基于聚类的大样本支持向量机研究*)

奉国和[1] 朱思铭[2]

(华南师范大学经济管理学院信息管理系 广州 510631)[1] (中山大学数学与计算科学学院 广州 510275)[2]

摘 要 针对大样本支持向量机内存开销大、训练速度慢的缺点,本文提出了基于聚类支持向量机,运用 *k-mean* 对样本聚类,压缩样本量,构造初始超平面,筛选出靠近超平面的支持类和可能支持向量,重新构造决策超平面。实验表明,在保持泛化精度基本一致前提下,改进算法的训练速度明显提高。

关键词 支持向量机分类,大样本,聚类

Clustering-based Large Scale Samples Support Vector Machines

FENG Guo-He[1] ZHU Si-Ming[2]

(College of Economics and Management, South Chian Normal University, Guangzhou 510631)[1]

(The School of Mathematics and Computational Science, Zhongshan University, Guangzhou 510275)[2]

Abstract Training a support vector machines on a data set of huge size exists one problem with slow training process. In this paper, we use a modified support vector machines- clustering-based support vector machines to resolve this problem. It speeds up the training process fastly comparing with conventional support vector machines under the almost same classification precision.

Keywords Support vector machine classification, Large-scale samples, *k-mean* Clustering

1 引言

对于样本数为 n 的分类支持向量机求解归结为 1 个等式约束、$2n$ 个线性不等式约束的凸二次规划优化问题[1,2],求解过程涉及一个 $n\times n$ 维 Hession 矩阵的计算和矩阵与向量相乘计算,求解的速度与样本数量有关。对于大样本,支持向量机训练会出现速度慢、占用内存大等缺点。近年就怎样提高大样本训练速度提出了一些算法,这些方法共同的思想就是分而治之(divide-and- conquer),如 Boser 和 Vapnik 的分块算法(chunking algorithm)[5]、Osuna 等的分解算法(decomposition algorithm)[6]、Platt 的贯序最小化算法(Sequential Minimal Optimization, SMO)[7]。这些算法为了寻找最优化解,需要反复地迭代,往往会出现收敛速度慢的情况。

针对大样本支持向量机回归,奉国和等提出了一种分解合作加权支持向量机算法[4]。本文针对大样本分类提出基于聚类的支持向量机(Clustering-Based Support Vector Machines, CBSVM),该算法不需要反复迭代,而是先对样本进行预抽取,选出可能的支持向量构造分类超平面。即将样本分为正负两类,分别对正负类 *k-mean* 聚类压缩样本,利用聚类中心构造初始超平面,筛选出支持类包含的样本,重新构造分类超平面。这样,选择靠在分类超平面附近的样本,删除无关样本,从而大大减少了训练样本的数量。实验结果表明,在保持泛化精度相同的情况下,训练速度明显提高。

2 分类支持向量机

支持向量机突出的优点是给出了实际风险的上界,并利用核函数将线性不可分转化为特征空间线性可分,求解化为一个线性约束的凸二次规划(QP)求解问题,解是全局最优和唯一的。

给定训练集 $G=\{(x_i,y_i)\}_{i=1}^n$,其中 $x_i\in R^d$,$y_i\in\{1,-1\}$,目的是找到一个最优超平面

$$(\phi(x)\cdot\omega)+b=0 \tag{1}$$

将训练集分开,其中$(\cdot)$表示在高维特征空间 Ω 中的内积,$\phi: x\to\Omega$,$b\in R$ 表示将输入空间的数据影射到高维的特征空间。考虑误差,引入松弛变量 $\xi_i\geqslant 0$,则问题转化为在约束条件

$$y_i[(\omega\cdot\phi(x_i))+b]\geqslant 1-\xi_i,\ i=1,2,\cdots,n \tag{2}$$

下最小化如下泛函:

$$\phi(\omega,\xi)=\frac{1}{2}\|\omega\|^2+C(\sum_{i=1}^n\xi_i) \tag{3}$$

引入拉格朗日函数,将该优化问题转化为其对偶优化问题[1,2]:

$$\max_{\alpha} Q(\alpha)=\max_{\alpha}\sum_{i=1}^n\alpha_i-\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n\alpha_i\alpha_n y_i y_j k(x_i,x_j) \tag{4}$$

约束条件为

$$\sum_{i=1}^n y_i\alpha_i=0\quad 0\leqslant\alpha_i\leqslant C,i=1,2,\cdots,n \tag{5}$$

其中 α_i 为与每一个样本对应的 Lagrange 乘子。由于所求的 α_i 中只有一少部分不为 0,不为 0 的 α_i 对应的样本称为支持向量(Support Vectors)。最后求出的最优分类函数为:

$$f(x)=(\omega\cdot\phi(x))+b=\sum_{i=1}^n\alpha_i y_i k(x_i,x)+b$$

$$y=\mathrm{sgn}(f(x)) \tag{6}$$

3 CBSVM 模型及算法设计

3.1 聚类特征(Clustering Feature, CF)

*)国家自然科学基金资助项目(基金编号:10371135)。奉国和 博士生,主要研究领域为人工智能、数据挖掘;朱思铭 教授,博导,主要研究领域为动力系统、人工智能。

给出训练样本子类 $S=\{x_i\}_{i=1}^{N}$，$x_i\in R^d$，则该子类的聚类特征定义为[3]：

$$CF=(N,\vec{S}_L,S_S), \tag{7}$$

其中

$$\vec{S}_L=\sum_{i=1}^{N}x_i,S_S=\sum_{i=1}^{N}x_i^2 \tag{8}$$

聚类特征是对子类的统计汇总，体现了对类信息的压缩。它没有包含所有的样本，而是类的 0 阶矩、1 阶矩、2 阶矩信息。同时，我们定义子类的中心和半径：

$$C=\frac{\sum_{i=1}^{N}x_i}{N}, \tag{9}$$

$$R=\sqrt{\frac{\sum_{i=1}^{N}\|x_i-C\|^2}{N}} \tag{10}$$

定理 1(聚类特征加和定理)[9] 给出子类 $S_1=\{x_i\}_{i=1}^{N_1}$，$x_i\in R^d$，$S_2=\{y_i\}_{i=1}^{N_2}$，$y_i\in R^d$，且 $S_1\cap S_2=\varnothing$，它们的聚类特征分别为 $CF_1=(N_1,\vec{S}_{L1},S_{S1})$，$CF_2=(N_2,\vec{S}_{L2},S_{S2})$，则 $S=S_1\cup S_2$ 的聚类特征为：

$$CF_1+CF_2=(N_1+N_2,\vec{S}_{L1}+\vec{S}_{L2},S_{S1}+S_{S2}) \tag{11}$$

定理 1 说明两个子类的汇总信息是可以加和的。这为我们合并类提供了一个好的信息处理手段。

3.2 *k-mean* 聚类

给定对象集合，将该对象集合划分为 k 类，使得同一类中的对象高度“相似”，而不同类中的对象是“相异”的。*k-mean* 聚类算法如下，

k-mean 聚类划分算法：

输入：一训练样本集 S 和族的数目 k
输出：一 k 个族，使得平方误差最小；
算法：

1. 任意选择 k 个对象作为初始的族中心；
2. Repeat
3. 根据族中对象的平均值，将每个对象赋给类似的族；
4. 利用聚类特征更新族的平均值，即计算每族中对象的平均值；
5. Until 不再发生变化。

图 1 *k-mean* 聚类划分算法

在 *k-mean* 聚类算法中，平方误差准则[3]定义如下，

$$E(S,k)=\sum_{i=1}^{k}\sum_{p\in Cluster_i}|p-C_i|^2=\sum_{i=1}^{k}\sum_{j=1}^{n_i}|p-C_i|^2 \tag{12}$$

其中 $E(S,k)$ 是所有对象的平方误差总和；p 为空间中一个对象，表示给定的数据对象；C_i 是族 $Cluster_i$ 的中心，即族 $Cluster_i$ 中对象的平均值；n_i 为族 $Cluster_i$ 对象的个数。设样本数为 n，子类数为 k，迭代次数为 l，则 *k-mean* 聚类的计算复杂度为 $O(nkl)$，其中 $k\ll n$，$l\ll n$，所以其复杂度同样本数 n 呈近似线性关系。

3.3 支持类

支持向量机求解与样本数密切相关，求出的支持向量是靠近分类超平面的样本。如果事先能够筛选出这些样本，则训练样本数大大减少了。基于这样一种思想，我们先将样本分正负类，分别对正负类进行 *k-mean* 聚类，利用聚类中心压缩的信息构造初始分类平面，并选择靠近平面的支持类，扩展其包含的样本，重新构造超平面。

支持向量机在最优解处满足 KKT 条件，设 $f(x)=(\omega\cdot\phi(x))+b$，推出：

$$\alpha_i=0\Rightarrow y_if(x_i)\geqslant 1 \tag{13}$$

$$0<\alpha_i<C\Rightarrow y_if(x_i)=1 \tag{14}$$

$$\alpha_i=C\Rightarrow y_if(x_i)\leqslant 1 \tag{15}$$

根据式(13)、(14)、(15)可将训练集 G 分为三类：第一类为满足 $f(x_i)=1$ 或者 $f(x_i)=-1$ 的数据，即称为边界支持向量(Boundary Support Vector，BSV)；第二类为满足 $-1\leqslant f(x_i)\leqslant 1$ 的数据点，即称为错误支持向量(Error Support Vector，ESV)；第三类为满足 $f(x_i)\geqslant 1$ 或者 $f(x_i)\leqslant -1$ 的数据点，称为可去支持向量(Romoved Support Vectors，RSV)。

定义 1(中心类，Center Clusters) 设子类中心构造的超平面为 h，关于 h 子类中心是边界支持向量的子类为中心类。

如图 2 所示，子类 15 和子类 04 则为中心类。

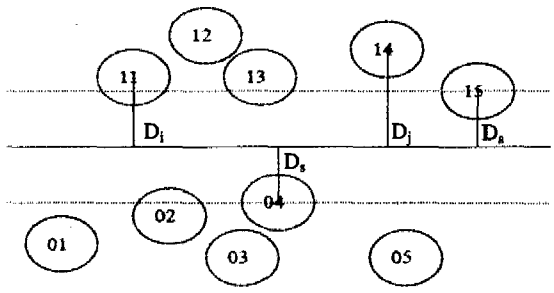

图 2 支持类示意图

定义 2(支持类，Support Clusters) 设中心类中心到超平面的距离为 D_s，D_i 为子类 $Cluster_i$ 中心到超平面距离，R_i 为 $Cluster_i$ 的半径，如果

$$D_i-R_i<D_s \tag{16}$$

则称子类 $Cluster_i$ 为支持类。

如图 2 所示，子类 11、13、15、02、04 都是支持子类。

我们的目的是利用初始超平面筛选出所有的支持类，展开支持类包含的样本构造最终分类超平面。

3.4 算法设计

Input：-train samples S；

Output：-a boundary function h；

Natation：—*GetPNSample*(S)将样本数据分为正、负样本；
—*KMC*(S)利用 *k-mean* 聚类得到类中心；
—*SVM. train*(S)利用样本 S 得到超平面；
—*GetSupportCluster*(h,S)获得支持类；
—*GetChildSanple*(S)获得支持类包含的样本；

Algorithm：

1. $(S_P,S_N)=GetPNSample(S)$；
2. $S_{PC}=KMC(S_P)$；$S_{NC}=KMC(S_N)$；
3. $h_{init}=SVM.\ train(S_{PC}\cup S_{NC})$；
4. $S_{SC}=GetSupportCluster(h_{init},S_{PC}\cup S_{NC})$；
5. $S_{TS}=GetChildSanple(S_{SC})$；
6. $h=SVM.\ train(S_{TS})$。

图 3 CBSVM 算法

3.5 计算复杂性分析

设训练对象数为 n，族数为 k，*k-mean* 聚类迭代次数为 l，支持类数为 m，所有支持类中包含的最大对象数为 n_{max}，则不做预处理的支持向量机的训练时间复杂度为 $t(SVM)=O(n^2)$，内存开销为 $m(SVM)=O(n^2)$；对于聚类支持向量机而言，*k-mean* 聚类的复杂度为 $O(nkl)$，寻找初始超平面的计算时间复杂度为 $t(SVM_{init})=O(4k^2)$，最终超平面的构造训练时间复杂读为 $t(SVM_{Final})=O((m\times n_{max})^2)$，所以总的时间复杂度为，

$$t(CBSVM)=O(nkl)+O(4k^2)+O(m^2\times n_{max}^2),$$

一般情况下，$k\ll n$，$m\ll 2k$，$n_{max}\ll n$，所以有

$$t(CBSVM)<t(SVM)。$$

4 实验结果

4.1 实验条件

表 1 实验条件

操作系统	Windows 2000 Pro
CPU	Celeron1.7G
内存	DDR256M
软件环境	Matlab6.5
计算软件	Steve Gunn SVM[8]

4.2 实验数据

数据 ringnorm 数据集是从 http://svm.first.gmd.de Delve Datasets 上下载的，是 Leo Breiman 用于两类划分的包含 7400 个样本点的数据集，数据均取自一个 20 维的多变量正态分布。类 1 的均值为 0，方差为 4 倍单位元；类 2 的均值为$(a,a,\cdots a)$，其中$a=2/\sqrt{20}$，方差为单位元。

4.3 结果分析

支持向量机涉及到一个 $n\times n$ 的 Hession 矩阵的计算，由于受到微机内存的限制，大样本数据支持向量机会造成内存的溢出。对于数据集 ringnorm 我们只取前 2000 个样本作为训练集，之后的 1000 个作为测试集。表 2 中 CBSVM 为本文提出的基于聚类的大样本支持向量机算法，而 SVM 则为标准支持向量机算法，计算采用的核函数为 RBF 径向基函数，其中参数 $C=100,\sigma=0.5,k=100$，计算结果如下：

表 2 实验结果

算法	参与训练样本数	支持向量数	测试正确率	训练时间
CBSVM	839	276	92.6%	863.5(s)
SVM	2000	543	93.0%	3762.5(s)

基于聚类的支持向量机参与训练样本数为 839，支持向量数为 276，测试正确率为 92.6%，训练时间为 863.5s；而标准支持向量机是全部 2000 个样本参与训练，得到 543 个支持向量，测试正确率为 93.0%，训练所用时间为 3762.5s。可以看出，经过聚类筛选出支持族后，参与训练的样本明显比标准支持向量机少，而得到的支持向量也要少。在保持分类精度基本一致的情况下，训练速度则是明显加快。实验说明本文提出的算法是有效的。

结论 采用改进的支持向量机算法——基于聚类的支持向量机做大样本分类训练，首先利用 *k-mean* 聚类筛选出超平面附近的支持类，得到可能的支持向量训练分类器，从而使训练样本大为压缩。理论和实验表明，在保持泛化精度一致情况下，改进算法的训练速度提高了。

参 考 文 献

1 Vapnik V N. The Nature of Statistical Learning Theory [M]. New York: Springer, 1999
2 vapnik V N. Statistical learning theory [M]. New York: Wiley, 1998
3 Han Jiawei 等著. 数据挖掘-概念与技术[M]. 范明，孟小峰译. 北京：机械工业出版社，2001
4 奉国和，等. 基于支持向量机的分解合作加权算法及其应用[M]. 计算机科学，2005，32(4)：91～93
5 Boser B E, Guyon I M, Vapnik V N. A Training Algorithm for optimal margin classifiers [J]. In: Fifth Annual Workshop on Computational Learning Theroy, Pittsburgh, PA: ACM Press, 1992. 144～152
6 Osuna E, Freund R, Girosi F. An Improved Training Algorithm for Support Vector Machines [C]. New york: ICNNSP97, 1997. 276～285
7 Platt J. Fast training of support vector machines using sequential m inimal optimization [A]. Scholkopf B, Burges C, Smola A. Advances in Kernel Methods-Support Vector Learning [C]. Cambridge, MA: MIT Press, 1999. 185～208
8 Gunn S. Support vector machines for classification and regression [R]: [TR 1998-10]. University of southampton, 1998
9 Zhang T, Ramakrishnan R, Livny M. BIRCH: an efficient data clustering method for very large databases [C]. In: Proc. ACM SIGMOOD Int Conf. Management of Data, 1996. 103～114

(上接第 142 页)

$t_{in}(y_1,\cdots,y_p))\wedge for_i(y_1,\cdots,y_p))))$，

下面证明如果 $tv_i(lit(x_1,\cdots,x_n))=T$，则有 $tv_i(for')=T$。

如果 $tv_i(lit(x_1,\cdots,x_n))=T$，则对任意论域中的个体 $a_1,\cdots,a_n$，都有 $tv_i(lit(a_1,\cdots,a_n))=T$，即 $lit(a_1,\cdots,a_n)\in i$。因为对于 Pr 的范式 Pr'，有 $T_{Pr}'(i)=T_{Pr}(i)$，所以如果 $T_{Pr}(i)=i$，则 $T_{Pr}'(i)=i$，从而 $lit(a_1,\cdots,a_n)\in T_{Pr}'(i)$，这表明存在范式 Pr' 中的规则基例

$lit(a_1,\cdots,a_n)\leftarrow\vee_{i=1}(\exists y_1\cdots\exists y_p(a1=t_{i1}(y_1,\cdots,y_p)\wedge\cdots\wedge a_n=t_{in}(y_1,\cdots,y_p)\wedge for_i(y_1,\cdots,y_p)))$，使得 $tv_i(\vee_{i=1}(\exists y1\cdots\exists y_p(a_1=t_{i1}(y_1,\cdots,y_p)\wedge\cdots\wedge a_n=t_{in}(y_1,\cdots,y_p)\wedge for_i(y_1,\cdots,y_p))))=T$。这表明 $tv_i(lit(x_1,\cdots,x_n)\rightarrow for')=T$。因此，$i$ 也满足规则 $lit(x_1,\cdots,x_n)\rightarrow for'$。

由(1)(2)得，i 是 $Comp(\rightarrow,\ Pr)$ 的 Herbrand 模型。

定理 8 设 Pr 是只带有正规则的三值逻辑程序，如果 $Comp(\rightarrow,Comp(\rightarrow,Pr))$ 也是一个三值逻辑程序，而且 i 是 $Comp(\rightarrow,Comp(\rightarrow,Pr))$ 的 Herbrand 模型，则 $F_{Pr}(i)=i$。

证明：对于只带有正规则的三值逻辑程序 Pr，如果 $Comp(\rightarrow,Comp(\rightarrow,Pr))$ 也是一个三值逻辑程序，而且 i 是 $Comp(\rightarrow,Comp(\rightarrow,Pr))$ 的 Herbrand 模型，根据定理 6 得，$T_{Comp(\rightarrow,Pr)}(i)=i$。再由定理 $T_{Comp(\rightarrow,Pr)}=F_{Pr}$，有 $F_{Pr}(i)=i$。

结束语 本文是继续文[8]的工作。首先通过两个反例，指出了文[7]中关于否定完备化程序 $Comp(\rightarrow,\ Pr)$ 和蕴涵完备化程序 $Comp(\rightarrow,\ Pr)$ 的两个定理(命题 2.20 和命题 2.21)都存在一定程度的错误。然后对这两个定理进行了修改，用后继算子 T_{Pr} 和 *Fitting* 算子 F_{Pr} 的不动点语义，分别给出了否定完备化程序 $Comp(\rightarrow,\ Pr)$ 和蕴涵完备化程序 $Comp(\rightarrow,\ Pr)$ 的 Herbrand 模型的充分条件和必要条件，这将在逻辑程序的最优不动点和最小不动点的研究中有着重要的应用价值。

参 考 文 献

1 Kowalski R A. The Relation Between Logic Programming and Logic Specification. In: Hoare C, Shepherdson J, eds. Mathematical Logic and Programming Languages [C]. Prentice-Hall: Englewood C, 1985. 11～27
2 Lloyd J W. Foundations of Logic Programming, 2nd Edition [M]. Berlin: Springer-Verlag, 1987. 1～203
3 王鼎兴，温冬婵，高耀清，黄志毅. 逻辑程序设计语言及其实现技术. 北京：清华大学出版社，广西：广西科学技术出版社，1996
4 Mycroft A. Logic Programs and Many-Valued Logic. STACS 84, Lecture Notes in Computer Science 166, Springer-Verlag, Berlin, 1984. 274～286
5 Lassez J L, Maher M J. Optimal Fixpoints of Logic Programs. Theoretical Computer Science, 1985, 39: 15～25
6 Fitting M. Partial Models and Logic Programming. Theoretical Computer Science, 1986, 48: 229～255
7 Delahaye J P, Thibau V. Programming in Three-Valued logic. Theoretical Computer Science, 1991, 78: 189～216
8 Ying Ming-sheng, Liu Fu-chun. Three-valued and four-valued approach to logic programming with negation. Chinese Journal of Advanced Software Research, 1999, 6(1): 71～83