基于 HBase 的多分类逻辑回归算法研究*

刘黎志[a,b]，邓介一[a,b]，吴云韬[a,b]
(武汉工程大学 a. 智能机器人湖北省重点实验室; b. 计算机科学与工程学院，武汉 430205)

摘 要: 为解决在大数据环境下，用于训练多分类逻辑回归模型的数据集可能会超过执行计算的客户端内存的问题，提出了块批量梯度下降算法，用于计算回归模型的系数。将训练数据集存入 HBase 后，通过设置表扫描对象的起始行键参数，可取出大小合适的含训练样本及结果值的数据块；同时为避免客户端到服务端频繁的 RPC 调用，取出的数据块可进行多次迭代计算，以加快系数的收敛。当取出的数据块达到指定的迭代次数后，再按行键次序取出下一个数据块。如此循环，直到系数收敛或达到指定的循环控制阈值。多分类的逻辑回归问题可转换为二分类来解决，因此需要为每一个分类在训练数据表中设定结果值列，结合训练样本列簇，按块批量梯度下降算法得到每个分类的回归系数。实验结果表明得到的回归系数能准确地对测试样本进行分类。

关键词: 块批量梯度下降; 多分类; 逻辑回归; 大数据; HBase
中图分类号: TP301.6 **文献标志码:** A **文章编号:** 1001-3695(2018)10-3007-04
doi:10.3969/j.issn.1001-3695.2018.10.029

Research on multi classification logistic regression based on HBase

Liu Lizhi[a,b], Deng Jieyi[a,b], Wu Yuntao[a,b]
(*a. Hubei Province Key Laboratory of Intelligent Robot, b. School of Computer Science & Engineering, Wuhan Institute of Technology, Wuhan* 430205, *China*)

Abstract: In big data environment, the training dataset for logistic regression model may exceed the memory size of the client machine that executes computing, so this paper proposed a chunk BGD algorithm to compute the coefficients of regression model. After putting the training dataset in HBase, a data chunk with appropriate size including training sample data and classification result value could be obtained by setting the StartRow and StopRow parameters of the scan object. In case of avoiding frequent RPC calls from client to server, the chunk could be iterated multi-times to accelerate the convergence of coefficients. When the obtained chunk reaches the specified iteration times, then next chunk was taken out according to the order of row keys. These kinds of circles would be repeated until the convergence of coefficients or reaching the loop control threshold. Multi classification logistic regression problem could be resolved by converting to two classification model, so the result value column qualifier for each classification must be added into training data table in HBase, combining with the training sample column family, each classification regression coefficients could be obtained by chunk BGD algorithm. The result of experiment proves that the testing samples can be classified accurately by the regression coefficients.

Key words: chunk BGD; multi classification; logistic regression; big data; HBase

0 引言

随着互联网 + 技术与各个行业的快速融合，数据的产生已经逐步从被动方式转换为主动或自动方式，导致数据量的爆炸式增长，大数据的概念受到越来越多的关注。由于大数具有的 4V 特征：大容量、多样性、高速度、价值性，传统的数据挖掘和机器学习方法在大数据环境下需要改进及扩展，研究大数据环境下的数据挖掘及机器学习算法成为学术界和产业界的热点[1~6]。分类算法是数据挖掘、机器学习中一个重要的研究领域，通过对已知类别训练数据集的分析，从中发现分类规则，以此预测新数据的类别[7,8]。逻辑回归是一种比较有效和实用的分类算法，利用分布式存储结构及计算框架对传统的逻辑回归算法进行改进，可有效地加快回归系数的收敛时间及提高分类的准确率[9~12]。在大数据环境下，用于逻辑回归模型训练的数据集可能会非常巨大，不可能一次性全部加载到内存中进行回归系数的计算。本文就如何利用 HBase 数据库存储训练数据集及使用块批量梯度下降算法得到多分类的逻辑回归模型展开了研究。

1 研究背景

逻辑回归是一种分类方法，主要用于二分类问题，即输出只有两种 1 或者 0，分别代表两个类别。逻辑回归使用非线性的 sigmoid 函数进行分类预测，函数形式为

$$g(z)=\frac{1}{1+e^{-z}} \tag{1}$$

设影响预测结果的特征向量为 $X(x_1,x_2,\cdots,x_n)$，回归系数为 $\theta(\theta_0,\theta_1,\theta_2,\cdots,\theta_n)$，则

$$\theta_0+\theta_1x_1+\theta_2x_2+,\cdots,+\theta_nx_n=\sum_{i=1}^{n}\theta_ix_i=\theta^{\mathrm{T}}x \tag{2}$$

构造预测函数为

$$h_\theta(x)=g(\theta^{\mathrm{T}}x)=\frac{1}{1+e^{-\theta^{\mathrm{T}}x}} \tag{3}$$

若 θ 已知，使用 $h_\theta(x)$ 计算某个特征向量 X，若结果大于

收稿日期: 2017-05-15; 修回日期: 2017-06-26 基金项目: 湖北省自然科学基金资助项目(2014CFB791); 湖北省高等学校优秀中青年科技创新团队计划资助项目(T201206)

作者简介: 刘黎志(1973-)，男，副教授，硕士，主要研究方向为基于移动互联网的计算机应用、云计算、大数据、数据仓库及数据挖掘(llz73@163.com); 邓介一(1992-)，男，硕士研究生，主要研究方向为大数据、数据仓库及数据挖掘; 吴云韬(1973-)，男，教授，博士(后)，主要研究方向为阵列信号处理中的参数估计及波束形成技术、统计信号处理中的信号检测和参数估计、无线传感器网络中的定位技术等.

0.5,则认为其属于分类 1,否则为分类 0。逻辑回归属于监督学习,模型系数的求解需要首先将已知数据集划分为训练数据集及测试数据集,根据训练数据集按梯度下降算法求解 θ; 然后使用测试数据集对模型进行评估。按梯度下降算法求解 θ 的过程描述如下:

设训练数据集中的训练样本数量为 m,则

$$P(y|x;\theta) = (h_\theta(x))^y(1-h_\theta(x))^{1-y} \tag{4}$$

取极大似然函数为

$$L(\theta) = \prod_{i=1}^{m} P(y_i|x_i;\theta) = \prod_{i=1}^{m}(h_\theta(x_i))^{y_i}(1-h_\theta(x_i))^{1-y_i} \tag{5}$$

似然函数取对数为

$$l(\theta) = \log L(\theta) = \sum_{i=1}^{m}(y_i \log h_\theta(x_i) + (1-y_i)\log(1-h_\theta(x_i))) \tag{6}$$

极大似然估计求的是使得 $l(\theta)$ 值最大的 θ 值。设损失函数 $J(\theta)$ 为

$$J(\theta) = -\frac{1}{m}l(\theta) \tag{7}$$

因为乘了系数 $-1/m$,所以取 $J(\theta)$ 为最小值时的 θ 为所需要的最佳系数。批量梯度下降法求最小值的 θ 更新过程为

$$\theta_j := \theta_j - \alpha\frac{\delta}{\delta_{\theta_j}}J(\theta) \tag{8}$$

其中: α 为学习率。对 $J(\theta)$ 求偏导数的求解过程在此省略,直接给出结论为

$$\frac{\delta}{\delta_{\theta_j}}J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left(\begin{array}{l} y_i\dfrac{1}{h_\theta(x_i)}\dfrac{\delta}{\delta_{\theta_j}}h_\theta(x_i) - (1-y_i)\dfrac{1}{1-h_\theta(x_i)}\dfrac{\delta}{\delta_{\theta_j}}h_\theta(x_i)) = \\ -\dfrac{1}{m}\sum_{i=1}^{m}(y_i - h_\theta(x_i) \end{array}\right)x_i^j \tag{9}$$

因此批量梯度下降 θ 的更新过程可以写成

$$\theta_j := \theta_j + \alpha\frac{1}{m}\sum_{i=1}^{m}(y_i - h_\theta(x_i))x_i^j \tag{10}$$

其中: $y_i - h_\theta(x_i)$ 表示每个样本的结果值与预测值的偏差量。

多分类的逻辑回归可以转换为二分类的逻辑回归来解决,具体做法是: 将需要计算逻辑回归系数 θ^i 的分类 i 的结果值设置为 1,其他分类的结果值设置为 0,利用训练数据计算 θ^i,从而可以得到每个分类的回归系数 θ^i。若有 k 个分类,就有 n 组回归系数(θ^1, θ^2,…, θ^k) 。在进行预测时,将 θ^i 代入 $h_\theta(x)$,逐个计算 x 的每个预测值,取预测值最大的分类为预测的结果。

2 基于 HBase 的多分类逻辑回归

HBase 是建立在 Hadoop 之上,具有高可靠性、高性能、列存储、可伸缩、实时读写特点的分布式数据库系统。它通过行键(row key) 和行键的范围来检索数据,能够为海量的数据提供高性能的数据维护及查询服务[13~15]。从逻辑上讲,HBase 将数据按照表、行和列进行存储。列簇(column family) 可以自由扩展,列簇存储在 HFile 文件中。HFile 由 HDFS 进行自动的合并或拆分管理,从而使得列簇中的数据可以分布式地存储在 Hadoop 集群的各个数据节点中,极大地提高了数据并行处理的能力。

为解决在大数据环境下训练数据集大的问题,可首先将训练数据集存放到 HBase 数据表中,以便对训练数据进行更好的管理和维护; 然后使用基于 HBase 的块批量梯度下降算法求出各分类的模型系数 θ; 最后导入测试数据集,对各分类模型系数进行评价。

2.1 存取训练数据集

对于多分类逻辑回归问题,可以在存储训练数据集的 HBase 表 trainData 中建立两个列簇,一个用于存储训练样本特征值,一个用于存储结果值。HBase 中的训练数据表结构如图 1 所示。

trainData(训练数据表)							
行键	*cfx*-训练样本特征值列簇				*cfy*-结果值列簇		
	d_0	d_1	d_2…	d_n	r_1	r_2…	r_k
1000000001	1	x_{11}	x_{12}…	x_{1n}	1	0…	0
…	…	…	…	…	…	…	…
9999999999	1	x_{m1}	x_{m2}…	x_{mn}	0	1…	0

图 1 训练数据表结构

HBase 默认按行键索引,用于数据的查询。行键按字符顺序自动排序,故一般将行键设置为等宽的数字字符序列。在写入 trainData 时,结果值列簇中的各列(*cfy*: r_1, *cfy*: r_2,…, *cfy*: r_k) 的值设置为(0,1,…,0) 的形式,训练数据行对应的分类设置为 1,其他的分类设置为 0。Scan 查询用于扫描数据表,获得多行数据。指定 scan 查询的列簇 *cfx* 及特定的结果值列 *cfy*: r_k 后,还可以设置查询的 startRow 及 stopRow 参数,用于设定查询的起始行键。查询结果为在设定的起始行键范围内的含有 *cfx* 及 *cfy*: r_k 所有数据行,称为一个数据块(chunk) 。如何将测试数据集存入到训练数据表 trainData 的过程在此省略。定义返回数据块的类为:

```
class trainData {
  public double[] x{get; set;}; //存储训练样本特征值
  public double[] y{get; set;}; //存储结果值
}
```

在存储训练数据集的 HBase 表中,取含训练样本及结果值数据块的过程如算法 1 所示。

算法 1 GetChunkData(tbName, *cfx*, *cfy*, r_k, startKey, endKey, trainData) {

输入: tbName 为存储训练数据的 HBase 表名; *cfx* 为训练样本列簇; *cfy* 为结果值列簇; r_k 为指定分类对应的结果值列; startKey 为数据块的开始行键; endKey 为数据块的结束行键; trainData 为返回数据块的类对象,含 x 及 y。

输出: 含指定 *cfx* 及 *cfy*: r_k 数据块的 trainData 类实例。

```
1 Table tb = con.getTable(TableName.valueOf(tbName));
        //获得表对象
2 Scan s = new Scan(); //得到 scan 对象实例
3 s.addFamily(Bytes.toBytes(cfx)); //添加训练样本列簇到 scan
4 s.addColumn(Bytes.toBytes(cfy),Bytes.toBytes(r_k));
        //添加结果值列到 scan
5 s.setStartRow(startKey.getBytes()); s.setStopRow(endKey.getBytes()); //设置 scan 起始行健
6 ResultScanner scanResult = tb.getScanner(s);
        //查询指定的数据块
7 初始化 x 一维数组及 y 数组的大小为数据块中包含的训练样本的个数;
8 for(Result r r = scanResult.next(); r r! = null; r r = scanResult.next()) { //取出查询结果
9     Cell[] cells = r r.listCells();
10    初始化 x 二维数组的大小为 cells 的长度 ql;
11    for(int j = 0; j < ql - 1; j ++) { trainData.x[][] = cells[][] 的值; } //循环取训练样本特征值
12    trainData.y[] = cells[ql - 1] 的值;
          //取训练样本对应的结果值
13    i ++; }}
```

2.2 块批量梯度下降

在大数据环境下,训练样本数据量一般非常大,批量梯度下降算法(BGD) 在更新系数 θ 时需要求全局的偏差量,但训练样本数据量可能会超出执行计算的客户端内存大小,因此 BGD 在大数据的环境下一般不可取。随机梯度下降算法(SGD) 可以每读取一行样本数据,就更新一次系数 θ,但容易受到数据波动的影响,噪声较多,不能保证系数 θ 的每次迭代都朝着整体最优的方向进行,且每读取一行数据,都需要客户端到服务端的一次 RPC 调用,过多地消耗了系统资源。合适的方法是根据客户端内存大小,每次取出一个训练样本数据块,按 BGD 的算法进行系数 θ 的更新; 同时为了避免客户端到

服务端的频繁 RPC 调用，取出的数据块可以进行指定次数的多次迭代，用于更新系数 θ；到达指定的迭代次数后，再取出样本数据中的下一个数据块更新系数，直到系数 θ 收敛为止。若整个训练样本按数据块取完后系数仍然不收敛，则回到训练样本开始的位置，再次取出第一个数据块进行迭代。如此循环，直到 θ 收敛。为避免 θ 在收敛处左右波动，始终不能满足收敛的条件，可以设置一个限制数据块迭代次数的阈值。当迭代整个样本数据块的迭代次数超过阈值后，不论 θ 是否收敛，都结束系数 θ 的更新过程。按块批量梯度下降算法完整地更新一次 θ_j 的过程描述如下：

设训练样本块为 $x=\begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}=\begin{bmatrix} 1 & x_{11} & \cdots & x_{1n} \\ & \vdots & & \vdots \\ 1 & x_{m1} & \cdots & x_{mn} \end{bmatrix}$，训练样本结果值块 $y=\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$，系数 $\theta=\begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$ 则

$$A=x\cdot\theta=\begin{bmatrix} \theta_0+\theta_1x_{11}+\theta_2x_{12}+\cdots+\theta_nx_{1n} \\ \vdots \\ \theta_0+\theta_1x_{m1}+\theta_2x_{m2}+\cdots+\theta_nx_{mn} \end{bmatrix} \tag{11}$$

$$E=y-h_\theta(x)=\begin{bmatrix} y_1-g(A_1) \\ \vdots \\ y_m-g(A_m) \end{bmatrix}=\begin{bmatrix} e_1 \\ \vdots \\ e_m \end{bmatrix} \tag{12}$$

$$\theta_j:=\theta_j+\alpha\frac{1}{m}\sum_{i=1}^{m}(y_i-h_\theta(x_i))x_i^j=$$

$$\theta_j+\alpha\frac{1}{m}\sum_{i=1}^{m}e_ix_i^j=\theta_j+\alpha\frac{1}{m}x_j^TE \tag{13}$$

大数据环境下的训练数据量可能非常大，所以 θ 的收敛判断不能以全局的 $J(\theta)$ 变化绝对值小于某个阈值来判断，较合适的方法是判断每次 θ 更新后的变化距离。若变化距离小于指定的阈值，则可判断系数已经收敛。具体步骤为：

a) 在块批量梯度下降算法中，将数据块每一次迭代更新后的系数 θ 保存到 θ'。

b) 数据块按设定的次数结束迭代后，计算最后一次更新的 θ 与 θ' 的距离，即

$$\text{diff}=\sum_{i=1}^{n}(\theta_i-\theta'_i)^2$$

c) 若 diff 小于指定的阈值，则判断系数已经收敛。

块批量梯度下降算法中求数据块的结果值与预测值的偏差量数组 E 的过程如算法 2 所示。

算法 2　GetXPlusθE(*x*,θ,*y*) {

输入：x 为训练样本二维数组；θ 为待求解系数；y 为结果值(0 或 1，一维数组)。

输出：每一个训练样本结果值与预测值偏差量的一维数组 E。

```
1 for( i =0; i < x 样本块的一维长度 m; i ++) {
2    for( j =0; j < x 样本块的二维长度 n; j ++) { A_i + = x [ ] [ ] *
θ [ ]; }
3   E [ ] = y [ ] - g( A_i ); }
4 return E; }
```

块批量梯度下降算法中求分类系数 θ 的过程如算法 3 所示。

算法 3　HBaseChunkBGD(*tb*Name,*cfx*,*cfy*,r_k,tdlTimes,cdlTimes,cdSize) {

输入：tbName,*cfx*,*cfy*,r_k 参数含义描述见算法 1。tdlTimes 为数据块的迭代次数限制阈值；cdlTimes 为每一个数据块的迭代次数；cdSize 为数据块的大小。

输出：θ 为求解出的系数数组。

```
1 初始化系数数组 θ、θ'; 学习率 α; startKey = 1; endKey = startKey +
cdSize; //定义第一个数据块的大小
2 for( int m =0; m < tdlTimes; m ++)
3 {   GetChunkData( tbName, cfx, cfy, r_k, startKey, endKey, trainDa-
ta); //调用算法 1 指定的取数据块
4   for( int p =0; p < cdlTimes; p ++)
5   { E = GetXPlusθE( trainData. x,θ,trainData. y);
    //调用算法 2 求偏差量数组 E
6     for( j =0; j < x 样本块的二维长度 n; j ++) {
      θ' [ ] = θ [ ]; //保存每个系数到 θ'
7   for( i =0; i < x 样本块的一维长度 m; i ++) { sum + = trainDa-
ta. x [ ] [ ] * E [ ]; }
8   θ [ ] = θ [ ] + α* sum; } }
    //梯度下降，更新系数，每一个数据块迭代 cdlTimes 次
9   计算 θ 与 θ' 的距离 diff,if( θ 收敛) { 终止循环; }
10  else{ 设定 startKey、endKey，取下一个数据块，若所有训练数据
按块全部取完后 θ 仍然不收敛，则重置 startKey、endKey，取训练数据的
第一个数据块; } }
11 return θ; }
    //系数 θ 收敛或者整个数据块的迭代次数超过阈值 tdlTimes
```

2.3　多分类逻辑回归模型评价

多分类逻辑回归属于监督类机器学习，需要首先将学习样本数据集划分为训练数据集和测试数据集两类，使用训练数据集得到逻辑回归模型，使用测试数据集对模型的准确性进行评价。测试数据集的大小一般为整个样本数据的 5% ~ 10%。由于需要对多分类逻辑回归模型进行评价，测试数据集中应尽量均匀包含每个分类的样本数据，以保证测试的准确性。对多分类逻辑回归模型进行评价的过程如算法 4 所示。

算法 4　MLGMain(trainDSFilePath,testDSFilePath) {

输入：训练数据集，测试数据集。

输出：测试数据集中每个样本的分类结果。

```
1 将训练数据集存储到 HBase 数据表 trainData 中;
2 设定 cfx,cfy,tdlTimes,cdlTimes,cdSize 的初始值;
3 for( i =0; i < k; i ++) {
  //调用算法 3 得到每个分类的回归系数数组 θ^i,k 为分类的个数
4   θ^i = HBaseChunkBGD( tbName,cfx,cfy,r_i,tdlTimes,cdlTimes,cd-
Size); }
5 读取测试数据集到 testDS;
6 foreach( x in testDS) {
  for( i =0; i < k; i ++) { z_i = g( θ^iT x); }
7 取 max( z_i) 为测试样本 x 的分类结果; } }
```

将测试数据集中每个样本数据的特征值数组 x 和训练得到的每个分类的系数数组 θ^i 代入到 $g(\theta^{iT}x)$ 函数计算结果值，取使结果值最大的系数数组所属的分类作为测试结果分类，然后与该样本数据真实的分类作比较，从而评价分类结果是否正确。

3　实验

实验用服务器为 Dell PowerEdge R720，其配置为 2 个物理 CPU(Intel Xeon E5-2620 V2 2.10 GHz，每个 CPU 含 6 个内核，共 12 个内核)，32 GB 内存，8 TB 硬盘，4 个物理网卡。服务器安装 VMWare esxi 6.0.0 操作系统，虚拟化整个服务器环境。客户端使用 VMWare VSphere client 6.0.0 将服务器划分为 4 个虚拟机，每个虚拟机的配置为 3 内核 CPU，8 GB 内存，2 TB 硬盘，1 个物理网卡。每个虚拟机安装 ubuntu-16.04.1-server-amd64 操作系统，Hadoop 2.7.3 分布式计算平台，组成含 1 个主节点、4 个数据节点(主节点也是数据节点)的集群。集群中安装的 HBase 版本为 1.2.3。客户端使用 Eclipse 4.6.2，Java SE 1.7 为开发环境。

以 Iris flower data set 作为测试数据。测试样本以 Sepal length、Sepal width、Petal length、Petal width 这四个特征值确定 Iris flower 为 setosa、versicolor、virginica 分类中的一类。实验中将每个分类 90% 的数据样本作为训练数据集，10% 的数据样本作为测试数据集。分别设置 setosa、versicolor、virginica 为 r_1、r_2、r_3，设置 Sepal length、Sepal width、Petal length、Petal width 为 d_1、d_2、d_3、d_4，按图 1 所示的存储模式，将训练数据集存入到 HBase 数据库的 trainData 表中。设置 tdlTimes 最大值为1 000，cdlTimes 为 500，cdSize 为 45，初始回归系数 θ 为{1,1,1,1,1}，系数收敛距离阈值为 0.000 000 1。调用算法 3 得到的各分类

回归系数如表 1 所示。使用得到的系数对测试数据集进行测试的结果如表 2 所示。

表 1 各分类系数

	学习率 α	tdlTimes	θ_0	θ_1	θ_2	θ_3	θ_4
θ^1-setosa	0.001	72	1.340	0.512	3.292	-4.709	-1.477
θ^2-versicolor	0.000 2	159	3.536	0.590	-2.834	0.938	-2.248
θ^3-virginica	0.000 1	333	-3.274	-3.658	-3.542	5.373	6.302

表 2 测试结果

Species		Sepal length	Sepal width	Petal length	Petal width	θ^1	θ^2	θ^3
setosa	1	4.8	3.0	1.4	0.3	0.999	0.183	0.000
setosa	1	5.1	3.8	1.6	0.2	1.000	0.040	0.000
setosa	1	4.6	3.2	1.4	0.2	0.999	0.124	0.000
setosa	1	5.3	3.7	1.5	0.2	1.000	0.054	0.000
setosa	1	5.0	3.3	1.4	0.2	1.000	0.119	0.000
versicolor	1	5.7	3.0	4.2	1.2	0.001	0.411	0.010
versicolor	1	5.7	2.9	4.2	1.3	0.000	0.425	0.026
versicolor	1	6.2	2.9	4.3	1.3	0.000	0.522	0.007
versicolor	1	5.1	2.5	3.0	1.1	0.027	0.451	0.000
versicolor	1	5.7	2.8	4.1	1.3	0.000	0.472	0.021
virginica	1	6.7	3.0	5.2	2.3	0.000	0.214	0.982
virginica	1	6.3	2.5	5.0	1.9	0.000	0.643	0.975
virginica	1	6.5	3.0	5.2	2.0	0.000	0.321	0.946
virginica	1	6.2	3.4	5.4	2.3	0.000	0.073	0.996
virginica	1	5.9	3.0	5.1	1.8	0.000	0.322	0.963

从实验结果分析，通过算法 3 将得到的各组回归系数（θ^1，θ^2，θ^3）及测试样本数据代入预测函数，得到的计算结果对测试数据进行了准确的分类。实验准确率高的原因是用于训练模型和测试模型的数据量都不够大，且样本特征值均经过了很好的泛化处理。在实际应用中，模型的预测准确率不可能达到 100%。

4 结束语

在实验的过程中发现，学习率的设置对系数的收敛非常重要，合适大小的学习率 α 能够使得系数快速收敛，但需要经过多次的测试才能找到合适的学习率。通过块批量梯度下降算法和传统批量梯度下降算法得到的回归系数虽然都能够对测试数据进行准确分类，但系数值之间还是存在较大的差距。因此在块批量梯度下降算法中，如何快速设定学习率及正确判断系数的收敛还需要做进一步的工作。将训练数据集存入 HBase 数据库，就可以借助于 MapReduce 或 Spark 平台对大规模的数据进行分布式计算。因此，改进多分类的逻辑回归算法可并行化，提高模型系数的收敛速度及预测的准确性，也是今后重点的研究方向。

参考文献:

[1] 何清，李宁，罗文娟，等. 大数据下的机器学习算法综述 [J]. 模式识别与人工智能，2014，27(4)：327-336.

[2] 梁吉业，大数据挖掘面临的挑战与思考 [J]. 计算机科学，2016，43(7)：1-2.

[3] 米允龙，米春桥，刘文奇. 海量数据挖掘过程相关技术研究进展 [J]. 计算机科学与探索，2015，9(6)：641-659.

[4] 黄宜华. 大数据机器学习系统研究进展 [J]. 大数据，2015，1(1)：28-47.

[5] Landset S，Khoshgoftaar T M，Richter A N，*et al*. A survey of open source tools for machine learning with big data in the Hadoop ecosystem [J]. Journal of Big Data，2015，2(1)：24.

[6] 吕婉琪，钟诚，唐印浒，等. Hadoop 分布式架构下大数据集的并行挖掘 [J]. 计算机技术与发展，2014，24(1)：22-25.

[7] Suthaharan S. Big data classification：problems and challenges in network intrusion prediction with machine learning [J]. ACM SIGMETRICS Performance Evaluation Review，2014，41(4)：70-73.

[8] 毛国君，胡殿军，谢松燕. 基于分布式数据流的大数据分类模型和算法 [J]. 计算机学报，2017，40(1)：161-175.

[9] Lin C Y，Tsai C H，Lee C P，*et al*. Large-scale logistic regression and linear support vector machines using spark [C]//Proc of IEEE International Conference on Big Data. Piscataway，NJ：IEEE Press，2014：519-528.

[10] 帅仁俊，沈阳，陈平，等. 基于 logistic 回归模型的 Hadoop 本地任务调度优化算法 [J]. 计算机应用研究，2017，34(3)：727-729，755.

[11] 苏汉宸，李红燕，苗高杉，等. PTLR：云计算平台上处理大规模移动数据的置信域逻辑回归算法 [J]. 计算机研究与发展，2010，47(z1)：414-419.

[12] 陈振宏，兰艳艳，郭嘉丰，等. 基于差异合并的分布式随机梯度下降算法 [J]. 计算机学报，2015，38(10)：2054-2063.

[13] 王珊，王会举，覃雄派，等. 架构大数据：挑战、现状与展望 [J]. 计算机学报，2011，34(10)：1741-1751.

[14] 孟小峰，慈祥. 大数据管理：概念、技术与挑战 [J]. 计算机研究与发展，2013，50(1)：146-169.

[15] 陈吉荣，乐嘉锦. 基于 Hadoop 生态系统的大数据解决方案综述 [J]. 计算机工程与科学，2013，35(10)：25-35.

(上接第 2946 页)

4 结束语

本文在邻域多粒度粗糙集的基础上，综合考虑属性集差异性和邻域半径多重性，构建了基于多阈值的变精度邻域多粒度粗糙集决策分析方法，以测试对象为标准，对不同特征的属性集数据采用不同的邻域半径，改进了传统的邻域多粒度粗糙集模型，并引入变精度粗糙集理论，使之更好地适用于含有噪声数据、不确定信息、多源数据及分布式数据的混合信息系统。理论分析与实践结果表明，所提出的基于多阈值的变精度邻域多粒度粗糙分析模型是一种处理多属性数据决策问题的有效方法。

参考文献:

[1] An Aijun，Shan Ning，Chan C，*et al*. Discovering rules for water demand prediction：an enhanced rough-set approach [J]. Engineering Applications of Artificial Intelligence，1996，9(6)：645-653.

[2] Qian Yuhua，Liang Jiye，Yao Yiyu，*et al*. MGRS：a multi-granulation rough set [J]. Information Science，2010，180(6)：949-970.

[3] Lin T Y. Granular computing on binary relations I：data mining and neighborhood systems [M]//Rough Sets in Knowledge Discovery. 1998：107-121.

[4] 胡清华，于达仁，谢宗霞，等. 基于邻域粒化和粗糙逼近的数值属性约简 [J]. 软件学报，2008，19(3)：640-649.

[5] Hu Qinghua，Zhang Lei，Zhang D，*et al*. Measuring relevance between discrete and continuous features based on neighborhood mutual information [J]. Expert Systems with Applications，2011，38(9)：10737-10750.

[6] 谢娟英，李楠，乔子芮. 基于邻域粗糙集的不完整决策系统特征选择算法 [J]. 南京大学学报，2011，47(4)：383-390.

[7] 徐久成，张灵均，孙林，等. 广义邻域关系下不完备混合决策系统的约简 [J]. 计算机科学，2013，40(4)：244-248.

[8] Wang Changzhong，Shao Mingwen，He Qiang，*et al*. Feature subset selection based on fuzzy neighborhood rough sets [J]. Knowledge-Based Systems，2016，111(1)：173-179.

[9] 孟军，李锐，郝涵. 基于相交邻域粗糙集的基因微阵列数据分类 [J]. 计算机科学，2015，42(6)：37-40.

[10] 续欣莹，刘海涛，谢珺，等. 信息观下基于不一致邻域矩阵的属性约简 [J]. 控制与决策，2016，31(1)：130-136.

[11] 徐怡，杨宏健，纪霞. 基于双重粒化准则的邻域多粒度粗糙集模型 [J]. 控制与决策，2015，30(8)：1469-1478.

[12] Li Weiwei，Huang Zhiqiu，Jia Xiuyi，*et al*. Neighborhood based decision-theoretic rough set models [J]. International Journal of Approximate Reasoning，2016，69(2)：1-17.