·人工智能及识别技术· 文章编号:1000-3428(2019)01-0192-07 文献标志码:A 中图分类号:TP182

变样本量学习最小二乘支持向量机算法

加尔肯别克,袁 杰

(新疆大学 电气工程学院,乌鲁木齐 830047)

摘 要: 为增加最小二乘支持向量机(LS-SVM)算法解的稀疏性,提高其运算效率,提出一种变样本量学习 LS-SVM 算法。从训练集中随机抽取部分样本作为初始工作集,在学习阶段将样本训练过程分为样本增量和样本减量 2 个阶段。在样本增量阶段,按 KKT 条件选取特定样本加入工作集并进行训练,在样本减量阶段,采用负松弛变量剪枝策略与基于对偶目标函数差的剪枝策略实现剪枝。在此基础上,采用工作集中的剩余样本构造学习分类器。实验结果表明,相对 SMO、SMO-new、ISLS-SVM 算法,该算法具有稀疏性高、运算速度快、无精度损失等优点。

关键词: 最小二乘支持向量机; 稀疏性; 变样本量学习; 预剪枝; KKT 条件

中文引用格式: 加尔肯别克,袁杰. 变样本量学习最小二乘支持向量机算法[J]. 计算机工程,2019,45(1):192-198,205.

英文引用格式: JIA Erkenbieke, YUAN Jie. Variable samples learning least square support vector machine algorithm[J]. Computer Engineering, 2019, 45(1):192-198, 205.

Variable Samples Learning Least Square Support Vector Machine Algorithm

JIA Erkenbieke, YUAN Jie

(School of Electrical Engineering, Xinjiang University, Urumqi 830047, China)

【Abstract】 In order to increase the sparseness of the solution of Least Squares Support Vector Machine (LS-SVM) algorithm and improve its operation efficiency, a variable samples learning LS-SVM algorithm is proposed. Some samples are randomly selected from the training set as the initial working set, and the training process is divided into two stages: sample increment and sample reduction. In the sample increment stage, select specific samples according to KKT conditions to join the working set and train. In the sample reduction stage, Negative Slack Variable Pruning Strategy (NSVPS) and Dual Objective Function Deviation Pruning Strategy (DOFDPS) pruning strategy are used to achieve pruning. On this basis, the residual classifier is used to construct learning classifier. Experimental results show that compared with SMO, SMO-new and ISLS-SVM algorithm, the algorithm has the advantages of high sparsity, fast operation speed and no loss of precision.

【Key words】 Least Squares Support Vector Machine (LS-SVM); sparseness; variable samples learning; pre-pruning; KKT condition

DOI: 10.19678/j.issn.1000-3428.0048673

0 概述

支持向量机(Support Vector Machine, SVM)是由文献[1]在统计学习理论的基础上建立起来的一种机器学习方法,它是一个重要的监督学习工具[2-3],得到了广泛的研究与应用[4-5]。SVM 算法解决分类、回归问题时需要求解含不等式约束的二次规划问题。最小二乘支持向量机(Least Squares Support Vector Machine, LS-SVM)算法是对 SVM 算法的扩展[6],它将不等式约束转换为等式约束,从而将解二次规划问题转化为解线性方程组问题,降低了算法的运算复杂度。但 LS-SVM 算法失去了解的稀疏性[7],导致其分类速度较慢、泛化能力较差。由于解的稀疏性有利于提升 LS-SVM 算法的泛化能力和特征学习效率,提高分类器的分类速度。因此,对 LS-SVM 算法解的稀疏性进行研究具有重要学术意义和潜在的应用价值。

为得到 LS-SVM 算法的稀疏解,有学者进行了相关研究并取得了重要的研究成果与进展。文献[8-9]提出构造稀疏 LS-SVM 的剪枝算法,通过解 LS-SVM 并根据样本贡献度来决定样本的取舍。文献[10]提出一种基于最小剪枝误差的剪枝算法。文献[11]在剪枝策略上提出一种基于 SMO 的剪枝策略。文献[12]在文献[13]对 L_0 范式进行迭代优化

基金项目: 国家自然科学基金(61863033);新疆维吾尔自治区自然科学基金(2016D01C032)。

作者简介: 加尔肯别克(1990—),男,硕士研究生,主研方向为机器学习、模式识别;袁 杰(通信作者),副教授、博士。

收稿日期: 2017-09-14 **修回日期:** 2017-11-30 **E-mail:** 565932985@qq.com

的基础上,将分类和回归 LS-SVM 都统一在一个系数和算法中。文献[14]将原始双 LS-SVM 与采用最小化 L_0 范数稀疏化的 LS-SVM 算法相结合,提出一种稀疏化 LS-SVM 算法。上述算法的共同特点是需要解非稀疏 LS-SVM,导致运算效率较低。文献[15]提出一种自下而上的剪枝策略,避开了解原始非稀疏 LS-SVM 算法的问题。文献[16]提出一种基于全局代表点选择的快速 LS-SVM 稀疏化算法,该算法可以一次性地选出最具有全局代表性的数据点,然后通过这些数据点构造支持向量集。这 2 种算法具有效率高、稀疏性好的特点,但是算法的精度需要进一步提升。

为获得 LS-SVM 的稀疏解,本文设计一种负松弛变量剪枝策略(Negative Slack Variable Pruning Strategy,NSVPS)进行剪枝,并提出一种变样本量学习最小二乘支持向量机(Variable Samples Learning LS-SVM,VSL-LSSVM)算法。将训练集分为工作集和非工作集,采用 KKT 条件选取非工作集中的特定样本加入工作集后进行训练。此外,为避免对高维矩阵进行求逆,本文采用双样本增量、多样本减量学习以及单样本减量学习方法。

1　最小二乘支持向量机

对于二分类问题,假设训练集 T 由 l 个样本点组成:

$$T=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_l,y_l)\} \tag{1}$$

其中,$\boldsymbol{x}_i\in R^n$ 是输入向量,$y_i\in\{-1,1\}$ 是 $\boldsymbol{x}_i$ 所属的类别。基于等式约束和最小二乘损失函数,文献[6]给出了如下的优化问题:

$$\min_{\boldsymbol{\omega},b,e} E(\boldsymbol{\omega},b,e)=\frac{1}{2}\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega}+\frac{\gamma}{2}\sum_{k=1}^{l}e_k^2 \tag{2}$$

$$\text{s.t. } y_k[\boldsymbol{\omega}^{\mathrm{T}}\varphi(\boldsymbol{x}_k)+b]=1-e_k,k=1,2,\cdots,l \tag{3}$$

其中,$\boldsymbol{\omega}$ 是权向量,γ 是正则化参数,e_k 是松弛变量,$\varphi(\cdot)$是从输入空间到高维特征空间的非线性映射,b 是一个偏量。

最优性条件为:

$$\begin{pmatrix}0 & \boldsymbol{Y}^{\mathrm{T}}\\ \boldsymbol{Y} & \boldsymbol{\Omega}+\gamma^{-1}\boldsymbol{I}\end{pmatrix}\begin{pmatrix}b\\ \boldsymbol{\alpha}\end{pmatrix}=\begin{pmatrix}0\\ 1\end{pmatrix} \tag{4}$$

其中,$\Omega_{ij}=y_iy_j\varphi(\boldsymbol{x}_i)^{\mathrm{T}}\varphi(\boldsymbol{x}_j)=K(\boldsymbol{x}_i,\boldsymbol{x}_j)$,$\boldsymbol{Y}=(y_1,y_2,\cdots,y_l)^{\mathrm{T}}$,$\boldsymbol{I}=(1,1,\cdots,1)^{\mathrm{T}}$,$\boldsymbol{\alpha}=(\alpha_1,\alpha_2,\cdots,\alpha_l)^{\mathrm{T}}$。

解式(4)得到 $\boldsymbol{\alpha}$ 和 b 后,针对输入 $\boldsymbol{x}$,其输出 $y(\boldsymbol{x})$可以根据下式进行计算:

$$y(\boldsymbol{x})=Sgn(\sum_{k=1}^{l}\alpha_k y_k K(\boldsymbol{x},\boldsymbol{x}_k)+b) \tag{5}$$

2　最大冲突对

本文针对分类 LS-SVM 给出最大冲突对,计算过程如下:

令:

$$\tilde{\boldsymbol{\omega}}=(\boldsymbol{\omega},\sqrt{\gamma}\boldsymbol{e})^{\mathrm{T}} \tag{6}$$

$$\tilde{\varphi}(\boldsymbol{x}_i)=\left(y_i\varphi(\boldsymbol{x}_i),\frac{\boldsymbol{\xi}_i}{\sqrt{\gamma}}\right) \tag{7}$$

$$\boldsymbol{e}=(e_1,e_2,\cdots,e_l) \tag{8}$$

$$\boldsymbol{\xi}_i=(0,0,\cdots,1,0,\cdots,0) \tag{9}$$

其中,$\boldsymbol{\xi}_i$ 向量中第 i 个元素等于 1,其余都等于 0。则优化问题式(2)、式(3)可表示为:

$$\min_{(\tilde{\boldsymbol{\omega}},b)} E(\tilde{\boldsymbol{\omega}},b)=\frac{1}{2}\tilde{\boldsymbol{\omega}}^{\mathrm{T}}\tilde{\boldsymbol{\omega}} \tag{10}$$

$$\text{s.t. } \tilde{\boldsymbol{\omega}}^{\mathrm{T}}\tilde{\varphi}(\boldsymbol{x}_k)+y_kb-1=0,k=1,2,\cdots,l \tag{11}$$

上述问题的对偶问题为:

$$\max_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha})=-\frac{1}{2}\|\tilde{\boldsymbol{\omega}}(\boldsymbol{\alpha})\|^2+\sum_{k=1}^{l}\alpha_k \tag{12}$$

$$\text{s.t. } \sum_{k=1}^{l}\alpha_k y_k=0 \tag{13}$$

对偶问题式(12)、式(13)的 Lagrange 函数为:

$$L(\boldsymbol{\alpha})=-\frac{1}{2}\|\tilde{\boldsymbol{\omega}}(\boldsymbol{\alpha})\|^2+\sum_{k=1}^{l}\alpha_k+\beta\sum_{k=1}^{l}\alpha_k y_k \tag{14}$$

令:

$$F_i=-\frac{\partial f}{\partial\alpha_i}=\tilde{\boldsymbol{\omega}}(\boldsymbol{\alpha})\tilde{\varphi}(\boldsymbol{x}_i)-1=\sum_{k=1}^{l}\alpha_k\tilde{K}(\boldsymbol{x}_i,\boldsymbol{x}_k)-1 \tag{15}$$

在式(15)中:

$$\tilde{K}(\boldsymbol{x}_i,\boldsymbol{x}_k)=\tilde{\varphi}(\boldsymbol{x}_i)\tilde{\varphi}(\boldsymbol{x}_k)=\begin{cases}y_ky_iK(\boldsymbol{x}_i,\boldsymbol{x}_k)+\dfrac{1}{\gamma},i=k\\ y_ky_iK(\boldsymbol{x}_i,\boldsymbol{x}_k),i\neq k\end{cases} \tag{16}$$

则对偶问题式(12)、式(13)的 KKT 条件为:

$$\frac{\partial L}{\partial\alpha_i}=0\rightarrow\beta-y_iF_i=0 \tag{17}$$

因此,可选最大冲突对$(\boldsymbol{x}_{i_{\mathrm{up}}},\boldsymbol{x}_{i_{\mathrm{low}}})$加入工作集。其中:

$$i_{\mathrm{up}}=\arg\max_i y_iF_i \tag{18}$$

$$i_{\mathrm{low}}=\arg\min_i y_iF_i \tag{19}$$

本文策略为:将$(\boldsymbol{x}_{i_{\mathrm{up}}},\boldsymbol{x}_{i_{\mathrm{low}}})$加入工作集中进行双样本增量学习,到工作集样本对应的目标函数值变化非常小时停止在工作集中加入新样本,即当满足式(20)时停止增量。

$$\left|\frac{f_1-f_2}{f_1}\right|<\varepsilon \tag{20}$$

其中,f_1为 $f=\frac{1}{2}\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega}+\frac{\gamma}{2}\sum_k e_k^2$ 的前一次循环值,f_2是本次循环值。

3 剪枝策略

3.1 负松弛变量剪枝策略

在 SVM 算法中，$\alpha_i = 0$ 对应的样本为非支持向量（如图 1 中的 Hyperplane + 上方的所有正类样本和 Hyperplane − 下方的所有负类样本），其不影响分类器。

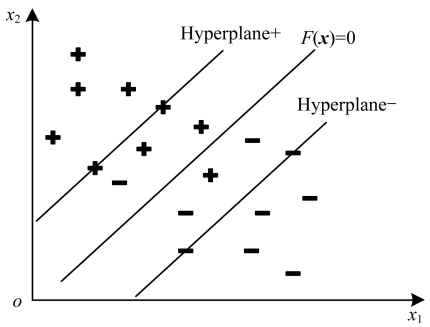

图 1 SVM 算法样本示意图

定理 1[2] 设 $\boldsymbol{\alpha}^* = (\alpha_1^*, \alpha_2^*, \cdots, \alpha_l^*)^{\mathrm{T}}$ 是 SVM 算法对偶问题的最优解，对任意给定的 $0 < \alpha_k^* < C$，$b^* = y_k - \sum_{i=1}^{l} y_i \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}_k)$，则有：

1）对任意的 $0 < \alpha_k^* < C$，$y_k(\sum_{i=1}^{l} y_i \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}_k) + b^*) = 1$（在图 1 中对应 Hyperplane + 上的正类和 Hyperplane − 上的负类）。

2）对任意的 $\alpha_k^* = C$，$y_k(\sum_{i=1}^{l} y_i \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}_k) + b^*) \leqslant 1$（在图 1 中对应 Hyperplane + 下方的正类和 Hyperplane − 上方的负类）。

3）对任意的 $\alpha_k^* = 0$，$y_k(\sum_{i=1}^{l} y_i \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}_k) + b^*) \geqslant 1$（在图 1 中对应 Hyperplane + 上方的正类和 Hyperplane − 下方的负类）。

在定理 1 中，C 为惩罚参数。由于 LS-SVM 算法将 SVM 算法中的不等式约束转换为等式约束，因此所有的约束条件都为起作用约束。故在 LS-SVM 算法中，几乎所有的 α_i 都不等于 0。通过比较 LS-SVM 算法与 SVM 算法，本文提出一种基于松弛变量 e_k 的剪枝策略 NSVPS。该策略具体步骤如下：

由式（3）得到下式：

$$\boldsymbol{\omega}^{\mathrm{T}} \varphi(\boldsymbol{x}_k) + b = y_k - y_k e_k \tag{21}$$

则当 $y_k = 1$ 时，有：

$$\boldsymbol{\omega}^{\mathrm{T}} \varphi(\boldsymbol{x}_k) + b + e_k = 1 \tag{22}$$

如图 2 所示，在 Hyperplane + 上方，正类样本对应的 e_k 小于 0；在 Hyperplane + 上，正类样本对应的 e_k 等于 0；在 Hyperplane + 到 $F(\boldsymbol{x}) = 0$ 之间，正类样本对应的 e_k 大于 0 小于 1；在 $F(\boldsymbol{x}) = 0$ 下方，正类样本对应的 e_k 大于 1。

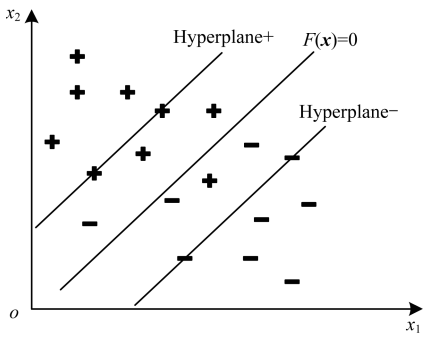

图 2 LS-SVM 算法样本示意图

与定理 1 对比得到以下对应关系：

$$\begin{cases} \alpha_k = 0, \text{相当于 } e_k < 0 \\ 0 < \alpha_k < C, \text{相当于 } e_k = 0 \\ \alpha_k = C, \text{相当于 } e_k > 0 \end{cases} \tag{23}$$

当 $y_k = -1$ 时式（21）变为：

$$\boldsymbol{\omega}^{\mathrm{T}} \varphi(\boldsymbol{x}_k) + b - e_k = -1 \tag{24}$$

同理，可以得到以下对应关系：

$$\begin{cases} \alpha_k = 0, \text{相当于 } e_k < 0 \\ 0 < \alpha_k < C, \text{相当于 } e_k = 0 \\ \alpha_k = C, \text{相当于 } e_k > 0 \end{cases} \tag{25}$$

由式（23）和式（25）可知，在 SVM 算法中，$\alpha_k = 0$ 相当于 LS-SVM 算法中 $e_k < 0$。因此，可以将 $e_k < 0$ 对应的样本从工作集中剔除，并对剩余的样本进行再次训练，重复此步骤直至工作集中无 $e_k < 0$ 的样本（在移除 $e_k < 0$ 的样本时可采用多样本减量学习方法）。NSVPS 剪枝策略具体步骤如下：

算法 1 NSVPS 剪枝策略

设剪枝前工作集为 W。

步骤 1 训练 W，计算工作集 W 里每个样本的 e_k。

步骤 2 检查工作集 W 里是否存在 $e_k < 0$ 的样本，若存在，则从工作集 W 中移除所有 $e_k < 0$ 的样本，转至步骤 1；否则，输出 W。

图 1 所示为 SVM 算法分类后的结果，图 2 所示为 LS-SVM 算法分类后的结果。由图 2 可以看出，$e_k > 1$ 对应于错划分的样本。

在工作集中剔除样本时，由于工作集里的样本满足 KKT 条件，因此算法 1 中的 e_k 由下式求得：

$$e_k = \frac{\alpha_k}{\gamma} \tag{26}$$

3.2 基于对偶目标函数差的剪枝策略

从工作集中将上述样本移除后还可以进一步移除一些样本。文献[11]提出一种基于序列最小优化（Sequential Minimal Optimization，SMO）的剪枝策略，其设计思想为移除对对偶目标函数影响较小的样本。

将样本 $\boldsymbol{x}_k$ 从工作集中移除后引起的对偶目标函数（式（12））的差为：

$$\Delta(f)=f(\boldsymbol{\alpha})-f(\tilde{\boldsymbol{\alpha}})=\frac{1}{2}\sum_{i=1}^{l}\alpha_i(1-F_i)-\frac{1}{2}\sum_{i=1,i\neq k}^{l}\alpha_i(1-\tilde{F}_i)=\frac{1}{2}\alpha_k^2\tilde{K}(\boldsymbol{x}_k,\boldsymbol{x}_k)-\alpha_k F_k \tag{27}$$

其中,$\tilde{F}_i$可由下式计算得到:

$$\tilde{F}_i=F_i-\alpha_i\tilde{K}(\boldsymbol{x}_k,\boldsymbol{x}_i) \tag{28}$$

由于本文采用的是分类形式的 LS-SVM,式(27)中的 $\tilde{K}(\boldsymbol{x}_k,\boldsymbol{x}_i)$ 可由式(16)求得,F_k 由式(15)求得。以下是基于对偶目标函数差的剪枝策略(Dual Objective Function Deviation Pruning Strategy,DOFDPS)的剪枝过程。

算法 2　DOFDPS 剪枝策略

设剪枝前工作集为 W_{old}。

步骤 1　令 K 等于使用 DOFDPS 剪枝前工作集中样本数量的 $s\%$(剪枝步长)。

步骤 2　从工作集 W_{old} 中移除使 $|\Delta(f)|$ 最小的样本(该过程进行 K 次,每移除一次分别采用式(28)和式(27)更新一次剩余样本的 F_i 和 $|\Delta(f)|$),移除样本后的工作集记为 W_{new}。

步骤 3　对工作集 W_{new} 进行训练,若此时算法的性能没有降低,则令 $W_{\text{old}}=W_{\text{new}}$ 并转入步骤 2,否则,令 $K=K-1$ 并转到步骤 4。

步骤 4　若 $K\neq 0$,则转入步骤 2,否则,停止计算并输出 W_{old} 所对应的分类器。

为提高算法的运算速度,在剪枝过程中,若 $K=1$,则可采用在第 4 节介绍的单样本减量学习方法,若 $K\neq 1$,则可采用多样本减量学习方法。

4　变样本量学习的 LS-SVM 算法

4.1　双样本增量学习

设增量前有:

$$\begin{pmatrix}0 & \boldsymbol{Y}^{\mathrm{T}}\\ \boldsymbol{Y} & \boldsymbol{\Omega}+\gamma^{-1}\boldsymbol{I}\end{pmatrix}=\boldsymbol{A}_{\text{old}}$$

$$\begin{pmatrix}b\\ \boldsymbol{\alpha}\end{pmatrix}=\boldsymbol{X}_{\text{old}}$$

$$\begin{pmatrix}0\\ 1\end{pmatrix}=\boldsymbol{B}_{\text{old}}$$

则式(4)为:

$$\boldsymbol{A}_{\text{old}}\boldsymbol{X}_{\text{old}}=\boldsymbol{B}_{\text{old}} \tag{29}$$

式(29)的解为:

$$\boldsymbol{X}_{\text{old}}=\boldsymbol{A}_{\text{old}}^{-1}\boldsymbol{B}_{\text{old}} \tag{30}$$

增加 2 个样本$(\boldsymbol{x}_{i_{\text{up}}},\boldsymbol{x}_{i_{\text{low}}})$后有:

$$\boldsymbol{A}_{\text{new}}=\begin{pmatrix}\boldsymbol{A}_{\text{old}} & \boldsymbol{Q}^{\mathrm{T}}\\ \boldsymbol{Q} & \boldsymbol{U}\end{pmatrix} \tag{31}$$

其中:

$$\boldsymbol{Q}=\begin{pmatrix}1 & \Omega_{n+1,1} & \Omega_{n+1,2} & \cdots & \Omega_{n+1,n}\\ 1 & \Omega_{n+2,1} & \Omega_{n+2,2} & \cdots & \Omega_{n+2,n}\end{pmatrix} \tag{32}$$

$$\boldsymbol{U}=\begin{pmatrix}\Omega_{n+1,n+1}+\dfrac{1}{\gamma} & \Omega_{n+1,n+2}\\ \Omega_{n+2,n+1} & \Omega_{n+2,n+2}+\dfrac{1}{\gamma}\end{pmatrix} \tag{33}$$

若令:

$$\boldsymbol{A}_{\text{new}}^{-1}=\begin{pmatrix}\boldsymbol{A}_{11} & \boldsymbol{A}_{12}\\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22}\end{pmatrix} \tag{34}$$

则由

$$\boldsymbol{A}_{\text{new}}\boldsymbol{A}_{\text{new}}^{-1}=\begin{pmatrix}\boldsymbol{E}_{11} & 0\\ 0 & \boldsymbol{E}_{22}\end{pmatrix} \tag{35}$$

可解出:

$$\boldsymbol{A}_{11}=\boldsymbol{A}_{\text{old}}^{-1}-\boldsymbol{A}_{\text{old}}^{-1}\boldsymbol{Q}^{\mathrm{T}}(\boldsymbol{Q}\boldsymbol{A}_{\text{old}}^{-1}\boldsymbol{Q}^{\mathrm{T}}-\boldsymbol{U})^{-1}\boldsymbol{Q}\boldsymbol{A}_{\text{old}}^{-1} \tag{36}$$

$$\boldsymbol{A}_{12}=\boldsymbol{A}_{\text{old}}^{-1}\boldsymbol{Q}^{\mathrm{T}}(\boldsymbol{Q}\boldsymbol{A}_{\text{old}}^{-1}\boldsymbol{Q}^{\mathrm{T}}-\boldsymbol{U})^{-1} \tag{37}$$

$$\boldsymbol{A}_{21}=(\boldsymbol{Q}\boldsymbol{A}_{\text{old}}^{-1}\boldsymbol{Q}^{\mathrm{T}}-\boldsymbol{U})^{-1}\boldsymbol{Q}\boldsymbol{A}_{\text{old}}^{-1} \tag{38}$$

$$\boldsymbol{A}_{22}=(\boldsymbol{U}-\boldsymbol{Q}\boldsymbol{A}_{\text{old}}^{-1}\boldsymbol{Q}^{\mathrm{T}})^{-1} \tag{39}$$

这样只需保存 $\boldsymbol{A}_{\text{old}}^{-1}$ 就可以求出 $\boldsymbol{A}_{\text{new}}^{-1}$,避免了对高维矩阵进行求逆。求得 $\boldsymbol{A}_{\text{new}}^{-1}$ 后可得:

$$\boldsymbol{X}_{\text{new}}=\boldsymbol{A}_{\text{new}}^{-1}\boldsymbol{B}_{\text{new}} \tag{40}$$

4.2　多样本减量学习

设工作集 W_n 中有 n 个样本,此时 W_n 所对应的系数矩阵为 $\boldsymbol{A}_{\text{old}}$。假设有 $k(k>1)$ 个样本可从工作集 W_n 中移除,现将此 k 个样本依次移到工作集 W_n 队列后面,步骤为:设第 i 次移动的样本是工作集 W_n 中的第 j 个样本,则将第 j 个样本与工作集 W_n 中的第 $n-i+1$ 个样本进行交换,同时将 A_{old}^{-1} 中的第 $j+1$ 行与第 $n-i+2$ 行、第 $j+1$ 列与第 $n-i+2$ 列分别进行交换。

变换后满足:

$$\boldsymbol{A}_{\text{old}}^{-1}=\begin{pmatrix}\boldsymbol{P} & \boldsymbol{N}^{\mathrm{T}}\\ \boldsymbol{N} & \boldsymbol{S}\end{pmatrix} \tag{41}$$

其中,$\boldsymbol{P}$ 为 $(n-k)\times(n-k)$ 阶矩阵,$\boldsymbol{N}$ 为 $k\times(n-k)$ 阶矩阵,$\boldsymbol{S}$ 为 $k\times k$ 阶矩阵。现从工作集 W_n 的队列中移除后面的 k 个样本,则由文献[17]可得:

$$\boldsymbol{A}_{\text{new}}^{-1}=\boldsymbol{P}-\boldsymbol{N}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{N} \tag{42}$$

因此,在计算时只需存储减量之前的逆矩阵就可求出减量之后的逆矩阵,由此避免了对高维矩阵进行求逆。

4.3　单样本减量学习

将 $\boldsymbol{x}_k$ 从工作集中移除。移除后,式(4)的解可表示为:

$$\boldsymbol{X}_{\text{new}}=\boldsymbol{A}_{\text{new}}^{-1}\boldsymbol{B}_{\text{new}} \tag{43}$$

其中,$\boldsymbol{A}_{\text{new}}^{-1}=(\hat{\alpha}_{ij})$ 为减量后的系数矩阵的逆。设未减量之前的系数矩阵的逆为 $\boldsymbol{A}_{\text{old}}^{-1}=(\alpha_{ij})$,则根据文献[18]可得:

$$\hat{\alpha}_{ij}=\begin{cases}\alpha_{ij}-\alpha_{kk}^{-1}\alpha_{ik}\alpha_{kj}, & i,j<k\\ \alpha_{i+1,j+1}-\alpha_{kk}^{-1}\alpha_{i+1,k}\alpha_{k,j+1}, & i,j\geqslant k\end{cases} \tag{44}$$

4.4　VSL-LSSVM 算法具体步骤

本文 VSL-LSSVM 算法步骤如下。

算法 3 VSL-LSSVM 算法

步骤 1 从训练集中随机选取 n 个样本作为初始工作集 W,剩余的样本为非工作集,求解式(4),构造分类器。

步骤 2 根据式(15)计算所有非工作集样本的 F_i,选取 y_iF_i 最大和最小的 2 个样本加入工作集并进行训练(可采用双样本增量学习方法)得到新的分类器 $y_{\text{incr}}(x)$ 与增量后的工作集 W_{incr}。检查是否满足式(20),若满足,转到步骤 3,否则,继续执行步骤 2。

步骤 3 采用 NSVPS 策略对增量后的工作集 W_{incr} 进行剪枝,得到新的工作集 W_{new}。

步骤 4 采用 DOFDPS 策略对减量后的工作集 W_{new} 进行剪枝,直至剪枝结束。输出剪枝后的分类器。

5 实验结果及分析

为测试算法性能,本文采用 UCI(University of California Irvine machine learning repository)数据库中的真实数据对算法进行验证分析。

5.1 数据集

表 1 所示为本文从 UCI 数据库中随机选取的用于分类的数据集以及高斯核参数 δ^2 和正则化参数 γ。对多分类数据集进行如下处理:在 Iris 数据集中将 Iris-setosa 和 Iris-virginica 标记为同一类,将 Iris-versicolor 标记为另一类;在 Wine 数据集中将 1 和 3 标记为同一类,将 2 标记为另一类。

表 1 UCI 数据库中的数据集及参数

数据集	样本数	δ^2	γ
Iris	150	1	64
Wine	178	1	90
IOnosphere(IO)	351	2	15
Balance Scale(BS)	625	1	16
Tic-Tac-Toe(TTT)	958	64	256
Parking Speech(PS)	1 040	32	64
Banknote Authentication(BA)	1 372	1	64
Steel Plates Faults(SPF)	1 941	16	64

5.2 实验说明

本次实验所涉及的运行时间指运行一次 10-折交叉验证所需的全部时间,单位为 s;测试精度指测试集分类正确的样本占测试集总样本的比例;支持向量(Support Vectors,SVs)指剪枝后工作集剩余样本;初始工作集大小指初始工作集里的样本数量,单位为个。采用稀疏率(Sparse Rate,SR)作为稀疏性的度量标准,其计算公式如下:

$$SR=\frac{N-N_{\text{sv}}}{N}$$

其中,N 为训练集样本总数,N_{sv} 为支持向量个数。

本文采用 SMO 算法[19]、SMO-new 算法[11]、ISLS-SVM 算法[12]作为对比算法。为正确衡量算法的性能,采用 10-折交叉验证法进行测试。对每个数据集进行 10 次 10-折交叉验证并将各指标(运行时间、测试精度、剪枝后剩余样本数、稀疏率)的平均值作为结果进行比较。所涉及的所有算法均采用 C# 编写,运行环境为 Visual Studio 2017。

5.3 对 NSVPS 策略的分析

首先,为验证 NSVPS 策略的有效性,本文直接对式(4)求解,然后采用 NSVPS 策略进行剪枝,记录剪枝前后的测试精度、剪枝后的支持向量个数以及剪枝次数,结果如表 2 所示。

表 2 NSVPS 策略实验结果

数据集	剪枝前		剪枝后		剪枝次数
	测试精度/%	SVs	测试精度/%	SVs	
Iris	96.47	133	96.47	34	5
Wine	97.45	158	97.45	25	5
BS	93.54	560	95.77	83	7
TTT	98.35	860	98.35	193	5

从表 2 可以看出,采用 NSVPS 剪枝策略的剪枝效率较高,只需少量剪枝操作就可使每个数据集上剪枝率(被剪枝 SVs 数/剪枝前 SVs 数)达到 75% ~ 85%,且并无精度损失。

5.4 剪枝步长对 DOFDPS 策略性能的影响

分析剪枝步长对 DOFDPS 策略性能的影响,实验结果如图 3 所示。

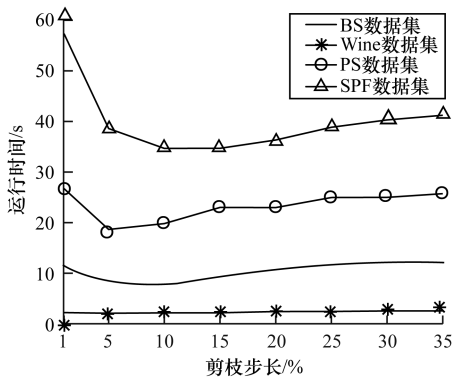

图 3 剪枝步长对 DOFDPS 策略运算速率的影响

从图 3 可以看出,剪枝步长对 DOFDPS 的运行时间有一定影响:步长过大或者过小都会使 DOFDPS 运行时间增加。当数据集较小时,剪枝步长对算法运行时间的影响非常小,可忽略不计,但是当数据集较大时,剪枝步长对算法的运行时间具有较明显的影响。从图 3 中的 SPF 数据集可以看出,当剪枝步长为 1% 时运行时间为 57.36 s,当剪枝步长增加到 10% 时运行时间为 34.68 s。在图 3 中,算法运行时间短的剪枝步长都处于 5% ~ 15% 之间,因此,本文建议的剪枝步长为 5% ~ 15%。

5.5 剪枝策略分析

对本文提出的剪枝策略进行对比分析,分别将 NSVPS、DOFDPS、NSVPS + DOFDPS 剪枝策略在 5 个数据集上进行比较,结果如表 3 所示。

表 3　剪枝策略性能对比

剪枝策略	Wine 数据集		BS 数据集		PS 数据集		BA 数据集		SPF 数据集	
	SVs	运行时间/s	SVs	运行时间/s	SVs	运行时间/s	SVs	运行时间/s	SVs	运行时间/s
NSVPS	27	1.23	79	5.56	63	11.32	52	7.04	53	23.32
DOFDPS	29	3.42	72	11.54	61	24.69	57	20.25	42	55.26
NSVPS + DOFDPS	17	2.80	45	7.26	40	19.27	21	10.02	16	35.33

从表 3 可以看出,相对结合 NSVPS 和 DOFDPS 2 种剪枝策略剪枝得到的支持向量个数,单独采用 NSVPS 或 DOFDPS 剪枝策略剪枝得到的支持向量个数较多。运用 DOFDPS 剪枝策略剪枝时,如果算法性能降低,则会出现对同一工作集进行多次剪枝的现象。若增量后工作集里的样本数较多时,直接采用 DOFDPS 对工作集进行剪枝会导致对高维矩阵进行多次运算而消耗较多的时间。NSVPS 不会像 DOFDPS 对同一工作集进行多次剪枝,因而剪枝次数少,剪枝速度快。在增量结束后,针对工作集先采用 NSVPS 剪枝策略,然后采用 DOFDPS 剪枝策略,这样可避免在工作集较大时直接采用 DOFDPS 剪枝策略产生的运算量过大问题,同时可以将支持向量个数减少至最少程度。

5.6　初始工作集大小对算法的影响

针对不同大小的初始工作集,VSL-LSSVM 算法的运行结果如图 4 所示。从图 4 可以看出,初始工作集大小对算法的精度和稀疏率几乎没有影响,但是对算法的运行速度有一定影响,随着初始工作集样本数量的增长,算法运行时间不断增长,在 PS 数据集上差距高达 46.73 s。产生该结果的原因为:1)初始工作集是随机选取的,选取的初始工作集里的样本很有可能不是支持向量,若这样的样本较多,则在剪枝阶段需要减去很多的样本,因此,会消耗很长的时间;2)若工作集里样本较多,则在解式(4)时也会消耗很长的时间。因此,应选取样本数尽量少的初始工作集以减少算法的运行时间。

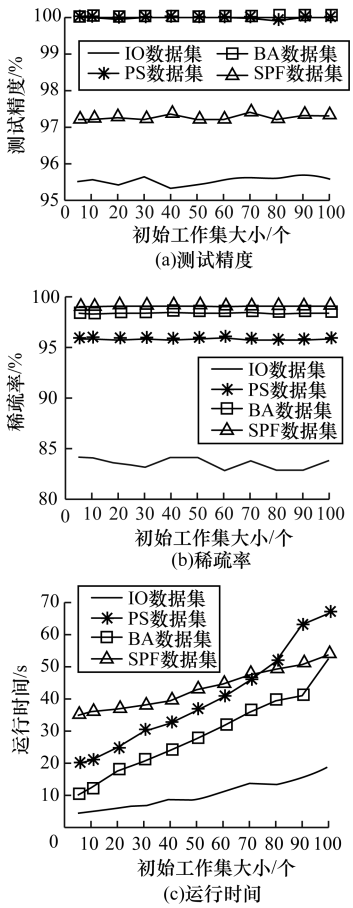

图 4　初始工作集大小对算法的影响

5.7　算法性能对比分析

表 4 所示为 4 种算法在不同数据集上的运行结果。其中,在 ISLS-SVM 算法中,选择使算法精度无损失的最小 L_0 范数。

表 4　4 种算法测试结果

算法	测试指标	Iris 数据集	Wine 数据集	IO 数据集	PS 数据集	BA 数据集	SPF 数据集
VSL-LSSVM 算法	测试精度/%	96.47	97.47	95.94	99.81	100	97.21
	SVs	19	16	49	39	20	16
	运行时间/s	1.27	3.08	4.23	17.27	10.02	34.92
SMO-new 算法	测试精度/%	97.65	98.50	95.67	99.34	100	97.21
	SVs	37	58	111	314	193	164
	运行时间/s	66.23	213.24	380.21	1 233.38	4 326.86	8 843.36
ISLS-SVM 算法	测试精度/%	97.65	98.97	94.03	98.87	99.56	97.21
	SVs	32	44	65	52	43	41
	运行时间/s	24.79	52.74	275.55	978.56	4 630.42	9 540.78
SMO 算法	测试精度/%	96.47	98	94.31	98.68	100	97.21
	SVs	129	152	301	903	1 196	1 686
	运行时间/s	12.50	36.81	40.58	270.61	1 238.04	4 620.26

首先采用 Nemenyi 检验方法[20]检验 4 种算法精度是否有显著性差异。根据表 4 中的数据可得到算法精度比较序值表:在每个数据集上根据测试精度由好到坏进行排序,并赋予对应序值 1、2、3、4,若算法测试精度相同,则平分序值。序值结果如表 5 所示。

表 5 算法精度比较序值结果

数据集	SMO-new	VSL-LSSVM	ISLS-SVM	SMO
Iris	1.5	2.5	1.5	2.5
Wine	2	4	1	3
IO	2	1	4	3
PS	2	1	3	4
BA	2	2	4	2
SPF	2.5	2.5	2.5	2.5
平均序值	2	2.167	2.667	2.833

采用 $CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$ 公式计算出平均序值差别的临界值域。其中,N 为数据集的个数。在显著度 $\alpha = 0.05$、算法数量 $k = 4$ 时,Tukey 分布的临界值 q_α 为 2.569,则平均序值差别的临界值域为 1.915。因为每 2 个算法之间的平均序值的差值小于临界值,所以在 $\alpha = 0.05$ 的显著性检验下,可接受这 4 种算法的精度无显著性差别的假设。因此,VSL-LSSVM 算法可在无精度损失的情况下得到稀疏解。

其次从稀疏度的角度看,这 4 种算法稀疏度由高到低为 VSL-LSSVM > ISLS-SVM > SMO-new > SMO。这是由于 VSL-LSSVM 算法在增量阶段采用 KKT 条件有效地选取了最有可能成为支持向量的一批样本,在减量阶段有效结合 2 种剪枝策略对工作集中冗余的样本进行了剔除,使其稀疏性得到进一步提升。ISLS-SVM 算法可以选定 L_0 范数来指定算法的稀疏度,但是当 L_0 过小时将无法达到指定的稀疏度。SMO-new 算法在每次进行剪枝时,如果性能变差就停止剪枝(此时工作集中可能还存在冗余样本)。SMO 算法不进行剪枝操作,支持向量的数量几乎等于训练集样本的数量。

从算法运行时间来看,4 种算法由快到慢的顺序为 VSL-LSSVM > SMO > SMO-new > ISLS-SVM。这是因为 VSL-LSSVM 算法在训练时,工作集里的样本数量远小于训练集中的样本数量,并且结合了矩阵增量、减量运算,故相比其他算法,其运算复杂度最低。而 SMO 算法在训练时,虽然每次只对2 个样本进行操作,但是需要对整个训练集进行多次迭代,因此,需要较长的运算时间。SMO-new 算法是一种基于 SMO 算法的改进算法,运算时需要先求解 SMO 算法然后再进行剪枝,故其运算复杂度比 SMO 算法高。ISLS-SVM 算法的复杂度随着稀疏度的降低而降低,但是由于需要对原非系数 LS-SVM 算法进行直接求解,因此算法的复杂度较高,当数据集较大、稀疏性要求较高时,算法的复杂度会变得更高。

由以上分析可知,相对 SMO、SMO-new、ISLS-SVM 算法,本文 VSL-LSSVM 算法具有稀疏性高、无精度损失、运算速度快的明显优势。

6 结束语

LS-SVM 算法解的稀疏性较低。为此,本文提出一种预剪枝 LS-SVM 算法 VSL-LSSVM。该算法分为样本增量和样本减量 2 个阶段。在增量阶段,根据 KKT 条件选取最大冲突对(最大程度违反 KKT 条件的2 个样本)加入工作集进行训练。在减量阶段,采用本文提出的负松弛变量剪枝策略 NSVPS 及基于对偶目标函数差的剪枝策略 DOFDPS 对工作集进行剪枝。实验结果表明,VSL-LSSVM 算法具有稀疏性高、运算速度快以及无精度损失的优点。下一步将探索高效、高精度的多分类学习算法,以进一步提升本文算法的分类能力。

参考文献

[1] CHERKASSKY V. The nature of statistical learning theory[J]. Technometrics,2002,38(4):409.

[2] 邓乃杨,田英杰. 支持向量机——理论、算法与拓展[M]. 北京:科学出版社,2009.

[3] DENG N,TIAN Y,ZHANG C. Support vector machines:optimization based theory,algorithms,and extensions[M]. [S.l.]:Chapman and Hall/CRC,2012.

[4] 李琦,李晓航,邢丽萍,等. 基于 lp-范数约束的 LSSVR 多核学习算法[J]. 控制与决策,2015,30(9):1603-1608.

[5] BRABANTER K D,BRABANTER J D,SUYKENS J A K,et al. Optimized fixed-size kernel models for large data sets[J]. Computational Statistics and Data Analysis,2010,54(6):1484-1504.

[6] SUYKENS J A K, VANDEWALLE J. Least squares support vector machine classifiers[J]. Neural Processing Letters,1999,9(3):293-300.

[7] SUYKENS J A K,BRABANTER J D,LUKAS L,et al. Weighted least squares support vector machines:robustness and sparse approximation[J]. Neurocomputing,2002,48(4):85-105.

[8] SUYKENS J A K,LUKAS L,VANDEWALLE J. Sparse approximation using least square vector machines[C]// Proceedings of 2000 IEEE International Symposium on Circuits and Systems. Washington D.C.,USA:IEEE Press,2000:757-760.

[9] SUYKENS J A K,LUKAS L,VANDEWALLE J. Sparse least squares support vector machine classifiers[C]// Proceedings of European Symposium on Artificial Neural Networks. Berlin,Germany:Springer,2000:37-42.

(下转第 205 页)

[8] BENGIO Y,COURVILLE A,VINCENT P. Representation learning: a review and new perspectives [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012,35(8):1798-1828.
[9] 李彧晟,王芳,朱雨倩,等. 基于深度置信网络的中国传统乐器分类方法: CN106328121A [P]. [2017-01-11].
[10] HAN Y,KIM J,LEE K,et al. Deep convolutional neural networks for predominant instrument recognition in polyphonic music [J]. IEEE/ACM Transactions on Audio Speech and Language Processing,2017,25(1): 208-221.
[11] PEETERS G, GIORDANO B, SUSINI P, et al. The timbre toolbox: audio descriptors of musical signals[J]. Journal of the Acoustical Society of America, 2011, 130(5):2902-2916.
[12] PONS J,SERRA X. Designing efficient architectures for modeling temporal features with convolutional neural networks [C]//Proceedings of IEEE International Conference on Acoustics,Speech and Signal Processing. Washington D. C. ,USA:IEEE Press,2017:2472-2476.
[13] MEDDIS R, LOPEZPOVEDA E, FAY R R, et al. Computational models of the auditory system [M]. Berlin,Germany:Springer,2010:135-149.
[14] KALCHBRENNER N,GREFENSTETTE E,BLUNSOM P. A convolutional neural network for modelling sentences [EB/OL]. [2017-11-11]. https://arxiv. org/pdf/1404. 2188. pdf.
[15] 李乐,王玉英,李小霞. 一种改进的小波能量熵语音端点检测算法[J]. 计算机工程,2017,43(5):268-274.
[16] University of IOWA electronic music studio: a musical instrument database [EB/OL]. [2017-11-09]. http://theremin. music. uiowa. edu/ MISflute. html.
[17] Google Inc: Tensorflow for deep learning [EB/OL]. [2017-11-09]. https://www. tensorflow. org.
[18] HE K, ZHANG X, REN S, et al. Delving deep into rectifiers: surpassing human-level performance on imagenet classification[C]//Proceedings of 2015 IEEE International Conference on Computer Vision. Washington D. C. ,USA:IEEE Press,2015:1026-1034.
[19] GLOROT X, BENGIO Y. Understanding the difficulty of training deep feedforward neural networks[J]. Journal of Machine Learning Research,2010,9:249-256.
[20] KINGMA D P, BA J. Adam: a method for stochastic optimization[EB/OL]. [2017-11-11] http://cn. arxiv. org/pdf/1412. 6980v9.
[21] HINTON G E, SRIVASTAVA N, KRIZHEVSKY A, et al. Improving neural networks by preventing co-adaptation of feature detectors[J]. Computer Science, 2012,3(4):212-223.
[22] MAATEN L V D, HINTON G. Viualizing data using t-SNE[J]. Journal of Machine Learning Research,2008, 9(2605):2579-2605.

编辑　刘盛龄

(上接第 198 页)

[10] KRUIF B J D,VRIES T J A D. Pruning error minimization in least squares support vector machines [J]. IEEE Transactions on Neural Networks,2003,14(3):696-702.
[11] ZENG X,CHEN X W. SMO-based pruning methods for sparse least squares support vector machines [M]. Washington D. C. ,USA:IEEE Press,2005.
[12] LÓPEZ L J, BRABANTER K D, DORRONSORO J R, et al. Sparse LSSVMs with L0-norm minimization [C]// Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learing. Berlin,Germany:Springer,2011:189-194.
[13] KAIZHU H. Sparse learning for support vector classification[J]. Pattern Recognition Letters,2010,31(13):1944-1951.
[14] QI L,LI X H,BA W. Sparse least squares support vector machine with L_0-norm in primal space [C]// Proceedings of 2015 IEEE International Conference on Information and Automation. Washington D. C. , USA: IEEE Press,2015:2778-2783.
[15] 杨晓伟,路节,张广全. 一种高效的最小二乘支持向量机分类器剪枝算法[J]. 计算机研究与发展,2007, 44(7):1128-1136.
[16] 马跃峰,梁循,周小平. 一种基于全局代表点的快速最小二乘支持向量机稀疏化算法[J]. 自动化学报, 2017,43(1):132-141.
[17] 周欣然,滕召胜,易钊. 构造稀疏最小二乘支持向量机的快速剪枝算法[J]. 电机与控制学报,2009,13(4): 626-630.
[18] CAUWENBERGHS G, POGGIO T. Incremental and decremental support vector machine learning [C]// Proceedings of International Conference on Neural Information Processing Systems. Cambridge, USA: MIT Press,2000:388-394.
[19] KEERTHI S S,SHEVADE S. SMO algorithm for least-squares SVM formulations [J]. Neural Computation, 2003,15(2):487-507.
[20] 周志华. 机器学习[M]. 北京:清华大学出版社,2016.

编辑　吴云芳