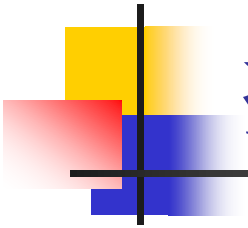
针对罗吉斯回归模型优化算法并行加速的研究

计算机系 彭昊若
指导老师 赵颖

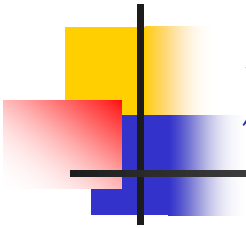


提纲

- 研究主题
  - 研究背景与相关工作
  - 研究内容与方式
- 研究过程
  - 并行系统简介
  - 算法简介
- 研究结果
- 总结

研究背景

- 大数据的挑战
  - 运算效率问题
  - 算法可扩展性
- 罗吉斯回归模型（LR）的广泛应用
  - eg. Anti-Spam Filtering & PageRank
  - 机器学习领域的发展
- 选择LR进行研究的优势
  - 简洁+理论基础强
  - 二分类->多分类 LR->其他模型

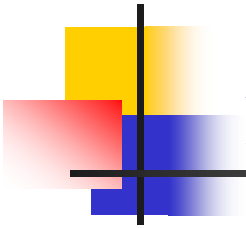


相关工作

- 大数据背景下的机器学习算法
  - PSVM、PLDA、DP、GraphLab
- 并行框架
  - Apache Hadoop
  - Apache Mahout
  - Apache Spark
- 算法框架
  - 次线性、并行梯度下降、随机梯度下降

研究内容与方式

- 从并行系统层面和算法层面综合解决罗吉斯回归模型优化算法的计算效率问题
- 主要加速方式：并行计算
- 研究重点在于充分认识
  - 在何种情况下选择并行
  - 选择什么并行系统
  - 选择怎样的并行算法
  - 次重点为新的次线性并行算法的提出

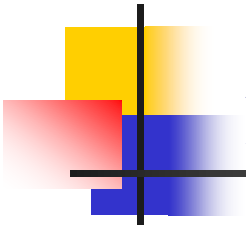


并行系统-Hadoop

- MapReduce框架
  - 键-值对
  - Map & Reduce
- HDFS
  - 文件分割
- 对节点失败的鲁棒性
  - 存储冗余

并行系统-Mahout

- 专门针对大规模机器学习算法
  - 主要4类支持的算法：推荐系统、聚类、分类、频繁模式
  - 其他包括： K-means、LDA、SVD、Bayes、Decision Tree, etc.
- 很多算法实现建立在Hadoop基础上
- 但其中针对LR的算法使用在线随机梯度下降法，没有使用多机并行，而使用单击多线程



并行系统-Spark

- 专门针对迭代算法
  - 迭代是机器学习算法一大特点
- 使用In-Memory策略
- 基本延续了MapReduce执行框架
  - 无Shuffle，单Reducer
- 弹性分布式数据集RDD
  - 线性操作Lineage
- 对节点失败鲁棒性
  - Lineage为主，Checkpointing为辅

LR模型下的优化问题

- 训练数据集定义
  - $\mathcal{X} = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$
    $\mathbf{x}_i \in \mathbb{R}^d \qquad y_i \in \{-1, 1\}$
  - $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^T$
    $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$

LR模型下的优化问题

- 罗吉斯回归模型定义
  - $P(y_i|\mathbf{x}_i) = \dfrac{1}{1+\exp(-y_i(\mathbf{x}_i^T\mathbf{w}+b))} \triangleq g_i(y_i)$
  - 考虑对数最大似然 $F(\mathbf{w}, b|\mathcal{X}) = \sum_{i=1}^{n} \log g_i(y_i)$
  - 最大后验

$$\max_{\mathbf{w},b} \left\{ \log p(\mathbf{w}, b|\mathcal{X}) \propto F(\mathbf{w}, b|\mathcal{X}) + \log p(\mathbf{w}) \right\}$$

LR模型下的优化问题

- 带二阶惩罚项（高斯先验）

$$\max_{\mathbf{w},b} \left\{ F(\mathbf{w}, b|\mathcal{X}) - \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \right\}$$

- 带一阶惩罚项（拉普拉斯先验）

$$\max_{\mathbf{w},b} \left\{ F(\mathbf{w}, b|\mathcal{X}) - \gamma\|\mathbf{w}\|_1 \right\}$$

次线性优化方法

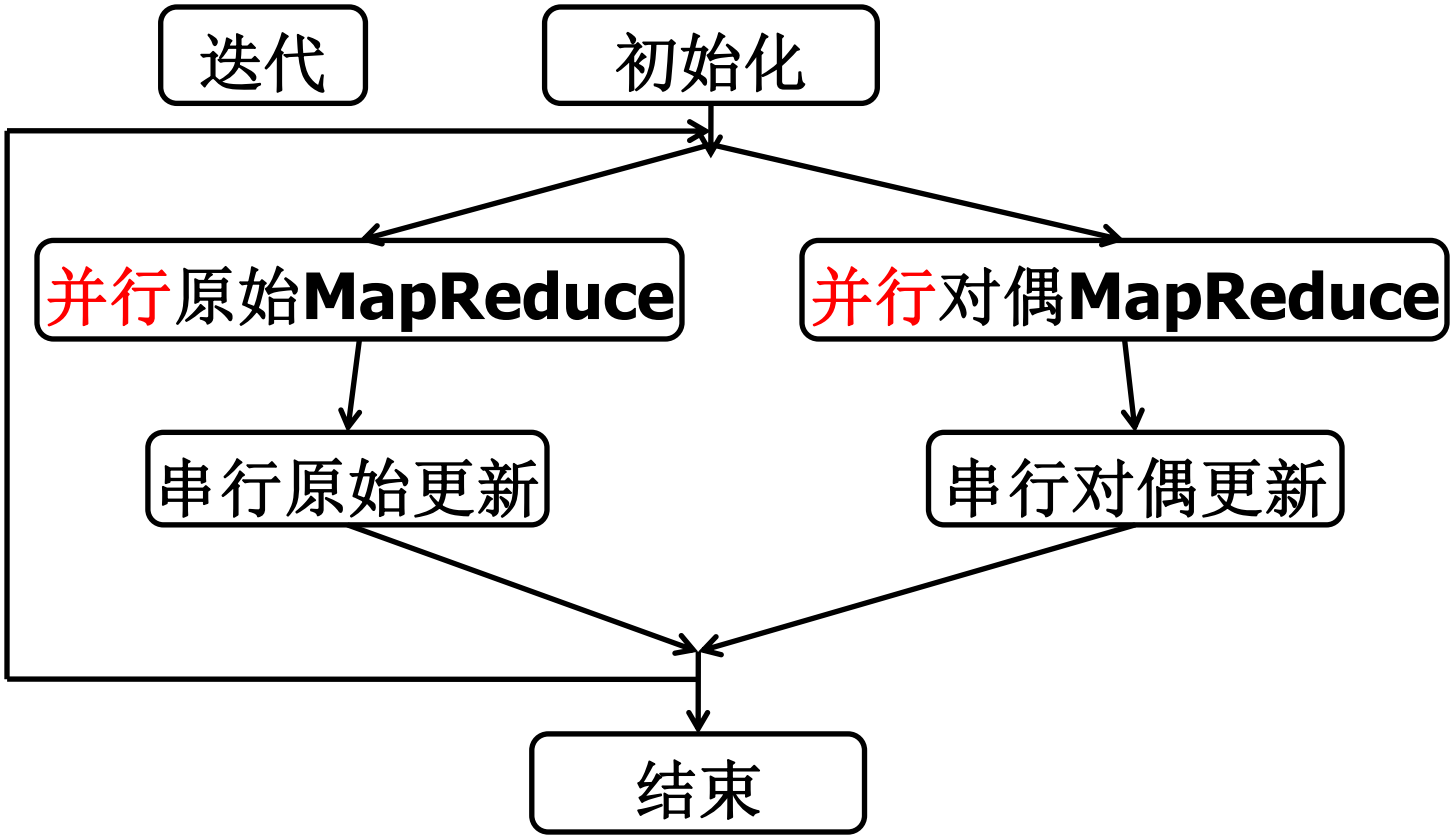
- 中心思想
  - 随机抽取（Sampling）
    - 训练数据不是每一维都有效
    - 不仅在样本空间中抽取，也在特征空间中抽取
  - 对偶方法
    - 样本空间与特征空间不断转换
  - 可乘式更新算法
    - 在特征空间中保持优化问题的梯度计算方向不变
- 算法细节可以参考引文[5]

次线性优化方法

- 算法框架（每次迭代）
  - *阶段1: 随机原始更新*
    - 根据概率向量 $\mathbf{p}$ 随机抽取一个训练样本
    - 计算该样本梯度方向，并更新回归向量 $\mathbf{w}$
  - *过渡：* 根据 $\mathbf{w}$ 抽取特征中的一维（关键）
  - *阶段2:随机对偶更新*
    - 对每个训练样本，根据抽取的一维特征，分别计算硬边际量加软边际量的估计值。
    - 根据上述数值，使用可乘式更新算法分别更新概率向量 $\mathbf{p}$ 的每一个维

并行次线性方法

- 算法框架



Hadoop上的并行次线性方法

- 参数传递
  - 选择直接读写HDFS文件
- 关于一阶和二阶惩罚项
  - 两个版本，实验结果均采用一阶惩罚项模型
- 训练数据集的稀疏问题
  - 代码优化

Spark上的并行次线性方法

1: Input parameters: ε, ν *or* γ, X, Y, n, d

2: Initialize parameters: $T, \eta, \mathbf{u}_0, \mathbf{w}_1, \mathbf{q}_1, b_1$

3: ps←spark.textFile(inputfile).map(parsePoint()).cache()

4: Iterations: $t = 1 \sim T$

5: g←ps.map($(1/(1+e^{-y(\mathbf{w}^T\mathbf{x}+b)})-1)*y*\mathbf{p}[index]$).reduce(_+_)

6: $(\mathbf{w}_{t+1}, b_{t+1})$←PrimalUpdate$(\mathbf{w}_t, b_t)$

7: Choose $j_t \leftarrow j$ with probability $\mathbf{w}_{t+1}(j)^2/\|\mathbf{w}_{t+1}\|_2{}^2$

8: pAdjust←points.map(MW-Update()).reduce(copy())

9: $\mathbf{p}_{t+1}$ ←DualUpdate$(\mathbf{p}_t)$

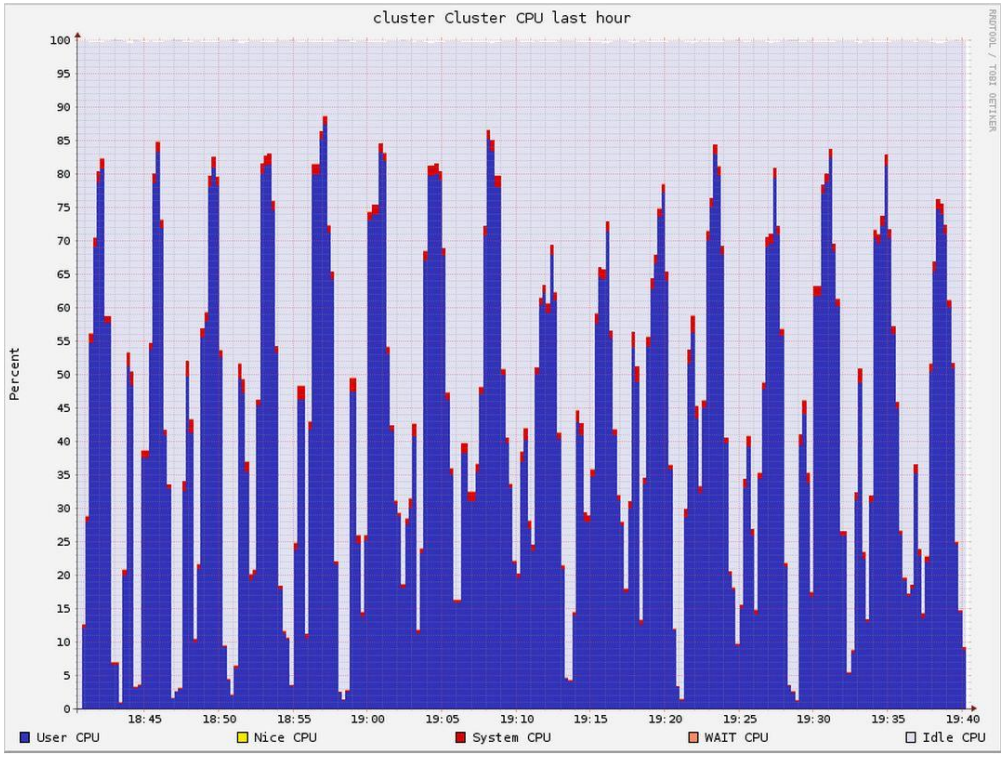
10: Output: $(\mathbf{w}, b)$

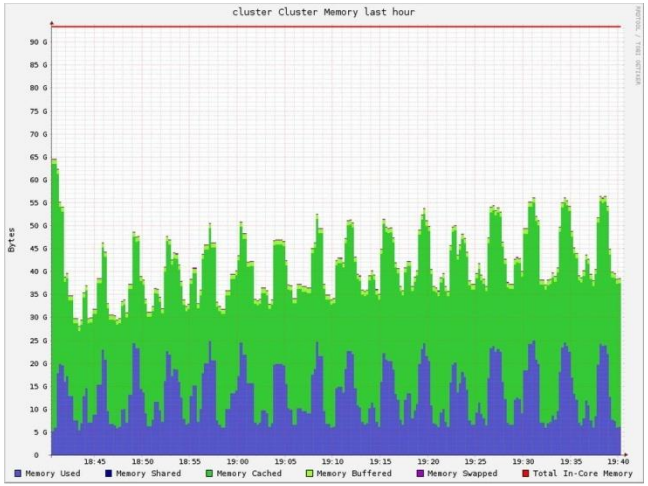
Spark上的并行次线性方法

- 时间复杂度
  - O(n*d)->O(n+d)->O(max{n,d})
- 输入数据解析
  - 一遍->在内存中
- 参数传递
  - 内存+缓存
- 训练数据集的稀疏问题
  - 基于的RDD代码优化

Hadoop VS Spark

- Memory Utilization





Spark上的并行梯度下降法

1: Input parameters: ε, ν *or* γ, X, Y, n, d

2: Initialize parameters: $T, \eta, \mathbf{u}_0, \mathbf{w}_1, \mathbf{q}_1, b_1$

3: ps←spark.textFile(inputfile).map(parsePoint()).cache()

4: Iterations: $t = 1 \sim T$

5: gradient←ps.map($(1/(1 + e^{-y(\mathbf{w}^T\mathbf{x}+b)}) - 1) * y$).reduce(_+_)

6: $\mathbf{w}_{t+1} = \mathbf{w}_t - gradient * \mathbf{x}$

7: $b = b - gradient$

8: Output: $(\mathbf{w}, b)$

Mahout上的在线SGD算法

- 算法函数
  - OnlineLogisticRegression
- 数据存储结构
  - RandomAccessSparseVector
- 交叉验证
  - 使用自己的CrossValidation函数
- 单机多线程

实验环境

- 数据集

名称	特征维数	数据个数	稀疏性	非零元素个数	正例个数与反例个数比
2D	2	200	1.0	400	1.000
20NewsGroup	16248	1988	7.384×10^{-3}	238511	1.006
Gisette	5000	7000	0.12998	4549319	1.000
ECUESpam	100249	10678	2.563×10^{-3}	2746159	5.882
URL-Reputation	3231961	2376130	3.608×10^{-5}	277058644	0.500

实验环境

- 集群

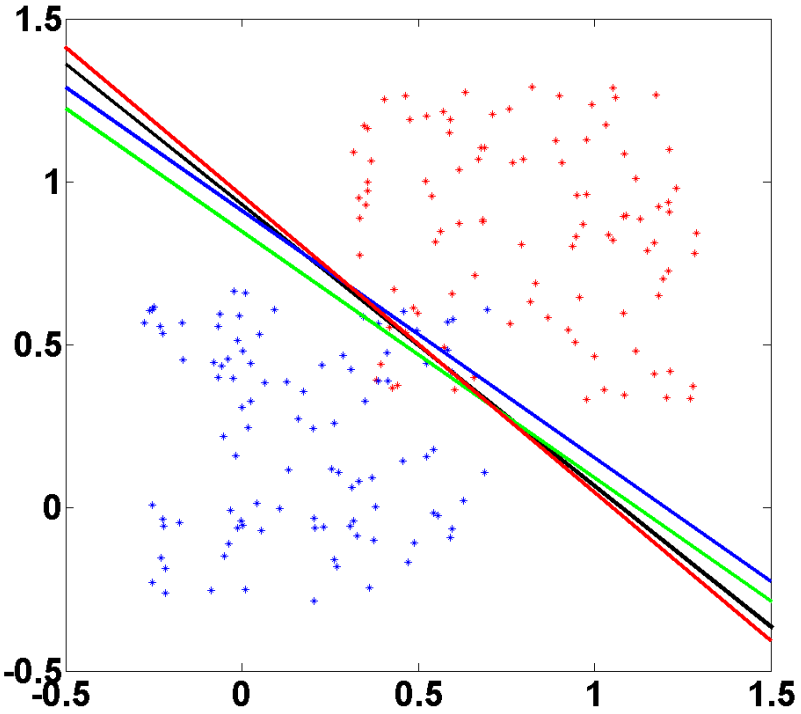
CPU 型号	Intel Xeon E5-1410
CPU 主频	2.80GHz
节点数	6
每个节点上 CPU 核数	4 核 8 线程
每个节点内存大小	16G
每个节点硬盘大小	4T HDD
节点间连接方式	Gigabyte Ethernet

实验环境

- 测试程序
  - Liblinear：串行
  - SLLR：串行
  - Mahout：单机多线程
  - PSUBPLR-MR：多机并行
  - PGDPLR-SPARK ：多机并行
  - PSUBPLR-SPARK：多机并行

实验结果

- 仿真2D数据集



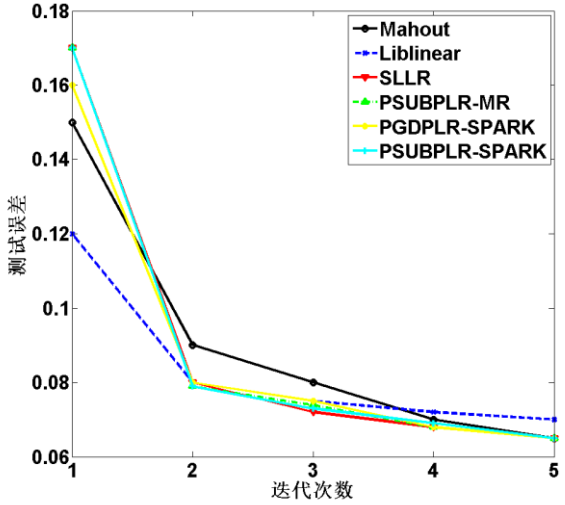
实验结果

- 学习精度结果

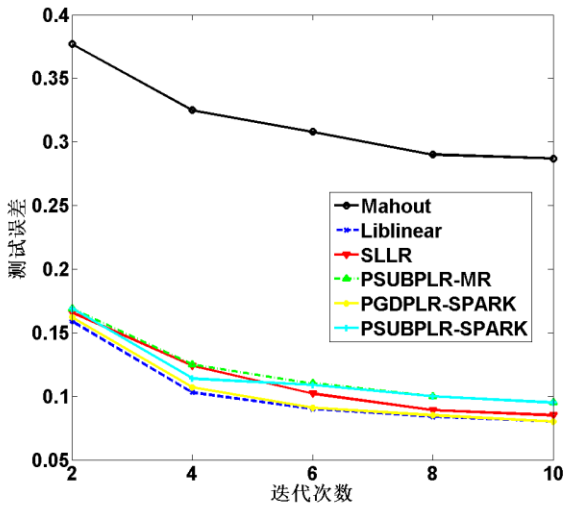
	2D	20NewsGroup	Gisette	ECUESpam	URL-Reputation
Mahout	93.5%	71.3%	91.5%	85.2%	91.5%
Liblinear	93.0%	92.0%	97.4%	97.1%	96.2%*
SLLR	93.5%	91.5%	94.8%	92.3%	94.2%
PSUBPLR-MR	93.5%	90.5%	94.6%	91.7%	93.8%
PGDPLR-SPARK	93.5%	92.0%	97.0%	93.7%	96.0%
PSUBPLR-SPARK	93.5%	90.5%	95.8%	91.7%	94.0%

实验结果

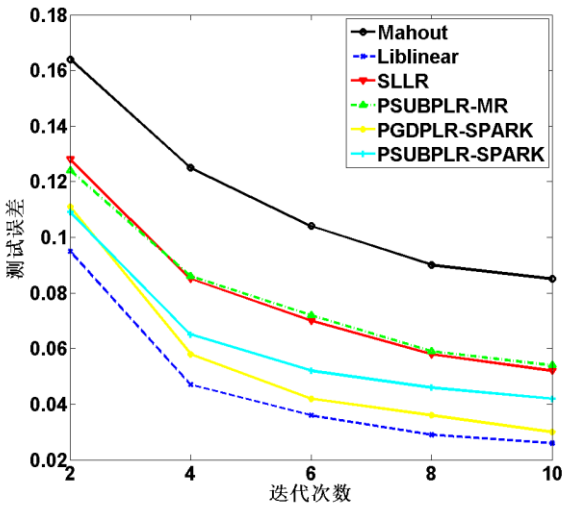
- 迭代-测试误差



仿真2D数据集



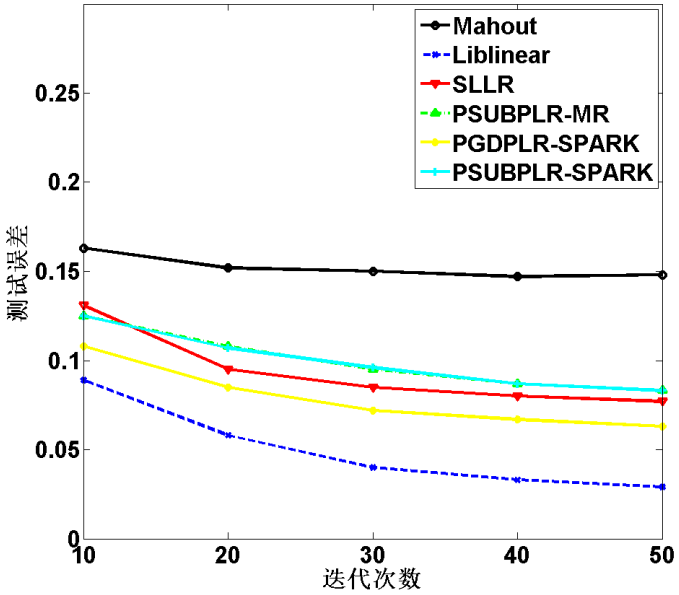
20NewsGroup数据集



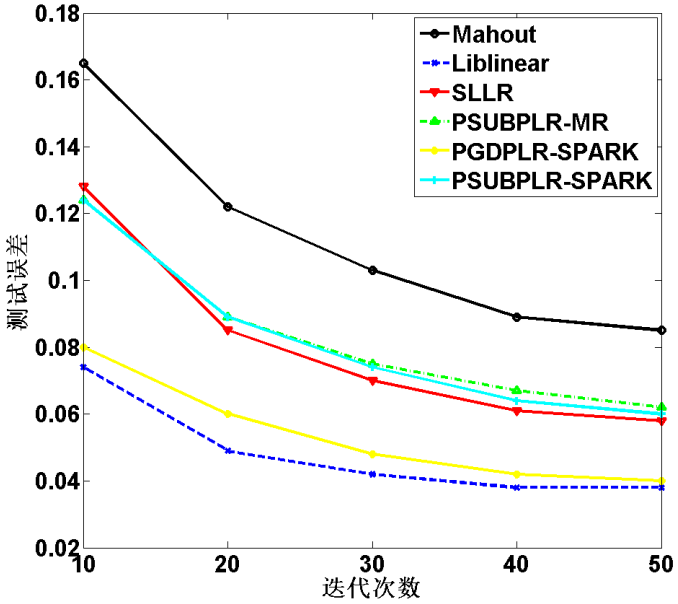
GISETTE数据集

实验结果

- 迭代-测试误差



ECUESpam数据集



URL-Reputation数据集

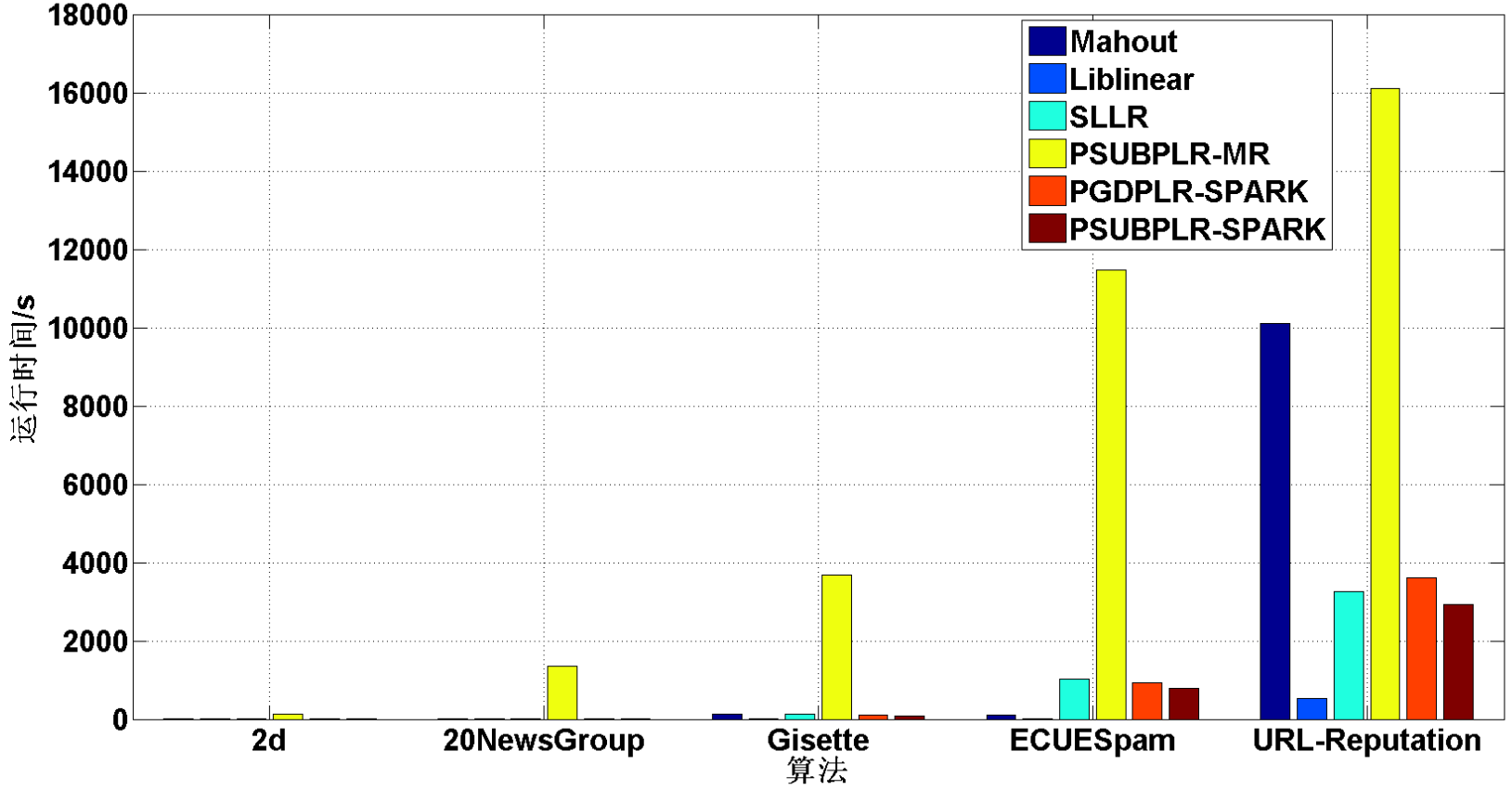
实验结果

- 运行时间结果

	2D	20NewsGroup	Gisette	ECUESpam	URL-Reputation
Mahout	0.595s	9.827s	131.807s	96.611s	10100.209s
Liblinear	0.078s	0.793s	2.364s	13.161s	519.115s*
SLLR	1.761s	20.046s	130.451s	1028.185s	3248.473s
PSUBPLR-MR	120.186s	1360.854s	3687.941s	11478.706s	16098.260s
PGDPLR-SPARK	0.681s	10.517s	99.156s	924.020s	3615.780s
PSUBPLR-SPARK	1.325s	8.571s	89.094s	796.802s	2918.470s

实验结果

- 运行时间结果比较



精度与时间结果分析

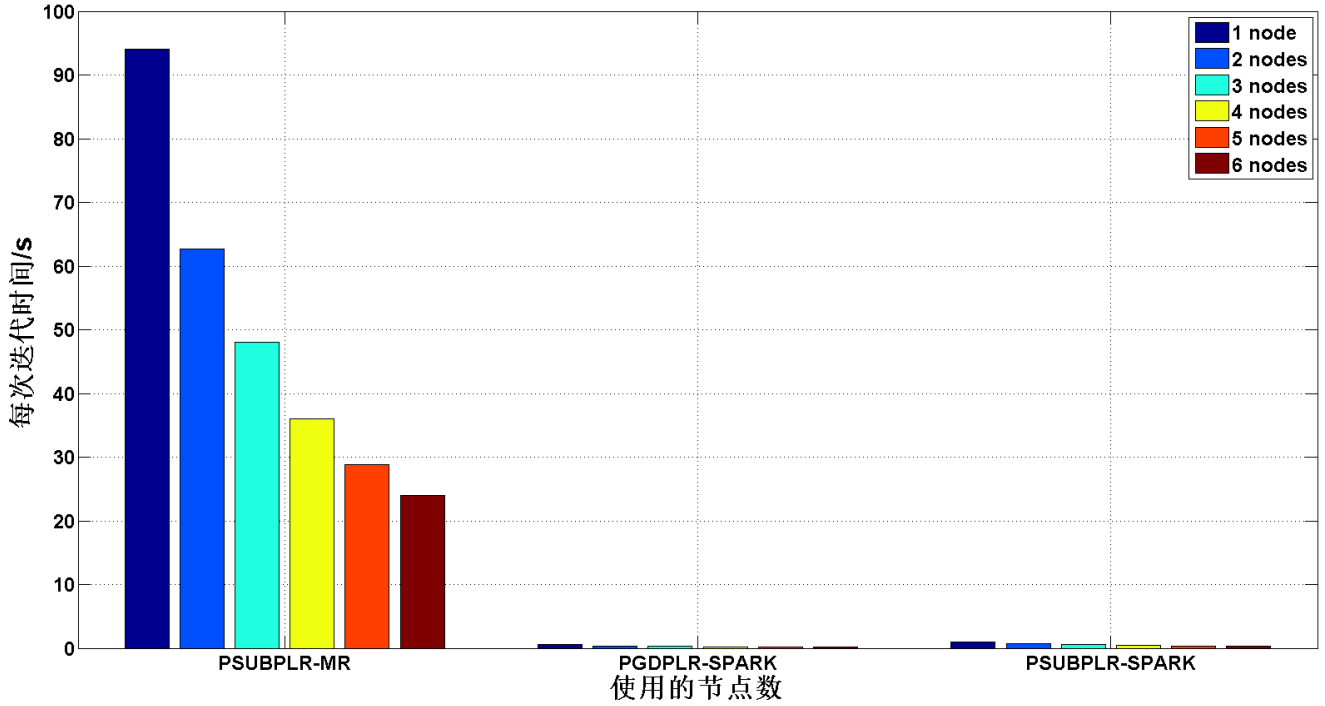
- Liblinear 精度高、速度快、大数据差
  - 数据集较小的情况（充分利用内存）
- Mahout 精度差、速度一般、支持大数据
  - 大数据、单机、内存有限（利用在线方法）
- Sublinear方法
  - 精度均接近最优
  - 精度相对于串行程序有微小下降

精度与时间结果分析

- Hadoop 支持大数据、速度慢
  - 设计缺陷：非环形数据流
  - 任务调度时间与文件读写时间
  - 原始更新部分是瓶颈
- Spark 支持大数据、速度较快
  - RDD的本地拷贝形式
  - 适合大数据、多计算节点
  - 追求精度推荐PGDPLR-SPARK算法
  - 追求速度推荐PSUBPLR-SPARK算法

实验结果

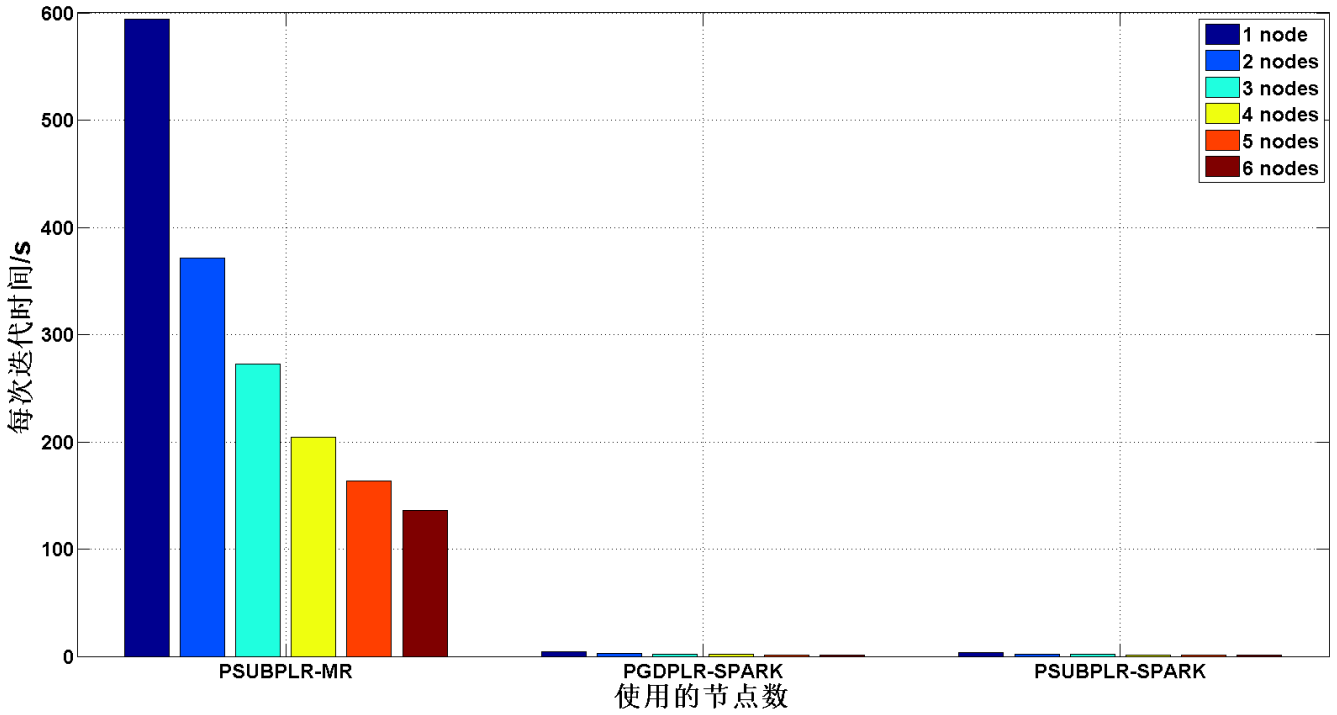
- 计算节点数-迭代时间



仿真2D数据集

实验结果

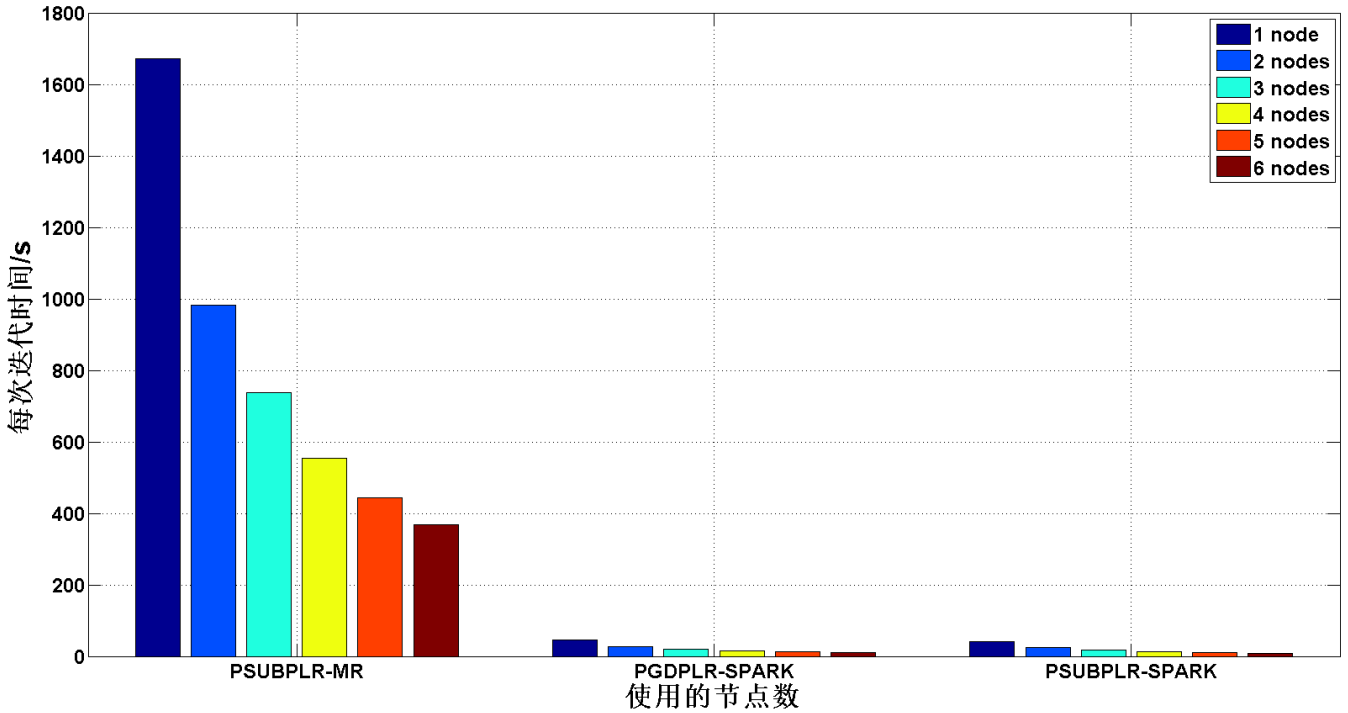
- 计算节点数-迭代时间



20NewsGroup数据集

实验结果

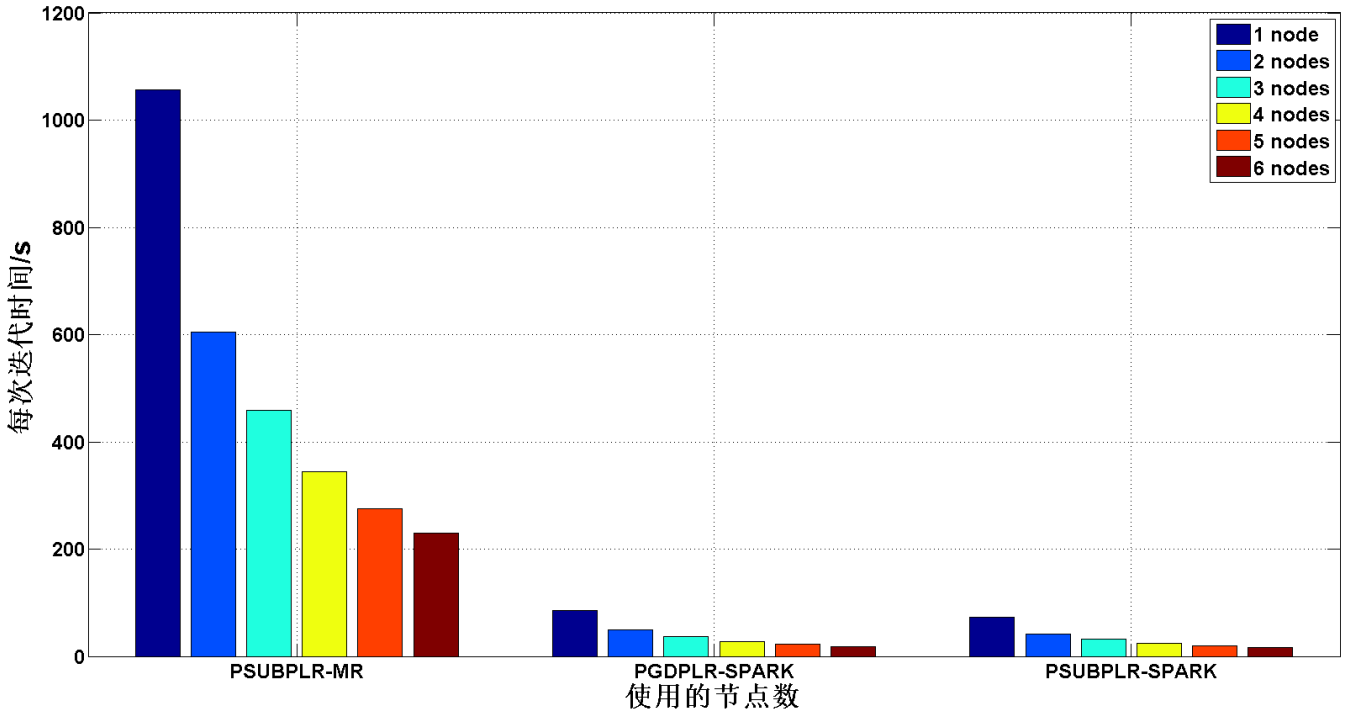
- 计算节点数-迭代时间



GISETTE数据集

实验结果

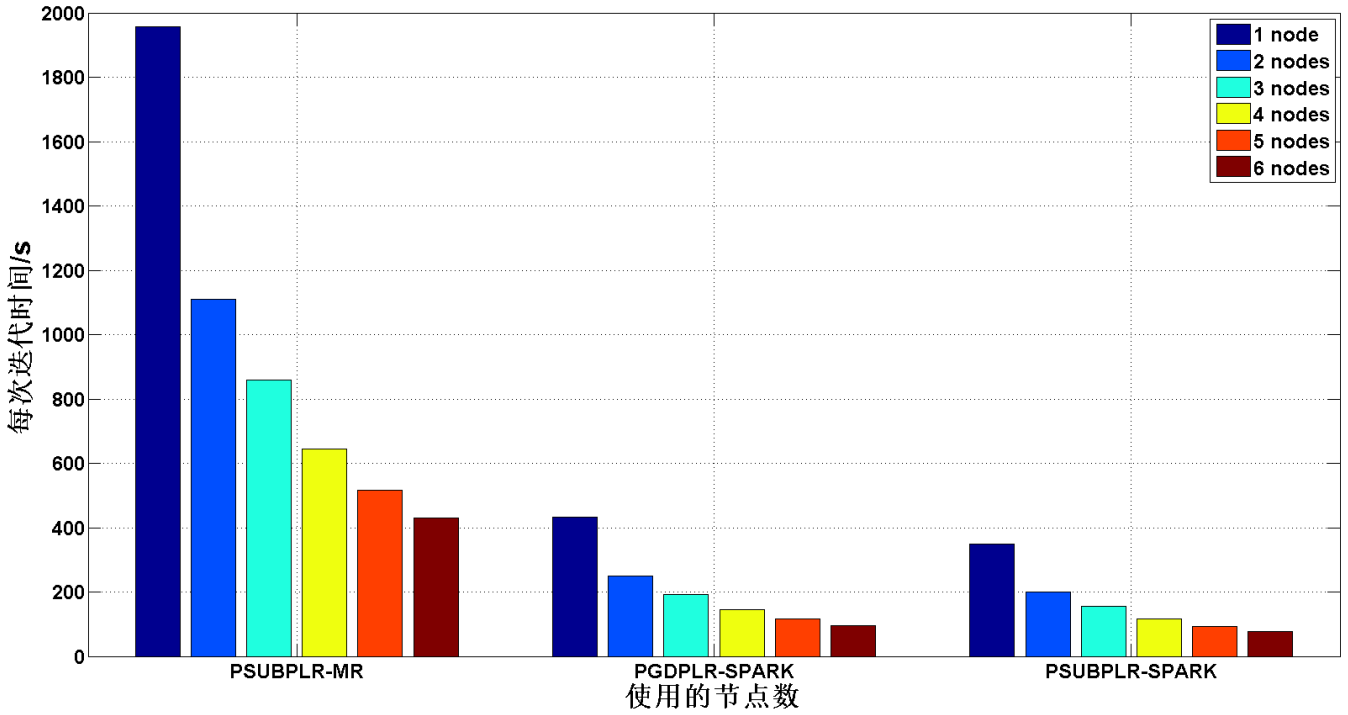
- 计算节点数-迭代时间



ECUESpam数据集

实验结果

- 计算节点数-迭代时间



URL-Reputation数据集

不同集群资源下测试结果分析

- 非线性过程
  - 算法在集群上的可扩展性极限
- ECUESpam数据集 VS GISETTE数据集
  - 特征维度与样本个数：高 VS 低
    - -> 迭代次数：多 VS 少
  - 稀疏性：高 VS 低
  - 非零元素：少 VS 多
    - -> 迭代时间：短 VS 长
  - 总运行时间：长 VS 短

实验结果

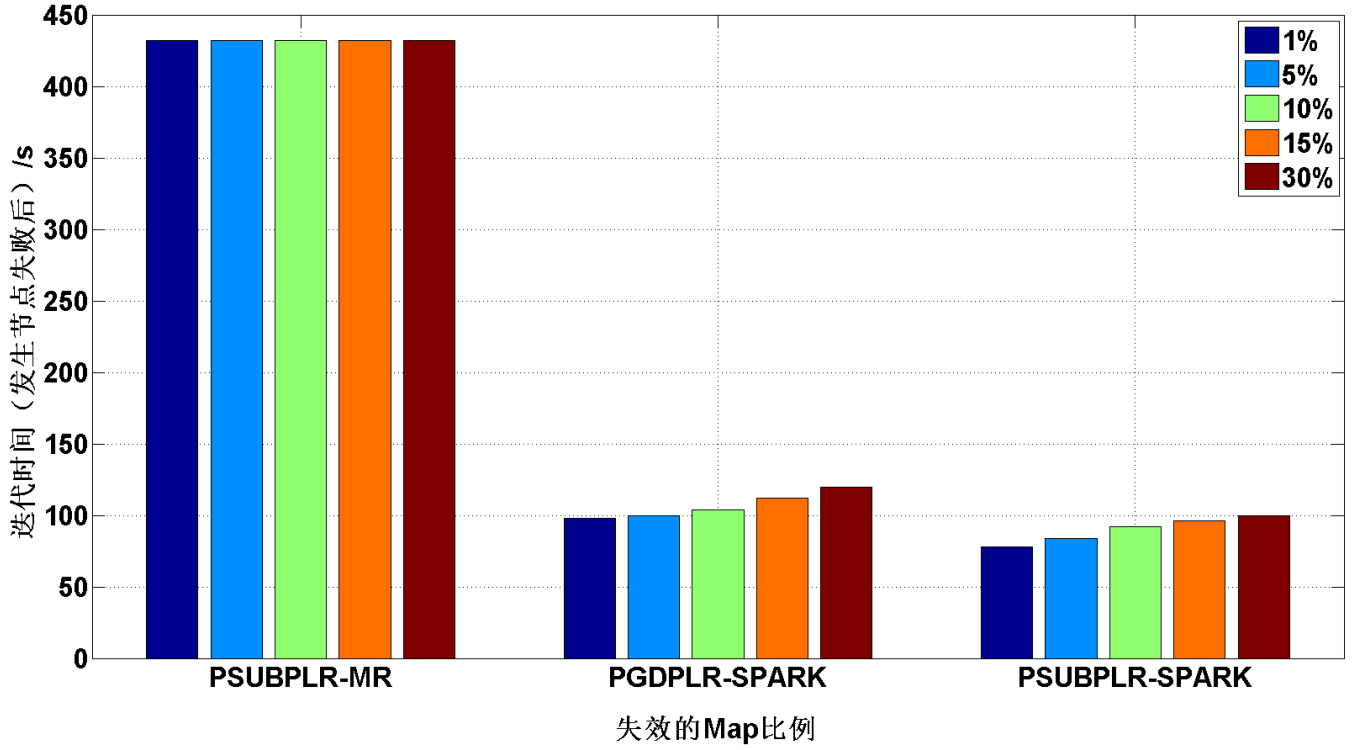
- 对节点失败鲁棒性

	System Level	Algorithm Level
PSUBPLR-MR	√	√
PGDPLR-SPARK	√	×
PSUBPLR-SPARK	√	√

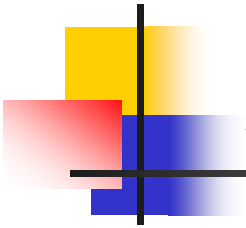
- Hadoop系统：冗余机制
- Spark系统：RDD的线性操作
- 次线性方法：随机算法特性

实验结果

- 对节点失败鲁棒性



URL-Reputation数据集



总结

- 主要成果
  - 针对罗吉斯回归模型优化求解机器学习算法
  - 大数据应用背景
  - 考虑了以下维度上的各个特点：
    - 数据集：Size, #Instances, #Features, Sparsity
    - 并行系统：Design, Feature, Mechanism
    - 算法：Design, Framework, Code
    - 综合测试：Correctness, Accuracy, Efficiency, Scalability, Fault Tolerance

总结

- 主要创新
  - 进行了客观综合地研究、测试、评价和比较。针对不同特点的数据集和不同的运行资源，给出了算法选择的建议。
  - 提出了针对罗吉斯回归模型的新的并行次线性算法。其在SPARK系统上的运行取得了优良的效果。
  - 将机器学习算法与系统综合考虑的研究思路也是本文的一个特色

下一步研究计划

- 需要发展与机器学习算法相匹配的分布式并行计算系统
- 进一步增加综合测试
  - 包括算法稳定性、算法收敛性等方面
  - 增加实验节点数，在更大规模数据集上测试
- 并行次线性方法有待继续优化
- 研究对象扩展
  - 多分类问题以及其他模型

主要参考文献

- [1] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker, Identifying Suspicious URLs: An Application of Large-Scale Online Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 681-688, Montreal, Quebec, June 2009.
- [2] http://qwone.com/~jason/20Newsgroups/
- [3] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror. Result analysis of the nips 2003 feature selection challenge. *Advances in Neural Information Processing Systems*, 17:545–552, 2004.
- [4] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle. A case-based technique for tracking concept drift in spam filtering. Knowledge-Based Systems, 18(4–5):187–195, 2005.

主要参考文献

- [5] Haoruo Peng, Zhengyu Wang, Edward Y. Chang, Shuchang Zhou and Zhihua Zhang. Sublinear Algorithms for Penalized Logistic Regression in Massive Datasets. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases,* 2012.
- [6] Mahout - Apache Software Foundation project homepage, http://lucene.apache.org/mahout
- [7] Fan R E, Chang K W, Hsieh C J, et al. LIBLINEAR: A library for large linear classification[J]. *The Journal of Machine Learning Research*, 2008, 9: 1871-1874.
- [8] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica. Spark: Cluster Computing with Working Sets. *HotCloud* 2010. June 2010.

感谢各位老师同学给予我的帮助

谢谢！