基于云计算的受限玻尔兹曼机推荐算法研究

郑志蕴 李步源 李 伦 李 钝
(郑州大学信息工程学院 郑州 450001)

摘 要 数据的指数级增长及算法本身的复杂性使受限玻尔兹曼机面临着计算效率的问题。在详细分析受限玻尔兹曼机的基础上,将受限玻尔兹曼机与Hadoop平台的并行计算架构相结合,提出基于云平台的受限玻尔兹曼机推荐算法。该算法通过复制机制解决数据相关性问题,并将传统的受限玻尔兹曼机过程分解为若干个Hadoop任务的循环,实现并行计算。实验结果表明,与在传统平台上的实现相比,基于Hadoop并行架构的受限玻尔兹曼机推荐算法在大体量数据集的条件下可大幅提高推荐计算效率。

关键词 协同过滤,受限玻尔兹曼机,并行处理,云计算,Hadoop
中图法分类号 TP391 **文献标识码** A

Research on Restricted Boltzmann Machines Recommendation Algorithm Based on Cloud Computing

ZHENG Zhi-yun LI Bu-yuan LI Lun LI Dun
(School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China)

Abstract Coupled with the exponential expansion of the data and the high computational complexity of Restricted Boltzmann Machines, efficient computing of Restricted Boltzmann Machines has become an important issue. Based on the detailed analysis, the article introduced Hadoop platform into Restricted Boltzmann Machines, and proposed Restricted Boltzmann Machines recommendation algorithm on cloud platform. The algorithm solves the problem of data correlation with replication mechanism, and divides traditional Restricted Boltzmann Machines process into several Hadoop jobs which implements parallel computing. In the experiments, the comparative analysis between Hadoop platform implementation and the previous implementation draws the conclusion that the Hadoop platform improves Restricted Boltzmann Machines computation efficiently under conditions of large data sets.

Keywords Collaborative filtering, Restricted boltzmann machines, Parallel processing, Cloud computing, Hadoop

1 引言

近年来,协同过滤算法在国内外得到了广泛应用[1-4]。与其它算法相比,协同过滤算法的推荐结果具有新颖、准确和个性化等优势。该算法通过分析用户特点,以兴趣相似度对用户群聚类,并根据同类用户对某一信息的评价,预测特定用户的喜好。按照不同的数据模型,可分为邻居模型、受限玻尔兹曼机模型(Restricted Boltzmann Machines, RBM)、因子模型和组合模型等。其中,受限玻尔兹曼机模型因其准确度较好,而受到较大关注。

在数据大爆炸的环境下,网站中需要处理的数据越来越多,处理过程也越来越复杂,如淘宝网的商品已超过一千万种,注册用户数已经达到3亿,每天产生的数据超过50TB。传统基于Web的架构在处理如此大数量级的数据时(如数据挖掘、个性化推荐等)已面临计算效率低、存储空间不足等问题,将数据处理业务迁往云计算平台已成为学术界和企业界的一种共识和趋势[5]。在众多的云计算模型中,Hadoop平台以计算效率高、计算稳定性好等特点,成为广泛应用的云计算平台标准。本文将协同过滤算法在云计算平台实现作为研究内容,以Hadoop作为云计算实现的基础平台,通过分析受限玻尔兹曼机理论及MapReduce的计算特点,利用数据冗余方法解决计算数据相关性问题,并将受限玻尔兹曼机过程分解为若干个Hadoop任务的循环,实现并行计算。实验表明,基于云计算的受限玻尔兹曼机推荐系统,在保证原有推荐准确度的条件下,解决了计算效率问题,得到了较好的应用效果。

2 受限玻尔兹曼机理论

2007年Hinton等人将受限玻尔兹曼机引入到协同过滤系统中[6]。受限玻尔兹曼机是一种两层的神经网络模型,由显示层和隐藏层组成,每层有若干个节点。其中,显示层节点表示对对象的评价,隐藏层表示评价对象的状态。处于相同层的节点相互无连接,不同层之间依靠消息来连接,当消息发生变化时,不同层的状态随之发生变化。其推荐过程分为初始化、训练学习和推荐3个阶段。在初始化阶段,随机初始化

到稿日期:2013-02-04 返修日期:2013-05-11
郑志蕴(1962—),女,博士,教授,主要研究方向为云计算、并行计算,E-mail:iezyzheng@zzu.edu.cn;李步源(1988—),男,硕士,主要研究方向为云计算、并行计算;李 伦(1978—),男,硕士,讲师,主要研究方向为并行计算;李 钝(1975—),女,博士,讲师,主要研究方向为信息检索、数据挖掘。

连接权重，设定隐藏层、显示层状态；在训练学习阶段，按照给定的学习样本不断更新各层状态及连接权重，直至参数小于阈值；在推荐阶段，结合学习阶段的结果，给出处于接受状态的显示层与隐藏单元(即推荐结果)。

下面以电影推荐为例来说明受限玻尔兹曼机推荐原理。

假设有一个训练数据集 $S(u,m,r)$，其中 u 表示用户，m 表示电影，r 表示用户对电影的评分集合，受限玻尔兹曼机推荐模型如图 1 所示。

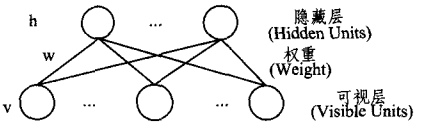

图 1　受限玻尔兹曼机模型

假设图 1 所示模型中的隐藏层 H 包括 K 个单元 $h_k(k=1,2,\cdots,K)$，h_k 取值 1 或 0，其中 1 表示接受状态，0 表示拒绝状态。显示层 V 包括 M 个单元 v_m，每个单元的取值为用户对电影的评分集合。权重 W 表示显示层与隐藏层之间的连接状态。

在给定隐藏层的情况下，假设显示层单元满足条件多项分布，而在给定显示层时隐藏单元满足伯努利分布，其分布函数如式(1)、式(2)所示[6-9]：

$$p(V_{u,m}=s|H)=\frac{\exp(b_{ms}+\sum_{k=1}^{K}h_{uk}w_{mks})}{\sum_{s'=1}^{S}\exp(b_{ms'}+\sum_{k=1}^{K}h_{uk}w_{mks'})} \tag{1}$$

$$p(H_{uk}=1|V)=\sigma(b_k+\sum_{m\in p_u}w_{mkv_{u,m}}) \tag{2}$$

式中，h_{uk} 表示用户 u 对应的第 k 个隐藏因子，$H\overset{\triangle}{=}(h_{uk})\in V^{U\times K}$；$W\overset{\triangle}{=}(w_{mks})\in V^{M\times K\times S}$ 为连接权重，U 表示用户数目的集合，M 表示电影数目的集合，K 表示隐藏层数目的集合，S 表示用户评分等级的集合，而 b_{ms} 与 b_k 为偏移，$\sigma()$ 是 sigmoid 函数，其数学表达式如下：

$$\sigma(x)=1/(1+e^{-x}) \tag{3}$$

显示层的边际分布函数如下：

$$p(v)=\sum_{h}\frac{\exp(-E(v,h))}{\sum_{v',h'}\exp(-E(v',h'))} \tag{4}$$

式中，$E(v,h)$ 代表激活能量，可用于计算各个单元的状态，其计算如式(5)所示：

$$E(v,h)=-\sum_{u=1}^{V}\sum_{k=1}^{K}\sum_{m=1}^{M}w_{mkv_{u,m}}h_kv_{u,m}-\sum_{u=1}^{V}\sum_{k=1}^{K}v_{u,m}b_{u,k}-\sum_{k=1}^{K}h_{u,k}b_k \tag{5}$$

在训练阶段，通过不断地训练学习升级权重和偏好参数，使用对比散度(Contrastive Divergence)来做计算，其公式如下：

$$\Delta w_{mkv_{u,m}}=\eta(\langle v_{u,m}h_k\rangle_{data}-\langle v_{u,m}h_k\rangle_T) \tag{6}$$

式中，η 表示学习因子，$\langle v_{u,m}h_k\rangle_{data}$ 表示隐藏层的状态为接受且显示层的状态值为 k 出现的频率，$\langle v_{u,m}h_k\rangle_T$ 表示使用 Gibbs 抽样所获得的抽样分布下的期望，T 代表抽样的次数。通常参数在开始阶段选取 $T=1$，伴随着学习过程的深入不断增加 T 的值，如果 T 的值足够大，$\langle v_{u,m}h_k\rangle_{data}$ 与 $\langle v_{u,m}h_k\rangle_T$ 之间的差值可以足够小。在实际应用中，通常 T 的取值为 1000，最大不超过 5000。

在推荐阶段，结合权重参数和偏好参数，使用式(7)－式(9)获得用户 u 对电影集合的预测评分：

$$\hat{v}_{u,m}=\sum_{s=1}^{S}s\cdot P(v_{u,m}=s|\hat{p}_u) \tag{7}$$

$$P(v_{u,m}=s|\hat{p}_u)=\frac{\exp(b_{ms}+\sum_{k=1}^{K}\hat{p}_{uk}w_{mks})}{\sum_{s'=1}^{S}\exp(b_{ms'}+\sum_{k=1}^{K}\hat{p}_{uk}w_{mks'})} \tag{8}$$

$$\hat{p}_{u,m}=P(H_{uk}=1|V)=\sigma(b_k+\sum_{m\in p_u}w_{mkv_{u,m}}) \tag{9}$$

3　MapReduce 架构下受限玻尔兹曼机算法设计与实现

3.1　MapReduce 计算框架

MapReduce[10]是一种并行处理框架，可自动处理如分布式存储、任务调度、负载均衡、容错处理以及网络通信等问题。与分布式计算和并行计算不同，MapReduce 编程模型要求并行处理的数据之间相互独立，不存在关联，并且表示为〈*key*，*value*〉对形式。MapReduce 的处理流程如图 2 所示：整个流程主要分为 map 和 reduce 两个过程。map 过程由 map 函数整合输入数据，reduce 过程由 reduce 函数进行逻辑处理。map 过程处理完成之后，Hadoop 对 map 结果进行排序，分发到不同的 reduce 过程处理，并将结果以〈*key*，*value*〉对形式输出[10,11]。其中，map 和 reduce 函数由用户编程实现。在资源分配和作业调度时，Hadoop 自动将数据、map 过程和 reduce 过程等操作分发给不同的节点进行处理，把计算结果输出到分布式文件系统 HDFS 中。

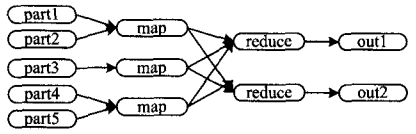

图 2　MapReduce 编程模型

3.2　算法设计

受限玻尔兹曼机算法分为训练过程和推荐过程，本文侧重描述训练过程的计算，推荐过程计算与训练过程类似。根据第 2 节给出的受限玻尔兹曼机理论，训练过程分为以下步骤：

1)随机初始化权重层数据；

2)利用式(2)－式(5)更新隐藏层状态；

3)利用式(1)、式(4)和式(5)更新显示层状态；

4)利用式(6)计算出对比散度 $\Delta w_{mkv_{u,m}}$ 更新连接权重；

5)重复步骤 2)－4)，直至 $\langle v_{u,m}h_k\rangle_{data}$ 与 $\langle v_{u,m}h_k\rangle_T$ 之间的差值小于阈值为止。

下面给出 MapReduce 编程框架下基本受限玻尔兹曼机训练算法。假设显示层为矩阵 V，权重为矩阵 W，隐藏层为矩阵 H，训练算法描述如下：

```
while(n 小于循环次数)
{
  s1:计算隐藏层 H=V * W;
  s2:计算权重更新参数 1,p_ij=h_j * v_i;
  s3:更新显示层 V'=H' * W^T;
  s4:更新隐藏层 H'=V' * W;
  s5:计算权重更新参数 2,n_ij=h_j' * v_i';
  s6:更新连接权重,w_ij=w_ij+η*(p_ij-n_ij);
}
```

理论上，当 p_{ij} 与 n_{ij} 之间的差值小到一定值后，认为训练数据集与实际情况一致。在实际应用中，常以循环次数来衡量训练数据与实际情况的吻合度。一般情况下，循环次数在1000左右，最大不超过5000次。

从上述过程可以看出，实现受限玻尔兹曼机训练过程的关键是矩阵运算，下面以训练算法第s1步 $H=V*W$ 说明Hadoop上的矩阵运算过程。假设矩阵 V 的元素表示为 v_{ij}，矩阵 W 的元素表示为 w_{jk}，依据矩阵相乘的原则，矩阵 H 的元素 $h_{ik}=\sum_{j=1}^{n} v_{ij}w_{jk}$。根据3.1节给出的Hadoop并行数据处理模型，设计矩阵运算过程如下。

在map阶段，将每个 v_{ij} 映射成 k 个〈*key*，*value*〉对，其中每个〈*key*，*value*〉对的 *key* 值分别为 $i\#1, i\#2, \cdots, i\#k$，*value* 值均为 v_{ij}。将每个 w_{jk} 映射成 i 份，其中每个〈*key*，*value*〉对的 *key* 值分别为 $k\#1, k\#2, \cdots, k\#i$，*value* 值均为 w_{jk}。

在reduce阶段，根据Hadoop系统默认的partion函数将 *key* 值相同的〈*key*，*value*〉对组成格式为〈$i\#k$，*list*(v_{i1}，w_{k1}，…)〉数据对送到同一个reduce进程上。其中，每个〈$i\#k$，*list*(v_{i1}，w_{k1}，…)〉包含计算 h_{ik} 元素需要的所有 v_{ij} 和 w_{jk}。reduce进程按照矩阵元素计算公式，将每个 *list*(v_{i1}，w_{k1}，…)中所有的值按顺序相乘后迭加，得到隐藏层矩阵元素 h_{ik}。

同理，其它节点的reduce进程也做同样的计算，得到所有隐藏层矩阵元素的值。

3.3 算法实现

3.3.1 数据规范

假设原始数据集包含 m 个用户、n 部电影，即用户评分的电影集合表示为 $V_{m\times n}$。数据集由文件输入，每一行由三元组〈用户ID，电影ID，评分〉组成，表示用户对某一部电影的评分。评分的数值范围为1至5分，最低分1表示非常不喜欢，最高分5表示非常喜欢。

在初始化阶段，将原始数据转换为统一定义的格式，具体规则如下：显示层矩阵元素表示为 $v\#i\#j\#r$，权重矩阵元素表示为 $w\#j\#k\#t$，隐藏层矩阵元素表示为 $h\#i\#k\#s$。$v\#i\#j\#r$ 表示显示层 v 中第 i 个用户对第 j 个电影的评分为 r，$w\#j\#k\#t$ 表示权重 W 中第 j 个显示层单元和第 k 个隐藏层单元之间的权重为 t，$h\#i\#k\#s$ 表示隐藏层 H 中对应的第 i 个用户评分所计算出来的第 k 个隐藏层单元的状态为 s。

例如，用户1对电影1、2、3的评分分别为2、4、5，原始数据评分记录为：

$$\begin{pmatrix}1 & 1 & 2\\ 1 & 2 & 4\\ 1 & 3 & 5\end{pmatrix}$$

对数据规范化后表示为：

$v\#1\#1\#2$

$v\#1\#2\#4$

$v\#1\#3\#5$

此数据作为map过程的输入数据。同样，也对权重数据和隐藏层数据进行规范化处理。

3.3.2 矩阵运算过程

整个受限玻尔兹曼机的实现基于大量的矩阵运算，本小节以隐藏层 H 为例给出矩阵运算的算法，其伪代码描述如下：

map函数：

```
if 显示层数据{
  while(m不大于显示层矩阵列数){
  key=矩阵元素行数+"#"+m
  value="V#"+矩阵元素列数+"#"+矩阵元素值
  }
}
else if 权重矩阵数据{
  while(n不大于权重矩阵行数){
  key=n+"#"+矩阵元素列数
  value="W#"+矩阵元素行数+"#"+矩阵元素值
  }
}
```

reduce函数：

```
while(value列表不为空){
  if 显示层数据{
    矩阵元素值存入数组 row
  }else if 权重层数据{
  矩阵元素值存入数组 column
  }
}
for{
  value=value+row*column
}
```

假设隐藏层有两个单元，显示层有3个单元，显示层矩阵为：

$$\begin{pmatrix}1 & 1 & 2\\ 1 & 2 & 4\\ 1 & 3 & 5\\ 2 & 1 & 5\\ 2 & 2 & 3\\ 2 & 3 & 1\end{pmatrix}$$

权重矩阵初始化为：

$$\begin{pmatrix}1 & 1 & 0.5\\ 1 & 2 & 0.6\\ 2 & 1 & 0.2\\ 2 & 2 & 0.4\\ 3 & 1 & 0.3\\ 3 & 2 & 0.2\end{pmatrix}$$

显示层数据经过规范化后为：

$v\#1\#1\#2$

$v\#1\#2\#4$

$v\#1\#3\#5$

$v\#2\#1\#5$

$v\#2\#2\#3$

$v\#2\#3\#1$

权重数据经过规范化后为：

$w\#1\#1\#0.5$

$w\#1\#2\#0.6$

$w\#2\#1\#0.2$

$w\#2\#2\#0.4$

$w\#3\#1\#0.3$

$w\#3\#2\#0.2$

map过程中，显示层矩阵第一行映射产生的〈*key*，*value*〉对

如下：

〈1#1,2〉〈1#2,2〉〈1#1,4〉〈1#2,4〉〈1#1,5〉〈1#2,5〉

而权重矩阵第一列映射产生的〈*key*,*value*〉对如下：

〈1#1,0.5〉〈2#1,0.5〉〈1#1,0.6〉〈2#1,0.6〉

〈1#1,0.2〉〈2#1,0.2〉

经过中间的 shuffle 过程，显示层数据变为：

〈1#1,list(2,4,5)〉〈1#2,list(2,4,5)〉

权重数据变为：

〈1#1,list(0.5,0.6,0.2)〉〈2#1,list(0.5,0.6,0.2)〉

在 reduce 过程中，*key* 值相同的〈*key*,*value*〉对被送到同一个 reduce 进程中。例如，MapReduce 计算框架将〈1#1,list(2,4,5)〉和〈1#1,list(0.5,0.6,0.2)〉送到同一个 reduce 进程进行处理。根据矩阵运算公式，将两个 list 中位置对应的元素值相乘迭加，得到 $2*0.5+4*0.6+5*0.2=4.4$，再根据式(3)的 sigmoid 函数计算出 h_{11}，即隐藏层矩阵第一行第一列的元素。

同理，map 进程和 reduce 进程对其它元素做相同的处理，得到整个隐藏层矩阵。

3.3.3 Hadoop 任务循环过程

受限玻尔兹曼机训练过程是一个循环运算过程，每次循环包含多个矩阵相乘运算。根据 3.2 节给出的基本受限玻尔兹曼机训练算法，本文将每一次循环过程分为 6 个子过程，分别为：

1) 计算隐藏层 H 子过程；

2) 计算权重更新参数 P 子过程；

3) 更新显示层 V' 子过程；

4) 更新隐藏层 H' 子过程；

5) 计算权重更新参数 N 子过程；

6) 更新连接权重 W 子过程。

根据 Hadoop 的计算特点，将以上每一个子过程作为一个 job，分别定义为 job1－job6。显然，每个 job 的计算过程依赖于前一个 job 的计算结果。因此，与常规的单 job 任务不同，需要利用到 Hadoop 提供的依赖关系组合式 MapReduce，其代码控制流程如下：

```
Configuration job1conf=new Configuration();
Job job1=new Job(job1conf,"Job1");
//job1 其他设置
Configuration job2conf=new Configuration();
Job job2=new Job(job2conf,"Job2");
//job2 其他设置
//其它 job 的设置
job2.addDepending(job1);//设置 job2 和 job1 的依赖关系
//其它 job 依赖关系设置的设置
JobControl JC=new JobControl("rbm");
JC.addJob(job1);//把第一个 job 加入到 jobcontorl 中
//将其它 job 也加入 jobcontrol 中
JC.run();
```

4 实验及分析

为了验证提出的受限玻尔兹曼机算法，搭建 Hadoop 集群，采用开放的数据集进行实验，并分析实验结果，评估此算法的性能和效率。

4.1 实验环境

实验中集群采用的 Hadoop 版本号为 0.20.205.0，两个集群的配置如下：1 个 Master、3 个 Slave 组成的集群和 1 个 Master、6 个 Slave 组成的集群。所有的机器配置情况相同，其配置参数均为：Pentium(R) Dual-Core E5300@3.06G 赫兹 CPU、2G 内存、320G 硬盘。集群中所有的计算机连接在一个交换机上，处于一个局域网中。

4.2 实验数据

在实验中，采用开放的 Netflix 电影评分数据集。该数据集包括了 480189 个用户 17770 部电影的 103297638 个评分。所有的评分值都是 1 到 5 分的整数值，数据集中评分越高表示用户对相应的电影的评价越高，即用户越喜欢此电影。为了测试不同容量数据集的计算效率，从这个数据集中分别随机抽取 100 万、500 万和 1000 万条评分记录作为训练集 set1、set2 和 set3，其余的作为评分测试集。

4.3 实验结果

为了对比不同计算资源规模对算法执行性能的影响，配置了 3 种计算环境，分别是：(1)单节点；(2)1 个 Master、3 个 Slave 的 Hadoop 集群；(3)1 个 Master、6 个 Slave 的 Hadoop 集群。一共进行两组实验，第 1 组是算法循环次数为 1 的实验，第 2 组是算法循环次数为 10 的实验，分别记录它们在上述 3 种实验环境下输入不同数据集的计算时间。

在第一组实验中，设定循环次数为 1，输入数据集分别为 set1、set2 和 set3。此时，在单节点、1 个 Master/3 个 Salve 和 1 个 Master/6 个 Salve 这 3 个实验环境下的计算时间如图 3 所示。

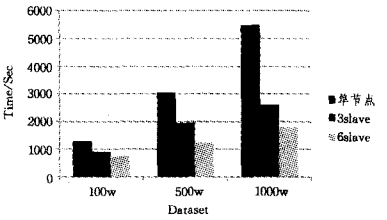

图 3 循环次数为 1 时的实验结果

由图 3 可以看出，当输入的数据量为 100 万时，受限玻尔兹曼机算法在 Hadoop 集群上的运行时间与在单节点上的运行时间大致相同，这是由 Hadoop 的计算框架特性决定的。Hadoop 将计算结果存储在磁盘中，master 需要频繁的任务调度，因此在数据量较小或计算比较简单时体现不出计算优势。随着数据输入量的增大，受限玻尔兹曼机算法在 Hadoop 集群上运行时间与在单节点上运行时间的差值越来越大。当输入数据量为 1000 万条时，其计算速度提高了一倍以上。很明显，当数据量增大时，Hadoop 程序的计算复杂度提高，计算效率也随之提高。

Hadoop 集群中节点增多，意味着其运算能力的加强，从图 3 可以看出，6 个 slave 节点的 Hadoop 集群运算效率比 3 个 slave 节点的 Hadoop 集群要高 10%～30%。当然，并不是机器越多越好，每个节点处理一个 map 任务，当节点数多于 map 任务时，节点的增多并不能提高运算效率，此时多余的机器会被闲置。所以在理想的情况下，map 任务数最好是 Hadoop 集群节点数的倍数，这样才能有效充分利用 Hadoop 集群的运算能力。

在第 2 组实验中，设定循环次数为 10，输入数据集分别为 set1、set2 和 set3。在上述 3 种情况下，其实验结果如图 4 所示。

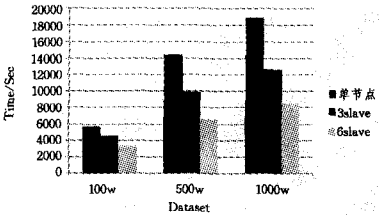

图 4　循环次数为 10 时的实验结果

第 2 组实验计算量是第一次的 10 倍，但计算时间只增加 4 倍左右，这表明随着数据量的增加，计算效率在提高。通过两次实验的比较可以看出，数据量越大，计算越复杂，受限玻尔兹曼机算法在 Hadoop 集群上的优势越显著。

结束语　本文主要针对受限玻尔兹曼机传统实现算法在数据量较大时计算效率较低的问题，提出利用云计算 Hadoop 平台实现受限玻尔兹曼机推荐的方法。在分析经典受限玻尔兹曼机推荐过程的基础之上，提出一种基于云平台的受限玻尔兹曼机推荐算法，并详细说明了推荐原理和实现过程。最后，通过一系列实验来验证其可行性和有效性。在大体量数据集的实验条件下，当输入数据集为 set1 时，Hadoop 平台的计算时间与单节点的计算时间持平。随着数据量和计算节点增加，Hadoop 逐渐表现出较大的优势，计算效率明显提高。这说明 Hadoop 在大体量数据集环境下能有效提高受限玻尔兹曼机推荐算法的计算效率。然而，受 Hadoop 集群节点少、物理机配置较低等条件的限制，此实验还未完全体现 Hadoop 的计算优势。下一步工作将从解决这些问题出发，进一步改进算法，充分发挥 Hadoop 在受限玻尔兹曼机推荐计算中的作用。

参 考 文 献

[1] 范波，程久军. 用户间多相似度协同过滤推荐算法[J]. 计算机科学，2012(1)：23-26

[2] 张光卫，李德毅，李鹏，等. 基于云模型的协同过滤推荐算法[J]. 软件学报，2007(10)：2403-2411

[3] 许海玲，吴潇，李晓东，等. 互联网推荐系统比较研究[J]. 软件学报，2009(2)：350-362

[4] 马宏伟，张光卫，李鹏. 协同过滤推荐算法综述[J]. 小型微型计算机系统，2009(7)：1282-1288

[5] 李乔，郑啸. 云计算研究现状综述[J]. 计算机科学，2011(04)：32-37

[6] Salakhutdinov R, Mnih A, Hinton G. Restricted Boltzmann Machines for Collaborative Filtering[C]// Proceedings of the 24th International Conference on Machine Learning. 2007：791-798

[7] Hinton G. A Practical Guide to Training Restricted Boltzmann Machines[EB/OL]. http://www.cs.toronto.edu/~hinton/absps/guideTR.pdf, 2010-08-02

[8] Fischer A, Igel C. An Introduction to Restricted Boltzmann Machines[C]// Progress in Pattern Recognition, Image Analysis, Computer Vision and Applications. 2012：14-36

[9] Cueto M A, Morton J, Sturmfels B. Geometry of the Restricted Boltzmann Machine[C]// AMS Special Session on Algebraic Methods in Statistics and Probability. 2010, 516：135-153

[10] Jeffrey D, Sanjay G. Mapreduce：Simplified data processing on large clusters[C]// Proceedings of the Sixth Symposium on Operating Systems Design and Implementation. 2004：137-149

[11] Apache HDFS Architecture[EB/OL]. http://hadoop.apache.org/docs/hdfs/current/dfs_design.html, 2011-04-12

(上接第 250 页)

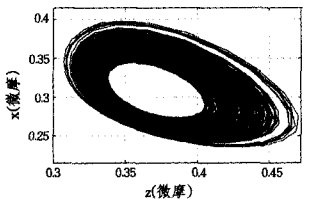

图 6　当 $k_{out}=9.86$，系统(1)的相图

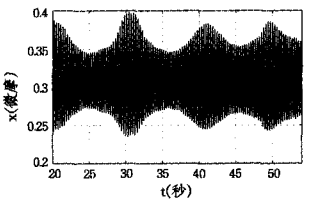

图 7　当 $k_{out}=9.86$，系统(1)的拟周期簇振荡图

结束语　Borghans-Dupont 系统较为复杂，用传统的频域分析方法[12,13]需要进行拉普拉斯变换，并构造传输矩阵，理论分析较繁琐，算法特性不强。我们利用构造中心流形的方法，算法设计思想清晰，而且执行效率高，并且通过设计的中心流形算法和对所得数据的分析，也获得了系统产生钙振荡的理论判据，发现这种振荡现象的产生与消失是由于 Hopf 分岔导致的。而且通过分析发现，系统随着分岔参数的改变，出现了两类不同的 Hopf 分岔，分别为超临界 Hopf 分岔和亚临界 Hopf 分岔。最后利用计算机仿真，绘制了系统的分岔图，并通过选取分岔区域中不同分岔参数，绘制了相应的相图与时序图，仿真的结果说明了振荡现象的存在，也验证了理论分析的结果。

参 考 文 献

[1] Kummer U, Olsen L F, Dixon C J, et al. Switching from simple to complex oscillations incalcium signaling[J]. Biophys J, 2000, 79：1188-1195

[2] Woods N M, Kuthbertson K S R, Cobbold P H. Agonist-induced *oscillations in hepatocytes*[J]. *Cells Calcium*, *1987*, *8*：*79-100*

[3] Goldbeter A. Biochemical oscillations and cellular rhythms[M]. Cambridge：Cambridge University Press, 1996

[4] Borghans J A M, Dupont G, Goldbeter A. Complex intracellular calcium oscillations. A theoretical exploration of possible mechanisms[J]. Biophys Chem, 1997, 66(1)：25-41

[5] Nabajyoti D, Tarini K D. Determination of supercritical and subcritical Hopf bifurcation on a two-dimensional chaotic model [J]. International Journal of Advanced Scientific and Technical Research, 2012, 1：207-220

[6] Jing X, Yu Z X, Yuan R. Stability and Hopf bifurcation in a symetrric lotka-volterra predator system with delays [J]. Electric Journal of Differential Equations, 2013：1-16

[7] 严传魁，刘深泉. 动态 IP_3-Ca^{2+} 振荡模型的数值分析[J]. 生物数学学报，2005，21(5)：339-344

[8] 周莉莉，李旭东，常玉. Houart-Dupont 钙振荡模型的复杂动态[J]. 北京化工大学学报，2010，37(3)：134-139

[9] Wiggin S. Introduction to applied nonlinear dynamical systems and chaos [M]. Berlin：Springer, 1990

[10] Jing Z J, Chang Y, Guo B L. Bifurcation and chaos in discrete FitzHugh Nagumo systems [J]. Chaos Solitions and Fractals, 2004, 21(3)：701-710

[11] 王青云，石霞，陆启韶. 神经元耦合系统的同步动力学[M]. 北京：科学出版社，2008

[12] 李绍文. 连续时延神经网络模型的 Hopf 分岔分析[J]. 计算机科学，2002，29(9)：47-49

[13] Gentile F S, Moiola J L, Paolini E E. On the study of bifurcation in delay-differential equations：A frequency-domain approach [J]. Int J Bifurcation Chaos, 2012, 22(6)：125-137