基于 Spark 的流程化机器学习分析方法①

赵玲玲[1,2], 刘 杰[2], 王 伟[2]

[1](中国科学院大学, 北京 100190)
[2](中国科学院软件研究所, 北京 100190)

摘 要: Spark 通过使用内存分布数据集, 更加适合负载数据挖掘与机器学习等需要大量迭代的工作. 但是数据分析师直接使用 Spark 进行开发十分复杂, 包括 scala 学习门槛高, 代码优化与系统部署需要丰富的经验, 同时代码的复用度低导致重复工作繁多. 本文设计并实现了一种基于 Spark 的可视化流程式机器学习的方法, 一方面设计组件模型来刻画机器学习的基本步骤, 包括数据预处理、特征处理、模型训练及验证评估, 另一方面提供可视化的流程建模工具, 支持分析者设计机器学习流程, 由工具自动翻译为 Spark 平台代码高效执行. 本工具可以极大的提高 Spark 平台机器学习应用开发的效率. 论文介绍了工具的方法理论和关键技术, 并通过案例表明工具的有效性.

关键词: 机器学习; 数据分析; 分布式; 大数据; Spark

Method of Implement Machine Learning Analysis with Workflow Based on Spark Platform

ZHAO Ling-Ling[1,2], LIU Jie[2], WANG Wei[2]

[1](University of Chinese Academy of Sciences, Beijing 10090, China)
[2](Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: By using resilient distributed dataset, Spark is more adapted to iterative algorithms, which are common in data mining and machine learning jobs. However, the development of Spark applications is complicated for data analysts on account of the high threshold to learn scala, the rich experience of code optimization and system deployment, as well as multiple duplicated work due to the low reusing of code. We design and develop a machine learning tool with visible workflow style based on Spark. We design the stages of machine learning with workflow modules, including data preprocessing, feature processing, model training and validation. Meanwhile, a friendly user interface is brought forward to accelerate the design of machine learning workflow model for analysts, with the support of auto parsing from modules to Spark jobs by server end. This tool can greatly improves the efficiency of machine learning development on Spark platform. We introduce the theoretical methods and critical techniques in the paper, and prove its validity with a real instance.

Key words: machine learning; data analysis; distributed; big data; Spark

1 引言

信息技术的发展带来生活的便利与快速增长的数据. 随着以机器学习为代表的大数据分析技术的日益成熟, 大数据为社会经济生活带来了巨大的影响, 并为商业决策提供了大量的帮助. 例如在电子商务行业, 淘宝通过对海量交易数据进行学习, 为用户提供专业的个性化推荐; 在广告行业, 网络广告通过追踪用户的点击对喜好进行预测, 提高用户体验.

但是, 传统的商业关系型数据管理系统已经无法处理海量数据的大容量、多样化与高维度的特点[1]. 为了解决大数据分析的问题, 分布式计算得到广泛的应用. Apache Hadoop[2]是近年广泛使用的分布式系统之

① 基金项目:国家自然科学基金(U1435220)
收稿时间:2016-03-21;收到修改稿时间:2016-04-11 [doi:10.15888/j.cnki.csa.005454]

一. Hadoop 采用 MapReduce 作为严格的计算框架. Hadoop 的出现促使了大规模数据处理平台的流行. 与 Hadoop 同样受到广泛应用的还有 Spark[3], 由伯克利大学的 AMPLab 开发的大数据架构. Spark 融合了批量分析、流分析、SQL 处理、图分析以及机器学习等应用. 相对于 Hadoop, Spark 具有快速, 灵活, 容错性等特点, 是运行机器学习分析程序的理想的选择方案. 但 Spark 是一个开发者使用工具, 要求分析人员具备一定的计算机技术能力, 并且花费大量时间去创建、部署与维护系统.

机器学习的结果严重依赖于数据质量与模型逻辑, 所以为了令分析人员能够专注于流程本身, 不在分析程序编译、运行、并行化等方面花费精力, 本文设计并实现了一个基于 Spark 的流程化机器学习分析工具. 形式上看, 每个机器学习分析任务被分解成不同的阶段, 以组件的方式组成, 降低了使用者的学习成本. 技术上, 通用的算法被封装成组件包进行复用, 通过参数设置实现训练过程的差异化, 减少了创建机器学习分析程序的时间成本. 使用者可以通过拖拽算法组件, 灵活地组建自己的分析流程, 提高应用的创建与执行效率.

本文将通过相关工作与目前存在的产品进行对比展示本工具的特点, 然后再从系统体系结构设计、使用案例阐述业务模型、深入系统模块说明功能运作等部分进行详细说明. 同时, 本文将在最后进行技术总结以及未来研究方向的展望.

2 相关工作

Azure Machine Learning(简称"AML")[4]是微软在其公有云 Azure 上推出的基于 Web 使用的一项机器学习服务, 它内置了基于监督学习和非监督学习的分类、回归、聚类等的 20 多种算法, 并且仍在不断的增加. 但 AML 基于 Hadoop 而且只能在 Azure 上使用, 与之不同, 本文的工具基于 Spark 设计与实现, 并且能够灵活的在不同的虚拟机或云环境上部署.

Apache Zeppline[5]是一个基于 Spark 的响应式的数据分析系统. 其目标是打造集成多种算法库的、互动的、可视化、可分享的 Web 应用. 现已成为开源的笔记式的分析工具, 支持大量的算法库以及多种语言. 但是 Zeppline 没有提供一个用户友好的图形接口, 所有分析程序需要用户编写脚本提交运行, 提高了用户的编程技术要求. 本论文的工具提供组件化的图形工具以及大量的机器学习算法, 用户可以简单快速的定义机器学习流程并运行得到结果.

文献[6]中介绍一个大数据分析服务平台 Haflow. 该系统使用了组件的设计, 可以拖拽组建流程化的分析程序. 并且开放了扩展接口, 可以使开发者创建自定义的分析算法组件. 目前 Haflow 仅仅支持 Hadoop 平台的 MapReduce 算法组件, 本文的工具以 Haflow 为基础, 使其能够支持 Spark 的组件应用, 并提供大量在 Spark 环境下运行的机器学习算法.

3 基于Spark的流程化机器学习分析工具

3.1 机器学习流程概述

本文旨在设计一个面向数据分析师的流程化机器学习工具, 所以需要实现常用的机器学习流程的功能. 机器学习可以为监督学习与非监督学习, 主要依据是否有具体的标签. 标签是观测数据的目标或预测的对象. 而观测数据是用来训练和测试机器学习模型的样本. 特征是观测数据的属性, 机器学习算法主要是从观测数据的特征中训练得到预测规律[7].

实践中, 机器学习流程包括一系列的阶段, 包括数据预处理、特征处理、模型拟合以及结果验证或预测. 例如, 将一组文本文档进行分类包括分词、清理、提取特征、训练分类模型以及输出分类结果[7].

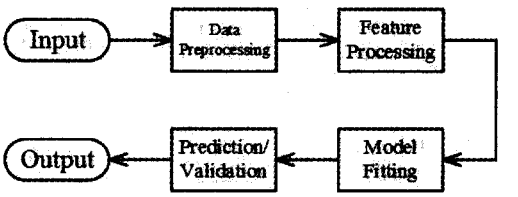

图 1 典型的机器学习流程

这些阶段可以看作是黑盒过程, 并且可以包装成组件. 虽然有很多算法库或是软件为每个阶段提供了程序, 但是这些程序很少是为大规模数据集或是分布式环境准备的, 并且这些程序并不是原生支持流程化, 需要开发人员去连接每一个阶段形成完整的流程.

所以本系统在提供大量机器学习算法组件的同时, 也要完成自动执行流程的功能, 兼顾流程的运行效率.

3.2 系统业务模块设计

本系统将组件做为主要业务功能提供给使用者. 分析人员可以将现有组件自由的组合成不同的分析流

程. 为了能够覆盖常用的机器学习流程, 本系统提供以下几类业务模块: 输入输出模块、数据预处理模块、特征处理模块、模型拟合模块以及结果预测模块. 与其他系统不同, 本工具设计的业务模块以流程中的各阶段为定义, 前后依赖.

① 输入输出模块. 本模块用来实现数据的获取与写入, 主要处理数据源的异构性, 是整个机器学习流程的起点与终点. 为了能够处理不同的数据类型, 本系统提供结构化数据(如 CSV 数据)、非结构化数据(如 TXT 数据)、半结构化数据(如 HTML 数据)的输入或输出功能.

② 数据预处理模块. 本模块包括数据清理、过滤、join/fork 与类型改变等功能. 数据质量决定了机器学习模型准确度的上限, 所以在进行特征提取前, 完善的数据预处理过程也是必需的. 本模块可以对空值或异常值的清理、更改数据类型, 并且可以过滤掉不符合条件的数据.

③ 特征处理模块. 特征处理是在对数据进行建模前最重要的环节, 包括特征选择与特征抽取两种功能本系统目前包含 25 种常用的特征处理算法, .

特征选择是对多维的特征进行选择, 利用算法挑选最有价值的特征, 选出的特征是原来特征的子集. 根据选择的算法不同分为信息增益选择器、卡方信息选择器与 Gini 系数选择器等组件.

特征抽取是将观测数据的特征按照一定算法转换成新的变量, 相对于数据预处理, 对数据的处理规则更加的复杂. 抽取后的特征是原有特征的映射, 包括以下几类:

I.标准化组件. 标准化是将数据的数值型特征映射到统一的量纲的算法. 经过标准化的特征被统一到相同的参考系下, 使训练出来的模型更加准确, 训练过程中收敛更快. 不同的标准化组件使用不同的统计量进行映射. 如 Normalizer 组件、StandardScaler 组件、MinMaxScaler 组件等.

II.文本处理组件. 文本类型的特征由于不能直接计算, 需要映射到新的数值类型变量上. 常用的算法有将文本进行分词建立索引的 TF-IDF 组件, 分词 Tokenizer 组件, 独热编码 OneHotEncoder 组件等.

III.降维类组件. 这类组件将原有的特征通过一定的算法, 将原有的特征信息进行压缩, 用更少的特征进行表示, 如主成分分析 PCA 组件等.

IV.自定义 UDF 组件. 用户可以输入 SQL 自定义特征处理的功能.

④ 模型拟合模块. 模型训练是用某种算法对数据进行学习, 得到的模型可以用于后续对数据的预测. 本系统目前提供大量的监督学习模型组件, 根据观测数据标签性质的不同, 可以分为分类模型与回归模型.

⑤ 结果预测模块. 本模块包括结果预测与验证两个功能.

通过以上通用的业务模块的设计, 用户可以在本系统环境下创建多样化的常用的机器学习分析流程.

3.3 系统体系结构设计

本系统通过 Web 提供用户接口, 以整体架构以 MVC 框架为主, 同时提供机器学习的业务模块以及流程的执行模块, 系统体系结构如图 2 所示.

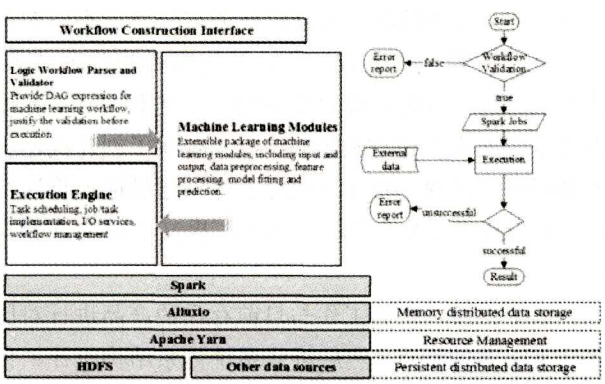

图 2　系统体系结构图与工作流程图

用户通过系统提供的 Web 界面创建形式上的机器学习流程, 提交给系统. 系统将把接收到的原始流程转换成逻辑流程图, 并对流程图进行有效性验证. 流程的有效性验证是分析流程在实际执行前的必要的一环, 当流程有明显的逻辑或数据不匹配等错误时, 能够立该返回错误, 而不是等执行到相应的组件时再报错, 提高了系统的运行效率.

系统的执行引擎是系统的关键模块, 实现多用户和多任务的流程执行功能. 它将验证有效的逻辑流程图翻译成相应的执行模型, 执行模型即是系统可识别的用来调度相应业务组件的数据结构. 执行模型的翻译是一个复杂的过程, 本文将在 4.3 节中进行详细介绍.

4　系统实现及关键技术研究

4.1 中间数据的存储与管理

4.1.1 中间数据的存储结构

在整个机器学习流程中，数据处于流动的状态，具有顺序依赖的组件需要传递中间数据. 为了避免中间数据异构性的问题，本系统规定组件间使用统一的基于 DataFrame[8] 的列式存储结构进行通信. DataFrame 是一种 Spark 支持的以列为主的分布式数据集合，在概念上类似于关系数据库的“表”，但在 Spark 底层对其运算执行做了很多优化. 这种方式保留了结构化数据的关系，并且对特殊的数据属性进行定义，规定 features 和 label 作为模型拟合阶段所需数据的头部，以方便流程的验证与执行.

这种列式存储结构可以被整个系统快速的持久化到中间数据存储层，并且在后面的组件使用时快速的还原成需要的数据对象.

4.1.2 中间数据的管理

中间数据在不同的生命周期需要不同的管理. 当组件对之前的数据进行处理后，即在中间数据的生成阶段，系统会记录中间数据的生成位置，用于传递给下一组件. 在流程执行结束后，所有该流程产生的中间数据将不再被使用，会被系统统一删除. 同时，单个流程的中间数据存储空间有规定的上限，当中间数据产生过多时，流程的资源管理器将采用近期最少使用算法(LRU, Least Recently Used)[9]对数据进行清除，以防止中间数据过多发生内存溢出的问题.

为了保证中间数据的 IO 效率，本系统使用 Alluxio[10]作为中间的存储层，将中间数据全部保存在内存中. Alluxio 是一种基于内存的虚拟分布式存储系统，可以大幅加速数据的读写速度.

4.2 机器学习业务组件的实现方法

4.2.1 基于 Spark MLlib 的机器学习分析组件的实现

本文在第 3.2 节详细的说明了系统的机器学习模块的设计，这些模块通过组件的形式完成主要的数据处理与建模功能. 为了快速的提供尽可能多的算法组件，除了少部分根据机器学习流程的特点编写了处理程序的组件，如输入输出组件、数据清理组件等，很多的组件功能通过 Spark MLlib 自动转换成相应的 Spark Job 完成. Spark MLlib[11]是 Spark 自带的机器学习算法库，包含了大量的分类、回归、聚类、降维等算法. 例如使用随机森林进行分类，系统的执行引擎根据流程的结点信息，实例化具有相应参数的 RandomForestClassifier 对象，调用 fit 方法对输入的数据进行拟合，生成相应的 Model 对象，然后通过中间数据管理模块将模型序列化保存，供后续的预测或验证组件使用. 通过这种方法，能够保证每个学习算法的质量，而且能与 Spark 社区同步，快速的添加新的算法组件.

4.2.2 共享 Spark 上下文执行流程中的组件

流程中的组件有两种运行方式. 一种是作为独立的 Spark 程序调用，每次运行都启动一次 Spark 上下文(SparkContext). Spark 程序在刚开始启动时，会创建上下文环境，确定资源分配，如调用多少线程、内存，之后再进行相应的任务调度. 一般的机器学习流程由很多个组件组成，将会花费大量的运行时间去完成上下文的启动与切换. 另一种方法，可以令每个流程共享同一个上下文，整个流程可以看作是一个大的 Spark 程序. 但系统的执行引擎需要为每个流程创建与管理上下文，在流程结束时也要将上下文对象释放回收资源.

为实现上下文的共享，每个组件都要继承 SparkJobLike 或者其子类，并实现创建组件对象(createInstance)与执行组件(execute)方法. 图 3 是类的设计与继承关系图. 其中，Transformers、Models、Predictors 分别是数据清理与数据预处理模型、学习训练模型、验证与预测模型的父类.

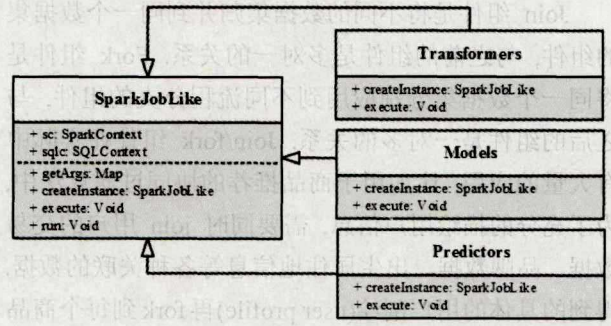

图 3 组件类设计与继承关系图

4.3 机器学习流程的创建与验证

当用户通过图形界面设计好机器学习分析流程并提交后，系统将开始创建逻辑上的分析流程. 系统首先通过对原始流程进行拓扑分析，生成以有向无环图(DAG, Directed Acyclic Graph)来表示的逻辑流程图. 逻辑流程图包括各组件的前后依赖与并行关系，以及输入输出、参数信息.

当前流程的逻辑结构生成后，将对整体流程的有效性进行验证. 具体步骤如下:

① 检查图中每个结点的输入与输出及其他必要的参数信息，缺少则返回错误，如特征处理的组件用户必须定义 input column 与 output column;

② 检查整个流程的完整性，如是否存在至少一个输入组件与输出组件作为开端和结束，否则返回错误;

③ 检查流程图中是否存在自循环，否则返回错误;

④ 检查各个组件是否符合机器学习流程的前后依赖关系，比如特征处理必须在模型拟合之前，不符合则返回错误.

4.4 机器学习流程的翻译与执行

对流程进行验证后，流程图将被提交给执行引擎. 首先系统需要将逻辑的流程图表示成可以直接执行的模型，再转换成基于 Spark MLlib 的机器学习算法组件再串行或并行执行，这个过程称为流程的翻译与执行. MLlib[11]是 Spark 内置支持的分布式机器学习算法库，优化了大规模数据和模型的并行存储和运算. 使用 Spark MLlib，可以快速开发出大量高效的组件程序. 这部分将着重介绍系统如何将流程翻译成可以执行的模型，加速机器学习分析流程的运行.

4.4.1 流程中同时发生多个并行 join/fork 任务

Join 组件是将不同的数据集归并到同一个数据集的组件，与之前的组件是多对一的关系. Fork 组件是将同一个数据集分别应用到不同流程分支的组件，与之后的组件是一对多的关系. Join/fork 组件在实际中有大量的应用，比如用于商品推荐的协同过滤算法中，为了充分的描绘用户信息，需要同时 join 用户的交易数据、品牌数据、出生居住地信息等各种关联的数据. 得到的具体的用户剖绘(user profile)再 fork 到每个商品得到相应的偏好概率[12].

当发生多个数据集同时 join 的任务时，为了高效率的并行执行流程，使用分治算法，将不同的 join 分支分别执行，最后再归并. 当从同一数据集 fork 出多个流程分支时，对每个流程分支并行执行，不影响最终的模型结果. 总之，对有多个 join 以及 fork 任务的机器学习流程要尽可能的并行执行，提高运行效率.

4.4.2 多个串行与并行任务的复合流程的翻译

上一节介绍了当流程中出现多个 join/fork 的并行任务时的翻译方法，但是实际中的机器学习流程并不会是单纯的串行或并行的关系，而是串行的任务和并行的任务组合成的，所以实际中的机器学习流程的情况更加复杂. 要将复杂的流程转换成执行引擎，其难点在于要尽可能的并行执行流程，但不会打乱组件的之间的数据依赖关系. 以下为复合流程的翻译方法:

① 对流程图进行广度优先遍历，确定业务组件间的拓扑关系;

② 以数据预处理、特征处理、模型拟合与预测的阶段为标准划分相同阶段的子流程;

③ 通过关键路径算法判断各子流程内部的执行情况，以拓扑情况确定子流程中分支的层次关系;

④ 上个步骤后得到的同一层次的分支再按照上一节的算法进行优化.

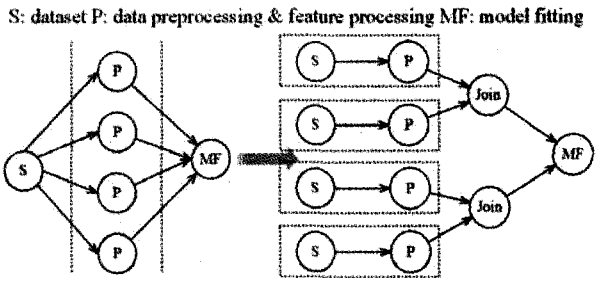

图 4 多个 join 和 fork 并行流程的翻译

5 案例分析

5.1 实验环境与数据说明

目前本系统尚处于原型阶段，为了实验系统功能，本文使用四核处理器、8G 内存、64 位 Ubuntu 系统的单机布署伪分布式的环境进行实验.

实验数据是来自 Kaggle[13]的公开数据集，通过 2003 年至 2015 年的洛杉矶城市的犯罪记录数据，对犯罪类别进行建模. 为了方便流程的展示说明，本文选取了三个原始特征，选用常用的机器学习分析方法创建流程，特征与标签的数据特点如表 1 所示. 总结来说特征与标签以字符串为主，需要数据预处理进行特征提取，并映射成数值型的特征.

表 1 数据特点说明

名称	说明	特点	示例
DayOfWeek	发生时间	只能是周一至周日	Monday
PdDistrict	发生地区	字符串类型，十个地区	BAYVIEW
Address	发生地址	长文本数据，包括标点	800 Block of INGE...
Category	案件类别	多分类标签，字符串	ARSON

5.2 机器学习流程的创建与说明

为了将原始特征转换成训练模型可以计算的数值型特征向量，需要进行一系列的数据预处理工作. 表 2 是对每个特征处理方法的说明，全部的参数设置一般为默认，如有改动会特别说明.

表 2 数据预处理说明

特征名称	处理方法说明	特征处理后
DayOfWeek	Binarizer，周末为 1，其他为 0	0 或 1
PdDistrict	OneHotEncoder，映射成独热编码	[0, 9]
Address	Tokenizer，对地址进行分词	[0,block,...
Tok_output	TF-IDF，计算地址分词的重要性	(100,[27,33...
Category	StringIndexer，映射到[0, 38]	[0, 38]

预处理后得到的特征将通过 Join 组件合并成 features 向量，经过 TF-IDF 后特征向量的维度高但比较稀疏，使用 ChiSqSelector 选择卡方信息量最大的 100 个特征拟合模型. 采用 LogisticRegression-WithLBFGS 拟合多分类模型，然后将测试数据通过训练好的模型进行预测，将结果输出保存成 CSV 文件. 图 5 是将上述分析流程在系统创建后的界面.

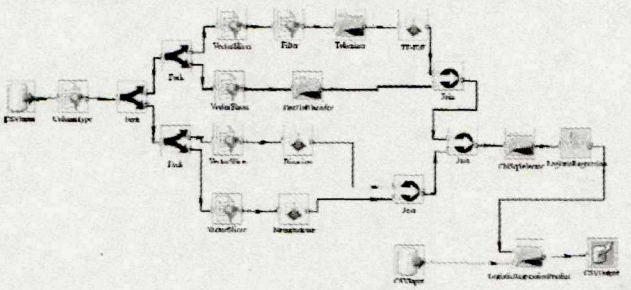

图 5 创建好的流程图界面

5.3 实验结果分析

通过比较测试数据的预测值和实际的 label，准确率在 72.54%左右. 如果向流程中添加更多的特征，模型的复杂度会变大，同时准确率也会上升. 使用本系统，可以方便快速的创建机器学习流程，用户可以专注于分析方法的改进.

本文在第四部分介绍了流程的并行执行优化，为了测试优化方法的有效性，将本实验的数据随机抽取，分成 10%、20%、30%...100%大小的十份数据，将这十份数据分别使用优化过的方法和没有优化的方法执行本实验的分析流程，没有优化是指将流程中的组件按照前后顺序串行执行，获得每个流程的运行时间，单位为 ms，如图 6 所示.

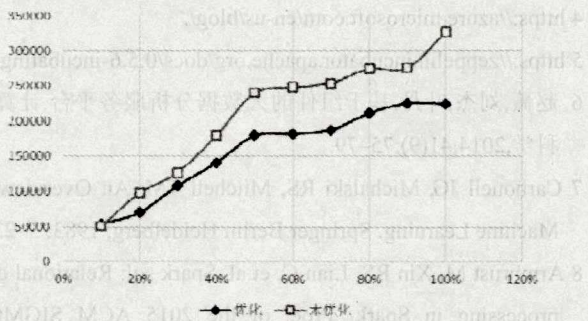

图 6 优化与未优化的时间效率对比图表

可以看出，随着数据量的线性增长，未优化的流程执行的时间增长的更加快，而且到后期时间的增长率有增大的趋势. 而经过优化的流程执行方案，随着数据量的增加，时间增长的相对缓慢，说明系统执行优化方案的有效性.

6 结论

本文为了解决数据分析师采用 Spark 开展大规模数据的机器学习分析的问题，设计并实现了一个分布式的、支持多种机器学习算法的流程化的分析系统的原型. 本文的第三部分从整体介绍了本系统的业务模型与体系结构. 第四部分从各个模块开始详细说明关键技术，包括中间数据的存储与管理、机器学习业务组件的实现、机器学习流程的创建与验证、机器学习流程的翻译与执行. 并且对复杂的机器学习流程的执行在逻辑上进行了优化，将逻辑流程图翻译成可以在物理执行阶段尽可能高效的并行执行的模型.

本系统目前将 Spark MLlib 所有算法自动转换为组件，仍需要在实践中不断的对算法库进行扩展. 同时，未来可以在数据依赖的方面进行研究，如系统可以对数据集自动进行分片，将同一数据集的不同特征的处理任务分配到不同的分布式结点并行处理，提高特征处理任务的执行效率以及分布式资源的利用率.

参考文献

1 Labrinidis A, Jagadish H V. Challenges and opportunities with big data. Proc. of the VLDB Endowment, 2012, 5(12): 2032–2033.

2 http://hadoop.apache.org/docs/current/.

3 Zaharia M, Chowdhury M, Franklin M J, et al. Spark: Cluster computing with working sets. HotCloud, 2010, 10: 10–10.

4 https://azure.microsoft.com/en-us/blog/.

5 https://zeppelin.incubator.apache.org/docs/0.5.6-incubating.

6 赵薇,刘杰,叶丹.基于组件的大数据分析服务平台.计算机科学,2014,41(9):75–79.

7 Carbonell JG, Michalski RS, Mitchell TM. An Overview of Machine Learning. Springer Berlin Heidelberg, 1983: 3–23.

8 Armbrust M, Xin RS, Lian C, et al. Spark sql: Relational data processing in Spark. Proc. of the 2015 ACM SIGMOD International Conference on Management of Data. ACM. 2015. 1383–1394.

9 Megiddo N, Modha DS. Outperforming LRU with an adaptive replacement cache algorithm. Computer, 2004, 37(4): 58–65.

10 Li H, Ghodsi A, Zaharia M, et al. Reliable, memory speed storage for cluster computing frameworks. Proc. SoCC, 2014.

11 Meng X, Bradley J, Yavuz B, et al. Mllib: Machine learning in apache Spark. arXiv preprint, arXiv:1505.06807, 2015.

12 邓爱林,朱扬勇,施伯乐.基于项目评分预测的协同过滤推荐算法.软件学报,2003,14(9):1621−1628.

13 https://www.kaggle.com/c/sf-crime.