核心向量机的深度研究与创新

吴颖馨[1], 张永停[2]

Abstract

SVM是分类问题中的经典算法，其通过最大化几何间隔来分类。最一般的SVM算法时间复杂度为$O(m^3)$，其中m是训练集大小。当数据集过大时，传统SVM算法需要耗费大量时间来训练。本文基于对SVM时间复杂度和空间复杂度改进的算法CVM, 该算法使用MEB的思想，使用近似值来逼近最优解，在使用概率加速后，该算法时间复杂度与训练集大小无关。

我们通过对包含但不限于课程指定的两篇论文的深刻理解，总结了 CVM 以及参考文献[2]与之对比的 SimpleSVM 的详细算法，之后较为完整地对论文的内容进行复现，并对复现结果进行思考与讨论。最后在复现的基础上，我们对原 CVM 算法进行了三点改进，即初始化方式，随机采样的参数调整，采用 AdaBoost 提高准确率，这三种改进均取得了不错的效果。

Keywords

Support Vector Machine，Minimum Enclosing Ball, Core Vector Machine, AdaBoost

[1] *Department of Big Data, USTC. wuyxin@mail.ustc.edu.cn*

[2] *Department of Computer Science, USTC. zytabcd@mail.ustc.edu.cn*

Contents

1. 算法详细总结

以下将对参考文献[1, 2]总结完整的数学推导。考虑到叙述的完整性，1.1节中将简述经典的 SVM，并推导详细的 L2 norm soft margin SVM 的对偶形式，这将在”Core Vector Machines: Fast SVM Training on Very Large Data Sets”中用到。1.2节将介绍CVM的主要算法，1.3节将讨论 CVM 的收敛性、时空复杂度,1.4节将介绍在论文[2]中与 CVM 进行对比的 SimpleSVM 算法。

1.1 Support Vector Machine

SVM 适用于非参数线性分类问题，所谓非参数机器学习算法，指的是不对目标函数的形式做太多假设。而在没有假设的情况下，算法可以自由地从训练数据中学习函数的任何形式。*SVM* 的另外一大优势在于，将其转化为对偶问题后，分类只需计算支持向量机与少数支持向量之间的距离，在计算复杂核函数时可以大大简化模型和计算。

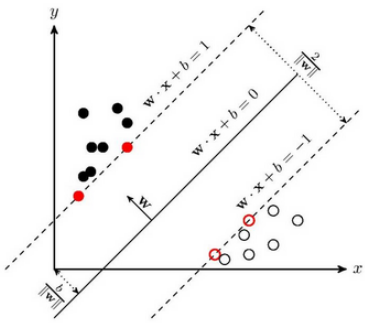

Figure 1. SVM在二维平面中的表示，通过最大化几何间隔来确定最优划分超平面

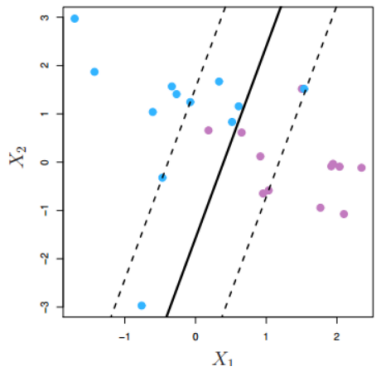

Figure 2. L2 Norm Soft Margin SVM。一方面考虑到数据(即使在高维空间)不一定线性可分，另一方面希望去除部分噪声点。即“允许但不鼓励一部分点跨过超平面”。

首先，SVM是基于线性可分的数据集 X，但是现实世界中，这一假设往往不能被实现。然而可以证明*，$\exists$ 高维度映射 $\phi, X \xrightarrow{\phi} \phi(X)$ 使得 X 线性可分。以下直接用 $\phi(X)$ 表示线性可分的数据集。

为最大化几何间隔，并将目标转换为凸函数形式，可以得到以下优化形式：

$$\begin{aligned} &\min_{\omega,b} \frac{1}{2}\|\omega\|^2 \\ &s.t.\ y_i(\omega^T\phi(x^{(i)})+b) \geq 1,\ i=1,2,...,m \end{aligned} \tag{1}$$

$y^{(i)}$ 为 第i个节点特征 $x^{(i)}$ 对应的label，ω, b 为待学习参数。相应的对偶问题为

$$\begin{aligned} &\min_{\alpha} W(\alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}\phi(x^{(i)})^T\phi(x^{(j)}) \\ &s.t.\ \sum_{i=1}^{m}\alpha_i y^{(i)} = 0, \\ &\alpha_i \geq 0,\ i=1,2,...,m \end{aligned} \tag{2}$$

1.1.1 L2 Norm Soft Margin SVM

在L2正则下，我们需要优化的目标为

$$\begin{aligned} &\min_{\omega,b,\xi} \frac{1}{2}\|\omega\|^2 + \frac{C}{2}\sum_{i=1}^{m}\xi_i^2 \\ &\text{s.t. } y^{(i)}\left(\omega^T x^{(i)}+b\right) \geq 1-\xi_i, \quad i=1,\ldots,m \end{aligned}$$

对上述限制采用拉普拉斯算子重新表述为

$$\begin{aligned} &\mathscr{L}(\omega,b,\xi,\alpha) \\ &= \frac{1}{2}\omega^T\omega + \frac{C}{2}\sum_{i=1}^{m}\xi_i^2 - \sum_{i=1}^{m}\alpha_i\left[y^{(i)}\left(\omega^T x^{(i)}+b\right)-1+\xi_i\right] \end{aligned}$$

其中$\alpha_i \geq 0, i=1,...,m$。注意到$\xi \geq 0$的限制在L2 Norm问题上已经不需要了。也就是说，无论这些约束条件是否合理，目标的最优值都是相同的。

接下来，对$\mathscr{L}(\omega,b,\xi,\alpha)$分别关于$\omega$, b, ξ求导，可得

$$0 = \nabla_\omega \mathscr{L} = \omega - \sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}$$

$$0 = \frac{\partial \mathscr{L}}{\partial b} = -\sum_{i=1}^{m}\alpha_i y^{(i)}$$

$$0 = \nabla_\xi \mathscr{L} = C\xi - \alpha$$

整理得：

$$w = \sum_{i=1}^{m}\alpha_i y^{(i)}x^{(i)}$$

$$0 = \sum_{i=1}^{m}\alpha_i y^{(i)}$$

$$0 = C\xi_i - \alpha_i \quad \Rightarrow \quad C\xi_i = \alpha_i$$

将上述三个约束条件代入$\mathscr{L}(\omega,b,\xi,\alpha)$中

$$\begin{aligned}
&\min_{\omega,b,\xi}\mathscr{L}(\omega,b,\xi,\alpha)=\\
&\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\left(\alpha_i y^{(i)}x^{(i)}\right)^T\left(\alpha_j y^{(j)}x^{(j)}\right)+\frac{1}{2}\sum_{i=1}^{m}\frac{\alpha_i}{\xi_i}\xi_i^2\\
&-\sum_{i=1}^{m}\alpha_i\left[y^{(i)}\left(\left(\sum_{j=1}^{m}\alpha_j y^{(j)}x^{(j)}\right)^T x^{(i)}+b\right)-1+\xi_i\right]\\
=&-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}\left(x^{(i)}\right)^T x^{(j)}+\frac{1}{2}\sum_{i=1}^{m}\alpha_i\xi_i\\
&-\left(\sum_{i=1}^{m}\alpha_i y^{(i)}\right)b+\sum_{i=1}^{m}\alpha_i-\sum_{i=1}^{m}\alpha_i\xi_i\\
=&\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}\left(x^{(i)}\right)^T x^{(j)}-\frac{1}{2}\sum_{i=1}^{m}\alpha_i\xi_i\\
=&\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}\left(x^{(i)}\right)^T x^{(j)}-\frac{1}{2}\sum_{i=1}^{m}\frac{\alpha_i^2}{C}
\end{aligned}$$

最终，我们得到了L2 Norm Soft Margin SVM的对偶形式

$$\begin{aligned}
&\max_{\alpha}\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}\left(x^{(i)}\right)^T x^{(j)}-\frac{1}{2}\sum_{i=1}^{m}\frac{\alpha_i^2}{C}\\
&\text{s.t. } \alpha_i\geq 0, i=1,\ldots,m\\
&\sum_{i=1}^{m}\alpha_i y^{(i)}=0
\end{aligned}\tag{3}$$

1.1.2 Sequential Minimal Optimization

Platt提出SMO以优化该对偶问题，算法思想十分清晰简洁。

Repeat{
Select α_i,α_j
Hold all α_k fixed, except α_i,α_j
Optimize W(α) w.r.t. α_i,α_j
(subject to all the constraints).
}

该方法和Coordinate Ascent Algorithm很类似，Coordinate Ascent Algorithm每次通过更新多元函数中的一维，经过多次迭代直到收敛来达到优化函数的目的。简单的讲就是不断地选中一个变量做一维最优化直到函数达到局部最优点。由于在SVM中，我们的α并不是完全独立的，而是具有约束的，即

$$\sum_{i=1}^{m}\alpha_i y^{(i)}=0$$

因此一个α_i 改变，另一个α_j也要随之变化以满足条件。关于α_i,α_j的选取也有许多改进的算法。

1.1.3 SVM的局限性

用 m 表示训练集的大小，对SVM进行训练需要 $O(m^3)$ 的时间复杂度和 $O(m^2)$ 的空间复杂度。而这二者的主要来源于1.1节中的二次最优化(QP)问题。以下总结了部分论文[1] 中对许多尝试在QP上进行效率改进的算法及其缺陷：

- Nystrom method , greedy approximation or matrix decompositions: 采用在kernel matrix中得到低秩近似，进而减少计算量的思路。但在大数据集上仍有较高的秩。
- chunking: 主要是将矩阵的维数从训练样本的平方维数降到非0的拉格朗日算子的平方这个维数。但如果训练样本过大，特征过多，还是解决不了矩阵太大，计算时内存溢出之类的问题。
- Osuna：实质上是一种分解算法，把一个大型QP问题分解成一个小型QP问题组成的序列，在上一步的QP子问题中至少加入一个违反KKT条件的点而构成下一步的QP子问题。每一步优化目标函数的值并始终保持满足约束条件，从而使这些QP子问题构成的序列收敛，直至最后所有的点都满足KKT条件，也即得到了原问题的解（SMO 可以看成是把 Osuna 的思想极端化, 即每次只考虑两个点及其约束条件）。
- Lagrangian SVM: SVM给大家带来一个普遍的误解是转换成对偶问题，然后直接求解。LSVM打破了这种常规思路，而是从KKT条件中找到解的循环公式。但是，按照原论文的说法，它对于非线性核仍需要求完整的kernel matrix的逆。
- Parallel Mixture: 提出了一种很容易地实现并行的混合支持向量机，但其在理论上没有保证，因此缺乏解释性。

1.2 Core Vector Machine

1.2.1 CVM的思路

在此先对 CVM 的思路进行总结，希望读者能在后面的推导中能更关注于 CVM 思想的本质，而不只是流于公式表面。

CVM 采用的仍然是核方法。我们知道，核函数是高维空间的内积，或者说是低维空间中的内积的某个函

数。具体来说，在线性不可分的情况下，支持向量机首先在低维空间中完成计算，然后通过核函数将输入空间映射到高维特征空间，最终在高维特征空间中构造出最优分离超平面，从而把平面上本身不好分开的非线性数据分开。既然核函数可以看成是低维与高维空间的一种联系（注意核函数并不是映射，只是映射的内积），那么能否通过改变核函数的形式，从而改变高维空间的结构？

就 SVM 的原始问题(见1.1节式(3))，我们实际上需要的是在 m 维空间的一个子空间中，找到由该最大化目标所确定的超平面的最优解。这样的高维空间可视为 grid-like 。

Minimum Enclosing Ball(MEB)，详见参考文献[4]，采用的是一个简单的迭代, 即在第t次迭代, 包括 $B(C_t,R_t)$ 外的最远点，将目前的得到的 Ball 逐步扩展，具体的特征空间可视为一个凸球状空间。

我们认为，CVM 的本质在于，通过恰当的 kernel 形式，将 SVM 的原始问题求解的 grid-like 高维空间，转化为更适合高效迭代求解 ball-like 高维空间。不仅如此，现实世界中数据往往呈团状或簇状分布，这对于问题在 ball-like 空间中求解也是有一定优势的。

1.2.2 符号表示

以下给出推导过程中的符号表示，便于读者查看

Variable	Meaning
k	核函数
ϕ	特征映射函数
c	MEB 球中心
R	MEB 球半径
α	$\mathbb{R}^m$, 拉格朗日乘子
$K_{m\times m}$	$k[x_i,x_j]$，核矩阵
$\mathbf{x}_i$	特征向量
y_i	类别标签
$\mathbf{z}_i$	$(\mathbf{x}_i,y_i)$
$\mathbf{y}$	$[y_1,\ldots,y_m]'$
S_t	t时刻的核心集

1.2.3 Minimum Enclosing Ball

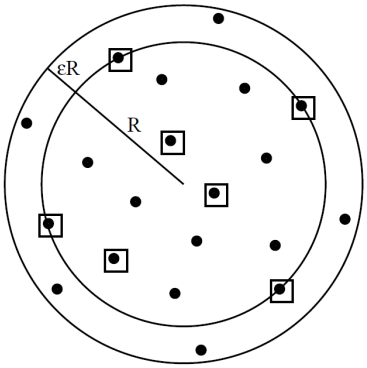

Figure 3. 最小闭包球问题，Badoiu et al. 通过迭代逐次拓展的方式，得到了该问题的$(1+\varepsilon)$近似解。以上方框中所有点的集合称为core-set。

SVDD与MEB在数据只有一类标签时是统一的，其通过训练出一个最小的超球面，将这堆数据全部“包起来”，识别一个新的数据点时，如果这个数据点落在超球面内，就属于这个类，否则不是。 MEB基于以下最小化问题：

$$\min R^2 : \|\mathbf{c}-\phi(\mathbf{x}_i)\|^2 \le R^2, i=1,\ldots,m \tag{4}$$

以下将其转换为对偶问题：

$$\mathscr{L}(R,c,\alpha) = R^2 - \sum_{i=1}^{m}\alpha_i\left[R^2 - \| c-\phi(\mathbf{x}_i)\|^2\right]$$

其中 $\alpha_i \ge 0, i=1,\ldots,m$。对$\mathscr{L}(R,c,\alpha)$分别关于$R,c$求导，可得

$$0 = \nabla_R\mathscr{L} = 2R(1-\sum_{i=1}^{m}\alpha_i) \Rightarrow \sum_{i=1}^{m}\alpha_i = 1$$

$$0 = \nabla_c\mathscr{L} = \nabla_c[R^2 - \sum_{i=1}^{m}\alpha_i\left(c^T-\phi^T(\mathbf{x}_i)\right)\left(c-\phi(\mathbf{x}_i)\right)]$$
$$\Rightarrow c = \sum_{i=1}^{m}\alpha_i\phi(\mathbf{x}_i)$$

反代入$\mathscr{L}$, 整理得

$$\begin{aligned}
&\min_{R,C}\mathscr{L}(R,c,\alpha)=R^2-\sum_{i=1}^{m}\alpha_i\times\\
&\left[R^2-\left(\sum_{j=1}^{m}\alpha_i\phi^T(\mathbf{x}_j)-\phi^T(\mathbf{x}_i)\right)\left(\sum_{k=1}^{m}\alpha_i\phi(\mathbf{x}_k)-\phi(\mathbf{x}_i)\right)\right]\\
&=\sum_{i=1}^{m}\alpha_i\left(\sum_{j=1}^{m}\alpha_i\phi^T(\mathbf{x}_j)-\phi^T(\mathbf{x}_i)\right)\left(\sum_{k=1}^{m}\alpha_i\phi(\mathbf{x}_k)-\phi(\mathbf{x}_i)\right)\\
&=\sum_{i=1}^{m}\alpha_i\phi^T(\mathbf{x}_i)\phi(\mathbf{x}_i)-\sum_{i,j=1}^{m}\alpha_i\alpha_j\phi^T(\mathbf{x}_i)\phi(\mathbf{x}_j)\\
&=\sum_{i=1}^{m}\alpha_i k(\mathbf{x}_i,\mathbf{x}_i)-\sum_{i,j=1}^{m}\alpha_i\alpha_j k(\mathbf{x}_i,\mathbf{x}_j)
\end{aligned}$$

可以得到最后的对偶问题，即参考文献[1]的第二式中的矩阵形式

$$\begin{aligned}
&\max_{\alpha_i}\sum_{i=1}^{m}\alpha_i k(\mathbf{x}_i,\mathbf{x}_i)-\sum_{i,j=1}^{m}\alpha_i\alpha_j k(\mathbf{x}_i,\mathbf{x}_j)\\
&\text{s.t. }\alpha_i\geq 0,\quad i=1,\ldots,m\\
&\qquad\sum_{i=1}^{m}\alpha_i=1\\
&\max_{\alpha}\alpha'\operatorname{diag}(\mathbf{K})-\alpha'\mathbf{K}\alpha:\alpha\geq 0,\alpha'\mathbf{1}=1
\end{aligned}\tag{5}$$

c的解已经在前述给出，另解得

$$R=\sqrt{\alpha'\operatorname{diag}(\mathbf{K})-\alpha'\mathbf{K}\alpha}$$

1.2.4 CVM问题形式化

在 1.2.2 节已经说过，CVM 通过恰当的 kernel 形式，将 SVM 的原始问题求解转化为更适合高效迭代求解 MEB 问题。特别地，CVM 原论文作者发现，当满足 $k(\mathbf{x},\mathbf{x})=\kappa$(常数) 时，可以得到 One-Class L2-SVM、Two-Class L2-SVM 与 MEB 对偶问题形式上的统一。

One-Class L2-SVM 传统意义上，很多的分类问题试图解决两类或者多类情况，机器学习应用的目标是采用训练数据，将测试数据属于哪个类进行区分。但是如果只有一类数据，目标便是测试新的数据并且检测它是否与训练数据相似。方法是创建一个参数为 ω,b 的超平面，该超平面与特征空间中的零点距离最大，并且将零点与所有的数据点分隔开。参考论文[1]中将其加入了软间隔后的优化目标为：

$$\min_{\omega,\rho,\xi_i}\|\omega\|^2-2\rho+C\sum_{i=1}^{m}\xi_i^2:\omega'\varphi(\mathbf{x}_i)\geq\rho-\xi_i\tag{6}$$

同样地，将其写成拉格朗日乘子形式

$$\begin{aligned}
&\mathscr{L}(\omega,\rho,\xi_i,\alpha)\\
&=\omega^T\omega-2\rho+C\sum_{i=1}^{m}\xi_i^2-2\sum_{i=1}^{m}\alpha_i(\omega^T\phi(\mathbf{x}_i)-\rho+\xi_i)
\end{aligned}$$

注意到这里拉格朗日乘子写作$2\alpha_i$不影响求解。

$$0=\nabla_\omega\mathscr{L}=2\omega-2\sum_{i=1}^{m}\alpha_i\phi(\mathbf{x}_i)\Rightarrow\omega=\sum_{i=1}^{m}\alpha_i\phi(\mathbf{x}_i)$$

$$0=\nabla_\rho\mathscr{L}=-2+2\sum_{i=1}^{m}\alpha_i\Rightarrow\sum_{i=1}^{m}\alpha_i=1$$

$$0=\nabla_{\xi_i}\mathscr{L}=2C\sum_{i=1}^{m}\xi_i-2\sum_{i=1}^{m}\alpha_i\Rightarrow\sum_{i=1}^{m}\xi_i=\frac{1}{C}$$

回代（不再赘述），可以得到最终形式

$$\begin{aligned}
&\max-\alpha'\left(\mathbf{K}+\frac{1}{C}\mathbf{I}\right)\alpha:0\leq\alpha,\alpha'\mathbf{1}=1\\
&=\max-\alpha'\tilde{\mathbf{K}}\alpha:0\leq\alpha,\alpha'\mathbf{1}=1
\end{aligned}\tag{7}$$

其中$\tilde{k}(\mathbf{z},\mathbf{z})=\kappa+\frac{1}{C}\equiv\tilde{\kappa},\tilde{\mathbf{K}}=\left[\tilde{k}(\mathbf{z}_i,\mathbf{z}_j)\right]$。这样，就完美地把单类别SVM问题转化成了在满足$k(\mathbf{x},\mathbf{x})=\kappa$下的 MEB 最优解问题，即

$$\max-\alpha'\mathbf{K}\alpha:0\leq\alpha,\alpha'\mathbf{1}=1\tag{8}$$

注意到MEB与L2-SVM具有同样的参数α, 而求出了参数α，实际上也就确定了SVM问题的解。

Two-Class L2-SVM 相似地，为了最优化

$$\begin{array}{ll}\min_{\omega,b,\rho,\xi_i} & \|\omega\|^2+b^2-2\rho+C\sum_{i=1}^{m}\xi_i^2\\ \text{s.t.} & y_i(\omega'\varphi(\mathbf{x}_i)+b)\geq\rho-\xi_i\end{array}\tag{9}$$

将其转化为对偶问题

$$\begin{aligned}
&\max_{0\leq\alpha}-\alpha'\left(\mathbf{K}\odot\mathbf{y}\mathbf{y}'+\mathbf{y}\mathbf{y}'+\frac{1}{C}\mathbf{I}\right)\alpha:\alpha'\mathbf{1}=1\\
&=\max-\alpha'\tilde{\mathbf{K}}\alpha:0\leq\alpha,\alpha'\mathbf{1}=1
\end{aligned}$$

此处的 kernel 具体定义为$\tilde{k}(\mathbf{z}_i,\mathbf{z}_j)=y_iy_jk(\mathbf{x}_i,\mathbf{x}_j)+y_iy_j+\frac{\delta_{ij}}{C}$。读者可以自行验证，此处符合条件$k(\mathbf{x},\mathbf{x})=$常量。

由此，问题的所有形式化如上，我们已经在计算上证明了MEB 与 SVM 核函数满足一定条件下的等价性，这种条件并不严苛。事实上，许多很常见的核如高斯核，点积核，规范化核都是满足这个条件的，由此可以看出 CVM 架构的通用性。

1.2.5 CVM具体求解过程

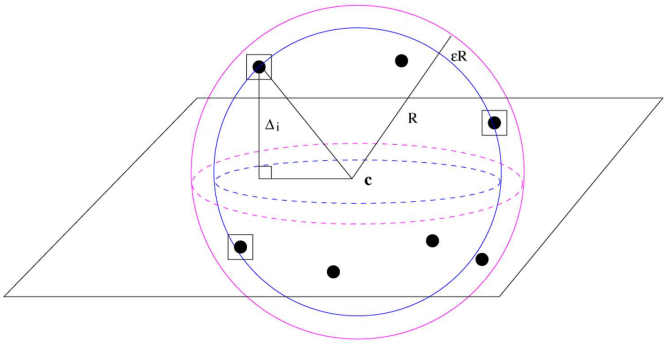

Figure 4. 三维下CVM的直观认识，方框中的点对Minimum Enclosing Ball的形成起到了关键作用。算法一直迭代直到$(1+\varepsilon)R$包含所有的点，最终得到的拉格朗日乘子作为MEB与SVM问题的共同参数，从而求解最终的SVM问题。

以下为Two-Class CVM算法框架，(One-Class CVM只在加入最远点的规则上有所区别)其中需要解释两个地方：

(1) Main 函数中的 while 循环的判断条件需要对没有被标记过是否在 $B(\mathbf{c}_t,(1+\varepsilon)R_t$ 中的点计算距离，具体为

$$\begin{aligned}\left\|\mathbf{c}_t-\tilde{\phi}\left(\mathbf{z}_\ell\right)\right\|^2=&\sum_{\mathbf{z}_i,\mathbf{z}_j\in S_t}\alpha_i\alpha_j\tilde{k}\left(\mathbf{z}_i,\mathbf{z}_j\right)-\\&2\sum_{\mathbf{z}_i\in S_t}\alpha_i\tilde{k}\left(\mathbf{z}_i,\mathbf{z}_\ell\right)+\tilde{k}\left(\mathbf{z}_\ell,\mathbf{z}_\ell\right)\end{aligned}\tag{10}$$

(2) 对于two-class分类问题而言，Add_Furthest_Point在(11)式第一个等价符号中结合(10)，在第二个等价号中利用

$$\tilde{k}\left(\mathbf{z}_i,\mathbf{z}_j\right)=y_iy_jk\left(\mathbf{x}_i,\mathbf{x}_j\right)+y_iy_j+\frac{\delta_{ij}}{C}$$

最后一个等价号中利用

$$\omega=\sum_{i=1}^m\alpha_iy_i\phi\left(\mathbf{x}_i\right),\quad b=\sum_{i=1}^m\alpha_iy_i$$

从而有

$$\begin{aligned}&\arg\max_{\mathbf{z}_\ell\notin B_t}\left\|\mathbf{c}_t-\tilde{\phi}\left(\mathbf{z}_\ell\right)\right\|^2\\&\Leftrightarrow\arg\max_{\mathbf{z}_\ell\notin B_t}-2\textstyle\sum_{\mathbf{z}_i\in S_t}\alpha_i\tilde{k}\left(\mathbf{z}_i,\mathbf{z}_\ell\right)+\tilde{k}\left(\mathbf{z}_\ell,\mathbf{z}_\ell\right)\\&\Leftrightarrow\arg\max_{\mathbf{z}_\ell\notin B_t}-2\textstyle\sum_{\mathbf{z}_i\in S_t}\left[\alpha_iy_iy_\ell k\left(\mathbf{x}_i,\mathbf{x}_\ell\right)+\alpha_iy_iy_\ell+\alpha_i\frac{\delta_{i\ell}}{C}\right]+\\&\quad k\left(\mathbf{x}_l,\mathbf{x}_l\right)+\tfrac{1}{C}\\&\Leftrightarrow\arg\max_{\mathbf{z}_\ell\notin B_t}-2\textstyle\sum_{\mathbf{z}_i\in S_t}\left[\alpha_iy_iy_\ell k\left(\mathbf{x}_i,\mathbf{x}_\ell\right)+\alpha_iy_iy_\ell\right]\\&\Leftrightarrow\arg\min_{\mathbf{z}_\ell\notin B_t}\textstyle\sum_{\mathbf{z}_i\in S_t}\alpha_iy_iy_\ell\left(k\left(\mathbf{x}_i,\mathbf{x}_\ell\right)+1\right)\\&\Leftrightarrow\arg\min_{\mathbf{z}_\ell\notin B_t}y_\ell\left(\omega'\phi\left(\mathbf{x}_\ell\right)+b\right)\end{aligned}\tag{11}$$

算法 1 Core Vector Machine

输入： kernel $\tilde{k}$, 特征空间 $\tilde{\mathscr{F}}$, 映射$\tilde{\phi}$, 常数 $\tilde{\kappa}$

输出： c_B,R_B， 核心集 S

function INITIALIZE($\tilde{\mathscr{F}}$, $\tilde{\phi}$)
  随机选取点 $z\in\tilde{\mathscr{F}}$
  选取在 $\tilde{\mathscr{F}}$中离 z最远的点 z_a
  选取在 $\tilde{\mathscr{F}}$中离 z_a最远的点 z_b
  $R_0:=\frac{1}{2}\|\tilde{\phi}\left(\mathbf{z}_a\right)-\tilde{\phi}\left(\mathbf{z}_b\right)\|$
  $=\frac{1}{2}\sqrt{\left\|\tilde{\phi}\left(\mathbf{z}_a\right)\right\|^2+\left\|\tilde{\phi}\left(\mathbf{z}_b\right)\right\|^2-2\tilde{\phi}\left(\mathbf{z}_a\right)'\tilde{\phi}\left(\mathbf{z}_b\right)}$
  $=\frac{1}{2}\sqrt{2\tilde{\kappa}-2\tilde{k}\left(\mathbf{z}_a,\mathbf{z}_b\right)}$
  $\mathbf{c}_0:=\frac{1}{2}\left(\tilde{\phi}\left(\mathbf{z}_a\right)+\tilde{\phi}\left(\mathbf{z}_b\right)\right)$
  $S_0:=\{\mathbf{z}_a,\mathbf{z}_b\}$
  return R_0,c_0,S_0
end function

function ADD_FURTHEST_POINT($\tilde{\mathscr{F}}$, ω, b, ϕ)
  return $\arg\min_{\mathbf{z}_\ell\notin B(\mathbf{c}_t,(1+\varepsilon)R_t)}y_\ell\left(\omega'\phi\left(\mathbf{x}_\ell\right)+b\right)$
end function

function MEB($\tilde{\mathscr{F}}$, $\tilde{\phi}$, S_{t+1})
  (With warm Start)
  $\alpha=SOLVE$(QP problem in S_t)
  $\mathbf{c}=\sum_{i=1}^m\alpha_i\tilde{\phi}\left(\mathbf{x}_i\right)$
  $R=\sqrt{\alpha'\mathrm{diag}(\mathbf{K})-\alpha'\mathbf{K}\alpha}$
return $\mathbf{c},R$
end function

function MAIN($\tilde{\mathscr{F}}$,$\tilde{k}$, $\tilde{\phi}$, $\tilde{\kappa}$)
  R_0,c_0,S_0 = INITIALIZE($\tilde{\mathscr{F}}$, $\tilde{\phi}$)
  $t=0$
  while $\exists\,\tilde{\phi}(\mathbf{z})$不包含在$B\left(\mathbf{c}_t,(1+\varepsilon)R_t\right)$中* **do**
    z = ADD_FURTHEST_POINT($\tilde{\mathscr{F}}$, ω, b, ϕ)*
    $S_{t+1}=S_t\cup\{\mathbf{z}\}$
    $\mathbf{c}_{t+1},R_{t+1}=$ MEB($\tilde{\mathscr{F}}$, $\tilde{\phi}$, S_{t+1})
    $t=t+1$
  end while
return c_B,R_B,S
end function

1.2.6 CVM概率加速方法

原始的CVM算法在每次迭代都需要计算最远的点，计算最远点的复杂度为$O(m|S_t|)$，当 m 非常大的时候，这是一笔很大的开销。 原论文的做法是取 S_t 的一个大小为 59 的子集，求该子集的最远点[8]。该参数的理论保证是该子集最远点有 95% 的概率在 S_t 中前5%个最远的点中，具体证明细节我们在3.2节将会详细描述。

1.3 时空复杂度

1.3.1 时间复杂度

首先，有如下引理:

Lemma 1. *若 $B_{c,r}$ 是 $\mathbf{R}^d$ 上 $S = B_1, ..., B_n (n \geq d+1)$ 的最小闭包球，则任意包含该最小闭包球球心 c 的闭半空间包含一个 B_i 中的距离球心 c 为 r 的点。 [12]*

Proof. 我们可以不妨设每个 S 中的球都与 $\partial B_{c,r}$ 相交。若否,任意严格属于 $B_{c,r}$ 的球可以被删除，且不影响 $B_{c,r}$ 的最优性。

更进一步的，显然存在 $S' \subset S$,其中 $|S'| \leq d+1$, s.t. $MEB(S') = MEB(S)$,且此时圆心 c 一定在由 S' 球心构成的凸包 Q 中[1]。

则对任意包含 c 的闭半空间 H,由于 $c \in Q$,则 H 至少包含 Q 的一个顶点，不妨记为 c',设 c' 是球 $B_{c',r'}$ 的球心。

记

$$p = \overrightarrow{cc'} \cap \partial B_{c,r}$$
$$q = \overrightarrow{cc'} \cap \partial B_{c',r'}$$

则有 $\|cp\| = r$

由三角不等式可知，$B_{c',r'} \backslash \{q\}$的所有点与$c$的距离至多为 q, 另一方面，由假设，$B_{c',r'}$与$\partial B_{c,r}$相交，从而$p = q$ □

Corollary 1. *对于 $\mathbf{R}^d$ 中的点集 T,若其最小闭包球为 $B(T)$，则任意通过 $c_{B(T)}$ 的闭半空间包含一个 T 中距离 $c_{B(T)}$ 距离为 $r_{B(T)}$ 的点*

Remark 1. *任意距离 $c_{B(T)}$ 距离为 K 的点 z，至少存在一个 T 中的点 t 使得 t 与 z 之间的距离至少 为 $\sqrt{r_{B(T)}^2 + K^2}$*

Therom 1. *存在点集 P 的核心集 S, S 的大小为 $\lceil 2/\varepsilon \rceil$。即，任意 P 中点与 $c_{B(S)}$ 的距离小于等 于$(1+\varepsilon) r_{B(P)}$[10]*

Proof. 取 P 中的任意点 p, 记 $S_0 = p$。记 $r_{B(S_i)}$ 为 r_i, $c_{B(S_i)}$ 为 c_i.

取 P 中距离 c_i 最远的点并加入 S_i, 记 S_{i+1} 为 $S_i \cup \{q\}$。重复以上过程至少 $2/\varepsilon$ 次。

记$c = c_{B(P)}$， $R = r_{B(P)}$， $\hat{R} = (1+\varepsilon)R$， $\lambda_i = r_i/\hat{R}_i$, $d_i = \|c - c_i\|$， $K_i = \|c_{i+1} - c_i\|$

若此时所有点距离 c 的距离都不超过 $\hat{R}$， 那么证明完成。若否，此时 P 中距离 c_i 最远的点有 $\|q - c_i\| > \hat{R}$ 。由三角不等式，我们有

$$\hat{R} < \|q - c_i\| \leq \|q - c_{i+1}\| + \|c_{i+1} - c_i\| \leq r_{i+1} + K_i$$

故 $r_{i+1} > \hat{R} - K_i$。由Remark1，将 S_i 当作 T, c_{i+1}当作 z，故存在 S_i 的一个点与 c_{i+1} 的距离至少为 $\sqrt{r_i^2 + K_i^2}$ 。从而我们得到

$$\lambda_{i+1}\hat{R} = r_{i+1} \geq max(\hat{R} - K_i, \sqrt{\lambda_i^2 \hat{R}^2 + K_i^2})$$

当 $\hat{R} - K_i = \sqrt{\lambda_i^2 \hat{R}^2 + K_i^2}$ 时，我们可以得到上式的一个相对最小的下界，即

$$\hat{R}^2 - 2K_i\hat{R} + K_i^2 = \lambda_i^2 \hat{R}^2 + K_i^2$$

，故可知，

$$K_i = \frac{(1 - \lambda_i^2)\hat{R}}{2}$$

由前对$\lambda_{i+i}\hat{R}$的估计，可得

$$\lambda_{i+1} \geq \frac{\hat{R} - \frac{(1-\lambda_i^2)\hat{R}}{2}}{\hat{R}} = \frac{1 + \lambda_i^2}{2}$$

令$\gamma_i = \frac{1}{1-\lambda_i}$，从而可得

$$\gamma_{i+1} \geq \frac{\gamma_i}{1 - 1/(2\gamma_i)} = \gamma_i \left(1 + \frac{1}{2\gamma_i} + \frac{1}{4\gamma_i^2} \cdots\right) \geq \gamma_i + 1/2$$

由$\lambda_0 = 0$，我们可知$\gamma_0 = 1$，从而$\gamma_i \geq 1 + i/2$，故$\lambda_i \geq 1 - \frac{1}{1+i/2}$。

为了使得$\lambda_i \geq \frac{1}{1+\varepsilon}$，可令$i \geq 2/\varepsilon$。

故当$i \geq 2/\varepsilon$时候，必已经满足定理要求。若否，则有$r_i = \lambda_i R > R$，但$S_i \subset P$, 故应有 $r_i \leq R$，矛盾

□

从而CVM方法迭代次数$\tau = O(1/\varepsilon)$。初始化花费时间$O(m)$。每次迭代距离计算花费$O((t+2)^2 + tm) = (t^2 + tm)$，寻找MEB花费$O((t+2)^3) = O(t^3)$，从而一次迭代花费时间为$O(t^3 + tm)$。故总的时间花费为

$$T = \sum_{t=1}^{\tau} O\left(tm + t^3\right) = O\left(\tau^2 m + \tau^4\right) = O\left(\frac{m}{\varepsilon^2} + \frac{1}{\varepsilon^4}\right)$$

可以看出，在不使用前述的概率加速的方法时，时间复杂度与数据集大小成正比例关系。下面考虑使用概率加速的CVM的时间复杂度

Therom 2. 存在点集P的核心集S,S的大小为$O(1/\varepsilon^2)$。即，任意P中点与$c_{B(S)}$的距离小于等于$(1+\varepsilon)r_{B(P)}$

Proof. 记Δ为P的直径。取P中任意一点x，取P中距离x最远的点y，记$S_0=x,y$，显然有$\|x-y\|\geq\Delta/2$。$r_i,c_i,\hat{R}$定义同定理一。则可知$r_0\geq\Delta/4$。
若此时所有点距离c的距离都不超过$\hat{R}$， 那么证明完成。若否，则存在一个点$p\in P$使得$\|p-c_i\|\geq\hat{R}$, 令$S_{i+1}=S_i\cup p$ (这里是和thm1不同的地方)。

Claim 1. $r_{i+1}\geq\left(1+\frac{\varepsilon^2}{16}\right)r_i$ *[9]*

Proof. 若$\|c_i-c_{i+1}\|<(\varepsilon/2)r_i$，由三角不等式，有

$$\|p-c_{i+1}\|\geq\|p-c_i\|-\|c_i-c_{i+1}\|\geq(1+\varepsilon)r_i-\frac{\varepsilon}{2}r_i$$

从而$\|p-c_{i+1}\|\geq\left(1+\frac{\varepsilon}{2}\right)r_i$。
若$\|c_i-c_{i+1}\|\geq(\varepsilon/2)r_i$，由Remark 1，可知存在一个点$x$，其中$x$与$c_i$的距离为$r_i$，有

$$r_{i+1}\geq\|c_{i+1}-x\|\geq\sqrt{r_i^2+\frac{\varepsilon^2}{4}r_i^2}\geq\left(1+\frac{\varepsilon^2}{16}\right)r_i$$

□

由于初始时$r_0\geq\Delta/4$，且由Claim1 知每次迭代半径增加至少$(\Delta/4)\varepsilon^2/16=\Delta\varepsilon^2/64$，从而至多迭代$64/\varepsilon^2$次。

□

从而使用概率的方法时候，迭代次数的上界是$O(1/\varepsilon^2)$。故时间复杂度为

$$T-\sum_{t=1}^{\tau}O\left(t^3\right)-O\left(\tau^4\right)-O\left(\frac{1}{\varepsilon^8}\right)$$

此时时间复杂度与数据集大小无关。

1.3.2 空间复杂度

首先分析没有概率加速的CVM。由于每次迭代需要的空间为 $O\left(|S_t|^2\right)$,且$\tau=O(1/\varepsilon)$，可知空间复杂度为$O(1/\varepsilon^2)$。
对于使用概率加速的CVM，仅与上有迭代次数的不同，即将$\tau=O(1/\varepsilon)$ 改为$\tau=O(1/\varepsilon^2)$

1.4 SimpleSVM

1.4.1 基本思想

对于一个优化问题，先寻找子集的一个最优解，之后每次向子集中添加一个点，然后继续寻找最优解。
为了方便叙述，我们如下定义变量:

Variable	Meaning
x_i	特征向量
y_i	类别标签
P	全部特征向量

对于一个分类问题，我们可以表达为

$$f(x)=\langle\mathbf{w},\Phi(x)\rangle+b=\sum_{j\in A}\alpha_j y_j k(x_j,x)+b$$

对于硬边界的SVM问题，所有的支持向量满足$y_i f(x_i)=1$,该算法可以很简单的写为

算法 2 Hard Margin SimpleSVM

输入: 数据集P
输出: $A,\{\alpha_i\ for\ i\in A\}$
寻找P中不同类别距离最近的两个点(x_{i+},x_{i-})
$A\leftarrow\{i_+,i_-\}$
计算f与α
while 存在点$x_v\notin A$，即$y_v f(x_v)<1$ **do**
$A\leftarrow A\cup\{v\}$
重新计算f和α 并且将A中不是支持向量的点移出A
end while

1.4.2 SimpleSVM算法

考虑一个general的带约束的优化问题

$$\begin{array}{ll}\underset{\alpha}{\text{minimize}} & \frac{1}{2}\alpha^\top H\alpha+c^\top\alpha\\ \text{subject to} & 0\leq\alpha_i\leq C\text{ and }A\alpha=b\end{array}$$

其中$H\in\mathbb{R}^{m\times m}$是正半定矩阵，易知soft-SVM可写成如上形式
变量说明如下

Variable	Meaning
S_0	$\{i\|\alpha_i=0\}$
S_C	$\{i\|\alpha_i=C\}$
S_{work}	$\{i\|\alpha_i\in(0,C)\}$
H_{ww}	仅由S_{work}中的项构成的H的子矩阵
H_{wC}	是由S_{work}构成的行和S_C的列组成的H的子矩阵
c_w	c的子矩阵
A_w	由S_{work}列构成的A的子矩阵
A_C	由S_C列构成的A的子矩阵

定义问题O_{work}，固定α_0, α_C:

$$\begin{aligned} \underset{\alpha_{\text{work}}}{\text{minimize}} \quad & \tfrac{1}{2}\alpha_{\text{work}}^{\top} H_{ww}\alpha_{\text{work}} + (c_w + H_{wC}\alpha_C)^{\top}\alpha_{\text{work}} \\ \text{subject to} \quad & 0 \le \alpha_i \le C \text{ and } A_w\alpha_{\text{work}} = b - A_C\alpha_C \end{aligned}$$

主要思想依旧是首先解决一个子集的最优解，之后每次添加一个点，逐步得到最优解。算法三展示了对于该soft-svm优化问题的求解过程。[11]

算法 3 Soft Margin SimpleSVM

输入： 数据集P

输出： $S_0, S_C, S_{work}, \alpha_{work}$

1: 初始化α_{work}使得α_{work}是O_{work}的解
2: **while** 存在S_0或者S_C的点违反KKT条件 **do**
3: 从S_0或者S_C中选择一个违背KKT条件的点加入S_{work}
4: 忽略对于α_i的约束条件，即$0 \le \alpha_i \le C$,解决新的优化问题
5: **if** 如果得到的新解α_{work}^{new}满足α的约束条件 **then**
6: Continue
7: **else**
8: 选择不满足约束条件的点并加入S_C或者S_0，转至步骤4
9: **end if**
10: **end while**

2. 实验复现

我们分别取了数据集的1%, 2%, 5%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%作为训练集，并将与之对应的剩下的数据作为测试集。在实验时，我们选用高斯核函数

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\|\mathbf{x}-\mathbf{y}\|^2/\beta\right)$$

其中有

$$\beta = \frac{1}{m^2}\sum_{i,j=1}^{m}\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2$$

取 $\beta = 10000, C = 10000, \varepsilon = 10^{-6}$。

由于CVM对内存需求较大，我们使用服务器进行训练，服务器配置为 Linux 16.04, 48 Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz, 64GB。

2.1 Forest结果

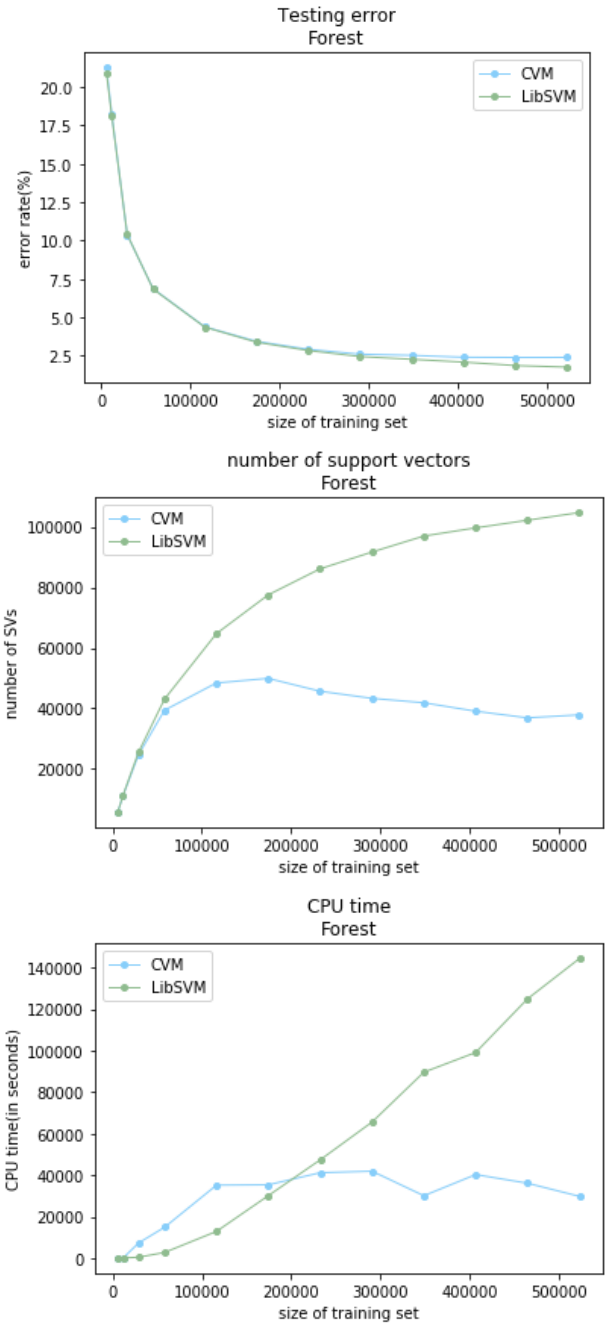

Figure 5. CVM与LibSVM在Forest数据集上的表现

由Figure5， 可知 CVM 准确率和 LibSVM 差距很小，且当达到一个极限值后，CVM的训练时间并不随训练集大小的增大而增大，而 LibSVM 的运行时间仍然保持增长的趋势。当训练集比较小的时候，增加训练集会增加额外的信息，从而有助于分类，故在训练集比较小的时候，CVM的支持向量的数目与训练集大小成正比关系。

2.2 Adult结果

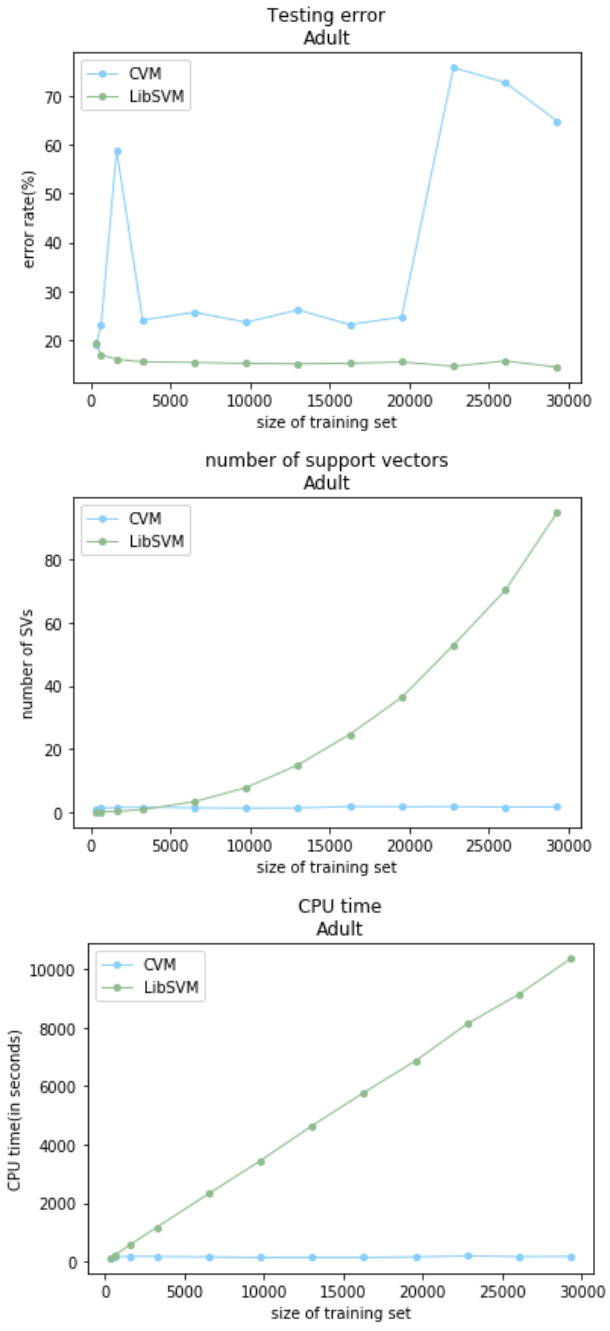

Figure 6. CVM与LibSVM在Adult数据集上的表现

CVM 算法在 Adult 数据集上的表现在数据集规模增大到一定规模时，准确率骤降。我们怀疑是C取的不够好，但在我们尝试了一些 C 值后，在数据集规模增大时，皆出现准确率骤降的情况。这一点很令人费解，我们试图对此进行解释： Adult 数据集的特征值与 Forest 数据集不同，其特征均是离散的0/1值。我们猜测，Adult 数据经过核函数映射后的不太适合 CVM 算法基于核空间中球状的结构，例如在找最远点时可能出现多个极值，这样或许不利于球的扩张，特别是在训练数据较大时。因此， CVM 在 Adult 上的准确率在训练集较大时极低。相反，这样的数据分布恰好适合用 LibSVM 找核空间中最佳分割平面的思路来划分，由此 LibSVM 在 Adult 数据集上呈现稳定的正确率。

2.3 不同C对于结果的影响

我们测试了三组不同的C，$C=1, C=1000, C=10^6$。不同的C对于训练时间，支持向量数目，准确率的影响见 Figure7。

由于时间与机器资源限制，我们只训练了forest数据集的 1%, 2%, 3%。 从实验结果可知， C 对准确率的影响很大，且 CVM 的算法并不是很稳定，比如在C取 10^6 时，准确率突然下降，而 LibSVM 则没有出现这种情况，这和 CVM 在 Adult 数据集的表现非常相像。

另外我们发现在 $C=1000$ 时 CVM 取得了最好的准确率，而在 C 过大时，CVM的准确率骤降，与此同时，CVM的支持向量的数目呈下降的趋势。

3. 改进与创新

说明: 创新结果为采用 Forest Covtype 数据集做出的实验结果。

3.1 初始化方式

让我们回顾一下原论文[1]中的初始化方式。简而言之，在最开始时随机选择点z_0, 之后在特征空间中找到 z_a 使得其与z_0的距离最远，然后再针对 z_a 找到另一个相对最远点 z_b，由此将 $\{z_a, z_b\}$ 作为初始化后的核心集 S_0。

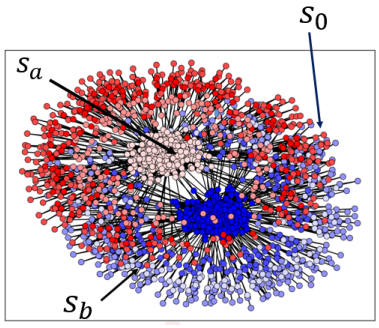

Figure 8. 原论文中初始化方式阐释图

加入随机采样之后这个问题的时间复杂度为O(D), D为每次随机采样的次数。我们将初始化方式修改为选择

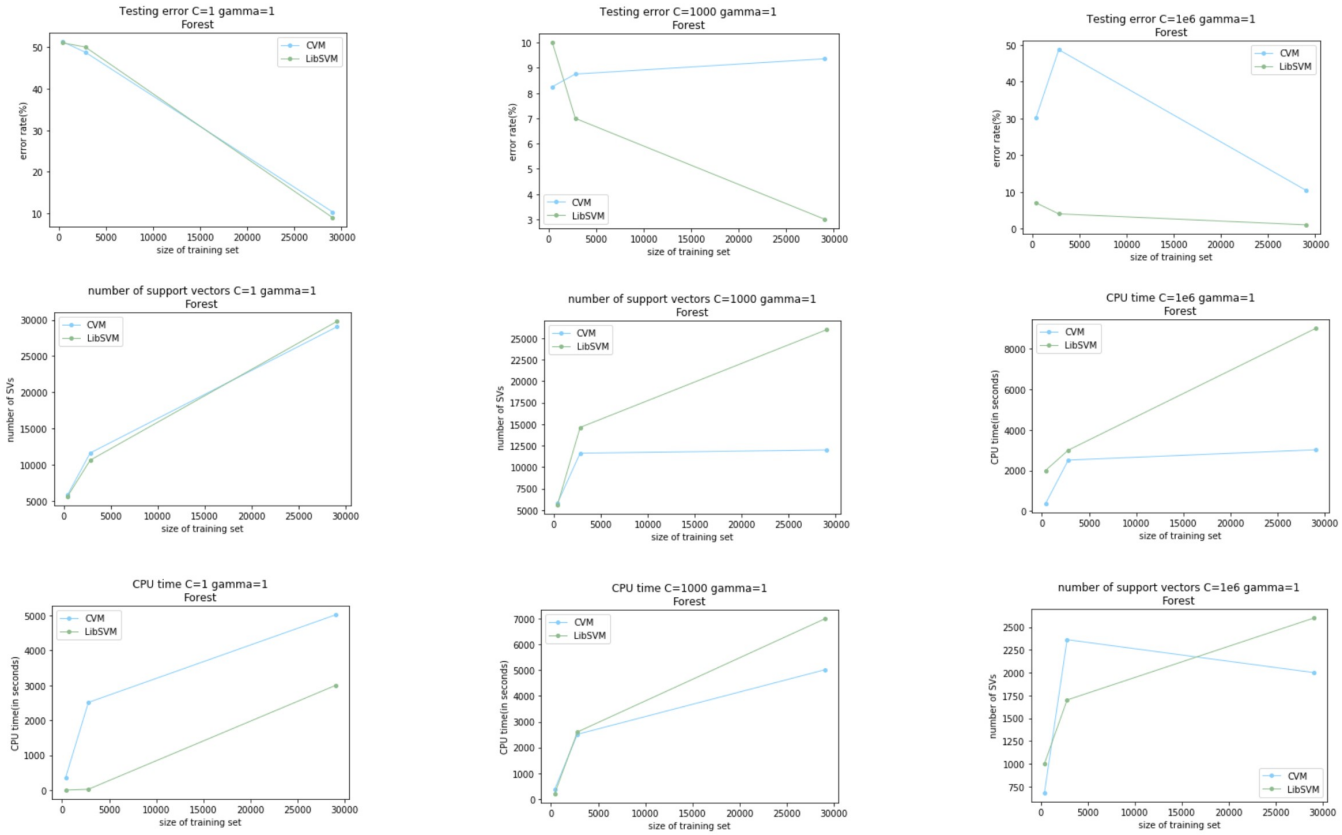

Figure 7. 不同的C对于准确率，训练时间，支持向量数目的影响

一个小子集，设子集的大小为 N, 则初始化需要的时间复杂度为 O(N) ($N < D$)。我们先简要分析一下这两种初始化方式的差别: 由于我们一开始将 Core-set 选定为一个大小为 N 的集合，对应的范围更广，初始化球半径更大，因此 MEB 迭代的总次数也会减小。而在迭代次数相同时，原本的初始化方法时间复杂度要更低。原因是单次迭代更新 MEB 的时间复杂度是 $O(t_i^3)$, 其中 t_i 为在第 i 次迭代 Core-set 的样本数。由于有 $t_i^{(1)} < t_i^{(2)}$，(1), (2) 对应改进前后的两种初始化方法，因此第 i 次迭代，改进后的方法时间复杂度要高一些。

这样的 trade-off 在各类算法中并不少见，具体的效果还得看具体的数据集以及实验结果。

而事实证明，这样的初始化方式在 Forest Covtype 数据集上能够得到比原先效率更高的结果。我们在训练集占总训练集比例为[0.01, 0.02, 0.05, 0.1]上做了实验，结果显示，改进后的初始化方法在时间效率上比原本的方法提高了 2.1% ∼ 5.5%，而准确率没有大的变化，改进后的初始化方法的准确度仅在原来的准确度的 2% 上下有微小波动。考虑到 CVM 这个算法本身有随机选点的因素，这样的波动可以视为正常的。

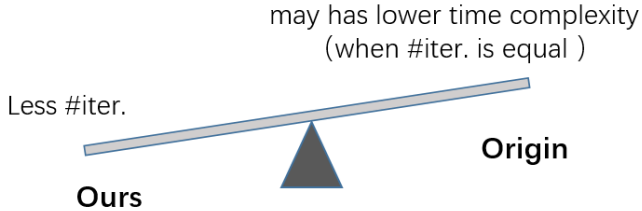

Figure 9. 改进后的初始化方式与原初始化方式的对比

3.2 随机采样的参数调整

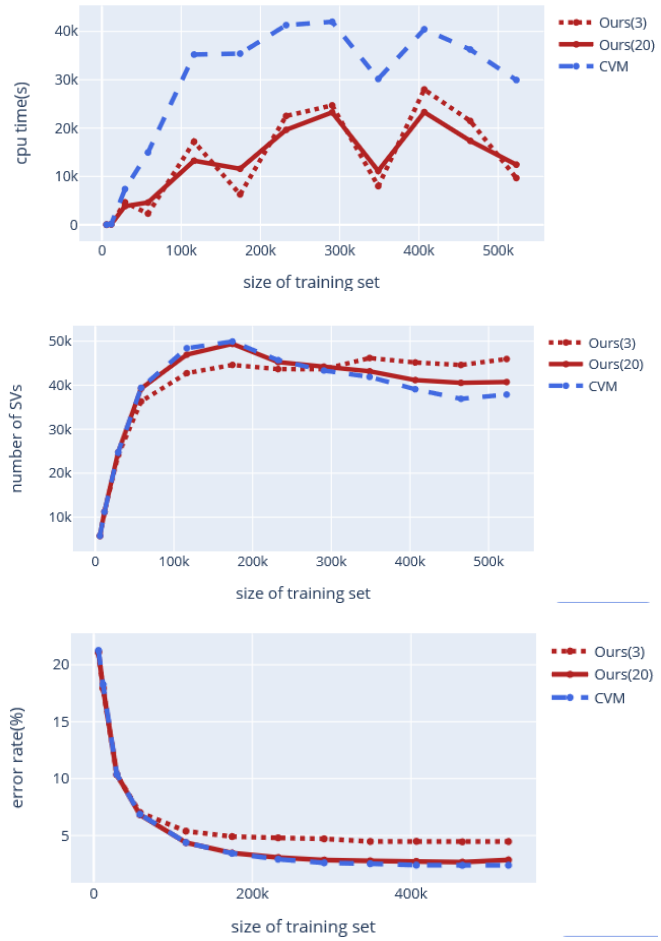

Figure 10. m值为 3，20，59 时对应的算法结果，时间效率上 Ours(3), Ours(20) 均为采样数为 CVM(59) 的两倍; SV 值中三条曲线较为接近; Ours(3) 的错误率与 CVM(59) 仅仅相差 2.08%， 而 Ours(20) 的错误率几乎与 CVM(59) 相同。

严格地添加最远点需要遍历，原论文中的改进是采用随机采样, 每次从没有被 $(1+\varepsilon)$ Ball 包住的点集中取 ℓ 个点中与圆心距离最大的点添加到 Core-set 中。原论文中取 $\ell = 59$，在解释该参数设置时，首先看以下引理(摘自参考文献[8])

Lemma 1 (Maximum of Random Variables) Denote by $\xi_1, \ldots, \xi_m$ identically distributed independent random variables with the common cumulative distribution function $F(\xi_i)$. Then the cumulative distribution function of $\xi := \max_{i\in[m]} \xi_i$ is $(F(\xi))^m$

由以上引理，我们来推导取 $m = 59$ 的依据。考虑若特征空间中所有的数据点服从均匀分布 U[0,1]，则有 $F(\varepsilon) = \varepsilon^m$。MEB 是一个不断迭代的随机采样过程，为了使得每次取到的最大值 ε 有 a 的概率达到 $b\%$ 以上的最高值比例，显然有 $a^m < 1-b$。 代入 a, b 值均为 0.95，由上述等式可以解出 $m > 58.4$ 正符合 m 的取值59。也就是说在概率近似下，为了保证一定的抽样质量，需要在每次迭代抽取59个点，则选出来的点中与圆心距离最大的点 A 在所有点与圆心的距离按降序的前 5% 的概率不小于 95%。

这个推导具有较高的理论保证，当然，前提是数据是符合均匀分布的。但从实验的角度来看，若采样量 m 不取 59，又会有什么样的效果呢？我们先进行分析：降低 m 的值很有可能给原本运行效率较高的 CVM 再次加速。当然，这将不再符合理论上严格的条件，可能会造成测试集上的准确度的降低。在效率与准确度的 ”trade-off” 中，如果能在准确度降低较小的情况下，得到效率上较高的提升，这样的改进将是有意义的。

我们分别选取 m = 3, m = 20，作出结果曲线如Figure 10所示。

综合时间效率与错误率来看，m = 20 的表现是最佳的。我们可以大致作一下理论分析，也就是说，当 $\varepsilon_i (i = 1, \ldots, 20)$ 服从 [0,1]上的均匀分布时，从引理一可以计算出，$P(\varepsilon > 0.95) = 0.642$; 而 m = 3时， $P(\varepsilon > 0.95)$ 仅仅为 0.143。很显然，在 m = 3 时，算法很难大概率地在每一次迭代中找到(ε_i降序排列)前 5% 的点，也很容易解释为什么 m = 3 时与其他两组的正确率有一定差异的原因。因此对于使用随机采样进行加速的算法，我们建议采用符合 $P(\varepsilon > 0.95) > 0.5$ 对应的 m 值。这样或许能够一方面确保最后得到的正确率，另一方面也可以提高运行效率。

3.3 AdaBoost-CVM

Adaptive Boosting（自适应增强）由Yoav Freund et al.在1995年提出。其自适应体现在：前一个基本分类器分错的样本会得到加强，加权后的全体样本再次被用来训练下一个基本分类器。同时，在每一轮迭代中加入一个新的弱分类器，直到达到某个足够小的错误率或迭代次数限制。AdaBoost 后来衍生出一个庞大的算法家族，统称Boosting，是机器学习中“集成学习”的主流代表性技术。其具体算法总结为三步，如下所示

算法 4 AdaBoost Algorithm

输入: 训统数据集 $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$, 其中 $x_i\in\mathscr{X}\subseteq R^n, y_i\in\mathscr{Y}=\{+1,-1\}, i=1,2,\cdots,N$;

输出: 分类器 $G(x)$

1). 初始化训练数据的权值分布

$$D_1=(w_{11},w_{12},\cdots,w_{1N}),\quad w_{1i}=\frac{1}{N},\quad i=1,2,\cdots,N$$

2). 对 $m=1,2,\cdots,M$

- 使用具有权值分布 D_m 的训练数据集学习, 得到基本分类器

$$G_m(x):\mathscr{X}\to\{-1,+1\}$$

- 计算 $G_m(x)$ 在训续数据集上的分类误差率

$$\begin{aligned} e_m&=\sum_{i=1}^{N}P(G_m(x_i)\neq y_i)\\ &=\sum_{i=1}^{N}w_{mi}I(G_m(x_i)\neq y_i)\end{aligned}\tag{12}$$

- 计算 $G_m(x)$ 的系数

$$\alpha_m=\frac{1}{2}\log\frac{1-e_m}{e_m}\tag{13}$$

- 更新训练数据集的权值分布

$$\begin{aligned} D_{m+1}&=(w_{m+1,1},\cdots,w_{m+1,i},\cdots,w_{m+1,N})\\ w_{m+1,i}&=\frac{w_{mi}}{Z_m}\exp(-\alpha_m y_i G_m(x_i))\\ &=\begin{cases}\frac{w_{mi}}{Z_m}\exp(-\alpha_m), G_m(x_i)=y_i\\ \frac{w_{mi}}{Z_m}\exp(\alpha_m), G_m(x_i)\neq y_i\end{cases}\quad i=1,2,\cdots,N\end{aligned}\tag{14}$$

其中， Z_m 是规范化因子

$$Z_m=\sum_{i=1}^{N}w_{mi}\exp(-\alpha_m y_i, G_m(x_i))\tag{15}$$

3). 构建基本分类器的线性组合

$$f(x)=\sum_{m=1}^{M}\alpha_m G_m(x)\tag{16}$$

得到最终分类器

$$G(x)=\text{sign}(f(x))=\text{sign}\left(\sum_{m=1}^{M}\alpha_m G_m(x)\right)\tag{17}$$

具体训练过程中，如果某个样本点已经被准确地分类，那么在构造下一个训练集中，它的权值就被降低；相反，如果某个样本点没有被准确地分类，那么它的权值就得到提高。然后，权值更新过的样本集被用于训练下一个分类器，整个训练过程如此迭代地进行下去。

各个弱分类器的训练过程结束后，加大分类误差率小的弱分类器的权重，使其在最终的分类函数中起着较大的决定作用，而降低分类误差率大的弱分类器的权重，使其在最终的分类函数中起着较小的决定作用。换言之，误差率低的弱分类器在最终分类器中占的权重较大，否则较小。

以下我们结合 CVM 具体的原理与代码构建，从而将 AdaBoost 的核心思想应用在提高 CVM 的分类准确度。

具体而言， Core-set 最终确定(停止循环)的条件是 $B(\mathbf{c}_t,(1+\varepsilon)R_t)$ 将空间中所有的点包含。在该算法中，只要最终的 Core-set 确定， CVM 的解也就随之确定。而 Core-set 很大程度上受到的是输入数据的影响。并且，包括 Add Furthest Point 在内的各个步骤中，涉及到的更多是几何信息，而不含对点的权值表示，因此无法在算法内部显式地更新训练数据的权值分布。为了解决这一难题，我们将点的权值转化为输入下一个分类器的概率(第i个输入数据独立，且服从参数为 P_i 的 Bernoulli分布)，即将 AdaBoost 中的 Weights Update 机制转化为能应用于 CVM 算法的 Input Probability Update机制，从而将这一算法应用在了 CVM 算法中(Figure 11)。

改写的各个公式如下所示，其步骤与AdaBoost算法一致，此处不再赘述。

$$e_m=\sum_{i=1}^{N}\frac{P_{mi}}{\sum_{j=1}^{N}P_{mj}}I(CVM_m(x_i)\neq y_i)\tag{18}$$

$$\alpha_m=\frac{1}{2}\log\frac{1-e_m}{e_m}\tag{19}$$

$$\begin{aligned} D_{m+1}&=(P_{m+1,1},\cdots,P_{m+1,i},\cdots,P_{m+1,N})\\ P_{m+1,i}&=\begin{cases}P_{mi}\exp(-\alpha_m), CVM_m(x_i)=y_i\\ 1, CVM_m(x_i)\neq y_i\end{cases}\quad i=1,\cdots,N\end{aligned}\tag{20}$$

$$G(x)=\text{sign}(f(x))=\text{sign}\left(\sum_{m=1}^{M}\alpha_m CVM_m(x)\right)\tag{21}$$

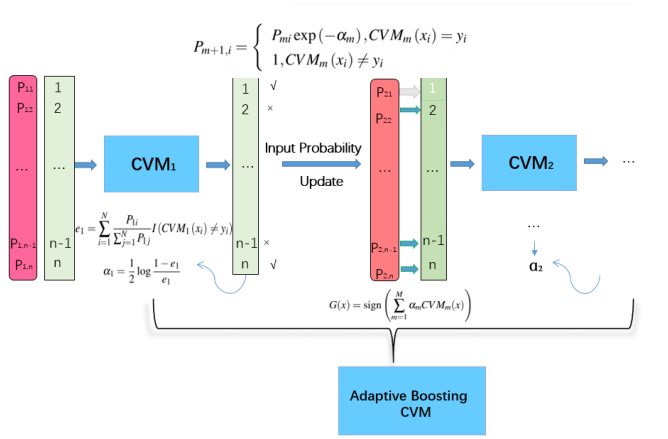

Figure 11. Adaptive Boosting 应用在 CVM 算法上的原理图，初始每个输入数据被分类器选中作为输入的概率$P_i = 1, i = 1, ..., n$。在CVM_1训练完成后，对训练集进行预测。与 AdaBoost 算法相似地，对于分类器计算分类误差率 e_1 与 α_1(见下式)，并以此更新输入数据概率分布，从而进行下一次迭代。最终的分类器将是各个分类器的线性组合。

经过对原算法的外框架改造，并且遵循上述公式，我们在训练集大小为 11.62k 时，使用 10 个弱分类器进行训练，在每一个弱分类器的错误率都较高(甚至有的弱分类器错误率接近 50%)的情况下，我们组合起来的弱分类器能够达到仅为 18.08% 的错误率(Figure 12)。这样一个较好的结果验证了我们的 AdaCVM 的框架是可行的。当然，若对参数作进一步调整，可能能够获得更好的结果。

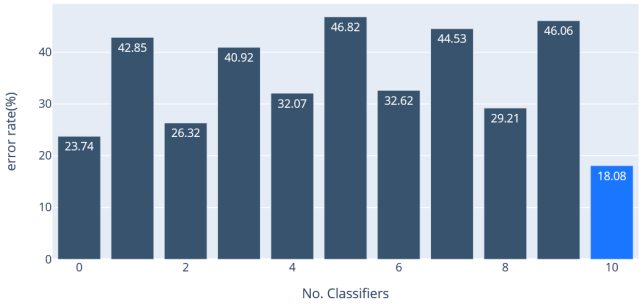

Figure 12. 10个弱分类器(左边)以及组合起来的强分类器(最右边)的错误率，其中10个弱分类器中最低错误率为约 23%, 最高的错误率将近 50%。

同时，这样的架构也存在着一些问题，一方面 AdaCVM 需要比 CVM 花费更长的时间，另一方面我们也有失败的案例显示 AdaCVM 最后合起来的强分类器可能不是特别理想。

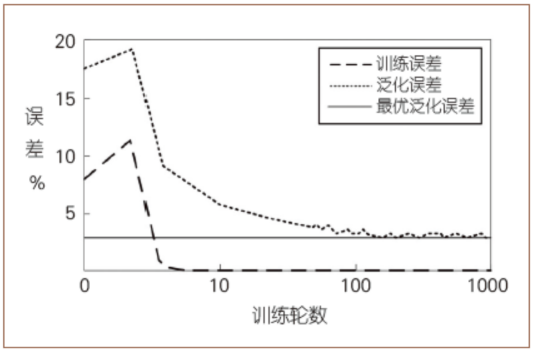

Figure 13. 周志华院士有关 AdaBoost 似乎不会发生过拟合的阐述图。其原文为“训练到第10轮和第1000轮时形成的假设（集成学习器）都与“实验观察”（训练数据）一致，前者仅包含10个基学习器、后者包含1000个基学习器。显然，根据奥卡姆剃刀应该选取前者，但实际上后者却更好”。

最后关于 AdaCVM 的性质作一些讨论。实验上，我们发现 AdaCVM 最终输出的准确率基本与弱分类器数目成正比。这是巧合吗？通过进一步调研，我们找到了周志华院士关于 AdaBoost 算法介绍的一篇文章，其中有一张图如 Figure 13 所示。 之后夏柏尔等人在1998年采用 “间隔” 这一概念成功解释了 AdaBoost 不会发生过拟合的原因，主要思想在于 AdaBoost 在多轮训练的过程中，即使训练误差为零，其分割间隔却能够进一步加大，因此泛化误差能进一步减小。也就是说，即使采用越多的弱分类器使得模型看上去复杂了，但事实上它的性能是保持提升的。当然，在后续也有关于 AdaBoost 泛化性能的解释，并且引入了”间隔方差”等概念。

AdaCVM 的整体架构是基于 AdaBoost， 因此可以断言，AdaCVM 也具有抑制分类中的过拟合的性质。

4. 其他改进方向

我们将关于 CVM 的一些其他的改进想法总结如下。

β_ε *stable* MEB

H. Ding(2019)提出了符合β_ε *stable* 的 计算 MEB 的次线性时间采样算法, 如算法 5 所示。其主要思想在于先对原始数据集进行随机采样，而后在这个比较小的新数据集上求解 MEB，得到其半径与圆心后，稍微拓展其半径，则可以证明，最后得到的球将包含至少 $(1-\beta_\varepsilon)n$ 个点，其中 n 为全体数据集的大小。

算法 5 MEB Algorithm

输入： A β_ε-stable instance P of MEB problem in $\mathbb{R}^d$

1.Sample a set S of $\Theta\left(\frac{d}{\beta_\varepsilon}\log\frac{d}{\beta_\varepsilon}\right)$ points from P uniformly at random.

2.Apply any approximate MEB algorithm (such as the core-set based algorithm [15]) to compute a $(1+\varepsilon)$approximate MEB of S, and let the resulting ball be $\mathbb{B}(c,r)$

3.Output the ball $\mathbb{B}\left(c, \frac{1+(2+\sqrt{2})\sqrt{\varepsilon}}{1-\varepsilon}r\right)$

Ball Vector Machine Ivor W. Tsang 等人在提出 CVM 后两年里提出了 BVM，其相对于 CVM 算法，时间复杂度要更低。其具体思想在于固定所要求解的球的半径，进而寻求关于圆心的最小调整方式。具体细节详见参考文献[7]。

5. 实验总结

SVM 是机器学习的经典问题以及处理大数据的有利工具，其中 ”间隔” 的概念更是影响深远。CVM 更是在巨人的肩膀上给予了我们更丰富的空间几何的印象。事实证明，当对一个算法的原理十分清晰时，是可以在大脑里一步一步思考其推导过程，琢磨其内涵。而在琢磨之后，加之以创新的灵感，才能得以作出自己的成果。当然，对于一些复杂的理论，如 AdaCVM 的收敛性证明，限于此时的数学功底，很遗憾我们无法给出。我们将会在日后更扎实地学习。我们小组在此次的实践中收获实多，感谢大数据算法课程，感谢丁虎老师与助教的指导!

参考文献

[1] Ivor W. Tsang, James T. Kwok, Pak-Ming Cheung: Core Vector Machines: Fast SVM Training on Very Large Data Sets. J. Mach. Learn. Res. 6: 363-392 ,2005.

[2] Gaëlle Loosli, Stéphane Canu: Comments on the ”Core Vector Machines: Fast SVM Training on Very Large Data Sets”. J. Mach. Learn. Res. 8: 291-301 ,2007.

[3] Badoiu, M., & Clarkson, K. Optimal core-sets for balls. DIMACS Workshop on Computational Geometry， 2000.

[5] H. Ding. Minimum Enclosing Ball Revisited: Stability and Sub-linear Time Algorithms. In arXiv 1904.03796.

[5] Xuchun Li, Lei Wang, Eric Sung. AdaBoost with SVM-based Component Classifiers. In IEEE TRANSACTIONS ON SYSTEM，2008.

[6] Elkin García, Fernando Lozano. Boosting Support Vector Machines. In MLDM，2007.

[7] Ivor W. Tsang, Andras Kocsor, James T. Kwok.Simpler Core Vector Machines with Enclosing Balls, 2007.

[8] A. Smola and B. *Schŏlkopf*. Sparse greedy matrix approximation for machine learning. In Proceedings of the Seventeenth International Conference onMachine Learning, pages 911–918, Stanford, CA, USA, June 2000.

[9] Badoiu, S. Har-Peled, and P. Indyk. Approximate clustering via core sets. In Proceedings of 34th Annual ACM Symposium on Theory of Computing, pages 250–257,2002.

[10] Mihai Bâdoiu and Kenneth L. Clarkson. 2003. Smaller core-sets for balls. In Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms (SODA ’ 03). Society for Industrial and Applied Mathematics, USA, 801–802.

[11] Vishwanathan, S. & Smola, Alex & Murty, M.. (2003). SimpleSVM. 760-767.

[12] Piyush Kumar, Joseph S. B. Mitchell, and E. Alper Yildirim. 2004. Approximate minimum enclosing balls in high dimensions using core-sets. J. Exp. Algorithmics 8 (2003), 1.1–es. DOI:https://doi.org/10.1145/996546.996548