2. 绪 论

## 课题背景及研究意义

当代，由于信息的高速增长，海量数据的迅速产生，推荐系统（Recommender System）对及时获取有效信息起着至关重要的作用，推荐系统改变了人们与信息的沟通方式[[1](#_ENREF_1)]。推荐系统通过对海量数据进行辨认、区分与排序，帮人们从海量信息中筛选出其最感兴趣、最有用的部分，大大简化了信息选择的过程。

而随着越来越多的在线服务被使用，推荐系统（Recommender System, RS）在互联网时代也发挥了越来越重要的作用。对个人而言，使用推荐系统RS可以让我们更加高效的使用信息。业界许多公司企业例如亚马逊（Amazon）、奈飞公司（NetFlix）、阿里巴巴（Alibaba），也越来越多的使用推荐系统RS来为顾客和用户提供精确地服务和商品推荐。

目前推荐系统相关技术已然非常成熟，相关特征与模型在实际应用中表现也令人满意，近年来推荐系统在商业活动中的应用也越来越广泛。在整个电子商务体系中，机器学习算法作为推荐系统的支撑，是以个性化推荐作为核心的，其中大多数系统则是基于协同过滤方法来进行推荐的。

大多数实际应用场景下的推荐服务，将计算结果的前N项展示给终端用户，根据用户对其的满意度对推荐结果的性能进行衡量。因此，Top-N推荐的进展在实际应用中有着深远的影响。

随着硬件计算能力的提升和互联网的深度发展，人们可掌握的计算能力突飞猛进，而相应的人们面对的场景也更丰富，更加多样化，数据的规模和复杂程度也迅速增长。在此背景下神经网络类算法得到了长足的发展，而基于神经网络的深度学习则表现出惊人的学习能力，在许多领域取得了长足的进步甚至是颠覆性进展，使机器智能化程度迅速提升。深度学习（Deep Learning, DL）是以神经网络为基础，RBM为核心的机器学习新兴研究领域，作为一类越来越成熟的技术，正在被逐步地引入到推荐系统，其强大的特征学习能力在推荐算法中表现出优异的学习效果。

推荐系统与深度学习的结合，是一种趋势。深度学习的动机在于建立并模拟人脑进行分析学习的神经网络，它模仿人脑的机制来解释信息，例如图像，声音和文本[[2](#_ENREF_2)]。使用深度学习，如自动编码机等，可以学习用户和商品的特征，从而可以在学到的特征上进行进一步的预测模型学习，或者将学到的特征应用到传统的推荐方法中[[3](#_ENREF_3)]。推荐系统为深度学习算法提供了新的应用场景，而深度学习则为推荐系统的发展开辟了新的方向，为其发展注入了新的生命力。

然而，随着信息的进一步丰富，数据膨胀速度越来越快，已经可以预见到不久的将来硬件的性能提升将无法满足数据量的快速膨胀，成为当代复杂运算的瓶颈；而由于推荐系统多次迭代的特性，也对计算性能提出了更进一步的苛刻要求。逐渐的人们开始采用具有众核体系的GPU、单机多设备并行以及分布式数据处理平台来扩展计算能力，满足计算需求。分布式计算平台以其极大的扩展空间，将作为基础构件，整合单机多卡并行，进一步助力信息技术产业的发展。

当深度学习应用到推荐场景下，面对海量计算数据时，基于分布式计算平台的协同深度学习不可避免地成为解决计算资源不足与数据量庞大和海量复杂运算等问题的可行选择，成为海量数据复杂推荐场景下的主要解决方案。

随着互联中的数据越来越丰富，冗余信息越来越多，从中提取出有效信息变得越加困难，研究基于深度学习的推荐问题及其算法，并引入并行学习方法，具有非常重要的现实意义。

## 课题相关国内外研究现状

推荐系统是信息过滤系统的一个子集，旨在通过商品的特性或用户的社交环境，猜测用户的喜爱偏好，向用户展示其关注的项目或者信息。目前，已经有许多方法被提出并应用到推荐系统中，但这些方法主要是基于协同过滤、矩阵分解等技术，这些技术难以分析出复杂的社会网络中的特征。

推荐算法主要有基于协同过滤（Collaborative filtering，CF）的推荐，基于人口统计学的推荐和基于内容的推荐等[[4](#_ENREF_4)]。其中，协同过滤是当下应用最广泛的推荐方法之一，该技术在不同的方向上都引起了研究者的广泛关注。协同过滤的本质就是利用其它用户对商品的历史行为记录来协助对目标用户的推荐[[5](#_ENREF_5)]。协同过滤技术可以分成基于邻域的协同过滤方法和基于模型的协同过滤方法两种[[6](#_ENREF_6)]。而在不同的协同过滤技术中，矩阵分解（Matrix factorization ，简称MF）是基于模型的协同过滤方法中最热门的一种。矩阵分解方法通过和用户-商品评分矩阵分解成一个用户相关的低秩矩阵（用户特征矩阵）和一个商品相关的低秩矩阵（商品特征矩阵），并使用分解后的两个低秩矩阵的乘积来拟合原评分矩阵，同时对未观察到的用户-商品对进行预测。一些改进工作已经被应用到该方法上，如带权重的矩阵分解方法[[7](#_ENREF_7)]、基于非负矩阵的矩阵分解方法[[8](#_ENREF_8)]，和基于矩阵局部性的矩阵分解方法[[9](#_ENREF_9)]等，这些方法都展示了其在推荐系统上的效果。基于矩阵分解的方法中，重要的因素是所学习到的用户特征因子矩阵和商品特征因子矩阵，好的用户商品特征表示能带来更好的预测结果。

传统的推荐系统假设用户之间是独立同分布的，从而忽略了用户之间的社会交互以及联系。然而实际应用于推荐中，社交推荐则时常发生，当用户有推荐需求并通过咨询亲人或朋友时，就是一种社会推荐。随着社交媒体的流行，越来越多研究者开始研究利用了用户间社交关系的社会化推荐系统，有效的利用社交网络用户关系信息可以提高推荐的效果。Hao Ma通过共享相同的用户低秩矩阵，将用户-商品评分矩阵和用户的社交信任网络结合在一起，提出了一个具有可解释性的概率因子分析模型[[10](#_ENREF_10)]。类似的，Hao Ma在矩阵分解的优化函数中引入了基于社会化的正则化项，该正则化项使得在对目标用户进行预测时，跟目标用户有较高行为相似度的好友的特征因子被更加关注[[11](#_ENREF_11)]。社会化正则化项有效的利用了用户的社交关系，提高了推荐效果。

近些年来，深度学习技术已经被证明在自然语言处理、计算机视觉和语音识别中能够学习到很好的数据特征表示。将深度学习技术应用到推荐系统中进行特征学习，能够学习到更具有代表性的用户商品特征。随着深度学习研究的深入，不断的有研究人员将深度学习和推荐系统结合起来，已经有一些工作致力于将深度学习应用到推荐系统中。其中Aaron等人2013年的工作将深度学习应用到音乐推荐中，在针对音乐做为项目的推荐场景下，使用深度学习技术对项目内容进行探测与表达，结合传统分类技术与推荐技术生成推荐结果[[12](#_ENREF_12)]，更是将两者的结合推向一个新的高潮。

个性化推荐系统中存在一些问题，主要包括稀疏性问题，冷启动问题和可扩展问题等，深度学习可以在一定程度上解决个性化推荐领域中的冷启动以及稀疏性问题。通过发现数据中的高级特征与隐含分布，进而推断项目间、用户间以及用户与项目间的内在联系，进而缓解推荐系统的局限性。深度学习通过神经网络发现用户与项目之间的内在联系。Zhang利用用户和项目通过神经网络构建分布向量，通过分布式表达将用户向量与项目向量同时对神经网络进行训练，进而提升目标系统的推荐效果[[13](#_ENREF_13)]。Liang提出了一种概率评分下的自动编码机模型用于无监督特征学习，通过自编码机发现用户的隐形特征，生成对用户偏好表达的隐式向量表达，结合最近邻模型的协同过滤方法提供个性化推荐[[14](#_ENREF_14)]。Salakhutdinov提出了基于玻尔兹曼机的协同过滤算法，得到了比Netix系统更好的推荐效果[[15](#_ENREF_15)]。为了有效利用外部信息，Wang提出了一个层次贝叶斯模型，称之为协同深度学习（Collaborative Deep Learning, CDL），使用深度学习来学习内容信息，并将之与协同过滤相结合来得到评分矩阵[[16](#_ENREF_16)]。类似于Aaron等人的工作，基于音乐内容的推荐结合深度学习，利用DBN进行内容特征提取进而提升推荐结果准确性，取得了令人满意的进展[[17](#_ENREF_17), [18](#_ENREF_18)]。业界也有将用户行为看做序列化信息或将推荐整个过程当做序列化信息，利用RNN对序列化数据进行学习和探测的研究[[19-21](#_ENREF_19)]。有研究利用自动编码机对特征矩阵进行降维，进而实现数据填充或者推荐算法运行效率的提高[[22](#_ENREF_22), [23](#_ENREF_23)]。

已有的工作证明，深度学习在推荐系统中学习到的特征具有代表性，对于解决推荐系统中数据稀疏问题有效，可以提高学习性能。应用中，推荐系统的数据是复杂且稀疏的，基于矩阵分解的协同过滤方法学习到有效特征信息变得十分困难。目前基于深度学习的推荐算法还比较少，但初步的研究成果已经表明它们在实际数据应用中的有效性。

## 现有算法存在的问题

目前已有的推荐系统可以被分为三大类：基于内容的推荐（Content-based methods），基于协同过滤的推荐（Collaborative Filtering based methods），和混合推荐（Hybrid methods）。基于内容的推荐利用用户画像和产品描述进行推荐；基于协同过滤的方法使用用户的历史行为和偏好进行推荐，并不关心用户或者产品的内容信息；而混合模型则是将两者进行联合，通过权衡两者的优劣进行平衡，进而达到理想的模型复杂度和结果表现。三类方法各有优劣，在不同应用场景下各有所长。

1. 内容类推荐（Content-based methods），内容类推荐是对推荐项目内容进行分析与统计，根据用户画像对用户兴趣进行提取。根据不同用户可能感兴趣的项目由项目的内容计算结果为主维度进行推荐。内容类推荐能够利用历史数据快速为用户生成推荐结果，不同内容的项目能够通过不同喜好的用户浏览记录以及用户的偏好信息差异性展示给用户。用户的偏好文件可以通过调查问卷、项目评分和用户分类调查信息来构造，进而推断不同用户的不同兴趣偏好。内容类推荐可以高效的利用用户画像信息和被推荐项目信息以及专家经验，为用户进行差异性展示；但此类方法对用户画像信息和项目内容信息依赖严重，当用户未进行反馈或者项目信息不够明确时，往往出现推荐失误。并且内容类推荐的实时性也相对其他类型推荐略有不足[[24](#_ENREF_24)]。
2. 协同过滤类推荐（Collaborative Filtering based methods），协同过滤中项目被推荐给特定的用户，是根据与其相似的用户对项目的偏好，或者是同样用户对相似项目的偏好信息。当用户间具有相同的偏好记录或者有相同的调查信息，系统将会为其中一部分用户展示另一部分用户感兴趣或者反馈好评的项目；当用户对不同的项目进行反馈，系统根据其反馈信息，向其推荐与正反馈关联度较强或相似的项目。协同过滤能够充分利用用户的操作习惯与反馈信息，结合历史记录进行推荐；不需要针对性对用户进行画像信息的统计与调查，同时不需要对项目内容进行关注，在模型复杂度上具有明显优势。此外能够实时记录用户操作习惯，模型修改相对容易[[25](#_ENREF_25)]。过滤系统收集用户的兴趣信息或者用户对项目的兴趣反馈来计算项目的相似性或者用户的兴趣偏好，具有相同兴趣的用户往往被分为同一类别，展示相似内容的项目。但协同过滤最大的缺点是冷启动问题，当用户或者项目为新加对象，系统往往不能很好的根据历史信息对其进行分析与操作，导致推荐性能大幅降低。此外该方法还有两个显著的不足：数据稀疏性（Sparsity）和模型可扩展性（Scalability），评分矩阵十分稀疏导致可利用信息率极低并且计算操作多为无效运算；同时推荐系统若给予CF，则模型计算成本随着用户和项目的数目增长呈现超线性增长[[26](#_ENREF_26), [27](#_ENREF_27)]。
3. 混合型推荐（Hybrid methods），结合上述两种方法，进行混合推荐的系统逐渐被提出。混合型推荐不仅整合两者优势，并且将两者不足在一定程度上互补，降低算法或者模型的局限性。在拥有较多历史信息和反馈记录的基础上，利用协同过滤推荐结合内容分析能够进一步提高算法表现，同时充分利用项目和用户的辅助信息，提高数据置信度。当历史数据较少，可根据已有历史数据进行推断，结合内容或者用户分析，削减冷启动问题对模型系统的局限。本文所提算法可归为此类。但由于深度学习作为新兴领域，其在内容推荐的应用较少；将深度学习与协同过滤结合进行推荐的研究更是凤毛麟角；传统方法无法充分使用海量数据，造成大数据时代下的数据浪费。深度学习理论上具有无限的模型复杂度，可充分利用数据进行内容学习[[28](#_ENREF_28)]；并且超强的特征提取能力也为协同过滤算法带来新的发展契机。

## 论文研究内容

本文结合协同深度学习算法范式[[16](#_ENREF_16)]，依托Apache Spark分布式计算平台，进行了协同深度学习算法的并行化研究，提高了推荐系统对数据信息的使用程度，同时提高了推荐系统在大数据时代下的可扩展性。利用分布式平台，将协同深度学习（CDL）算法并行化。CDL通过联合自动编码机对项目内容的深度表示学习和协同过滤评分矩阵的反馈信息，构建了一种多层的贝叶斯模型。本文在原CDL基础上对其进行了细微改进。并在真实数据集上结合多组实验证明了相对于传统推荐，CDL所带来的提升。随后，描述了CDL移植到Spark分布式平台计算框架下的成果，同时产生了一套基于Spark分布式平台的基于pipeline的机器学习框架。通过对数据一系列处理，验证海量数据下并行化CDL的可扩展性，并与单机性能做了对比分析。

本文贡献可简要概括为：

1. 对协同深度学习进行改进，提出了一种CDL算法范式下的一种算法实例。
2. 利用分布式计算平台Spark，将新兴推荐算法应用到推荐系统，并赋予强可扩展能力。
3. 实现并完善一套基于Spark的Pipeline分布式机器学习框架，简化机器学习工作流程。

## 论文的组织结构

本文介绍了一种将协同深度学习并行化的方法，主要分为五部分：

1. 对课题背景、意义，国内外研究现状做了调研与介绍，并对现有算法进行了简要分析，发现当前算法的待解决问题与不足。
2. 介绍了课题相关理论，包括协同深度学习（CDL）、自动编码机、概率矩阵分解；机器学习算法模型并行化现状与思想；此外介绍了基于Spark的分布式并行技术。
3. 详细分析CDL算法，并对多种模式下的算法模型进行探索与实验对比。在原CDL基础上对其进行了细微改进。并通过在真实数据集上的多组实验证明了相对于传统推荐，CDL所带来的提升。
4. 详细描述CDL移植到Spark分布式平台计算框架下实现并行训练的思路与实现，并产生一套基于Spark分布式平台的pipeline机器学习框架。通过对数据一系列处理，验证海量数据下并行化CDL的可扩展性，并与单机性能做了对比分析。
5. 作出结论。CDL在推荐准确率上较大优势。在海量数据情景下有较强可扩展能力。机器学习框架提升了算法的易用性。
6. 深度学习与推荐系统相关理论研究

## 协同深度学习

协同深度学习（CDL, Collaborative Deep Learning）是将深度学习算法与协同过滤算法相结合，应用于推荐系统，提升推荐系统性能，改善推荐结果的一种算法范式[[16](#_ENREF_16)]，或称之为一类算法。不同于早前的研究，神经网络仅仅作为被推荐项目内容的识别算法；CDL提供了项目内容与用户评分矩阵共同训练的崭新思路。作为一种算法范式，能够在不同应用场景下灵活切换其组成部分，具有在不同场景下均有令人满意表现的能力。

本文介绍的CDL作为协同深度学习范式的一种情况，分别使用了堆叠降噪自动编码机和概率矩阵分解作为算法的能动结构，分别对被推荐项目的内容和评分矩阵进行识别与计算，给出推荐结果。

### 自动编码机

自动编码机（Auto-Encoder）是一种无监督学习算法[[29](#_ENREF_29)]，作为神经网络的一种变体也被称作自编码神经网络，使用反向传播（Back-propagation）算法，将输入值作为目标输出，比如。自动编码机主要由两部分组成，分别是编码部分和解码部分。训练中，编码部分和解码部分同时优化，使得。亦即，它试图逼近一个恒等函数，从而使输出接近于输入。自动编码机的逻辑结构如图2-1 。

|  |
| --- |
|  |
| 1. 自动编码机逻辑结构 |

一般的，自动编码机隐含层节点少于输入层节点，迫使其学习输入数据的压缩表示，换言之自动编码机需要由较少的隐层神经元输出重构出原始高维数据。通常情况下，数据中隐含着一些特定结构，使得不同维度间彼此相关，那么就使得算法可以发现输入数据中的相关性，进而得到数据的有效低维表示，即数据压缩表示，或者说数据去除冗余信息后的特征表示。图2-2 为我们展示了一个简单自动编码机的结构。

|  |
| --- |
|  |
| 1. 简单的自动编码机结构 |

图2-2中的自动编码机，作为前馈神经网络，分别由输入层、隐含层和输出层组成。其中输入层为数据线性表示层，隐含层和输出层中每个节点代表神经网络的一个神经元。神经元由上层网络输出的加权和通过激活函数得到输出，并提供下一层网络使用。神经元可以简单表示为如图2-3 所示，激活函数一般为非线性函数，常用Sigmoid函数，其函数表达式为：

|  |  |
| --- | --- |
|  |  |

一般的，自动编码机包括编码部分和解码部分，两部分可以概括为：

|  |  |
| --- | --- |
|  |  |
|  |  |

其中即为非线性激活函数，权值矩阵分别为、，偏置项为、，隐含层输出为。给定一组输入数据，那么重构误差可以表示为。整个自动编码机的最终目标就是通过最小化重构误差，得到恰当的权值、以及偏置项、。整个网络的重构误差可表示为。

|  |
| --- |
|  |
| 1. 神经元 |

截至目前给出了自动编码机（AutoEncoder）的详细定义与计算，并给出了其求解的目标函数。其中当多个自动编码机进行堆叠，上一层编码层（Encoder）的输出为下一个编码机的输入时，我们将他们并称为堆叠的自动编码机（Stacked AutoEncoder, SAE）或者栈式自动编码机，如图2-4 所示。堆叠的自动编码机相对朴素自动编码机可以学习更加深层次的抽样表达特征，业界已经证明了在神经网络充分训练的前提下，神经网络对函数的表征能力与网络层数成正相关[[28](#_ENREF_28)]，简单来说就是神经网络层数越深，就越能够提取出相比原始数据更加抽象和更加一般的特征，进而加强模型的鲁棒性和数据拟合表现。

对应的，如果为自动编码机输入的数据是经过噪声处理的，而最终学习结果为干净的数据；那么模型学习的过程中会迫使自动编码机将噪声过滤，进而获得鲁棒性更强的数据压缩表示[[3](#_ENREF_3)]。我们称这种自动编码机为降噪自动编码机（Denoise AutoEncoder）。其结构可简单表示为图2-5 。

|  |  |
| --- | --- |
|  |  |
| 1. 堆叠神经网络 | 1. 降噪自动编码机 |

在使用自动编码机的过程中，常常将堆叠和降噪两种模式同时使用，一般称之为堆栈降噪自动编码机（Stacked Denoising AutoEncoder， SDAE）。SDAE不仅拥有自动编码机的对数据学习近似恒等的函数表达；同时具有SAE提取高度抽象特征的能力与DAE获得数据鲁棒性特征的能力。能够获得更加通用的恒等函数和数据特征表达，同时提取特征也更加鲁棒，具有更强的泛化性能。相对的，SDAE学习过程相对缓慢，同时面临梯度弥散的问题。当单层AE足够获取所需特征表达，往往倾向于浅层网络结构。

### 概率矩阵分解

传统的传统的基于协同过滤的方法是通过用户的历史记录来推断相似的模式，进而推荐相关项目，而并没有用到项目的内容信息。而相对的在推荐方法中有一类被称为隐含引子模型，也取得了巨大的成功，相比利用邻近算法的模型能够取得更良好的表现[[30-36](#_ENREF_30)]。在本文中，我们主要关注在隐含模型中的矩阵分解相关算法。

隐含引子模型中，基于矩阵分解的算法在推荐系统中有着良好的表现，该类算法一矩阵分解为基础，将用户和项目表示到一个共享的低秩空间中。

|  |  |
| --- | --- |
|  |  |

并利用公式（2-4）来预测用户u对项目v的喜好程度。通过已有矩阵项对现有评分矩阵进行分解，利用分解得到的隐含向量进行矩阵乘法操作，可以实现缺失数据填充的操作，进而利用不断对隐含向量的学习，得到越来越精准的缺失填充值。矩阵分解算法预测过程为图2-6 。

|  |
| --- |
|  |
| 1. 矩阵分解 |

在使用矩阵分解模型进行推荐前，我们需要计算算法模型的相关参数，也就是隐式空间下的用户向量和项目向量。通常利用已有的用户对项目的评分数据构成矩阵来进行矩阵分解，利用公式（2-5）来计算各自隐含向量的参数。

|  |  |
| --- | --- |
|  |  |

其中，，和分别为正则化参数。

此外，在推荐系统中，矩阵分解往往会利用概率模型，当算法引入专家的先验经验后，算法模型升级为带有状态的概率模型。一般的，由专家经验为隐含向量加入先验条件，这种加入先验概率的矩阵分解算法就是概率矩阵分解（Probabilistic Matrix Factorization，PMF）。

PMF在推荐系统能够取得良好的预测表现，但有些许不足。

1. 学习到的低纬度空间向量解释性不佳。
2. PMF对于冷启动问题十分敏感，当用户或者项目没有历史数据时，PMF无法工作。
3. 未能考虑用户间的关系，造成数据信息丢失。

## 机器学习的并行加速

机器学习作为人工智能的主体与核心，各种机器学习算法已经应用到各个领域，如语音识别、图片分类、推荐系统、手写输入识别、漏洞发现等领域。极大提高了计算机智能化程度和各种系统的性能。

随着大数据时代的来临，处理海量数据，建立大规模计算模型成为必然需求。现有数据种类繁多，许多数据有非常稀疏的特点，比如推荐系统关系矩阵、文本词袋模型等。在实际应用场景下，数据量已然达到快速处理有困难的程度，而当数据稀疏度上升，数据量和机器学习算法的计算量更随之成几何级上升。故而海量数据场景下的算法并行变得十分必要。本文提出了一种基于Spark的机器学习框架，利用Spark分布式内存计算的固有优势高效处理海量数据。

## 基于Spark的分布式计算框架

|  |
| --- |
|  |
| 1. Apache Spark计算平台生态 |

Apache Spark作为Apache基金会的顶级项目，以稳定、快捷、高效的特点迅速占据了大数据时代的制高点。其中包含了丰富的应用库，可以满足多种大数据环境的使用需求[[1]](#footnote-2)。其中的Spark MLLib更是成为了大数据分布式机器学习的代表产品。Spark发展至今已经形成完整的生态如图2-7 。

虽然其中包括了多种数据处理方法和丰富的算法实现，但不同环境下不同需求仍需要对逻辑进行定制。现有的大数据推荐场景往往有着复杂的业务逻辑，不仅涉及Hive、Kafka、Storm、Redis、Hdfs等一系列大数据处理分布式平台，而且其数据预处理与特征提取的逻辑十分复杂。在Hadoop时代，往往通过业务人员和开发人员协作，手动进行操作，不仅效率低下而且误操作漏操作概率也相应上升。

鉴于此本文基于Spark开发了一套针对特定数据的大数据机器学习框架，依托于Spark及其Spark MLLib组件实现了大数据分布式机器学习框架，利用Scala Reflect统一了算法使用流程，简化代码逻辑和开发工作。不仅使从源数据到最终模型生成完全自动化，也降低了人工的参与程度，提高了业务流程稳定性。业务人员只需配置配置文件中的制定参数，即可一步运行机器学习算法。框架的提出旨在提高机器学习处理流程的效率，降低机器学习算法使用的门槛。

### Spark分布式平台

Apache Spark，2014年成为Apache基金顶级项目。它以快速、通用、简单等特点成为当前流行大数据处理模型。Spark提出的分布式内存计算框架，既保留了MapReduce的可扩展性、容错性、兼容性，又弥补了MapReduce在这些应用上的不足。由于采用基于内存的集群计算，所以Spark在这些应用上相对MapReduce有100倍左右的加速[[37](#_ENREF_37)]。

|  |
| --- |
|  |
| 1. 大数据时代下Spark地位与组件 |

此外Spark可以部署在Hadoop集群环境下，利用Yarn进行资源管理，拥有直接访问HDFS文件系统的能力。但不同于MapReduce中间过程和计算结果需要读写HDFS，Spark将计算结果保存在内存中，从而不必频繁读写HDFS，大幅减少了IO操作，提升算法运行效率，使算法运算时间大幅缩短。

值得强调的是，分布式系统需要一个统一调度角色，一般称为管理者（manager），而其他各运算节点称为工作者（worker）。一般情况下，并行程序运行过程中，由workers分别计算相对独立的并行步骤，manager统一调度、管理、统计。Spark在云计算领域所处地位以及其主要功能模块如图2-8 。

### Spark MLLib

Spark机器学习算法库（Spark Machine Learning Library, Spark MLLib），是Spark框架下原生的机器学习算法库，包含了多种算法与数据处理逻辑。基于RDD和DataSet，依托于分布式集群，可以高效快速的完成海量数据下机器学习模型的训练。由于Spark基于内存运算的优势，机器学习的模型训练效率得到了大幅度提升。

|  |
| --- |
|  |
| 1. Spark MLLib组件构成 |

截止Spark 2.0.1，Spark MLLib中已经支持了包括统计、假设检验、线性运算、高效算法模型格式、数据预处理、特征提取、分类、聚类、回归、频繁项集挖掘、协同过滤等一系列机器学习逻辑[[2]](#footnote-3)。同时借鉴Python机器学习库Scikit-learn，引入了DataSet数据类型，自Spark 1.6版本后引入了基于DataSet的机器学习处理逻辑，进一步提升了模型训练的效率与开发鲁棒性。

Spark MLLib作为Spark的重要组成部分，正在处于高速发展与极度活跃时期，Intel的BigDL作为基于Spark和Intel MKL的分布式深度学习已经开源且受到了业界高度关注[[3]](#footnote-4)，同时基于Spark的神经网络已有简单的MLP被作为Word2Vec支撑组建被引入。Spark平台下的深度学习实现与集成正在越来越完善[[4]](#footnote-5)[[5]](#footnote-6)。

此外Spark MLLib还引入了模型评估逻辑，可以根据用户的配置，自动高效的实现多种参数组合下模型的训练与评估，并给出最优模型及其参数组合。大大简化了机器学习分布式开发的流程，降低了大规模数据下机器学习的开发门槛。

### Scala Reflection

反射（Reflection）是一种程序检查特性，用于监控程序运行时的状态。同时，反射还拥有对自身进行更新的能力，也就为运行时多态提供了另一种思路。反射拥有着悠久的历史，在面向对象式编程、函数式编程和逻辑编程中都有它的影子。许多语言本身就将反射作为其自有的一种编程范式；在此同时，随着时间的发展，越来越多的编程语言逐渐演变出了对反射的支持。

在一款程序中，反射能够提供对隐式元素的重现（例如，对其进行解释），这些隐私的元素可能是静态程序元素类似于类、方法和解释；也可以是动态的程序对象类似多线程中断、运行时事件、方法调用和字段访问等。通常区分编译时反射和运行时反射是靠反射是什么时候被执行来判断。编译时反射为程序开发提供了强大的转换器和生成器，与此同时运行时反射则常被用来适应语言的语义或者支持不同组件的延迟绑定。

在Scala 2.10之前，Scala并没有原生的反射支持，而是采用Java的反射API来实现对类和对象的动态检查以及对其成员的访问。然而，许多Scala特有的元素在Java的反射下会丢失特性且不可恢复，例如Java的元素没有函数和特质，暴露出的Java对象类型也与Scala相出入，无法表示更高阶类型和路径依赖。此外Scala的通用类型在Java的反射中也不得不被限制。在Scala 2.10后，Scala引入了一个新的反射库，不仅解决了Java对Scala特定和通用类型运行时反射的不足，而且为Scala增加了更为强大的反射功能。在对Scala类型和泛型全面支持运行时反射外，Scala 2.10还提供了宏编译时反射，以及将Scala表达式转换为抽象语法树（Abstract Syntax Trees, AST）的能力[[6]](#footnote-7)。

1. * + 1. 运行时反射

运行时反射是为Scala提供运行时反射能力的模块，在运行时（Runtime）给定一个对象的实例，反射可以为Scala提供如下能力：

1. 检查该对象的类型，支持通用类型的对象。
2. 实例化新对象。
3. 访问或者调用该对象实例的成员。

由于Scala运行与JVM（Java Virtual Machine）环境下，则类似于其他JVM语言，Scala在编译时会被抹去自有的类型，这就意味着如果对一个对象的实例进行类型审查时，可能无法对其类型信息以及成员进行访问。而运行时反射则避免了这种情况的发生。

* + - 1. 编译时反射

Scala反射支持元编程（meta-programming）的范式。元编程的提供似的程序能够在编译阶段对自身进行修改。Scala的编译时反射利用宏定义的形式实现元编程范式，提供编译时反射能力，进而可以使Scala在编译时能够通过操作抽象语法书来执行方法。

Scala的基于宏的编译时反射提供了与运行时反射一致的应用程序接口（Application Programming Interfaces, API），这使得Scala编写的代码可以通用与编译时反射和运行时反射。

本文所提出的基于Spark计算平台的机器学习框架利用Scala反射实现运行时的不同情景下代码逻辑的统一，精简了代码量，并简化了机器学习处理的逻辑流程，将框架使用接口统一。在确保逻辑的基础上，实现算法接口调用的统一，程序入口统一以及用户操作统一。

## 本章小结

本章简单介绍了本文所使用基础技术构件，包括协同深度学习算法的基础组成算法自动编码机和概率矩阵分解；将算法并行化的分布式操作平台Apache Spark 及其Spark MLLib组件库；为统一逻辑提供运行时多态的Scala反射。本文随后的协同深度学习的并行化研究和基于Spark的机器学习框架的实现，将基于以上技术基础实现进一步的研究与论述。

1. 基于自编码机和矩阵分解的协同深度学习

## 引言

深度学习的蓬勃发展向人们展示了其强大的特征学习能力，在推荐系统中的应用逐渐引起人们的关注。利用深度学习学到的项目内容特征作为推荐系统的辅助信息，提高推荐算法的性能已经有了许多成果。而神经网络本身作为推荐系统的基础算法直接应用于推荐相关研究却很少。

协同深度学习，是基于神经网络，同时利用神经网络的特征学习能力和神经网络能够拟合任意复杂数据分布的能力，联合对数据进行分析和学习得到推荐结果的模型。CDL可以自动的提取项目内容的高度抽象特征，并且捕获项目或者用户间隐含关系和相似性。作为一个通用的概率模型框架，CDL首次将深度学习模型与推荐系统利用统一的贝叶斯视角整合在一起，同时由于贝叶斯概率视角的天然优势，使得学习到的项目的附加信息可以很容易的被解释。在实际实验中也CDL也表现了良好的性能与结果。

## 问题定义及算法模型

类似于[[16](#_ENREF_16)]中使用训练数据的隐式反馈提升推荐系统的性能，本文定义全量数据集包含个项目，用维矩阵表示。中的第行为对应项目的内容向量，用维向量表示。对于个用户，对个项目的评分可用维度的评分矩阵表示。当第个用户对第个项目有正向反馈时，；否则为0。

自动编码机中、、分别表示加入噪声的输入、原始数据输入和自动编码机第层网络的输出。、为维矩阵，为维度矩阵。类似于，的第行表示为。Wl和bl分别为第l层网络的权值矩阵和偏置向量。表示矩阵的第列。神经网络的总层数为。是权值矩阵和偏置向量的简称。

本文所描述算法，需要从数据集中分别对个项目进行学习，自动编码机对内容进行提取。将得到的内容向量与偏置向量整合后作为推荐系统的项目信息。利用矩阵分解进行评分预测，通过对计算评分进行排序，按分数由高到低进行推荐。获得Top-N推荐结果。

### 自动编码机的概率图模型

更一般的，类比于文献[[16](#_ENREF_16), [38-41](#_ENREF_38)]，我们可以利用以下步骤定义广义堆叠降噪自动编码机（Generalized Bayesian SDAE）：

1、对于SDAE的每一层网络：

1. 记当前列为*n*，对每一列的权值矩阵进行服从如公式（3-1）分布的采样。

|  |  |
| --- | --- |
|  |  |

1. 对第*l*层偏差向量，采样使之服从如公式（7）的先验分布。

|  |  |
| --- | --- |
|  |  |

1. 对第*l*层数据矩阵的每行数据*j*，采样使之服从形如公式（8）的分布。

|  |  |
| --- | --- |
|  |  |

2、对于每一条输入数据，利用对公式（9）的抽样获得其初始数据。

|  |  |
| --- | --- |
|  |  |

为对自动编码机进行求解，本文选择反向传播（Backpropgation, BP）来进行学习和优化，此外目标函数学习使用了梯度下降（Gradient Descent）思想。另外，为提高算法运行效率，我们采用随机梯度下降算法（Stochastic Gradient Descent, SGD），即计算每一条样本，实时更新学习模型。

当模型训练开始，manager首先进行map操作，将初始化参数分发至各worker节点，包括数据规模，隐藏层配置，输入数据路径，正则化参数，随机数种子等。此后每个worker独立读取数据，利用随机数种子分别初始化自动编码机的各项参数，其中由相同的随机种子保证各worker上神经网络初始状态一致。

随后各worker分别利用各自数据分片训练自动编码机，worker每次处理一条数据样本。将读入样本进行正向传播，即利用输入数据计算隐藏层和输出层的结果。完成后进行反向传播，利用得到的输出数据误差计算出神经网络的参数梯度。利用所得梯度，完成网络参数更新，至此完成模型的一次迭代。整个训练过程需要重复迭代，保证所分得数据分片中的每一例样本至少计算一遍。

各worker利用一部分数据完成模型训练后，进行Reduce操作，将各节点计算结果进行约减。其中约减操作定义为worker将计算所得模型参数上传至 manager，由manager对各参数取平均。至此完成系统的一次迭代。

对系统进行迭代，保证所有worker均完全训练，至少保证训练集中所有数据均参与到系统模型的训练中。最终在manager上得到一个训练完毕的模型。值得指出的是，实际操作中可以根据集群实际情况，对数据进行多种划分模式，包括但不限于差异化节点数目、节点间不同数据条数、各次迭代处理的数据条数以及系统整体对数据比例划分。

自动编码机模型可具体表示为如下公式：

正向传播：

|  |  |
| --- | --- |
|  |  |
|  |  |

反向传播：

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

权值更新：

|  |  |
| --- | --- |
|  |  |

其中公式中符号表示含义如表3-1所示。

|  |  |
| --- | --- |
| 1. 公式符号含义 | |
| 符号 | 含义 |
|  | 列向量，输入数据 |
|  | ，矩阵，权值矩阵 |
|  | ，列向量，权值向量 |
|  | 激活函数 |
|  | 表3-1（续表） |
| 符号 | 含义 |
|  | 列向量，隐藏层输出 |
|  | 列向量，输出 |
|  | 指示函数 |
|  | ，列向量，误差 |
|  | ，矩阵，更新值 |
|  | 常数，步长 |
|  | 正则化参数 |

其中指示器函数为：

|  |  |
| --- | --- |
|  |  |

自动编码机可以简单罗列为一种神经网络，结构图如图3-1 。N为输入数据样本数量；M为输入数据维度；K表示隐藏层节点目。

|  |  |
| --- | --- |
|  |  |
| 1. 堆叠降噪自动编码机实例 | 1. 堆叠降噪自动编码机图模型 |

广义自动编码机，根据定义可以表述为概率图模型，形如图3-2 。其中、作为、事件的参数条件，分别决定两个概率事件的表现。

### 概率矩阵分解的概率图模型

类似于自动编码机，我们对概率矩阵分解模型的初始化通过一系列操作完成。得到表3-2。

|  |
| --- |
| 1. PMF模型定义 |
| 概率矩阵分解 | |
| 1 对每个用户i，初始化用户隐含向量为~  2 对每个项目j，初始化项目向量隐含向量为~  3 对每个用户-项目评分对，初始化反馈值为~ | |

其中c是准确化参数，表示预测评分的置信度。

至此概率矩阵分解模型构建完成，我们可以用概率图模型来表示，如图3-3 。

|  |
| --- |
|  |
| 1. 概率矩阵分解概率图模型 |

当c为1时，模型的最大后验估计符合公式min，当c越大时，表示评分可信度越高。实际观测中，如果评分为0，则可能有两种情况，用户i对项目j不感兴趣或者为进行过评分反馈。试验中我们令c取值满足公式（3-21）。其中a>b>0，作为超参数调整模型对待预测值的预测。本文在构建概率矩阵模型后，利用随机梯度下降对参数进行计算学习来拟合模型。

### CDL的概率图模型

类似于[[16](#_ENREF_16)]，本文使用SDAE作为算法的组成部分，一般化的协同深度过滤构建过程可被描述为：

1. 对堆叠降噪自编码网络的每层网络分别做a)、b)、c)操作。
2. 对当前层网络权值矩阵记，对权值矩阵的每一列阵进行服从如（3-13）分布的采样。

|  |  |
| --- | --- |
|  |  |

1. 对第*l*层偏差向量，采样使之服从如公式（3-14）描述分布。

|  |  |
| --- | --- |
|  |  |

1. 对第层数据矩阵的每行数据，按公式（3-15）进行采样。

|  |  |
| --- | --- |
|  |  |

1. 对于每一条输入数据，进行初始化操作。
2. 初始化模型输入，使之服从分布（3-16）。

|  |  |
| --- | --- |
|  |  |

1. 创建项目的隐含偏置向量服从于公式（3-17），并由公式（3-18）得到项目隐含向量。

|  |  |
| --- | --- |
|  |  |
|  |  |

1. 为用户创建隐含向量，对每位用户按照公式（3-19）进行初始化操作。

|  |  |
| --- | --- |
|  |  |

1. 创建评分矩阵初始值，对于每对用户项目评分，按照公式（3-20）进行采样操作。

|  |  |
| --- | --- |
|  |  |

此处的分别为模型的超参数，则是对于评分矩阵中每对用户-项目评分矩阵的置信度。类似[[30](#_ENREF_30)]，本文按公式（3-21）对进行赋值。

|  |  |
| --- | --- |
|  |  |

其中作为联络用户评分和内容信息的通道，配合的使用，能够学习到项目间相似度或隐含关系的高效表达。考虑到计算上的效率，设定为无穷大。

|  |
| --- |
|  |
| 1. the Graphic Model for Collaborative DeepLearning 2. 协同深度学习概率图模型 |

## 协同深度学习算法描述

在按照图3-2 、图3-3 、图3-4 利用相关公式构建模型后，得到了加入根据经验设定先验概率的初始化模型，随后我们分别对各部分进行学习和优化，最后对整个模型进行调优，模型收敛后得到学习完成的模型。

### CDL算法的思路

CDL训练时主要对概率图模型进行优化，为便于理解可以类似EM算法看做两部分，分别是PMF的优化和SDAE的优化。当给定用户和项目隐含向量时，使用坐标梯度下降对PMF的权值及项目j对应的偏置向量进行调整；在完成一次PMF的调整后，根据PMF的结果对SDAE进行优化，在对调整后的项目偏置向量进行剥离后得到的项目隐式表达，可得到SDAE对应输出，利用反向传播对神经网络进行调整。

调整策略为：

|  |  |
| --- | --- |
|  |  |
|  |  |

其中，，，是用户i的所有评分组成的列向量。代表前文提到的置信度参数。

|  |  |
| --- | --- |
|  |  |
|  |  |

算法进行预测的步骤公式（3-26）所示，取近似后得到形如（3-27）所示公式。

|  |  |
| --- | --- |
|  |  |
|  |  |

### CDL算法的伪码

|  |
| --- |
| CDL算法伪代码 |
| 1 按照图模型初始化算法模型  2 配置相关参数与超参数  3分别的对算法进行EM步骤，逐渐收敛 |

## 协同深度学习（CDL）算法的实验分析

### 实验环境

本章所用环境为惠普工作站ML310e v3，CPU为Intel Xeon E3-1220 3.10GHz，内存4G，硬盘500G。系统为Ubuntu 14.04.1 64位，JDK版本为1.8.0\_60，Scala版本为2.11.8，Python版本2.7.6，MXNet版本为0.9.1。

### 数据集描述（NetFlix、CiteULike）

我们使用三个数据集进行试验，分别是，和。前两个数据集来自[[16](#_ENREF_16), [30](#_ENREF_30), [42](#_ENREF_42)]，分别独立地取自[[7]](#footnote-8)。是一个文献标记网站，允许用户创建个性化的个人图书文献库。当一篇文献被某用户选中，在该用户的个人图书馆中，将会包含文献的摘要、题目和关键词等信息。本文所使用数据集为收集并过滤后的结果，其中数据集作者手动的选择了273个种子标签，并且收集了所有包含这些标签的文献。但其中的用户数据有删选和过滤，当一位用户在个人图书馆中收藏的文献低于三篇时，则不被数据集所包括。

筛选后的结果各有不同，包括了5551名用户和16980篇文献作为项目推荐集；包括了7947名用户和25975篇文献作为推荐集。实验中，当用户u收藏了文献v，则将pair(u,v)置1，类似于评分矩阵中的“喜欢”对应分值。在用户-项目评分对上，包含有210537个收藏操作，亦即数据集转换为评分矩阵后，数据稀疏度为0.218%；包含了142807个收藏，对应的可以计算其评分矩阵中仅含有0.065%的数据。

最后一个数据集，来自于NetFlix Prize，为奈飞公司提供的推荐大奖赛数据集，数据集中包括了用户对电影的评分。实验中，为与前两个数据集起点一致，本文仿照[[43](#_ENREF_43)]的方法将用户评分转换为隐式反馈操作。本文仅考虑正向积极反馈，即评分为5分的用户-项目对。在剔除积极评价次数低于3次的用户和没有得到内容信息的电影后，我们保留了407261名用户和9228部电影。在此条件下，我们的数据集包含了15348808个用户评分，同前两者计算数据稀疏度为0.408%。

### 实验结果分析

* + - 1. 深度学习模型与传统推荐结果对比

在本目实验中，由于传统机器学习并未针对Top-N推荐进行优化。为确保量纲一致，我们利用平均绝对误差（Mean Absolute Error，MAE）和平均根方差差（Root Mean Squared Error，RSME）作为衡量算法效果的度量值。

其中平均绝对值误差MAE的计算如公式（3-28）所示。

|  |  |
| --- | --- |
|  |  |

RMSE的计算由公式（3-29）给出。

|  |  |
| --- | --- |
|  |  |

相应的，在由自动编码机做推荐时，我们分别的使用了用户和项目作为基准推荐维度。以此更好的训练神经网络模型。本文使用表示基于用户的自动编码机推荐，表示基于项目的DAE推荐。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1. 推荐结果对比 | | | | | |
|  | NMF | PMF | DAEu | DAEi |
| MAE | 0.7891 | 0.8292 | 0.7970 | 0.7666 |
| RMSE | 1.0198 | 1.0097 | 0.9926 | 0.9656 |
| TIME(s) | 6.6 | 108.4 | 42.6 | 12.4 |

表3-3中数据表明，自动编码机做数据填充，应用于个性化推荐，表现良好。在推荐系统中的表现优于传统模型。且模型训练的时间表现也优于许多模型。神经网络对数据提取的抽样特征，提高了推荐系统的算法表现，降低了数据中冗余信息的比例，剔除了数据中无用的隐性联系，对推荐系统表现有一定的提升。

* + - 1. 针对自动编码机中不同结构的结果分析

自动编码机的网络规模由人为设定，一般与数据规模相关。但其网络激活函数与数据无关，仅作为人为设计网络结构时规定。故而在试验中我们分析了不同网络组件下的自编码机性能。

网络激活函数配置如表3-4所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 1. 网络激活函数配置 | | | |
|  | 隐含层 | 输出层 | 损失函数 |
| M1 | Identity | Identity | Square |
| M2 | Identity | Sigmoid | Logistic |
| M3 | Sigmoid | Identity | Square |
| M4 | Sigmoid | Sigmoid | Logistic |

其中线性映射函数配置为恒等函数。此时，网络中的每一层网络退化为线性回归模型。当映射函数为Sigmoid时，单层网络实际上是逻辑回归模型，只是各结点对应模型的参数不同。公式（2-1）给出了逻辑回归所使用的Sigmoid映射。SquareLoss可根据前文定义自编码重构误差函数求得。

反正切映射函数Tanh类似于Sigmoid。相比逻辑回归映射，其变化幅度更巨大，值域范围也扩展至（-1，1）。作用于Sigmoid类似，其非线性映射函数的特性，被用于表示神经元的激活与否。

本目试验中，使用针对Top-N推荐优化的评估指标。平均期望准确率（Mean Average Precision，MAP）和召回率（Recall）。

平均期望准确率的定义分为两部分，分别为准确率和准确率均值。定义由公式（3-30）（3-31）给出。计算得到准确率均值后，对所有用户的AP@N计算均值可得MAP。

|  |  |
| --- | --- |
|  |  |
|  |  |

公式（3-30）给出了本文中召回率的定义。

|  |  |
| --- | --- |
|  |  |

公式中为在Top-N推荐中，给出的推荐结果；为在测试集中用户对项目的采纳情况，表示用户在评分中给出正向反馈的项目集合。表示一个指示器函数，当项目在Top-k中被采纳则为1，否则为0。

在NetFlix数据集上，不同网络结构计算推荐结果如表3-5所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 1. Top-N推荐中不同结构DAE表现 | | | |
|  | MAP@1 | MAP@5 | MAP@10 |
| M1 | 0.1301 | 0.0695 | 0.0571 |
| M2 | 0.2608 | 0.1431 | 0.1199 |
| M3 | 0.2000 | 0.1162 | 0.1011 |
| M4 | 0.2474 | 0.1370 | 0.1143 |

可以看出非线性映射函数极大的提高了模型推荐表现。均衡考虑效果和实现难度，本文随后的实验将基于Sigmoid作为网络神经元的激活函数。

图中同时展示了在对数据加入噪声方面，本文实验了加噪比例分别为0%，20%，40%，60%，80%，100%时，不同水平下模型Autoencoder的表现。实现表明当噪声比例在40%左右时，模型表现最佳。

当网络结构固定时，我们探索了不同隐层维度与最终效果的关系。如图（）所示。M4

由实验数据可以看出，在一定范围内，网络表现与隐层维度规模相关，但到达一定程度后，性能不再提升。综合考虑性能与效率，本文选择50作为网络隐层规模参数。

不同Top-N的N的取值，对召回的影响。（要剔除无关模型）

有实验数据得出，无论在MAP和Recall，神经网络表现均优于传统模型。

此外，我们对不同层数的堆叠自编码机进行了对比，实验显示在推荐数据集下，堆叠网络表现并未明显优于3层网络。

* + - 1. CDL实验结果

基于SDAE的实验结果，本目使用CDL进行推荐，结果如图（）。在MAP和召回率上，CDL有着明显优势。

|  |  |  |  |
| --- | --- | --- | --- |
| 1. MAP@500 for three datasets of CDL in Top-N Recommendation 2. Top-N推荐中CDL不同数据集表现 | | | |
|  | CiteULike-a | CiteULike-t | Netflix |
| CDL | 0.0514 | 0.0453 | 0.0312 |
| M2 | --- | --- | 0.0223 |
| M4 | --- | --- | 0.0167 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1. Recall@300 for three datasets of CDL(%) 2. Top-N推荐中CDL不同数据集表现 | | | |
|  | CiteULike-a | CiteULike-t | Netflix |
| CDL-1 | 27.89 | 31.06 | 30.70 |
| CDL-2 | 32.58 | 34.67 | 35.48 |
| CDL-3 | 29.20 | 30.50 | 31.01 |

## 本章小结

本章详细描述了协同深度过滤算法的基础、构造、优化方法。详述了算法思路与流程。并且进行了多组实验，不仅对比了神经网络中自编码机在推荐系统中的应用，也对比了不同结构的自动编码机推荐结果性能表现。最后通过DAE与CDL在推荐召回结果上对比，验证了CDL在推荐系统上具有良好表现。

1. 协同深度推荐模型的并行化

## 引言（动机）

前文已经介绍了协同深度过滤模型，也取得了较好的算法表现。但注意到训练过程中算法对内存消耗较大，且运行时间较长。为应对越来越庞大的数据，本文提出了将CDL迁移至分布式平台的思路，对其进行适当修改进行并行化。

## 机器学习的并行加速

机器学习作为人工智能的主体与核心，各种机器学习算法已经应用到各个领域，如语音识别、图片分类、推荐系统、手写输入识别、漏洞发现等领域。极大提高了计算机智能化程度和各种系统的性能。

随着大数据时代的来临，处理海量数据，建立大规模计算模型成为必然需求。现有数据种类繁多，许多数据有非常稀疏的特点，比如推荐系统关系矩阵、文本词袋模型等。在实际应用场景下，数据量已然达到快速处理有困难的程度，而当数据稀疏度上升，数据量和机器学习算法的计算量更随之成几何级上升。故而海量数据场景下的算法并行变得十分必要。本文提出了一种基于Spark的机器学习框架，利用Spark分布式内存计算的固有优势高效处理海量数据。

### 机器学习算法并行现状

随着数据量呈井喷式增长，数据复杂度越来越复杂，数据多样性日渐丰富，客观上对机器学习算法的性能与可扩展性提出了更高的要求。目前主流的机器学习并行算法主要分为三大类，分别是基于数据并行的机器学习、基于模型并行的机器学习和混合模式下的机器学习算法。

### 基于数据并行的机器学习

基于数据并行的机器学习算法主要思想为算法模型基本不做改动，将数据进行切分，分别使用部分数据训练多个模型实例，对多个分片的数据并行训练，最后将多个模型进行融合达到充分提升算法性能的目的。基于数据并行的思想实现简单且可控，可扩展性较好，但需要机器学习算法的模型具有可融合性或可叠加性。而且数据层面需要数据满足独立同分布，以及数据倾斜可控。基本结构如图4-1所示。

|  |
| --- |
|  |
| 1. The Frame of Data Parallel 2. 数据并行框架图[[44](#_ENREF_44)] |

 来帮助完成。即模型的变化量需要汇报，由负责更新为最新的模型，然后再将最新的模型分发给训练程序，以便从新的起点开始训练。

 数据并行有同步模式和异步模式之分。同步模式中，所有训练程序同时训练一个批次的训练数据，完成后经过同步，再同时交换参数。参数交换完成后所有的训练程序就有了共同的新模型作为起点，再训练下一个批次。而异步模式中，训练程序完成一个批次的训练数据，立即和参数服务器交换参数，不考虑其他训练程序的状态。异步模式中一个训练程序的最新结果不会立刻体现在其他训练程序中，直到他们进行下次参数交换[[8]](#footnote-9)。

参数服务器只是一个逻辑上的概念，不一定部署为独立的一台服务器。有时候它会附属在某一个训练程序上，有时也会将参数服务器按照模型划分为不同的分片，分别部署。

### 基于模型并行的机器学习

基于模型并行的机器学习算法主要思想为将算法模型进行拆分，由几个训练单元分别持有，分别计算模型的不同部分。当一部分算法的输入依赖于另一部分的输出时，产生通信开销。训练完成后，对模型进行组装、融合等操作得到最终模型。基于模型并行的思想实现难度相对较大，且需要模型组件独立可分互不影响或组件部分间的可计算性。但相比数据并行，模型并行对模型拆分，所需计算开销可控且精简，可以充分利用训练数据以及模型期望较佳等特性。

|  |
| --- |
|  |
| 1. 模型并行框架图[[44](#_ENREF_44)] |

多数情况下，模型并行带来的通信开销和同步消耗超过数据并行，因此加速比也不及数据并行。 但对于单机内存无法容纳的大模型来说，模型并行是一个很好的选择。

### 混合模式下的机器学习

令人遗憾的是，数据并行和模型并行都不能无限扩展。数据并行的训练程序太多时，不得不减小学习率，以保证训练过程的平稳；模型并行的分片太多时，神经元输出值的交换量会急剧增加，效率大幅下降。

混合模式下的机器学习算法综合了前两者，同时进行模型并行和数据并行。混合模式下，系统分别将数据与模型进行拆分，不仅拥有极强的可扩展性，而且对模型体积拥有控制能力，能够充分利用现有计算资源，且得到的模型期望可预测。但实现难度大，并行过程中通信开销大，模型学习过程中易出现不稳定等缺点。

因此，混合模型也是一种常见的方案。如图4-2所示，4个GPU分为两组，GPU0，1为一组模型并行，GPU2，3为另一组，每组模型并行在计算过程中交换输出值和残差。两组GPU之间形成数据并行，Mini-batch结束后交换模型权重，考虑到模型的蓝色部分由GPU0和GPU2持有，而黄色部分由GPU1和GPU3持有，因此只有同色的GPU之间需要交换权重。

|  |
| --- |
|  |
| 1. 混合并行示意图[并行概述:17] |

### 单机并行与多机并行

目前算法并行加速可以使用单机也可以使用多机集群。两者各有优劣。

1. * + 1. 单机并行

一般单机多卡加速是应对数据量不是特别巨大，算法主要开销在数据计算上的应用场景。单机并行可以充分利用CPU或GPU资源，最大限度发挥单机计算性能，在可承受容量下，加速算法训练。单机处理最大的优势在于调度与通信开销，相比多机网络通信，单机通信交互效率能有数量级上的优势。但单机本身扩展容量有限，且当性能提升到一定程度后成本将会迅速上升。

* + - 1. 多机并行

为应对单机无法承受的海量数据以及由数据庞大带来的巨大计算开销，业界常通过搭建计算集群作为解决方案。其优势在于利用分布式计算集群的丰富的计算容量，海量储存空间，将算法模型改造为具有可分布式存储、参数利用通信交互特性的分布式算法模型，进而实现快速训练算法模型的需求。相对的，集群稳定性不如单机，网络交互性能也相对较差，并在集群管理调度上实现难度较大。因此分布式模型训练往往应用于单机无法容纳的模型或单机无法再可接受时间内运算出结果的模型场景。Andrew Ng和Jeff Dean在Google用上千台计算机，完成了超大规模的深度神经网络训练并将其应用到实际场景取得了良好的表现[[44](#_ENREF_44)]。

 结合高性能计算卡和集群计算技术，构建高性能浮点运算集群正在成为加速大规模算法模型训练的有效解决方案。高性能计算集群搭建在分布式计算系统之上，采用万兆网卡或无限带宽（Infiniband）等更加高效的通信设备，以及树形拓扑等更加高效可控的网络拓扑结构，结合GPU超强计算性能搭建。在发挥出单节点较高计算能力的基础上，充分挖掘集群中多台服务器的协同计算能力，加速海量数据下大规模算法模型训练的任务。

## 协同深度学习（CDL）并行

尽管前文数据量无法称得上海量，但由单机来进行训练，依然很吃力。单机计算资源将在当代将越来越随着数据量的增长而成为算法应用瓶颈。

### 数据并行训练流程

数据并行中，本文将CDL进行模块化封装，将数据进行切片，每个分布式节点维护一个完整的模型系统，在模型分别收敛后，对模型进行融合。

神经网络具有性质：“相同初始值的神经网络，对同分布下不同数据样本进行拟合，最终所拟合数据分布相同，即得到网络参数分布相近。”本文以此为基础，在初始化时保证模型配置完全一致，同分布下的数据的不同分片进行模型训练。由调度管理结点负责将同样的初始化策略和参数配置进行分发，得到相同的模型参数，随后利用结点各自分得的数据训练，待模型收敛后，对参数进行取平均融合。得到最终模型。

数据模型具有实现容易，配置简单，容易维护的特点。但进行模型融合时，需要保证每个节点的数据足够达到模型收敛或者近似收敛，同时模型的数据分布相差也不应过大。进而可以保证模型饱和，充分训练。

### 模型并行训练流程

相比较模型并行，本文提出了将PMF和SDAE分开训练的思路，虽然算法的一致性有一定的损失，但在模型体积庞大的情况下进行神经网络的训练和矩阵分解，能够使得一定计算容量下，模型资源充分使用计算资源。

借鉴小批量梯度下降（Mini-Batch Gradient Decent）的思想，我们将SDAE与PMF分开训练，按小批量进行EM，而不是一步一调整。使用一批数据，只进行PMF的训练，调整权值。当一批数据训练完成后，固定PMF参数，仍使用当前数据对SDAE进行训练和学习。这样降低了数据对内存的开销，同时降低了IO次数。虽然对计算精度有一定影响，但提升了模型训练效率。

## 协同深度学习（CDL）多机分布式并行

### 数据分布式存储与数据并行

数据分布式储存于集群HDFS上，由Spark直接读取。集群并行模式由于集群实际环境未知且实现难度较大，未采用模型并行模式，本文采用了数据并行的协同深度学习并在集群环境下对其进行并行训练。是对现有协同深度学习针对稀疏数据的修改后，将其迁移到Spark计算平台上的成果。

系统的结构包括一个负责分发job、收集数据、调度任务的manager，以及若干个负责具体计算的worker。系统结构如图4-4所示。

|  |
| --- |
|  |
| 1. The Frame of HypridParallel 2. 混合并行示意图 |

其中，本系统于分布式系统的各计算节点初始化同样的神经网络。每计算一定量的数据，对模型的参数进行收集归并，不断迭代得到最终权值。

此外，本系统针对自动编码机低效处理稀疏数据的缺陷，提出避免计算过程中无效计算和无效存储开销的思路。使用指示器函数对算法进行改进。指示器函数形如：

|  |  |
| --- | --- |
|  |  |

通过指示器函数的选择，在计算中，系统只关注有效值及其对应参数，训练时有效避免无效计算与操作。在损失一部分模型推荐精度的前提下，模型训练速度得到提高。

### 分布式并行可扩展性分析

## 并行化CDL的实验分析

### 实验环境

集群采用新浪微博公司开发集群，拥有1500台结点机，受限于YARN配置，可使用内存为7GB，Spark版本2.0.1。框架代码版本0.6.0。

集群采用中国科学院计算技术研究所智能信息处理重点实验室计算平台，拥有6台结点机，任务提交配置为48个Core，Driver-memory为30GB，Excutor-memory为30GB，数据由HDFS读入。

### 数据准备

为使数据达到海量级别，本文在用户维度上进行了复制，对数据进行扩展，进而得到海量数据。值得注意的是，在数据并行的思路中，项目的数目直接影响到模型的规模。故而在当前实验环境下，本文只探讨当用户量海量时，集群对海量数据处理的效率。

### 单机试验结果分析

本文算法基于Spark和Scala实现，由于Spark本身具有数据的落盘操作，可以对海量数据的运算及储存进行优化，且前文试验中，单机运行算法进行训练及预测推荐耗时冗长。故而本节进行了单机计算资源下的试验性能分析，比较纯语言线性库的实现和基于Spark分布式平台优化的实现在计算资源有限的情况下的优劣。

|  |
| --- |
|  |
| 1. Spark单机运行效果对比 |

实验证明了基于Spark实现的算法在精度上略有损失，推荐精度、评分预测等维度上表现相较于高精度的单机版实现有2~5%的精度损失，在业界误差定义中，此范围的损失是可以接受的。在速度上，Spark平台优化表现出优势，在同样的数据量和同等资源开销的情况下，Spark相对单机算法有着1.3倍的速度提升。而在模型扩展性方面，相较于单机版算法出现的内存溢出、计算时间指数级上涨，基于Spark的算法有着稳定的时间开销。在单机资源应用中，Spark充分利用磁盘操作在损失一定计算时间的代价下极大的扩展了可应对的数据规模。实验证明基于Spark的算法在固定资源条件下，相较于单机算法，可处理的数据量维度上有着强势提升，极大地提高了单机计算可提供的计算能力。

### 并行试验结果分析

前文进行了单机条件下，Spark并行算法和单机算法性能对比。实验证明了在单机条件下，基于Spark的并行算法性能已经远超过单机朴素算法。为充分验证并行算法在数据规模可扩展性的能力，本文进行了Spark集群条件下的实验。同时与单机算法、单机Spark算法和不同集群规模下的算法表现进行了对比。实验结果见图4-6。

在实验中从数据中进行抽样或复制确保对数据规模进行控制，保证在实验过程中在同样数据条件下，测试不同的集群规模。由图4-6看到在数据量较小时，并行算法时间开销相对较大，变化趋势与数据量并无明显联系，这是因为数据量较小时，分布式计算中平台通讯开销所占系统时间开销比例较大。

|  |
| --- |
|  |
| 1. 单机算法和并行算法性能对比 |

随着数据量的不断增大，可以看出到当数据量达到60000条时，系统时间开销开始有明显的增长，此时系统的计算时间占比逐渐变大，计算时间的变化对系统总时间开销的影响开始显现出来。与单机算法对比，首先分布式的算法能够处理的数据量上限远远大于单机算法；其次由图线对比得出，在数据量逐渐增大的过程中，系统时间开销随数据量增大而线性增长，相比SAE二次增长的时间曲线，具有明显优势。

## 本章小结

本章将协同深度学习迁移到分布式平台下进行并行训练。首先分析了算法并行化的动机与背景；其次简述了当前机器学习算法的并行现状；随后简述了主流算法并行的思想，并针对性阐述了单机并行与分布式并行。紧接着，本文提出了针对协同深度学习CDL的并行化改进，并且在计算中对网络做了针对数据稀疏性的优化操作。最后在不同的集群配置下，验证了CDL并行能够提高模型的可扩展性，能够应对海量数据场景下的推荐任务。此外还对分布式并行可扩展性进行了分析，不同方式的并行将达到不同的实验结果。

1. 基于Spark的分布式机器学习框架

## 引言

数据挖掘与分析与大数据的结合已经成为无可争议的时代趋势，虽然Spark平台提供了MLLib算法库和丰富的大数据处理API，但实际情况下的数据仍然不能直接满足算法模型训练的额需求，数据预处理、特征提取等流程十分繁杂。而不同场景下的应用往往意味着完全不同的处理逻辑与需求，致使需要开发人员进行大量低效且重复的编码工作。

在微博研发中心算法平台部的协助下，本文介绍了一种基于Spark的机器学习平台旨在为广大开发者（包括但不限于数据分析师、数据科学家、算法工程师）提供统一、易用、高效、完善的机器学习相关功能和服务。针对常见的机器学习流程与多种复杂操作，进行了模块化开发。提高了算法训练效率、简化了数据训练的流程逻辑。

当前框架依托于Spark分布式平台，基于Scala反射实现了统一灵活的机器学习框架。框架内包含了多种数据处理逻辑、多种算法模型、多种算法组合方式、调参策略和模型可解释性导出等逻辑，并可通过配置文件对相关逻辑进行定制，以及协同分布式计算平台其他框架进行协同化操作。基于Spark的机器学习pipeline框架

## 框架建立动机

机器学习已经越来越多的深入业界的每个角落，在当前信息技术的发展中起了极大的助推作用。然而机器学习算法相对传统应用程序计算量大、迭代次数多、逻辑复杂且多变，在数据量井喷式发展的当代，机器学习算法的并行处理成为一种高效的应对方案。而数据量逐渐膨胀到单机无法承受的地步时，分布式计算不可避免的进入了人们的视野，并迅速成为大数据信息智能的核心技术手段。

在Spark并不成熟的时期，为了应对大数据场景，业务逻辑由Hadoop分布式数据处理平台完成，得到处理完成的数据，交由特定的高性能计算机进行模型训练与算法优化。随后将特定模型文件部署线上由算法进行业务干预。Hadoop平台吞吐量大，但响应效率低下，且相对于复杂的数据处理逻辑和机器学习大量迭代场景略显吃力。并且多个处理流程基于多种大数据平台，需要人工参与，导致效率低下且对异常控制的减弱。

随着Apache Spark大数据分布式平台的不断成熟，其有向无环图带来的数据处理效率的提升、高可扩展性、基于内存的分布式计算和丰富的组件库等特性正在迅速成为大数据处理的综合平台。基于丰富的开发接口与方法以及Scala语言特性和灵活的开发范式，本文实现了一种基于Spark的机器学习框架，将多种业务逻辑统一在有限的几个入口中，对多个机器学习算法进行封装统一化操作，使得由数据到训练样本，再到模型训练，最后到可读模型保存的一系列流程均可通过配置文件完成，简化了机器学习应用场景的业务逻辑，提高了使用效率。

## 已有算法模型简述

2. 线性回归

线性回归（Linear Regression），线性最小二乘法是回归问题中最常见的范式，作为一种线性方法可以被定义为

|  |  |
| --- | --- |
|  |  |

同时损失函数采用平方损失函数：

|  |  |
| --- | --- |
|  |  |

不同回归方法是通过不同的正则项来进行区分的。在没有正则项的情况下是普通最小二乘法或者线性最小二乘法回归，L1正则以为这Lasso回归，而L2正则项则代表着岭回归。所有的这些方法都是通过均方差来作为损失函数的。

框架中采用随机梯度下降对线性回归模型进行优化，从而可以拟合简单的有标签数据，进行监督学习。

1. 基于L-BFGS的逻辑回归

逻辑回归作为最基础也最常用的机器学习算法，由于其实现简单，效果良好，被广泛应用于各种环境下。如果将逻辑回归函数定义为逻辑损失，则逻辑回归算法同样可以被归类为线性模型。

框架中包含了多种优化方法的逻辑回归算法，其中L-BFGS作为一种拟牛顿法被Spark所采用，通过求解海森矩阵对算法模型进行优化和计算。

逻辑回归对数据拟合有良好的表现，更多的我们利用逻辑回归对数据进行拟合后，去除其分类阈值，通过算法学习到的参数对不同维度进行加权，供其他模型使用，取得了良好的数据表现。

1. 基于SGD的逻辑回归

当数据量巨大，数据表现复杂的情况下，对逻辑回归模型求解将导致巨大的计算开销。为了提高框架算法的工作范围，我们引入了基于随机梯度下降（Stochastic Gradient Descent, SGD）的逻辑回归算法。

尽管基于SGD的逻辑回归效果相比L-BFGS较差，且模型收敛速度慢，但在计算资源有限或者数据量极大的情况下，仍然有着极佳的表现。框架中逻辑回归默认使用了L2正则化。

1. 基于Tron的逻辑回归

置信域算法（Trust Region Methods, Tron）是一种求解非线性优化问题的主流方法，具有效率高，可靠性强，收敛稳定等特性。不同于梯度下降方法的寻找梯度，Tron在进行参数更新前优先计算合适的试探步长（Trial Step），随后根据双折线法求解优化方向。

框架引入基于Tron的逻辑回归算法，是在当数据量不是特别巨大时，提升模型训练效率和结果。

1. 基于XGBoost的逻辑回归

框架整合了DMLC发布的XGBoost算法库，并实现了基于XGBoost的逻辑回归算法，可以在GPU上进行计算，同时具有在Yarn集群资源管理失效时，仍然利用集群计算资源进行模型求解。

1. 朴素贝叶斯

朴素贝叶斯（Naive Bayes）是一种基于数据各个特征间相互独立的简单多分类算法。算法通过统计所有分类标签下的条件概率分布，利用贝叶斯公式计算各维度下数据标签的条件概率分布，随后利用观测数据的维度与其条件概率分布计算数据所对应的分类标签的概率，并通过此概率值进行分类预测。故而朴素贝叶斯的训练十分高效，只需要对数据集进行一次迭代即可完成模型的训练。其中为了提高算法健壮性和正确性，对算法加入了平滑参数。

1. 频繁模式挖掘

频繁模式挖掘（Frequent Pattern Mining）作为数据挖掘算法中极重要的一类算法，应用极度广泛，多年活跃于数据挖掘的研究中。它可以挖掘频繁项目、频繁项集和频繁子序列或者其他的频繁子集。框架在应对关联关系或者隐式拓扑关系的场景下，其他机器学习算法并不能很好的适应需求，故而加入了频繁模式挖掘算法。

1. 隐式狄利克雷分布LDA

概率主题模型常被用来发现一个文档集中的不同主题。隐式狄利克雷分布模型-LDA，作为一个典型的概率主题模型，能够用来对文档进行主题聚类，常被用来探索发现文档主题并对文档进行聚类。此外LDA不同于其他聚类模型的一个文档智能被聚类到一个主题中，LDA允许文档具有多个主题，结果以概率形式呈现。为了进行深入的文本主题分析，框架融入了LDA算法模型。

1. 决策树

决策树（Decision Tree）作为一种根据数据构建核心为树状结构的分类器算法，能够在决策树生长过程中，根据信息熵增益或信息混淆程度来决定在特定维度上如何决策，进而能够自动地根据数据特性生成特定维度的分割节点，来对数据进行决策分析，预测和回归。在训练决策树的过程中，可以理解为不同维度特征的自动组合，进而可以求得简单的特征组合表征，在特征工程中，以GBDT的形式有着重要的应用。

1. 基于SGD的GBDT

梯度下降决策树（GBDT），以决策树作为基础模型组件，通过梯度下降的策略进行Boosting，作为经典的集合算法，能够将多棵决策树学习到的只是进行融合，进而达到更加优秀的算法表现。在GBDT能够充分学习数据中包含知识的过程中，GBDT能够对数据不同维度自动地组合，得到优良的交叉特征。在GBDT的叶子节点被广泛应用的同时，提取数据在GBDT中的决策路径，进而进行特征组合也越来越成为其重要应用场景。本框架实现了基于随机梯度下降的梯度下降决策树，训练完毕后得到模型参数，对数据根据其决策路径进行特征交叉再编码，得到基于信息熵、信息增益等计算结果的交叉特征，能够极大地提高数据特征丰富程度，为下一步算法模型的性能提升奠定了坚实基础。

## 数据处理与算法支持逻辑简述

由于Spark平台MLLib的多种算法以及自实现的算法，对数据格式有着一定的要求，而实际生产数据又多种多样。同时在优化算法表现方面，不同算法有着不同的需求定义，不同场景下的数据也有着不同的分布表现。此外生产数据往往含有噪声而导致数据不洁净，算法计算出现偏差甚至直接无法使用。故而往往实际生产数据需要经过一系列的处理才能作为训练样本进行算法训练求解。基于此框架整合了多种数据处理功能，并且加入了一系列提升框架鲁棒性的支撑模块。

* 1. 数据离散化

数据离散化在简化模型运算，提升模型表现上有着重要应用，框架中对原始数据进行离散化处理，生成基于LibSVM格式的训练样本输入给算法进行训练，包含了单点映射编码（One-hot-encoding）、数据分段（PieceWise）、最大值截取（Max-Interception encoding）等多种离散手段。

单点映射离散编码（One-Hot-Encoding）作为业界常用的数据特征计算方法，是将原始数据的一个维度通过一系列计算映射到多个维度中，目前框架支持对数离散化（以2为底；以10为底；自然对数等）、基本运算离散化（包括加减乘除、取余等）、自定义公式离散化等一系列单点映射操作。例如当某条数据中的一个维度为17983146，将其取以10为底的对数后结果为8，离散化后的表示为00000001。

数据分段编码常用于无法直接进行计算的特征。通过数据分段映射，实现特征的离散化。例如某条数据一个维度取四季变化春夏秋冬，当值为夏时，可将其表示为0100。

最大值截取，可以防止数据异常点对模型的负影响，可通过配置维度取值最大值来将所有大于最大值的数据设定为最大值，从而保证模型参数的稳定，方差可控。在异常检测领用有较广泛的应用。

数据离散化可以将数据拆分为更加精细的表示，将一系列的无法计算的特征进行预处理，提高算法效率与精度。

* 1. 数据定比例抽样

在许多场景下，数据的正负样本分布并不均衡。业界常采用对样本增加权重或者对数据按一定比例采样来实现平衡数据分布的目的。此时，对数据的同分布采样将变得至关重要。

框架通过对训练数据正负样本进行统计，利用分布式WordCount的思想，将数据特征作为Key，统计数据的倾斜程度和样本比例。进一步按照统计所得比例对数据进行采样，保证采样所得训练样本与原始数据同分布，进而降低算法模型受数据倾斜的影响。

此外对数据进行采样，降低了数据样本的数目，一定程度上提升了模型训练速度。

* 1. 样本生成与特征交叉

框架在对数据进行一系列预处理后，在训练算法模型前需要将数据整理成为训练样本，使得算法能够接受。框架使用业界常用的LibSVM格式，在对数据进行抽样、离散化、降噪、截取后，将特征空间下表示的数据整理成LibSVM格式，进行算法训练。此功能单独抽取后，通过配置文件进行配置，实现可定制。

此外在某种特殊的场景下，不同维度间需要计算联合表示或者交叉维度，框架提供了基于特征空间的特征交叉，基于分布式平台可以提供上亿维度的特征交叉计算，随后整理为训练样本。

* 1. 模型训练交叉验证与自动化调参

Spark MLLib提供了算法模型的自动化评估功能，框架引入其对算法模型进行评估。在配置文件中指定待测试的模型参数与评估指标，框架将自动化的对模型训练结果进行评估，得到最佳表现模型，并给出用户定义指标下最佳模型的表现和当前模型训练所使用的参数组合。

* 1. 训练模型可解释版本保存

Spark MLLib自带的模型保存接口，保存格式为Apache Parquet格式。尽管效率极高且容易维护，但对用户不友好，在不依赖相关依赖库的情况下解析复杂，对模型分析造成困扰。基于此，框架摒弃了使用Spark高度抽象的API进行模型保存，转而自定义模型文件，在算法训练模型完毕后，对模型进行拆分解读，按照特定格式进行储存。虽然开发难度更大且在存储效率上有一定损失，但得到的模型文件解析容易、人工可读，进而便于跨平台开发和模型分析。并且在脱离Spark MLLib的环境下重构模型进行预测十分容易。

* 1. 特征权重评估

在推荐重排序的应用场景下，往往需要由算法模型计算的参数进一步计算得到各维度对最终结果的影响程度，为不同维度设置权重。故而框架囊括了对算法导出模型的分析与统计，并根据业务需求支持维度自动计算，节约业务分析成本，提高机器学习流相关工作的效率。

* 1. 鲁棒性功能添加

包括但不限于：

* 样本通过多路径读取
* 数据清洗
* 样本过滤
* 自定义字段分隔符
* 特征维度不限量翻番
* 离散特征支持列表参数
* 自定义文件参数
* 卡方检验

此部分作为框架功能性代码，提供框架附加功能需求或者鲁棒性支持，亦或提高框架使用便捷性或者针对特定数据下算法表现进行性能表征的优化。作为支撑功能，不属于框架核心逻辑，在此不再赘述。

## 框架扩展性介绍

为了统一框架逻辑，我们使用Scala语言中的特质（trait）来进行操作方法上的统一。特质相当于一种特殊的抽象类，在Scala中特质支持多重继承，进而可以在不影响业务逻辑情况下引入框架统一的操作符定义。其次将算法对象进行自定义命名，利用Scala运行时反射来实现运行时多种逻辑的统一。并利用配置文件获得对自定义算法逻辑的暴露，在不破坏框架统一性操作的前提下实现新算法与框架的整合。

MLRunnable特质的声明：

trait MLRunnable {

def run（spark: SparkSession, input: AnyRef, conf: Map[String, String]）: AnyRef

}

派生自定义算法逻辑：

object myObject extends MLRunnable{

def run（spark:SparkSession, input:AnyRef, conf:Map[String,String]） = {

//my code

0.asInstanceOf[AnyRef]

}

}

算法配置操作，通过pipeline.xml配置文件读入算法配置参数，进而对算法进行定制：

<process name="taskName">

<objectname>packageName.objectName</objectname>

<args>

<arg name="key1">value1</arg>

<arg name="key2">value2</arg>

</args>

</process>

暴露算法逻辑：

在依照上面的规范完成开发后，只需将pakcageName.objectName指定在pipeline.xml的<objectname>字段中即可。

<objectname>myObjectName</objectname>

至此，自定义算法作为模块通过框架扩展性支持，在逻辑整体保持不变的情况下，融合进机器学习框架。值得指出的是，作为Spark平台的应用级框架，受限于Spark不同模块的访问权限，对Spark特定的开发功能模块，需要在同一命名空间下实现；算法操作的优化算法既可以自定义，也可以在保证命名空间可访问的情况下通过Spark平台中的optimization功能实现基于Spark原生模块的分布式高效操作。

## Spark机器学习框架下深度协同推荐算法

利用框架MLRunnable接口，很容易实现了算法模块化的添加。在完成算法逻辑的编码与测试后，我们将算法模块添加进了框架，并能够利用框架的多种特性对算法进行更好的支撑和使用。

### 协同深度学习（CDL）模块化

### 数据处理

### 模型训练

## 机器学习框架使用实例

## 本章小结

结 论

协同过滤作为一类被广泛应用推荐系统中并非常成功的方法，取得了良好的表现。但常规方法无法充分利用实际业务场景数据，仅通过单一数据源作为信息和知识来源，当数据稀疏度较低的场景下能够快速准确发现用户、项目间的内在联系与隐式分布。但当面对新入条目，在没有历史数据或历史数据稀少，信息量欠缺的情况下，算法表现则急剧下降。为应对所述局限与不足，业界引入了多种方案来应对，通过使用一些评分外信息或附加信息来提升模型表现。

当代神经网络以极强的特征提取能力和模型拟合鲁棒性，为机器学习高速发展提供了新的方向，发展出了深度学习分支。深度学习在多个领域表现出了颠覆性的优势。在数据上深度学习极强的拟合能力，为推荐系统在数据极度稀疏情况下提供了新的思路。

深度学习模型训练具有海量参数，需要漫长的训练过程去寻找模型最佳状态；此外当代数据量飞速扩张，传统推荐系统在应对海量数据已十分吃力，结合深度学习模型后，计算量更是远高于以往。在无法快速计算和模型迭代的情况下，推荐模型会出现评分矩阵以及项目内容模型的训练变得难以维护的不足，进而引发多种不可预料的问题。为应对数据量的扩张，以及模型改进引入的海量计算操作，引入分布式计算的必要性日渐凸显。

本文介绍了一种将协同深度学习实例并行化的方法，基于分布式平台，对协同深度学习（CDL）算法范式下的一种算法实例进行并行化研究。CDL通过协同训练，从多个数据源获取项目或用户信息，结合已有评分矩阵进行反馈学习，进而提高推荐系统的性能和表现。通过对算法的并行化研究，提出了一种分布式并行化的协同深度学习，极大提高了模型容量和算法可扩展性。

本文中，在研究课题的同时，介绍了课题背景与意义；简述了课题相关理论，包括协同深度学习（CDL）、自动编码机、概率矩阵分解等；总结了机器学习算法模型并行化现状，对比几种主流思想间的优势与不足；在分析基于Spark的分布式并行技术优势与不足的基础上，进行了CDL的并行化研究。

CDL作为一种算法范式，构建了一种多层的贝叶斯模型，通过联合深度模型对项目内容学习与协同过滤评分矩阵的反馈信息，实现了结合多源信息进行推荐，本文利用自动编码机及其变体，在该框架下实现一种算法实例，并在原CDL基础上对其进行了细微改进，并在真实数据集上验证其有效性。

在算法实例的基础上，本文结合Spark分布式计算平台，对算法的并行化进行了研究、实验与分析。实现了算法的并行化训练，得到一系列基于客观数据集的实验数据，通过对数据进行分析对比，本文证明了并行化研究对算法表现的提升，验证了分布式计算在推荐模型的意义，得到了高可扩展性的并行化CDL算法模型。

此外在对CDL并行化研究的过程中产生了一套基于Spark分布式平台的基于pipeline的机器学习框架，将多种算法操作统一在一种逻辑之下，为机器学习算法的使用提供了便捷，改进算法的易用性。框架整合了从源数据到样本生成，模型训练到参数调整，参数评估与模型导出等一系列数据研究流程，形成一套pipeline逻辑，自动化大多数操作，降低人为干预程度，提高算法使用效率与系统运行的鲁棒性。

本课题实验表明，CDL在推荐准确率上相比已有算法有明显的提高，在海量数据情景下有较强的可扩展能力，机器学习框架的搭建使算法使用流程更加高效与健壮。

由于算法中使用了Top-N推荐作为目标函数训练模型，未能对全局推荐优化，导致了在推荐场景的长尾序列中仍有较大改进空间，对推荐要求的评分无法准确预测。深度学习模型的训练成本较高，未来的研究将着力于提高模型训练效率方面，边缘化概率求解将是一种可行的方案。本课题中，算法仍依赖于协同过滤反馈信息训练，形成了一种不完全的冷启动（Incomplete Cold Start, ICS）场景，在面对完全冷启动（Complete Cold Start, CCS）仍有不足，同时深度学习模型的强大学习能力也并未完全展现，尚有极大的提升空间。此外本课题并行研究基于数据并行，对模型拆分进行了简单分析，尚未深入研究与实验，接下来将在模型并行和参数服务器改进方面进行下一步工作，提升并行效率，进一步提高模型的可扩展性，并尝试于多种细分领域进行应用。

1. http://spark.apache.org/ [↑](#footnote-ref-2)
2. http://spark.apache.org/mllib/ [↑](#footnote-ref-3)
3. https://github.com/intel-analytics/BigDL [↑](#footnote-ref-4)
4. https://deeplearning4j.org/ [↑](#footnote-ref-5)
5. https://github.com/yahoo/CaffeOnSpark [↑](#footnote-ref-6)
6. http://docs.scala-lang.org/overviews/reflection/overview.html [↑](#footnote-ref-7)
7. http://www.citeulike.org. [↑](#footnote-ref-8)
8. http://djt.qq.com/article/view/1245 [↑](#footnote-ref-9)