$ for each coefficient α_i.

to achieve sparsity for classifiers. We then justify the proposed framework and present several important propositions. Following that, we demonstrate how to use the framework to attain the L_0-norm SVM as a typical and significant example. Issues such as optimization and complexity are also discussed in this section.

3.1 L_2-Norm, L_1-Norm, and Bayesian Learning. L_2-norm or L_1-norm is often employed as a regularization term in order to control model complexity. For example, the standard SVM uses the L_2-norm $\|\mathbf{w}\|_2^2 = \alpha^T \mathbf{H}\alpha$, which has an interpretation of margin maximization for classification. In kernel ridge regression (Hoerl & Kennard, 1970; Saunders, Gammerman, & Vovk, 1998) and radial basis function (RBF) neural networks, the regularization term is also an L_2-norm $\|\boldsymbol{\alpha}\|_2^2 = \sum |\alpha_i|^2$. L_1-norm $\|\alpha\|_1^1 = \sum |\alpha_i|$ is also widely used. One appealing feature of L_1-norm is that it can force sparsity, for example, in the L_1-norm SVM (Zhu, Rosset, Hastie, & Tibshirani, 2003) or sparse coding algorithms (Olshausen & Field, 1997).

Using norm-based terms as the regularization has a Bayesian interpretation. More specifically, L_2-norm actually assumes a gaussian prior probability over α, while L_1-norm assumes a Laplacian prior probability. This can be readily observed if we treat the regularization term as the log of the probability for α, namely,

$$-\log p(\boldsymbol{\alpha}) = \begin{cases} \boldsymbol{\alpha}^T \mathbf{A}^{-1} \boldsymbol{\alpha}, & \text{if } p(\boldsymbol{\alpha}) = \mathcal{N}(0, \mathbf{A}) \\ \sum |\alpha_i|, & \text{if } p(\boldsymbol{\alpha}) = \prod \exp(|\alpha_i|). \end{cases}$$

Hence, a zero-mean and unit-variance gaussian prior over $\boldsymbol{\alpha}$ is actually assumed for kernel ridge regression and RBF neural networks, while a Laplacian prior over the coefficient is enforced in linear programming regression and sparse coding.

Unfortunately, the gaussian prior or L_2-norm generates too many coefficients close to zero but not exactly equal to zero. Hence, it yields a nonsparse model. The independent Laplacian prior is well known for its ability to make many coefficients α_i exactly equal to zero. However, the resulting model is still not sparse enough in practice. The gaussian prior and the Laplacian priors are plotted in Figures 2a and 2b. Ideally, L_0-norm $\|\boldsymbol{\alpha}\|_0^0 = \lim_{n \to +\infty} \sum |\alpha_i|^{1/n} = \sum I_{\{\alpha_i \neq 0\}}$ could be the most suitable

form for inducing the sparest solution, where I is an indicator function; that is, if $\alpha_i \in \{\alpha_i \neq 0\}$, then $I = 1$; otherwise, $I = 0$. Similarly, the corresponding prior for L_0-norm is the so-called pulse prior defined as $p(\boldsymbol{\alpha}) \propto \exp(-I_{\{\alpha_i \neq 0\}}(\alpha_i))$, which is plotted in Figure 2c. Unfortunately, it is difficult to optimize the L_0-norm due to the discontinuity and nonconvexity involved. In this letter, we show that L_0-norm can be asymptotically achieved by applying a hierarchical Bayesian model. Hence, the proposed model can attain a very sparse solution.

Remark. Enforcing sparsity has two advantages. First, the greater the number of the coefficients α_i equal to zero, the faster the calculation required for predicting a new data point, since the decision function is given as $f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i k(\mathbf{x}, \mathbf{x}_i) + b$. Second, in kernel classifiers, generalization performance increases with the degree of the sparsity (Figueiredo, 2002). It is recognized in many fields that a simple structure is more resistant to over fitting.

3.2 Hierarchical Bayesian Models, L_1-Norm, and L_0-Norm. In this section, we show that hierarchical Bayesian models can asymptotically achieve L_1-norm and L_0-norm. This fact therefore motivates the proposed framework.

The Bayesian approaches often treat the output z of the learned linear classifier as corrupted by a zero-mean and unit-variance variable w; $z(\mathbf{x}, \boldsymbol{\alpha}) = \boldsymbol{\alpha}^T \mathbf{h}(\mathbf{x}) + w$, where $\mathbf{h}(\mathbf{x})$ is defined as $[1, k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_l)]^T$, and b is incorporated in $\boldsymbol{\alpha}$. Given the training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, we could simply write the gaussian noise-corrupted formula as a matrix form $\mathbf{z} = \mathbf{H}\alpha + \mathbf{w}$, where $\mathbf{H}$ is defined as $[\mathbf{h}^T(\mathbf{x}_1), \ldots, \mathbf{h}^T(\mathbf{x}_l)]$ and $\mathbf{w}$ is a vector with each element as a zero-mean and unit-variance gaussian variable. If $\mathbf{z}$ is treated as missing variables, the expectation-maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977; Neal & Hinton, 1999) can be used to find the maximum a posteriori (MAP) $\boldsymbol{\alpha}$, provided that a prior of $\boldsymbol{\alpha}$ is given.

RVM (Tipping, 2000, 2001) assumes the prior of $\boldsymbol{\alpha}$ as the two-level Bayesian model. It first assumes a zero-mean gaussian prior over $\boldsymbol{\alpha}$, $p(\alpha_i|\tau_i) = \mathcal{N}(\alpha_i|0, 1/\tau_i)$, and then a flat hyperprior over the inverse variances τ_i. In Figueiredo (2002), two hierarchical Bayesian models are assumed. In the first level, both models assume a zero-mean gaussian over the coefficients $p(\alpha_i \mid \tau_i) = \mathcal{N}(\alpha_i \mid 0, \tau_i)$. In the second level, the first model assumes an exponential hyperprior over τ_i, $p(\tau_i \mid \gamma) = \frac{\gamma}{2}\exp\{-\frac{\gamma\tau_i}{2}\}$, while the second model assumes a noninformative Jeffreys hyperprior over τ_i, $p(\tau_i) \propto 1/\tau_i$, for $\tau_i > 0$.

One important feature for the latter two hierarchical Bayesian models is that an analytical form can be attained for the integral calculations when the E-step is conducted. More specifically, it is easily verified that the

expectation of z_i can be expressed in a closed form as

$$\mathrm{E}[z_i \mid \widehat{\alpha}_{(t)}, \mathbf{y}] = \begin{cases} \boldsymbol{\alpha}^T \mathbf{h}(\mathbf{x}_i) + \dfrac{\mathcal{N}(\boldsymbol{\alpha}^T \mathbf{h}(\mathbf{x}_i) \mid 0, 1)}{1 - \mathcal{S}(-\boldsymbol{\alpha}^T \mathbf{h}(\mathbf{x}_i) \mid 0, 1)} & \text{if } y_i = 1 \\ \boldsymbol{\alpha}^T \mathbf{h}(\mathbf{x}_i) - \dfrac{\mathcal{N}(\boldsymbol{\alpha}^T \mathbf{h}(\mathbf{x}_i) \mid 0, 1)}{\mathcal{S}(-\boldsymbol{\alpha}^T \mathbf{h}(\mathbf{x}_i) \mid 0, 1)} & \text{if } y_i = -1 \end{cases}, \tag{3.1}$$

where $\mathcal{S}(. \mid 0, 1)$ denotes the probability under a cumulative normal distribution, and the subscript t represents the tth step in the EM procedure.

If the noninformative Jeffreys hyperprior is assumed in the second stage, the expectation over τ_i^{-1} is obtained as follows:

$$\begin{aligned} \mathrm{E}[\tau_i^{-1} \mid \widehat{\alpha}_{(t)}, \mathbf{y}] &= \frac{\int_0^{+\infty} \frac{1}{\tau_i} p\big(\tau_i \mid \widehat{\alpha}_{(t)}, \mathbf{y}\big) d\tau_i}{\int_0^{+\infty} p\big(\tau_i \mid \widehat{\alpha}_{(t)}, \mathbf{y}\big) d\tau_i} \\ &= \frac{\int_0^{+\infty} \frac{1}{\tau_i} p(\tau_i) p\big(\widehat{\alpha}_{(t)} \mid \tau_i\big) d\tau_i}{\int_0^{+\infty} p(\tau_i) p\big(\widehat{\alpha}_{(t)} \mid \tau_i\big) d\tau_i} \\ &= \big|\widehat{\alpha}_{i,(t)}\big|^{-2}. \end{aligned} \tag{3.2}$$

Similarly, the expectation of τ_i^{-1} can be calculated as an analytical form for the exponential hyperprior:

$$\mathrm{E}\big[\tau_i^{-1} \,\big|\, \widehat{\alpha}_{(t)}, \mathbf{y}\big] = \gamma \big|\widehat{\alpha}_{i,(t)}\big|^{-1}. \tag{3.3}$$

On the other hand, the complete logposterior to be maximized in the M-step can be written as

$$\begin{aligned} \log p(\boldsymbol{\alpha} \mid \mathbf{y}, \mathbf{z}) &\propto \log p(\mathbf{z} \mid \boldsymbol{\alpha}) + \log p(\alpha) \\ &\propto -\|\mathbf{H}\alpha - \mathbf{z}\|^2 - \boldsymbol{\alpha}^T \boldsymbol{\Lambda} \boldsymbol{\alpha}, \end{aligned} \tag{3.4}$$

where $\boldsymbol{\Lambda} = \mathrm{diag}(1/\tau_1, \ldots, 1/\tau_l)$. The first term corresponds to the errors between the output of the learned classifier $f(\mathbf{x}) = \boldsymbol{\alpha}^T \mathbf{h}(\mathbf{x})$ and the actual output $z(\mathbf{x}, \boldsymbol{\alpha})$; the second term represents the prior imposed by the assumption over $\boldsymbol{\alpha}$.

If the expectations of equations 3.1, 3.2, or 3.3 are substituted into equation 3.4, the above maximization with respect to α can be simply computed in a closed form. The E and M steps are then conducted iteratively until a stable solution of α is obtained.

If one inspects equation 3.4 in the case of $t \to +\infty$, one can find that the second term, $\boldsymbol{\alpha}^T \boldsymbol{\Lambda} \boldsymbol{\alpha}$, changes to the L_0-norm or L_1-norm of $\boldsymbol{\alpha}$ if the

expectation of τ_i^{-1} is substituted by using equation 3.2 or 3.3, respectively. This immediately implies propositions 1 and 2, presented in the next section.

3.3 Main Results of Asymptotical Equivalence. We propose four propositions as a summary of the above derivations showing the asymptotical equivalence between hierarchical Bayesian models and L_p-norms:

Proposition 1. *The two-level hierarchical-Bayes model* $p(\alpha_i \mid \tau_i) = N(\alpha_i \mid 0, \tau_i)$, $p(\tau_i \mid \gamma) = (\gamma/2)exp(-\gamma\tau_i/2)$, $\tau_i > 0$ *over* α_i *is equivalent to the* L_1*-norm regularization term* $\|\boldsymbol{\alpha}\|_1^1$ *asymptotically.*

Proposition 2. *The two-level hierarchical-Bayes model* $p(\alpha_i \mid \tau_i) = N(\alpha_i \mid 0, \tau_i)$, $p(\tau_i) = 1/\tau_i$, $\tau_i > 0$ *over* α_i *is equivalent to the* L_0*-norm regularization term* $\|\boldsymbol{\alpha}\|_0^0$ *asymptotically.*

Proposition 3. *The priors assumed in* L_1*-norm and* L_0*-norm are related only to the term* $\boldsymbol{\alpha}^T\boldsymbol{\Lambda}\boldsymbol{\alpha}$ *as defined in the EM process, where* $\boldsymbol{\Lambda} = diag(1/\tau_1, \ldots, 1/\tau_l)$, $1/\tau_i$ $(i = 1, \ldots, l)$ *can be iteratively updated by* $\gamma \mid \widehat{\alpha}_{i,(t)} \mid^{-1}$ *for the* L_1*-norm regularization and* $\mid \widehat{\alpha}_{i,(t)} \mid^{-2}$ *for the* L_0*-norm regularization, respectively.*

Propositions 1 and 2 can be easily proved by simply substituting the stable value of $\boldsymbol{\Lambda}$. More specifically, when $t \to +\infty, \widehat{\alpha}_{i,(t)} = \widehat{\alpha}_{i,(t+1)} = \alpha_i$. Hence, the second term $\boldsymbol{\alpha}^T\boldsymbol{\Lambda}\boldsymbol{\alpha}$ in the log posterior 3.4 becomes $\gamma\|\alpha\|_1^1$ for the exponential prior and $\|\alpha\|_0^0$ for the Jeffreys prior. On the other hand, the first term of equation 3.4 is the logarithm likelihood, which represents how closely the obtained classifier agrees with the given training data. It is the second term that controls the structure complexity, and this is precisely the underlying rationale for proposition 3. Proposition 3 is important in the sense that the L_0-norm or L_1-norm can be straightforwardly integrated in many kernel machines (e.g., SVM) by simply plugging the term $\boldsymbol{\alpha}^T\boldsymbol{\Lambda}\boldsymbol{\alpha}$ into the optimization. We demonstrate a direct application of proposition 3 on SVM in the following section.

An interesting extension from propositions 1 to 3 is that we can define a term related to L_p-norm. This is shown in proposition 4:

Proposition 4. *The priors assumed in* L_p*-norm* $(0 \le p \le 2$ *or* $p = \infty)$ *are only related to the term* $\boldsymbol{\alpha}^T\boldsymbol{\Lambda}\boldsymbol{\alpha}$ *as defined in the EM process, where* $\boldsymbol{\Lambda} = diag(1/\tau_1, \ldots, 1/\tau_l)$, $1/\tau_i$ $(i = 1, \ldots, l)$ *can be iteratively updated by* $\gamma \mid \widehat{\alpha}_{i,(t)} \mid^{-(2-p)}$, *respectively.*

Proposition 4 is a significant extension. We can achieve the same effect without knowing the prior for L_p-norm explicitly. More interestingly, we can also define an L_∞-norm where $\boldsymbol{\Lambda} = \mathrm{diag}(0, \ldots, 0, 1/\alpha_{i_{\max},(t)}, 0, \ldots, 0)$ is updated iteratively by a matrix with $\alpha_{i_{\max},(t)} = \max_i \alpha_{i,(t)}$. Similar to proposition 3, this extension builds a bridge for incorporating any-norm-based

priors into most kernel learning methods without knowing the explicit prior probabilities.

3.4 L_0-Norm SVM. In this section, we show how to apply our proposed framework to the L_0-norm SVM. We provide the model definition and then present the detailed optimization algorithm.

3.4.1 Model Definition. From propositions 2 and 3, we know that L_0-norm can be asymptotically implemented by an iterative process. We integrate this implementation in SVM as follows:

$$\begin{aligned} \widehat{\boldsymbol{\alpha}}_{(t+1)} = \arg\min_{\alpha,b} \; & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C_\xi \mathbf{1}^T\boldsymbol{\xi} + C_{\boldsymbol{\alpha}}\boldsymbol{\alpha}^T\boldsymbol{\Lambda}_{(t)}\boldsymbol{\alpha} \\ \text{s.t.} \quad & y_i(\mathbf{w}^T\Phi(\mathbf{x}_i) + b) - 1 + \xi_i \geq 0, \xi_i \geq 0, \forall i, \end{aligned} \tag{3.5}$$

where $\boldsymbol{\Lambda}_{(t)} = \text{diag}(\mid \widehat{\alpha}_{1,(t)} \mid^{-2}, \ldots, \mid \widehat{\alpha}_{l,(t)} \mid^{-2})$, C_ξ and C_α are two trade-off constants tuned by the user, and $\mathbf{w} = \sum_i^l \alpha_i \Phi(\mathbf{x}_i)$.

Compared with the Bayesian learning approach for L_0-norm, the above optimization adopts the same expectation of $1/\tau$ in the E-step, while the M-step is replaced by the support vector machine optimization. If we look back to the M-step of equation 3.4, the first term $\|\mathbf{H}\alpha - \mathbf{z}\|^2$ corresponds to the errors between the output of the learned classifier $f(\mathbf{x}) = \boldsymbol{\alpha}^T\mathbf{h}(\mathbf{x})$ and the actual output $z(\mathbf{x}, \boldsymbol{\alpha})$. This is quite similar to the error term $C_\xi \mathbf{1}^T\boldsymbol{\xi}$ presented in the above model. Moreover, the second term of equation 3.4 is the same as the last term in equation 3.5. Hence, the above support vector machine optimization actually contains a meaning very similar to equation 3.4. The only difference lies at the margin maximization term $\frac{1}{2}\mathbf{w}^T\mathbf{w}$, which can be considered a regularization term and could be removed as discussed shortly. Hence, the iterated optimization of the above model describes an EM process very similar to the one presented in section 3.2. As we showed in the previous section, the prior given by the Bayesian models is exclusively determined by the term $\boldsymbol{\alpha}^T\boldsymbol{\Lambda}_{(t)}\boldsymbol{\alpha}$. It will then conveniently incorporate the prior into the kernel machines, that is, L_0-norm will be ultimately achieved.

Two points should be highlighted. First, in contrast to traditional SVMs, we do not require α to be the Lagrange multipliers. Thus, the decision function, equation 2.3, can be written as $f(\mathbf{x}) = \sum_{i=1}^l \alpha_i k(\mathbf{x}, \mathbf{x}_i) + b$. Removal of Lagrangian constraints makes the support vectors unnecessarily appear within the boundary of the final decision function. Interestingly, they represent typical or prototype samples from the data. Removal of the constraints also has another advantage: the misclassified samples are unnecessarily SVs. This will benefit classification problems when mislabeled data exist. Since all the mislabeled data points tend to be "misclassified," these mislabeled samples are highly likely to be regarded as support vectors in the L_2-norm SVM. In contrast, our proposed L_0-norm SVM avoids such problems by discarding these constraints. This will also be verified later.

Second, it is noted that the margin maximization term $\frac{1}{2}\mathbf{w}^T\mathbf{w}$ could be removed from the objective function of equation 3.5 since L_0-norm itself controls the structure complexity. Employing L_0-norm as the regularization term can be seen in sparse learning algorithms (Tipping, 2000; Figueiredo, 2002). However, we have intentionally retained the margin maximization term for three reasons. First, margin maximization controls not only the structure complexity but, more important, the generalizability. As a simple example, all the linear classifiers enjoy the same structure complexity, but it is widely recognized that the one with the maximal margin usually generalizes better. Second, the proposed model describes a rich class of approaches, depending on the values of $C_{\boldsymbol{\alpha}}$ and C_ξ. More specifically, when both $C_{\boldsymbol{\alpha}}$ and C_ξ are large enough, the margin maximization term finally disappears, and the model reduces to the true L_0-norm SVM. When $C_{\boldsymbol{\alpha}}$ is small enough, the model naturally degrades to the standard L_2-norm SVM. Finally, we note that the optimization objective is a simple combination of the L_0-norm term with the standard SVM. This shows how our proposed framework can be easily and conveniently integrated into most kernel methods.

3.4.2 Solving Method.

Theorem 1. *The above sparse learning model for SVM is equivalent to the following dual optimization problem:*

$$\begin{aligned} \boldsymbol{\beta}_{(t+1)} = & \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\boldsymbol{\beta}^T\tilde{\mathbf{H}}_{(t)}\boldsymbol{\beta} - \mathbf{1}^T\boldsymbol{\beta} \\ & s.t. \quad \mathbf{y}^T\boldsymbol{\beta} = 0, 0 \le \beta_i \le C_\xi, \forall i, \end{aligned} \tag{3.6}$$

where $\tilde{\boldsymbol{H}}_{(t)}$ denotes a symmetric matrix with elements $\tilde{h}_{ij,(t)} = y_i y_j \tilde{k}_{ij,(t)}$, $\tilde{k}_{ij,(t)}$ are the elements of a regularized kernel matrix $\tilde{\boldsymbol{K}}_{(t)} = \boldsymbol{K}(\boldsymbol{K} + C_\alpha \boldsymbol{\Lambda}_{(t)})^{-1}\boldsymbol{K}$, and $\boldsymbol{\alpha}$ and β satisfy $(\boldsymbol{K} + C_\alpha \boldsymbol{\Lambda}_{(t)})\boldsymbol{\alpha} = \boldsymbol{K}diag(\boldsymbol{y})\boldsymbol{\beta}$.

Proof. We can remove $\mathbf{w}$ from equation 3.5 by using $\mathbf{w} = \sum_i^l \alpha_i \Phi(\mathbf{x}_i)$, that is,

$$\begin{aligned} & \min \frac{1}{2}\boldsymbol{\alpha}^T\big(\mathbf{K} + C_\alpha \boldsymbol{\Lambda}_{(t)}\big)\boldsymbol{\alpha} + C_\xi \mathbf{1}^T\boldsymbol{\xi} \\ & \text{s.t.} \;\; y_i(\mathbf{K}_i^T\boldsymbol{\alpha} + b) - 1 + \xi_i \ge 0, \xi_i \ge 0, \forall i, \end{aligned} \tag{3.7}$$

where $\mathbf{K}$ denotes a symmetric kernel matrix with elements $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{K}_i$ the ith column of $\mathbf{K}$. Introducing Lagrange multipliers $\beta_i \ge 0$ and $\gamma_i \ge 0$, $i = 1, \ldots, l$, for the constraints and the slack variables, we get the Lagrangian of equation 3.7:

$$\frac{1}{2}\boldsymbol{\alpha}^T\big(\mathbf{K} + C_\alpha \boldsymbol{\Lambda}_{(t)}\big)\boldsymbol{\alpha} + C_\xi \mathbf{1}^T\boldsymbol{\xi} - \boldsymbol{\beta}^T(\mathrm{diag}(\mathbf{y})\mathbf{K}\boldsymbol{\alpha} + b\mathbf{y} - \mathbf{1} + \boldsymbol{\xi}) - \boldsymbol{\gamma}^T\boldsymbol{\xi}. \tag{3.8}$$

Taking the derivatives with respect to $\boldsymbol{\alpha}$, b, and $\boldsymbol{\xi}$ gives the following equations:

$$\begin{cases} \big(\mathbf{K} + C_\alpha \boldsymbol{\Lambda}_{(t)}\big)\boldsymbol{\alpha} = \mathbf{K}\text{diag}(\mathbf{y})\boldsymbol{\beta} \\ \mathbf{y}^T\boldsymbol{\beta} = 0 \\ \boldsymbol{\beta} + \boldsymbol{\gamma} = C_\xi \mathbf{1}. \end{cases} \tag{3.9}$$

Substituting these equations into equation 3.8, we can obtain the dual problem exactly.

It is observed that the above formulation is precisely the standard SVM optimization with the kernel matrix regularized by the sparse term. Hence, it can be efficiently solved by the sequential minimization optimization (SMO) algorithm.

After obtaining the optimal solution $\boldsymbol{\beta}_{(t+1)}$ of equation 3.6 at the tth iteration, we update the vector of coefficients by

$$\widehat{\boldsymbol{\alpha}}_{(t+1)} = \mathbf{K}_{(t)}\mathbf{K}\tilde{\boldsymbol{\beta}}_{(t+1)}, \tag{3.10}$$

where $\tilde{\beta}_{i,(t+1)} = y_i\beta_{i,(t+1)}$, $\mathbf{K}_{(t)} = (K + C_\alpha \boldsymbol{\Lambda}_{(t)})^{-1}$.

Finally, the sequence converges at the stationary $\boldsymbol{\beta}^*$ and $\boldsymbol{\alpha}^*$, and we calculate the bias term b by

$$b = \frac{1}{\mid I_{\beta^*} \mid} \sum_{i \in I_{\beta^*}} (y_i - \alpha^{*T}\mathbf{K}_i), \tag{3.11}$$

where the subindex set $I_{\beta^*} = \{i \mid 0 < \beta_i^* < C_\xi\}$.

3.4.3 Practical Issues in Optimization. We discuss two important practical issues in solving the above optimization problems.

Avoid Inversion of Zero Elements of $\widehat{\alpha}_{(t)}$. In order to calculate $\widehat{\alpha}_{(t+1)}$, we need to invert the matrix $K + C_\alpha \boldsymbol{\Lambda}_{(t)}$. This may result in problems in practice, since many elements of $\widehat{\alpha}_{(t)}$ may approach zero after several steps. This makes calculating $\boldsymbol{\Lambda}_{(t)}$ difficult. To avoid such inversions of the zero elements of $\widehat{\boldsymbol{\alpha}}_{(t)}$, we can rewrite $\tilde{\mathbf{K}}_{(t)}$ as

$$\tilde{\mathbf{K}}_{(t)} = \mathbf{K}\mathbf{K}_{(t)}\mathbf{K}, \tag{3.12}$$

where $\mathbf{K}_{(t)} = \tilde{\boldsymbol{\Lambda}}_{(t)}(\tilde{\boldsymbol{\Lambda}}_{(t)}\mathbf{K}\tilde{\boldsymbol{\Lambda}}_{(t)} + C_\alpha \mathbf{I})^{-1}\tilde{\boldsymbol{\Lambda}}_{(t)}$, $\tilde{\boldsymbol{\Lambda}}_{(t)} = \text{diag}(\mid \widehat{\alpha}_{1,(t)} \mid, \ldots, \mid \widehat{\alpha}_{l,(t)} \mid)$, and $\mathbf{I}$ is an identity matrix.

Avoid Matrix Inversion. In the above, based on matrix manipulations, we avoid the inversion of zero elements of $\widehat{\alpha}_{(t)}$, which is involved in the optimization steps. However, this is still not an adequate solution in practice, especially for large-scale problems. In such problems, matrix inversion

involves a time complexity of $\mathcal{O}(l^3)$. The workload is still big for practical problems. In the following, we view the optimization from the margin maximization and propose an analytical solution for this problem.

We recall that the requirement for maximal margin is satisfied by minimizing $\frac{1}{2}\|\mathbf{w}\|_2^2 = \frac{1}{2}\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}$. In examining the expansion $f(\mathbf{x}) = \sum_i \alpha_i k(\mathbf{x}, \mathbf{x}_i) + b$, we can regard $\Phi : \mathbf{x} \mapsto (k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_l))$ as an explicit nonlinear mapping. Therefore, the separating hyperplane is $f(\mathbf{x}) = \boldsymbol{\alpha}^T\Phi(\mathbf{x}) + b$, and the margin becomes $\frac{2}{\|\boldsymbol{\alpha}\|}$. In this sense, by using the term $\frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{\alpha}$ instead of $\frac{1}{2}\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}$, equation 3.5 will still lead to a maximal margin classifier. This substitution is important in that replacement of $\mathbf{K}_{(t)} = (\mathbf{K} + C_\alpha\boldsymbol{\Lambda})^{-1}$ by $\mathbf{K}_{(t)} = (\mathbf{I} + C_\alpha\boldsymbol{\Lambda})^{-1} = \mathrm{diag}(\frac{\alpha_1^2}{\alpha_1^2 + C_\alpha}, \ldots, \frac{\alpha_l^2}{\alpha_l^2 + C_\alpha})$ would avoid the matrix inversion problem and hence speed up the training greatly.

As a summary, we present the detailed training algorithm:

Input: training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^l$, kernel function $k(\mathbf{x}, \mathbf{x}')$, trade-off constants C_ξ and C_α;
Step 1. Initialize each coefficient $\alpha_{i,(0)} = 1$, each Lagrange multiplier $\beta_{i,(0)} = 0$, and the number of iterations $t = 0$;
Step 2. Find all nonzero coefficients $I_\alpha = \{i \,|\, \alpha_i| \geq \epsilon\}$ (e.g., a small constant $\epsilon = 10^{-4}$) and $m = |I_\alpha|$;
Step 3. Calculate the pruned matrix,
$\mathbf{K}_{(t)}^{m\times m} = \tilde{\boldsymbol{\Lambda}}_{(t)}^{m\times m}(\tilde{\boldsymbol{\Lambda}}_{(t)}^{m\times m}\mathbf{K}^{m\times m}\tilde{\boldsymbol{\Lambda}}_{(t)}^{m\times m} + C_\alpha\mathbf{I})^{-1}\tilde{\boldsymbol{\Lambda}}_{(t)}^{m\times m}$, where $\tilde{\boldsymbol{\Lambda}}_{(t)}^{m\times m}$ and $\mathbf{K}^{m\times m}$ consist of only the I_α rows and the I_α columns of $\tilde{\boldsymbol{\Lambda}}_{(t)}$ and $\mathbf{K}$.
Step 4. Calculate the regularized kernel matrix,

$$\tilde{\mathbf{K}}_{(t)} = \mathbf{K}^{l\times m}\mathbf{K}_{(t)}^{m\times m}\mathbf{K}^{m\times l},$$

where $\mathbf{K}^{m\times l}$ consists of the I_α rows of $\mathbf{K}$ and $\mathbf{K}^{l\times m}$ consists of the I_α columns of $\mathbf{K}$, and update the Hessian matrix $\tilde{\mathbf{H}}_{(t)}$.
Step 5. Take $\boldsymbol{\beta}_{(t)}$ as an initial feasible solution, and optimize the quadratic programming problem, equation 3.6, by an SMO algorithm;
Step 6. Update $\widehat{\boldsymbol{\alpha}}_{(t+1)}$ by $\boldsymbol{\beta}_{(t+1)}$ and equation 3.10;
Step 7. If $t < T_{\max}$ (e.g., $T_{\max} = 100$) and $\max_i |\widehat{\alpha}_{i,(t+1)} - \widehat{\alpha}_{i,(t)}| \geq \epsilon$, then $t \leftarrow t + 1$ and return to step 2;
Step 8. Calculate the bias term b by equation 3.11;
Output: nonzero coefficients $\{\widehat{\alpha}_i \,|\, \widehat{\alpha}_i| \geq \epsilon\}$, support vectors $\{\mathbf{x}_i \,|\, \widehat{\alpha}_i| \geq \epsilon\}$ and bias term b.

We analyze the overall training algorithm as follows. In step 1, the coefficients α_i are all initialized to 1 so that fair opportunities are provided for them to determine support vectors. In step 3, the matrices are pruned from a scale of $l \times l$ to $m \times m$, which makes calculating the matrix inversion faster and faster as the training continues. The matrix $\mathbf{K}_{(t)}$ is obtained by multiplying each element of the inversion matrix by $|\alpha_i\alpha_j|$. That is why the pruning rule can be carried out. In step 5, the initial feasible solution $\boldsymbol{\beta}_{(t)}$

is already a suboptimal solution for the quadratic programming problem, equation 3.6, so the SMO algorithm converges upon the optimal solution $\boldsymbol{\beta}_{(t+1)}$ after only a small number of iterations.

3.4.4 Complexity. We now analyze the time complexity of the proposed model. The training time is spent mainly on the matrix inversion and the SMO algorithm during the first few iterations, especially the former. However, we can successfully avoid this matrix inversion problem by employing a direct analytical form as described in section 3.4.3. After the improvement is made, the main training time is primarily decided by a sequence of SMO optimization (Platt, 1998; Flake & Lawrence, 2002), which requires a time complexity and a space complexity considerably less than $\mathcal{O}(l^3)$ and $\mathcal{O}(l^2)$, respectively, in each iteration. According to the experiments, one appealing feature of our proposed algorithm is that after only a small number of iterations (typically four or five iterations), the solution will be quite close to the stable point, which lowers the required optimization steps in each SMO process dramatically.

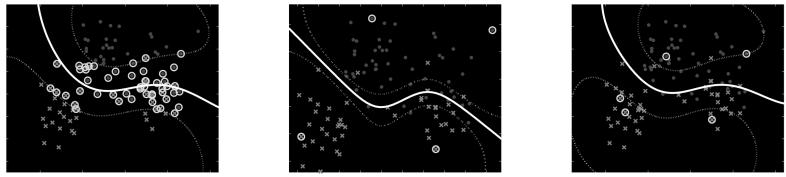
Remarks. Integration of the Bayesian learning into the large margin methods enables in each sequence the application of SMO, which proves to deliver a much smaller complexity than $\mathcal{O}(l^3)$ (Platt, 1998). In contrast, RVM, exclusively derived from the Bayesian learning approach, has to solve an $\mathcal{O}(l^3)$ problem in each step. Hence, our algorithm is much faster than RVM.

4 Experiments

In this section, we report experimental results on Ripley data and real data from the UCI machine learning repository (Blake & Merz, 1998). We take the RVM[2] and the L_2-norm SVM as the baseline in order to validate the effectiveness of our proposed L_0-norm SVM. All experiments were conducted on a PC with 4G RAM and a 3.00 GHz CPU. The kernel function used in the experiments is the popular gaussian RBF kernel.

4.1 Evaluations on Ripley Data. We first evaluate our proposed L_0-norm SVM on the Ripley data as an illustrative example. We follow the same setup as Tipping (2000) and Figueiredo and Jain (2000). The training set consists of 100 samples randomly extracted from the original set, while the test set consists of 1000 samples. The width parameter for the gaussian kernel is set to 0.5 for the three algorithms. C_ξ and C_α are, respectively, set to 1 and 0.2 for the L_0-norm SVM. The results reported are the average values from 20 classifiers learned from 20 random subsets of the 100-sample training set.

[2]The RVM Matlab package can be downloaded from http://www.miketipping.com/index.php?page=rvm.



(a) L_2-norm SVM (b) RVM (c) Proposed L_0-norm SVM

Figure 3: Comparisons of our proposed L_0-norm SVM with the L_2-norm SVM and RVM on Ripley data. $\times$'s and $\bullet$'s represent two types of data. Samples circled by $\circ$'s are the support vectors. The solid lines represent the decision boundaries, while the two dashed lines in each figure show the upper and lower margins ($f(\mathbf{x}) = \pm 1$).

Table 1: Performance on Ripley Data.

Approach	Training Time (s)	Number of SVs	Error Rate (%)
L_2-norm SVM	0.0098	49.8	9.65
L_0-norm SVM	0.0972	4.15	9.36
RVM	0.3325	4.20	9.38

To demonstrate the results visually, we plot in Figure 3 the decision boundaries given by different algorithms on one training set. It is observed that the three algorithms achieve similar classification performance. While the L_2-norm SVM outputs almost 50 support vectors, the other two algorithms generate only 4 or 5 support vectors. The L_0-norm SVM obtains almost the same decision boundary as the L_2-norm SVM but with many fewer support vectors. This shows that most of the support vectors in the L_2-norm SVM are redundant. As mentioned earlier, the support vectors of our proposed L_0-norm SVM do not necessarily appear within the margins. Instead they would rather represent the "prototype" or "typical" samples from data. This is similar to RVM.

The advantage of the proposed L_0-norm SVM over RVM is its faster training speed due to the SMO training. In Table 1, we show the training time, the number of support vectors, and the error rate for the proposed L_0-norm SVM in comparison with the L_2-norm SVM and RVM. The proposed L_0-norm SVM is slower than the L_2-norm SVM in terms of training since the L_0-norm SVM requires solving a sequence of SMO optimization. However, it is much faster than RVM, which involves matrix inversion at each step. Furthermore, the L_0-norm SVM is the most accurate among the three methods in this data set. The evaluations on Ripley data clearly demonstrate the advantages of our proposed algorithm.

In practice, it is also very common that the training data may contain some mislabeled samples. It is of great concern whether such mislabeled

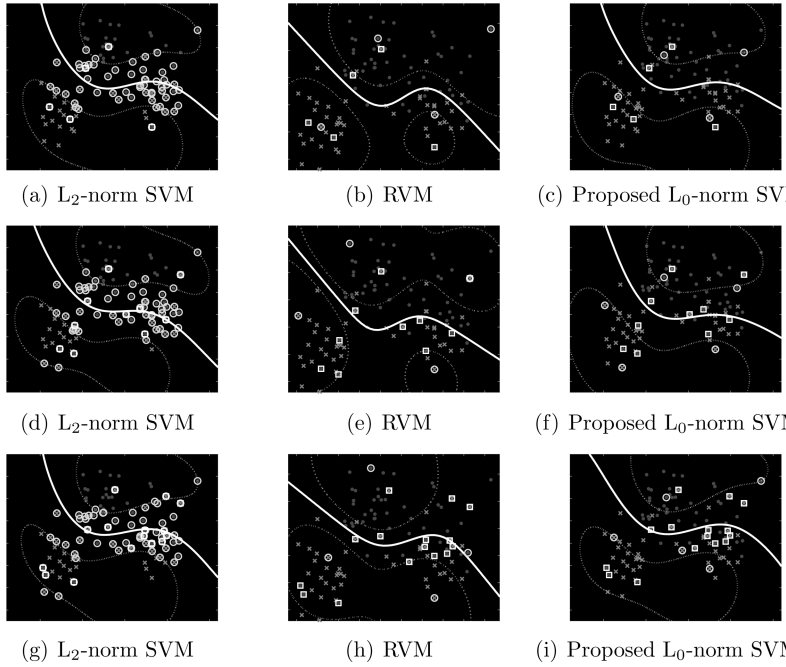
Table 2: Performance on Ripley Data When Training Data Are Corrupted with Mislabeled Samples.

Mislabeled Proportion	Approach	Number of SVs	Error Rate (%)
5%	L_2-norm SVM	55.50	9.53
	L_0-norm SVM	4.20	9.46
	RVM	4.60	9.81
10%	L_2-norm SVM	61.50	10.59
	L_0-norm SVM	4.20	9.58
	RVM	4.70	10.08
15%	L_2-norm SVM	67.40	10.90
	L_0-norm SVM	4.10	10.11
	RVM	4.30	10.86

data would influence the performance of the proposed L_0-norm SVM. In the following, we intentionally generate mislabeled data samples in order to answer this question. Specifically, the training set still consists of 100 samples randomly extracted from the original Ripley set, but we randomly switch the labels of a proportion of training samples. We then train the decision boundary based on the "polluted" samples and perform tests on 1000 samples randomly extracted from the set. This process is repeated 20 times with the mislabeled proportion set to 5%, 10%, and 15%, respectively. We report the average performance in Table 2.

Two observations are highlighted as follows. First, it is evident that mislabeled data do not affect the accuracy of the three algorithms significantly. In more detail, the errors with mislabeled samples are only marginally higher than the errors without mislabeled data. This demonstrates the advantages of large margin classifiers due to their inherit resistance to overfitting. Second, since the misclassified data points are always the SVs in the L_2-norm SVM, it is not odd that the number of SVs increases as the mislabeled proportion increases for the L_2-norm SVM. In contrast, the number of SVs is almost unchanged in the proposed L_0-norm SVM and RVM, implying that the sparse models might be more suitable in handling polluted data. Such phenomena can also be observed in Figure 4. In this figure, we plot the decision boundaries for the three algorithms on the training set, the same used in Figure 3, but randomly mislabeling 5%, 10%, and 15% samples, respectively. Again, the proposed L_0-norm SVM and the RVM generate almost the same boundary as the L_2-norm SVM, but with many fewer SVs. Moreover, all the mislabeled data points (circled by open squares) are always identified as SVs in L_2-norm SVM, while none of such samples is the SV of the proposed L_0-norm SVM. This shows that the boundary of the L_0-norm SVM is stable and hence is perhaps more suitable in handling mislabeling problems.

4.2 Evaluations on Real-World UCI Data. We evaluate the proposed L_0-norm algorithm on eight real-world data sets. The data descriptions are



(a) L_2-norm SVM (b) RVM (c) Proposed L_0-norm SVM

(d) L_2-norm SVM (e) RVM (f) Proposed L_0-norm SVM

(g) L_2-norm SVM (h) RVM (i) Proposed L_0-norm SVM

Figure 4: Comparisons of our proposed L_0-norm SVM with the L_2-norm SVM and RVM on Ripley data when mislabeled data exist. (a–c, e–f, g–i) The results plotted, respectively, when 5%, 10%, and 15% training samples are mislabeled. $\times$'s and $\bullet$'s represents two types of data. Samples circled by $\circ$'s are the support vectors. Those samples circled by $\square$'s are mislabeled training samples. The solid lines represent the decision boundaries, and the two dashed lines in each figure show the upper and lower margins ($f(\mathbf{x}) = \pm 1$).

summarized in Table 3. The first six data sets are from the UCI machine learning repository (Blake & Merz, 1998), and the remaining two are microarray gene data sets with a high dimensionality but a small number of samples.

The width parameter for all the data sets is chosen by cross-validation (CV), as are the parameters C_ξ and C_α. The reported results are the average values obtained by a 10-fold CV for the first five and the last two small-size or medium-size data sets and a five-fold CV for the remaining large-size data set, Twonorm. We report the detailed results, including the recognition accuracy, the number of support vectors, and the test time (in seconds) in Table 4.

It is observed that the L_0-norm SVM performs as competitively as or marginally better than the L_2-norm SVM in terms of accuracy. However, the support vectors needed by the L_0-norm SVM are far fewer than the

Table 3: Data Description.

Data Set	Dimension	Number in Sample
Diabetes	8	768
Solar	9	1066
German	20	1000
Thyroid	5	215
Titanic	3	2201
Twonorm	20	7400
Colon	2000	62
Lymphoma	4026	96

L_2-norm SVM. This leads to a much faster test speed. The last row of Table 4 presents the average ratio of the L_2-norm SVM, the L_0-norm SVM, and RVM to the L_2-norm SVM in terms of each measurement: the accuracy, the number of SVs, and the test time. The number of support vectors used by the L_0-norm SVM amounts to only 9.46% of those used by the L_2-norm SVM on average. The test time of the L_0-norm SVM consequently is just 18.51% as much as that of the L_2-norm SVM. Furthermore, when compared with RVM, in Diabetes, German, Thyroid, Colon, and Lymphoma, the L_0-norm SVM is comparable in terms of both accuracy and the number of support vectors. In Solar and Titanic, the L_0-norm SVM is much more accurate, with significantly fewer support vectors. In the remaining Twonorm data set, the L_0-norm SVM has almost the same accuracy as RVM but with fewer support vectors. Since RVM is dependent on matrix inversion, unlike the other models studied, RVM sometimes encounters mathematical problems, especially for medium- or large-scale tasks.[3] In comparison, our method uses the SMO algorithm, which exhibits significantly reduced time and space complexity, making the evaluation results much more stable.

To further differentiate our proposed L_0-norm algorithm from RVM, we also show their average training time required in all eight data sets (see Table 5). It can be clearly observed that the proposed sparse model is significantly faster -7.16 times faster on average when compared with RVM, although it is still slower than the L_2-norm SVM. It is not odd that the differences between Colon and Lymphoma are not evident due to their small data size. In comparison, the differences are very distinct for some large data sets, such as Titanic and Twonorm. This clearly shows the advantages of our model over RVM.

To examine the convergence performance of the proposed L_0-norm SVM, we plot the number of support vectors at each iteration or epoch in Figure 5.[4]

[3]In fact, the RVM package used sometimes crashed in our experiments because of these numerical problems.

[4]For succinctness, we plot only the number of SVs in the training at the first CV procedure for each data set.

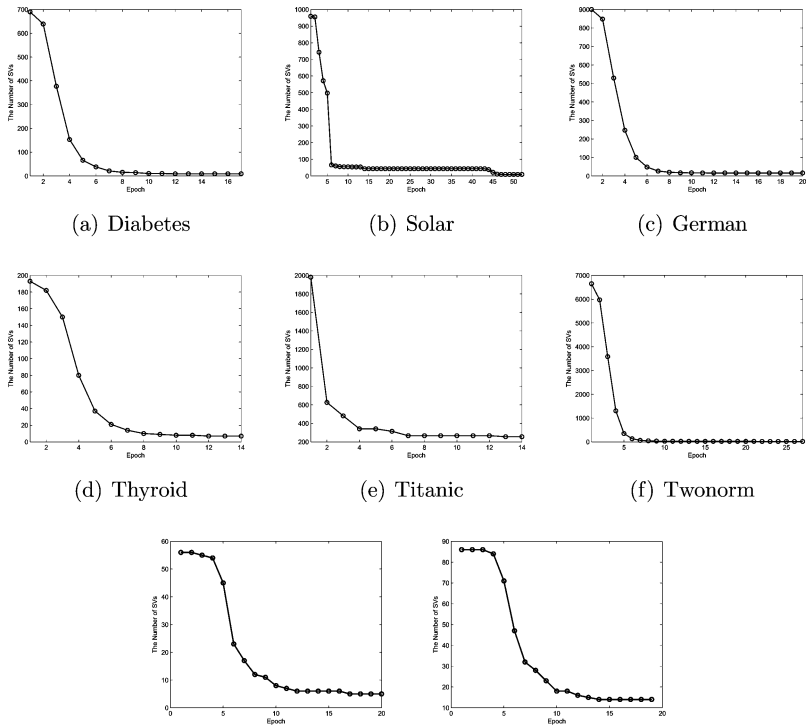
Table 4: Experimental Results on Eight Data Sets.

	L_2-norm SVM			L_0-norm SVM			RVM		
Data Set	Recognition Accuracy (%)	Number of SVs	Test Time	Recognition Accuracy (%)	Number of SVs	Test Time	Recognition Accuracy (%)	Number of SVs	Test Time
Diabetes	76.31	363.00	0.0035	76.95	8.10	0.0004	76.82	10.90	0.0004
Solar	67.36	819.90	0.0153	67.37	7.30	0.0004	62.23	268.70	0.0059
German	76.40	504.30	0.0092	76.20	17.00	0.0006	76.20	25.90	0.0008
Thyroid	96.82	26.90	0.0003	96.82	7.20	0.0002	96.82	5.00	0.0002
Titanic	78.86	1981.00	0.123	78.82	256.70	0.0131	77.81	1768.92	0.1100
Twonorm	97.70	537.40	0.2116	97.81	16.60	0.0059	97.47	39.20	0.0150
Colon	86.67	41.40	0.0033	86.67	4.50	0.0008	86.67	5.20	0.0011
Lymphoma	93.30	70.20	0.0121	93.33	10.90	0.0028	93.30	8.20	0.0026
Average ratio	1.0000	1.0000	1.0000	1.0009	0.0946	0.1851	0.9890	0.2254	0.3459

Note: Test time is in seconds.

Table 5: Training Time (s).

Data Set	L_2-Norm SVM	L_0-Norm SVM	RVM
Diabetes	0.49	4.85	13.63
Solar	0.96	17.51	166.69
German	0.79	10.74	34.25
Thyroid	0.02	0.28	0.86
Titanic	1.69	44.86	1104.00
Twonorm	17.90	717.80	8082.97
Colon	0.01	0.12	0.18
lymphoma	0.08	0.28	0.41



(a) Diabetes (b) Solar (c) German

(d) Thyroid (e) Titanic (f) Twonorm

(g) Colon (h) Lymphoma

Figure 5: Number of support vectors at each epoch in training the proposed L_0-norm SVM.

There are two interesting observations. First, the proposed algorithm converges well for all the data sets. The iterations usually stop within 50 epochs. Actually, only the Solar data set stops around 50 epochs; the remaining data sets converge within 20 epochs. Second, it is interesting that the number of SVs is reduced significantly within 5 epochs for all the data

sets. For the later epochs after five iterations, the optimization steps in each SMO required will be much more smaller than for the first few epochs since the active set is smaller. Hence, the training speed is determined mainly by a few iterations at the beginning, making the optimization quite fast.

5 Related Work

Our framework is inspired by Bayesian learning approaches. In the literature on Bayesian learning, the relevance vector machine (Tipping, 2000, 2001) and the sparse model presented in Figueiredo (2002) can be also used to achieve sparseness. These are closely related to our proposed L_0-norm model. However, these models are purely based on Bayesian learning. To extend to other norm-based classifiers, such as, L_∞-norm classifiers, they need to specify the priors explicitly, which appears not to be an easy task.

Minimal kernel classifiers (Fung, Mangasarian, & Smola, 2002) also accomplish a sparse kernel. The authors achieve the sparseness by minimizing a leave-one-out error bound. This model has two shortcomings. First, they build their work on top of the L_1-norm SVM, taking advantage of linear programming. However, for other kernel methods, it is nontrivial to use the complex derivations involved. Second, they approximate the L_0-norm $\|\boldsymbol{\alpha}\|_0^0$ by $\sum_i 1 - \exp\{\eta\alpha_i\}$, where η is a positive constant. In this sense, the classifiers are not truly minimal, since they do not achieve the true L_0-norm. In comparison, our proposed model is asymptotically equivalent to L_0-norm, so it can provide true minimal kernel classifiers.

Moreover, many methods have been used to reduce the number of support vectors. Burges (1996) proposes a reduced set method, which computes an approximation to the SVM decision rule in terms of a reduced set of vectors. A more comprehensive introduction to the reduced set method, such as reduced set selection and reduced set construction, can be found in Schölkopf et al. (1999) and Nguyen and Ho (2005). The shortcomings of the reduced set methods are that they are often computationally expensive, and the local extremum problem exists. Downs, Gates, and Masters (2002) propose a method to recognize and delete some unnecessary support vectors that are linearly dependent on other support vectors in the feature space, but it remains unclear whether the reduction can avoid any possible decline in accuracy. Li and Zhang (2006) present a method to repeat training the SVM on a gradually reduced training set until the decline in training accuracy is unacceptable or the number of support vectors stops decreasing. The above methods do not rigorously handle the support vectors reduction, since they obtain only a sparse decision function by finding an approximation to the solution of the SVM or training the SVM on the nested subsets of the training set. In comparison, our proposed L_0-norm SVM selects the most informative support vectors systematically and rigorously. The selection of SVs and the optimization is conducted in an integrated step.

In addition, we note that there are many L_0-norm-based algorithms in the literature (Weston, Elisseeff, Schölkopf, & Tipping, 2003). However, these approaches exploit L_0-norm only for feature selection, which is mainly conducted in the primal space within the linear context. It remains nontrivial or difficult to use their algorithms in the dual space whose target is to use as few kernels as possible. Moreover, similar to Fung et al. (2002), almost all of these methods approximate L_0-norm instead of achieving the true L_0-norm. Indeed, as we are proposing a general framework to achieve any norm-based algorithm, it remains interesting to determine whether our work can be smoothly applied to the feature selection field.

6 Conclusion

We have proposed a framework of arbitrary norm-support vector machines. Starting from the Bayesian learning approach, our proposed method can model any norm-based support vector machines adaptively in polynomial time. More generally, our framework can incorporate any L_p-norm priors into most kernel machines (not limited to SVMs) without knowing the explicit priors. This builds a bridge between Bayesian methods and kernel machines. We have implemented an L_0-norm SVM as a typical and important example. A series of experiments demonstrated the effectiveness of our proposed framework. The implemented L_0-norm SVM is shown to be competitive with or even better than the standard L_2-norm SVM in terms of the accuracy, but with many fewer support vectors. When compared with the classical Bayesian method; the relevance vector machine, the L_0-norm SVM has demonstrated significantly faster training speed. Future work includes investigating how to perform parameter tuning quickly and automatically. Moreover, investigating the application of our proposed framework to other fields such as regression and feature selection would be interesting topics.

Acknowledgments

The work described in this letter was fully supported by two grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK4150/07E and No. CUHK4125/07). We are grateful to the anonymous reviewers for their valuable comments.

References

Blake, C. L., & Merz, C. J. (1998). *Repository of machine learning databases*. University of California, Irvine. Available online at http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Burges, C. (1996). Simplified support vector decision rules. In *Proceedings of the 13th International Conference on Machine Learning* (pp. 71–77). San Mateo, CA: Morgan Kaufmann.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B, 39*(1), 1–38.

Downs, T., Gates, K., & Masters, A. (2002). An efficient method for simplifying support vector machines. *Journal of Machine Learning Research, 2*, 293–297.

Figueiredo, M. (2002). Adaptive sparseness using Jeffreys prior. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in neural information processing systems, 14*. Cambridge, MA: MIT Press.

Figueiredo, M., & Jain, A. K. (2000). Bayesian learning of sparse classifiers. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR' 2000)*. Washington, DC: IEEE Computer Society.

Flake, G. W., & Lawrence, S. (2002). Efficient SVM regression training with SMO. *Machine Learning, 46*(1), 271–290.

Fung, G. M., Mangasarian, O. L., & Smola, A. J. (2002). Minimal kernel classifers. *Journal of Machine Learning Research, 3*, 303–321.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics, 12*(1), 55–67.

Huang, K., Yang, H., King, I., & Lyu, M. R. (2008). Maxi-min margin machine: Learning large margin classifiers globally and locally. *IEEE Transactions on Neural Networks, 19*, 260–272.

Huang, K., Yang, H., King, I., Lyu, M. R., & Chan, L. (2004). The minimum error minimax probability machine. *Journal of Machine Learning Research, 5*, 1253–1286.

Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., & Murthy, K. R. K. (2001). Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation, 13*, 637–649.

Li, Y., & Zhang, W. (2006). Simplify support vector machines by iterative learning. *Neural Information Processing: Letters and Reviews, 10*, 11–17.

Neal, R., & Hinton, G. (1999). A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan (Ed.), *Learning in graphical models* (pp. 355–368). Cambridge, MA: MIT Press.

Nguyen, D., & Ho, T. (2005). An efficient method for simplifying support vector machines. In *Proceedings of the 22nd International Conference on Machine Learning* (pp. 617–624). New York: ACM.

Olshausen, B. A., & Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research, 37*, 3311–3325.

Platt, J. (1998). *Sequential minimal optimization: A fast algorithm for training support vector machines* (Tech. Rep. MSR-TR-98-14). Redmond, WA: Microsoft.

Saunders, C., Gammerman, A., & Vovk, V. (1998). Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning* (pp. 515–521). San Francisco: Morgan Kaufmann.

Schölkopf, B., Mika, S., Burges, C., Knirsch, P., Müller, K., Ratsch, G., et al. (1999). Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks, 10*, 1000–1017.

Tipping, M. (2000). The relevance vector machine. In S. A. Solla, T. K. Leen, & K.-R. Müller (Eds.), *Advances in neural information processing systems, 12*. Cambridge, MA: MIT Press.

Tipping, M. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research, 1*, 211–244.

Vapnik, V. N. (2000). *The nature of statistical learning theory* (2nd ed.). New York: Springer.

Weston, J., Elisseeff, A., Schölkopf, B., & Tipping, M. (2003). Use of the zero norm with linear models and kernel methods. *Journal of Machine Learning Research, 3*, 1439–1461.

Zhu, J., Rosset, S., Hastie, T., & Tibshirani, R. (2003). One-norm support vector machines. In S. Thrün, L. K. Saul, & B. Scholköpf (Eds.), *Advances in neural information processing systems, 16*. Cambridge, MA: MIT Press.

Received December 14, 2007; accepted June 18, 2008.