文章编号: 1000-5862(2012)04-0388-07

# 基于 MapReduce 的粒子群投影寻踪模型的设计与实现

黄奕平，万剑怡，万中英，王明文*

(江西师范大学计算机信息工程学院，江西 南昌 330022)

摘要：利用 MapReduce 模式设计并实现了粒子群投影寻踪算法的并行化，以提高算法的效率. 在分类阶段使用了基于 MapReduce 的 KNN 分类算法并行，实验结果表明：基于 MapReduce 实现的粒子群投影寻踪模型能够有效地寻找到较好的投影方向，确保分类效果；并且与其串行算法相比，在实现效率上有较大的提高.

关键词：投影寻踪；粒子群优化; MapReduce；文本分类；并行化
中图分类号: TP 391.1　　文献标志码: A

## 0　引言

在数据挖掘中，常遇到高维数据和大规模数据的处理. 面对高维数据时，会遇到算法计算效率和“维数灾难”导致的常用方法失效等问题. 于是许多数据降维方法相继被提出，将高维数据降到低维后再进行分析，以提高学习性能；另一方面可以通过实现算法并行化来提高算法效率. 投影寻踪(projection pursuit, PP)[1-2]是一种通过寻找出能反映原高维数据结构或特征的投影方向，将高维数据投影到低维子空间上，以达到在低维空间处理和分析高维数据的降维方法.

PP 模型的建立关键是它的投影指标的优化问题，已有多种优化算法被采用，如 GA、回归函数等. 粒子群优化(particle swarm optimization, PSO)算法[3-5]是由 J. Kennedy 和 R. C. Eberhart 提出的全局优化进化算法，在优化问题中得到广泛应用，但还未有人对 PSO 算法与投影寻踪结合进行 MapReduce 公司的并行化研究. MapReduce 计算模式[6-7]是 Google 实验室提出的用于实现大规模数据的分布式并行处理编程模式，它使得程序的并行化更简易化.

在文本分类中，本文通过投影寻踪模型将文本投影到低维空间再进行分类；在模型投影指标(投影方向)的优化过程中使用了 PSO 算法. 由于算法符合 MapReduce 编程模式的性质，为提高算法的执行效率，本文将该模型并行化，设计并实现了基于 MapReduce 的粒子群投影寻踪模型. 投影后的分类阶段本文采用基于 MapReduce 的 KNN 算法对文本进行分类. 在 Hadoop 集群环境中采用复旦大学的数据集进行实验，实验结果表明：基于 MapReduce 实现的粒子群投影寻踪模型在保证分类效果的同时提高了算法的效率.

## 1　投影寻踪和粒子群算法

### 1.1　投影寻踪

投影寻踪是一种有效处理和分析维数据的方法，其通过某种组合(投影寻踪指标)将原始高维数据投影到一个较低维(1~3 维)的子空间上，通过优化(极大化或极小化)这个投影寻踪指标函数来寻踪出极能反应原高维数据的结构或特征的投影，实现在低维空间中对高维数据进行分析和处理的目的. 投影寻踪模型的构建过程中投影指标的构建和优化是极其重要的部分.

在分类应用中，本文采用文献[2]的方法，以每类投影数据的均值作为类中心的度量、标准差为类内距离的度量来共同构造投影指标. 将原数据投影到 1 维空间，设训练过程中有 $n$ 个对象，对象 $i$ 为 $Y_i \in R^m$ ($i$=1,2,⋯, $n$; $m$ 为特征数; $R^m$ 为 $m$ 维空间), $Y_i = (y_{i1}, y_{i2}, \cdots, y_{im})$ , $a$ 为投影方 $\in R^m$ ，则对象 $Y_i$ 的

收稿日期：2012-03-06
基金项目：国家自然科学基金(61163006, 60963014)和江西省自然科学基金(20114BAB201037)资助项目.
作者简介：王明文(1965-)，男，江西南康人，教授，博士，主要从事信息检察和并行计算的研究.

投影值为

$$z_i = \sum_{j=1}^{m} a_j y_{ij} = a^T \mathrm{Y}_i \quad (i = 1,2,\cdots,n,\ j = 1,2,\cdots,m), \quad (1)$$

构造的投影指标为

$$Q(z) = B(z)/D(z), \quad (2)$$

其中 $z$ 为将 $m$ 维数据投影到 1 维的投影值, $B(z)$为 2 类中心距离，$D(z)$为类内散布的平均值，$B(z)$ 和 $D(z)$ 的定义如下:

$$B(z) = |E(z^{(1)}) - E(z^{(2)})|, \quad (3)$$

$$D(z) = \left[\frac{\sum_{j=1}^{n_1}(z_j^{(1)} - E(z^{(1)}))^2 + \sum_{j=1}^{n_2}(z_j^{(2)} - E(z^{(2)}))^2}{n_1 + n_2}\right]^{1/2}, \quad (4)$$

其中 $E(z^{(1)})$、$E(z^{(2)})$ 分别表示第 1 类和第 2 类投影值的均值，$z_j^{(1)}$、$z_j^{(2)}$ 分别表示第 1 类和第 2 类对应对象的投影值, $n_1$、$n_2$ 分别表示 2 类的对象数，且 $n_1$+$n_2$=$n$. 投影指标公式(2)计算得到的值越大说明对类别的区分性越强.

### 1.2　粒子群优化算法

粒子群优化算法是一种基于群体的演化算法. PSO 算法中首先生成一群粒子，每个粒子都有自己的初始位置、飞行速度，还有衡量粒子好坏的目标函数. 每个粒子根据自身的局部最优位置和全局最优位置来调整自己的速度和位置，进行动态更新. 设目标搜索空间为 $n$ 维，第 $i$ 个粒子位置为 $X_i = (x_{i1}, x_{i2}, \cdots, x_{in})$，它经历过的最好位置 $P_i = (p_{i1}, p_{i2}, \cdots, p_{in})$，也称为局部最优位置 $P_{\mathrm{best}_i}$. 所有粒子经历过的最好位置为 $P_g$，称为全局最优位置 $g_{\mathrm{best}_i}$. 粒子 $i$ 的速度用 $V_i = (v_{i1}, v_{i2}, \cdots, v_{in})$ 表示. 对每一代，其第 $d$ 维($1 \leqslant d \leqslant n$) 根据如下方程变化:

$$v_{id}(t+1) = wv_{id}(t) + c_1 rand_1(p_{id} - x_{id}) + c_2 rand_2(p_{gd} - x_{id}), \quad (5)$$

$$x_{id} = x_{id} + v_{id}, \quad (6)$$

其中 $w$ 为惯性权重，通常是从0.9线性减小到0.4; $c_1$ 和 $c_2$ 为加速常数，通常取值为2.05; $rand_1$ 和 $rand_2$ 为2个在[0 , 1 ]范围内变化的随机函数.

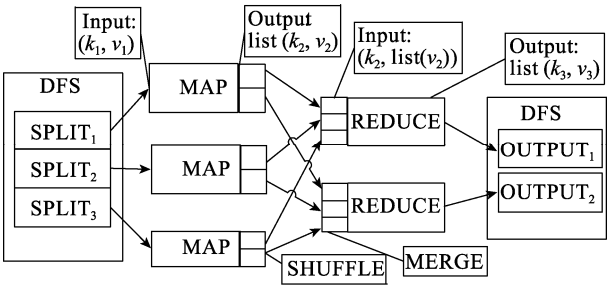
## 2　MapReduce 分布式编程模式

MapReduce 分布式编程模式是由 Google 实验室首先提出的，主要用于大规模数据集的并行计算. 它是鉴于函数式的编程模式，把海量数据集的操作抽象为 Map 和 Reduce 的 2 个集合操作，并对底层分布式过程进行了封装，大大简化了程序和并行化的实现. Map(映射)过程和 Reduce(规约)过程是 MapReduce 的 2 个关键过程. 在 MapReduce 计算模式中需要用户提供 Map 函数和 Reduce 函数以实现映射和规约过程，这 2 个函数对一组输入的键值对(*key*/*value*)进行计算，得出另一组键值对:

Map: ($k_1$, $v_1$)—>list($k$, $v$),

Reduce: ($k_2$, list($v_2$))—>list($k_3$, $v_3$).

Map 函数接收一组输入键值对($k_1$, $v_1$)经过处理产生一组中间键值对($k_2$, $v_2$)，然后 MapReduce 函数库将所有相同的 $k_2$ 键值对应的 $v_2$ 产生值的集合 list($v_2$)发送给 Reduce 函数，进一步处理、归并中间键的集合，最后形成键值对集合 list($k_3$, $v_3$). 图 1 是数据流在 MapReduce 计算过程中的传输过程示意图，首先将任务分割后进入 Map 阶段，然后将 Map 阶段的中间输出传递给 Reduce 函数, Reduce 函数经过聚合输出相应的键值对.




注: DFS 为环境中的分布式文件系统.

**图 1　MapReduce 计算过程中的数据流图**

## 3　基于 MapReduce 的粒子群投影寻踪模型的设计与实现

由于基本的 PSO 算法有着算法简单、易实现、功能强大且参数少等优点，本文利用粒子群算法来寻找最佳投影方向，采用 MapReduce 编程模式，设计和实现了基于 MapReduce 的粒子群投影寻踪模型. 另外，为了降低或防止粒子群算法容易陷入局部最优导致的收敛精度低和不易收敛的情况，在粒子信息改变时对一定概率的粒子进行突变，使粒子更具活力，该模型记为 MR_GPSO_PP，未采用突变的模型记为 MR_PSO_PP. MR_PSO_PP 和 MR_GPSO_PP 是本文设计和实现的 2 个基于 MapReduce 的粒子投影寻踪模型.

### 3.1 基于 MapReduce 的粒子群投影寻踪模型

3.1.1 *粒子包含信息及符号* 每个粒子包括{*pb_value*, *gb_value*, $X_i$, $V_i$, $g_{\text{best}_i}$, $P_{\text{best}_i}$}, 其中 *pb_value* 为 $g_{\text{best}_i}$ 对应的局部最优值, *gb_value* 为 $P_{\text{best}_i}$ 对应的全局最优值; $X_i$, $V_i$, $g_{\text{best}_i}$, $P_{\text{best}_i}$, 本文模型用于文本分类处理, 则它们维数和预处理后文本向量的维数相同; 初始化时, 粒子的 $X_i$, $V_i$ 的每一维是[–1, 1]的随机数, 并对其进行归一化, 且每次迭代也进行归一化使其在一个单位球内搜索最优位置不超出位置; *pb_value*, *gb_value* 初始为 0; $P_{\text{best}_i} = X_i$, $g_{\text{best}_i}$ 为 0 的向量.

3.1.2 *基于 MapReduce 的粒子群投影寻踪模型的设计与实现* 投影寻踪模型建立包括 4 个主要阶段: (i) 数据预处理, (ii) 构造投影指标, (iii) 优化投影指标, (iv) 建立系统模型. 这里最重要的是投影指标的优化, 也就是投影方向的优化过程, 本文基于 MapReduce 的粒子群投影寻踪模型投影方向优化过程如下:

input: 粒子群, output: 最优粒子位置.

step1: 初始化一群粒子, 生成粒子群文档和设置公式(5)相关参数;

step2: 将粒子群文档 split 到不同的 map 函数中(*key* 每行的偏移地址, *value* 为该行代表的粒子信息);

step3: 将每个粒子在 map 函数中通过与数据集进行训练, 通过公式(5)和公式(6)更新粒子的位置和速度, 通过目标函数公式(2)计算当前适应值, 判断是否为局部最优位置, 是则更新局部最优部分信息, 或同时对符合条件的粒子进行突变, 最后更新粒子信息. (*key* 是 *N*, *value* 是更新后的粒子信息);

step4: 通过 Reduce 接受上面 step3 传输的中间信息, 在 *value* 列表中, 通过局部最优得到全局最优值信息, 更新所有粒子的全局最优信息;

step5: 判断终止条件(最大迭代次数或满足某一阈值), 满足则退出, 否则以前一次 Reduce 输出的粒子群 part 文件作为处理文档进入 step2;

step6: 得到最优粒子位置(投影方向).

基于 MapReduce 的粒子群投影寻踪模型的算法核心是要实现寻踪过程中的 Map 函数和 Reduce 函数. 上述过程中的 step2、step3 在 Map 函数完成, step4 在 Reduce 函数完成. 这 2 个关键函数的具体实现分别见 Function1 和 Function2 伪代码. Function1 中"/*"与"*/"之间代码是粒子更新过程中添加了粒子突变的方法, 判断粒子是否满足突变概率, 满足则对其位置信息进行一定比例的突变属于 MR_GPSO_PP, 不包括这部分则为模型 MR_PSO_PP. Function1 中突变部分只利用了遗传算法复制、交叉、变异中的变异, 这部分粒子的突变概率由于 GA 算法中变异概率一般取 0.1 或更小[8], 所有在实验中也是取 0.1 的变异概率进行位置突变.

**Function 1 MR_PSO_PP Map or MR_GPSO_PP Map**

```
// key 为文件每行的偏移地址,value 为该行的所有信息
define mapper(key, value):
  //读取 value 信息,实例化
  Particle particle =new Particle(value);
  #下面更新粒子信息
  //更新粒子的位置和速度
  new_position, new_velocity=Particle.pso_motion(particle);
  //归一化,使它们不超出范围
  normalization(new_position, new_velocity);
  //fit 是粒子适应度,fit 越大,说明投影方向越好,
    由投影指标 Q(z) = B(z) / D(z) 得到
  fit=Particle.evluate_function(new_position);
  //判断是否为当前该粒子的局部最优位置
  if(fit>particle.pb_value)
     particle.pb_value=fit;
    particle.pbest_position=new_position;
     /*
     //粒子是否满足突变条件(属于 MR_GPSO_PP 添加
      的部分)
     if(GA)
       mutation(particle); normalization(particle);
     */
     # 输出中间数据,传递给 reducer
     // Repaticetovalue 方法将粒子转换为相应的中间输
        出类型.
     emit(key,Repaticetovalue(particle)).
```

**Function 2 MR_PSO_PP and MR_GPSO_PP Reduce**

```
define reducer(key ,value_list)
  //粒子列表,用来存储进来的粒子
  List<Particel> particle=None;
  Particle best=None;
  #对所有是输入 value_list,找到 gb_value
  for value in value_list:
     Particle record = new Particle(value);
    if (best is None) or (record.pb_vlaue>=best.gb_value):
     best = record;
    if not recond.is_message():
     particle.add(record);
  #更新所有粒子的全局最优信息(最优适应值和
   最优位置(即寻踪的投影方向))
  if particle is not None:
     particle.gb_candidate(best.gb_value,gbest_positon);
     emit(key,Repaticetovalue(particle));
```

### 3.2 投影后文本的分类

模型寻找到最优投影方向后将文本投影到该维空间, 采用 KNN 分类算法来对其进行分类, 同样使

KNN 算法基于 MapReduce 并行化.

3.2.1　MRKNN(基于 MapReduce 的 KNN 算法)过程及实现步骤　MRKNN 的分类过程如下:

input: 待分类文本, output: 分类结果.

step1: 将预处理好的待分文档集划分到不同的 Map 函数中;

step2: 文档 *d* 与通过投影方向投影到 1 维空间, 判断 *d* 最近的 *K* 个邻居(*key* 为文档对象, *value* 为文档特征向量);

step3: 判断 *d* 的类别; 输出中间结果(*key* 为 *d* 类标, *value* 为文档号 *N*);

step4: 进入 Reduce 阶段, 组合相同类标的文档.

以上 step1~step3 属 Map 函数的主要功能, step4 属 Reduce 函数的主要功能; 具体实现如 Function 3 和 Function 4 伪代码.

**Function 3 MRKNN Map**

```
//key 为文档集的偏移地址,
value 为文档集中的 key 所对应的文档数据
define mapper(key , value)
  #对文档查找 k 个最近邻,然后进行标类
  Document document=new Document(value);
  //d_z 为文档投影到该维的值
  d_z=knn_computz(document,projection_space);
  //找到该文档的 k 个最近邻,
training_set_z 为低维空间文档集合的信息.
find_k(d_z,training_set_z);
  lable(document.d_id);//判断文档类别,给文档标类
  #输出中间数据
// key 为类标 ,value 为文档 id, 都转换为 Text 类输出
    emit(key , Text(value)).
```

**Function 4 MRKNN Reduce**

```
//key 为类标,value_list 是 Map 函数传出经过
shuffle 和 merger 的同类标的文档编号列表
define reducer(key ,value_list)
  document_list;
#对类标相同的文档加入到同一个列表中
  for value in value_list:
    document_list.add(value.tostring);
#输出统计整合的结果,key 为类标
value 为该类的文档的集
    emit(key,Text(document_list));
```

### 3.3　基于 MapReduce 的 PSO 投影寻踪分类模型流程图

在文本分类应用中，首先进行数据预处理，其次通过基于 MapReduce 的粒子群投影寻踪模型寻找数据投影方向，再次将数据投影到 1 维空间后通过 KNN 方法来进行分类，整个应用过程如图 2.

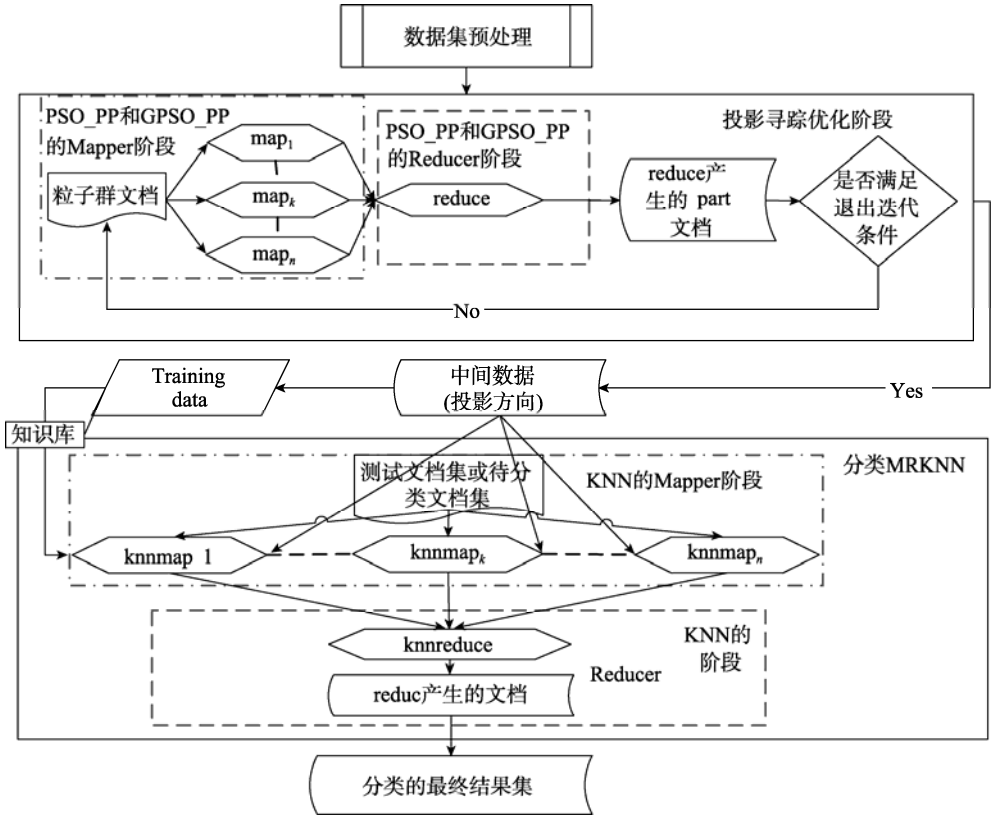



图 2　基于 **MapReduce** 的粒子群投影寻踪分类模型流程图

# 4 实验

## 4.1 Hadoop 集群平台的搭建

在实验室局域网中 8 台 PC 机群上进行 Hadoop 平台的搭建和配置. Hadoop 版本为 apache 发布的 Hadoop0.20.2, 操作系统为 ubuntu10.4, JDK 版本为 1.6.3; PC 机的硬件环境同为 Pentium(R) Dual-core CPU E6300 @2.8 GHz 双核处理器, ADAT 2Gbit 内存, Hitachi 320 Gbit 硬盘.

## 4.2 数据集预处理、粒子信息组成及参数设置

实验在复旦大学的中文文档集中进行, 去重后的数据集中训练集和测试集的总文档数分别为 8 213、6 163, 共 20 个类, 其中有 9 个大类, 11 个小类, 实验中选用前 9 个大类进行实验, 各类的训练集正例数及测试集正例数如表 1 所示.

**表1** 复旦大学的文档集中前**9**个类的名称及文档的个数

| 类别 | Economy | Sports | Computer | Politics | Agriculture | Environment | Art | Space | History |
|---|---|---|---|---|---|---|---|---|---|
| 训练集正例 | 1 369 | 1 204 | 1 019 | 1 010 | 847 | 805 | 510 | 506 | 466 |
| 测试集正例 | 1 127 | 980 | 591 | 989 | 635 | 371 | 286 | 248 | 468 |

实验主要目的在于比较本文模型在寻找投影效率和效果, 于是文本分类中的预处理阶段独立处理, 在该阶段每篇文档进行去重、分词, 进行了 $\chi^2$ 算法选择维数, 词的权重采用 LTC 算法, 将整个复旦文档集处理为 500 维的向量文档集, 每一维表示的是一个特征词.

实验中随机生成 200 个粒子信息(因为粒子群优化算法在比较复杂问题一般取 100 或 200 粒子), 由于预处理成的文档向量 500 维, 所以每个粒子相应的信息维数为 500 维, 并在按 3.1.1 节所述初始生成. 粒子的适应度是通过投影指标函数 $Q(z)=B(z)/D(z)$ 来衡量, 这个值在文本训练集通过 PSO 进行优化, $Q(z)$ 越大说明投影方向越好, 降低到低维后越能保持原数据的信息. 另外最大迭代次数 $T_{\max}$=300(实验发现在 300 迭代后基本不变, 所有本文分类的所有类别的分类过程中 $T_{\max}$ 都取 300), $C_1$=$C_2$=2.05, $W_0$=0.9, $W_t$=0.9–(0.9–0.4)$t/T_{\max}$ ($W_0$,$W_1$,…,$W_t$ 为迭代过程中 0~$t$ 次的惯性权重), 这些参数的设置根据基本的粒子群优化算法的一般值设定.

## 4.3 实验比较

文本进行了 2 组比较实验, 第1组是串行的粒子群投影寻踪模型和基于 MapReduce 的并行粒子群投影寻踪模型在寻踪最优投影方向时粒子迭代时间上的比较; 第 2 组是基于 MapReduce 的投影寻踪模型将文档投影到 1 维后的分类效果和原 500 维数据上的分类效果比较. 为突出算法效率效果的比较, 实验中与基于 MapReduce 的并行程序对应的串行程序都是采用同样的编程语言并调用同样的方法和函数, 以避免语言级的效率影响.

4.3.1 *投影寻踪实现效率比较* 对每个类别投影方向的寻优时, 都是按上述要求生产一群粒子, 形成个粒子文档. 在复旦大学的 3个文档集中训练, 经过迭代, 记录寻找到最优投影方向所用的时间. 表 2 给出了寻找训练 Economy 类区别其他类投影方向 S_PSO_PP 在一台相同的配置的 PC 机上执行的时间与 MR_PSO_PP 在不同数目的基于 Hadoop 的集群中执行的时间, 对它类具有同样效果.

从图 3 和表 2 中可知, 从执行时间来看, 在普通 PC 机群 Hadoop 分布式集群执行时, MR_PSO_PP 相比 S_PSO_PP 在时间上有了较大的提高. 当只用一台机器执行时, 由于通信等方面的耗时基于 MapReduce 的耗时要多一点, 但是当机器群扩大时, 可以看到执行时间迅速下降, 比单机执行耗时要少很多, 当机群用到 6 到 8 台, 时时间基本趋于稳定, 这与执行的任务大小有关, 当机器数目已经满足 MapReduce 分配时, 增加机器不再对执行时间产生显著影响. 对于到底要多大的机群来执行任务可以达到最好的效果和利用率,

**表 2 MR_PSO_PP 与 S_PSO_PP 的执行时间比(/s)**

| 机器数 | 1 | 2 | 4 | 6 | 8 |
|---|---|---|---|---|---|
| S_PSO_PP | 2 376 | 2 376 | 2 376 | 2 376 | 2 376 |
| MR_PSO_PP | 2 613 | 1 436 | 718 | 520 | 498 |

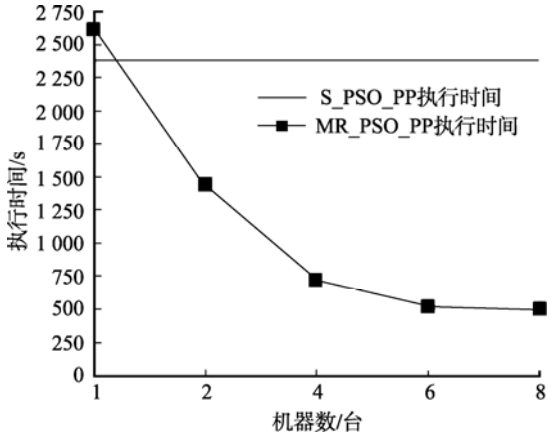



**图 3** 执行时间折线图

可以根据数据量的大小来决定. 实验中MR_GPSO_PP与其序列程序 S_GPSO_PP 执行效率相比具有同上述一样的结果.

4.3.2　*投影后的分类效果验证结果*　本文模型与基本的 KNN 算法直接对预处理文档进行分类(图表中记为 KNN)的效果进行对比，前 9 个大类的分类效果 $F_1$ 值如表 3 所示. MR_PSO_PP_KNN 是 MR_PSO_PP 算法得到投影方向，然后用 MRKNN 进行分类；MR_GPSO_PP_KNN 是在 MR_PS0_PP 中结合了粒子突变的 MR_GPS0_PP，再用 MRKNN 进行分类.

从表3和图 4 中可以看出，直接用粒子群算法得到的各个类的 $F_1$ 值基本上要比未投影的分类效果差一些，说明 MR_PSO_PP 在寻优过程中粒子较易陷入局部最优；而在粒子更新过程中加入突变时，可以看到MR_GPSO_PP投影寻踪后，得到的分类效果与原来的分类效果相比较，每个类的效果都非常接近，而且还有几个类要比原维数上的分类效果要好，如 Sports 等类；另外宏平均 $F_1$ 值(M_$F_1$)相差不到 0.54%，说明该方法能够有效的限制粒子快速陷入局部最优，更好地寻找到极好的投影方向. 说明在文本分类中可以通过基于 MapReduce 的粒子群投影寻找模型将文本从高维空间投影到低维空间再通过分类算法进行分类是有效可行的. 另外降维后分类算法的计算复杂度大大地降低，并且并行的KNN与串行的KNN相比在分布式集群环境中对大数据集处理的时效上更具优越性.

表 **3**　对复旦文档集的分类结果 $F_1$ 值的比较

| | Economy | Sports | Computer | Politics | Agriculture | Environment | Art | Space | History | $M\_F_1$ |
|---|---|---|---|---|---|---|---|---|---|---|
| KNN | 0.883 8 | 0.908 7 | 0.932 1 | 0.849 4 | 0.907 7 | 0.862 1 | 0.758 1 | 0.755 9 | 0.658 8 | **0.835 2** |
| MR_PSO_PP_KNN | 0.675 4 | 0.800 0 | 0.872 6 | 0.694 4 | 0.838 8 | 0.662 5 | 0.665 7 | 0.595 8 | 0.402 3 | **0.689 7** |
| MR_GPSO_PP_KNN | 0.858 3 | 0.928 7 | 0.917 0 | 0.834 7 | 0.899 4 | 0.882 3 | 0.752 4 | 0.768 2 | 0.635 0 | **0.830 7** |

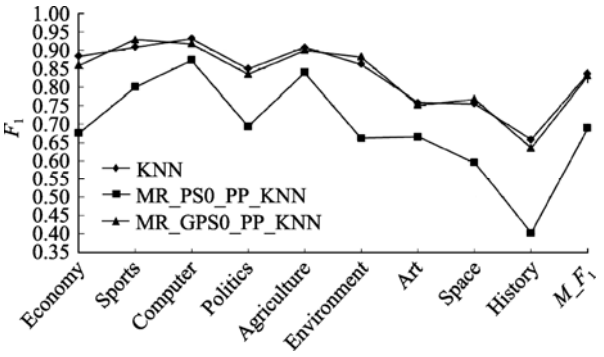



图 **4**　**9** 个大类分类效果折线对比图

## 5　结束语

本文研究工作表明，在投影寻踪模型用于文本降维后分类应用时，投影方向的寻优过程中，与串行算法相比，基于 MapReduce 的粒子群投影寻踪算法在普通 PC 集群环境下并行化的实现大大减少了数据的训练时间，这点尤其有利于在实际应用中对高维数据或大训练数据集中投影方向的训练，同时通过粒子群算法结合投影寻踪可以较好地找到一个保持原数据信息的投影方向，保证降维的效果；已知在保证降维后原数据特性的情况下，在低维空间对数据进行分析有更效率，所以本文研究对高维数据和大规模数据的处理具有一定意义.

## 6　参考文献

[1] Marina Mandelzweig, Aleksander B, Demko, et al. A projection method for the visualization of high-demensional biomedial datasets [J]. Electrical and Computer Engineering, 2003, 3: 1453-1456.

[2] 廖海波，万中英，王明文. 免疫进化的投影寻踪模型在文本分类中的应用 [J]. 广西师范大学学报：自然科学版，2011(1), 123-128.

[3] James Kennedy, Russell C Eberhart. Particle swarm optimization [J]. Neural Networks, 1995, 4: 1942-1948.

[4] 朱小平，赵曦. 一种改进的离散粒子群优化算法在 TSP 问题中的应用 [J]. 江西师范大学学报：自然科学版，2010，34(4), 369-373.

[5] Shi Y, Eberhart R C. Empirical study of particle swarm optimization[J]. Proceeding of Congress on Evolutionary Computation, 1999, 3: 1945-1949.

[6] Jimmy Lin, Chris Dyer. Data-Intensive text processing with MapReduce [EB/OL].[2012-01-18]. http://lintool.github.com/MapReduce Algorithms/MapReduce-book-final.pdf.

[7] Tom White. Hadoop: the definitive guide [M]. New York: O'Reilly Media Inc, 2009.

[8] 维基百科. 遗传算法 [EB/OL]. [2011.9]. http: //zh. wikipedia. org/wiki/%E9%81%97%E4%BC%A0%E7%AE%97%E6%B3%95.

# The Design and Implementing for Projection Pursuit Model Using PSO Based on MapReduce

HUANG Yi-ping, WAN Jian-yi, WAN Zhong-ying, WANG Ming-wen*

(College of Computer Information Engineering, Jiangxi Normal University, Nanchang Jiangxi 330022, China)

**Abstract:** It programs the model on MapReduce model. In the classification stage, KNN based on MapReduce is designed and used,and the classification experiment is performed on Fudan dataset. The result shows that parallel particle swarm optimization for projection pursuit based on MapReduce has both well effectiveness and higher efficiency than its serial counterpart.

**Key words**: projection pursuit; PSO; MapReduce; text classification; parallel

(责任编辑: 冉小晓)

(上接第 387 页)

# A Method for Query Intent Identification Based on Markov Network Clique

CAI Gui-xiu, WANG Ming-wen*, JIE An-quan, WANG Xiao-qing

(School of Computer Information Engineering, Jiangxi Normal University, Nanchang Jiangxi 330022, China)

**Abstract:** A new method about query intent classification is proposed. Making use of manually labeled queries form Sogou’s query log (about 2 250) as training data, and use the ten classes of data to construct the Markov network. So we can effectively get information of the queries. After this process research the queries in this data. Returning the relevance results of the queries, and classifying these results according to the classifier trained by the ten classes data. At last, the queries intent is predicted. In the experiments use the 11_avg and 3_avg as assessment process. Experiment results demonstrate that the algorithm presents some advantages compared with other methods.

**Key words**: query classification; Markov network; text classification; 11_avg; 3_avg

(责任编辑: 冉小晓)