并行 SVM 算法在 Flink 平台的应用研究

白玉辛,刘晓燕

(昆明理工大学 信息工程与自动化学院,昆明 650500)
E-mail:237557359@qq.com

摘 要: 在大数据时代背景下,数据规模成指数级增长,传统支持向量机(SVM)已无法适应大数据环境,所以需要将传统支持向量机算法改进使其可以应用于大数据计算框架.针对计算过程中存在占用内存大、寻优速度慢等问题,提出一种基于 Flink 平台的并行支持向量机算法.该方法首先基于层叠支持向量机(Cascade SVM)的合并策略以及训练结构,通过 Flink 分布式计算框架实现;其次,通过优化并行操作算子的性能引入分布式广播变量,优化算法,有效解决单机 SVM 算法训练效率低的问题.实验结果表明,结合 Flink 框架实现 SVM 算法并行化,能有效的减少了训练时间,提高模型的训练效率.

关 键 词: 并行计算;支持向量机;大规模数据集处理;Flink
中图分类号: TP301 **文献标识码:** A **文 章 编 号:** 1000-1220(2021)05-1003-05

Application Research of Parallel SVM Algorithm on Flink Platform

BAI Yu-xin, LIU Xiao-yan
(College of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, China)

Abstract: In the context of the era of big data, the scale of data has grown exponentially, and traditional support vector machines (SVM) have been unable to adapt to the environment of big data, so the traditional support vector machine algorithm needs to be improved so that it can be applied to the big data computing framework. Aiming at the problems of large memory occupation and slow optimization speed in the calculation process, a parallel support vector machine algorithm based on Flink platform is proposed. This method is based on the cascading support vector machine (Cascade SVM) merge strategy and training structure, and is implemented through the Flink distributed computing framework. Secondly, the distributed broadcast variables are introduced by optimizing the performance of the parallel operation operator, and the algorithm is optimized to effectively solve the problem of low training efficiency of the stand-alone SVM algorithm. Experimental results show that the parallelization of SVM algorithm combined with Flink framework can effectively reduce the training time and improve the training efficiency of the model.

Key words: parallel computing; support vector machine; large-scale dataset processing; Flink

1 引 言

大数据时代背景下,信用卡交易、传感器测量、机器日志、网站或移动应用程序上的用户交互等各种类型的数据都是作为事件流产生的,传统数据挖掘算法处理这些大规模数据集的能力出现瓶颈,因此需要将传统数据挖掘算法和现有大数据框架进行结合,改变传统数据挖掘算法无法高效处理大规模数据集的现状.支持向量机[1](Support Vector Machine, SVM)基于统计学习理论,在模式识别、文本分类和图像识别等众多领域表现出优异实践性能的机器学习算法.SVM 相比较其他常用的数据挖掘算法而言,在算法训练过程中很少会出现过度拟合、属性特征过多造成的维数灾难对算法性能影响微乎其微、对核函数运用巧妙,可以让算法处理数据集线性不可分的情况.但是,当传统 SVM 算法处理大规模数据集时,会出现训练速度慢,内存溢出,运行崩溃等性能低下问题[2-4].

针对传统单机 SVM 算法面对大规模数据集处理效率低下等问题,最近几年数据挖掘领域的专家主要使用两种方法改进传统支持向量机算法:1)采用“分而治之”的思想处理大规模数据集,该思想是将一个完整的数据集切分成若干训练子集,在每一个训练子集上训练局部支持向量并剔除大量非样本边界样本点,使其并行训练,达到算法并行化训练的目的;2)利用 GPU 或 FPGA 强大的矩阵运算能力来提高收敛速度.如 Li 等人[5]基于 GPU 设备设计实现了一个并行 SVM 智能分类系统,提高 SVM 预测及训练阶段算法执行的效率.胡福平等人[6]使用 FPGA 实现 SVM 并行计算结构,借助 Verilog HDL 程序设计语言完成了各模块的结构设计,该结构实现的 SVM 的分类性能要略优于 Libsvm.丁宣宣等人[7]利用 MapReduce 和 Bagging 的并行组合支持向量机训练算法,采用 Bagging 集成算法的思想,结合随机次梯度 SVM 算法对剩余的支持向量训练,以提高算法的分类精度.刘泽燊[8]等人使用基于内存计算的分布式处理框架—Spark,实现基于 Spark 的并行 SVM 算法,克服了并行 SVM 算法在 MapReduce模型中迭代计算的不足.李坤等人[9]和何经纬等

收稿日期:2020-04-21 收修改稿日期:2020-05-20 基金项目:国家自然科学基金项目(61462055)资助. 作者简介:白玉辛,男,1996 年生,硕士研究生,研究方向为大数据技术分析;刘晓燕(通讯作者),女,1964 年生,博士,副教授,研究方向为软件工程模型驱动.

人[10]则研究如何在 Spark 框架提高 SVM 参数寻优的时间. 通过这些研究一定程度上可以解决传统支持向量机训练效率低下等问题,但随着电子商务的高速发展,互联网电商平台的数据呈指数级别的增长,鉴于实时机器学习的思想提出,因此,需要使用实时处理框架来对传统 SVM 算法做出改进,使其能够适应框架,最终达到线下训练,线上实时预测的目的.

2008 年,柏林理工大学研发出一个大数据处理平台,此为 Flink[11]的前身,随后在 2014 年成为了 ASF(Apache Software Foundation)的顶级项目之一. Apache Flink 是一个框架和分布式处理引擎,用于对无边界和有边界数据流上进行有状态的计算,并且它可以用于 Google 数据流模型[12],同时 Flink 能在所有常见集群环境中运行,以内存速度和任意规模进行计算. 目前互联网领域的实时搜索、数据分析和机器学习等任务都可以在 Flink 平台上运行.

本文对 Cascade SVM 算法和 Flink 计算框架进行了深入研究分析,采用数据切分的思想,实现了基于 Flink 平台并行 SVM 算法,有助于向线下训练,线上实时预测模型准确度. 具体工作主要有:1)深入研究 CascadeSVM 算法的训练结构及合并策略,设计基于 Flink 计算框架的并行 SVM 算法;2)对部分并行操作算子进行优化,通过广播变量的方式将训练过程中的局部支持向量发送给下一级算子,优化了原本中间结果需要写到本地磁盘的过程. 实验对比表明,该算法能够加快模型训练的速度,保证精度损失较小的前提下,提高了算法执行效率.

2 Apache Flink 技术

Apache Flink[1] 一款能够同时支持高吞吐、低延迟、高性能的分布式处理框架,在 2010 年-2014 年间,由柏林工业大学、柏林洪堡大学和哈索普拉特纳研究所联合发起名为“Stratosphere:Information Management on the Cloud”研究项目,并于 2014 年 4 月,贡献给 Apache 基金会孵化器项目,期间改名 Flink. 自 2015 年 9 月发布第一个稳定版本到现在为止,更多的社区开发成员逐步加入,现在 Flink 在全球范围内拥有 350 多位开发人员,在国内比较出名的互联网公司如阿里巴巴,滴滴等都在大规模使用 Flink 作为企业的分布式大数据处理引擎. 在不久的将来,Flink 也将成为企业内部主流的数据处理框架,最终成为下一代大数据处理的标准.

Apache Flink 是一个集众多具有竞争力的特性于一身的第 3 代流处理引擎. 它支持精确的流处理,能同时满足各种规模下对高吞吐和低延迟的要求,尤其是以下功能使其在同类系统中脱颖而出:1)同时支持事件时间和处理时间语义. 事件时间语义能够针对无序事件提供一致、精确的结果. 处理时间语义能够用在具有极低延迟需求的应用中;2)提供精确一次(exactly-once)的状态一致性保障;3)在每秒处理数百万条事件的同时保持毫秒级延迟,同时基于 Flink 的应用可以扩展到数千核心之上;4)层次化的 API 在表达能力和易用性方面各有权衡;5)支持高可用性配置(无单点失效),如 Apache Kafka、Apache Cassandra、JDBC、Elasticsearch 以及分布式文件系统(HDFS 和 S3)等.

[1] http://flink.apache.org

现实世界中,所有的数据都是以流式的形态产生的,根据现实的数据产生方式和数据产生是否含有边界(具有起始点和终止点)角度,将数据分为两种类型的数据集,一种是有界数据集,另一种是无界数据集. Flink 正是用于这两种数据集上进行有状态计算,其核心模块是一个数据流执行引擎,主要是通过 Java 代码实现的. 其中有界数据集具有时间边界,在处理过程中数据一定会在某个时间范围内起始或结束,对有界数据集的数据处理方式被称为批处理(Batch Process);对于无界数据集,数据从一开始生成就持续不断地产生新的数据,因此数据是没有边界的,对无界数据集的数据处理方式被称为流处理(Streaming Process). Flink 最近提出了本地闭环迭代操作符[13]和基于成本的自动优化器,它能够重新排序操作符,并更好地支持流,所以 Flink 在处理迭代和实时数据处理问题时,能够在保证稳定性和状态一致性的同时提高系统的运行速度,支持数据高吞吐和低延迟等优点.

3 相关概念和原理介绍

3.1 支持向量机

支持向量机 SVM 算法是建立在统计学习理论和结构风险最小化原理基础上,其基本模型是在特征空间上寻找具有最大边缘距离(Maximum Marginal)分类超平面的线性分类器,有效克服了传统机器学习算法中出现维数灾难、过度拟合和局部最小值等缺点. 算法描述如下:给定线性可分训练集 $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$,其中 $x_i\in X=R^n$,$y_i\in Y=\{+1,-1\}$,$i=1,2,\cdots,M$;对于给定的训练集,构造并求解约束最优化问题,然后引入拉格朗日乘子,对其参数求偏导,得到与原问题对应的对偶问题,如式(1)所示:

$$\begin{aligned}&\min_{\alpha}\quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(x_i\cdot x_j)-\sum_{i=1}^{N}\alpha_i\\&s.t\quad \sum_{i=1}^{N}\alpha_i y_i=0\\&\alpha_i\geqslant 0,i=1,2,\cdots,N\end{aligned}\tag{1}$$

引入对偶问题的目的是相比较原问题更容易求解;其次在数据集线性不可分的情况下可以使用核函数学习非线性支持向量机做铺垫. 得出最优解 $\alpha^*=(\alpha_1^*,\alpha_2^*,\cdots,\alpha_N^*)^T$,然后计算 $\omega^*=\sum_{i=1}^{N}a_i^*y_ix_i$,并选择 α^* 的一个正分量 $a_j^*>0$,计算 $b^*=y_i-\sum_{i=1}^{N}a_i^*y_i(x_i\cdot x_j)$,可以求出分离超平面为 $\omega^*\cdot x+b^*=0$,分类决策函数:$f(x)=sign(\omega^*+b^*)$.

对于训练样本集线性不可分的情况,设置惩罚函数 C,使分类器模型容忍部分噪声点的同时避免了过拟合学习的情况,求解线性分类问题,线性分类支持向量机是一种非常有效的方法,但如果训练数据线性不可分时,对偶问题和决策函数中有样本点的内积计算,通过引入核函数将样本从低维空间映射到高维空间,等价于隐式地在高维空间中学习非线性支持向量机,这样的方法称为核技巧,优点是实质的分类效果表现在高维空间上,使原问题变成线性可分,通过使用核技巧(knernel trick)及软间隔最大化,学习非线性支持向量机. 核函数包括多项式核、高斯核和线性核,一般的核函数表达式为:

$$K(x,z)=<\sigma(x)\cdot\sigma(z)>\tag{2}$$

3.2 层叠分组并行 SVM 算法

Deng Kun 等人和 Grafs 等人提出了分组训练和层叠训练算法,目的是为了解决单机 SVM 处理大规模数据集出现训练速度慢,内存溢出,运行崩溃等性能低下问题,但文献[14]指出分组训练 SVM 模型在分组数目分配过多的情况下,会导致训练子集中的局部支持分布与原样本集的分布差别过大,导致准确率下降;层叠训练 SVM 模型在第一层训练过程中剔除大部分非支持向量,随后层级训练过程中消耗大量的时间,且过滤的非支持向量较少,效率低,所以将这两种算法结合使用训练模型. 算法基本思路是采用 3 层固定串联训练结构,层与层之间采用随机合并的方式,首先将训练样本集随机划分成大小相等的样本子集(TD1-TD8),在样本子集上并行训练 SVM,将得到的支持向量合并再随机划分成大小相等新的样本子集(TD9-TD12),同样进行并行训练,最后一次迭代将这 4 个训练子集合并进行全局训练得到全局支持向量机模型. 算法的详细流程图如图 1 所示,TD_i 表示第 i 个样本子集,SVi 表示在 TD_i 样本子集上训练得到 SVM 模型.

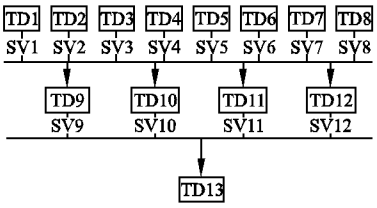

图 1　层叠分组并行 SVM 训练算法流程

Fig. 1　Cascaded parallel SVM training algorithm flow

第 1 层:根据大数据“分而治之”的思想,将样本随机划分成大小相等的样本子集(TD1-TD8),在各个训练子集上并行训练 SVM 模型,本层将滤掉大量非支持向量,同时剔除非边界样本.

第 2 层:将上层得到的局部支持向量先合并,然后再随机划分成大小相等新的训练子集(TD9-TD12),并在各自样本子集上并行训练 SVM 模型.

第 3 层:将第 2 层得到 4 个训练样本子集合并,得到一个训练集(TD13),然后进行全局支持向量的训练,得到全局支持向量机模型,同时判断是否满足迭代停止条件. 如果不满足,将结果送回反馈,重复以上步骤,直到满足收敛条件.

4 基于 Flink 的并行 SVM 算法的设计与实现

4.1 基于 Flink 框架并行 SVM 算法实现

深入研究层叠分组 SVM 算法训练策略,结合 Flink 计算框架的优势,设计基于 Flink 的并行 SVM 算法. 算法的训练流程如图 2 所示,在训练模型之前需将训练集样本及测试集样本上传到 Flink 集群上,Flink 设置批处理执行环境 ExecutionEnvironment,通过 readTextFile( ) 方法读取文件,Flink 中的 JobManager 在集群中创建任务同时负责集群任务的调度以及资源管理. 算法在 Flink 集群上进行分层的并行 SVM 训练将局部支持向量进行合并,直至最后训练完成,得到全局支持向量模型输出到本地文件系统.

算法在 Flink 分布式框架上的并行实现首先搭建 Flink 集群,设置集群模式为 standalone,上传训练数据集到集群,通过 readTextFile( ) 方法读取训练集并转换成 DataSet[String] 数据模型,同时设置任务并行度 setParalleism(int),代表将大规模数据集随机切分成大小适中的独立数据块并分区. 然后使用mapPartition( ) 方法并行SVM模型训练,过滤掉样本子

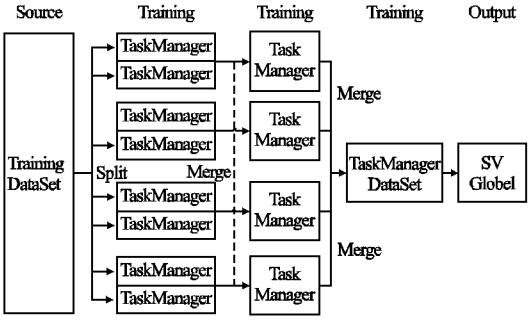

图 2　基于 Flink 的并行 SVM 的训练流程

Fig. 2　Flink-based parallel SVM training process

集中大量非边界样本同时保留局部支持向量 SV_i,大幅度减少后续训练样本数据的大小,加快了算法的训练速度,同时第一层训练完以后将局部支持向量合并再随机划分为大小相同的数据块 TD_i,同时以广播变量的方式发送给下次迭代,作为下一次迭代的输入,最后得到一个数据集 GTDi,在 GTDi 上训练 SVM,得到全局支持向量模型 GModel. 基于 Flink 的并行 SVM 算法的描述如算法 1 所示.

算法 1. 基于 Flink 的并行 SVM 算法

输入: Training Datasets, FilePath

输出: SVi, GModel

```
1. val env = ExecutionEnvironment. getExecutionEnvironment
2. val env. setParalleism(8)
3. val DataSet[String]←env. readTextFile(FilePath)
4. Do i = 1,…,L
5. val SVi←DataSet. mapPartition(Train_SVM). setBroadcast( )
6. val TDi←SVi. rebalence( )
7. val SVi←TDi. mapPartition(Train_SVM). setBroadcast( )
8. val GTDi←SVi. rebalence( )
9. val GModel←GTDi. mapPartition(Train_SVM)
10. Until 满足收敛条件 Return GModel
```

4.2 算法调优

Flink 有 3 种数据分区模式:Rebalance, Hash-Partition 和 Range-Partition 并行操作算子. 其中 Rebalance 根据轮询调度算法,将数据均匀地分发给下一级节点,平衡了有些节点局部支持向量较少,有些节点局部支持向量过多的情况,有效利用集群资源,达到均衡负载的目的. 其次 Flink 中广播变量(Broadcast Variable)是算子的多个并行实例间共享数据的一类方法. 广播变量以集合的方式定义在某个需要共享的算子上,算子的每一个实例可以通过集合访问共享变量,将局部支持向量以广播变量的方式发送给下一级算子,减少两个算子的网络通信消耗,提高了算法训练速度,但广播变量存储在 TaskManager 的内存里,其大小不易过大,所以需要通过第一层剔除大量非边界样本点后得到部分支持向量,才能将其以广播变量的方式发送给下一级算子.

5 实验过程与结果分析

5.1 实验环境与数据集

实验使用 LibSVM3.24 工具,利用 5 台 PC 机构建 Flink 集群,包括 1 台 Master 节点和 5 个 Slave 节点(其中一台机器即是 Master 节点又是 Slave 节点),集群的任务调度模式为 standalone. 硬件配置:CPU 双核,内存 4GB,硬盘 20G. 软件配置:集群搭建使用 Flink-1.7.2-bin-scala_2.11,Scala 选用 Scala2.11,Java 选用 JDK1.8(Linux 版本),操作系统选择 CentOS7. 训练集采用 a9a,covtype. Binary 两个标准数据集[2]. a9a 数据集是用来预测一个成年人的年收入,该数据集是一个二分类问题,类标号分别是 -1 和 +1,该数据集拥有 123 个属性、32562 个训练样本和 16281 个测试样例;covtype. binary 数据集原始为森林覆盖率数据,有 581912 个样本,每个样本具有 54 维特征.

5.2 实验结果及性能评估

5.2.1 使用广播变量性能分析

首先使用 covtype. binary 数据集进行实验,将数据集切分成 8 个大小相同的数据块,然后再相同条件下使用广播变量和不使用广播变量的情况下,训练不同规模的样本集,相关结果展示在图 3 中,可以看出,随着样本数量的增加,两者的训练时间都在增大,但是当样本数量超过 20 万时,使用广播变量将局部支持向量转发给下一级算子的训练时间明显少于不使用广播变量,这是因为将局部支持向量保存为广播变量是存储在内存中,不用写到本地磁盘,下一级算子直接从内存中读取上一层局部支持向量继续计算,减少了中间结果写到本地磁盘和从本地磁盘读取中间结果的时间,算法整体效率更高,极大地减少了合并的时间. 因此最后采用广播变量的方式将局部支持向量发送给下一级算子.

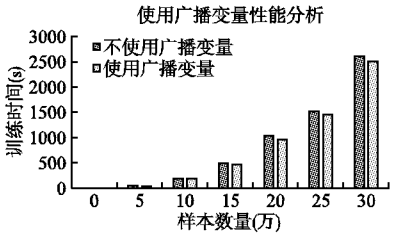

图 3 使用广播变量

Fig. 3 Use broadcast variables

5.2.2 训练结果与计算节点数量的关系

为了评估算法在不同计算节点数量下训练时间和准确率的变化情况下,选用了 a9a 数据集进行实验,不同规模的集群(集群中 Slave 节点的数量)在训练时间取 3 次训练的平均时间,准确率是通过训练出的 SVM 模型对相应测试集进行预测. 最终结果如图 4 所示.

从图 4 可以看出,随着计算节点数目的增加,算法的训练时间逐步下降,这是由于多个 taskManager 同时对算法并行训练,虽然有效的降低模型训练时间,但是同时增加了分区数据传输的网络开销. 开始时,可以看见算法训练时间急剧下降,这是单机与并行训练最大的不同,当计算节点达到一定数量后,集群中每个 taskManager 计算时间趋于平稳,所以整个训练时间下降会变得平缓;算法的测试准确率随着计算节点数量的增加而降低,这是因为将数据集切分过多的数据块,每个小数据块与原始数据集的支持向量分布差别过大,所以导致训练出来的全局支持向量与单机 SVM 训练的结果不一致. 平衡集群的计算能力与数据集切分大小的关系是至关重要的,我们不能一味追求计算速度的提升而忽略算法的准确率.

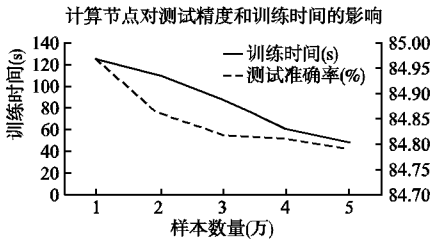

图 4 计算节点对测试精度和训练时间的影响

Fig. 4 Influence of computing nodes on test accuracy and training time

5.2.3 算法运行时间对比

算法运行时间对比实验采用 covtype. binary 数据集,因为此数据集数据量大,可以有效看出单机 SVM 算法、Cascade SVM算法和FL-SVM算法训练的模型所需时间的差异,图5展示了 3 种算法在不同规模的数据集上训练所耗费的时间. 当数据集较小时,3 种算法训练时间差别不是很大. 随着样本规模增大,并行算法在集群上运行的优势也就明显,说明在大规模数据集下训练支持向量的任务初始化和网络开销占整个作业运行时间的比重小,同时并行 SVM 算法会将样本随机均匀划分,同时分发给集群中的计算节点上,从而节省大量的训练时间. Flink 的任务调度方式以及作业"流水线"的执行等等,使得 FL-SVM 算法在效率上要明显优于 Cascade SVM 算法,能够有效在大规模数据集上训练中减少训练时间并提高分类效率.

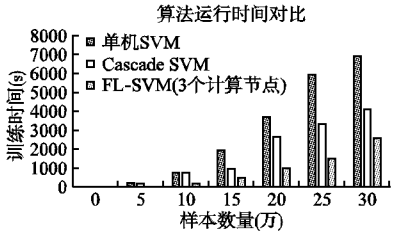

图 5 算法运行时间对比

Fig. 5 Algorithm running time comparison

5.2.4 算法准确率对比

为了验证改进后的 FL-SVM 算法模型的准确率,本次实验采用 a9a 和 covtype. binary 数据集进行验证实验,使用单机 SVM 算法、Cascade SVM 算法和 Fl-SVM 算法对这两个数据集进行训练,并得到支持向量机模型,然后使用测试数据集对模型进行评测,得到测试准确率如图 6 所示,从中可以看出

[2] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets

FL-SVM 算法在 2 个数据集的准确明显高于 Cascade SVM 算法但略低于单机SVM算法,且误差范围不超过0.01.总体而

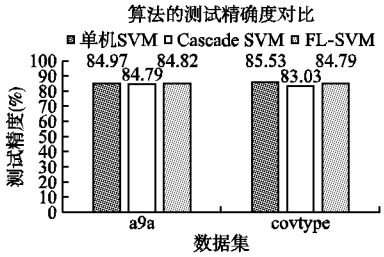

图 6　算法的测试精确度对比

Fig. 6　Algorithm test accuracy comparison

言,FL-SVM 算法的精度损失在我们能忍受的范围内,同时其能有效减少训练时间,避免内存溢出的等情况,所以采用 FL-SVM 算法处理大规模数据集是可行的.

6　结　语

支持向量机算法从问世至今已有 20 年的历史,在大数据时代背景下的今天,传统的支持向量机算法已无法高效处理大规模数据集.本文对分组层叠 SVM 算法的并行训练策略深入的研究,结合 Flink 计算框架,实现了一种基于 Flink 的并行 SVM 训练算法,有效克服单机 SVM 算法在处理大规模数据集出现内存溢出,训练效率低,训练时间慢等问题.实验结果表明 FL-SVM 算法在保证一定的准确率的情况下,能大幅度提高训练速度,是 SVM 算法解决大规模数据集的一种有效的解决方案.

以后的工作中重点着手于 Flink 平台的性能优化,以及实时机器学习等方面,将支持向量机算法结合 Flink 实时计算的低延迟、高吞吐、高性能的优势,达到线下训练模型,线上实时预测模型.同时,也会深入研究 Flink 平台的参数调优,以达到更好的训练效果目的.

References:

[1] Vapnik V N. The nature of statistical learning theory[M]. Springer New York,1995:988-999.

[2] Rudra P. Verification and validation of parallel support vector machine algorithm based on mapreduce program model on hadoop cluster[C]//International Conference on Advanced Computing & Communication Systems,IEEE,2013.

[3] Nie W,Fan B,Kong X,et al. Optimization of multi kernel parallel support vector machine based on Hadoop[C]//2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference(IMCEC),IEEE,2016.

[4] Zhu Ying-ying,Yin Chuan-huan,Mu Shao-min. An improved local support vector machine algorithm [J]. Computer Engineering and Science,2013,35(2):91-95.

[5] Li Q,Salman R,Kecman V. An intelligent system for accelerating parallel SVM classification problems on large datasets using GPU [C]//10th International Conference on Intelligent Systems Design and Applications(ISDA),2010.

[6] Hu Fu-ping,Xu Mei-hua,Shen Hua-ming. SVM parallel computing structure research and FPGA implementation[J]. Microelectronics and Computer,2018,35(6):79-83.

[7] Ding Xuan-xuan, Huang Wei, Guo Yuan-bo, et al. Parallel combined support vector machine based on MapReduce and Bagging [J]. Journal of Information Engineering University,2018,19(2): 196-202+208.

[8] Liu Ze-shen,Pan Zhi-song. Spark-based parallel SVM algorithm research[J]. Computer Science,2016,43(5):238-242.

[9] Li Kun,Liu Peng,Lv Ya-jie,et al. Parallel optimization algorithm of libsvm parameters based on spark[J]. Journal of Nanjing University(Natural Science),2016,52(2):343-352.

[10] He Jing-wei,Liu Li-zhi,Peng Bei,et al. Study on parallel Spark SVM parameter optimization algorithm based on Spark[J]. Journal of Wuhan University of Technology,2019,41(3):283-289.

[11] Alexandrov A,Bergmann R,Ewen S,et al. The stratosphere platform for big data analytics[J]. Vldb Journal,2014,23(6):939-964.

[12] Akidau T,Schmidt E,Whittle S,et al. The dataflow model:a practical approach to balancing correctness,latency,and cost in massive-scale,unbounded, out-of-order data processing[J]. Proceedings of the Vldb Endowment,2015,8(12):1792-1803.

[13] Zhang Y,Gao Q,Gao L,et al. iMapReduce:a distributed computing framework for iterative computation[J]. Journal of Grid Computing,2012,10(1):47-68.

[14] Guo Xin-xin. Optimization of SVM algorithm based on distributed computing[D]. Xi'an:Xidian University,2014.

附中文参考文献:

[4] 朱莹莹,尹传环,牟少敏.一种改进的局部支持向量机算法[J].计算机工程与科学,2013,35(2):91-95.

[6] 胡福平,徐美华,沈华明.SVM 的并行计算结构研究及 FPGA 实现[J].微电子学与计算机,2018,35(6):79-83.

[7] 丁宣宣,黄　伟,郭渊博,等.基于 MapReduce 和 Bagging 的并行组合支持向量机[J].信息工程大学学报,2018,19(2):196-202+208.

[8] 刘泽燊,潘志松.基于 Spark 的并行 SVM 算法研究[J].计算机科学,2016,43(5):238-242.

[9] 李　坤,刘　鹏,吕雅洁,等.基于 Spark 的 LIBSVM 参数优选并行化算法[J].南京大学学报(自然科学),2016,52(2):343-352.

[10] 何经纬,刘黎志,彭　贝,等.基于 Spark 并行 SVM 参数寻优算法的研究[J].武汉工程大学学报,2019,41(3):283-289.

[14] 郭欣欣.基于分布式计算的 SVM 算法优化[D].西安:西安电子科技大学,2014.