基于多线程的顾客满意度 PLS 路径模型算法①

潘王海, 高艳艳

(天津大学 管理与经济学部, 天津 300072)

摘 要: 在计算顾客满意度模型时, 普遍使用结构方程模型来求解复杂的多维度关系. 而针对结构方程模型的计算, 偏最小二乘(PLS)有效地解决了模型分析过程中大量潜变量无法直接采样获得的问题, 建立了显变量与潜变量之间的关系, 增强了模型的直观性和通用性. 为了提升偏最小二乘算法的效率, 在针对结构方程模型的计算中, 利用多线程将模型计算并行化, 以提升计算效率. 探讨了基于 Java 多线程的顾客满意度 PLS 路径模型的优化算法, 并在多核计算机上实验证明, 多线程化的 PLS 路径模型能够显著提升计算速度.

关键词: 顾客满意度; PLS 路径模型; 多线程; 并行计算

Customer Satisfaction PLS Path Model Algorithm Implement Based on Multi-Thread

PAN Wang-Hai, GAO Yan-Yan

(School of Management and Economic, Tianjin University, Tianjin 300072, China)

Abstract: While computing Customer Satisfaction Model, structural equation model is always used to solve the influence of complicated multi-dimensional. Partial Least Squares (PLS), used in computing of structural equation model, works effectively with the problem that a large amount of latent variables cannot be sampling directly. In the process of model analysis, the PLS path model builds up the relationship between latent variables and manifest variables, and enhances the intuitive and generality of model. In order to improve the efficiency of PLS, we parallelize the computational process to get more efficiency. This article mainly talks about the PLS path model based on customer satisfaction model and the distributed algorithm based on Java multi-thread. It is proved on multi-core computer that PLS path model implementing on multi-thread can obvious improve speed of computing.

Key words: customer satisfaction; PLS path model; multi-thread; parallel computing

经济的发展带动了评定经济健康的标准, 顾客满意度逐渐成为了生产力指标评价的有益补充, 并且完善了现有的经济测评体系. Fornell 等人通过对 ACSI(美国顾客满意度指数)的研究发现, ACSI 越高公司表现越好. 顾客满意指数的变化, 可以与公司的盈利状况、股票价格、CPI、生产率和失业率、GDP 进行比较, 进而利用这些指数来预测宏观经济发展变化的趋势. 顾客满意指数已经成为一种新的度量现代经济运行质量的方式, 其重要性显而易见. 而最具代表性的满意度计算模型为 SCSB、ACSI 和 ECSI[1]. 我国基于满意度指数的构建起步比较晚, 从 1999 年开始, 我国开始中国国家顾客满意度的研究探究.

满意度的评测模型通常使用 PLS 和 LISREL 来建立, 相对而言, PLS 更具有优势一些, 针对潜变量可以明确求出对应的值, 不需要假定数据分布, 样本量要求比较低, 所以通常使用 PLS 路径模型进行满意度模型的建模与求解. 传统的 PLS 算法依据模型结构进行单线程求解, 当数据量较大, 测评模型较复杂时, 计算强度较大. 因此可以采用基于多线程的并行计算优化 PLS 路径模型的计算流程, 以提升计算效率.

1 PLS路径模型

通常的满意度模型采用结构方程模型进行建模, 结构方程模型包括三部分: 测量模型、结构模型和模

① 基金项目:国家科技支撑计划(2011BAH15B04)
收稿时间:2012-09-13;收到修改稿时间:2012-10-28

型假设[2].

1.1 PLS 路径模型分析

在结构方程模型中，测量模型表示显变量(即观测值)与潜变量之间的关系；结构模型表示潜变量之间的关系. 显变量是可以直接进行观测并得出分值的变量，而潜变量是不能直接测量的变量，需要通过观测变量来衡量. 结构方程模型结构示意图如下:

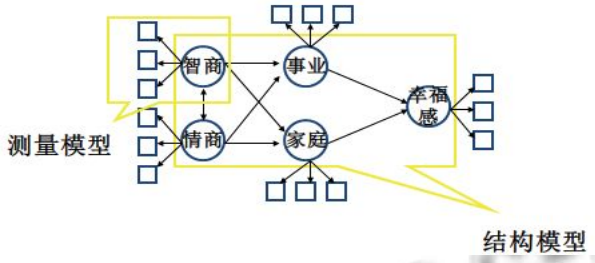

图 1　结构方程模型结构图

① 测量模型

测量模型分为反应式与构成式: 由显变量反应潜变量，即反应式；由显变量构成潜变量，即构成式. 反应式(The reflective way)通常写成如下的测量方程:

$$E(X_h \mid \xi) = \pi_{h0} - \pi_{h\xi} \tag{1}$$

其结构图为:

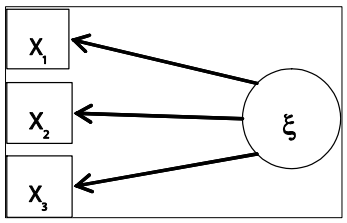

图 2　反应式结构图

构成式(The Constructive way)通常写成如下的方程:

$$E(\xi \mid X_1,...X_p) = \Sigma_{h=1}^{p} \tilde{W}_h X_h \tag{2}$$

其结构图为:

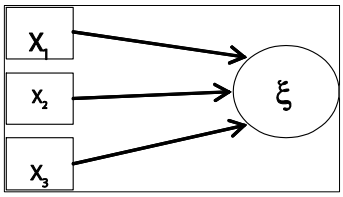

图 3　构成式结构图

其中,

$X_h = (X_1, X_2, ..., X_h)^T$ 为显变量的构成的向量

ξ 为潜变量的估计值

π_h 为参数的估计值

$\tilde{W}_h$ 为权重的向量

② 测量模型

结构模型表示潜变量之间的关系，每一个潜变量都与其他潜变量之间构成联系，整个结构模型构成一个树. 通常写成如下的测量方程:

$$\xi_j = \beta_{j0} + \sum_i \beta_{ji}\xi_i + \nu_j \tag{3}$$

其结构图为:

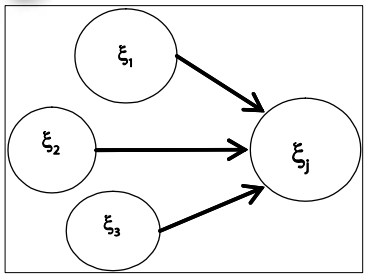

图 4　结构模型结构图

1.2 模型求解

PLS 路径模型求解分为测量模型计算和结构模型计算.

① 测量模型计算

首先根据具体分值与权重进行潜变量的估计，计算完成之后，需要进一步将 y 标准化为方差为 1:

$$y_j \propto \pm \sum_h w_{jh}(x_{jh} - \bar{x}_{jh}) \tag{4}$$

同时根据标准化后的权重计算出 $\hat{m}_j$ 值和 $\hat{\xi}_j$ 值.

$$y_j = \sum_h \tilde{w}_{jh}(x_{jh} - \bar{x}_{jh}) \tag{5}$$

$$\hat{m}_j = \sum_h \tilde{w}_{jh}\bar{x}_{jh} \tag{6}$$

$$\hat{\xi}_j = \sum_h \tilde{w}_{jh}x_{jh} = y_j + \hat{m}_j \tag{7}$$

② 结构模型计算

进行结构模型计算时，根据测量模型估计出的 y 进行 z_j 的估计.

$$z_j \propto \sum_{j':\xi_{j'}\text{与}\xi_j\text{相连}} e_{jj'}y_{j'} \tag{8}$$

当计算结构模型的时候，有三种不同的计算模式，分别是: Centroid Scheme, Factorial Scheme 和 Path

weighting Scheme. 这三种不同的模式在结构模型估计潜变量值时使用$e_{jj'}$的计算方法不同.

Centroid Scheme: $e_{jj'}$为y_j和$y_{j'}$之间的符号

Factorial Scheme: $e_{jj'}$为y_j和$y_{j'}$之间的相关系数

Path weighting Scheme: $e_{jj'}$为

y_j指向$y_{j'}$: 相关系数,

$y_{j'}$指向y_j: 回归系数

③ 估计w_{jh}

对于反应式

$$w_{jh} = \mathrm{cov}(x_{jh}, z_j) \tag{9}$$

对于构成式

$$w_j = (X_j^{'} X_j)^{-1} X_j^{'} z_j \tag{10}$$

④ 估计出潜变量值后, 再用回归方法重新计算各类系数.

2 基于多线程并行算法设计与实现

2.1 PLS 路径模型并行性分析

① 数据并行性

偏最小二乘路径模型的计算中, 数据的标准化、缺失值补充与初始化可以在数据拆分时同时完成. 每一个变量的数据将会反复进入测量模型计算, 故针对数据进行按测量模型拆分, 有利于并行计算. 所以数据层面并行性是可行的.

② 依赖关系

考虑依赖关系, 测量模型依赖于数据与上一次迭代结果. 结构模型依赖于测量模型的估计结果与上一次的迭代结果. 考虑依赖关系中的数据依赖, 因为针对数据只有读取操作, 而读取操作是并行友好的. 考虑上一次迭代结果, 针对本次计算也是读取操作, 是并行友好的. 所以根据依赖关系, 测量模型的计算是可以并行化进行.

③ 迭代计算

PLS 路径模型需要反复迭代计算测量模型与结构模型, 每一次计算都是由各部分独立计算完毕, 继而进行合并、下一步处理, 并再一次迭代. 各部分独立进行的计算属于并行友好, 可以进行多线程计算.

所以根据数据、依赖和迭代三部分的并行分析, 可以得出 PLS 路径模型是可以并行化的.

2.2 并行算法流程

Java 语言原生对多线程有着良好的支持, 在 java.util.concurrent 包中有着对并发编程的良好支持, 并且在 JVM 的进程中对并发有着良好的控制. 我们可以使用 JDK 中的 ThreadPoolExecutor 类来构建线程池, 使用 Runnable 接口实现对应的并行执行部分[3].

考虑 PLS 路径模型的计算步骤, 可以得出如下的算法流程.

① 读取数据, 进行数据异常值处理, 缺失值补全, 数据标准化.

② 根据变量对数据进行拆分, 考虑数据以读取为主, 选择将数据存储于 LinkedList 中, 以提供最快的遍历速度. 以 ConcurrentHashMap 作为变量与数据列表的存储.

③ 针对测量模型, 对权重进行初始化, 一般测量模型的初始化权重为$(1, 0, 0, \ldots, 0)^{\mathrm{T}}$.

④ 使用多线程依据式(4)到(7)进行测量模型运算, 估计潜变量的值.

⑤ 使用式(8)进行 z 值估计, 并使用 z 值, 根据测量模型的类型, 进行权重估计, 如式(9)和(10)所示. 估计出权重需要与上一次估计出的权重进行对比, 若两者差距较大, 则根据估计出的权重再次运算, 重新估计潜变量的值; 若权重已经收敛, 即两次的差小于10^{-5}, 则计算结束, 进入下一步.

⑥ 计算各类回归系数, 并计算出各个变量的估计值. 求出满意度结果. 流程图如图 5 所示:

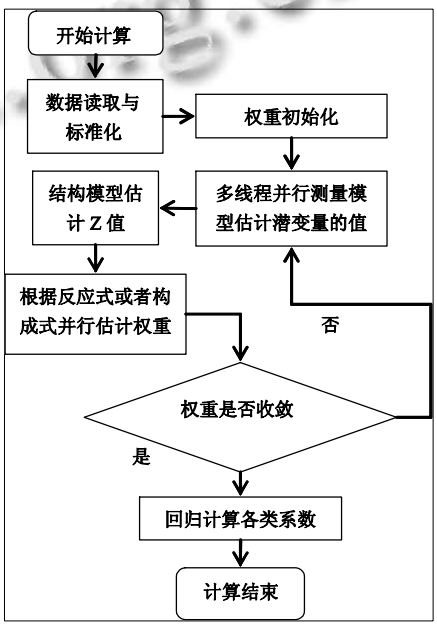

图 5 PLS 并行算法整体结构

2.3 模型算法实现

2.3.1 模型核心计算实现

① 权重新定算法

缺失值权重的计算: ①模式 A(Mode A): 权重根据公式计算; ②模式 B(Mode B): 当没有缺失值按照公式计算; 有缺失值时, 采用成对删除法把对应的缺失样本值删除, 即不考虑在内, 然后利用公式来计算权重[4]. 在存在缺失值时, 只要该样本中有一个显变量的值没有缺失, 不应该删除该样本, 应采用“权重新定”对权重进行处理. 权重在原有权重总和为 1 的基础上进行重新计算. 使用“权重新定”计算潜变量估计值的代码如下:

```
public double[] estimationByWSN(double[][] SX,double[] W) {
  //初始化数组以存储估计值
  double[] rtn = new double[SX.length];
  for (int i = 0; i < SX.length; i++) {
    double[] rowdata = SX[i];
    double wsum = 0.0; //去掉缺失值后的权重之和
double e = 0.0;
      for (int j = 0; j < rowdata.length; j++) {
        //当数据不是缺失值时, 即不是 NaN 时, 将权重相加, 估计值则为数据与权重相乘
        if (rowdata[j] != Double.NaN) {
          wsum += W[j];
          e += rowdata[j] * W[j];
        }
      }
      //当存在非缺失值, 将最后估计值使用权重之和标准化, 以满足权重之和为 1
      if (wsum != 0.0) {
        rtn[i] = e / wsum;
      }
      else {
      //当某变量的所有数据皆缺失, 则估计值为 NaN
        rtn[i] = Double.NaN;
      }
  }
  return rtn;
}
```

② 多线程线程池

多线程计算时, 需要维护线程的数目, 以节约系统资源, 避免大量的建立与销毁线程带来的损耗. 使用线程池对线程进行管理, 可以很好的复用线程, 并能够良好的处理线程的生命周期.

并行计算代码如下:传入测量模型对象, 对象中包括测量模型的基本信息与数据存储. 创建线程数目为 CPU 内核数目的两倍. 以保证最大程度的使用 CPU.

```
public void runInParallel(List<Measure> measures){
  int cpuCount=Runtime.getRuntime().availableProcessors();
  //建立线程池, 最小线程数为内核数目, 最大为内核数目的两倍,
  ThreadPoolExecutor threadPool = new
ThreadPoolExecutor(cpuCount, cpuCount*2, 3,
TimeUnit.MILLISECONDS,
  new ArrayBlockingQueue<Runnable>(cpuCount),
  new ThreadPoolExecutor.DiscardOldestPolicy());
  for (Measure measure:measures) {
    try {
      //创建任务并提交到线程池中
      //MeasureCalculator 实现 Runnable 接口, 并且在 run 方法中实现测量模型计算, 包括潜变量估计, 系数计算等
    threadPool.execute(newMeasureCalculator(measure));
  } catch (Exception e) {
    e.printStackTrace();
  }
 }
}
```

③ 回归系数计算

该计算方法中, 最核心的部分即为回归计算相关系数. 根据路径指向的不同, 测量模型分为反应式和构成式.

计算测量模型回归系数的核心代码如下:

```
public computeMeasureCoefficient(){
  if(isReflective()){
  //如果是反应式, 计算公式为, 则采用一元回归计算
  SimpleRegression regression = AnalysisUtil.
computeSingleRegression(xs.get(md),Ksai.get(lvId));
  //返回回归系数
  double b = regression.getSlope();
  }else if(isContructive()){
  //如果是构成式, 即由显变量指向潜变量, 计算公式为: E(ξ | X1,...Xp) = ∑_{h=1}^{p} W̃h Xh, 采用多元线性回归.
```

```
    //读取潜变量对应的显变量的数据
    Double[][] mvXs = readMvData(lvId);
    OLSMultipleLinearRegression reg = AnalysisUtil.
computeMultiRegression(ksai.get(lvId), //执行回归
    AnalysisUtil.transpose(mvXs)); //数组转置
    double[] bs = reg.estimateRegressionParameters();//获
得回归系数
    double r = reg.calculateRSquared();
    }
}
```

2.3.2 海量数据计算扩展

当数据量超过单机可以承受的范围，或者模型极度复杂，计算所需资源过大时，可以将多线程计算扩展为多机并发计算. 可以考虑采用 Web Service 进行远程计算，或者使用 RPC 通讯. 也可以使用主流架构如 Hadoop 进行并行编程. 使用 HDFS 进行数据存储，使用 Zookeeper 进行任务协调，使用 MapReduce 架构进行并行调度与计算[5].

3 实验效果

① 实验基础环境

实验分别在 1 核 2 核与 4 核的机器上运行，系统为 Fedora15，内存为 2G，运行环境 Java 版本为 Java 1.6_06. 除了 CPU 的区别外，其他环境配置三台机器皆相同.

② 实验数据集

实验使用的数据集为通过定向问卷调查获得的数据. 调查问卷基于学生满意度设计的中国大学生成长体验调查问卷进行调查. 该满意度模型的 PLS 路径模型共有 12 个潜变量，89 个显变量. 共收集调查问卷 1439 份，其中有效问卷 1365 份.

③ 实验结果

单线程运行时，分别针对流程不同阶段进行时间测量，测试分别得到数据处理，模型创建，迭代执行和结果处理的时间，结果如下:

表 1 模型单线程下各个环节时间表

样本数目	500	900	1300
数据读取与处理（ms）	238	321	553
模型创建及初始化（ms）	12	24	65
模型迭代计算（ms）	1870	4437	9438
结果生产与处理（ms）	31	71	119

可以发现，单线程运行时，数据读取与迭代计算为最消耗时间的两个部分，而初始化与结果处理，消耗时间相对比较少. 数据读取涉及到数据库会话的连接与销毁和数据传输的网络 IO 所占用的时间，其中数据库会话是很昂贵、很消耗资源的，但是并行读取会创建更多的会话，占用更多的资源，相比而言提升空间不大. 所以模型计算的瓶颈在于迭代计算.

迭代过程使用多线程并行化后，在三台不同内核数目的机器上进行完整的 PLS 路径模型求解的测试，使用数据集的数据量为 1300，得到的测试数据如下:

表 2 多核不同线程数计算时间对比

CPU内核数	单线程执行(ms)	多线程并行(ms)	时间倍数
1核	10175	11938	0.85
2核	10114	6012	1.7
4核	10051	3310	3.1

可以看出，多核的 CPU 在多线程执行时，执行速度上有较大提升，但是提升的幅度相比内核数目的提升仍要少一些. 因为多核 CPU 几个内核之间的调度仍然需要消耗资源，所以内核数目越多，提升倍数越少. 由于模型计算更多的是 CPU 密集性运算，单核的 CPU 在多线程运行下，反而出现多个线程争抢 CPU 时间，增多 CPU 调度次数的情况，消耗了更多的系统资源.

针对多核 CPU 机器下单线程执行，多核 CPU 可以提供更多的 CPU 时间片，执行效率上有一定提升.

4 结语

本文结合满意度中的结构方程模型和 PLS 路径建模，分析了 PLS 路径模型计算过程中的并行性，将满意度模型的 PLS 路径模型算法中相对消耗时间并且可以并行化的回归计算使用多线程方式并行计算，并且基于 CPU 核数进行测试对比，实验使用了一核，双核和四核的三种计算机，通过数据对比，可以发现多线程明显提升了模型的计算速度. 文章同时给出了基于 java concurrent 的代码实现.

参考文献

1 Fornell C,刘金兰.顾客满意度与 ACSI.天津:天津大学出版社,2006.

2 赵富强,张磊,陈钒.基于PLS路径模型的顾客满意度测评研究.北京理工大学学报,2010,12(4):61–65.

3 Oaks S, Wong H. Java Thread 3rd Ed.O'Reilly Inc. 2004,9.

4 Tenenhaus M, Vinzi VE, Chatelin YM, Lauro C. PLS path modeling. Computational Statistics and Data Analysis, 2005, 48(1).

5 Dean J, Ghemawat S. MapReduce: Simplied Data Processing on Large Clusters. Google Inc. 2004.