基于MapReduce的特征选择并行化研究

陆 江 李 云
(南京邮电大学计算机学院 南京 210023)

摘 要 特征选择已经成为一种对高维数据进行预处理的必不可少的手段。随着数据规模的爆炸性增长,传统的特征选择算法已经不能满足当前高维大规模数据的处理要求。采用Google的MapReduce编程模型,设计了一种分布式的基于局部学习的特征选择算法D-logsf。在多个现实和合成数据集上的实验表明,分布式特征选择算法D-logsf具有较好的可靠性,且与传统特征选择算法Logsf相比可以获得接近线性的加速比,同时可以有效处理大规模数据集。

关键词 特征选择,局部学习,分布式,MapReduce
中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.8.009

MapReduce Based Feature Selection Parallelization

LU Jiang LI Yun
(College of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China)

Abstract Feature selection has become a necessary preprocessing procedure for high-dimensional data. With the explosive growth of data size, the traditional feature selection algorithm can not meet the current requirements of processing large-scale and high-dimensional data. Resorting to Google's MapReduce programming model, we designed a distributed local learning-based feature selection algorithm D-logsf. Experiments were conducted on several real and synthesis data sets. The results show that the D-logsf algorithm is correct and has good reliability. Compared with traditional feature selection algorithm Logsf, D-logsf can obtain approximate linear speedup. Moreover, D-logsf can effectively handle large-scale data set.

Keywords Feature selection, Local learning, Distributed, MapReduce

1 引言

随着信息技术和生物技术的飞速发展,大量的数据相应产生。他们具有样本数量大、特征维度高等特点,因此利用特征选择算法对其进行预处理时,所面临的计算问题日益加剧[1]。面对高维大规模数据,传统的特征选择算法已经不能够在合理的时间内得到需要的模型。因此,需要研究新的有效处理大规模数据的特征选择方法。目前处理大规模数据最有效的方法之一就是分布式并行计算[2],而最具代表性的分布式计算框架当属Google工程师J. Dean等提出的MapReduce编程模型[3]。因此将MapReduce编程模型与传统特征选择算法相结合,实现分布式地特征选择,以有效处理大规模数据。据我们所知,目前对分布式特征选择算法的研究并不是很多。开源项目[4]依据文献[5]开发了部分分布式机器学习算法,但是其中并没有特征选择算法的相关研究。文献[1]提出一种并行化大规模数据的特征选择算法,文献[6]提出基于Variance Preservation的并行化特征选择,这些都能够很好地处理大规模数据。而本文将更多关注基于局部学习的特征选择算法并行化。

MapReduce是一个分布式计算框架,主要由两部分组成:编程模型和运行时环境。其中,编程模型为用户提供了非常易用的编程接口,用户只需要编写Mapper函数和Reducer函数就可以实现一个分布式程序。key/value是MapReduce编程模型的基本数据结构。Mapper函数将所有输入的key/value对处理为不同数量的中间key/value对。Reducer函数处理具有相同key的values,最终输出key/value对。MapReduce编程模型的数据处理流程如图1所示。MapReduce能够解决的问题有一个共同点:任务可以被分解为多个子问题,且这些子问题相对独立,彼此之间不会有牵制,待并行处理完这些子问题后,原始任务便被解决[7]。在实际应用中,这类问题特别庞大,谷歌在文献[3]中提到了MapReduce的一些典型应用,包括分布式grep、URL访问频率统计、Web连接图反转、倒排索引构建、分布式排序等。学术界也掀起了学习研究使用MapReduce解决大数据问题的热潮。文献[5]详细阐述了如何在多核上基于MapReduce实现常见的机器学习算法,例如:K-mean、SVM、Neural Network和Naive Bayes等等。文献[8]使用MapReduce解决了文档相似度的计算问题,文献[9]应用MapReduce编程模型并行化Top-K Similiarity Join问题,文献[10]使用MapReduce解决模式匹配问题。

到稿日期:2014-05-23 返修日期:2014-07-08 本文受江苏省自然科学基金(BK20131378),国家自然科学基金(61105082)资助。
陆 江(1990—),男,硕士,主要研究方向为机器学习、云计算;李 云(1974—),男,博士,教授,主要研究方向为机器学习、模式识别,E-mail:liyun@njupt.edu.cn。

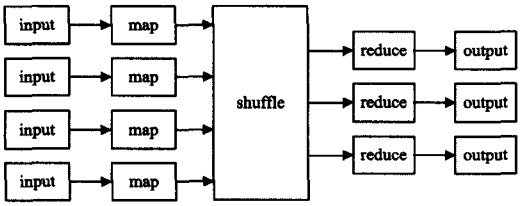


图 1　MapReduce 编程模型的数据处理流程

本文从高维大规模数据出发，分析了基于局部学习的特征选择算法 Logsf，并对 Logsf 进行并行化，提出了能够处理大数据的分布式特征选择算法 D-logsf。第 2 节从局部学习的角度分析特征选择算法 Logsf；第 3 节提出分布式算法 D-logsf 以及分析 Hadoop 平台对 D-logsf 算法的影响因素；第 4 节在现实和人工合成数据集上进行实验，验证所提算法的可靠性和性能；最后总结全文。

2　基于局部学习的特征选择算法 Logsf

局部学习特征选择算法的主要思想是将任一复杂的非线性问题转化为一组局部线性问题。对于存在大量不相关特征的数据集，采用基于局部学习的特征选择算法可以获得较理想的结果[11]。假设训练样本集 $D=\{X,Y\}=\{x_i,y_i\}_{i=1}^N$，其中 x_i 是第 i 个训练样本，y_i 是与其对应的标记，N 代表训练样本数量。每个训练样本是由 d 维矢量组成 $x_i=(x_{i1},x_{i2},\cdots x_{id})\in R^d$。则样本 x_i 的基于局部学习的损失函数定义如下：

$$L(w,x_i)=\log(1+\exp(-w^{\mathrm{T}}z_i)) \tag{1}$$

其中，$z_i=|x_i-nearmiss(x_i)|-|x_i-nearhit(x_i)|$。$||$是一个绝对值运算符，$nearhit(x_i)$代表与样本 x_i 标记相同且距离最近的样本，而 $nearmiss(x_i)$表示与样本 x_i 距离最近且标记不同的样本，w 代表特征权重。通过最小化损失函数找到最佳特征权重 w，使得样本 x_i 与其最近邻样本 $nearhit(x_i)$之间的距离小于样本 x_i 与 $nearmiss(x_i)$之间的距离，以符合目标函数优化要求。特征选择的评价准则定义为所有样本的损失函数之和：

$$e(w)=\frac{1}{N}\sum_{i=1}^{N}L(w,x_i)=\frac{1}{N}\sum_{i=1}^{N}\log(1+\exp(-w^{\mathrm{T}}z_i)) \tag{2}$$

对于式(2)，应该考虑使用有效方法最小化损失函数以便得到合适的能量函数[12]，从而获取最佳的参数矢量 w^*。由于 $e(w)$的平滑特性，采用最优化方法中的梯度下降算法寻找其最佳权重 w^*，使得评价函数值最小。而矢量 w 表示特征权重，从而可以根据获取的特征权重大小对原始特征进行排序，特征权重值大的为重要特征，反之为不重要特征。

梯度下降算法是一种最优化算法，通常也称为最速下降算法，常用于机器学习和人工智能中，用来递归性地逼近最小偏差模型[14]。为了快速得到权重值，采用随机梯度下降算法，其迭代公式如下：

$$w=w-\alpha\frac{\partial L(w,x_i)}{\partial w} \tag{3}$$

其中，w 表示特征的权重，α 表示步长，∂表示求偏导符号。将式(1)代入式(3)得：

$$w=w-\alpha\frac{\partial\{\log(1+\exp(-w^{\mathrm{T}}z_i))\}}{\partial w} \tag{4}$$

化简为：

$$w=w-\alpha*\frac{-1}{1+\exp(w^{\mathrm{T}}z_i)}z_i \tag{5}$$

算法 1 给出了基于局部学习的特征算法 Logsf 的伪代码。

算法 1　特征选择算法 Logsf 的伪代码

输入：训练集 $D=\{X,Y\}=\{x_i,y_i\}_{i=1}^N$，步长 α

输出：特征权重 w^*

1. 初始化特征权重 $w=(1,1,\cdots,1)\in R^d$
2. for all $i=\{1\cdots N\}$
3. 　对于样本 x_i，计算其 nearhit(x_i)，nearmiss(x_i)
4. 　计算 $z_i=|x_i-\text{nearmiss}(x_i)|-|x_i-\text{nearhit}(x_i)|$
5. 　依据公式 $w=w-\alpha\frac{\partial L(w,x_i)}{\partial w}$更新特征权重 w
6. end for
7. 得到的特征权重 w 就是梯度下降算法的最优值 w^*
8. 输出 w^*

3　分布式特征选择算法 D-logsf

3.1　曼哈顿距离计算的分布式算法

为了获得最佳的特征权重值，采用梯度下降算法来最小化评价函数。在式(2)中需要计算 $nearhit(x_i)$与 $nearmiss(x_i)$。因此，每个样本需要与其他的样本进行曼哈顿距离的计算，然后依据类别得到对应的 $nearhit(x_i)$与 $nearmiss(x_i)$。设有样本集 $D=\{X,Y\}=\{x_i,y_i\}_{i=1}^N$，样本 x_i 与 x_j 之间的曼哈顿距离计算定义如下：

$$d(x_i,x_j)=\sum_{k=1}^{d}|x_{ik}-x_{jk}| \tag{6}$$

其中，$||$代表绝对值，x_{ik}代表第 i 条样本的第 k 个特征值，x_{jk}代表第 j 条样本的第 k 个特征值，d 为特征维数。我们设计一种有效的并行方式来解决样本间曼哈顿距离的计算，主要有 2 个 MapReduce 作业。

1)建立索引：建立一个标准的倒排索引。其中每项包括特征的编号 $f_j(j=1,\cdots,d)$，样本的编号 $id_i(i=1,\cdots,N)$和对应样本的对应特征值 $x_{ij}(i=1,\cdots,N,j=1,\cdots,d)$。通过 Map 函数输出 key/value 对，其中 key 为特征的编号，value 为样本的编号与对应的特征值的组合。MapReduce 运行时依据 key 值自动合并具有相同 key 的 value，Reduce 将这些 key/value 输出到磁盘。

2)距离计算：步骤 1)的输出作为该步骤的输入。输入键值对中 key 为特征编号 $f_j(j=1,\cdots,d)$，value 为$(\langle id_1,x_{1j}\rangle\langle id_2,x_{2j}\rangle\cdots\langle id_n,x_{Nj}\rangle)$。通过 Map 函数对 value 中样本编号尽可能地组合，对应位的特征值相减取绝对值，输出类似$(\langle id_1,id_2\rangle,|x_{1j}-x_{2j}|)$的键值对。Reduce 函数将具有相同 key 的 value 进行相加运算就得到最终的样本间的相似度。

3.2　梯度下降的分布式算法

算法 2 给出分布式随机梯度下降算法的伪代码，其中 $D=\{X,Y\}=\{x_i,y_i\}_{i=1}^N$为训练样本集，$\alpha$ 为步长，k 为机器节点数量。首先为每个节点随机地分配 $T=\lfloor N/k\rfloor$个样本，然后每个节点并行学习。迭代计算公式如下：

$$w^l=w^l-\alpha*\frac{\partial L(w^l,x_t^l)}{\partial w} \tag{7}$$

其中，w^l 表示第 l 个节点上 T 个样本的特征权重，$\frac{\partial L(w^l,x_t^l)}{\partial w}$表示第 l 个节点上第 t 个样本的损失函数对权重 w 求偏导。待所有的计算节点训练完毕，求 k 个节点特征权重的平均值

作为最终的特征选择结果。

$$w=\frac{1}{k}\sum_{l=1}^{k}w^l \tag{8}$$

得到的权重 w 就是最优权重值 w^*。

算法 2 分布式随机梯度下降算法的伪代码

输入：训练集 $D=\{X,Y\}=\{x_i,y_i\}_{i=1}^{N}$，步长 α，机器数 k

输出：w^*

1. $T=\lfloor N/k \rfloor$，初始化特征权重 $w=(1,1,\cdots,1)\in R^d$
2. 随机为每个节点分配 T 个样本
3. for all $l\in\{1\cdots k\}$ //这一步是 k 个节点并行计算
4. 第 l 个节点上的数据随机读取
5. for all $t\in\{1\cdots T\}$
6. 读取第 l 个节点的第 t 个样本 x_t^l
7. 对于样本 x_t^l，计算其 $nearhit(x_t^l)$，$nearmiss(x_t^l)$
8. 计算 $z_t^l=|x_t^l-nearmiss(x_t^l)|-|x_t^l-nearhit(x_t^l)|$
9. 依据 $w^l=w^l-\alpha*\frac{\partial L(w^l,x_t^l)}{\partial w}$，更新特征权重 w^l
10. end for
11. end for
12. 集成所有节点的特征加权结果 $w=\frac{1}{k}\sum_{l=1}^{k}w^l$
13. $w^*=w$
14. 输出 w^*

3.3 时间复杂度分析

为了方便分析，迭代算法的时间复杂度只计算一次，因此它们的实际运行时间可能更长。传统特征选择算法 Logsf 的计算时间主要由两部分组成：1)样本间距离计算；2)梯度下降算法计算。因此 Logsf 总的时间复杂度为 $O(N^2d+Nd^2)$。分布式特征选择算法 D-logsf 将样本间曼哈顿距离计算分成建立索引和距离计算两个作业，建立索引阶段的时间复杂度为 $O(N*d/k)$，距离计算的时间复杂度是 $O(N(N-1)*d^2/2k)$。而批量梯度下降算法的时间复杂度为 $O(N*d^2/k)$，因此基于局部学习的分布式特征选择算法 D-logsf 的时间复杂度为 $O(N*d(d+1)/k+n*d^2(N-1)/2k)$。

3.4 Hadoop 对 D-logsf 算法的影响因素

A)数据块大小：数据在 HDFS 上会被分块存储，Hadoop0.20.2 默认分块的大小是 64M，用户可以依据应用的需要手动设置块的大小。在默认情况下，一个数据文件被分成的块数会直接影响 map 函数的个数[15]。如果设置的数据块太小，会导致启动的 map 函数过多，从而带来不必要的系统开销，同时每个 map 函数处理的数据量又特别小，因此系统的整体性能会下降。如果数据块太大，则会导致 map 函数的任务过重，可能在 map 阶段发生负载不均或内存溢出等意想不到的问题。因此，对于不同的数据集设置合适的块大小很有必要，它可以有效地加快并行计算的速度。

B)节点个数：顾名思义就是集群的规模，即机器数量。Hadoop 具有良好的扩展性，很大程度上与集群的规模可以轻松扩展有很大关系。当数据量增加到一定程度后，现有的集群可能已经无法满足其计算能力和存储能力，这时通过添加机器以达到线性扩展集群能力的目的[7]。

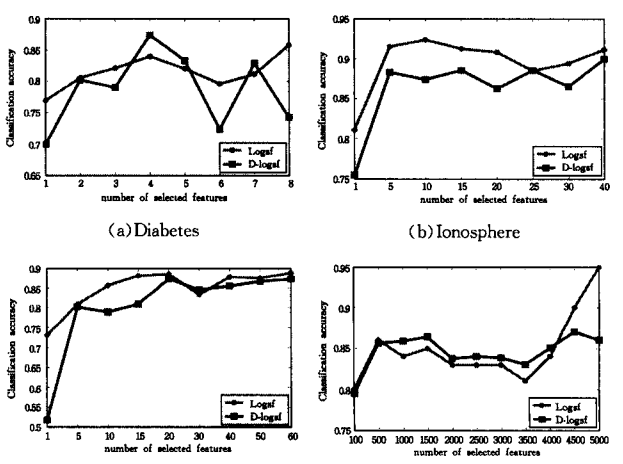
4 实验

实验环境：在一台 IBM 服务器(Intel(R) Xeon(R) E5-26090 @ 2.4GHz，8 核，16GB 内存)虚拟 6 个节点集群，每个节点配置双核 2GB 内存，Apache Hadoop0.20.0，Ubuntu 12.04，JDK1.7。进行 3 组实验：第一组实验利用传统特征选择算法 Logsf 和分布式特征选择算法 D-logsf 得到特征子集模型，验证在不同数据集上的分类准确率，从而验证所提出的分布式算法 D-logsf 的可靠性；其次利用合成数据进行特征选择，进一步验证 D-logsf 算法的可靠性。第二组实验在同等条件下比较 Logsf 和 D-logsf 算法的时间效率。第三组实验主要验证 Hadoop 平台自身的因素对 D-logsf 算法的影响。本实验采用 KNN 分类器，距离使用欧氏距离。数据集 Diabetes、Ionosphere 和 Sonar 来自 UCI 数据库[16]，Gisette 来自文献[17]。为了更加有效地验证分布式特征选择算法的可靠性，人工合成数据集 Rs，数据集 Rs 是依据特征之间的均差与协方差合成的数据集，其中 1－10 维特征为重要特征，且重要性依次递减，11－20 维特征为随机生成的不重要特征，该数据集共 10000 条样本，4000 条 0 类样本，6000 条 1 类样本，数据集的具体信息如表 1 所列。

表 1 实验数据集的描述

数据集	样本数	特征数	类别数	文件大小
Diabetes	768	8	2	22.7KB
Ionosphere	351	34	2	74.6KB
Sonar	208	60	2	85.7KB
Gisette	2000	5000	2	21.5MB
Rs	10000	20	2	3.5MB

第一组实验：首先对原始数据集分别进行 5 次重复抽样，每次抽取 90%的样本构成新的训练样本子集，然后分别使用 Logsf 特征选择算法和 D-logsf 特征选择算法进行特征选择，将得到 5 个不同的特征权重结果，并进行线性相加和排序，得到特征子集。对原始数据集分别进行 5 次重复抽样，每次抽取 50%的样本构成测试样本子集，然后使用上述 Logsf 算法和 D-logsf 算法得到特征子集，最后利用 KNN 分类器得到 5 个分类准确率，以它们的平均分类准确率作为最终的分类准确率。Logsf 算法和 D-logsf 在 4 个数据集上的分类准确率如图 2 所示。依据前面的实验步骤对人工合成的数据集 Rs 利用 Logsf 和 D-logsf 算法进行特征排序，特征排序的结果如表 2 所列。



(a)Diabetes (b)Ionosphere

(c)Sonar (d)Gisette

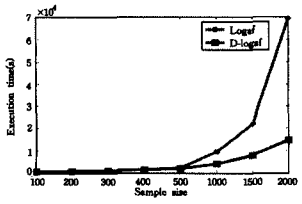
图 2 算法 Logsf 和 D-logsf 在 4 个数据集上的分类准确率

表 2　人工合成数据集 Rs 的特征选择结果

算法	特征选择重要特征排序
Logsf	1,2,3,4,5,6,7,9,8,10,18,13,16,12,15,19,17,20,21,11
D-logsf	1,2,3,4,5,7,6,9,8,12,18,16,19,13,10,11,17,15,20,14

从图 2 可以看出，数据集 Diabetes 使用 Logsf 算法得到稳定的分类准确率，而 D-logsf 算法得到的分类准确率不是很稳定，但其整体分类准确率还是在 Logsf 算法附近波动。在数据集 Ionosphere 上随着特征数量的增加 Logsf 算法和 D-logsf 算法保持一致趋势，虽然同等条件下 D-logsf 的分类准确率比 Logsf 稍微低一些，但是依然可以认为它们具有一致的分类准确率。数据集 Sonar 选择的特征大于 20 时，Logsf 算法和 D-logsf 算法基本保持了一致的分类准确率；而选择的特征小于 20 时，Logsf 算法的分类准确率比 D-logsf 算法要稳定而且更高。数据集 Gisette 选择的特征小于 4000 时，D-logsf 算法和 Logsf 算法的分类准确率保持一致的走势且前者略高于后者；但是维数大于 4000 时，Logsf 算法的分类性能要明显高于 D-logsf。综上所述，可以大体认为 D-logsf 算法与 Logsf 算法在同等条件下得到的特征选择器具有一致的性能。因此，基于局部学习的分布式特征选择算法 D-logsf 具有可靠性。从表 2 可以看出算法 Logsf 与 D-logsf 可以实现重要特征的靠前排序，进一步证明了 D-logsf 算法具有可靠性和正确性。

第二组实验：对数据集 Gisette 分别随机地抽取 100、200、300、400、500、1000、1500、2000 条样本作为训练数据集。分别利用 Logsf 和 D-logsf 依据第一组实验过程进行特征选择，随着数据规模增加，Logsf 和 D-logsf 算法的时间开销实验结果如图 3 所示，对应的加速比如图 4 所示。



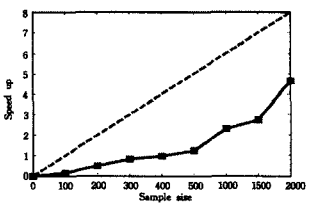
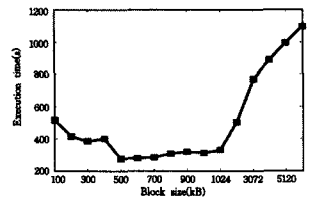


图 3　Logsf 和 D-logsf 在不同规模数据集上的时间开销　　图 4　执行过程的加速比曲线

从图 3 看出，Gisette 数据集小于 500 条样本时，Logsf 比 D-logsf 具有更短的运行时间，这是由于作业启动具有高延迟特性，数据量很少时作业延迟的时间不可被忽略。然而样本大于 500 时，Logsf 的时间开销会急剧增加，而 D-logsf 的时间开销平缓增加。加速比等于同等条件下 Logsf 执行时间和 D-logsf 执行时间的比值。在图 4 中，虚线为参考线，实线为加速比曲线，由此可知，由于样本小于 500 和作业延迟，导致加速比小于 1。样本大于 500 之后，作业延迟不再是主要时间开销，因此加速比急剧增加。可以预测随着样本规模逐渐增加，可以得到一个接近于线性的加速比。

第三组实验：主要验证 3.4 节中阐述的 Hadoop 平台自身的一些因子对 D-logsf 算法的性能影响。随机地从 Gisette 数据集中抽取 500 个样本点作为训练样本，然后分别设置其在 HDFS 上数据块的大小为[100kB，200kB，300kB，400kB，500kB，600kB，700kB，800kB，900kB，1000kB，1024kB，2048kB，3072kB，4096kB，5120kB，6144kB]使用 D-logsf 算法在 Hadoop 集群上分别测试运行时间，实验结果如图 5 所示。还是在数据集 Gisette 上随机抽取 500 个样本点作为训练样本，数据块设置为 500kB，使用 D-logsf 算法分别在 2－6 个节点的集群上运行，观察运行的时间，结果如图 6 所示。



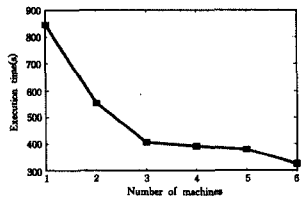


图 5　D-logsf 在 Gisette 数据集上不同数据块的时间开销　　图 6　节点个数不同时的执行时间

从图 5 可以看出，对于训练数据集 Gisette(500 个样本点)，当 BlockSize 从 500kB 增大到 1024kB 时，它们的运行时间都相对较小。而 BlockSize 太大或者太小都会导致集群运行的时间过长，具体原因如 3.4 节所阐述。从实验结果可以得出这样一个结论：对于不同大小数据集存储在 HDFS 上，数据块大小会直接影响 MapReduce 作业的运行效率，同时使得作业能够高效运行的数据块的值应该是一个有限的区间。从图 6 可以明显看出随着集群规模的扩大，数据处理的时间会减少，因而对于大规模的数据，只要增加节点数量就可以很好地处理数据。

结束语　为有效实现大规模数据的特征选择，本文利用 MapReduce 编程模型，实现基于局部学习的特征选择算法的并行化，其主要包括局部学习中寻找近邻样本的并行化和特征选择评价准则的并行优化；同时在多个现实和合成数据集上验证并行算法的有效性和正确性，以及处理大规模数据时算法的可扩展性和可保证的算法运行效率。在未来的工作中，将更深入地分析并行化本身对算法性能的影响以及其它经典特征选择算法的并行化。

参 考 文 献

[1] Singh S, Kubica J, Larsen S, et al. Parallel large scale feature selection for logistic regression[C]// SDM. 2009: 1171-1182

[2] Vernica R, Carey M J, Li C. Efficient parallel set-similarity joins using MapReduce[C]// Proceedings of the International Conference on Management of Data. 2010: 495-506

[3] Dean J, Ghemawat S. Mapreduce: Simplified data processing on large clusters[J]. Operating Systems Design and Implementation, 2004: 137-149

[4] Apache Mahout[OL]. http://mahout.apache.org/

[5] Chu C T, Kim S K, Lin Y A, et al. MapReduce for machine learning on multicore[C]// Advances in Neural Information Processing Systems. 2007: 186-194

[6] Zhao Z, Zhang R, Cox J, et al. Massively parallel feature selection: an approach based on variance preservation[M]// Machine Learning and Knowledge Discovery in Databases. 2012: 237-252

[7] White T. Hadoop: The Definitive Guide [M]. O'Reilly Media, Inc, USA, 2012

[8] Elsayed T, Lin J, Douglas, et al. Pairwise document similarity in large collections with MapReduce[C]// Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies. 2008: 265-268

[9] Kim Y, Shim K. Parallel Top-K Similarity Join Algorithms Using MapReduce[C]// Proceedings of the 2012 IEEE 28th International Conference on Data Engineering. 2012: 510-521

(下转第 81 页)

表 2　在 Yale 人脸库上不同算法的最大识别率及对应的特征维数

		LDA	MMC	MMMC		
				2×2	2×4	4×4
4	正确识别率	68.09	74.76	85.71	85.71	87.62
	鉴别特征维数	7	9	14	9	11
5	正确识别率	71.11	82.53	85.56	85.56	87.78
	鉴别特征维数	3	10	9	8	8
6	正确识别率	77.33	82.67	88.50	88.00	89.67
	鉴别特征维数	13	2	8	9	12

4.3　实验结果分析

(1)表 1、表 2 中的数据均表明无论采用哪种分块方式，实验结果都比原来方法的人脸识别率要高。随着训练样本数目的不同，正确识别率也有所不同。

(2)从图 4 及图 7 中可以看出，对于 3 种不同的分块方式，本文方法的结果均明显优于 LDA 及 MMC 方法。

(3)Modular MMC 方法是 MMC 方法的推广，通过对图像进行分块，再对每一子块抽取得到与直接将 MMC 方法用于原始图像抽取的全局特征相比更具有鉴别性的局部特征，这些局部特征更能表现图像的差异性，从而取得了较好的识别效果。

结束语　本文提出了一种基于图像分块的最大间距准则(MMMC)的人脸识别方法，该方法是对 MMC 方法的改进和推广。与 MMC 方法相比，它的突出优点是在人脸图像局部差异(姿态、表情、光照、细节)存在的情况下，通过分块能抽取到图像的局部特征，这些局部特征更好地体现了图像间的差异性，有利于分类，降低了 MMC 方法提取特征矩阵的维数，有利于加快特征提取速度。实验结果表明，MMMC 方法能得到比全局方法 MMC 更好的特征提取性能。然而在实验中发现，子图像的划分不同，得出的识别性能有所不同，如何找到最佳分块来获得最佳的识别性能还需要进一步的研究。未来将扩展 MMMC 算法到流形学习上，提出多流形分块算法。

参 考 文 献

[1] 马汝成. 基于子空间线性投影的人脸识别方法研究[D]. 南昌：华东交通大学，2008
Ma Ru-cheng. Study of Face Recognition based on Subspace Linear Projection[D]. Nanchang: East China Jiaotong University, 2008

[2] Turk M, Pentland A. Eigenfaces for Recognition [J]. Journal of Cognitive Neuroscience, 1991, 3(1): 71-86

[3] Belhumeur P N, Hespanha J P, Kriengman D J. Eigenfaces vs. Fisherfaces: recognition using class specific linear projection [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997, 19(7): 711-720

[4] Raudys S J, Jain A K. Small sample size effects in statistical pattern recognition: recommendations for practitioners [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1991, 13(3): 252-264

[5] Swets Daniel L, Weng J. Using discriminant eigenfeatures for image retrieval [J]. IEEE Trans. Pattern Anal. Machine Intell (S0162-8828), 1996, 18(8): 831-836

[6] Belhumeur Peter N, Hespanha Joao P, Kriegm David J. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection [J]. IEEE Trans. Pattern Anal. Machine Intell (S0162-8828), 1997, 19(7): 711-720

[7] Chen L, Liao H, Ko M, et al. A new LDA-based face recognition system which can solve the small sample size problem[J]. Pattern Recognition, 2000, 33(10): 1713-1726

[8] Yu H, Yang J. A direct LDA algorithm for high-dimensional data-with application to face recognition[J]. Pattern Recognition, 2001, 34(11): 2067-2070

[9] Yang J, Yang J-Y. Why can LDA be performed in PCA transformed space? [J]. Pattern Recognition, 2003(36): 563-566

[10] Dai D Q, Yuen P C. Regularized discriminant analysis and its application on face recognition[J]. Pattern Recognition, 2003, 36 (1): 845-847

[11] Li H, Jiang T, Zhang K. Efficient and robust feature extraction by maximum margin criterion [C] // Proc. Adv. Neural Inf. Process. Syst.. 2003: 97-104

[12] Li H, Jiang T, Zhang K. Efficient and robust feature extraction by maximum margin criterion [J]. IEEE Transactions on Neural Networks, 2006, 17(1): 1157-1165

[13] Olivetti & Oracle Research Laboratory. The Olivetti & Oracle Research Laboratory Face Database of Faces [OL]. http://www.cam-orl.co.uk/facedatabase.html, 1994

[14] Yale University Face Database[OL]. http://cvc.yale.edu/projects/yalefaces/yalefaces.html, 2002

(上接第 47 页)

[10] Rizvandi N B, Taheri J, Zomaya A Y. On using pattern matching algorithms in MapReduce applications[C] // The 9th IEEE International Symposium on Parallel and Distributed Processing with Applications(ISPA). 2011: 75-80

[11] Sun Y J, Todorvic S, Goodison S. Local Learning Based Feature Selection for High Dimensional Data Analysis[J]. On Pattern Analysis and Machine Intelligence, 2010, 32(9): 1610-1626

[12] Li Y, Lu B L. Feature selection based on loss margin of nearest neighbor classification[J]. Pattern Recognition, 2009, 42: 1914-1921

[13] Han Y, Yu L. A variance reduction framework for stable feature selection[C] // Proc. Int'l Conf. on Data Mining. 2010: 206-215

[14] Zinkevich M, Weimer M, Smola A, et al. Parallelized stochastic gradient descent[J]. Advances in Neural Information Processing Systems, 2010: 1-37

[15] Ranger C, Raghuraman R, Penmetsa A, et al. Evaluating MapReduce for Multicore and Multiprocessor Systems[C] // Proceedings of 2007 IEEE 13th International Symposium on High Performance Computer Architecture. 2007: 13-24

[16] Lichman M. UCI Machine Learning Repository[OL]. http://archive.ics.uci.edu/ml/

[17] http://www.nipsfsc.ecs.soton.ac.uk/datasets/