支撑向量机回归的简化 SMO 算法

杨 杰，叶晨洲，全 勇，陈念贻

（上海交通大学 图像处理及模式识别研究所，上海 200030）

摘 要：统计学习理论中提出的支撑向量机回归（SVR）遵循了结构风险最小化原则，从而避免了一味追求经验风险最小化带来的弊端。采用扩展方法使 SVR 与支撑向量机分类（SVC）具有相似的数学形式，并在此基础上提出了一种用于 SVR 的简化 SMO 算法。与 SVR 现有的 SMO 算法相比，简化算法的数学形式简洁直观，在不增加算法空间和时间复杂度的前提下避免了大量繁复的判别条件，较大幅度地简化了算法实现，有利于 SVR 的广泛使用。

关键词：支撑向量机回归； 支撑向量机分类； 顺序最优化方法

中图分类号：TP18 **文献标识码**：A **文章编号**：1007-2276(2004)05-0533-05

Simplified SMO algorithm for Support Vector Regression

YANG Jie, YE Chen-zhou, QUAN Yong, CHEN Nian-yi

(Institute of Image Processing and Pattern Recognition, Shanghai JiaoTong University, Shanghai 200030, China)

Abstract: A new way of Support Vector Regression (SVR)——a new regression technique based on the structural risk minimization principle is proposed, it has the similar mathematic form as that of support vector classification, and a simplified Sequential Minimal Optimization (SMO) algorithm based on it is given. Compared with the existing algorithms, the simplified one is simpler in mathematic form and much easier to be implemented without loss in space and time efficiency, which is of benefit to the extensive application of SVR.

Key words: Support Vector Regression; Support Vector Classification; Sequential Minimal Optimization

0 引 言

统计学习理论中提出的支撑向量机（SVM）包括了支撑向量机分类和支撑向量机回归问题[1]。它们遵循了结构风险最小化原则，从而避免了一味追求经验风险最小化带来的弊端，将它们作为现实的定性或定量的数据分析工具已成为趋势。相对于 SVC，SVR 的数学形式较为复杂，使得其实现相对困难。为此，本文研究了 SVR 算法，采用扩展方法使之与 SVC 具有相似的数学形式，并在此基础上提出了一种用于 SVR 的简化 SMO（Sequential Minimal

收稿日期：2003-11-20； **修订日期**：2004-03-18

作者简介：杨杰（1964-），男，河北青县人，教授，德国汉堡大学计算机科学博士，博士生导师，主要从事目标检测和识别，数据融合和数据挖掘，医学图像处理方面的研究。

Optimization)算法，以下称为简化算法。

1 支撑向量机分类及其 SMO 算法

假设训练样本集 D_n 由 n 个样本$(\vec{x}_i, y_i)(i=1, 2, \cdots, n)$组成，这里 $\vec{x}_i \in X, y_i \in \{-1, 1\}$。SVC 的训练通过求解以下二次规划来实现：

$$(P_1)\begin{cases}\max \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j k(\vec{x}_i, \vec{x}_j) \\ \text{s. t.} \quad 0 \leqslant \alpha_i \leqslant C, \quad i = 1, \cdots, n \\ \sum_{i=1}^{n}\alpha_i y_i = 0\end{cases} \tag{1}$$

式中 $\alpha_i(i=1, \cdots, n)$是对应于第 i 个训练样本的拉格朗日系数(P_1 获解时与不为 0 的 α_k 对应的 $\vec{x}_k$ 称为支撑向量)；$k(\vec{x}_i, \vec{x}_j)$是为了使 SVC 更好地处理非线性可分问题而引入的内积函数；C 是一个正常数，也可称为惩罚因子，C 越大，说明对于错分的惩罚越大；s. t. 表示 subject to。

P_1 获解时，对所有的 $i(i=1, \cdots, n)$，KKT(Karush-Kuhn-Tucker)条件成立：

$$\begin{aligned}&\alpha_i = 0 \Rightarrow y_i f(\vec{x}_i) \geqslant 1, \\ &0 < \alpha_i < C \Rightarrow y_i f(\vec{x}_i) = 1, \\ &\alpha_i = C \Rightarrow y_i f(\vec{x}_i) \leqslant 1\end{aligned} \tag{2}$$

式中 $f(\vec{x}_*) = \operatorname{sgn}\left(\sum_{\text{与支撑向量对应的}i} y_i\alpha_i k(\vec{x}_i, \vec{x}_*) + b\right)$为 SVC 的训练结果，它表示了一个分类曲面；$\vec{x}_*$ 为任意样本。

求解规划问题 P_1 是 SVC 训练算法的主要组成部分。可以将它看作是一个带有约束条件的二次规划(QP)问题[2]。QP 问题的求解算法复杂且实现难度较大。更严重的是随着 SVC 训练样本数的增长，QP 问题对存储空间的需求以样本数的平方级增长。这些原因阻碍了 SVC 更广泛的应用。为此人们提出了多种分解算法：将一个大规模的 QP 问题分解为一系列小规模的 QP 问题进行求解，常见的有：Chunking、Osuna 和 SMO[3]。SVR 与常见的回归算法如线性和非线性回归算法明显的不同在于其目标函数中包含了一额外项，并且采用了 ε 不敏感损失函数而不是平方差损失函数衡量回归值与目标值间的差别。

SMO 算法[3]Platt 在 1998 年提出的。算法在每一次迭代中采用有限的启发式方法选择两个样本组成 QP 子问题。这样，整个过程中 QP 子问题的规模维持在 2，而这是满足(P_1)中等式约束的最低限度。对每个 QP 子问题，SMO 采用解析方法求解，从而大大提高了求解速度。当所有的 α_i 满足 KKT 条件时，算法结束。SMO 算法所需的计算机内存与训练样本数目 n 成线性关系，因而可以处理非常大的训练样本集。目前它已成为训练 SVC 最常用的算法之一。1998 年，Smola 根据 Platt 为 SVC 设计的 SMO 算法提出了针对 SVR 的 SMO 算法[4]，该算法在实现的复杂程度上大大超过了 Platt 的 SMO 算法。2002 年，Flake 等人提出了一种简化的针对 SVR 的 SMO 算法[5]，该算法在数学推导以及实现方面的简洁性仍不如 Platt 为 SVC 设计的 SMO 算法。陶卿等人提出的基于支撑向量机分类的回归方法从另一角度看待 SVR[6]。该方法只能进行线性回归，需借用现有的 SVC 训练算法，而闭凸包的收缩使得回归偏差不对回归方程产生直接影响，导致这一方法的结果与 SVR 方法的结果存在差异。

2 SVR 的简化 SMO 算法

事实上，采用简单的扩展方法，可以使 SVR 中的规划问题(P_2)与 SVC 中的规划问题(P_1)具有非常相似的表述方法。这为简化 SVR 的训练算法，甚至为将 SVC 中的 SMO 算法向 SVR 作直接移植提供了可能。

2.1 SVR 的扩展表示方法

在 SVC 中，(P_1)的矩阵表示形式为：

$$(P_{1a})\begin{cases}\max \boldsymbol{c}^{\mathrm{T}}\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{\alpha} \\ \text{s. t.} \quad \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{Y} = 0 \\ \qquad 0 \leqslant \alpha_i \leqslant C, \quad \forall i \in [0, n]\end{cases} \tag{3}$$

式中 $\boldsymbol{H} = \boldsymbol{Z}\boldsymbol{Z}^{\mathrm{T}}$；$\boldsymbol{c} = \begin{bmatrix}1 \\ \vdots \\ 1\end{bmatrix}_{n\times 1}$；$\boldsymbol{\alpha} = \begin{bmatrix}\alpha_1 \\ \vdots \\ \alpha_n\end{bmatrix}_{n\times 1}$；

$\boldsymbol{Y} = \begin{bmatrix}y_1 \\ \vdots \\ y_n\end{bmatrix}_{n\times 1}$；$\boldsymbol{Z} = \begin{bmatrix}y_1\vec{x}_1 \\ \vdots \\ y_n\vec{x}_n\end{bmatrix}_{n\cdot 1}$。

为了使 SVR 也具有类似的表示形式，将原有的训练样本集 D_n 扩展成了一个虚拟的样本集合 D_{2n}^0。它由 $2n$ 个样本$(\vec{x}_i^0, y_i^0, (i=1, 2, \cdots, 2n)$组成，当 $1 \leqslant$

$i\leqslant n$ 时，$\vec{x}_i^0=\vec{x}_i$，$y_i^0=1$；当 $n+1\leqslant i\leqslant 2n$ 时，$\vec{x}_i^0=\vec{x}_{i-n}$，$y_i^0=-1$。而 D_n 中原有的 $y_i(i=1,\cdots,n)$ 将被矩阵 $\boldsymbol{c}$ 使用。扩展过程引入了 n 个新系数 $\alpha_{n+i}=\alpha_i^*$ $(i=1,2,\cdots,n)$ 取代 α_i^*。这样 D_{2n}^0 中每个样本只与一个系数 $\alpha_i(i=1,2,\cdots,2n)$ 对应，从而解决了令算法复杂化的系数组合问题。经过扩展处理，SVR 中的 (P_2) 可以表示为：

$$(P_{2a})\begin{cases}\max \boldsymbol{c}^{\mathrm{T}}\boldsymbol{\alpha}-\dfrac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{\alpha}\\ \text{s. t.}\quad \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{Y}^0=0\\ \qquad 0\leqslant\alpha_i\leqslant C,\quad \forall i\in[0,2n]\end{cases}\tag{4}$$

式中 $\boldsymbol{H}=\boldsymbol{Z}\boldsymbol{Z}^{\mathrm{T}}$；$\boldsymbol{c}=\begin{bmatrix}c_1\\ \vdots\\ c_n\\ c_{n+1}\\ \vdots\\ c_{2n}\end{bmatrix}_{2n\times 1}=\begin{bmatrix}-(y_1+\varepsilon)\\ \vdots\\ -(y_n+\varepsilon)\\ y_1-\varepsilon\\ \vdots\\ y_n-\varepsilon\end{bmatrix}_{2n\times 1}$；

$\boldsymbol{\alpha}=\begin{bmatrix}\alpha_1\\ \vdots\\ \alpha_n\\ \alpha_{n+1}\\ \vdots\\ \alpha_{2n}\end{bmatrix}_{2n\times 1}=\begin{bmatrix}\alpha_1\\ \vdots\\ \alpha_n\\ \alpha_1^*\\ \vdots\\ \alpha_n^*\end{bmatrix}_{2n\times 1}$；$\boldsymbol{Y}^0=\begin{bmatrix}y_1^0\\ \vdots\\ y_n^0\\ y_{n+1}^0\\ \vdots\\ y_{2n}^0\end{bmatrix}_{2n\times 1}=\begin{bmatrix}1\\ \vdots\\ 1\\ -1\\ \vdots\\ -1\end{bmatrix}_{2n\times 1}$；

$\boldsymbol{Z}=\begin{bmatrix}y_1^0\vec{x}_1^0\\ \vdots\\ y_n^0\vec{x}_n^0\\ y_{n+1}^0\vec{x}_{n+1}^0\\ \vdots\\ y_{2n}^0\vec{x}_{2n}^0\end{bmatrix}_{2n\times 1}=\begin{bmatrix}y_1^0\vec{x}_1\\ \vdots\\ y_n^0\vec{x}_n\\ y_{n+1}^0\vec{x}_1\\ \vdots\\ y_{2n}^0\vec{x}_n\end{bmatrix}_{2n\times 1}$。

若不计矩阵的组成，也不计在 (P_{2a}) 与 (P_{1a}) 中 i 取值范围上的差别，那么 (P_{2a}) 与 (P_{1a}) 完全相同。扩展后回归函数可以表示为：

$$f(\vec{x}_*)=-\sum_{i=1}^{2n}\alpha_i y_i^0 k(\vec{x}_i^0,\vec{x}_*)+b\tag{5}$$

而 b 可由下式获得，当 $\alpha_l\in(0,C)$，$l\leqslant n$ 时，

$$b=y_l+\sum_{i=1}^{2n}\alpha_i y_i^0 k(\vec{x}_i^0,\vec{x}_l^0)+\varepsilon=\sum_{i=1}^{2n}\alpha_i y_i^0 k(\vec{x}_i^0,\vec{x}_l^0)-c_l\tag{6}$$

当 $\alpha_l\in(0,C)$，$l>n$ 时，

$$b=y_{l-n}+\sum_{i=1}^{2n}\alpha_i y_i^0 k(\vec{x}_i^0,\vec{x}_l^0)-\varepsilon=\sum_{i=1}^{2n}\alpha_i y_i^0 k(\vec{x}_i^0,\vec{x}_l^0)+c_l$$

定义一个辅助函数：$f_c(\vec{x}_*)=\sum_{i=1}^{2n}\alpha_i y_i^0 k(\vec{x}_i^0,\vec{x}_*)+b$，它与公式(2)SVC 使用的中符号函数内的表达式具有相同的形式。

2.2　解析解的获得

不失一般性，假设在 SMO 算法中 QP 子问题涉及的两个训练样本为 $\vec{x}_1$、$\vec{x}_2$，对应的系数为 α_1、α_2。(P_{2a}) 中的目标函数可以表示为它们的函数：

$$W(\alpha_1,\alpha_2)=c_1\alpha_1+c_2\alpha_2-\frac{1}{2}k_{11}\alpha_1^2-\frac{1}{2}k_{22}\alpha_2^2-sk_{12}\alpha_1\alpha_2-y_1^0\alpha_1\nu_1-y_2^0\alpha_2\nu_2+W_{常数}\tag{7}$$

式中 $k_{ij}=k(\vec{x}_i^0,\vec{x}_j^0)$；$\nu_i=\sum_{j=3}^{2n}y_j^0\alpha_j^{\mathrm{old}}k_{ij}=f_c^{\mathrm{old}}(\vec{x}_i)+b^{\mathrm{old}}-y_1^0\alpha_1^{\mathrm{old}}k_{1i}-y_2^0\alpha_2^{\mathrm{old}}k_{2i}$。

由 (P_{2a}) 中等式限制条件可得：

$$\alpha_1^{\mathrm{new}}+s\alpha_2^{\mathrm{new}}=\alpha_1^{\mathrm{old}}+s\alpha_2^{\mathrm{old}}=r,\quad s=y_1^0y_2^0\tag{8}$$

这样公式(5)可以表示为只与 α_2 有关的函数：

$$W=c_1(r-s\alpha_2)+c_2\alpha_2-\frac{1}{2}k_{11}(r-s\alpha_2)_1^2-\frac{1}{2}k_{22}\alpha_2^2-sk_{12}(r-s\alpha_2)\alpha_2-y_1^0(r-s\alpha_2)\nu_1-y_2^0\alpha_2\nu_2+W_{常数}\tag{9}$$

对这一函数求极值点可以获得：

$$\frac{\mathrm{d}W}{\mathrm{d}\alpha_2}=-(k_{11}+k_{22}-2k_{12})\alpha_2+s(k_{11}-k_{12})r+y_2^0(\nu_1-\nu_2)-c_1s+c_2=0\tag{10}$$

将 r 和 ν_1 代入上式，可得：

$$\alpha_2^{\mathrm{new}}=\alpha_2^{\mathrm{old}}-\frac{y_2^0(E_1-E_2)}{\eta}\tag{11}$$

式中 $\eta=2k_{12}-k_{11}-k_{22}$；$E_i=f_c(\vec{x}_i)-y_i^0c_i$。

忽略 c_i 或者矩阵 $\boldsymbol{c}$ 上的差别，前述推导与 Platt 在 SMO 中采用的一致（后者中，因 $c_i=1$ 总是成立，所以在推导过程中被隐去）。

由于 (P_{2a}) 的约束条件与 (P_1) 中的相同，α_2^{new} 的可行范围 $[L,H]$ 与 Platt 在 SMO 中采用的一致，只需根据 y_1^0 与 y_2^0 间的联系考虑两种情况（而在 Smola 算法中，类似的判别需考虑 4 种情况）：

如果 $y_1^0=y_2^0$：$L=\max(0,\alpha_1^{\mathrm{old}}+\alpha_2^{\mathrm{old}}-C)$，$H=\min(C,\alpha_1^{\mathrm{old}}+\alpha_2^{\mathrm{old}})$。

如果 $y_1^0\neq y_2^0$：$L=\max(0,\alpha_2^{\mathrm{old}}-\alpha_1^{\mathrm{old}})$，$H=$

$\min(C, C+\alpha_2^{old}+\alpha_1^{old})$。

根据 α_2^{new} 的值的可行范围对其裁剪后，可以根据公式(8)获得 α_1^{new}。由于只有两个系数发生了变动，所以算法涉及的 b 和 E_i 可以通过形式简单的迭代式获得。

对 b：若 $\exists l\in\{1,2\}$，使 $\alpha_l^{new}\in(0,C)$，那么

$$b^{new}=E_l^{old}+y_1^0(\alpha_1^{new}-\alpha_1^{old})k_{l1}+y_2^0(\alpha_2^{new}-\alpha_2^{old})k_{l2}-b^{old} \tag{12}$$

否则 b 不更新。

在个别情况下，如 $n=2$ 时，b 在整个训练过程中得不到更新，此时可以对 $l=1$ 和 $l=2$ 分别按公式(12)计算 b_1 和 b_2，然后令 $b^{new}=b_1+b_2$。

对 $E_i(i=1,\cdots,2n)$：

$$E_i^{new}=E_i^{old}+y_1^0(\alpha_1^{new}-\alpha_1^{old})k_{i1}+y_2^0\times(\alpha_2^{new}-\alpha_2^{old})k_{i2}-b^{old}+b^{new} \tag{13}$$

上述更新形式也与 Platt 在 SMO 中采用的一致(除了 b 前的符号，这一差别是表述习惯引起的)。实际上，$E_i(i=1,\cdots,2n)$ 中只需按公式(13)计算前 n 项，后 n 项可以用 $E_i=E_{i-n}-2\varepsilon(i=n+1,\cdots,2n)$ 获得。

2.3 与现有算法的比较

与 Flake 等人的算法相比，本文提出的简化算法更接近于 Platt 为 SVC 设计的 SMO 算法(扩展处理后两者几乎一致)，从而在数学推导以及实现方面更具有直观性和简洁性。与 Smola 算法相比，简化算法的优越性体现在以下 4 个方面：

(1) 对于每个 QP 子问题，Smola 算法需处理 4 个系数，而简化算法只需处理 2 个；

(2) Smola 算法中 QP 子问题的求解涉及 4 种不同的途径，而简化算法只需 1 种；

(3)对于每个 QP 子问题，Smola 算法在求解完一组系数后，还将判断是否需要处理其他系数组合，而简化算法中不存在这一问题；

(4) 简化算法对 b 和 E_i 的更新方式也较 Smola 算法简单。

简化算法中由扩展处理带来的存储空间上的增加可以忽略。这是因为训练样本集合的扩展实际上是“虚拟”的，在实现上只需采用一代换措施而不需实际存储 D_{2n}^0，$\alpha_i(i=n+1,\cdots,2n)$ 引起的空间开销与 Smola 算法中 $\alpha_i^*(i=1,\cdots,n)$ 的空间开销相抵。而 $E_i(i=n+1,\cdots,2n)$ 可由 $E_i-2\varepsilon(i=1,\cdots,n)$ 代替。矩阵 c 也不必实际生成。

通常，在 SMO 算法运行过程中进行内积计算的次数是衡量其时间复杂度的主要指标。为了对工作集合 W 中每个样本 $\vec{x}_i$ 对应的 α_i 进行优化。Platt 的 SMO 算法的时间复杂度为 $O(p^*w+(1-p)^*n)$，其中 p 是 $\vec{x}_i$ 来自于 W 的概率，w 是 W 中包含的样本个数，n 是训练样本的总数(对每个候选样本的评估最多需 3 次内积计算)。

虽然简化算法扩大了训练样本集合，但是上述时间复杂度的增加也可忽略。这是因为简化算法同样只需计算 3 次内积。若 W 包含所有(扩展后的)训练样本时，通过保存 $k_{ij}=k(\vec{x}_i^0,\vec{x}_j^0)$ 以及 $k_{jj}=k(\vec{x}_j^0,\vec{x}_j^0)$(其中 i 固定而 $j\leqslant n$)的计算结果并利用 $k_{i,j+1}=k(\vec{x}_i^0,\vec{x}_{j+n}^0)=k_{ij}$ 以及 $k_{j+n,j+n}=k(\vec{x}_{j+n}^0,\vec{x}_{j+n}^0)=k_{jj}$，可以使前述的时间复杂度不变。而当 W 包含所有对应系数 $\alpha_i\in(0,C)$ 的训练样本时，在同等情况下，扩展前后 W 的尺寸不变。这是因为 SVR 中 α_i、α_i^* $(i=1,\cdots,n)$ 两者中最多只有一个不为 0，而简化算法中 $\alpha_{n+i}=\alpha_i^*(i=1,2,\cdots,n)$。此时采用保存内积的方法也可以使前述的时间复杂度不变。

每个 QP 子问题中，Smola 算法最多对相关的系数组合(α_i、α_i^*、α_j 和 α_j^* 的组合)进行两次求解，而后当新解与旧解的差别大于一个预设的阈值时对 b 和 E_i 进行一次更新。简化 SMO 算法中，每个 QP 子问题的求解最多只对相关系数(α_i 和 α_j 的组合)进行一次求解，当新解与旧解的差别大于一个预设的阈值时更新 b 和 E_i。当 Smola 算法中 QP 子问题只涉及一次求解时，因为要多次判断是否需要处理其他系数组合，所以比简化算法更耗时；但当它涉及两次求解时，Smola 算法可以省去一次更新 b 和 E_i 所需的时间开销。表面看来这种情况下 Smola 算法在时间上更经济，但是由于简化算法在构造 QP 子问题时，采用启发式方法在所有系数间进行两两挑选，提高了 QP 子问题获解时规划目标获得的增益，从而有可能弥补额外的时间开销(以下的实验证实了这一点)。另外，简化算法在判别条件和更新方式上的简化也可能减小时间开销，甚至使算法在实际中有更快的求解速度。由于两种算法在构造 QP 子问题的过程中都采用了启发式的方法和随机选择的方法，所以很难对整个训练过程的时间复杂度进行更严格的比较。

3 实 验

实验从回归结果、支撑向量数以及 E_i 数组的更

新次数三个方面比较了 Smola 算法和本文提出的简化算法(Flake 的算法在本质上与 Smola 算法相同，因此未参与比较)。E_i 数组的每一次更新表示了算法对目标值的一次优化，它也是两种算法中最为耗时的操作，用这一操作的次数衡量算法的运行时间。由于两种算法都是在新旧系数间的差别大于某个阈值时才执行这一操作的，所以为了保证实验比较的公平性对这一阈值作了相同的规定。此外，在某些情况下，两种算法都使用了随机选择方法构造 QP 子问题，因此即使是同一算法对同一训练样本集合的不同次拟合也会在这些指标上获得不同的结果。为此表 1 取了运行 10 次后获得的平均值。各次实验情况如下：

实验 1　训练样本集合包含 20 个样本，维数为 1，样本近似分布在 $f(x)=x+2, x\in[1,20]$ 上。核函数采用 $k(\vec{x}_1,\vec{x}_2)=\vec{x}_1\cdot\vec{x}_2$。$C=1,\varepsilon=1$。两种方法的线性回归结果都为 $y=0.905882x+2.988235$。其他比较如表 1 所示(以下同)。

表 1　两种用于 SVR 的 SMO 算法的回归结果比较

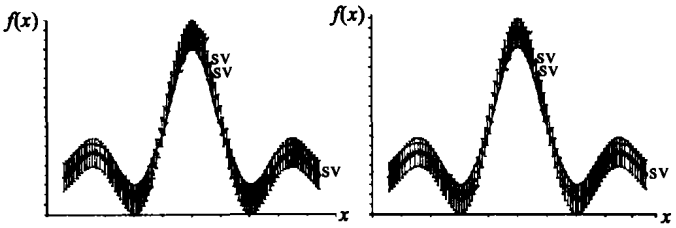
Tab. 1　Comparison of the regression results of two SMO algorithms for SVR

No	Smola algorithm		Simplified SMO algorithm	
	Number of support vectors	Time of array revision	Number of support vectors	Time of array revision
1	3.41±2.14	52.30±38.31	2.0±0.0	46.00±4.40
2	3.0±0.0	30.60±14.62	3.0±0.0	58.80±48.28
3	3.20±0.40	87.70±28.84	3.0±0.0	85.88±5.78
4	24.70±1.73	26250.80±9353.90	21.30±0.46	15844.60±4243.37

实验 2　训练样本集合与实验 1 基本相同，但有 1 个样本远离该直线。核函数采用 $k(\vec{x}_1,\vec{x}_2)=\vec{x}_1\cdot\vec{x}_2$。$C=1,\varepsilon=4$。两种方法的线性回归结果都为 $y=x+5.8$。

实验 3　训练样本集合包含 60 个样本，维数为 1，样本分布在曲线 $f(x)=\dfrac{\sin x}{x}$、$x\in[-3.14,3.04]$ 上。核函数采用 $k(\vec{x}_1,\vec{x}_2)=(\vec{x}_1\cdot\vec{x}_2+1)^2$。$C=1,\varepsilon=0.1$。

实验 4　训练样本集合包含 100 个样本，维数为 1，样本分布在曲线 $f(x)=\dfrac{\sin x}{x}$、$x\in[-10,9.8]$ 上。核函数采用 $k(\vec{x}_1,\vec{x}_2)=e^{-(\vec{x}_1-\vec{x}_2)^T(\vec{x}_1-\vec{x}_2)}$。$C=1,\varepsilon=0.1$。图 1 为两种算法的拟合结果。



(a) Smola 算法　　(b) 简化算法

(a) Smola algorithm　　(b) Simplified algorithm

图 1　两种算法的回归结果(+：样本点；I：ε；实线：回归结果；SV：支撑向量)

Fig. 1　Regression results of two algorithms (+: training sample; I: ε; solid line: regression result; SV: support vector)

4　结　论

与 SVR 现有的 SMO 方法相比，简化算法的数学形式简洁直观，在不增加算法空间和时间复杂度的前提下避免了大量繁复的判别条件，较大幅度地简化了算法实现，并为 SVC 采用的 SMO 算法向 SVR 作直接移植提供了可能，有利于 SVR 的广泛使用。Keerthi 等人指出 SMO 算法中对阈值 b 的处理不合理，使得算法在实际上应该终止时仍然进行费时的优化操作。采用一个范围代替单个值，可以使改进后的 SMO 算法具有更高的效率(实验中训练耗时减小的幅度可达 50%)。Flake 等人对 SVR 的 SMO 算法提出了一系列常规的改进(实验中训练耗时减小的幅度可达 90%)[5]，以便应用在本文提出的简化算法中。

参考文献：

[1] Vapnik V. The Nature of Statistical Learning Theory. 2nd ed. [M]. NewYork: Springer-Verlag, 1998.

[2] 袁亚湘，孙文瑜. 最优化理论与方法[M]. 北京：科学出版社，1999.

[3] Platt J C. Fast Training of Support Vector Machines Using Sequential Minimal Optimization[M]. Advances in Kernel Methods: Support Vector Machines (Edited by Scholkopf B, Burges C, Smola A)[M]. Cambridge MA: MIT Press, 1998. 185-208.

[4] Smola Alex J, Scholkopf Bernhard. A Tutorial on Support Vector Regression[EB/OL]. http://www.neurocolt.com NeuroCOLT2 Technical Report Series NC2-TR-1998-030.

[5] Flake Gary William, Lawrence Steve. Efficient SVM Regression Training with SMO[J]. Machine Learning, 2002, 46(1/2/3): 271-290.

[6] 陶卿，曹进德，孙德敏. 基于支撑向量机分类的回归方法[J]. 软件学报，2002，13(5)：1024-1027.