聚簇的分布式 SVM 算法优化

王　瑞，向　新，肖冰松✉
（空军工程大学航空工程学院，西安，710038）

摘要　针对分层并行 SVM 算法采用完全随机划分方法生成的子样本集与原始样本集的分布情况存在偏差的问题，提出分布式 k-means 聚簇的导向随机划分方法。该方法并非将上一层的训练结果直接作为下一层的输入，而是使用 k-means 聚簇算法聚成下一层节点数 N 的不同簇，然后把每一簇样本再随机划分成 N 份，从每一簇中随机取出一份重新组合成下一层训练的 N 个子样本集，进而保证子样本集与原始样本集的分布情况具有相似性。结果表明，该方法既能有效提高学习能力，又能减少多次训练模型的抖动。

关键词　支持向量机；导向随机划分；聚簇；分布式 k-means；并行计算；模型抖动
DOI　10.3969/j.issn.1009-3516.2018.02.015
中图分类号　TP391　　**文献标志码**　A　　**文章编号**　1009-3516(2018)02-0086-07

A Distributed SVM Algorithm Optimization of Clustering

WANG Rui, XIANG Xin, XIAO Bingsong✉
(Aeronautics Engineering College, Air Force Engineering University, Xi'an 710038, China)

Abstract: Aimed at the problems that the layered parallel SVM algorithm is to generate sub-sample sets by completely adopting the random partition method, and the distribution deviation exists in between the sub-sample sets and the original sample set, this paper proposes a random-oriented partition method based on distributed k-means clustering. Not that the method is used to take a layer of the training results directly as input of the next layer, but that the k-means clustering algorithm is used to cluster into the number of the next layer node clusters. Then, the paper divides each cluster samples into N parts randomly, and takes out one from each cluster reassembled into N sub-sample sets to next layer of training to ensure the distribution of the sub-sample sets similar to original sample set. The results show that this method can not only improve learning ability effectively, but also reduce the jitter of training model.

Key words: support vector machine; random-oriented partition; clustering; distributed k-means; parallel computing; model jitter

支持向量机(Support Vector Machine,SVM)[1]是一种有监督的机器学习方法，也是目前解决分类问题常用的方法之一。SVM 使用核函数[1-3]隐式地将线性不可分的数据集从低维采样空间映射到高维特征空间，在高维特征空间使用线性分类器得到最优分类超平面，从而解决非线性数据分类问题。

收稿日期：2017-02-22
基金项目：陕西省自然科学基础研究计划(DF011000306)
作者简介：王　瑞(1990—)，女，陕西榆林人，硕士，主要从事机器学习、数据挖掘研究. E-mail:15353724115@163.com
通信作者：肖冰松(1982—)，男，湖北荆门人，讲师，主要从事武器系统仿真、传感器管理研究. E-mail:58818252@qq.com

引用格式：王瑞，向新，肖冰松. 聚簇的分布式 SVM 算法优化[J]. 空军工程大学学报(自然科学版)，2018，19(2)：86-92. WANG Rui, XIANG Xin, XIAO Bingsong. A Distributed SVM Algorithm Optimization of Clustering [J]. Journal of Air Force Engineering University (Natural Science Edition), 2018, 19(2): 86-92.

SVM 分类器的核心是求解一个二次规划(Quadratic Programming，QP)问题，即寻找最优解 $\alpha=(\alpha_1,\alpha_2,\cdots,\alpha_l)$，使得目标函数最小：

$$\min\left\{\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\boldsymbol{y}_i\boldsymbol{y}_j\alpha_i\alpha_j K(\boldsymbol{x}_i,\boldsymbol{x}_j)-\sum_{i=1}^{l}\alpha_i\right\},$$
$$\text{s.t.}\sum_{i=1}^{l}\boldsymbol{y}_i\alpha_i=0,0\leqslant\alpha_i\leqslant C \qquad (1)$$

式中：$\boldsymbol{x}_i,\boldsymbol{x}_j$ 为 d 维输入样本向量；$\boldsymbol{y}\in\{-1,+1\}$ 为类别标签；l 为训练数据集大小；$K(\boldsymbol{x}_i,\boldsymbol{x}_j)$ 为已选的满足 Mercer 条件的核函数；$\alpha=(\alpha_1,\alpha_2,\cdots,\alpha_l)$ 是求解的目标参数。

随着数据集规模的扩大，SVM 训练时间长、占用内存大的缺点限制了它在海量数据领域的应用，为了加快求解速度，研究并行 SVM 算法[4-5]尤为必要。当前有 2 种策略：①“分块”策略。将原始训练样本集分成若干活动子集，对每个活动子集迭代优化，直到达到全局最优。著名的算法有：最小序贯优先算法(Sequential Minimal Optimization，SMO)[6]、连续超松弛算法(Successive Over-Relaxation，SOR)[7]和 SVMlight[8]；②在训练过程中尽早剔除非支持向量，保留支持向量，逐层迭代优化直到达到全局最优。就目前来看，第 2 种策略虽然缩小了单节点训练样本集的规模、大大减少了运算和存储开销，但预测准确率相对较低。

Vishnu A 等人[9]提出采用消息传递接口(Message Passing Interface，MPI)[10]实现一种分布式计算的 SVM 算法，该方法在训练过程中能够自适应地剔除对定义分类器边界贡献不大的样本。另外，为了保证最终分类结果的准确性，还设计了梯度同步数据结构和启发式收缩算法。Sun 等人[11]提出基于图像处理单元(Graphics Processing Unit，GPU)的 MapReduce 框架加快 SVM 的学习速度，该方法实现 2 个层次的并行：线程级和任务级。前者由 GPU 线程实现，后者由 MapReduce 框架实现。Grafs 等人[12]提出多层迭代优化和全反馈的 Cascade SVM 算法框架。算法的基本思想是：首先，将训练样本集随机划分成一些子集(如 TD1-TD8)；然后，将这些子集分配给第 1 层的子节点，在各个子节点上使用单机 SVM 进行训练，得到子节点的支持向量(SV1-SV8)；接着，将这些子节点的支持向量两两合并形成新的训练样本集作为下一层训练的输入，重复这一过程，直到最后一层的子节点数目为 1 时，一轮训练结束，如果得到的支持向量达到全局最优或满足预设条件，迭代停止。否则，将结果反馈到第 1 层继续下一轮迭代，如此循环，直至最终的支持向量达到全局最优。Alham 等人[13-14]使用 MapReduce 框架实现 Cascade SVM 算法在图像标注领域的应用，取得了满意的效果，该算法框架称为 MRSVM。

以上方法仅仅考虑使用分布式环境加快求解问题速度，但经研究发现：在并行条件下，如果同一数据集每次随机划分得到的子数据集不同，那么训练得到的模型参数不同、泛化学习能力也不同，即多次训练预测准确率存在明显的抖动现象。尤其是 Cascade SVM 算法框架使用完全随机划分方法生成子数据集，显然没有考虑这一点。为此，文章提出 k-means 聚簇的子数据集划分方法，并给出聚簇的分布式并行 SVM 算法框架，该框架能够保证子数据集与原始数据集的分布情况尽可能相似、且每次划分的结果尽量相同，从而得到较稳定的泛化学习能力，多次实验结果也验证了这一结论。提出新方法的创新点在于：用无监督数据聚类算法有效辅助有监督的数据分类算法，从而提高分布式条件下的泛化学习能力，并有效减轻多次训练学习模型的抖动现象。

1　并行 SVM 数据的聚簇划分

聚类是一种无监督的数据分析方法。目前，常见的聚类算法[15]主要有 5 类，k-means 是典型的基于划分的聚类算法，在处理大规模数据集时具有可伸缩性、高效性(时间复杂度满足多项式平滑[16-17])等优势。

1.1　并行 k-means 聚类算法

采用 MapReduce 框架[18]实现并行 k-means 算法[19]能够显著提高运算速度且具有一定的扩展性。算法基本思路：串行算法的一次迭代过程对应 MapReduce 的一次作业提交过程，每提交一次作业应完成 2 项任务，即计算所有样本向量到聚类中心的距离以及产生新的聚类中心。

Map 函数的任务是计算每个样本点到所有聚类中心距离，根据距离最小原则重新标记其所属的新类别。Map 函数的输入为待聚类的所有样本向量和上一轮迭代(或初始聚类)的聚类中心，每一类聚类中心信息都在聚类中心描述文件中，因此每个 Map 函数都应读取聚类中心描述文件。Map 函数的输出为聚类类别 ID 和对应的样本向量集。

Reduce 函数的任务是根据 Map 函数得到的中间结果计算出新的聚类中心，供下一轮 MapReduce 作业使用。Reduce 函数输入的 key-value 对形式为<类别 ID，{该类别的样本向量集}>，计算 key 相同的样本总数，样本向量各维度值的和，进而求出各

维度值的均值,得到新的聚类中心描述文件。输出 key-value 对的形式为<类别 ID,均值向量>。

1.2 完全随机划分与聚簇划分的结果比较

假设某两分类问题的原始数据集分布见图 1(a),正负样本分别用"○"和"△"表示,H1、H2 为单节点 SVM 训练得到的样本边界,边界上的样本点即支持向量。图 1(b)使用随机划分方法将原始数据集划分成 2 个子样本集的其中一种可视化结果,图 1(c)中虚线 H11、H12 为划分 I(图 1(b)中用"□"框起的样本)训练得到的样本边界,虚线 H21、H22 为划分 II(除划分 I 以外的样本)训练得到的样本边界。从图中不难发现,灰色的三角和圆圈是被遗漏的全局支持向量。随着划分数目的增多,该遗漏全局支持向量的现象尤为明显,所以分布式并行 Cascade SVM 算法采用完全随机划分方法会遗漏大量全局支持向量,导致最终学习模型的泛化能力较差且预测精度存在明显的抖动现象(具体比较结果参见第 3 部分实验结果分析)。

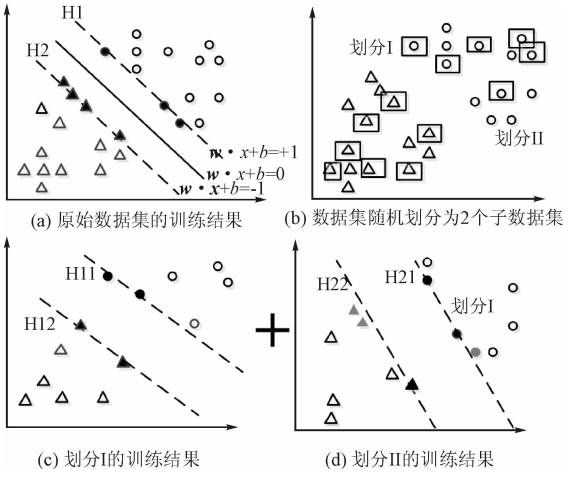


图 1 随机划分原始数据集和训练结果

Fig. 1 Partition the original data set with randomly and the training results

k-means 聚簇的导向随机划分方法,能够使划分的子数据集与原始数据集的分布情况尽可能相似,这样既可以有效避免划分子集对寻找全局支持向量的影响,也不会随着划分数目的增多,遗漏大量真正的全局支持向量。举例说明如下,图 2(a)是假设聚类簇数 $K=3$ 时的结果,图中灰色三角和圆圈为随机选取的初始聚类中心;图 2(b)是对(a)中每一簇样本使用随机划分方法得到 2 个子样本集的其中一种可能划分形式,但这里的随机不是完全随机,而是每一簇样本按正负样本总数等量随机的分配给每一个划分,其好处是避免划分中正负样本数目悬殊较大,造成每个划分训练的超平面与原始超平面相比发生大幅度的偏移;图 2(c)中 H11、H12 为划分 I 训练得到的样本边界,H21、H22 为划分 II 训练得到的样本边界。该示例也表明,导向随机划分数据不像完全随机划分数据一样会遗漏大量全局支持向量,因此对寻找全局最优解的影响较小,能够保证训练得到的模型具有较好的泛化学习能力。

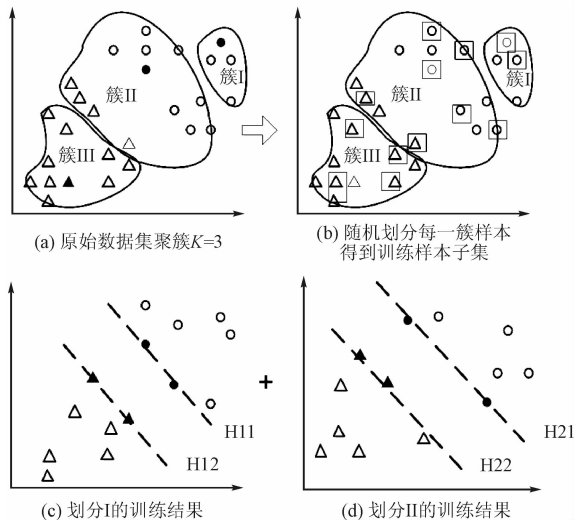


图 2 *k*-means 聚簇划分原始数据集和训练结果

Fig. 2 Divide the original data set with *k*-means and the training results

2 聚簇的并行 SVM 算法设计与实现

2.1 *k*-means 聚簇的分布式 SVM 算法框架

k-means 聚簇的分布式 SVM 算法框架见图 3,该算法框架与 Grafs 等人提出的 Cascade SVM 算法框架具有明显不同。该方法能够对原始数据集有导向性的进行划分,使每次迭代过程中在每个子节点上训练的数据集没有重叠部分且与原始数据集的分布情况相似。

图 3 为算法框架,首先使用分布式 *k*-means 聚类算法将原始训练样本集(TD)聚成 k 簇(CTD),每一簇中的样本集随机划分成 N 份,从每簇中分别取出一份重新组合成一个子样本集,最终得到 N 个子样本集(CTD/N)(以 8 个划分为例,根据集群大小划分数目可调),这些子样本集作为各子节点训练的输入样本,每个子节点使用单机 SVM(如 libsvm[20-21])训练得到支持向量 SV1~SV8,融合后的支持向量 SVs 再使用 *k*-means 聚簇划分方法划分成 $N/2$ 个子样本集(CSV1/(N/2))继续训练,之后训练的每一层节点数目均为上一层的一半,如此循环,直至最后一层节点数目为 1 时,判断输出的支持向量是否达到全局最优,即使得目标函数最小。如果不满足就将得到的支持向量反馈到原始训练样本集中,按照以上步骤继续下一轮训练,重复这一迭

代过程；如果满足，则停止训练并给出最终的学习模型。

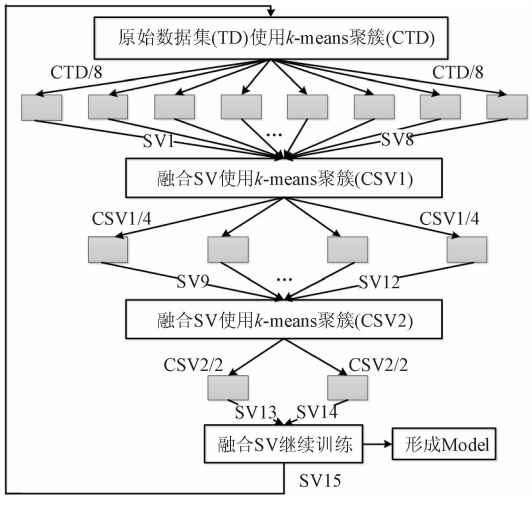


图 3　k-means 聚簇的并行 SVM 算法模型

Fig. 3　The model of parallel SVM algorithm with k-means clustering

该算法框架的核心思想主要有 3 点：①通过逐层剔除非支持向量，保留支持向量，那么最终被保留下来的支持向量就是全局最优支持向量；②该分布式并行训练模型框架不仅大大缩短了 SVM 的整体训练时间，而且由于在训练过程中采用聚簇导向随机划分方法，减少了遗漏全局支持向量的数目，从而保证分类准确率，并显著减轻学习精度的抖动现象；③若某一轮训练得到的支持向量精度达不到预期，则将这一轮训练得到的支持向量全部撒到原始数据集中，这样在下一轮训练时，各个划分子集就有机会得到上一轮其他划分子集训练产生的支持向量，进而达到模型调节的目的，经过多轮循环即可找到全局最优支持向量。

2.2　聚簇的分布式 SVM 算法 MapReduce 实现

分布式并行 SVM 算法可以使用 MapReduce 框架实现(如：MRSVM)，分布式 k-means 聚类算法也可以使用 MapReduce 框架实现。因此，使用分布式 k-means 聚类算法完成数据集的导向划分，能够与分布式并行 SVM 算法无缝融合。在训练过程中，执行聚类算法虽然也需要额外时间，但在海量数据集上使用分布式 k-means 聚类算法并不会花费太多时间，而且这种时间上的开销是值得的，它能够显著提高预测准确率，避免多次随机划分预测准确率的不稳定性。

k-means 聚簇的分布式 SVM 算法 MapReduce 伪代码分为两部分子算法：

算法 1 实现了图 3 算法框架中分布式并行 k-means 聚类算法的 map、reduce 函数设计。

map 函数的输入为训练样本集 S，输出为聚类类别 ID 和样本向量集。算法第 3 行获取聚类类别数目 K，表示将数据集 S 聚成 K 簇，第 4 行读取聚类中心描述文件为计算样本向量到各聚类中心的欧氏距离做准备，第 5 行计算样本向量到聚类中心的欧氏距离 d_{ij}，$i\in\{1,2,\cdots,l\}$，$j\in\{1,2,\cdots,K\}$，然后根据距离最近原则重新标注训练样本的所属类别。reduce 函数的输入即 map 函数的输出，输入的 key-value 对形式为：＜聚类类别 ID，{样本向量集}＞，第 10 行计算样本总数和样本向量各维度的和，第 11 行计算样本向量各维度的均值，得到均值向量。reduce 函数输出的 key-value 对形式为＜聚类类别 ID，均值向量＞。经过一次 map-reduce 过程就可以得到一组新的聚类中心，然后用新的聚类中心与旧的聚类中心进行比较，如果误差 e 满足预设阈值或达到最大迭代次数，迭代停止；否则，更新旧的聚类中心描述文件，启动下一轮 map-reduce 作业。

算法 1：k-means 的 map()、reduce()函数设计

MAP_m $m\in\{1,2,\ldots,M\}$ number of mappers M

1. INPUT：Data set S

2. OUTPUT：key-value pair＜Clustering category ID, corresponding sample vector＞

3. INITIALIZE：Class number for clustering K

4. READ：Clustering center description file for the clustering centers

5. COMPUTE：Euclidean distance d_{ij}, sample vector x_i, $i\in\{1,2,\ldots,n\}$ to the center of the cluster C_j, $j\in\{1,2,\ldots,K\}$

6. FIND：min Euclidean distance and corresponding category ID_j, $j\in\{1,2,\ldots,K\}$ enter ID ← ID_j

7. EMIT：key-value pair ＜center ID, x_i＞
$REDUCE_j$ $j\in\{1,2,\ldots,K\}$

8. INPUT：namely the output of the map function, key-value pair ＜Clustering categery ID, List {sample vector}＞

9. OUTPUT：key-value pair ＜Clustering category ID, mean vector＞

10. COMPUTE：size of List；sum ＜each component of the sample vector＞

11. Using avorage formula sum/size compute mean vector;
MS_j ← Mean ＜each component of the sample vector ＞, $j\in\{1,2,\ldots,K\}$

12. EMIT：key-value pair ＜key, MJ_j＞

算法 2 为图 3 算法框架中某一层分布式并行

SVM 的 map、reduce 函数设计。逐层迭代并行 SVM 算法在完成层级训练时,每一层的各节点单独优化一个子样本集,多个节点并行执行,然后合并各子节点的局部最优解,作为下一层迭代的输入。使用 MapReduce 实现层级优化时,每一个 Mapper 优化一个子训练样本集,多个 Mapper 并行执行,实现训练样本集的全局优化。

算法 2:一层并行 SVM 的 map()、reduce()函数设计

```
MAP_m m∈{1,2,...,M} number of mappers M
1. INPUT:path of sample subSets
2. OUTPUT:Suport Vectors (SVs)
3. INITIALIZE: node training parameter
   input_file_name ← value;model_file_name ← parameter.modelFile
4. TRAINING:svm_train.main(input_file_name,model_file_name)
5. MODEL:svm_model.model ← svm_train.model
6. SV:getSV(molde)
7. EMIT:key-value pair <0,SVi> i∈{1,2,...,sv_length}
   REDUCE
8. INPUT:namely the output of the map function;
      key-value pair <0,List{SVs}>
9. if nodeNum==1
      if global_optimum
         stopIter
      else feedback SVs first layer
10. else
      nodeNum=nodeNum/2
11. svPath ← getSVPath(vorkDix,iterId)
12. schedulerParaneter.SVPath ← svPath
13. according to svPath write SVs to HDFS
```

单层训练 map 函数的输入为经 k-means 聚簇划分后的子数据集路径,输出为优化后的局部最优支持向量,算法第 3 行获取单节点训练所需的参数,第 4 行调用单机 libsvm 算法完成局部优化,第 5 行获取训练 model,第 6 行从 model 中得到局部最优支持向量并输出,注意这里输出的 key 设为 0,因为 key 相同的记录被送到同一个 Reduce 端,这样就可以合并所有 Mapper 的输出结果。reduce 函数首先判断训练的节点数目是否为 1,如果为 1 进一步判断是否满足迭代停止条件;若不满足,将 SVs 反馈到第 1 层继续下一轮训练;如果节点数目不为 1,将节点数目减半,并获取支持向量输出路径,将支持向量写到 Hadoop 分布式文件系统(Hadoop Distributed File System,HDFS)[22] 中,修改层级调度参数 SVPath,为启动 k-means 聚类作业做准备。

3 实验数据结果分析

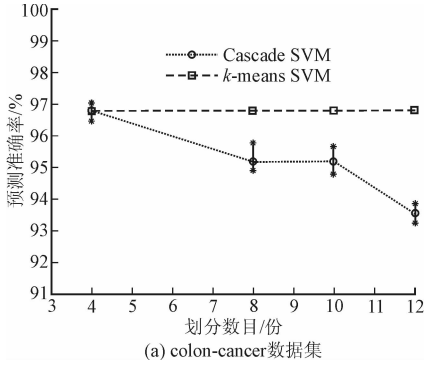
实验环境使用 Hadoop-2.6.0[23] 版本,数据集使用 libsvm 网站提供的数据集(见表 1),训练/预测数据集的大小按 8/2 比例划分。在搭建好的 Hadoop 分布式平台运行 k-means 聚簇的并行 SVM 算法,得到实验结果。分布式运行程序[24] 可下载。

表 1 实验数据集

Tab. 1 Data sets of experiment

数据集名称	特征数	样本数	libsvm 预测准确率/%
colon-cancer	2 000	62	96.77
rcv1	47 236	697 641	96.60
w8a	300	64 700	97.44
covtype.binary	54	581 012	96.10
splice	60	3 175	90.20

目前,分布式并行 SVM 预测准确率比较高的算法框架是 Graf 等人提出的 Cascade SVM,因此,实验结果与 Cascade SVM 进行比较。实验从 3 个角度进行分析:①对同一数据集在不同划分数目下,预测准确率的稳定性比较;②不同 SVM 算法训练时间比较;③同一数据集,在相同划分数目下,多次训练的预测准确率稳定性比较。图 4 是数据集 colon-cancer,rcv1,w8a 在不同划分数目下,预测准确率的比较结果。图中虚线上的"○"和"□"是同一划分数目下 5 次训练预测准确率的均值,误差棒的上下界是预测准确率的最大值和最小值。从图中不难发现,Cascade SVM 算法使用完全随机划分方法生成子样本集,且随着划分数目的增多,预测准确率呈下降趋势;但 k-means 聚簇的分布式并行 SVM 由于对训练数据集进行导向性的划分,其预测准确率并不会随着划分数目的增多而明显下降,且具有一定的稳定性。



(a) colon-cancer数据集

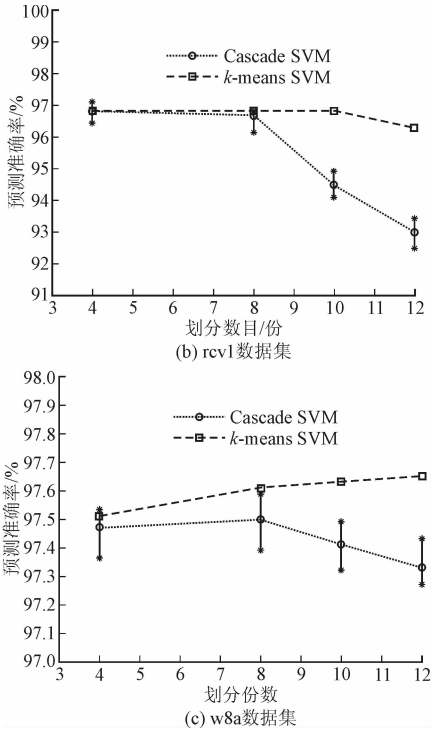
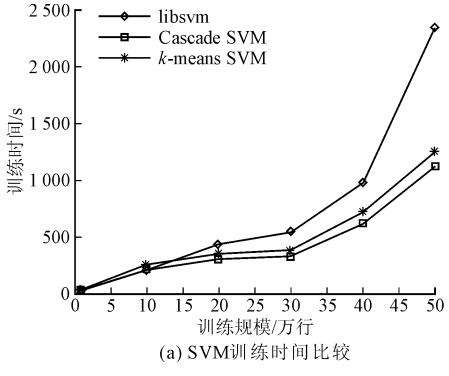


图 4　不同划分数目预测准确率结果比较

Fig. 4　Comparison prediction accuracy with different partition number

图 5(a)是 libsvm、Cascade SVM 和分布式 *k*-means SVM 训练时间比较结果，测试数据集是 covtype. binary。图中横坐标是数据集规模(1～50 万行，数据集以每行一个样本书写)，纵坐标是训练时间。从图中可以看出：①单机 libsvm 随着数据集规模的扩大，训练时间迅速增长，分布式并行 SVM 训练时间增速缓慢；②*k*-means SVM 在训练过程中加入聚类操作，所以训练时间比 Cascade SVM 稍长，但 *k*-means 也采用分布式并行执行，所以整体训练时间并没有显著增加。图 5(b)是 Cascade SVM 与 *k*-means SVM 算法对 splice 数据集做 10 次测试预测准确率稳定性的比较结果，划分数目分别取 8 和 12。



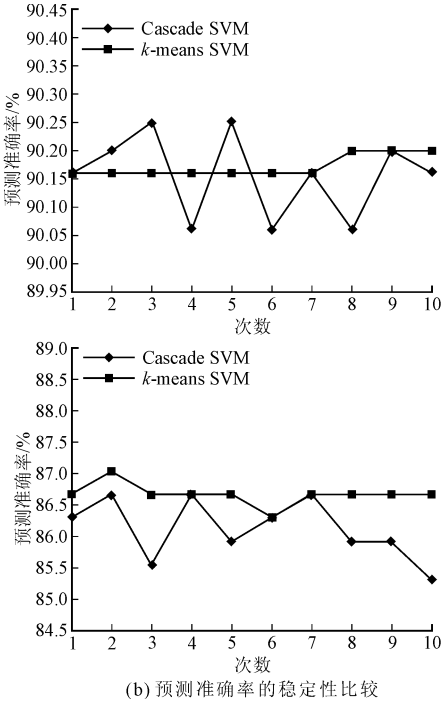


图 5　实验数据结果比较

Fig. 5　Comparison the results of experiment

4　结语

设计和实现分布式并行 SVM 算法必须考虑 2 个关键性问题：数据集如何有效划分和如何进行迭代计算。针对 Cascade SVM 使用完全随机划分生成子数据集的方法在寻找全局最优解时常常出现遗漏全局支持向量的现象，笔者尝试采用 *k*-means 聚簇的导向随机划分方法，该方法能够保证子样本集与原始样本集的分布情况具有相似性，从而提高整体预测准确率，并且避免多次随机划分预测准确率的不稳定性。与对比算法相比，时间开销仅多了分布式 *k*-means 聚簇处理过程，每一层聚簇的时间复杂度为 $O(1/\sigma)$，σ 是聚簇样本的标准方差。为存储 k 个聚类中心，空间复杂度为 $O(k)$。多次实验结果也验证新型算法框架在分布式大数据领域的学习能力具有较好的鲁棒性和稳定性。本文实验数据测试以两分类为主，但该解决问题的思路具有普适性。因此，下一步重点研究多分类问题并应用到实践中。

参考文献(References)：

[1]　VAPNIK V N. The Nature of Statistical Learning Theory [M]. 2nd ed. New York: Springer Press, 1999: 89-155.

[2]　HUANG H Y, LIN C J. Linear and Kernel Classification: When to Use Which? [J]. SIAM International Conference on Data Mining, 2016, 16(3): 302

-311.

[3] 王晓，刘小芳. 基于 NSVM 的核空间训练数据减少方法 [J]. 电子科技大学学报，2013，42(4)：592-597.
WANG X，LIU X F. Nonlinear Support Vector Machine for Training Data Reduction in Kernel Space [J]. Journal of University of Electronic Science and Technology of China，2013，42(4)：592-597. (in Chinese)

[4] ZHAO J，ZHU L，YONG Y. Parallelized Incremental Support Vector Machines Based on MapReduce and Bagging Technique [C] // International Conference on Information Science and Technology. Wuhan，China：IEEE Press，2012：297-301.

[5] LAI W C，HUANG P H，LEE Y J，et al. A Distributed Ensemble Scheme for Nonlinear Support Vector Machine [C] // Intelligent Sensors，Sensor Networks and Information Processing (ISSNIP). Singapore：IEEE，2015：1-6.

[6] CHANG P F，BI Z，FENG Y Y. Parallel SMO Algorithm Implementation Based on Open MP [C] // International Conference on System Science and Engineering. Shanghai，China：IEEE，2014：236-240.

[7] PANG R B，XU J L，ZHANG Y Q，et al. Parallel Solving Method of SOR Based on the Numerical Marine Forecasting Model [C] // IEEE/ACM International Symposium on Cluster，Cloud and Grid Computing. Shenzhen，China：IEEE，2015：733-736.

[8] JOACHIMS T. Making Large-Scale SVM Learning Practical [J]. Technical Reports，1998，8(3)：499-526.

[9] VISHNU A，NARASIMHAN J，HOLDER L，et al. Fast and Accurate Support Vector Machines on Large Scale Systems [C] // IEEE International Conference on Cluster Computing. Chicago，USA：IEEE Press，2015：110-119.

[10] GROPP W，LUSK E，DOSS N，et al. A High-Performance Portable Implementation of the MPI Message Passing Interface Standard [J]. Parallel Computing，1996，22(6)：789-828.

[11] SUN T Y，WANG H L，SHEN Y，et al. Accelerating Support Vector Machine Learning with GPU-based MapReduce [C] // IEEE International Conference on Systems，Man，and Cybernetics. Kowloon：IEEE Press，2015：876-881.

[12] GRAF H P，COSATTO E，BOTTOU L，et al. Parallel Support Vector Machines：The Cascade SVM [C] // Advances in Neural Information Processing Systems (NIPS). USA：MIT Press，2005：521-528.

[13] ALHAM N K，LI M Z，HAMMOUD S，et al. A Distributed SVM for Image Annotation [C] // Fuzzy Systems and Knowledge Discovery (FSKD). Yan-Tai，China：IEEE，2010：2983-2987.

[14] ALHAM N K，LI M Z，LIU Y，et al. A Distributed SVM for Scalable Image Annotation [C] // Fuzzy Systems and Knowledge Discovery (FSKD). Shanghai，China：IEEE Press，2011：2655-2658.

[15] FAHAD A，ALSHATRI N，TARI Z，et al. A Survey of Clustering Algorithms for Big Data：Taxonomy and Empirical Analysis [J]. IEEE Transactions on Emerging Topics in Computing，2014，2(3)：267-279.

[16] ARTHUR D，MANTHEY B，HEIKO R. *k*-means Has Polynomial Smoothed Complexity [J]. Foundations of Computer Science Annual Symposium on，2009，157(1)：405-414.

[17] SIRITEERKUL T，BOONJING V. Support Vector Machine Accuracy Improvement with *k*-means Clustering [C] // Computer Science and Engineering Conference (ICSEC). IEEE，2013：218-221.

[18] LÄMMEL R. Google's MapReduce Programming Model-Revisited [J]. Science of Computer Programming，2008，70(1)：1-30.

[19] SHETTAR R，PUROHIT B V. A Map Reduce Framework to Implement Enhanced *k*-means Algorithm [C] // International Conference on Applied and Theoretical Computing and Communication Technology. Davangere，India：IEEE，2015：361-363.

[20] CHANG C C，LIN C J. LIBSVM：A Library for Support Vector Machines [J]. ACM Transactions on Intelligent Systems & Technology，2011，2(3)：389-396.

[21] HSU C W，CHANG C C，LIN C J. A Practical Guide to Support Vector Classification [CP/OL]. (2016-05-19)[2017-02-17]. http：// www. csie. ntu. edu. tw/～cjlin.

[22] SHVACHKO K，KUANG H，RADIA S，et al. The Hadoop Distributed File System [C] // Mass Storage Systems and Technologies (MSST). USA：IEEE，2010：1-10.

[23] Apache Hadoop. Welcome to Apache Hadoop! [EB/OL]. (2014-11-18)[2016-10-15]. http：// hadoop. apache. org/releases. Html # News.

[24] WANG R. Distributed SVM Source Code. [EB/OL]. [2017-02-17]. http：// pan. baidu. com/s/1boSELh9.

(编辑：徐楠楠)