# 面向超大数据集的SVM近似训练算法

曾志强[1,2] 廖备水[2] 高 济[2]

(厦门理工学院计算机科学与技术系 厦门 361024)[1] (浙江大学计算机科学与技术学院 杭州 310027)[2]

**摘 要** 标准SVM学习算法运行所需的时间和空间复杂度分别为$O(l^3)$和$O(l^2)$,$l$为训练样本的数量,因此不适用于对超大数据集进行训练。提出一种基于近似解的SVM训练算法:Approximate Vector Machine (AVM)。AVM采用增量学习的策略来寻找近似最优分类超平面,并且在迭代过程中采用热启动及抽样技巧来加快训练速度。理论分析表明,该算法的计算复杂度与训练样本的数量无关,因此具有良好的时间与空间扩展性。在超大数据集上的实验结果表明,该算法在极大提高训练速度的同时,仍然保持了原始分类器的泛化性能,并且训练完毕具有较少的支持向量,因此结果分类器具有更快的分类速度。

**关键词** 支持向量机,核函数,增量学习,近似解,核心集

**中图法分类号** TP181 **文献标识码** A

## Approximate Approach to Train SVM on Very Large Data Sets

ZENG Zhi-qiang[1,2] LIAO Bei-shui[2] GAO Ji[2]

(Department of Computer Science and Technology, Xiamen University of Technology, Xiamen 361024, China)[1]

(College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China)[2]

**Abstract** Standard Support Vector Machine (SVM) training has $O(l^3)$ time and $O(l^2)$ space complexities, where $l$ is the training set size. It is thus computationally infeasible on very large data sets. A novel SVM training method, Approximate Vector Machine (AVM), based on approximate solution was presented to scale up kernel methods on very large data sets. This approach only obtains an approximately optimal hyper plane by incremental learning, and uses probabilistic speedup and hot start tricks to accelerate training speed during each iterative stage. Theoretical analysis indicates that AVM has the time and space complexities that are independent of training set size. Experiments on very large data sets show that the proposed method not only preserves the generalization performance of the original SVM classifiers, but outperforms existing scale-up methods in terms of training time and number of support vectors.

**Keywords** Support vector machine, Kernel function, Incremental learning, Approximate solution, Core set

SVM训练问题的本质就是求解一等式约束的二次规划问题,当样本数量很大时,训练SVM需要极大的时间和空间开销[1,2],因此设计适用于大样本的训练算法成为SVM研究中的重要内容。经过研究者的不懈努力,已存在许多扩展的SVM训练算法能够有效处理大规模数据集,然而,对于超大规模数据集($>10^5$),目前能够对其高效处理的SVM训练算法很少,并且大多依靠经验观察,缺乏理论保障。文献[3]中,H. Yu等人将层次聚类方法用于SVM训练超大数据集,然而由于聚类发生在输入空间而距离计算进行在特征空间,因此,此种方法只适用于线性可分的样本集,极大地限制了其应用范围。Erdem[4]等人采用AdaBoosting和增量学习相结合的方法来训练超大数据集,虽然此种方法能够有效减少训练时间及内存开销,但训练完毕后会产生数量巨大的弱SVM分类器,从而增大了分类复杂度。Ivor Tsang等人将SVM训练问题转化为求特征空间中最小超球问题,并据此提出了训练算法:Core Vector Machine(CVM)[5],理论分析及实验结果表明,CVM适合于进行超大数据集训练,然而,该算法由于只能采用各向同性核函数$k(x,y)=K(||x-y||)$(例如高斯核函数),因此限制了其应用范围。另一方面,CVM对应的SVM二次式只能采用二次损失函数,而采用二次损失函数不如采用一次损失函数更易获得稳定的分类性能[6]。

鉴于此,本文提出了一种新的适用于处理超大规模数据集的SVM训练算法:Approximate Vector Machine (AVM),AVM采用近似最优解和增量学习相结合的方式来求解SVM,它不仅具有坚实的理论基础,而且克服了上述算法的不足之处。实验结果表明,该算法在极大提高超大规模数据集上的SVM学习速度的同时,基本保持了原始分类器的分类精度。

## 1 SVM简介

训练SVM的本质就是求解一最优分类超平面问题,给定训练样本$(x_i,y_i)$,$i=1,\cdots,l$,其中$x_i\in R^n$,$y_i\in\{-1,1\}$,

到稿日期:2008-12-24 返修日期:2009-03-12 本文受国家自然科学基金(60773177),福建省青年人才项目(2008F3108),厦门理工学院引进人才项目(YKJ08003R)资助。

曾志强(1971-),男,博士,讲师,主要研究领域为模式识别、机器学习,E-mail:lbxzzq@163.com;廖备水(1971-),男,博士,副教授,主要研究领域为Agent技术、机器学习;高 济(1946-),男,教授,博士生导师,主要研究领域为人工智能、Agent技术、网格计算。

求解最优分类超平面可以转化为一个二次优化问题[1]：

$$\min_{w,b,\xi}\frac{1}{2}w^Tw+C\sum_{i=1}^{l}\xi_i \tag{1}$$

subject to $y_i(w^T\varphi(x_i)+b)\geqslant 1-\xi_i,\xi_i\geqslant 0,i=1,\cdots,l$

其中，$\xi_i,i=1,\cdots,l$ 为松弛因子，$C>0$ 为惩罚算子，采用 Lagrange 乘子法可以获得式(1)的对偶问题：

$$\min_{\alpha}\frac{1}{2}\alpha^TQ\alpha-e^T\alpha \tag{2}$$

subject to $0\leqslant\alpha_i\leqslant C,i=1,\cdots,l,y^T\alpha=0$

其中，$e$ 是都为 1 的向量，$\alpha_i,i=1,\cdots,l$ 为 Lagrange 乘子，$Q$ 为 $l\times l$ 的对称矩阵，称为 Hessian 矩阵，$Q_{ij}=y_iy_jk(x_i,x_j)=y_iy_j\varphi(x_i)^T\varphi(x_j)$，$k(x_i,x_j)$为核函数，它对应于采用非线性映射 $\varphi:R^n\mapsto F$ 将训练样本从输入空间映射到某一特征空间 $F$，在该特征空间中，样本是线性可分的。SVM 训练所得分类超平面的函数形式如下所示：

$$f(x)=\sum_{i=1}^{N_{sv}}\alpha_iy_ik(x_i,x)+b \tag{3}$$

所有样本点必须满足 KKT[1] 条件是 SVM 二次优化问题最优解的充要条件。KKT 条件描述如下：假设对偶问题的最优解为 $\alpha^T=[\alpha_1,\alpha_2,\cdots,\alpha_l]$，$f(x)$为最优分类超平面所对应的函数，则样本点 $x_i$ 满足：

$$\begin{aligned}&\alpha_i=0\Rightarrow y_if(x_i)\geqslant 1\\&0<\alpha_i<C\Rightarrow y_if(x_i)=1\\&\alpha_i=C\Rightarrow y_if(x_i)\leqslant 1\end{aligned} \tag{4}$$

即最优分类超平面将所有训练样本分为 3 类，$\alpha_i=0$ 对应的样本点分布在分类间隔之外，$0<\alpha_i<C$ 的样本在分类间隔之上，$\alpha_i=C$ 的样本在分类间隔之内。根据式(3)可知，最优分类超平面仅由支持向量，即 $\alpha_i\neq 0$ 所对应的样本点来确定。

## 2 Approximate Vector Machine

根据上节介绍的 SVM 特性可知，最优分类超平面仅由支持向量来确定，即由分类间隔之上和之内的样本点来确定，分类间隔之外的样本点对超平面的形成没有影响。AVM 算法就是根据此特性来设计的，它仅在训练样本的某一子集上进行训练，此训练子集将包含最优分类超平面分类间隔之上和之内的样本点，因此可以看作一潜在支持向量集。定义样本全集为 $U$，该训练子集为 $V$，所有样本的初始 $\alpha$ 值都为零，AVM 采用增量学习的方式来训练 SVM，从初始训练子集 $V_0$ 开始，假设第 $t$ 次迭代的训练子集为 $V_t$，获得的分类超平面为 $h_t$，则根据 $h_t$ 在 $U/V_t$ 中寻找潜在支持向量$(x_t,y_t)$，令 $V_{t+1}=V_t\cup\{(x_t,y_t)\}$，在 $V_{t+1}$ 上重新训练 SVM 以获得新的分类超平面，重复迭代过程直到：

$$\arg\min_{(x_i,y_i)\in U/V_t}y_if_t(x_i)>1 \tag{5}$$

其中，$f_t(x)$为超平面 $h_t$ 所对应的函数。根据式(5)可知，当迭代终止时，集合 $U/V$ 里的样本点都在最终所得超平面的分类间隔之外，并且其对应的 $\alpha$ 值等于零，因此满足 KKT 条件。此时，全集 $U$ 的所有样本点都满足 KKT 条件，因此，最终所得分类超平面就是最优分类超平面。显然，采用式(5)这个停机条件，AVM 算法将得到 SVM 的精确解。

### 2.1 近似最优分类超平面

为了得到准确的最优分类超平面，AVM 算法必须迭代寻找所有支持向量，对于超大数据集或支持向量浓密解的训练集，这个迭代过程将非常冗长耗时。文献[5,7]中，Badoiu 等人通过求取最小超球的近似解来加快训练速度，本文借鉴其策略，通过求取特征空间中的近似最优分类超平面来减少迭代次数，提高训练效率。

训练 SVM 也可以看作是求取最大分类间隔的过程，给定数据集 $D$，训练 SVM 等价于：

$$\chi(D)=\arg\max_{w\in R^d,||w||=1}\min_{(x,y)\in D}y(w\cdot\varphi(x)) \tag{6}$$

相对于其它分类超平面，最优分类超平面 $w\cdot\varphi(x)=0$ 是在数据集 $D$ 上取得最大分类间隔的超平面。假设 $\rho(h^*,D)$表示在数据集 $D$ 上训练所得最优分类超平面 $h^*$ 的分类间隔，则最优分类超平面的$(1-\varepsilon)$近似和核心集定义如下[8]：

**定义 1**($(1-\varepsilon)$近似) 如果分类超平面 $h$ 满足 $\rho(h,D)\geqslant(1-\varepsilon)\rho(h^*,D)$，$\varepsilon\in(0,1)$，则称 $h$ 为最优分类超平面 $h^*$ 的$(1-\varepsilon)$近似。

**定义 2**(核心集) 如果集合 $E\subset D$ 并且分类超平面 $h=\chi(E)$是最优分类超平面 $h^*=\chi(D)$的$(1-\varepsilon)$近似，则称集合 $E$ 为核心集。

根据上述定义，修改 AVM 算法的停机条件如下：

$$\arg\min_{(x_i,y_i)\in U/V_t}y_if_t(x_i)\geqslant 1-\varepsilon \tag{7}$$

此时的训练子集 $V$ 就是核心集，迭代完成后，集合 $U/V$ 里的样本点都能被近似最优分类超平面 $h$ 正确分类，并且它们和 $h$ 的距离大于等于 $1-\varepsilon$。改进后的 AVM 算法通过减小分类间隔来减少算法的迭代次数，以提高训练效率。下文的理论分析表明，参数 $\varepsilon$ 的值在算法的计算复杂度和结果分类器的分类精度间起折衷作用。

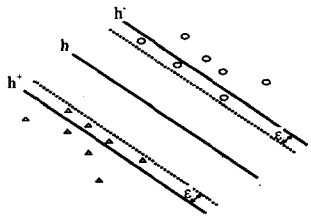


图 1 集合 U/V 中的样本点的分布状况

### 2.2 寻找核心集元素

AVM 算法在每次迭代中都要根据式(7)判断算法是否满足停机条件，如果不满足，则需要在集合 $U/V_t$ 选取一样本点加入核心集 $V_t$ 中，重新生成分类超平面。该样本点的选取非常重要，因为它直接关系到 AVM 算法的收敛速率。本文中，选取满足下式的样本点加入核心集中：

$$(x_t,y_t)=\arg\min_{(x_i,y_i)\in U/V_t}y_if_t(x_i) \tag{8}$$

其中，$V_t$ 和 $f_t(x)$分别代表当前迭代阶段的核心集和分类超平面。当 $y_tf_t(x_t)<0$ 时，所选取的样本点就是距离当前分类超平面最远的分类错误点；当 $y_tf_t(x_t)\geqslant 0$ 时，它就是距离当前分类超平面最近的，并且被正确分类的样本点。由于集合 $U/V_t$ 中的元素所对应的 $\alpha$ 值等于零，根据式(4)和式(8)可知，每次迭代所选取的样本点也就是当前最违反 KKT 条件的样本点。

采用式(8)选取核心集元素需要计算集合 $U/V_t$ 中的每个元素到当前分类超平面的距离，假设第 $t$ 次迭代所获得的分类超平面是由 $n_t$ 个支持向量确定的，集合 $U/V_t$ 含有 $m_t$ 个元素，则计算 $U/V_t$ 中的每个元素到当前分类超平面的距离所需时间为 $O(n_tm_t)$(以核函数计算次数来度量)，由于本算

法所要处理的是超大规模数据集，通常 $m_t$ 的值很大，因此，这个距离计算需要较大的时间开销。为解决此问题，算法设计从集合 $U/V_t$ 中随机选取一包含 59 个元素的子集 $F_t$，然后在此子集中选取对应 $yf_t(x)$ 值最小的样本点加入核心集：

$$(x_t,y_t)=\arg\min_{(x_i,y_i)\in F_t} y_i f_t(x_i) \tag{9}$$

这个样本点位于前 5%最小的 $\{y_i f_t(x_i)\mid(x_i,y_i)\in U/V_t, i=1,\cdots,|U/V_t|\}$ 之内的概率为 95%[9]。

### 2.3 SMO 算法的热启动

在 AVM 算法中，每次迭代都需要在当前核心集 $V_t$ 上训练 SVM，以获得新的分类超平面，本文采用 SMO[10] 算法作为标准的 SVM 训练算法。虽然 SMO 算法具有较快的训练速度，然而，当核心集中样本数较多时，SMO 算法的训练效率就会下降，尤为重要的是，每次迭代重新训练 SVM 必将造成巨大的时间开销。考虑到当前核心集 $V_t$ 和上次迭代的核心集 $V_{t-1}$ 相比只增加一个新的元素，因此采用“热启动”的方法来训练 SVM，每次 SVM 训练都以上次 SMO 算法的训练终点为起点。设 $\alpha=[\alpha_1 \quad \cdots \quad \alpha_{t-1}]^T$ 为 $t-1$ 次迭代 SMO 算法在 $V_{t-1}$ 上训练所得的最优解，$Q_{t-1}$ 为对应的 Hessian 矩阵，$e_{t-1}$ 为都为 1 的向量。$\alpha_t=0$ 为新增加到 $V_t$ 的样本点所对应的 Lagrange 乘子，则第 $t$ 次迭代采用热启动方式的 SMO 算法起始于以下优化函数 $W$ 和梯度 $\nabla W$：

$$\begin{aligned} W&=\frac{1}{2}\begin{bmatrix}\alpha\\ \alpha_t\end{bmatrix}^T\begin{bmatrix}Q_{t-1} & Q_{t-1,t}\\ Q_{t,t-1} & Q_t\end{bmatrix}\begin{bmatrix}\alpha\\ \alpha_t\end{bmatrix}-\begin{bmatrix}e_{t-1}\\ 1\end{bmatrix}^T\begin{bmatrix}\alpha\\ \alpha_t\end{bmatrix}\\ \nabla W&=\begin{bmatrix}\alpha\\ \alpha_t\end{bmatrix}^T\begin{bmatrix}Q_{t-1} & Q_{t-1,t}\\ Q_{t,t-1} & Q_t\end{bmatrix}-\begin{bmatrix}e_{t-1}\\ 1\end{bmatrix}\end{aligned} \tag{10}$$

其中，

$$\begin{aligned} &Q_{t-1,t}=[y_1y_tk(x_1,x_t) \quad \cdots \quad y_{t-1}y_tk(x_{t-1},x_t)]^T\\ &Q_{t,t-1}=Q_{t-1,t}^T, Q_t=y_ty_tk(x_t,x_t)\end{aligned} \tag{11}$$

$\alpha$ 为 $t-1$ 次迭代 SMO 算法在 $V_{t-1}$ 上训练所得的最优解意味着 $\sum_{i=1}^{t-1}\alpha_i y_i=0$，又因为 $\alpha_t=0$，可以推出 $\sum_{i=1}^{t}\alpha_i y_i=0$，满足式(2)的约束条件，因此，该起始点是可行的。

由于上次 SVM 训练已经达到平衡点，并且此次训练在保持原平衡点基础上只增加一个新的样本点，它对原二次优化问题的影响较小，因此，以 $V_{t-1}$ 上的训练结果为基础，SMO 算法在 $V_t$ 上经过少量迭代即可收敛。

### 2.4 算法描述

AVM 算法采用增量学习的方法来求解近似最优分类超平面，每次迭代中，算法根据式(7)判断当前分类超平面是否是最优分类超平面的$(1-\varepsilon)$近似，如果是，则返回当前的分类超平面作为最终结果，否则，根据式(8)从集合 $U/V_t$ 中选取一最违反 KKT 条件的样本点加入核心集 $V_t$，在新的核心集上重新生成分类超平面。在 2.2 节已经提到，为了节省时间开销，要加入当前核心集的样本点是在集合 $U/V_t$ 一随机抽样的子集中选取，因此，为了保证算法终止时，几乎集合 $U/V$ 中的所有样本点到分类超平面的距离都不小于 $1-\varepsilon$，设定一抽样阈值，如果抽样次数达到抽样阈值，并且根据式(9)从每个抽样子集选取的样本点到分类超平面的距离都不小于 $1-\varepsilon$，则认为集合 $U/V_t$ 中的所有样本点到分类超平面的距离都不小于 $1-\varepsilon$，即满足停机条件，迭代终止，返回当前的分类超平面作为最终结果。AVM 算法的总体框架如下所示：

输入：训练集 $U$，抽样阈值 $T$，SMO 算法参数向量 $Z$，参数 $\varepsilon$

输出：近似分类超平面 $h$

函数：

getHyperPlane($P$,$Z$)：返回采用 SMO 算法对数据集 $P$ 进行训练所获得的分类超平面，SMO 算法的参数设为 $Z$

getHyperPlane_HotStart($P$,$Z$)：返回采用 SMO 算法的热启动方式对数据集 $P$ 进行训练所获得的分类超平面，SMO 算法的参数设为 $Z$

getSampleSet($P$,$l$)：返回通过随机抽样方式从数据集 $P$ 中抽出的子集，该子集包含 $l$ 个元素

getCoreVector($P$,$h$)：返回一样本点 $(x,y)=\min_{(x_i,y_i)\in P} y_i f(x_i)$，其中 $f(x)$ 为分类超平面 $h$ 所对应的函数

getDistance($h$,$(x_t,y_t)$)：返回样本点 $(x_t,y_t)$ 到分类超平面 $h$ 的距离 $y_t f(x_t)$，其中 $f(x)$ 为分类超平面 $h$ 所对应的函数

算法：

```
1) Initialize V; //初始化核心集 V
2) h:= getHyperPlane(V,Z);
3) For t:=1 to |U/V|
4) {
5)   For i:=1 to T
6)   {
7)     S:= getSampleSet(U/V,59);
8)     (x,y):= getCoreVector(S,h);
9)     d:= getDistance(h,(x,y));
10)    If (d≥(1-ε)) then
11)      Continue;
12)    Else
13)      Break;
14)    EndIf
15)  }
16)  If (i>T) then
17)    Break;
18)  Else
19)  {
20)    V:=V∪{(x,y)};
21)    h:= getHyperPlane_HotStart(V,Z);
22)  }
23) EndIf
24)}
25) Return h
```

### 2.5 计算复杂度分析

采用核心集求取最优分类超平面$(1-\varepsilon)$近似解的方法最多经过 $\tilde{\omega}=(G/\rho^*)^2/\varepsilon$ 次迭代后收敛，其中 $G=\max_{(x,y)\in U}\|x\|$，$\rho^*$ 为最优分类超平面所对应的分类间隔[8]。相应地，迭代终止时，核心集的元素最多为 $|V_0|+(G/\rho^*)^2/\varepsilon$，其中 $V_0$ 代表初始核心集合。

在 AVM 算法中，每次迭代所需存储空间主要用于存储对应于当前核心集 $V_t$ 的 Hessian 矩阵，显然，AVM 算法最后一次迭代对应的核心集包含元素最多，所需存储空间最大，因此，该算法的空间复杂度为：

$$O((|V_0|+\tilde{\omega})^2)=O(\tilde{\omega}^2)=O((G/\rho^*)^4/\varepsilon^2) \tag{12}$$

在第 $t$ 次迭代，AVM 算法所耗费的时间主要由两部分组成(假设训练样本全集 $|U|=l$)：

1) 在集合 $U/V_t$ 中寻找当前最违反 KKT 条件的样本点，所需时间最多为 $O(l(|V_0|+t))=O(lt)$

2) 采用 SMO 算法训练当前核心集，所需时间为 $O((|V_0|+t)^3)=O(t^3)$

因此，第 $t$ 次迭代所需时间为 $O(lt+t^3)$，AVM 算法的时间复杂度为：

$$\Lambda=\sum_{t=1}^{\tilde{\omega}}O(lt+t^3)=O(l\tilde{\omega}^2+\tilde{\omega}^4)=O(l(G/\rho^*)^4/\varepsilon^2+(G/\rho^*)^8/\varepsilon^4) \tag{13}$$

由于采用了 2.2 节所描述的抽样方法，因此每次迭代寻找最违反 KKT 条件的样本点实际所需时间最多为：$O(59T(|V_0|+t))$，其中 $T$ 表示抽样阈值。相应地，每次迭代所需时间变更为 $O(t^3)$，AVM 算法的时间复杂度变更为：

$$\Lambda=\sum_{t=1}^{\tilde{\omega}}O(t^3)=O(\tilde{\omega}^4)=O((G/\rho*)^8/\varepsilon^4) \tag{14}$$

根据式(12)，式(14)可知，AVM 算法的时间复杂度和空间复杂度与训练集本身的特性 $G/\rho^*$ 及参数 $\varepsilon$ 有关，而与训练集样本的数量无关，因此具有良好的时间和空间扩展性。根据近似最优分类超平面的定义及式(12)，式(14)可知，当 $\varepsilon$ 减小时，AVM 算法训练所得的分类超平面就越接近最优分类超平面，然而，所需的时间和空间复杂度就越大，因此，参数 $\varepsilon$ 在算法的效率和结果的精度间起折衷作用。

## 3 实验结果及分析

作者在 LIBSVM[10] 的基础上实现了所提出的 AVM 算法，并将它和标准的 SVM 训练算法(采用 LIBSVM 作为标准的 SVM 训练算法)及 CVM 算法进行比较。实验数据集是从 USPS 数据集扩展而来的[5,11]，扩展后的 USPS 为一超大数据集，它包含 266,079 个样本点，测试集包含 75,383 个样本点。本次实验分别采用高斯核函数和多项式核函数，高斯核函数的参数 $g=2^{-8}$，多项式核函数的参数 $d=2$，惩罚算子 $C=10$。惩罚算子 $C$ 及核函数参数都是采用 LIBSVM 在实验数据集中一随机抽取的子集上 10 次交叉测试所得结果的最优值。AVM 和 CVM 算法的参数 $\varepsilon$ 值都设为 $10^{-6}$，这是文献[5]中的建议值，初始核心集样本数定为样本全集的 1%。

本次采用高斯核函数的实验结果如图 2 所示。根据图 2(a)，(c)可知，在 USPS 数据集上，AVM 算法的训练速度略高于 CVM，并且它们二者的训练速度都远远高于 LIBSVM。虽然 AVM 训练所得分类器的分类准确率略低于 LIBSVM 和 CVM，但基本维持了原始分类器的分类精度。这说明了，采用核心集求解近似最优分类超平面可以在基本维持原始分类器泛化性能的同时，极大地提高 SVM 的训练速度。从图 2(b)可以看出，AVM 和 CVM 训练所得的支持向量数远远少于 LIBSVM，并且 AVM 的支持向量数少于 CVM，这是因为 AVM 算法所对应的优化问题采用一次损失函数，而 CVM 算法采用二次损失函数，一次损失函数所对应的 SVM 优化问题具有更少的支持向量数[6]。

从图 2(b)的核心集数量曲线可以看出，AVM 算法收敛所需的迭代次数少于 CVM。CVM 算法将 SVM 训练问题转化为求取特征空间中包含所有样本点的近似最小超球问题，算法实质也是采用核心集求取近似最优解，收敛所需迭代次数也有其理论界[12]。然而，对于此种采用核心集迭代求解近似最优解的算法而言，理论上的界是对于最坏情况的结论，在很多情况下是较松弛的，其实际迭代次数取决于核心集元素的选取方式，一般情况下，远远小于其理论界。CVM 每次迭代选取距离当前超球球心最远的样本点加入核心集，选取公式如下所示(针对第 $t$ 次迭代)[5]：

$$\arg\max_{(x_i,y_i)\in U/V_t}||c_t-\tilde{\varphi}(x_i)||^2=\arg\max_{(x_i,y_i)\in U/V_t}\left(\sum_{(x_p,y_p),(x_q,y_q)\in V_t}\alpha_p\alpha_q\tilde{k}(x_p,x_q)-2\sum_{(x_p,y_p)\in V_t}\alpha_p\tilde{k}(x_p,x_i)+\tilde{k}(x_i,x_i)\right) \tag{15}$$

其中核函数：

$$\tilde{k}(x_i,x_j)=y_iy_jk(x_i,x_j)+y_iy_j+\delta_{ij}/C \tag{16}$$

当 $i=j$ 时，$\delta_{ij}=1$，否则，$\delta_{ij}=0$。$\tilde{\varphi}$ 为对应核函数 $\tilde{k}(.,.)$ 的非线性映射，$c_t=\sum_{(x_s,y_s)\in V_t}\alpha_s\tilde{\varphi}(x_s)$ 表示当前超球的球心。仔细观察式(15)可以发现，在同一迭代阶段，其右边的第一、三项都是固定值(由于 CVM 只能采用各向同性核函数，因此 $\tilde{k}(x_i,x_i)$ 是常数)，因此可以推出：

$$\begin{aligned}&\arg\max_{(x_i,y_i)\in U/V_t}||c_t-\tilde{\varphi}(x_i)||^2\\&=\arg\min_{(x_i,y_i)\in U/V_t}\sum_{(x_p,y_p)\in V_t}\alpha_p\tilde{k}(x_p,x_i)\\&=\arg\min_{(x_i,y_i)\in U/V_t}\sum_{(x_p,y_p)\in V_t}\alpha_py_py_i(k(x_p,x_i)+1)\\&=\arg\min_{(x_i,y_i)\in U/V_t}y_i(w_t^T\varphi(x_i)+b)\\&=\arg\min_{(x_i,y_i)\in U/V_t}y_if_t(x_i)\end{aligned} \tag{17}$$

比较式(8)和式(17)可知，CVM 和 AVM 每次迭代寻找核心集元素的理念是一致的，都是寻找当前最违反 KKT 条件的样本点。换句话说，CVM 算法中，这个距离当前超球球心最远的点也就是其对应 SVM 问题的当前最违反 KKT 条件的样本点，因此，两个算法具有一致的核心集元素选取方式。

两个算法每次迭代选取核心集元素可以看作是一个寻找支持向量的过程，根据以上分析可知，AVM 和 CVM 选取核心集元素的方式是一致的，也就是说二者寻找支持向量的方式是一致的，得益于相应 SVM 优化问题具有较少的支持向量数，AVM 算法找到所有支持向量所需的迭代次数少于 CVM，并且，AVM 在每次迭代采用 SMO 热启动算法来加快训练速度，因此具有更高的训练效率(图 2(a))。另一方面，较少的支持向量意味着 AVM 算法训练所得结果分类器比 CVM 训练所得具有更快的分类速度。

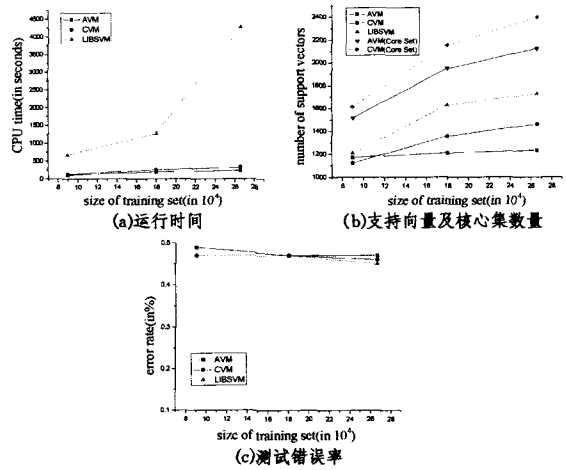


图 2 AVM，CVM，LIBSVM 在扩展 USPS 数据集上的运行结果比较(采用高斯核函数)

采用多项式核函数的实验结果如图3所示。根据图3可知，在USPS数据集上，采用多项式核函数的AVM算法取得了和采用高斯核函数类似的结果。相较于CVM算法只能采用各向同性核函数（例如高斯核函数），AVM算法可以采用任意的核函数，因此具有更宽广的适用范围。

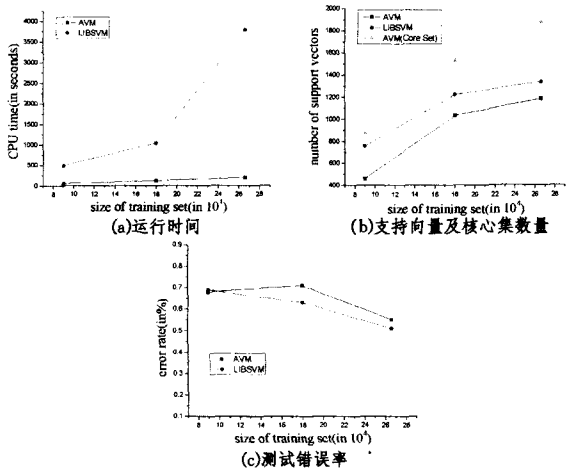


图3 AVM和LIBSVM在扩展USPS数据集上的运行结果比较（采用多项式核函数）

**结束语** 本文提出了一种适用于超大规模数据集的SVM学习算法：AVM，它采用增量学习和近似最优解相结合的方法来解决SVM在超大规模数据集上的训练问题。理论分析和实验结果表明，该算法的计算复杂度与训练样本的数量无关，因此具有良好的时间与空间扩展性。和目前存在的适用于超大规模数据集的SVM训练算法相比，AVM具有如下优势：

1) AVM算法的设计思路并不是单纯依靠经验观察，而是具有坚实的理论基础。

2) AVM算法具有更快的训练速度，并且训练完毕具有更少的支持向量，因此相应的结果分类器具有更快的分类速度。

3) AVM算法对采用何种核函数和损失函数没有限制，因此更易获得稳定的分类性能和具有更宽广的适用范围。

目前的AVM算法只适用于分类问题，将其推广到回归问题是今后研究工作的一部分。

## 参考文献


[1] Vapnik V. Statistical Learning Theory[M]. New York: Wiley, 1998

[2] 边肇祺，张学工，等. 模式识别（第二版）[M]. 北京：清华大学出版社，2000：284-303

[3] Yu H, Yang J, Han J. Classifying large data sets using SVM with hierarchical clusters[C]// Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Washington DC, USA, 2003: 306-315

[4] Erdem Z, Polikar R, Gurgen F S, et al. Ensemble of SVMs for Incremental Learning. In Multiple Classifier Systems, 2005: 246-256

[5] Tsang I W, Kwok J T, Cheung P. Core vector machines : Fast svm training on very large data sets[J]. JMLR, 2005, 6: 363-392

[6] 朱永生，王成栋，张优云. 二次损失函数支持向量机性能的研究[J]. 计算机学报，2003，26(8)：982-989

[7] Badoiu M, Clarkson K. Optimal core - sets for balls [C]// DIMACS Workshop on Computational Geometry. 2002

[8] Har-Peled S, Roth D, Zimak D. Maximum Margin Coresets for Active and Noise Tolerant Learning[C]// Proceedings of the twentieth International joint Conference on Artificial Intelligence. Hyderabad, India, 2007

[9] Li Xuchun, Zhu Yan, Sung E. Sequential bootstrapped support vector machines[J]. IEEE Trans. Neural Netw, 2005, 10(5): 1000-1017

[10] Chang C-C, Lin C-J. LIBSVM: a library for support vector machines[OL]. 2001. http://www.csie.ntu.edu.tw/?cjlin/libsvm

[11] Murphy P M, Aha D W. UCI repository of machine learning databases. Irvine, CA[OL]. http://www.ics.uci.edu/~mlearn/MLRepository.html), 2004

[12] Agarwal P, Har - Peled S, Varadajan K. Geometric approximations via coresets. Manuscript[OL]. http://valis.cs.uiuc.edu/~sariel/papers/04/survey/, 2004


（上接第207页）


[5] Cheung D W, et al. Efficient Mining of Association Rules in Distributed Databases[J]. IEEE Trans. Knowledge and Data Eng., 1996, 8(6): 911-922

[6] Schuster A, Wolff R. Communication-efficient Distributed Mining of Association Rules[C]// Proc. ACM SIGMOD Int'l Conf. Management of Data. ACM Press, 2001: 473-484

[7] Ashrafi M Z. Monash University ODAM: An Optimized Distributed Association Rule Mining Algorithm, IEEE DISTRIBUTED SYSTEMS ONLINE 1541-4922 © 2004[J]. IEEE Computer Society, 2004, 5(3)

[8] Ma Y, Liu B, Wong C K. Web for Data Mining: Organizing and Interpreting the Discovered Rules Using the Web[J]. SIGKDD Explorations, New York: ACM Press, 2000, 2(1): 16-23

[9] Kimball R, Ross M. The Data Warehouse Toolkit, The Complete Guide to Dimensional Modeling. 2nd edn[M]. John Wiley & Sons, New York, 2002

[10] Nestorov S, Jukic N. Ad - Hoc Association - Rule Mining within the Data Warehouse[C]// Abstract Proc. Hawaii Int. Conf. on System Sciences (HICSS'03). IEEE Computer Society Press (2003) 232. http://people.cs.uchicago.edu/~evtimov/pubs/hicss03.pdf

[11] Zaky M J, Parthasarathy S, Ogihara M, et al. New Algorithm for Fast Discovery of Association Rules[R]. No. 261. University of Rochester, 1997

[12] Hand D, Manilla H, Smyth P. Principles of Data Mining[M]. Cambridge-London: MIT Press, 2001