Core Vector Regression for Very Large Regression Problems

Ivor W. Tsang IVOR@CS.UST.HK
James T. Kwok JAMESK@CS.UST.HK
Kimo T. Lai KIMO@CS.UST.HK
Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

Abstract

In this paper, we extend the recently proposed Core Vector Machine algorithm to the regression setting by generalizing the underlying minimum enclosing ball problem. The resultant Core Vector Regression (CVR) algorithm can be used with any linear/nonlinear kernels and can obtain provably approximately optimal solutions. Its asymptotic time complexity is linear in the number of training patterns m, while its space complexity is independent of m. Experiments show that CVR has comparable performance with SVR, but is much faster and produces much fewer support vectors on very large data sets. It is also successfully applied to large 3D point sets in computer graphics for the modeling of implicit surfaces.

1. Introduction

Kernel methods have been highly successful in various machine learning problems. Among them, support vector machines (SVM) and support vector regression (SVR) are especially prominent (Schölkopf & Smola, 2002). Many kernel methods are formulated as quadratic programming (QP) problems. If m is the number of training patterns, then the training time complexity of QP is $O(m^3)$ and its space complexity is at least quadratic. Hence, a major stumbling block is in scaling up these QP's to large data sets. For example, Schölkopf et al. (2005) recently proposed a kernel method for modeling the implicit surface of a geometric object from its 3D point set. However, optimization took almost 2 hours even for some small objects.

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

To reduce the time and space complexities, a popular technique is to obtain low-rank approximations on the kernel matrix by using the Nyström method, greedy approximation or matrix decompositions. However, on very large data sets, the resulting rank of the kernel matrix may still be too high to be handled efficiently.

Another approach to scale up kernel methods is by chunking or more sophisticated decomposition methods (Collobert & Bengio, 2001; Osuna et al., 1997; Platt, 1999). In particular, the highly-popular sequential minimal optimization (SMO) algorithm (Platt, 1999) breaks the original QP into a series of smallest possible QPs, each involving only two variables. Similar in spirit to decomposition algorithms are methods that combine a large number of small SVMs. Alternatively, one can simply use a random rectangular subset of the kernel matrix, as in the reduced SVM (Lee & Mangasarian, 2001). Recently, Kao et al. (2004) and Yang et al. (2005) also proposed scale-up methods that are specially designed for the linear and Gaussian kernels, respectively.

In practice, SVM implementations typically have a training time complexity that scales between $O(m)$ and $O(m^{2.3})$ (Platt, 1999). This can be further driven down to $O(m)$ with the use of a parallel mixture (Collobert et al., 2002). However, these are only empirical observations and not theoretical guarantees.

Recently, Tsang et al. (2005) proposed the Core Vector Machine (CVM) by exploiting the "approximateness" in the design of SVM implementations. A key observation is that practical SVM implementations, as in many numerical routines, only *approximate* the optimal solution by an iterative strategy. Typically, the stopping criterion utilizes either the precision of the Lagrange multipliers or the duality gap. For example, in SMO, SVM^{light} and SimpleSVM, training stops when the Karush-Kuhn-Tucker (KKT) conditions are fulfilled within a tolerance parameter ϵ. Experience with this software indicates that near-optimal solutions are often good enough in practical applications. By utilizing an approximation algorithm for the *minimum enclosing ball* (MEB) problem in computational geometry, the CVM algorithm has an asymptotic[1] time complexity that is *linear* in m and a space complexity that is *independent* of m. Experiments on large classification data sets also demonstrated that the CVM is as accurate as existing SVM implementations, but is much faster and can handle much larger data sets than existing scale-up methods.

However, applicability of the CVM algorithm depends on the following two conditions being satisfied: 1) the kernel function k satisfies $k(\mathbf{x},\mathbf{x}) = \text{constant}$; and 2) the QP of the kernel method is of a special form. In particular, there is no linear term in the QP's objective. However, as will be shown in Section 3, the dual objective of SVR contains a linear term and so is not of the required form.

In this paper, we propose an enhancement of the CVM that allows a more general QP formulation. It turns out that this also allows the condition on the kernel to be lifted. In other words, the algorithm can now be used with *any* linear/nonlinear kernels. The rest of this paper is organized as follows. Section 2 gives a review on the CVM algorithm. Section 3 then describes the proposed extension for regression problems. Experimental results are presented in Section 4, and the last section gives some concluding remarks.

2. Core Vector Machine (CVM)

In this Section, we first review the CVM algorithm in (Tsang et al., 2005). It utilizes an approximation algorithm for the *minimum enclosing ball* (MEB) problem, which is briefly introduced in Section 2.1. The connection between the MEB problem and kernel methods, particularly the one-class and two-class SVMs, is described in Section 2.2. Finally, the CVM algorithm is presented in Section 2.3.

2.1. Approximate Minimum Enclosing Ball

Given a set of points $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, where $\mathbf{x}_i \in \mathbb{R}^D$, the minimum enclosing ball of $\mathcal{S}$ (denoted $\text{MEB}(\mathcal{S})$) is the smallest ball that contains all the points in $\mathcal{S}$. Traditional algorithms for finding exact MEBs are not efficient for problems with $D > 30$. Hence, it is of practical interest to study faster approximation algorithms that return a solution within a multiplicative factor of $1+\epsilon$ to the optimal value, where ϵ is a small positive number. Let $B(\mathbf{c}, R)$ be the ball with center $\mathbf{c}$ and radius R. Given an $\epsilon > 0$, a ball $B(\mathbf{c}, (1+\epsilon)R)$ is an *$(1+\epsilon)$-approximation* of $\text{MEB}(\mathcal{S})$ if $R \leq r_{\text{MEB}(\mathcal{S})}$ and $\mathcal{S} \subset B(\mathbf{c}, (1+\epsilon)R)$. In many shape fitting problems, it is found that solving the problem on a subset, called the *core-set*, $\mathcal{Q}$ of points from $\mathcal{S}$ can often give an accurate and efficient approximation. More formally, a subset $\mathcal{Q} \subseteq \mathcal{S}$ is a core-set of $\mathcal{S}$ if an expansion by a factor $(1+\epsilon)$ of its MEB contains $\mathcal{S}$, i.e., $\mathcal{S} \subset B(\mathbf{c}, (1+\epsilon)R)$, where $B(\mathbf{c}, R) = \text{MEB}(\mathcal{Q})$.

A breakthrough on achieving such an $(1+\epsilon)$-approximation was recently obtained by Bădoiu and Clarkson (2002). They used a simple iterative scheme: At the tth iteration, the current estimate $B(\mathbf{c}_t, R_t)$ is expanded by including the furthest point outside the $(1+\epsilon)$-ball $B(\mathbf{c}_t, (1+\epsilon)R_t)$. This is repeated until all the points in $\mathcal{S}$ are covered by $B(\mathbf{c}_t, (1+\epsilon)R_t)$. Despite its simplicity, a surprising property is that the number of iterations, and thus the size of the final core-set, depends only on ϵ but *not* on d or m.

2.2. Kernel Methods as MEB Problems

Consider the hard-margin SVDD (Tax & Duin, 1999):

$$\min R^2 \;:\; \|\mathbf{c} - \varphi(\mathbf{x}_i)\|^2 \leq R^2, \;\; i = 1, \ldots, m, \qquad (1)$$

where φ is the feature map associated with a given kernel k, and $B(\mathbf{c}, R)$ is the desired MEB in the kernel-induced feature space. Its dual is the QP: $\max\; \boldsymbol{\alpha}'\text{diag}(\mathbf{K}) - \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \,:\, \boldsymbol{\alpha} \geq \mathbf{0},\; \boldsymbol{\alpha}'\mathbf{1} = 1$, where $\boldsymbol{\alpha} = [\alpha_i, \ldots, \alpha_m]'$ are the Lagrange multipliers, $\mathbf{0} = [0, \ldots, 0]'$, $\mathbf{1} = [1, \ldots, 1]'$ and $\mathbf{K}_{m\times m} = [k(\mathbf{x}_i, \mathbf{x}_j)] = [\varphi(\mathbf{x}_i)'\varphi(\mathbf{x}_j)]$ is the kernel matrix. When k satisfies

$$k(\mathbf{x}, \mathbf{x}) = \kappa \quad \text{(a constant)}, \qquad (2)$$

we have $\boldsymbol{\alpha}'\text{diag}(\mathbf{K}) = \kappa$ using $\boldsymbol{\alpha}'\mathbf{1} = 1$. Dropping this constant term from the QP, we obtain the simpler optimization problem:

$$\max\; -\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \;:\; \boldsymbol{\alpha} \geq \mathbf{0},\;\; \boldsymbol{\alpha}'\mathbf{1} = 1. \qquad (3)$$

Conversely, whenever the kernel k satisfies (2), any QP of the form (3) can be regarded as a MEB problem. For example, the dual of the one-class L2-SVM is:

$$\max\; -\boldsymbol{\alpha}'\tilde{\mathbf{K}}\boldsymbol{\alpha} \;:\; \boldsymbol{\alpha} \geq \mathbf{0},\;\; \boldsymbol{\alpha}'\mathbf{1} = 1, \qquad (4)$$

where $[\tilde{\mathbf{K}}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) + \frac{\delta_{ij}}{C}$. Hence, if k satisfies (2), $\tilde{k}(\mathbf{x}, \mathbf{x}) = \kappa + \frac{1}{C}$ is a constant and thus the one-class L2-SVM corresponds to a MEB problem. Another example is the two-class L2-SVM. Denote the training

[1]As we are interested in handling very large data sets, an algorithm that is asymptotically more efficient (in time and space) will be the best choice. However, on smaller problems, this may be outperformed by algorithms that are not as efficient asymptotically.

set by $\{\mathbf{z}_i = (\mathbf{x}_i, y_i)\}_{i=1}^m$, where $y_i \in \{\pm 1\}$. Its dual is:

$$\max \ -\boldsymbol{\alpha}'\tilde{\mathbf{K}}\boldsymbol{\alpha} : \boldsymbol{\alpha} \geq \mathbf{0}, \ \boldsymbol{\alpha}'\mathbf{1} = 1, \tag{5}$$

where $[\tilde{\mathbf{K}}]_{ij} = \tilde{k}(\mathbf{z}_i, \mathbf{z}_j) = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) + y_i y_j + \frac{\delta_{ij}}{C}$. Again, if k satisfies (2), $\tilde{k}(\mathbf{z}, \mathbf{z}) = \kappa + 1 + \frac{1}{C}$, a constant and so the two-class L2-SVM is also a MEB problem.

Note that the 2-norm error is used here because it allows a soft-margin L2-SVM to be transformed to a hard-margin one. In theory, this could be less robust in the presence of outliers. However, experimentally, its generalization performance is often comparable to that of the L1-SVM (Mangasarian & Musicant, 2001).

2.3. The CVM Algorithm

In the following, we denote the core-set, the ball's center and radius at the tth iteration by $\mathcal{S}_t, \mathbf{c}_t$ and R_t respectively. Also, the center and radius of a ball B are denoted by $\mathbf{c}_B$ and r_B. Given an $\epsilon > 0$, the CVM proceeds as follows[2]:

```
1: Initialize $\mathcal{S}_0$, $\mathbf{c}_0$ and $R_0$.
2: Terminate if there is no training point $\mathbf{z}$ such that $\tilde{\varphi}(\mathbf{z})$ falls outside the $(1+\epsilon)$-ball $B(\mathbf{c}_t, (1+\epsilon)R_t)$.
3: Find (core vector) $\mathbf{z}$ such that $\tilde{\varphi}(\mathbf{z})$ is furthest away from $\mathbf{c}_t$. Set $S_{t+1} = \mathcal{S}_t \cup \{\mathbf{z}\}$. This can be made more efficient by using the probabilistic speedup method in (Smola & Schölkopf, 2000) that finds a $\mathbf{z}$ which is only approximately the furthest.
4: Find the new MEB($\mathcal{S}_{t+1}$) and set $\mathbf{c}_{t+1} = \mathbf{c}_{\mathrm{MEB}(\mathcal{S}_{t+1})}$ and $R_{t+1} = r_{\mathrm{MEB}(\mathcal{S}_{t+1})}$.
5: Increment $t$ by 1 and go back to Step 2.
```

3. Core Vector Regression (CVR)

Applicability of the CVM procedure depends on the following two conditions being satisfied: 1) $k(\mathbf{x}, \mathbf{x})$ is a constant; and 2) the QP problem is of the form (3). However, as will be demonstrated in Section 3.1, the dual objective of SVR contains a linear term and so is not of the form in (3). In Section 3.2, we thus propose an extension of the CVM that allows the inclusion of a linear term into the QP. It turns out that this also allows the condition on the kernel function to be lifted.

3.1. L2-Support Vector Regression

In SVR, we are given a training set $\{\mathbf{z}_i = (\mathbf{x}_i, y_i)\}_{i=1}^m$ with input $\mathbf{x}_i$ and output $y_i \in \mathbb{R}$. A linear function $f(\mathbf{x}) = \mathbf{w}'\varphi(\mathbf{x}) + b$ is then constructed in the kernel-induced feature space so that it deviates least from the training data according to the $\bar{\varepsilon}$-insensitive loss function[3], while at the same time is as "flat" as possible. We adopt the following primal which is similar to that in ν-SVR (Schölkopf & Smola, 2002):

$$\begin{aligned} \min \quad & \|\mathbf{w}\|^2 + b^2 + \frac{C}{\mu m}\sum_{i=1}^m (\xi_i^2 + \xi_i^{*2}) + 2C\bar{\varepsilon} \\ \text{s.t.} \quad & y_i - (\mathbf{w}'\varphi(\mathbf{x}_i) + b) \leq \bar{\varepsilon} + \xi_i, \\ & (\mathbf{w}'\varphi(\mathbf{x}_i) + b) - y_i \leq \bar{\varepsilon} + \xi_i^*. \end{aligned} \tag{6}$$

Here, $\mu > 0$ is a parameter (analogous to the ν in ν-SVR) that controls the size of $\bar{\varepsilon}$. Also, as in Section 2, the bias b is penalized and the two-norm errors (ξ_i^2 and ξ_i^{*2}) are used. Note that the constraints $\xi_i, \xi_i^* \geq 0$ are automatically satisfied. The corresponding dual is:

$$\begin{aligned} \max \quad & [\boldsymbol{\lambda}' \ \boldsymbol{\lambda}^{*\prime}] \begin{bmatrix} \frac{2}{C}\mathbf{y} \\ -\frac{2}{C}\mathbf{y} \end{bmatrix} - [\boldsymbol{\lambda}' \ \boldsymbol{\lambda}^{*\prime}]\tilde{\mathbf{K}} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\lambda}^* \end{bmatrix} \\ \text{s.t.} \quad & [\boldsymbol{\lambda}' \ \boldsymbol{\lambda}^{*\prime}]\mathbf{1} = 1, \ \boldsymbol{\lambda}, \boldsymbol{\lambda}^* \geq \mathbf{0}, \end{aligned} \tag{7}$$

where $\mathbf{y} = [y_1, \ldots, y_m]'$, $\boldsymbol{\lambda} = [\lambda_1 \ldots \lambda_m]'$, $\boldsymbol{\lambda}^* = [\lambda_1^* \ldots \lambda_m^*]'$, and

$$\begin{aligned} \tilde{\mathbf{K}} &= [\tilde{k}(\mathbf{z}_i, \mathbf{z}_j)] \\ &= \begin{bmatrix} \mathbf{K} + \mathbf{1}\mathbf{1}' + \frac{\mu m}{C}\mathbf{I} & -(\mathbf{K} + \mathbf{1}\mathbf{1}') \\ -(\mathbf{K} + \mathbf{1}\mathbf{1}') & \mathbf{K} + \mathbf{1}\mathbf{1}' + \frac{\mu m}{C}\mathbf{I} \end{bmatrix}. \end{aligned} \tag{8}$$

The primal variables can be recovered as

$$\begin{aligned} &\mathbf{w} = C\textstyle\sum_{i=1}^m (\lambda_i - \lambda_i^*)\varphi(\mathbf{x}_i), \quad b = C\sum_{i=1}^m (\lambda_i - \lambda_i^*), \\ &\xi_i = \lambda_i \mu m, \quad \xi_i^* = \lambda_i^* \mu m, \end{aligned} \tag{9}$$

$$\bar{\varepsilon} = [\boldsymbol{\lambda}' \ \boldsymbol{\lambda}^{*\prime}] \begin{bmatrix} \mathbf{y} \\ -\mathbf{y} \end{bmatrix} - C[\boldsymbol{\lambda}' \ \boldsymbol{\lambda}^{*\prime}]\tilde{\mathbf{K}} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\lambda}^* \end{bmatrix}. \tag{10}$$

Moreover, since $\sum_{i=1}^m (\lambda_i + \lambda_i^*) = 1$, so from (9), we have $\mu = \sum_{i=1}^m (\xi_i + \xi_i^*)/m$. Thus, μ, analogous to ν in ν-SVR, can be interpreted as the expected error. However, this QP is not of the required form in (3).

3.2. The Center-Constrained MEB Problem

The MEB (1) finds the smallest ball containing all $\varphi(\mathbf{x}_i)$'s in $\mathcal{S}$. Now, we augment each $\varphi(\mathbf{x}_i)$ by an extra $\Delta_i \in \mathbb{R}$ to form $[\varphi(\mathbf{x}_i)', \Delta_i]'$, and then find the MEB for these augmented points, while constraining the last coordinate of the ball's center to be zero (i.e., of the form $[\mathbf{c}', 0]'$) (Figure 1). The primal in (1) changes to:

$$\min R^2 : \|\mathbf{c} - \varphi(\mathbf{x}_i)\|^2 + \Delta_i^2 \leq R^2, \ \ i = 1, \ldots, m. \tag{11}$$

Denote $\boldsymbol{\Delta} = [\Delta_1^2, \ldots, \Delta_m^2]' \geq \mathbf{0}$. The new dual is:

$$\max \boldsymbol{\alpha}'(\mathrm{diag}(\mathbf{K}) + \boldsymbol{\Delta}) - \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \ : \ \boldsymbol{\alpha} \geq \mathbf{0}, \boldsymbol{\alpha}'\mathbf{1} = 1.$$

[2]Note that a similar termination parameter (ϵ) is also required in many SVM implementations (e.g., SMO).

[3]To avoid confusion with the ϵ in $(1+\epsilon)$-approximation, we add a bar to the ε in the ε-insensitive loss function.

Using the optimal $\boldsymbol{\alpha}$, we can recover R and $\mathbf{c}$ as

$$\begin{aligned} R &= \sqrt{\boldsymbol{\alpha}'(\text{diag}(\mathbf{K}) + \boldsymbol{\Delta}) - \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha}}, \\ \mathbf{c} &= \textstyle\sum_{i=1}^m \alpha_i \varphi(\mathbf{x}_i), \end{aligned} \tag{12}$$

and the square distance between $\mathbf{c}$ and any point is

$$\|\mathbf{c} - \varphi(\mathbf{x}_\ell)\|^2 + \Delta_\ell^2 = \|\mathbf{c}\|^2 - 2(\mathbf{K}\boldsymbol{\alpha})_\ell + k_{\ell\ell} + \Delta_\ell^2. \tag{13}$$

Because of the constraint $\boldsymbol{\alpha}'\mathbf{1} = 1$, an arbitrary multiple of $\boldsymbol{\alpha}'\mathbf{1}$ can be added to the objective without affecting the $\boldsymbol{\alpha}$ solution. In other words, for an arbitrary $\eta \in \mathbb{R}$, we can change the dual to

$$\begin{aligned} \max \quad & \boldsymbol{\alpha}'(\text{diag}(\mathbf{K}) + \boldsymbol{\Delta} - \eta\mathbf{1}) - \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha} \\ \text{s.t.} \quad & \boldsymbol{\alpha} \geq \mathbf{0}, \ \ \boldsymbol{\alpha}'\mathbf{1} = 1. \end{aligned} \tag{14}$$

As in Section 2, any QP of the form (14), with $\boldsymbol{\Delta} \geq \mathbf{0}$, can now be regarded as a MEB problem (11). Note that (14) allows a linear term in the objective. Besides, it does not require the kernel to satisfy condition (2).

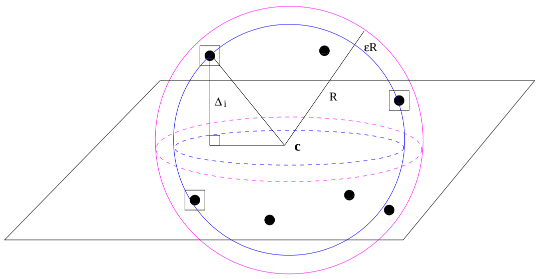


Figure 1. The center-constrained MEB problem.

Returning to the QP in (7), define $\tilde{\boldsymbol{\alpha}} = \left[\boldsymbol{\lambda}' \ \boldsymbol{\lambda}^{*\prime}\right]'$ and

$$\boldsymbol{\Delta} = -\text{diag}(\tilde{\mathbf{K}}) + \eta\mathbf{1} + \frac{2}{C}\left[\begin{array}{c} \mathbf{y} \\ -\mathbf{y} \end{array}\right] \tag{15}$$

for η large enough such that $\boldsymbol{\Delta} \geq \mathbf{0}$, (7) can then be written as $\max \ \tilde{\boldsymbol{\alpha}}'(\text{diag}(\tilde{\mathbf{K}}) + \boldsymbol{\Delta} - \eta\mathbf{1}) - \tilde{\boldsymbol{\alpha}}'\tilde{\mathbf{K}}\tilde{\boldsymbol{\alpha}} : \tilde{\boldsymbol{\alpha}} \geq \mathbf{0}, \ \tilde{\boldsymbol{\alpha}}'\mathbf{1} = 1$, which is thus of the form in (14). In other words, L2-SVR now becomes a MEB problem (11).

This extension is also useful in training the one-class and two-class L2-SVMs in Section 2.2 when the kernel does not satisfy (2). By defining $\boldsymbol{\Delta} = \max_i(\{\tilde{\mathbf{K}}_{ii}\})\mathbf{1} - \text{diag}(\tilde{\mathbf{K}}) \geq \mathbf{0}$ and $\eta = \max_i(\{\tilde{\mathbf{K}}_{ii}\})$, both duals, (4) and (5), are of the form in (14) and thus both are also instances of the MEB problem (11).

3.3. The CVR Algorithm

The algorithm in Section 2.3 can now be modified accordingly as:

```
1: Initialize[4] $\mathcal{S}_0$, $\mathbf{c}_0$ and $R_0$.
2: Terminate if there is no training point $\mathbf{z}_i$ falls
   outside the $(1+\epsilon)$-ball $B(\mathbf{c}_t, (1+\epsilon)R_t)$. i.e.,
   $\sqrt{\|\mathbf{c}_t - \varphi(\mathbf{z}_i)\|^2 + \Delta_i^2} > (1+\epsilon)R_t$.
3: Find $\mathbf{z}_i$ such that $\tilde{\varphi}(\mathbf{z}_i)$ is furthest away from $\mathbf{c}_t$.
   Set $S_{t+1} = \mathcal{S}_t \cup \{\mathbf{z}_i\}$.
4: Find the new MEB$(\mathcal{S}_{t+1})$ and set $\mathbf{c}_{t+1} =$
   $\mathbf{c}_{\text{MEB}(\mathcal{S}_{t+1})}$ and $R_{t+1} = r_{\text{MEB}(\mathcal{S}_{t+1})}$.
5: Increment $t$ by 1 and go back to Step 2.
```

Thus, the only modification is to change the distance computation between $\mathbf{c}_t$ and any point $\mathbf{z}_i$ (in Steps 2 and 3) to $\sqrt{\|\mathbf{c}_t - \varphi(\mathbf{z}_i)\|^2 + \Delta_i^2}$ using (13). Moreover, as will be shown in the following, this preserves all the properties of the original CVM algorithm as expected. However, note that the detailed proofs are different because of the use of different formulations.

3.3.1. Properties

To ensure good scaling behavior of the CVM, a key property is that it converges in at most $2/\epsilon$ iterations, independent of the feature dimensionality and the size of $\mathcal{S}$ (Bădoiu & Clarkson, 2002). The following shows that this still holds for the CVR algorithm.

Property 1 *There exists a set $\mathcal{S}_t \subset \mathcal{S}$ of size $2/\epsilon$ such that the distance between $\mathbf{c}_{MEB(\mathcal{S})}$ and any point $\mathbf{z}_i$ of $\mathcal{S}$ is at most $(1+\epsilon)r_{MEB(\mathcal{S})}$.*

Proof. The proof is adapted from that of Theorem 2.2 in (Bădoiu & Clarkson, 2002). Because of space limitations, we do not show the whole proof. Define $e_t = \|\mathbf{c}_{t+1} - \mathbf{c}_t\|$ and $\tilde{r} = (1+\epsilon)r_{\text{MEB}(\mathcal{S})}$. The only modification is in proving $R_{t+1} + e_t \geq \tilde{r}$. If all the points in $\mathcal{S}$ are at distances at most $\tilde{r}$ from $\mathbf{c}_t$, we are done. Otherwise, there exists a point $\mathbf{z}_i \in \mathcal{S}$ such that $\|\mathbf{z}_i - \mathbf{c}_t\| = \sqrt{\|\varphi(\mathbf{z}_i) - \mathbf{c}_t\|^2 + \Delta_i^2} > \tilde{r}$. This $\mathbf{z}_i$ will be added to MEB$(\mathcal{S}_{t+1})$. By the triangle inequality,

$$\begin{aligned} R_{t+1} + e_t &\geq \|\mathbf{z}_i - \mathbf{c}_{t+1}\| + \|\mathbf{c}_{t+1} - \mathbf{c}_t\| \\ &\geq \|\mathbf{z}_i - \mathbf{c}_t\| \geq \tilde{r}. \qquad \square \end{aligned}$$

Points outside MEB$(\mathcal{S}_t)$ have zero α_i's and so violate the KKT conditions of the dual problem.

Property 2 *Choosing the point furthest away from the center in Step 3 is the same as choosing the worst violating pattern corresponding to the KKT constraint. Moreover, when the CVR terminates, all the training patterns satisfy loose KKT conditions.*

Similar to the CVM, a nice property of the CVR is that its approximation ratio[5] can be obtained.

[4] As in (Tsang et al., 2005), all the $\tilde{\alpha}_i$'s (except those for the initial core vectors) are initialized to zero.

[5] An approximate algorithm has *approximation ratio* $\rho(n)$ for an input size n if $\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq \rho(n)$. Intu-

Property 3 *When $\epsilon = 0$, CVR outputs the kernel problem's exact solution. When $\epsilon > 0$ and CVR terminates at the τth iteration, the optimal primal objective p^* of the kernel problem in (6) satisfies* $\max\left(\frac{R_\tau^2}{p^*/C^2+\eta}, \frac{p^*/C^2+\eta}{R_\tau^2}\right) \leq (1+\epsilon)^2$.

Proof. The case when $\epsilon = 0$ is similar to that in (Tsang et al., 2005). When $\epsilon > 0$, assume that the algorithm terminates at the τth iteration, then $R_\tau \leq r_{\text{MEB}(\mathcal{S})} \leq (1+\epsilon)R_\tau$ by definition. Recall that the optimal primal objective p^* of the kernel problem (6) is equal to the optimal dual objective $C^2 d^*$ in (7), which in turn is related to the optimal dual objective $r^2_{\text{MEB}(\mathcal{S})} = d^* + \eta$ using (14). We can then bound p^* as

$$R_\tau^2 \leq p^*/C^2 + \eta \leq (1+\epsilon)^2 R_\tau^2. \qquad \square$$

Remarks. In other words, CVR is also an $(1+\epsilon)^2$-approximation algorithm.

3.3.2. Time and Space Complexities

Computations of the time and space complexities are analogous to those in (Tsang et al., 2005). When probabilistic speedup is not used in Step 3, the overall time for $\tau = O(1/\epsilon)$ iterations can be shown to be $O\left(\frac{m}{\epsilon^2} + \frac{1}{\epsilon^4}\right)$, linear in m for a fixed ϵ. When probabilistic speedup, one may not find the furthest point in each iteration, and so τ may be larger than $2/\epsilon$. However, it can still be bounded by $O(1/\epsilon^2)$ (Bădoiu et al., 2002). It can be shown that the whole procedure then takes $O\left(\frac{1}{\epsilon^8}\right)$, which is independent of m for a fixed ϵ. As for the space complexity[6], it can be shown that the whole algorithm requires only $O(1/\epsilon^2)$ space, independent of m for a fixed ϵ.

4. Experiments

Our CVR implementation is adapted from the LIBSVM and uses SMO for solving each QP sub-problem in Step 4. For simplicity, shrinking (Joachims, 1999) is not used in our current implementation. As in LIBSVM, our CVR uses caching and stores all the training patterns in main memory. Besides, we employ the probabilistic speedup method[7] in Step 3. The value of ϵ is fixed at 10^{-6} in all the experiments. As in other decomposition methods, the use of a very stringent stopping criterion is not necessary in practice. Preliminary studies show that $\epsilon = 10^{-6}$ is acceptable for most tasks. Using an even smaller ϵ does not show improved generalization performance, but may increase the training time unnecessarily.

4.1. CVR on Large Benchmark Data Sets

The following data sets are used[8] (Table 1):

1. Census housing: The task is to predict the median price of the house based on certain demographic information. Following (Musicant & Feinberg, 2004), we use 121 features for prediction.
2. Computer activity: Given a number of computer systems activity measures, the task is to predict the portion of time CPUs run in user mode.
3. Elevators: The task is related to an action taken on the elevators of an F16 aircraft.
4. Friedman: This is an artificial data set. The input attributes $(x_1, \ldots, x_{10})$ are generated independently, each of which uniformly distributed over $[0,1]$. The target is defined by $y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \sigma(0,1)$.
5. Pole Telecomm: The data describes a telecommunication problem. Details are not available.
6. Forest cover type: Following (Collobert et al., 2002), we aim at separating class 2 from the other classes. As in (Collobert & Bengio, 2001), we then convert this classification problem to a regression problem by predicting $+1$ for examples in class 2 and -1 for others.

For each data, $10\% - 90\%$ of the whole set is used for training while the remaining 10% are for testing.

For comparison, we also run the SVR implementations[9] of LIBSVM and SVM^{light}. The root mean squared error (RMSE) and the mean relative error (MRE)[10] are used as evaluation criteria. As the forest cover type data is a classification task, the result

itively, this ratio measures how bad the approximate solution is compared with the optimal solution. A large (small) approximation ratio means the solution is much worse than (more or less the same as) the optimal solution. If the ratio does not depend on n, we may just write ρ and call the algorithm an *ρ-approximation algorithm*.

[6]Here, we have ignored the $O(m)$ space required for storing the m training patterns, as they may be stored outside the core memory.

[7]Following (Smola & Schölkopf, 2000), a random sample of size 59 is used.

[8]The census housing data set is downloaded from http://www.cs.toronto.edu/~delve/data/census−house; forest from http://kdd.ics.uci.edu/databases/covertype/; others from http://www.niaad.liacc.up.pt/~ltorgo/Regression.

[9]LIBSVM and SVM^{light} can be downloaded from http://www.csie.ntu.edu.tw/~cjlin/libsvm/ and http://svmlight.joachims.org/ respectively.

[10]These are defined as RMSE $= \frac{1}{\max y_i}\sqrt{\frac{1}{n}\sum_{i=1}^n (f(\mathbf{x}_i) - y_i)^2}$ and MRE $= \frac{1}{n}\sum_{i=1}^n \left|\frac{f(\mathbf{x}_i)-y_i}{y_i}\right|$, respectively, where n is the number of test patterns.

Table 1. Benchmark data sets used.

date set	max # training patterns	# attr
census housing	22,732	121
computer activity	8,192	21
elevators	16,599	18
friedman	100,000	10
pole telecomm	15,000	48
forest cover type	581,012	54

is compared to the classification SVM using the classification error instead. We use the Gaussian kernel $\exp(-\|\mathbf{x}-\mathbf{y}\|^2/\beta)$, with $\beta = \frac{1}{m^2}\sum_{i,j=1}^{m}\|\mathbf{x}_i - \mathbf{x}_j\|^2$. The C and μ parameters in (6) are tuned by using a small subset of the training data. The LIBSVM and SVM^{light} implementations are tuned in a similar manner. Moreover, all implementations are in C++. Experiments are performed on a 3.2GHz Pentium–4 machine with 512M RAM, running Windows XP.

As can be seen from Figures 2 and 3, the performance of CVR is often comparable or even better than the other SVR implementations on some error criteria. Moreover, CVR is much faster and produces far fewer support vectors when the data set is large. Besides, almost all the core vectors are useful support vectors. On the very large forest cover type data set, it also agrees with our theoretical finding that the time required is constant w.r.t. the training set size.

4.2. CVR for Implicit Surface Modeling

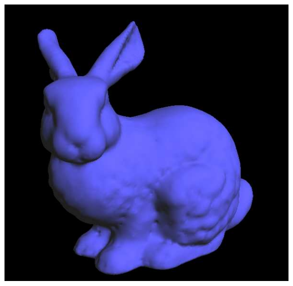
The ability to perform regression on large data sets is particularly attractive for geometric modeling with implicit surfaces. Here, a large set of 3D points (can be in the millions) is acquired from the surface of a geometric object and then modeled by the zero-set, i.e., $f^{-1}(0)$, of a smooth function $f : \mathbb{R}^3 \to \mathbb{R}$ learned from this point set.

In this Section, we use SVR for implicit surface modeling. However, a direct application of SVR can only obtain a degenerate solution with $\mathbf{w} = \mathbf{0}, b = 0$, and zero output values. To avoid this problem, we model the surface by the "one-set" $f^{-1}(1)$, instead of the zero-set. Also, the bias b is penalized, as has been done in our formulation of the L2-SVR.

Experiments are performed on the Stanford bunny and Skeleton hand data sets[11], containing 35,947 and 327,323 points respectively. We use the Laplacian kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|/\sigma)$ and a multi-scale approach as in (Schölkopf et al., 2005). A crude model is first learned by using CVR with a relatively large value of σ. Additional CVR's, each with successively smaller values of σ, are then used to fit the residuals. A total of three CVR's are used in the experiments. Finally, the mesh is computed by the commonly-used marching cubes algorithm.

[11]http://www.cc.gatech.edu/projects/large_models/.

Results are shown in Figure 4 and Table 2. Note that, again, CVR is very fast. Moreover, the set of support vectors obtained is typically small. This allows fast testing and consequently fast rendering, which only takes 493.5 seconds and 415.5 seconds for the Stanford bunny and Skeleton hand respectively.


(a) Stanford bunny.

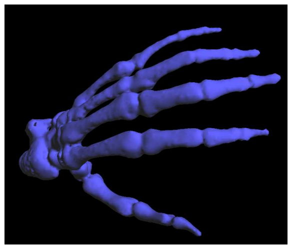

(b) Skeleton hand.

Figure 4. Reconstructed implicit surface models.

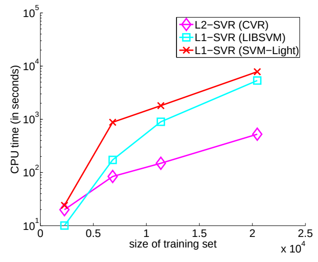
Table 2. Performance of CVR on the geometric data sets.

data set	# CVs	# SVs	CPU time
Stanford bunny	4,567	4,560	265.8s
Skeleton hand	3,810	3,808	255.8s

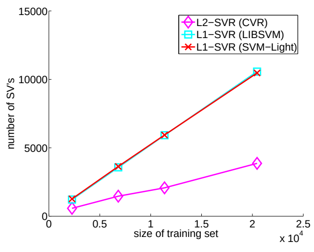
5. Conclusion

The original CVM algorithm is restricted to kernels k with constant $k(\mathbf{x}, \mathbf{x})$, and to certain kernel methods whose dual objectives do not have a linear term. In this paper, we proposed an extension that allows a more general form for the dual objective and also any linear/nonlinear kernels to be handled. In particular, regression using SVR can now be performed under this framework. The resultant CVR procedure inherits the simplicity of CVM, and has small asymptotic time and space complexities. Experimentally, it is as accurate as existing SVR implementations, but is much faster and produces far fewer support vectors (and thus faster testing) on large data sets. Using CVR, implicit surface modeling now takes minutes, instead of hours as in (Schölkopf et al., 2005).

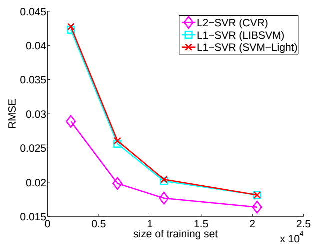
While regression has been the focus in this paper, the proposed extension can also be used for scaling up other kernel methods, such as the ranking SVM, SVMs



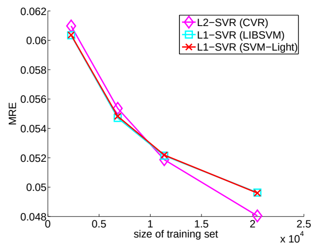
(a) CPU time.



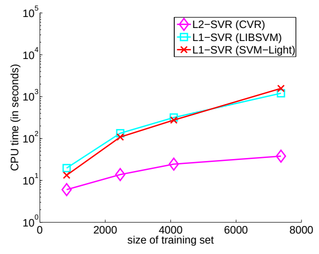
(b) number of SVs.



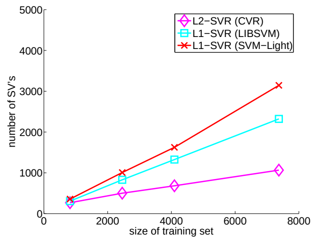
(c) RMSE.



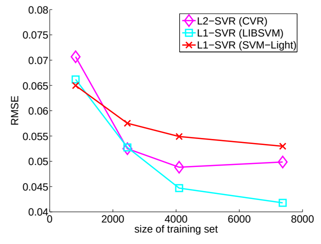
(d) MRE.



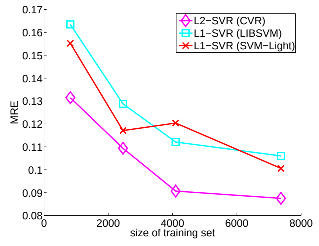
(e) CPU time.



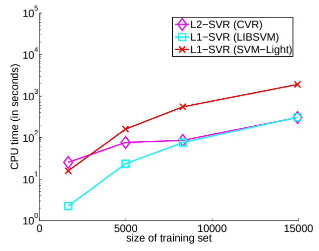
(f) number of SVs.



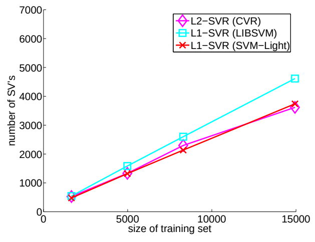
(g) RMSE.



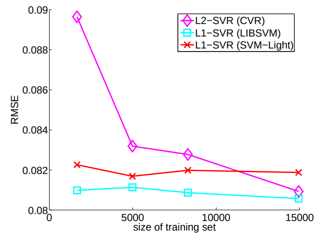
(h) MRE.



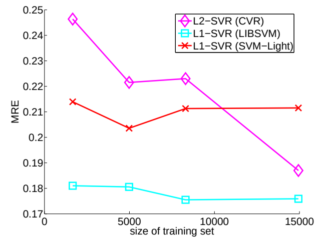
(i) CPU time.



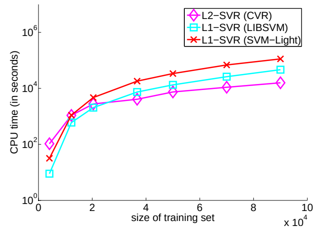
(j) number of SVs.



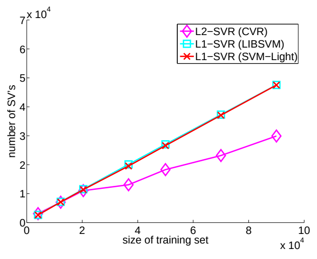
(k) RMSE.



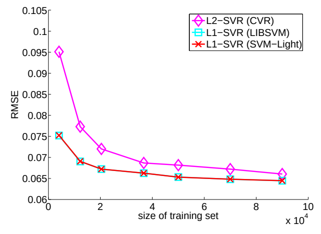
(l) MRE.



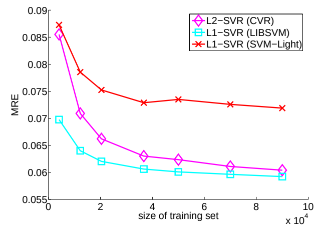
(m) CPU time.



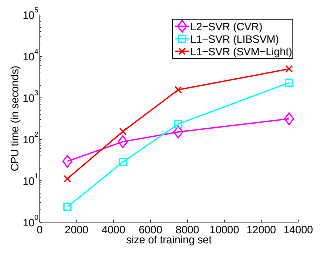
(n) number of SVs.



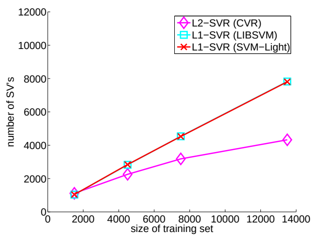
(o) RMSE.



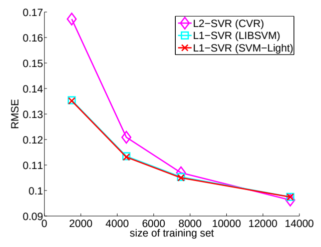
(p) MRE.



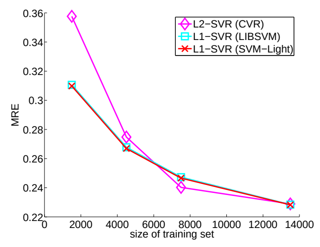
(q) CPU time.



(r) number of SVs.

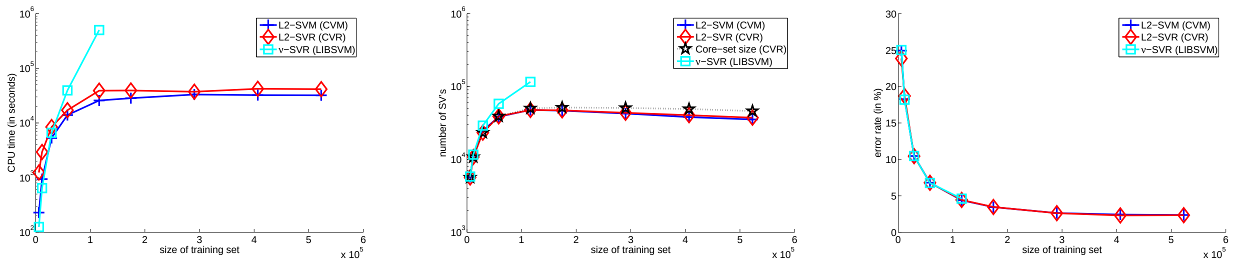


(s) RMSE.



(t) MRE.

Figure 2. Results on the regression benchmark data sets. Note that some axes are in log scale. Row 1: census housing; Row 2: computer activity; Row 3: elevators; Row 4: friedman; Row 5: pole telecomm.



(a) CPU time. (b) number of support vectors. (c) classification error on test set.

Figure 3. Results on the forest cover type classification data set. Note that some axes are in log scale.

in imbalanced learning problems, and SVMs with interdependent and structured outputs. Details will be reported in the longer version of this paper.

References

Bădoiu, M., & Clarkson, K. (2002). Optimal core-sets for balls. *DIMACS Workshop on Computational Geometry*.

Bădoiu, M., Har-Peled, S., & Indyk, P. (2002). Approximate clustering via core-sets. *Proceedings of 34th Annual ACM Symposium on Theory of Computing* (pp. 250–257). Montréal, Québec, Canada.

Collobert, R., & Bengio, S. (2001). SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, *1*, 143–160.

Collobert, R., Bengio, S., & Bengio, Y. (2002). A parallel mixture of SVMs for very large scale problems. *Neural Computation*, *14*, 1105–1114.

Joachims, T. (1999). Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges and A. Smola (Eds.), *Advances in kernel methods – Support vector learning*, 169–184. Cambridge, MA: MIT Press.

Kao, W.-C., Chung, K.-M., Sun, C.-L., & Lin, C.-J. (2004). Decomposition methods for linear support vector machines. *Neural Computation*, *16*, 1689–1704.

Lee, Y.-J., & Mangasarian, O. (2001). RSVM: Reduced support vector machines. *Proceeding of the First SIAM International Conference on Data Mining*.

Mangasarian, O., & Musicant, D. (2001). Lagrangian support vector machines. *Journal of Machine Learning Research*, *1*, 161–177.

Musicant, D., & Feinberg, A. (2004). Active set support vector regression. *IEEE Transactions on Neural Networks*, *15*, 268–275.

Osuna, E., Freund, R., & Girosi, F. (1997). Training support vector machines: an application to face detection. *Proceedings of the International Conference on Computer Vision and Pattern Recognition* (pp. 130–136).

Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges and A. Smola (Eds.), *Advances in kernel methods – Support vector learning*, 185–208. Cambridge, MA: MIT Press.

Schölkopf, B., Giesen, J., & Spalinger, S. (2005). Kernel methods for implicit surface modeling. *Advances in Neural Information Processing Systems 17*. Cambridge, MA: MIT Press.

Schölkopf, B., & Smola, A. (2002). *Learning with kernels*. Cambridge, MA: MIT Press.

Smola, A., & Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning. *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 911–918).

Tax, D., & Duin, R. (1999). Support vector domain description. *Pattern Recognition Letters*, *20*, 1191–1199.

Tsang, I., Kwok, J., & Cheung, P.-M. (2005). Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, *6*, 363–392.

Yang, C., Duraiswami, R., & Davis, L. (2005). Efficient kernel machines using the improved fast Gauss transform. *Advances in Neural Information Processing Systems 17*. Cambridge, MA: MIT Press.