第49卷第10期 中国科学技术大学学报 Vol. 49, No. 10
2019年10月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Oct. 2019

文章编号:0253-2778(2019)10-0842-09

基于 Spark 的并行 ISOMAP 算法

石陆魁[1,2],郭林林[1],房子哲[1],张 军[1,2]

(1.河北工业大学人工智能与数据科学学院,天津 300401;2.河北省大数据计算重点实验室,天津 300401)

摘要:为了实现大数据环境下非线性高维数据的降维,提出了基于 Spark 的并行 ISOMAP 算法.在该方法中,为了快速求解大规模矩阵的特征值和特征向量,设计并实现了基于 Spark 的并行块 Davidson 方法;同时,针对大规模矩阵计算和传输困难的问题,提出了基于 RDD 分区的行块式矩阵乘法策略,该策略把每个分区中的矩阵行转换成块矩阵,行块式矩阵可不受 map 算子对 RDD 逐条计算的限制,并可以利用 Spark 中的线性代数库参与矩阵级别的运算.实验结果表明,行块式矩阵乘法策略有效提高了矩阵运算的效率,并行块 Davidson 方法能够快速求解大规模矩阵特征值和特征向量,有效提高了并行 ISOMAP 算法的性能,表明并行 ISOMAP 算法可以适应大数据环境下的降维处理.

关键词:ISOMAP;行块式矩阵;块 Davidson 方法;Spark

中图分类号:TP181 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2019.10.010

引用格式:石陆魁,郭林林,房子哲,等. 基于 Spark 的并行 ISOMAP 算法[J]. 中国科学技术大学学报,2019,49(10):842-850.

SHI Lukui, GUO Linlin, FANG Zizhe, et al. Parallel ISOMAP algorithm based on Spark[J]. Journal of University of Science and Technology of China, 2019,49(10):842-850.

Parallel ISOMAP algorithm based on Spark

SHI Lukui[1,2], GUO Linlin[1], FANG Zizhe[1], ZHANG Jun[1,2]

(1. *School of Artificial Intelligence, Hebei University of Technology, Tianjin* 300401, *China*;
2. *Hebei Province Bigdata Computation Key Laboratory, Tianjin* 300401, *China*)

Abstract: To reduce the dimension of the nonlinear high-dimensional data in the big data environment, a parallel ISOMAP algorithm based on Spark is proposed, where a Spark-based parallel block Davidson method is designed and implemented to quickly solve eigenvalues and eigenvectors of the large scale matrices. Simultaneously, a row-block matrix multiplication strategy based on RDD partition is proposed for the difficulty of computation and transmission of the large scale matrices, which converts the matrix rows in each partition into block matrices. The row-block matrices are not restricted by the map operator to RDD calculation one by one, and can treat operations at the matrix level by using linear algebraic Library in Spark. The experimental results show that the row-block matrix multiplication strategy effectively improves the efficiency of matrix operations; the parallel block Davidson method can quickly solve the eigenvalues and eigenvectors of the large scale matrices and effectively improve the performance

收稿日期:2019-05-15;**修回日期**:2019-08-26

基金项目:河北省自然科学基金(F2017202145)资助.

作者简介:石陆魁,男,1974 年生,博士/教授. 研究方向:人工智能、数据挖掘. E-mail:shilukui@scse.hebut.edu.cn

通讯作者:张军,博士/副教授. E-mail:zhangjun@scse.hebut.edu.cn

of parallel ISOMAP algorithm; and the parallel ISOMAP algorithm can adapt to dimensionality reduction in the big data environment.

Key words: ISOMAP; row-block matrix; block Davidson method; Spark

0　引言

随着互联网、物联网、电子商务、云计算等技术的发展,数据呈现爆炸式增长,并且数据的结构越来越复杂,维数越来越高.数据规模的增大和维数的增加,大大增加了数据分析处理的复杂度,耗费大量计算资源,引发"维数灾难".

为了挖掘隐藏在大数据背后的价值,必须降低大数据的复杂性,而降低数据的维数是降低大数据复杂性的主要方法之一.数据降维的目的就是在保持原始数据的分类和决策能力前提下,去掉数据中的冗余信息.流形学习方法可以发现隐藏在高维数据中的内在流形结构,降低数据的维数.等距特征映射法(isometric mapping,ISOMAP)[1]是一种典型的流形学习方法,在机器学习、模式识别、医疗等领域有着广泛应用[2-5].即便如此,面对现实世界中的大数据,串行的 ISOMAP 算法也无能为力.

并行流形学习方法是在传统流形学习方法基础上为适应大数据处理的需求而发展起来的,这些方法通常会利用一些高性能的计算平台或计算工具,并结合自身特点改进整体或部分过程,提高大规模数据集的降维效率.文献[6-7]利用 GPU 实现了并行流形学习算法,文献[8-11]基于 Hadoop 平台实现了并行流形学习算法,文献[12-13]基于 CUDA 实现了并行流形学习算法.文献[14]提出了一种大规模数据降维并行框架,在大型并行机器上实现了特征分解通用计算块.文献[15]提出了基于 Spark 的并行 ISOMAP 算法,利用并行幂法迭代特征值求解方法可以求出最大特征值.文献[16]利用梯度下降方法求解低维嵌入的方法代替 LLE 算法中大型稀疏矩阵的特征值分解,明显降低了时间复杂度和存储消耗.

特征值求解是 ISOMAP 算法的关键步骤,为了实现 ISOMAP 的并行必须实现特征值求解的并行.在多数并行流形学习算法中,并行特征值求解较多使用集成工具或接口中的特征值计算函数,大量内存耗费在大规模矩阵的存储和读取上;而且矩阵特征值求解复杂度高,当矩阵规模较大时,特征值求解耗时较多.在单机环境下,由于内存的限制可能无法求解,因此为了实现 ISOMAP 的并行必须设计分布式平台下的特征值并行求解算法.

本文提出了一种基于 Spark 的并行 ISOMAP 算法,针对大规模矩阵计算和传输困难的问题,提出了一种基于 RDD 分区的行块式矩阵乘法策略;设计并实现了基于 Spark 的并行块 Davidson[17]方法,可较快地计算出大规模矩阵的部分特征值和特征向量.实验结果表明,本文提出的并行方法可以有效地将高维大数据映射到低维空间中.

1　ISOMAP 算法介绍

ISOMAP 算法是 MDS 算法的非线性推广,能够保持数据的全局特性.ISOMAP 用测地线距离代替欧式距离,在将高维数据映射到低维空间时保持测地线距离不变.ISOMAP 算法包括 3 个步骤:构建邻域、估计测地线距离及特征值求解,详细过程可参考文献[1].

如果高维数据所在的低维流形与欧式空间中的某个子集整体等距,并且这个子集是一个凸集,那么 ISOMAP 算法会取得较好的降维结果.ISOMAP 算法得到的特征值求解矩阵是一个非稀疏矩阵,计算复杂度较高,因此串行 ISOMAP 不适合大数据环境下的降维处理.

2　行块式矩阵乘法策略

大规模矩阵运算在单机环境下执行速度较慢甚至会产生内存溢出.Spark 可以利用 RDD 存储矩阵,当 RDD 不再参与运算时,可以取消持久化,从而合理利用内存资源.由于矩阵可按行或列排列,将大规模矩阵划分成行块式矩阵,可充分利用 Spark 中的线性代数库,提高计算效率,为此提出了基于 RDD 分区的行块式矩阵乘法策略.

2.1　行块式矩阵划分

Spark 中,矩阵存储最简单直接的方式就是把 n 阶方阵从存储文件按行读取到 RDD 中,读取过程如图 1 所示.在读取过程中,把每行数据保存到向量中并按原矩阵的行号次序从 0 编号,即可转化成以 RDD 存储的＜lineID,lineVec＞(键值对)形式,记作 lineRDD,其中 lineID 为原矩阵的行号,lineVec

为原矩阵对应的行向量.使用这种行矩阵,每次只有一行数据参与向量级别的运算,效率较低.

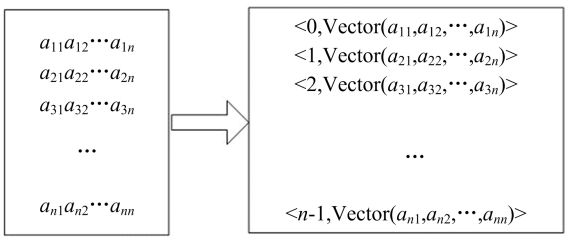

图 1 行矩阵存储

Fig. 1 The storage of matrices by rows

为了提高计算效率,设计了基于 RDD 分区(Partition)的行块式矩阵.行块式矩阵利用 RDD 分区对行矩阵进行划分,如对图 1 中的行矩阵进行转换,转换原理如图 2 所示.在实际存储行矩阵时,每个分区中会有一定数量的矩阵行,令 RDD 分区个数为 q,则每个分区的矩阵行数为 $r=n/q$(理想情况下,假设每个分区的矩阵行数相同).

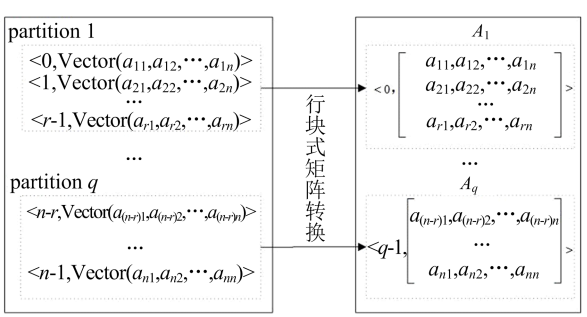

图 2 行块式矩阵转换

Fig. 2 The conversion principle of row-block matrices

从图 2 可知,行块式矩阵是把每个分区中的 r 条<lineID,lineVec>键值对转换为一条键值对,可表示为<blkID,blkMatrix>形式,记作 blkRDD,其中 blkID 为行块式矩阵的块号,与分区号保持一致,blkMatrix 为块号对应的子矩阵.

2.2 乘法策略原理

设 $\boldsymbol{A}*\boldsymbol{B}=\boldsymbol{C}$,把 n 阶方阵 $\boldsymbol{A}$ 划分成 q 个行块式矩阵,$\boldsymbol{B}$ 是 $n*k$ 阶矩阵,分别计算 $\boldsymbol{A}_i*\boldsymbol{B}=\boldsymbol{C}_i$($i=1,\cdots,q$),最后把 $\boldsymbol{C}_i$ 按块号顺序排列得到 $n*k$ 阶 $\boldsymbol{C}$ 矩阵,具体过程如图 3 所示.

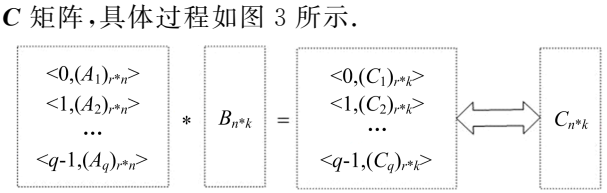

图 3 行块式矩阵乘法原理

Fig. 3 Principle of row-block matrix multiplication

上述乘法计算中,k 是比较小的整数,矩阵 $\boldsymbol{B}$ 可看作小矩阵,所以不做分块处理并不影响计算效率.根据上述行块式矩阵乘法原理,设计基于 Spark 的行块式矩阵乘法,并以包含一个 Master 节点和两个 Worker 节点的 Spark 集群为例介绍乘法的具体过程,如图 4 所示.

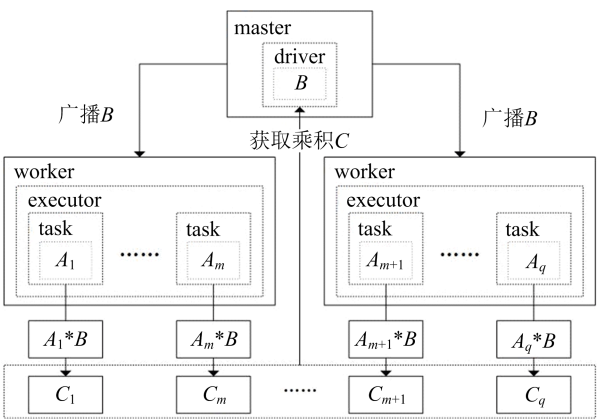

图 4 基于 Spark 的行块式矩阵乘法

Fig. 4 Principle of row-block matrix multiplication based on Spark

乘法运算时,小矩阵 $\boldsymbol{B}$ 存放于 Master 节点中,矩阵 $\boldsymbol{A}$ 存放在具有 q 个分区的 RDD 中,每个分区对应一个行块式矩阵 $\boldsymbol{A}_i$,分别存储在不同的 Worker 节点中.在执行 Spark 作业时,Worker 节点会启动若干个 Executor 进程,Master 节点下的 Driver 进程会为 Executor 进程分配 task 任务.由于 RDD 的每个分区的行块式矩阵乘法只能在一次 task 任务中执行,所以总共需要执行 q 次 task 任务.Driver 进程启动之后把矩阵 $\boldsymbol{B}$ 广播到所有 Worker 节点下,方便每个 Worker 节点中的行块式矩阵 $\boldsymbol{A}_i$ 与 $\boldsymbol{B}$ 做乘法运算,然后得到矩阵 $\boldsymbol{C}_i$.Master 节点收集所有 Worker 节点中的 $\boldsymbol{C}_i$,并按照块号顺序组合成矩阵 $\boldsymbol{C}$.当输入矩阵是行块式矩阵时,分布式矩阵和小矩阵之间的乘法可以看作两个小矩阵的乘法,将多组矩阵乘积最后组合成一个大矩阵.

3 基于块 Davidson 的并行特征值求解方法

3.1 Davidson 方法

Davidson[17]方法是一种近似特征值求解方法,它以 Rayleigh-Ritz 方法为基础,利用子空间迭代法直接求解最大特征值或最小特征值,在效率上优于一般特征值求解方法,非常适合工程应用.Davidson 方法在 Rayleigh-Ritz 方法的基础上,定义了残量 r_k

和产生新向量 t 的预处理公式，通过迭代方式获取近似特征值和特征向量. 设 $\boldsymbol{V}_k$ 为标准正交子空间，求解矩阵 $\boldsymbol{A}$ 的特征值和特征向量的 Davidson 方法具体过程如下：

(Ⅰ)计算 Rayleigh 矩阵 $\boldsymbol{H}_k=\boldsymbol{V}_k^{\mathrm{T}}\boldsymbol{A}\boldsymbol{V}_k$；

(Ⅱ)计算 $\boldsymbol{H}_k$ 的最大(或最小)特征值和对应的特征向量(μ_k,y_k)；

(Ⅲ)计算 Ritz 向量 $x_k=\boldsymbol{V}_k y_k$；

(Ⅳ)计算残量 $r_k=(\mu_k\boldsymbol{I}-\boldsymbol{A})x_k$；

(Ⅴ)进行收敛性检查，如果 $\|r_k\|<\varepsilon$，可得到目标特征值和对应的特征向量；

(Ⅵ)若不收敛，则令 $t=(\mu_k\boldsymbol{I}-\boldsymbol{D}_A)^{-1}r_k$；

(Ⅶ)正交化处理 $\boldsymbol{V}_k=\mathrm{MGS}(\boldsymbol{V}_k,t)$，继续执行上述过程.

Davidson 方法首先通过低维 Rayleight 矩阵 $\boldsymbol{H}_k$ 的特征向量估计高维矩阵 $\boldsymbol{A}$ 的特征向量，并利用预处理技术精化迭代过程产生的残量 r_k，得到新的向量 t；然后将 $\boldsymbol{V}_k$ 和 t 正交化的结果更新 $\boldsymbol{V}_k$，以增加子空间的基底数量.

3.2　块 Davidson 方法

由于 Davidson 方法计算耗费较高，于是 Sadkane 等[18]提出了基于重新开始技术的块 Davidson 方法. 该方法能够求出矩阵的多个最大特征值和特征向量，当矩阵的目标特征值分布比较密集，与非目标特征值较为接近时，也能够保持良好的收敛速度和准确率.

块 Davidson 方法把初始向量空间扩展到 l 维，每次计算出 l 个 Ritz 值 $\mu_{k,i}$ 和 Ritz 向量 $x_{k,i}$，并通过公式 $r_k=(\mu_k\boldsymbol{I}-\boldsymbol{A})x_k$ 计算 l 个残量值 $r_{k,i}$. 当不满足 $\|r_k\|<\varepsilon$ 时，利用公式 $t=(\mu_k\boldsymbol{I}-\boldsymbol{D}_A)^{-1}r_k$ 产生 l 个新的向量 $t_{k,i}$，并对 $t_{k,i}$ 和当前向量空间做正交化处理，利用得到的 l 个处理后的向量更新特征向量空间. 同时加入重新开始技术，即当特征向量空间 $\boldsymbol{V}_k$ 的列数超过 m 时，保留当前迭代的 Ritz 向量和新特征空间向量 $t_{k,i}$ 重新为 $\boldsymbol{V}_k$ 赋值. 块 Davidson 方法可以快速求解出实对称矩阵的多组特征值和特征向量，ISOMAP 算法中的矩阵也是实对称矩阵，因此可以用块 Davidson 方法进行求解.

3.3　基于 Spark 的并行块 Davidson 方法

块 Davidson 方法的主要计算过程是矩阵和矩阵或矩阵和向量的乘积，所以块 Davidson 方法本身就具有很强的并行性. 为了提高块 Davidson 方法在分布式环境下的计算效率，本文借鉴了王顺绪等[19]在微机网络环境下实现的并行块 Davidson 方法，在此基础上利用 2.2 节中的行块式矩阵乘法策略进行改进，实现了 Spark 下的并行块 Davidson 方法，称为 SPB-Davidson 方法，具体步骤如下：

(Ⅰ)实现矩阵的分布式存储：为充分利用 Spark 集群中每个节点的存储资源，把大规模矩阵 $\boldsymbol{A}$ 存储到 HDFS 中.

(Ⅱ)生成行块式矩阵 $\boldsymbol{A}_i$：从 HDFS 文件的每个分块中按行读取矩阵信息，得到分布式行矩阵并转化成基于 RDD 分区的行块式矩阵 $\boldsymbol{A}_i$.

(Ⅲ)定义特征向量空间的最大列数和初始矩阵空间：选取正整数 m 作为最大列数，随机生成初始矩阵 $\boldsymbol{V}_1=[v_1,v_2,\cdots,v_l]$ 并保存在主节点上.

(Ⅳ)计算 Rayleigh 矩阵 $\boldsymbol{H}_k$：利用行块式矩阵乘法策略，把 $\boldsymbol{V}_k$ 从主节点广播到所有计算节点，分别在每个计算节点中计算 $\boldsymbol{C}_i=\boldsymbol{A}_i\boldsymbol{V}_k$，然后把所有节点中的 $\boldsymbol{C}_i$ 按矩阵块号顺序排列，得到一个阶数较小的乘积矩阵 $\boldsymbol{C}$，最后左乘特征空间向量的转置 $\boldsymbol{V}_k^{\mathrm{T}}$.

(Ⅴ)计算 $\boldsymbol{H}_k$ 的 l 个最大特征值和对应的特征向量 $(\mu_{k,j},y_{k,j})(j=1,\cdots,l)$：由于 $\boldsymbol{H}_k$ 的阶数和 $\boldsymbol{V}_k$ 的列数相同且远远小于 n，所以可利用 Spark 中的特征值求解函数 eig 来计算 $\boldsymbol{H}_k$ 的 l 个最大特征值和对应的特征向量.

(Ⅵ)计算 Ritz 向量 $x_{k,j}$：在主节点上利用第(Ⅴ)步中求得的特征向量 $y_{k,j}$ 来计算 Ritz 向量.

(Ⅶ)计算残量 $r_{k,j}=(\mu_{k,j}\boldsymbol{I}-\boldsymbol{A}_i)x_{k,j}$：将所有的 $x_{k,j}$ 组成 $n*l$ 阶的小矩阵 $\boldsymbol{X}$ 并广播到各计算节点中，每个计算节点先独立计算 $\boldsymbol{A}_i$ 和 $\boldsymbol{X}$ 的乘积，然后主节点把所有计算节点中的乘积矩阵按块号顺序排列，得到最终的乘积矩阵，最后按列计算 l 个残量 $r_{k,j}$.

(Ⅷ)判断收敛性：在主节点上利用 $\|r_k\|<\varepsilon$ 判断收敛性，满足条件则停止迭代.

(Ⅸ)计算新特征空间向量 $t_{k,j}$：在主节点上求解方程 $\boldsymbol{C}_{k,j}t_{k,j}=r_{k,j}(j=1,\cdots,l)$.

(Ⅹ)生成新的向量空间：判断当前空间是否超过最大列数 m，如果未超过则在主节点上计算 $\boldsymbol{V}_{k+1}=\mathrm{MGS}(\boldsymbol{V}_k,t_{k,1},\cdots,t_{k,l})$；否则在主节点上采用重新开始技术为初始向量空间赋值 $\boldsymbol{V}_1=\mathrm{MGS}(x_{k,1},\cdots,x_{k,l},t_{k,1},\cdots,t_{k,l})$，最后把新生成的向量空间 $\boldsymbol{V}_{k+1}$ 或 $\boldsymbol{V}_1$ 投入第(Ⅳ)步继续迭代.

在 SPB-Davidson 方法中，如果设置分区个数是

集群节点个数的整数倍，可保证每个节点上的分区数相同，因而每个 RDD 分区对应的行块式矩阵也会最大限度地自动均衡分配到 Spark 集群的各个节点上，从而提高内存资源利用率和计算效率. 在 Spark 中执行一次任务完成一个行块式矩阵的相关操作，每次任务针对不同的行块式矩阵执行相同的代码，这种方式的优点是可以实现移动计算，即把计算任务下发到行块式矩阵所在的节点进行处理.

ISOMAP 算法进行特征映射时需要求解矩阵的 k 个最大特征值，并行块 Davidson 方法能够同时计算多个最大特征值，并能充分发挥 Spark 集群分布式内存的计算优势，把大规模对称矩阵划分成行块式小矩阵，从而提高大规模对称矩阵特征值的并行求解效率.

4 基于 Spark 的并行 ISOMAP 算法

ISOMAP 算法主要包括 3 个步骤：求 k 近邻、估计测地线距离及特征映射. ISOMAP 的并行就是这 3 个步骤的并行，算法流程如图 5 所示.

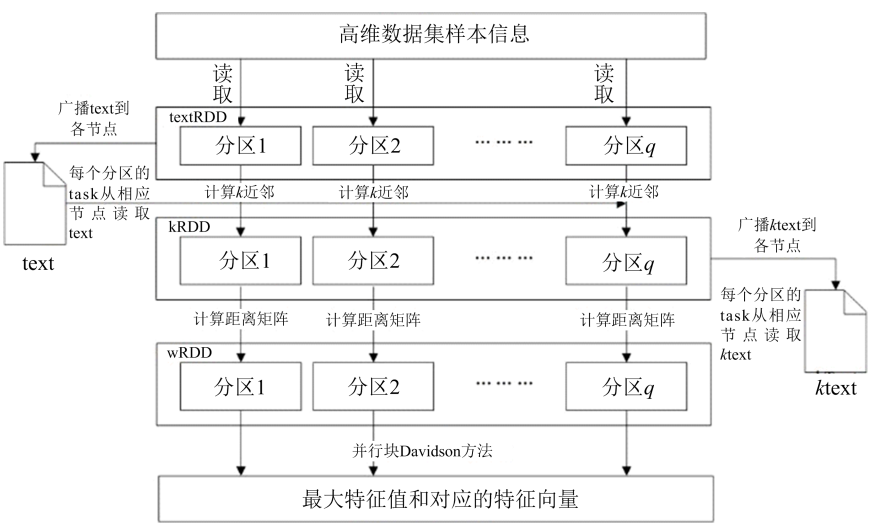

图 5 并行 ISOMAP 算法流程

Fig. 5 The flow of parallel ISOMAP

(Ⅰ)求 k 近邻

为了把高维样本信息存储到 HDFS 文件中，q 个 RDD 分区同时从文件中读取不同的数据，把高维样本信息转化为键值对信息，生成 textRDD. 主节点从 q 个分区中收集 textRDD 得到 text 并广播到所有计算节点上. 计算每个分区中的样本点和其他样本点的距离时，可先从 text 中读取相关键值对信息，然后把求得的 kRDD 近邻信息保存到 kText 中. 本文使用精确欧式位置敏感哈希算法[20]求解 k 近邻. 其核心思想是先求出每个样本的候选近邻，然后利用线性查找方法从候选近邻中查找 k 近邻. 在 Spark 上并行实现 k 近邻求解过程如下：

(a)主节点生成位置敏感哈希函数并广播到各个计算节点；

(b)利用位置敏感哈希函数，在计算节点上求出当前样本点对应的候选近邻点；

(c)在计算节点上构建当前节点的 k 个元素的大根堆完成距离排序；

(d)依次读取大根堆元素，获取 k 个近邻点.

(Ⅱ)估计测地线距离

每个 RDD 分区的任务从相应节点中读取 kText 信息，构建与相应的近邻点之间的距离矩阵 wRDD. 测地线距离用最短路径来近似，采用 Dijkstra 算法计算最短路径. 具体过程如下：

(a)把各个分区中每个样本 A 看作源点，存入集合 S 中；

(b)在源点 A 的 k 个近邻中，找出与它距离最近的样本 B，然后把 B 存入 S 中；

(c)利用 kText 的信息，获取其他点到 B 的距离，更新其他点到 A 的最短路径. 如果此距离小于 S 外任何点到 A 的距离，则存入 S；否则，将距离更小的点存入 S；

(d)继续寻找最小距离，直至所有样本进入 S 中.

(Ⅲ)获得低维嵌入

wRDD 先经过变换得到目标矩阵 gRDD，然后

利用 SPB-Davidson 方法求得最大特征值和特征向量，具体过程如下：

(a)把 wRDD 输入到 MDS 算法得到目标矩阵 gRDD；

(b)把 gRDD 转化为行块式矩阵 blkRDD；

(c)把 blkRDD 投入 SPB-Davidson 方法中，得到 k 个最大特征值和对应的特征向量.

在并行 ISOMAP 算法中，使用精确欧式位置敏感哈希算法求解 k 近邻，得到的是近似近邻，其复杂度远远低于线性搜索算法，同时具有很好的并行性. 通过利用 Dijkstra 算法计算最短路径来近似测地线距离，并行实现可以大大提高计算效率. 利用并行块 Davidson 方法可以快速求得矩阵的最大特征值和特征向量. 总之，通过并行极大地提高了 ISOMAP 算法的计算效率，能够实现较大规模数据的快速降维.

5　实验结果

5.1　实验环境与实验数据

实验使用 Hadoop 集群和 Spark 集群，为方便两种集群共用 5 个虚拟节点，每个节点的配置为：12 核 CPU，16 GB 内存. 实验数据为 Swiss Roll 数据集和 S-curve 数据集，如图 6 和 7 所示. 在实现并行算法过程中，为方便数据在 Spark 上以 RDD 键值对方式存储和运算，从 0 开始为每个样本逐个添加索引.

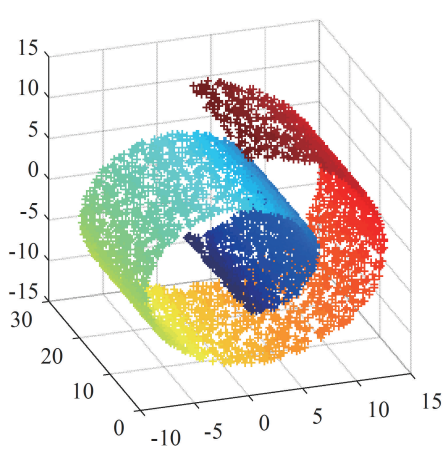

图 6　Swiss Roll 数据集

Fig. 6　Swiss Roll data

实验包括 4 部分：测试集群节点数对并行块 Davidson 方法的影响；测试行块式矩阵乘积策略对并行块 Davidson 方法的影响；并行块 Davidson 方法的性能测试；并行 ISOMAP 算法的性能测试.

根据文献[1,21]，两个数据集上选择近邻个数为 7 时，既能保证近邻图的连通性，又能取得好的嵌入结果，所以在本文的实验中也设定近邻个数为 7. 对于 RDD 分区数，如果分区数太多，执行 collect 操作时需要将分散在各节点上的数据提取到主节点上，节点间数据传输增多将会降低算法的整体效率，实验结合硬件配置兼顾并行度和算法的整体执行效率，分区数设置为 12. 分区数据分配策略为：如果数据行数能被分区数整除，则将数据平均分配到每个分区；如果不能平均分配，Spark 内部会根据分区数，保证前面 q-1 个分区的数据相同，最后一个分区保存剩余数据.

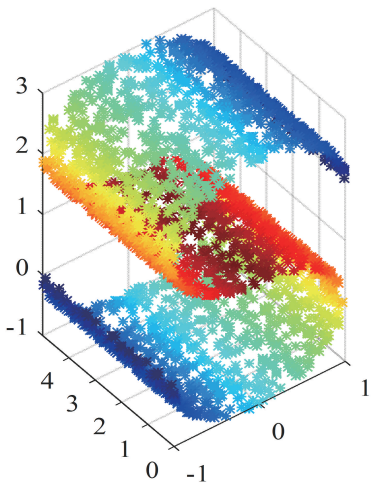

图 7　*S*-curve 数据集

Fig. 7　*S*-curve data

5.2　集群节点数对并行块 Davidson 方法的影响

为分析集群节点数对并行块 Davidson 方法的影响，分别从 Swiss Roll 和 S-curve 数据集上选取 10 000 个样本点进行测试. 实验中，节点最多为 5 个. 在两个数据集上，分别利用并行块 Davidson 方法计算部分最大特征值，统计算法执行时间，算法执行时间随集群节点数变化情况如表 1 所示.

表 1　并行块 Davison 方法执行时间随集群节点数变化

Tab. 1　Time of solving the eigenvalue for different number of nodes (s)

数据集	节点数				
	1	2	3	4	5
Swiss Roll	125	118	112	108	104
S-curve	107	102	98	96	94

由表 1 实验结果可知，对同一数据集，在相同的分区数和近邻个数条件下，随着集群节点数的增加，算法执行时间随之减少. 这是因为结点越多，每个节

点分配到的数据就越少,每个节点计算所需时间就越少,进而整体计算时间减少.

5.3 行块式矩阵乘法策略对并行块 Davidson 方法的影响

为测试行块式矩阵乘法策略对并行特征值求解方法的影响,与未使用行块式矩阵乘法策略的情况(使用行矩阵计算)进行比较.分别从 Swiss Roll 和 S-curve 数据集中选取 1 500、3 000、6 000、12 000 个样本点,利用并行块 Davidson 方法求解部分最大特征值,所用时间如表 2 和表 3 所示.实验中,集群节点数为 5.

表 2 Swiss Roll 数据上并行块 Davidson 方法分块与未分块比较(s)

Tab. 2 Comparison of blocking and unblocking on Swiss Roll dataset (s)

方法	数据量			
	1 500	3 000	6 000	12 000
未分块	15	27	156	790
分块	20	26	39	138

表 3 S-curve 数据集上并行块 Davidson 方法分块与未分块比较(s)

Tab. 3 Comparison of blocking and unblocking on S-curve dataset (s)

方法	数据量			
	1 500	3 000	6 000	12 000
未分块	19	34	165	810
分块	21	29	38	126

由表 2,3 的结果可知,数据量较小时,两种情况效率相近.随着数据增多,采用行块式策略的效率高于未采用行块式策略的,而且数据越多,效果越明显.造成这种现象的原因是,当数据量较小时,由于集群性能优势,两种方式的运行速度都比较快.数据增多后,由于未采用行块式策略计算时使用的是简单的向量-向量运算或者矩阵-向量运算,存在大量类似重复运算,导致计算消耗大大增加.行块式矩阵可以利用 Spark 中的矩阵运算库,具有较高的运算效率.总之,行块式矩阵乘法策略在一定程度上提高了并行块 Davidson 方法的效率.

5.4 并行块 Davidson 方法性能测试

为了测试并行块 Davidson 方法的性能,与单机下的块 Davidson 方法进行比较.分别从 Swiss Roll 和 S-curve 数据集中选取 1 500、3 000、6 000、12 000 个样本点,利用并行块 Davidson 方法求解部分最大特征值,所用时间如表 4 和 5 所示.实验中,集群节点数为 5.

表 4 Swiss Roll 数据集上两种环境下并行块 Davidson 方法执行时间比较(s)

Tab. 4 Comparison of run time of parallel block Davidson method on Swiss Roll dataset for two environments (s)

环境	据量				
	1 500	3 000	6 000	12 000	20 000
单机	13	35	370	内存溢出	内存溢出
Spark	20	26	39	138	276

表 5 S-curve 数据集上两种环境下并行块 Davidson 方法执行时间比较(s)

Tab. 5 Comparison of run time of parallel block Davidson method on S-curve dataset for two environments (s)

环境	数据量				
	1 500	3 000	6 000	12 000	20 000
单机	20	32	365	内存溢出	内存溢出
Spark	21	29	38	126	270

从表 4,5 的结果可以看出,数据量较小时,两种环境下计算效率相近.这是因为 Spark 集群中主节点和从节点之间的 I/O 消耗了一部分时间.随着数据量增大,单机耗时越来越长,甚至发生内存溢出,Spark 集群的优势越来越明显.导致这种结果主要有两方面因素:一是并行块 Davidson 方法本身都是基于向量空间的迭代算法,能够充分利用 Spark 的内存迭代计算优势;二是算法中使用了行块式矩阵,能够充分利用 Spark 自带的矩阵计算库,提高了计算效率.

5.5 并行 ISOMAP 算法性能测试

为了分析并行 ISOMAP 算法的整体性能,分别从 Swiss Roll 和 S-curve 数据集中选取 3 000、6 000、12 000 个样本点,在 Spark 和单机下分别执行并行 ISOMAP 算法,比较二者的执行时间,并计算数据量为 6 000 时的加速比.执行时间如表 6 和表 7 所示,加速比如图 8 和 9 所示.执行并行 ISOMAP 算法时,集群节点个数为 5.

表 6 Swiss Roll 上并行 ISOMAP 算法执行时间(s)

Tab. 6 Execution time of parallel ISOMAP on Swiss Roll for two cases (s)

环境	数据量			
	3 000	6 000	12 000	20 000
单机	86	764	内存溢出	内存溢出
Spark	55	98	345	580

表 7　*S*-curve 上并行 ISOMAP 算法执行时间(s)

Tab. 7　Execution time of parallel ISOMAP on *S*-curve for two cases (s)

环境	数据量			
	3 000	6 000	12 000	20 000
单机	68	724	内存溢出	内存溢出
Spark	58	95	315	574

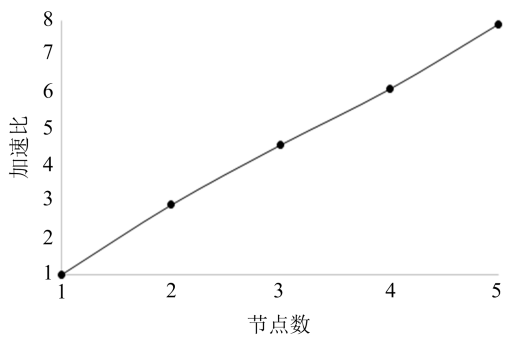

图 8　ISOMAP 在 Swiss Roll 的加速比

Fig. 8　Speedup on Swiss Roll

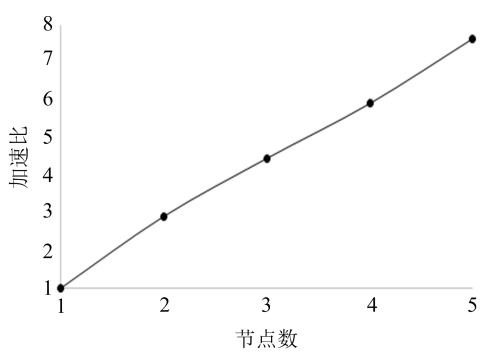

图 9　ISOMAP 在 *S*-curve 上的加速比

Fig. 9　Speedup on *S*-curve

从上述实验结果可以看出，随着数据的增多，ISOMAP 算法在两个数据集上的执行时间越来越长，而且单机上的增长率高于 Spark. 随着节点个数的增加，加速比也随之增长，并且当节点个数在 2 到 5 之间时，加速比几乎呈线性增长趋势. 这主要得益于 ISOMAP 算法 3 个步骤的并行处理，通过并行极大地提高了求解 k 近邻和估计测地线距离的效率；同时，在 Spark 上执行并行块 Davidson 方法求解特征值时，矩阵在内存中持久化，迭代时直接复用内存中的数据，提高了计算效率. 此外，在并行算法的执行过程中，利用行块式矩阵乘法策略实现分布式矩阵乘法，极大地提高了各步骤中矩阵运算的效率.

为了直观地展示并行 ISOMAP 算法的效果，分别对含有 10 000 个样本点的 Swiss Roll 和 *S*-curve 数据集进行降维处理，结果如图 10 和 11 所示.

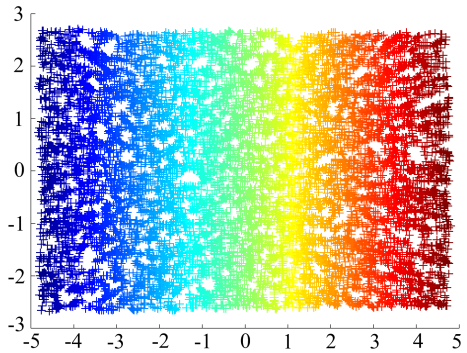

图 10　ISOMAP 在 Swiss Roll 上降维结果

Fig. 10　Results on Swiss Roll

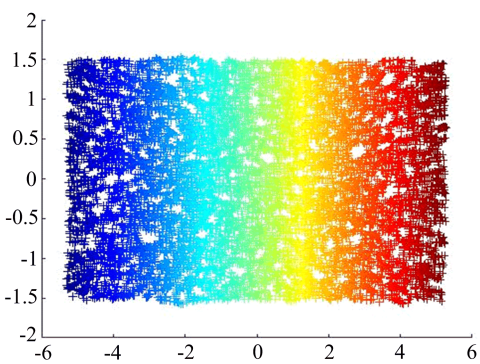

图 11　ISOMAP 在 *S*-curve 上降维结果

Fig. 11　Results on *S*-curve

从图中结果可以看出，并行 ISOMAP 算法对 Swiss Roll 和 *S*-curve 数据集降维处理后都能得到较好的低维嵌入.

6　结论

ISOMAP 算法可以发现隐藏在高维数据中的低维流形，它通过将高维数据映射到低维空间中来降低数据的复杂性. 由于其时空复杂度较高，需要求解矩阵的特征值，当数据量较大时，单机上难以运行. 于是本文提出了一种基于 Spark 的并行块 Davidson 方法，该方法可以快速地求解大规模矩阵的最大特征值；同时本文提出一种行块式矩阵乘法策略以提高矩阵乘法的效率；在此基础上进一步提出了基于 Spark 的并行 ISOMAP 算法. 实验结果表明，并行块 Davidson 方法极大地提高了大规模矩阵特征值求解效率，从而提高了并行 ISOMAP 算法的性能. 下一步的工作是将并行 ISOMAP 算法应用到实际应用中并研究多流形下 ISOMAP 算法的并行实现.

参考文献(References)

[1] TENENBAUM J B, DE SILVA V, LANGFORD J C. A global geometric framework for nonlinear dimensionality reduction[J]. Science, 2000, 290(5): 2319 - 2323.

[2] PLESS R, SOUVENIR R. A survey of manifold learning for images [J]. IPSJ Transactions on Computer Vision & Applications, 2009, 1(1):83-94.

[3] SARVENIAZI A. An actual survey of dimensionality reduction [J]. American Journal of Computational Mathematics, 2014, 4: 55-72.

[4] SINGH K P, BHAI R, MISHRA V, etal. Localization in wireless sensor network using LLE-ISOMAP algorithm [C]// Proceedings of IEEE Region 10 Conference. Penang, Malaysia: IEEE, 2017:393-397.

[5] RAJAGURU H, KUMAR PRABHAKAR S. Performance analysis of local linear embedding and Hessian LLE with hybrid ABC-PSO for epilepsy classification from EEG signals[C]//Proceedings of International Conference on Inventive Research in Computing Applications. Coimbatore, India: IEEE, 2018:1084-1088.

[6] YEH T T, CHEN T Y , CHEN Y C , et al. Efficient parallel algorithm for nonlinear dimensionality reduction on GPU[C]// Proceedings of International Conference on Granular Computing. San Jose, USA: IEEE, 2010:592-597.

[7] LI W, ZHANG L, DU B. GPU parallel implementation of isometric mapping for hyperspectral classification [J]. IEEE Geoscience and Remote Sensing Letters, 2017, 14(9): 1532-1536.

[8] 李毅. 基于 Hadoop 平台的局部线性嵌入算法研究[D]. 广州,华南理工大学,2011.

[9] 卞云龙. 基于云计算平台的大规模流形学习算法研究[D]. 南京,南京理工大学,2012.

[10] 刘勇. 头部姿态估计的监督流形学习研究及其并行化扩展[D]. 厦门,厦门大学,2013.

[11] 薛永坚, 倪志伟. 基于 MapReduce 的大规模数据集流形学习降维研究[J]. 系统工程理论与实践, 2014, 34: 151-157.
XUE Yongjian, NI Zhiwei. Research of large scale manifold learning based on MapReduce[J]. Systems Engineering - Theory & Practice, 2014, 34: 151-157.

[12] CAMPANA-OLIVO R, MANIAN V B. Parallel implementation of nonlinear dimensionality reduction methods applied in object segmentation using CUDA in GPU[C]// Proceedings of the International Society for Optical Engineering. SPIE, 2011:289-293.

[13] MOUSTAFA M, EBEID H M, HELMY A, et al. Parallel implementation of super-resolution based neighbor embedding using GPU [C]// Proceedings of the International Conference on Advanced Intelligent Systems and Informatics. Springer, 2016: 628-638.

[14] SAMUDRALA S K, ZOLA J, ALURU S, et al. Parallel framework for dimensionality reduction of large-scale datasets [J]. Scientific Programming, 2015, No. 1: 1-12.

[15] SCHOENEMAN F, ZOLA J. Scalable manifold learning for big data with Apache Spark [C]// Proceedings of IEEE International Conference on Big Data. Seattle, USA: IEEE, 2018:272-281.

[16] WANG YH, GAO Y, XU C. Manifold learning method for large scale dataset based on gradient descent [C]//Proceedings of 3rd International Conference on Multimedia Technology. Springer, 2013, 84:1187-1194.

[17] ZHOU Y. A block Chebyshev-Davidson method with inner-outer restart for large eigenvalue problems[J]. Journal of Computational Physics, 2010, 229(24): 9188-9200.

[18] SADKANE M, SIDJE R B. Implementation of a variable block Davidson method with deflation for solving large sparse eigenproblems [J]. Numerical Algorithms, 1999, 20:217-240

[19] 王顺绪,戴华. 求解大型矩阵特征值问题的并行块 Davidson 方法[J]. 南京航空航天大学学报,2007,39(6): 814-818.
WANG Shunxu, DAI Hua. Parallel Block Davidson Method for Solving Large Scale Eigenvalue Problem [J]. Journal of Nanjing University of Aeronautics & Astronautics,2007,39(6): 814-818.

[20] DATAR M, IMMORLICA N, INDYK P, et al. Locality-sensitive hashing scheme based on p-stable distributions [C] // Proceedings 20th of Annual symposium on Computational Geometry. New York, USA: ACM, 2004: 253-262.

[21] SAUL L K, ROWEIS S. An introduction to locally linear embedding [EB/OL]. Journal of Machine Learning Research, [2019-8-30] http://www. cs. toronto. edu/~roweis/lle,2001.