doi:10.3969/j.issn.1005-3026.2015.06.003

分布式环境中基于核函数的极限学习机

赵相国,毕 鑫,张 祯,杨洪波
(东北大学 信息科学与工程学院,辽宁 沈阳 110819)

摘 要:针对海量数据规模下的集中式核函数极限学习机的性能问题,将基于核函数的极限学习机扩展到云计算技术框架下,提出了基于MapReduce的分布式核函数极限学习机MR-KELM.该算法将分布式径向基核函数计算出的核函数矩阵进行分布式矩阵分解,并通过分布式矩阵向量乘法得到分类器输出权重,减小了网络通讯和数据交换代价.实验结果表明,MR-KELM算法能够在不影响基于核函数的极限学习机的计算理论的前提下,具有较好的可扩展性和分类训练性能.

关 键 词:极限学习机;核函数;分类;分布式;MapReduce
中图分类号:TP 311.13 **文献标志码**:A **文章编号**:1005-3026(2015)06-0769-04

Kernelized Extreme Learning Machine in Distributed Environment

ZHAO Xiang-guo, *BI Xin*, *ZHANG Zhen*, *YANG Hong-bo*
(School of Information Science & Engineering, Northeastern University, Shenyang 110819, China. Corresponding author: ZHAO Xiang-guo, E-mail: zhaoxiangguo@mail.neu.edu.cn)

Abstract: With the exponentially increasing volume of training data, the performance of centralized ELM with kernels suffers due to large matrix operations. A distributed algorithm named MapReduce based kernelized ELM (MR-KELM) was proposed, which realized an implementation of ELM with kernels on MapReduce in the cloud. The kernel matrix generated by distributed radial basis function was decomposed and then the output weights by distributed multiplication of matrix and vector were calculated by the proposed algorithm. Communications and data exchanges in distributed matrix operations were reduced and good scalability was achieved by MR-KELM. Extensive experiments on synthetic datasets were conducted to verify the training performance and scalability of MR-KELM. Experimental results showed that MR-KELM was effective and efficient for massive learning applications.

Key words: ELM(extreme learning machine); ELM with kernels; classification; distributed; MapReduce

随着互联网技术的迅猛发展,大量网络应用和服务不断地产生了海量的数据,使得传统分类算法遇到了极大的挑战.其中基于核函数的极限学习机(kernelized extreme learning machine, KELM)[1]主要的计算量是矩阵求逆和矩阵相乘等运算,随着训练样本规模的增加,庞大的核函数矩阵会严重影响矩阵运算效率.作为海量数据处理的代表技术,MapReduce[2]计算模型不天然支持全局信息交换和迭代,无法直接部署核函数极限学习机.因此,本文针对海量数据规模下核函数极限学习机的训练效率问题,提出了基于MapReduce的分布式核函数极限学习机(MapReduce based kernelized ELM,MR-KELM).

1 核函数极限学习机

极限学习机(extreme learning machine, ELM)[3-4]是一种广义单隐层前馈神经网络,在

收稿日期:2014-04-16
基金项目:国家自然科学基金资助项目(61272181,61202087);武汉大学开放基金资助项目(SKLSE2012-09-40).
作者简介:赵相国(1973-),男,吉林通化人,东北大学副教授.

随机生成输入权值和偏置后,通过矩阵计算得到输出权值,从而确定分类器学习模型. 相比传统的前馈后传神经网络,极限学习机的训练速度极快,泛化性能更好,实现更容易. Huang 等[5]指出极限学习机和支持向量机(support vector machine, SVM)[6]的优化角度是一致的,并提出了核函数极限学习机 KELM,将最小二乘支持向量机 LS-SVM[7]中的核函数引入极限学习机中,使得该优化问题无需关注输入权值、输出权值、隐藏层节点数、激活函数及偏置,并且相比最小二乘支持向量机具有更快的训练速度和泛化性能.

极限学习机的输出权重矩阵 $\boldsymbol{\beta}$ 通过矩阵计算公式 $\boldsymbol{\beta}=\boldsymbol{H}^{\dagger}\boldsymbol{T}$ 计算,其中 $\boldsymbol{H}^{\dagger}$ 是特征映射矩阵 $\boldsymbol{H}$ 的广义逆矩阵. $\boldsymbol{H}$ 是极限学习机的随机特征映射矩阵,它将输入样本映射到 L 维隐藏层空间上,L 是隐藏层节点数,$h(\boldsymbol{x})=[h_1(\boldsymbol{x}),\ldots,h_L(\boldsymbol{x})]$ 是输入 $\boldsymbol{x}$ 经过隐藏层映射后的输出. 由此,极限学习机的输出函数如下:

$$f(\boldsymbol{x})=h(\boldsymbol{x})\boldsymbol{\beta}=h(\boldsymbol{x})\boldsymbol{H}^{\mathrm{T}}\left(\frac{\boldsymbol{I}}{C}+\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}}\right)\boldsymbol{T}. \quad (1)$$

传统极限学习机的隐藏层输出函数 $G(\boldsymbol{w},\boldsymbol{x},b)$ 已知时,$h(\boldsymbol{x})$ 的计算公式如下:

$$h(\boldsymbol{x})=[G(\boldsymbol{w}_1\cdot\boldsymbol{x}+b_1),\cdots,G(\boldsymbol{w}_L\cdot\boldsymbol{x}+b_L)]. \quad (2)$$

核函数极限学习机引入核函数代替特征矩阵运算 $\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}$,其输出函数如下:

$$f(\boldsymbol{x})=\begin{bmatrix}K(\boldsymbol{x},\boldsymbol{x}_1)\\ \vdots\\ K(\boldsymbol{x},\boldsymbol{x}_N)\end{bmatrix}\left(\frac{\boldsymbol{I}}{C}+\boldsymbol{\Omega}_{\mathrm{ELM}}\right)^{-1}\boldsymbol{T}. \quad (3)$$

其中:$\boldsymbol{\Omega}_{\mathrm{ELM}\,i,j}=K(\boldsymbol{x}_i,\boldsymbol{x}_j)=h(\boldsymbol{x}_i)\cdot h(\boldsymbol{x}_j)$;$\frac{\boldsymbol{I}}{C}$ 表示将该对称矩阵的对角线元素加上偏置常量 $1/C$,从而增加稳定性和泛化性能.

2 分布式核函数极限学习机

核函数极限学习机的训练阶段包括 3 个计算:1)核函数矩阵 $\boldsymbol{\Omega}$ 的计算;2)$\boldsymbol{\Omega}$ 逆矩阵的计算;3)矩阵和向量乘法. 计算仅与训练数据有关,并且随着样本个数 N 的增大,$\boldsymbol{\Omega}$ 矩阵的运算代价急剧增加,单台机器有限的内存资源使得训练效率大大降低.

2.1 分布式特性分析

MapReduce[2]是一个批处理分布式计算框架,支持海量规模数据的高效分布式处理. MapReduce 主要包含 Map, Shuffle 和 Reduce 三个阶段. 其中:1)Map 对于输入数据进行并行处理,产生中间结果形式的 < key, value > 键值对;2)Shuffle 将 Map 产生的中间结果根据 key 进行排序并分组传输至相应的 Reduce;3)Reduce 整合每个 key 的 value 列表进行进一步聚合处理并生成最终结果.

根据核函数极限学习机的计算理论,其基于 MapReduce 的并行化主要需要考虑以下两方面:

1)数据分片. 在云计算技术体系下,数据根据一定的规则进行分片并存储在分布式文件系统中. 核函数极限学习机的主要计算代价是核函数矩阵运算,任意两个训练数据集样本之间都会产生计算过程,因此可以将数据集按列分片,以列编号和列数据作为 Map 的输入键值对.

2)计算模型. 极限学习机中的伪逆矩阵运算涉及矩阵及其转置相乘,可看作矩阵元素之间内部运算,因此天然支持在 MapReduce 框架上进行扩展. 核函数极限学习机中的核矩阵运算没有这种特性,其求逆过程无法直接利用传统极限学习机的计算方法. 此外,目前还没有能在不修改 MapReduce 框架的前提下进行分布式矩阵求逆的方法,矩阵之间的乘法也涉及大量的中间结果交换和循环迭代,这与 MapReduce 模型的理念相冲突. 因此如何将核函数极限学习机的计算过程在 MapReduce 上最大程度地并行化是研究重点.

2.2 分布式核函数极限学习机

根据奇异值分解定理,式(3)中需要求解逆矩阵的部分首先被分解为 3 个矩阵,即

$$\left(\frac{\boldsymbol{I}}{C}+\boldsymbol{\Omega}\right)=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}. \quad (4)$$

其中:$\boldsymbol{\Sigma}$ 是对角矩阵;$\boldsymbol{U}$ 和 $\boldsymbol{V}$ 是正交矩阵. 因此,其逆矩阵可以按式(5)计算:

$$\left(\frac{\boldsymbol{I}}{C}+\boldsymbol{\Omega}\right)^{-1}=(\boldsymbol{V}^{\mathrm{T}})^{-1}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^{-1}=\boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^{\boldsymbol{T}}. \quad (5)$$

由此可得核函数极限学习机的输出权重

$$\boldsymbol{w}=\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{T}. \quad (6)$$

式(6)可分为 $\boldsymbol{V}\boldsymbol{\Sigma}$ 和 $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{T}$ 两个部分,由于对角矩阵 $\boldsymbol{\Sigma}$ 按照对角线元素存储为 $N\times 1$ 的向量,因此这两个乘积的计算形式都可视为一个 $N\times N$ 的矩阵乘以一个 $N\times 1$ 的向量,从而可以被整合在同一个 MapReduce 计算逻辑中.

在 RBF 核函数的分布式计算方面,如图 1 所示,按列存储的训练样本以 < colID, column > 键值对作为 Map 的输入,其中的列数据 column 首先被转换成向量 $\boldsymbol{X}$. 向量 $\boldsymbol{X}$ 中的任意两个元素相减并平方,即两个样本向量之间某列的二范数平方和,并以 $<<i,j>,\boldsymbol{X}(i,j)>$ 键值对的形式发

送给 Reduce. Reduce 将相同 $<i,j>$ 对应的部分和进行进一步计算，得到最终矩阵的结果. 算法 1 给出了核函数矩阵的分布式计算的详细过程.

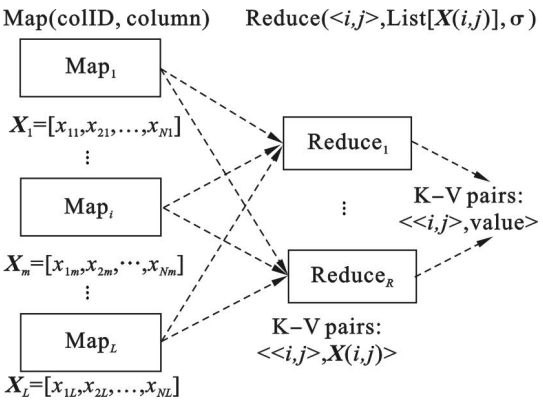


图 1　分布式核函数计算

Fig. 1　Distributed kernel calculation

算法 1　分布式核函数

输入：训练数据集

输出：核函数矩阵

```
1. Function Map(colID,column)
2.     Vector X = parse(column);
3.     For i = 1 to X.size() Do
4.         For j = 1 to i Do
5.             X(i,j)=(X[i]-X[j])×(X[i]-X[j]);
6.             emit(<i,j>,X(i,j));
7.         EndFor
8.     EndFor
9. EndFunction
10. Function Reduce(<i,j>,list[X(i,j)],σ)
11.     value = list[X(i,j)].sum();
12.     value = exp(-value/(2σ²));
13.     emit(<i,j>,value);
14.     emit(<j,i>,value);
15. EndFunction
```

在计算出核函数矩阵后，首先要将核函数矩阵进行奇异值分解. 这一步通过 Mahout 项目中的相关 API 可直接实现. 在获取核函数矩阵的 3 个分解矩阵后，对角矩阵 $\boldsymbol{\Sigma}$ 和类标签向量 $\boldsymbol{T}$ 均以向量的形式存储在 DistributedCache 中作为全局输入. 随后开启新一轮 MapReduce 任务，对上述中间结果矩阵进行相乘运算，如图 2 所示.

由于 $\boldsymbol{V\Sigma}$ 和 $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{T}$ 的计算细节不完全一致，因此需要在 Map 中对 $\boldsymbol{V},\boldsymbol{U}$ 矩阵的计算过程通过标记 matID 进行区别. 算法中 $\boldsymbol{Z}$ 矩阵泛指 $\boldsymbol{U}$ 或 $\boldsymbol{V}$ 矩阵.

对于 $\boldsymbol{V\Sigma}$ 的情况，由于向量 $\boldsymbol{\Sigma}$ 本质上是 $N\times N$ 的对角矩阵，其计算的结果是一个 $N\times N$ 的矩阵. 因此 $\boldsymbol{Z}[i]\boldsymbol{S}[i]$ 是矩阵相乘的最终元素；对于 $\boldsymbol{UT}$ 的情况，每个 $\boldsymbol{Z}[i]\boldsymbol{T}[i]$ 是结果矩阵中元素的部分和，所有同一个元素的部分和都被相同的 $<$matID，rowID，colID$>$ 标记. 由于 $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{T}$ 的结果是一个 $N\times 1$ 的向量，因此无需对结果所属的列进行标记，colID 被设为 null. Reduce 将所有属于结果向量中的同一个元素的中间结果相加，得到矩阵相乘最终结果向量. 具体过程如算法 2 所示.

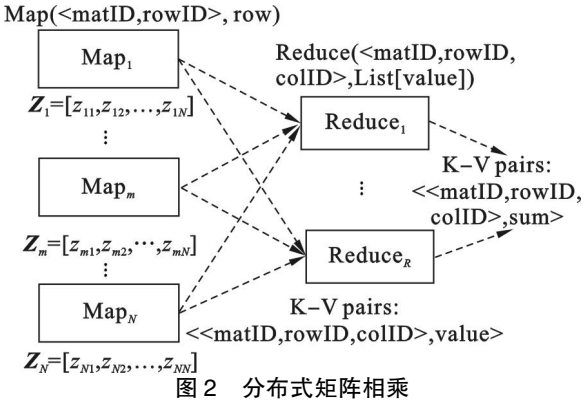


图 2　分布式矩阵相乘

Fig. 2　Distributed matrix multiplication

算法 2　MR－KELM

输入：核函数矩阵

输出：输出权重

```
1. Function Map(<matID,rowID>,row)
2.     Vector S = DistributedCache.get(SIGMA);
3.     Vector T = DistributedCache.get(T);
4.     Vector Z = parse(row);
5.     For i = 1 to Z.size() Do
6.         If matID = = "U" Then
7.             colID = null;
8.         Else If matID = = "V" Then
9.             colID = i;
10.        EndIf
11.        value = Z[i]T[i];
12.        emit(<matID,rowID,colID>,value);
13.    EndFor
14. EndFunction
15. Function Reduce(<matID,rowID,colID>,
list[value])
16.     sum = list[value].sum();
17.     emit(<matID,rowID,colID>, sum);
18. EndFunction
```

由此，MR－KELM 已计算出极限学习机的输出权重矩阵，从而确定了整个分类器模型.

3　实验分析

本文的实验环境包括 1 台 Master 节点和 8

台 Slave 节点组成的 Hadoop 0.20.2 集群,每台机器配有 2.66 GHz 的英特尔酷睿四核处理器、4 GB内存和 CentOS 5.6 操作系统. MR – KELM 中的 SVD 矩阵分解通过 Apache Mahout 实现.

图 3 展示了在 0.92,1.5,2.0 和 2.5 GB 四种不同数据集规模下,MR – KELM 算法各个阶段所消耗的时间. 随着样本数 N 的增加,核函数矩阵规模呈平方倍增长,因此核函数矩阵的计算时间理论上同样应为平方倍增长;此外,矩阵计算涉及嵌套循环,因此矩阵计算的时间理论上也呈平方倍增长. 实验结果表明,MR – KELM 有效地控制了数据集增长时运算代价的平方倍增长,训练阶段的时间增长幅度较为理想.

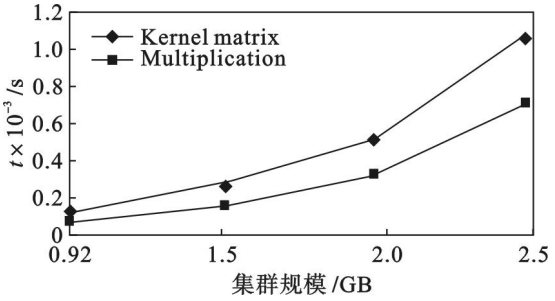
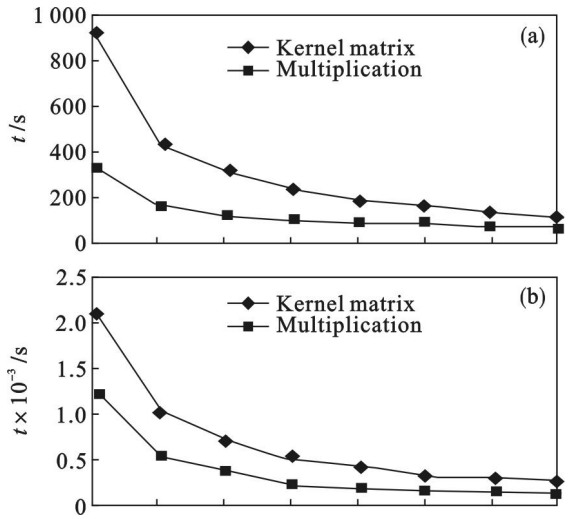


图 3 数据集大小与 MR – KELM 性能

Fig. 3 Time of MR-KELM on varied datasets

图 4 是 MR – KELM 算法在不同集群规模下的运行时间. 从图中可以看出,由于核函数的并行计算和矩阵相乘需要分别启动各自的 MapReduce 任务按照先后顺序执行,当 Slave 节点数为 1 时,需要消耗大量时间进行训练;然而随着集群的计算节点增加,MR – KELM的训练时间急剧减少,这反映出MR – KELM良好的加速比和扩展性能.



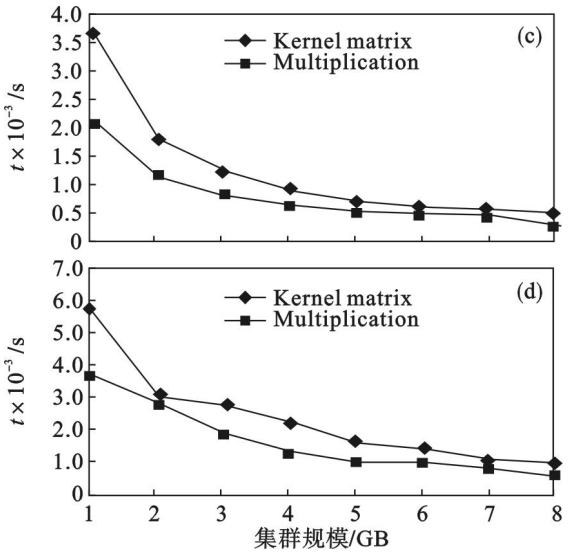


图 4 集群大小与 MR – KELM 运行时间

Fig. 4 Time of MR-KELM on varied slaves number

(a)—0.92 GB 数据集;(b)—1.5 GB 数据集;(c)—2 GB 数据集;(d)—2.5 GB 数据集.

4 结 论

本文针对大规模训练数据规模下核函数极限学习机性能下降的问题,提出了基于 MapRedcue 的分布式核函数极限学习机 MR – KELM,实现了分布式核函数矩阵的计算和矩阵向量相乘. 实验结果表明,MR – KELM 在不影响核函数极限学习机理论的前提下,具有良好的加速比和扩展性.

参考文献:

[1] Huang G B, Zhou H M, Ding X J, et al. Extreme learning machine for regression and multiclass classification[J]. *IEEE Transactions on Systems, Man, and Cybernetics*, 2012, 42(2): 513 – 529.

[2] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[C]// Proceedings of the 6th Symposium on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2004: 137 – 149.

[3] Huang G B, Zhu Q Y, Siew C K. Extreme learning machine: a new learning scheme of feedforward neural networks[C]// Proceedings of the 2004 IEEE International Joint Conference on Neural Networks. Piscataway: IEEE, 2004: 985 – 990.

[4] Huang G B, Zhu Q Y, Siew C K. Extreme learning machine: theory and applications[J]. *Neurocomputing*, 2006, 70(1/2/3): 489 – 501.

[5] Huang G B, Ding X J, Zhou H M. Optimization method based extreme learning machine for classification [J]. *Neurocomputing*, 2010, 74(1/2/3): 155 – 163.

[6] Cortes C, Vapnik V. Support vector networks[J]. *Machine Learning*, 1995, 20(3): 273 – 297.

[7] Suykens J A K, Vandewalle J. Least squares support vector machine classifiers[J]. *Neural Processing Letters*, 1999, 9(3): 293 – 300.