朴素并行 LDA

高 阳 严建峰 刘晓升
(苏州大学计算机科学与技术学院 苏州 215006)

摘 要 并行潜在狄利克雷分配(LDA)主题模型在计算与通信两方面的时间消耗较大,导致训练模型的时间过长,因而无法被广泛应用。提出朴素并行LDA算法,针对计算和通信分别提出改进方法。一方面通过加入单词影响因子以及设置阈值的方法来降低文本训练的粒度,另一方面通过降低通信频率来减少通信时间。实验结果表明,优化后的并行LDA在保证精度损失为1%的前提下,将训练速度提高了36%,有效提高了并行的加速比。

关键词 潜在狄利克雷分配,并行,加速优化
中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.6.051

Naïve Parallel LDA

GAO Yang YAN Jian-feng LIU Xiao-sheng
(School of Computer Science and Technology, Soochow University, Suzhou 215006, China)

Abstract The parallel latent Dirichlet allocation (LDA) costs a lot of time in computation and communication, which brings about long time to train a LDA model and then it can't be widely applied. This paper proposed naïve parallel LDA algorithm, presenting two methods to solve this problem. One is to add impact factor of each word and set threshold to reduce the amount of corpus, the other is to reduce the communication frequency to decrease the communication time. Experimental results show that the optimized distributed LDA can accelerate the total training time by 36% and improve the speedup ratio, while the loss of accuracy is below 1%.

Keywords Latent Dirichlet allocation, Parallel, Speedup optimization

1 引言

随着数据的不断膨胀,结构化与非结构化信息需要被有效地处理与理解。而网络中文本信息占据着主要的地位。各类处理文本的模型被相继提出。从最初的一元语言模型与混合一元语言模型,进化到潜在语义索引(latent semantic index, LSI)[1],进而衍生了概率潜在语义索引(probabilistic LSI, pLSI)[2]和潜在狄利克雷分布(latent Dirichlet allocation, LDA)[3],这类主题模型在文本上的语义挖掘目前已经较为成熟,有多种算法被用来推理主题模型的后验概率。其中作为目前更符合文本生成过程的LDA,变分贝叶斯(variational Bayes, VB)[3]、吉布斯采样(Gibbs sampling, GS)[4]以及置信传播(belief propagation, BP)[5]是其3种主要的近似推理算法,而置信传播在精度上有明显优势。

对于大规模数据,单机单核的程序已经无法满足大数据的要求。目前处理大规模数据的几种方法包括共享内存方式与非共享内存方式。相关的框架有多线程的OpenMP、多进程的消息传递机制(Message Passing Interface, MPI)[6]、MapReduce框架编程等等。

本文基于非共享内存方式,利用MPI来进行进程间的通信,采用Newman[7]等人提出的近似分布式LDA的主体算法框架,并针对该并行算法的计算时间以及通信时间两个方面,对近似分布LDA模型进行了改进,其中采用置信传播算法作为近似推理算法。实验证明,通过针对性的时间改进,提高了算法的运行效率以及并行的加速比,且精度方面得到一定保证。

2 相关工作

基于共享内存,以及非共享内存这两种框架,大规模主题模型有了较好的并行实现。Newman[7]等人首先提出了近似分布LDA(AD-LDA),基于吉布斯采样,采用全局同步的思想,首先基于全局初始化的主题模型参数对每一个分布的主题模型进行优化,然后同步融合得到全局的主题模型参数。Asuncion[8]等人提出了异步分布LDA(AS-LDA),采用处理器两两交互信息的方式,异步通信,不需要等待所有局部的主题模型优化完毕,这种方式也导致了收敛的速度慢。Parallel-LDA(PLDA)[9]是对AD-LDA进行了MPI以及MapReduce的实现,结果表明以MPI为框架的并行LDA实现比MapReduce有着更高的加速比。Liu[10]等人提出了一个与AD-LDA截然不同的框架PLDA+,即将单词表划分,并重新捆绑,使

到稿日期:2014-09-07 返修日期:2014-10-27 本文受国家自然科学基金(61003154,61373092,61033013,61272449,61202029),江苏省教育厅重大项目(12KJA520004),苏州大学创新团队(SDT2012B02),广东省重点实验室开放课题(SZU-GDPHPCL-2012-09)资助。
高 阳(1991-),女,硕士生,主要研究方向为机器学习;严建峰(1978-),男,副教授,硕士生导师,主要研究方向为机器学习、传感器网络,E-mail:yanjf@suda.edu.cn;刘晓升(1976-),男,博士生,主要研究方向为机器学习。

用管道处理机制，用 CPU 计算时间来屏蔽通信时间。Mr. LDA[11]是 MapReduce 框架下的实现，基于变分贝叶斯，将 VB 中的 EM 过程很好地结合到 Map 过程和 Reduce 过程中。除 MPI 以及 MapReduce 外，多线程共享内存也是一种并行的方式。Yan[12]等人在 GPU 上提出了一种二维切分的方法，使得内存小的 GPU 可以处理一小部分的数据，其利用 GPU 上数以千计的核数进行多线程并行。

3 朴素并行 LDA

LDA 模型是一种 3 层的贝叶斯模型。模型假设整个文本集共有 K 个主题，每个主题 k 是单词表的概率分布。每篇文档 d 是这 K 个主题的概率分布。一篇文档的生成过程如下所示，其中 Dir 表示狄利克雷分布：

$$\theta_d \sim \mathrm{Dir}(\alpha), \phi_k \sim \mathrm{Dir}(\beta), z_i \sim \theta_d, x_i \sim \phi_{z_i} \quad (1)$$

即从以 α 的狄利克雷分布中获得一篇文档 d 的分布 θ_d，从以 β 的狄利克雷分布中获得每个主题 k 的分布 ϕ_k，从 θ_d 中获得一个主题 z_i，再从主题单词分布 ϕ_k 中获得一个单词 x_i，重复这样的过程直到得到所有的文档。图 1 为 LDA 的图模型表示。

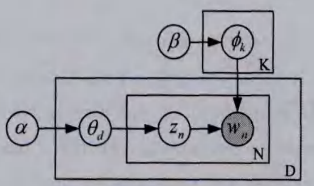

图 1 LDA 图模型

3.1 并行 LDA

近似分布 LDA(approximate distribution LDA，ADLDA)是基于吉布斯采样的近似分布同步框架的 LDA，其主要思想是将 D 篇文档均匀分布到 P 个处理器上，则每个处理器大约有 $D_p = D/P$ 篇文档，每个处理器单独进行一次迭代后，对各个处理器上的信息进行汇总之后再进行下一次迭代。本文使用 BP 算法来替换 ADLDA 里的 GS 算法，在精度方面 BP 相比 GS 有足够的优势；并且，GS 采样不同文档之间有所依赖，导致不是完全的并行，而是近似的并行，而在 ADLDA 架构下使用 BP 算法则是完全的并行，其并行的精度可以保证与单机的一致，这是因为 BP 算法的更新公式在文档之间没有相互依赖关系，彼此之间不相互影响。所以，我们选择 BP 算法作为本文的 LDA 近似推理算法，记为 Parallel Belief Propagation(PBP)。其主要过程如下所示：

算法 1 并行 LDA(PBP)

重复：

1. For each processor p in parallel do：

2. 复制局部参数到全局参数：$\phi_{w|k,p} \leftarrow \phi_{w|k}$

3. 更新 $\mu(z_{w,d}=k), \theta_{k|d}$

4. End for

5. 同步，更新全局的 ϕ：$\phi_{w|k} \leftarrow \phi_{w|k} + \sum_p (\phi_{w|k,p} - \phi_{w|k})$

直到达到停止条件

对于每一次迭代，各个处理器并行地执行，首先将全局参数 $\phi_{w|k}$ 复制给局部的 $\phi_{w|k,p}$，再使用局部的 $\phi_{w|k,p}$ 来进行 $\mu(z_{w,d}=k), \theta_{k|d}$ 的更新，更新的方法可以是几种近似推理算法中的一种。等到所有处理器完成一次迭代之后，再对所有处理器上局部的 $\phi_{w|k,p}$ 进行汇总，并将结果赋给全局的 $\phi_{w|k}$。重复此过程，直到达到停止条件为止。

3.2 朴素加速优化

3.2.1 计算时间优化

计算时间的优化，即减少每次迭代更新相关参数的时间。LDA 的计算时间都用于对每篇文档的每个单词进行主题的重分配。通过不断地迭代，使单词的主题分配达到稳定状态。然而，每个单词的影响因子不一样。词频高的单词的影响因子较大，需要更新的频率以及概率较高；相反，词频低的单词的影响因子较小，需要更新的频率以及概率较低。

通过统计词频，获得单词表里每个单词的影响因子，得到一个单词表大小的影响因子向量。例如，单词表大小为 4，其对应的单词的词频由大到小分别为{10,5,4,1}。将该词频向量归一化即得到影响因子向量，为{0.5,0.25,0.2,0.05}。使用[0,1]上的均匀分布产生随机概率，将该随机概率与对应单词的影响因子相乘，得到的概率与一个阈值作比较。若该概率大于阈值，则更新对应单词的主题分布，否则不更新。由于 LDA 是基于词袋(Bag of Words，BOW)的模型，可以控制每个单词是否更新参数来减少计算时间。而阈值的设定对模型的好坏有一定的影响，需要具体探讨与研究。

3.2.2 通信时间优化

通信时间由两个部分组成：等待时间和实际通信时间。等待时间即最快完成的处理器等待最慢的处理器完成所需要花费的时间。这部分时间需要考虑负载平衡问题，而一般情况下的输入文本都是均匀分配，因此我们不讨论该部分。

实际通信时间和处理器数目、通信数据量有关。处理器数目越多，信息的融合过程越繁琐，通信时间越长。通信数据量越大，用于传输的时间就越长，同样也会延长通信时间。由于无法避免要使用更多的处理器，因此可以改进的部分即是通信的数据量。

我们尝试减少通信的频率来降低通信数据量。将原来一次迭代后同步更新参数换成每隔几次迭代，该方法带来两个问题。首先，一次迭代之后，最慢的处理器比最快的处理器慢的时间，会随着更新间隔的加长而加长，即可能会带来等待时间的加剧或者没有改善等待时间；另一个问题就是精度的下降，由于更新不及时，会导致收敛的速度变慢。同等迭代总数的情况下，精度不及以前。鉴于此，我们将通过具体实验来选择较优的更新间隔，以使得这两个问题不明显。

结合计算时间与通信时间的优化，可以得到如算法 2 所示的加速优化并行 LDA 的算法过程，其中，需要调节的参数是通信间隔的迭代次数以及随机阈值。

算法 2 加速优化并行 LDA(Naive PBP)

重复：

For each processor p in parallel do：

复制局部参数到全局参数：$\phi_{w|k,p} \leftarrow \phi_{w|k}$

m=0

While m<间隔迭代次数 do：

对处理器 p 上的所有单词 w：

If w 影响因子×rand(0,1)> 阈值：

更新 $\mu(z_{w,d}=k), \theta_{k|d}$

End if

m=m+1

End while

End for

同步，更新全局的 ϕ：$\phi_{w|k} \leftarrow \phi_{w|k} + \sum_p (\phi_{w|k,p} - \phi_{w|k})$

直到达到停止条件

4 实验分析

4.1 实验环境与数据集

本文基于单机多核服务器进行相关实验，总共有 4 个 CPU，每个 CPU 有 6 核，采用超线程技术，最多可以模拟 48 核。单机多核服务器在通信速度方面较集群或机群等更快，但不影响实验结果。

本文使用的 3 个数据集分别是 KOS、BLOG 和 ENRON，具体统计如表 1 所列。

表 1 LDA 模型

数据集	D	W	ntokens	nnz
KOS	3430	6906	467714	353160
BLOG	5177	33574	1125982	772032
ENRON	39861	28102	6412172	3710420

表 1 概括统计实验用的 3 个数据集，其中 D 是文档的总数目，W 是单词表的大小，*ntokens* 是所有文档的单词总数，*nnz* 是各个文档里不相同单词的数目之和。

4.2 评价标准

本文采用主题模型领域通用的性能评价指标即混淆度(Perplexity)和并行技术中的加速比(Speedup)来评价算法的性能。混淆度经常被用在语言模型中，是用来衡量语言模型对于测试语料库的建模能力的好坏，即似然的大小。混淆度的计算公式如下：

$$Perp=\exp\{-\frac{\sum_{w,d}x_{w,d}\log[\sum_{k}\theta_d(k)\phi_w(k)]}{\sum_{w,d}x_{w,d}}\} \quad (2)$$

越低的混淆度值表示了越好的泛化性能。而加速比是用来衡量程序并行化的性能和效果，即同一任务或程序在单处理器下运行时间与 P 个处理器并行系统下的运行时间之比。

4.3 实验结果分析

实验先验参数初始化都设置为 $\alpha=50/K$，$\beta=0.01$，这里 K 是主题的数目。由于并行 LDA 优化的两个因素和主题数大小的关系不大，因此本文中的所有实验都是基于主题数 $K=100$ 的，对应的 $\alpha=0.5$。实验将表 1 中的文本集随机平均分割成两个部分，一部分作为训练集，一部分作为测试集。使用 3 交叉验证方法，在训练集上求得模型，在测试集上得到测试混淆度结果，而计算时间与通信时间是训练集上的平均每次迭代的计算时间与通信时间。本文对并行 LDA 即 PBP 进行计算时间和通信时间两方面的优化，记为朴素并行 LDA(Naive PBP)。

4.3.1 计算时间实验

图 2 和图 3 分别是 KOS 与 BLOG 文本集上计算时间和测试混淆度随概率阈值的变化结果。阈值为 0，说明是未优化的 PBP；当阈值大于 0 时，即是优化计算时间后的 PBP。图 2 与图 3 中的阈值取值有{0.0,0.1,0.3,0.5,0.7,0.9}。由于随机概率是从[0,1]均匀分布中取值的，其计算时间的减少趋势呈现线性，与单词影响因子相结合后，计算时间略有波动。

从测试混淆度可以看出，阈值设到 0.5 时，其测试混淆度的增长还不明显，而阈值设为 0.9 时，测试混淆度上升加剧。阈值条件设置得越严格，更新的单词就越少，在相同迭代次数下，导致了测试混淆度的下降。因此，阈值设定在 0.5 之下对模型的影响较小，而且计算时间也缩短较多。

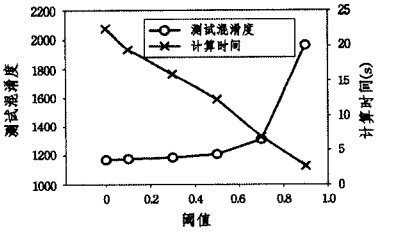

图 2 KOS 在不同阈值下测试混淆度以及计算时间的变化

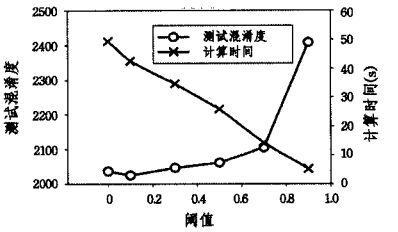

图 3 BLOG 在不同阈值下测试混淆度以及计算时间的变化

4.3.2 通信时间实验

图 4 和图 5 分别是 KOS 与 BLOG 文本集上通信时间和测试混淆度随通信间隔的变化结果。通信间隔为 1，说明是未优化的 PBP；当通信间隔大于 1 时，即是优化通信时间后的 PBP。图 4 与图 5 中的通信间隔取值有{1,2,4,6,8,10}。由于通信时间受等待时间的影响，其随着通信间隔的下降没有呈现直线下降的趋势。在 KOS 数据集上，其测试混淆度在通信间隔为 8 时比通信间隔为 6 时要低，可能是随机初始化导致的。从图 4 和图 5 中可以看出，在 BLOG 数据集上，当通信间隔设为 4 时，测试混淆度增长了 1%，而通信时间减少了 2.5 倍，有明显作用。而在 KOS 数据集上，下降得不明显，原因可能在于处理器通信间隔增大，导致同步等待时间也随之延长。

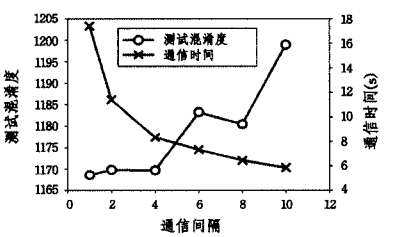

图 4 KOS 在不同通信间隔下测试混淆度以及通信时间的变化

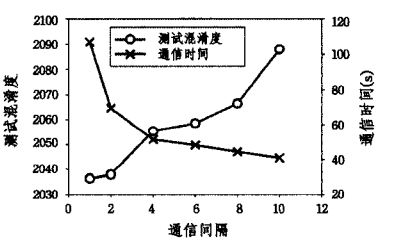

图 5 BLOG 在不同通信间隔下测试混淆度以及通信时间的变化

4.3.3 综合实验

图 6－图 8 分别是 KOS、BLOG、ENRON 文本集上未优化的 PBP 与优化后的 PBP，以及 Baseline 的加速比的对比结果，Baseline 为理想的加速比。根据以上概率阈值以及通信间隔的实验结果，在此，设定阈值为 0.3，通信间隔为 4 次迭代，在此设置下，测试集的上升小于 1%。实验记录 10 次迭代的时间，并取平均。图 6－图 8 中，处理器数目设为{1,2,4,8,16}，随着处理器数目的增加，优化后的 PBP 相比未优化的 PBP，加速比增加更大。优化后的 PBP 在 KOS 以及

BLOG上的加速比优化较大，在处理器数目为16时，有36%的优化，在ENRON上有19%的优化。

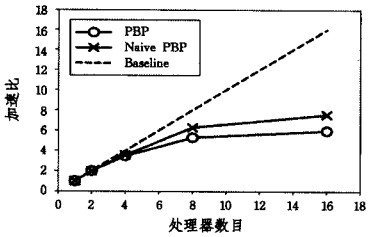

图6 KOS数据集上的加速比

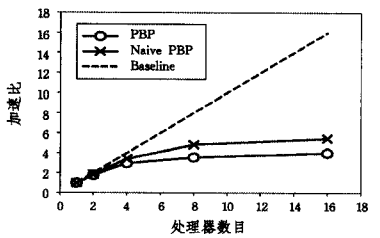

图7 BLOG数据集上的加速比

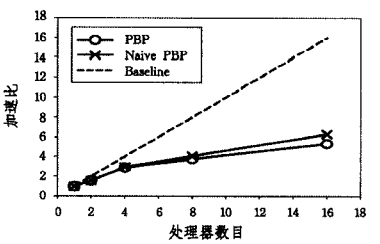

图8 ENRON数据集上的加速比

结束语 本文介绍了并行LDA模型的一般架构，提出朴素并行LDA算法，从计算时间和通信时间两个角度对并行LDA模型进行分析，给出了基于单词影响因子、阈值，以及增大通信间隔的方法来改善并行LDA在时间上的消耗。通过对比实验，选择最优的阈值参数以及通信间隔参数，使得并行LDA的加速比显著提高，并在精度上得到一定保证，精度损失在1%以下。

参考文献

[1] Deerwester S C, Dumais S T, Landauer T K, et al. Indexing by latent semantic analysis[J]. Journal of the American Society for Information Science, 1990, 41(6): 391-407

[2] Hofmann T. Probabilistic latent semantic indexing[C]// Special Inspector General for Iraq Reconstruction. 1999: 50-57

[3] Blei D M, Ng A Y, Jordan M I. Latent dirichlet allocation[C]// Neural Information Processing Systems. 2001: 601-608

[4] Griffiths T L, Steyvers M. Finding scientific topics [J]. Proceedings of the National Academy of Sciences, 2004, 101(1): 5228-5235

[5] Zeng J, Cheung W K, Liu J. Learning topic models by belief propagation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013, 35(5): 1121-1134

[6] 都志辉，等. 高性能计算并行编程技术——MPI并行程序设计[M]. 北京：清华大学出版社，2001
Du Zhi-hui, et al. High performance computing parallel programming technology——MPI parallel program design[M]. Peking: Tsinghua University Press, 2001

[7] Newman D, Asuncion A U, Smyth P, et al. Distributed inference for latent dirichlet allocation[C]// Neural Information Processing Systems. 2007

[8] Asuncion A U, Smyth P, Welling M. Asynchronous distributed learning of topic models[C]// Neural Information Processing Systems. 2008: 81-88

[9] Wang Y, Bai H, Stanton M, et al. Plda: Parallel latent dirichlet allocation for large-scale applications[C]// AAIM. 2009: 301-314

[10] Liu Z, Zhang Y, Chang E Y, et al. Plda+: Parallel latent dirichlet allocation with data placement and pipeline processing [J]. ACM TIST, 2011, 2(3): 1-18

[11] Zhai K, Boyd-Graber J L, Asadi N, et al. lda: a flexible large scale topic modeling package using variational inference in mapreduce[C]// WWW. 2012: 879-888

[12] Yan F, Xu N, Qi Y. Parallel inference for latent dirichlet allocation on graphics processing units[C]// Neural Information Processing Systems. 2009: 2134-2142

(上接第242页)

[3] Wang L, Jia H D, Li J. Training Robust Support Vector Machine with Smooth Ramp Loss in the Primal [J]. Neurocomputing, 2008, 71: 3020-3025

[4] Lin C F, Wan g S D. Fuzzy Support Vector Machine [J]. IEEE Transactions on Neural Networks, 2002, 13(2): 464-471

[5] Xu Yi-tian, Liu Chun-mei. A rough margin-based one class support vector machine[J]. Neural Comput & Applic, 2013, 22: 1077-1084

[6] Bishop C M. Pattern Recognition and Machine Learning [M]. Cambridge: Springer, 2007: 291-320

[7] 王磊，杨一帆，周启海. 粗糙one-class支持向量机[J]. 计算机科学，2009，36(9)：242-245
Wang Lei, Yang Yi-fan, Zhou Qi-hai. Rough Set based One-class Support Vector Machine[J]. Computer Science, 2009, 36(9): 242-245

[8] 秦玉平，王祎，伦淑娴，等. 基于超椭球支持向量机的兼类文本分类算法[J]. 计算机科学，2013，40(11A)：98-100
Qin Yu-ping, Wang Yi, Lun Shu-xian, et al. Multi-label Text Classification Algorithm Based on Hyper Ellipsoidal SVM[J]. Computer Science, 2013, 40(11A): 98-100

[9] Jeong Y-S, Kang I-H, Jeong M-K, et al. A New Feature Selection Method for One-Class Classification Problems [J]. IEEE Transactions on Systems, Man, and Cybernetics—part c: Applications and Reviews, 2012, 42(6): 1500-1509

[11] Asharaf S, Shevade S K, Murty M N. Rough support vector clustering[J]. Pattern Recogn, 2005, 38(10): 1779-1783

[12] Xu Y. Classification algorithm based on feature selection and samples selection [J]. 6th International Symposium on Neural Networks(ISNN 2009). 2009

[13] Bicego M, Figueiredo M A T. Soft clustering using weighted one-class support vector machines [J]. Pattern Recognition, 2009, 42(1): 27-32

[14] Schölkopf B, Platt J, Shawe-Taylor J, et al. Estimating the support of a high-dimensional distribution [J]. Neural Comput, 2001, 13(7): 1443-1471

[15] Junejo I N, Bhutta A A, Foroosh H. Single-class SVM for dynamic scene modeling [J]. Signal, Image and Video Processing, 2013, 7(1): 45-52

[16] Shawe-Taylor J, Cfistianini N. Kernel methods for pattern analysis [M]. Cambridge: Cambridge University Press, 2004