计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(2):94−100 [doi: 10.15888/j.cnki.csa.006781]
©中国科学院软件研究所版权所有.
E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

# 基于改进 Apriori 算法的糖尿病预诊系统①

张　冲, 张云华

(浙江理工大学 信息学院, 杭州 310018)
通讯作者: 张　冲, E-mail: chung567115@163.com

**摘　要**: 本研究改进了 Apriori 算法, 并应用于糖尿病高危因素关联分析. 改进算法通过单次扫库, 将事务数据库转化为布尔矩阵, 依照特定性质对矩阵进行压缩, 极大减少运算量和冗余. 基于此, 本研究设计了一款糖尿病预诊分析系统, 系统采用时下最新技术栈, 微服务框架 Spring Boot、RPC 框架 Dubbo、消息队列中间件 RabbitMQ、缓存 Redis 以及数据库 MySQL. 该系统辅助医护人员进行糖尿病诊断, 为用户自诊提供有效依据.

**关键词**: Apriori 算法; 糖尿病; 关联分析; 微服务; 预诊系统

引用格式: 张冲,张云华.基于改进 Apriori 算法的糖尿病预诊系统.计算机系统应用,2019,28(2):94–100. http://www.c-s-a.org.cn/1003-3254/6781.html

## Diabetes Prediagnosis System Based on Improved Apriori Algorithm

ZHANG Chong, ZHANG Yun-Hua

(School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China)

**Abstract**: This study has improved Apriori algorithm and applied it to association analysis of diabetic risk factors. The improved algorithm transforms transaction database into a Boolean matrix in one scan time and compressing matrix according to its specific properties, it greatly reduces computation and redundancy through this way. Based on above, a diabetes prediagnosis system has been designed which adopts the latest technology stack such as micro-service framework Spring Boot and RPC framework Dubbo, RabbitMQ as message queue middleware, Redis as cache, and MySQL as database. The system assists medical staff in diagnosing diabetes and provides an effective basis for users' self-diagnosis.

**Key words**: Apriori algorithm; diabetes; association analysis; micro service; prediagnosis system

## 1　引言

糖尿病 (diabetes mellitus, DM) 已成为世界性疾病, 据世界卫生组织 (WHO) 报道, 2025 年全球 DM 患者将达到 3 亿. 而更严重的是, DM 相关疾病死亡人数为每年 320 万[1]. 随着人们生活水平的提高和生活方式的改变, DM 发病率呈现低龄化, 我国的 DM 患病率急剧增加. 有关调查显示, 在我国经济发达地区 DM 患病率已高达 9%~10%[2].

一方面, 由于各方面诊断差异, 糖尿病常被漏诊误诊, 对患者和社会医疗造成巨大负担. 另一方面, 互联网发展迅猛, 医疗领域产生了海量临床数据, 人工处理和分析所耗费成本过大. 因此, 需要一套良好的系统来完成对大量已确诊病例的挖掘分析.

使用关联规则可以挖掘分析高危因素与糖尿病之间的关系, 但是, 经典 Apriori 算法主要存在两大缺陷[3]: ① 重复扫描数据库, 导致大量时间耗费于 I/O 操作[4]. ② 频繁项集自身连接时产生大量候选集, 导致空间消耗以指数形式增长. 针对以上缺陷, 本研究使用矩阵压缩[5]的方式对 Apriori 算法进行改进, 并应用于糖尿病高危因素分析. 基于此, 设计了一款糖尿病预诊分析系统,

① 收稿时间: 2018-08-19; 修改时间: 2018-09-18; 采用时间: 2018-09-26; csa 在线出版时间: 2019-01-28



© 中国科学院软件研究所 http://www.c-s-a.org.cn

该系统能够分析患病概率, 辅助医疗诊断.

## 2 改进算法

### 2.1 经典 Apriori 算法

关联分析中的 Apriori 算法是一种最具影响力的挖掘关联规则频繁项集的算法, 其核心是基于两阶段频繁项集思想的递推算法. 即首先找出满足最小支持度的所有频繁项集, 然后探索同时满足最小支持度阈值和最小置信度阈值的强关联规则, 从中发现隐藏在数据间的相关性. 将经典 Apriori 算法用数学语言可描述如下:

记项集 $X$ 与 $Y$, 关联规则可表示为如下蕴涵式:

$$R = X \to Y(X \cap Y = \varnothing) \tag{1}$$

记式 (1) 中项集 $X$ 出现的事务次数为$\sigma(X)$ , 事务数据库为 $T$, 则支持度的定义可表示为式 (2):

$$s(X \to Y) = \frac{\sigma(X \cup Y)}{|T|} \tag{2}$$

置信度的定义可表示为式 (3), 其中, 支持度大于阈值的项集, 称作频繁项集, 记为$F$.

$$c(X \to Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} \tag{3}$$

经典 Apriori 算法使用逐层搜索的迭代方法, 用 $k$-项集探索 ($k$+1)-项集. 首先找到频繁 1-项集, 再依据频繁 1-项集寻找频繁 2-项集, 直到找出所有频繁项集, 并产生大于阈值的关联规则, 如式 (4):

$$\begin{aligned} s(X \to Y) \ge min\_sup \\ c(X \to Y) \ge min\_conf \end{aligned} \tag{4}$$

记 $k$-项集为$C_k$, 记频繁 $k$-项集为$F_k$, 记事务长度为 $|T_k|$, Apriori 算法有如下三条性质:

性质 1. $F_k$出现的事务次数不小于最小支持数.

性质 2. 长度小于 $k$ 的事务不包含任何$F_k$[6].

性质 3. 由$C_{k-1}$生成$C_k$过程中, 若$C_{k-1}-2$项不同的项做自身连接, 将得到冗余项集或非频繁项集[7].

### 2.2 改进 Apriori 算法

经典算法每找一个频繁项集$F_k$都需要扫库一次, 导致大量时间耗费于 I/O 操作; 其次, 经典算法探索下一个频繁项集过程中做自身连接来产生候选项集时, 存在大量冗余.

现有较多 Apriori 算法的优化或改进方案, 文献[8]将算法剪枝与矩阵相联系来减少扫库次数, 提高了算法效率, 但未处理自连接产生的冗余. 文献[9]采用事务矩阵相乘的方法得到频繁项集, 一定程度上减少 I/O 操作, 但在事务数据库体量偏大时, 矩阵相乘将会占用较多 CPU 计算时间.

尽管 Apriori 算法已存在诸多改进方案, 但从空间占用率上讲, 较少有改进方案处理自连接的冗余, 从时间效率上讲, 也尚有进一步优化和改进的余地. 本文改进算法在自连接过程中去除项集冗余, 缩减空间占用率, 以避免无效计算. 并用事务布尔矩阵压缩的方式对计算方式进行改进, 进一步提高算法时间性能.

记项 $I$ 的向量为$\overline{B_i}$, 如式 (5). 其中 $m$ 为事务总数, $T_i$为第 $i$ 个事务.

$$\overline{B_i} = \begin{bmatrix} t_{1i} \\ t_{2i} \\ \vdots \\ t_{mi} \end{bmatrix}, \quad t_{ji} = \begin{cases} 0, & I \notin T_i \\ 1, & I \in T_i \end{cases} \tag{5}$$

根据式 (5) 建立形如式 (6) 的事务布尔矩阵 **B**. 其中 $m$ 为事务总数, $n$ 为项总数.

$$\mathbf{B} = \begin{bmatrix} \overline{B_1} & \overline{B_2} & \cdots & \overline{B_n} \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1n} \\ t_{21} & t_{22} & \cdots & t_{2n} \\ \vdots & \vdots & & \vdots \\ t_{m1} & t_{m2} & \cdots & t_{mn} \end{bmatrix} \tag{6}$$

记 2-项集$\{I_i, I_j\}$的向量为$\overline{B_i} \times \overline{B_j}$, 根据式 (5) 定义形如式 (7) 的运算.

$$\overline{B_i} \cdot \overline{B_j} = \begin{bmatrix} t_{1i} \times t_{1j} \\ t_{2i} \times t_{2j} \\ \vdots \\ t_{mi} \times t_{mj} \end{bmatrix} \tag{7}$$

结合以上定义与 Apriori 算法的三大性质, 可推导得到如下三条对布尔矩阵进行压缩的规则:

规则 1. 布尔矩阵里, 项$\overline{B_i}$出现事务次数不小于最小支持数, 即列应满足式 (8):

$$\sigma\left(\overline{B_i}\right) \ge |\mathrm{T}| \times min\_sup \tag{8}$$

规则 2. 布尔矩阵里, 长度小于 $k$ 的事务可以在挖掘$F_k$过程中删除. 即行应满足式 (9):

$$|T_{id}| \ge k \tag{9}$$

规则 3. 布尔矩阵里, 满足前 $k$–2 项相同的项才有必要在挖掘$F_k$过程中进行自身连接.

下面分析一个建立事务布尔矩阵, 在经典算法的基础上对扫库和自连接步骤进行改进, 即根据压缩矩



© 中国科学院软件研究所 http://www.c-s-a.org.cn

阵获取频繁项集和探索关联规则的实例:

① 输入事务数据库 $T$, 如表 1. 输入最小支持度 $min\_sup = 0.5$, 最小置信度$min_conf = 0.7$.

表 1 事务数据库

| TID | Transaction |
|---|---|
| 1 | A, B, C, D |
| 2 | A, B, C |
| 3 | C, D |
| 4 | A, B, E |

② 扫描事务数据库, 建立候选布尔矩阵$\mathbf{B}'_1$, 并于首行首列和末行末列各类辅助值, 如表 2 所示.

表 2 候选布尔矩阵 $\mathbf{B}'_1$

| | A | B | C | D | E | $\lvert T_{id}\rvert$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 0 | 4 |
| 2 | 1 | 1 | 1 | 0 | 0 | 3 |
| 3 | 0 | 0 | 1 | 1 | 0 | 2 |
| 4 | 1 | 1 | 0 | 0 | 1 | 3 |
| $\sigma(\overline{B_i})$ | 3 | 3 | 3 | 2 | 1 | |

③ 根据规则 1 压缩矩阵 $\mathbf{B}'_1$, 即保留 A、B、C、D 列, 得到矩阵 $\mathbf{B}_1$ 如表 3. 分析 $\mathbf{B}_1$ 得频繁 1-项集如下:

$$F_1 = \left\{ \{A\} \quad \{B\} \quad \{C\} \quad \{D\} \right\}$$

表 3 频繁布尔矩阵 $\mathbf{B}_1$

| | A | B | C | D | $\lvert T_{id}\rvert$ |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 4 |
| 2 | 1 | 1 | 1 | 0 | 3 |
| 3 | 0 | 0 | 1 | 1 | 2 |
| 4 | 1 | 1 | 0 | 0 | 2 |
| $\sigma(\overline{B_i})$ | 3 | 3 | 3 | 2 | |

④ 根据规则 2 压缩矩阵$\mathbf{B}_1$, 即保留所有行; 再根据规则 3 对矩阵$\mathbf{B}_1$进行自身连接, 即连接 AB、AC、AD、BC、BD、CD 列; 最后得到矩阵$\mathbf{B}'_2$如表 4.

表 4 候选布尔矩阵 $\mathbf{B}'_2$

| | AB | AC | AD | BC | BD | CD | $\lvert T_{id}\rvert$ |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 6 |
| 2 | 1 | 1 | 0 | 1 | 0 | 0 | 3 |
| 3 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| $\sigma(\overline{B_i})$ | 3 | 2 | 1 | 2 | 1 | 2 | |

⑤ 重复③: 根据规则 1 压缩矩阵$\mathbf{B}'_2$, 即保留 AB、AC、BC、CD 列, 得到矩阵$\mathbf{B}_2$如表 5. 分析$\mathbf{B}_2$可得频繁 2-项集和关联规则如下:

$$F_2 = \left\{ \{AB\} \quad \{AC\} \quad \{BC\} \quad \{CD\} \right\}$$

$$R_2 = \left\{ \{A \to B\} \quad \{B \to A\} \quad \{D \to C\} \right\}$$

表 5 频繁布尔矩阵 $\mathbf{B}_2$

| | AB | AC | BC | CD | $\lvert T_{id}\rvert$ |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 4 |
| 2 | 1 | 1 | 1 | 0 | 3 |
| 3 | 0 | 0 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 | 0 | 1 |
| $\sigma(\overline{B_i})$ | 3 | 2 | 2 | 2 | |

⑥ 以此类推, 根据规则 1 和规则 2 不断压缩矩阵, 直到$\mathbf{B}'_i \neq \emptyset$, 算法结束.

结合上述分析, 改进 Apriori 算法伪代码如下:

input min_sup, min_conf, $k = 0$; scan transaction $T$

$$\mathbf{B}'_1 = \begin{bmatrix} & \overline{B_1} & \overline{B_2} & \cdots & \overline{B_i} & \lvert T_{id}\rvert \\ T_{id1} & t_{11} & t_{12} & \cdots & t_{1i} & \sum_{j=1}^{i} t_{1i} \\ T_{id2} & t_{21} & t_{22} & \cdots & t_{2i} & \sum_{j=1}^{i} t_{2i} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ T_{id_{\lvert T\rvert}} & t_{\lvert T\rvert 1} & t_{\lvert T\rvert 2} & \cdots & t_{\lvert T\rvert i} & \sum_{j=1}^{i} t_{\lvert T\rvert i} \\ \sigma(\overline{B_i}) & \sum_{j=1}^{\lvert T\rvert} t_{j1} & \sum_{j=1}^{\lvert T\rvert} t_{j2} & \cdots & \sum_{j=1}^{\lvert T\rvert} t_{ji} & \end{bmatrix}$$

do

$k = k + 1$

for each $\overline{B_i} \in \mathbf{B}'_k$

if $(\sigma(\overline{B_i}) < \lvert T\rvert \times min\_sup)$ delete $\sum col(\overline{B_i})$

gain $\mathbf{B}_k$ and output frequent itemsets $F_k$

for each $f_k^* \subseteq (every f_k \in F_k)$ and $f_k^* \neq \emptyset$

if $(\frac{\sigma(f_k)}{\sigma(f_k^*)} \geqslant min\_conf)$ output rule $f_k^* \to f_k - f_k^*$

for each $T_{id} \in T$

if $(\lvert T_{id}\rvert < k)$ delete $\sum row(T_{id})$

for each $\overline{B_i}, \overline{B_j} \in \mathbf{B}'_k$

if $(item_i = item_j$ from 1 to $k$ - 2) $\overline{B_{ij}} = \overline{B_i} \cdot \overline{B_j}$

gain $\mathbf{B}'_{k+1}$

while $(\mathbf{B}'_{k+1} \neq \emptyset)$

### 2.3 算法分析与对比

为验证性能是否得到提升, 对两个算法进行实验对比. 实验环境为: Intel Core i5 CPU 3.20 GHz, 4 GB 内存, 512 GB 硬盘, Windows 10 操作系统, 在 Matlab R2017b 中得到以下关于两个算法的性能对比图.



© 中国科学院软件研究所 http://www.c-s-a.org.cn

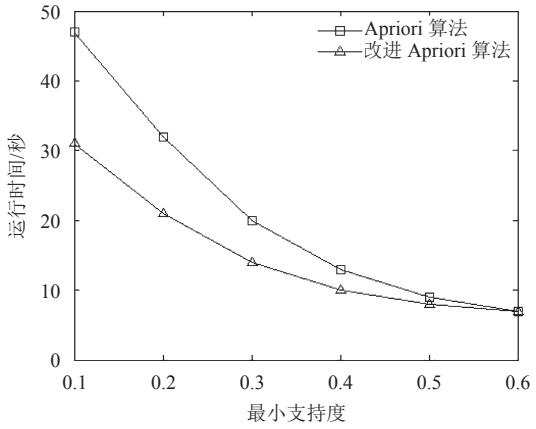


图 1 最小支持度与运行时间

从图 1 可以看出, 在相同的最小支持度下, 改进 Apriori 算法在运行速度上得到了明显性能提升.

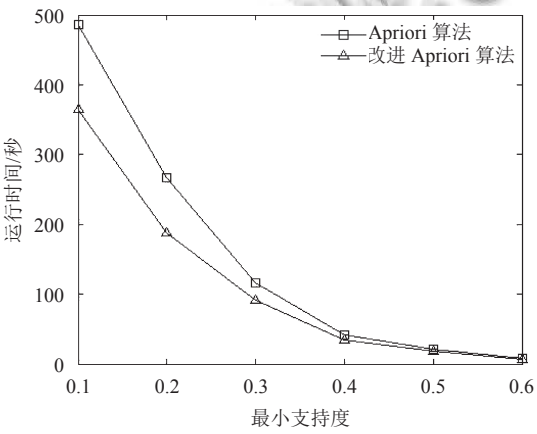


图 2 最小支持度与频繁项集数

从图 2 可以看出, 在相同的最小支持度下, 改进 Apriori 算法在探索频繁项集的过程中比经典 Apriori 算法冗余更少, 相应所占用空间也更小.

## 3 糖尿病预诊系统设计

### 3.1 糖尿病高危因素分析

本研究从 UCL 糖尿病数据集选取其中 8 个相关危险因素[10]进行分析, 分别是: 年龄, 有无高血压或高血脂病史, 身体质量指数 (BMI), 腰臀比 (WHR), 是否吸烟, 是否饮酒, 是否过度饮食和运动量是否达标. 将以上因素分别记为项 A 到项 H, 针对其中若干非布尔类型的数据预处理[11], 年龄大于 45 记为 1, BMI 大于 28 记为 1, 男性 WHR 大于 0.85、女性 WHR 大于 0.8 记为 1. 最后再加入预诊结果项 I, 将所有数据整理为事务数据库, 便于后续工作进行挖掘和分析.

### 3.2 系统架构设计

系统选用时下热门技术栈: RPC 框架 Dubbo, 微服务框架 Spring Boot, 消息中间件 RabbitMQ, 关系型数据库 MySQL, 以及作为缓存的 Redis.

Dubbo 是一款开源 RPC 框架, 它提供了三大核心能力: 面向接口的远程方法调用, 智能容错和负载均衡, 以及服务自动注册和发现. 该框架不仅实现了高性能、高可用性, 而且使用方便, 扩展性极佳[12].

Spring Boot 是 Java 领域知名的微服务框架, 微服务的目的在于化解整体架构服务的复杂性, 以简单快速的方式实现各个服务的部署和变更. 而 Spring Boot 提供了形式多样的库, 支持 JPA、RESTFul、Docker 等技术, 能够让配置、部署和监控变得简单方便[13].

RabbitMQ 基于 Erlang 语言编写, 用于在分布式系统中存储转发异步消息, 将彼此独立的计算机连接起来组成松耦合的系统, RabbitMQ 在易用性扩展性、高可用性等方面表现不俗[14].

MySQL 是一款由瑞典的公司开发并且广泛应用于中小型企业或组织的免费数据库, 基于 Linux 操作系统开发, MySQL 体积小、速度快、总体拥有成本低.

Redis 是一款基于内存的、可持久化的非关系型 Key-Value 存储系统, 它支持多种数据类型, 并支持原子性操作[15]. Redis 与其他 Cache 相比, 拥有更多的数据结构并支持更丰富的数据操作.

基于上述所选技术栈进行系统架构设计, 将系统划分为如图 3 所示的若干层面.

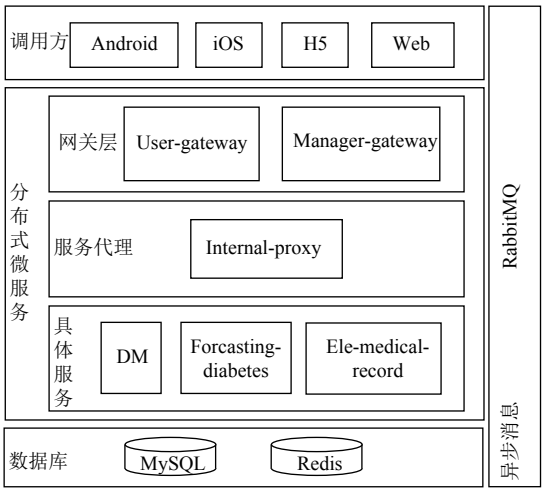


图 3 糖尿病预诊系统架构图



© 中国科学院软件研究所 http://www.c-s-a.org.cn

网关层: 作为统一请求入口, 处理权限认证及负载均衡等, 向外提供 RESTFul API, 并采用令牌桶算法实现 API 的动态限流.

服务代理层: 为提高系统扩展性和可复用性, 抽取公用服务接口, 由代理将请求路由至具体服务.

具体服务层: 具体实现三大核心功能, 包括关联规则挖掘、糖尿病预诊分析、电子病历处理.

预诊分析模块: 基于用户电子病历中的数据, 计算各项高危因素的指标, 并匹配满足置信度的关联规则, 分析糖尿病的患病概率.

电子病历模块: 为用户建立电子病历, 涵盖用户各项相关高危因素信息, 并针对特定项进行量化入库.

规则挖掘模块: 对于管理员设定的支持度和置信度, 基于所建事务数据库, 在后台挖掘满足支持度和置信度阈值的关联规则, 并将关联规则落库.

### 3.3 系统流程设计

系统具体使用流程如图 4 所示, 主要包括三大核心功能的使用流程: 糖尿病自查流程、电子病历录入流程和关联规则挖掘流程.

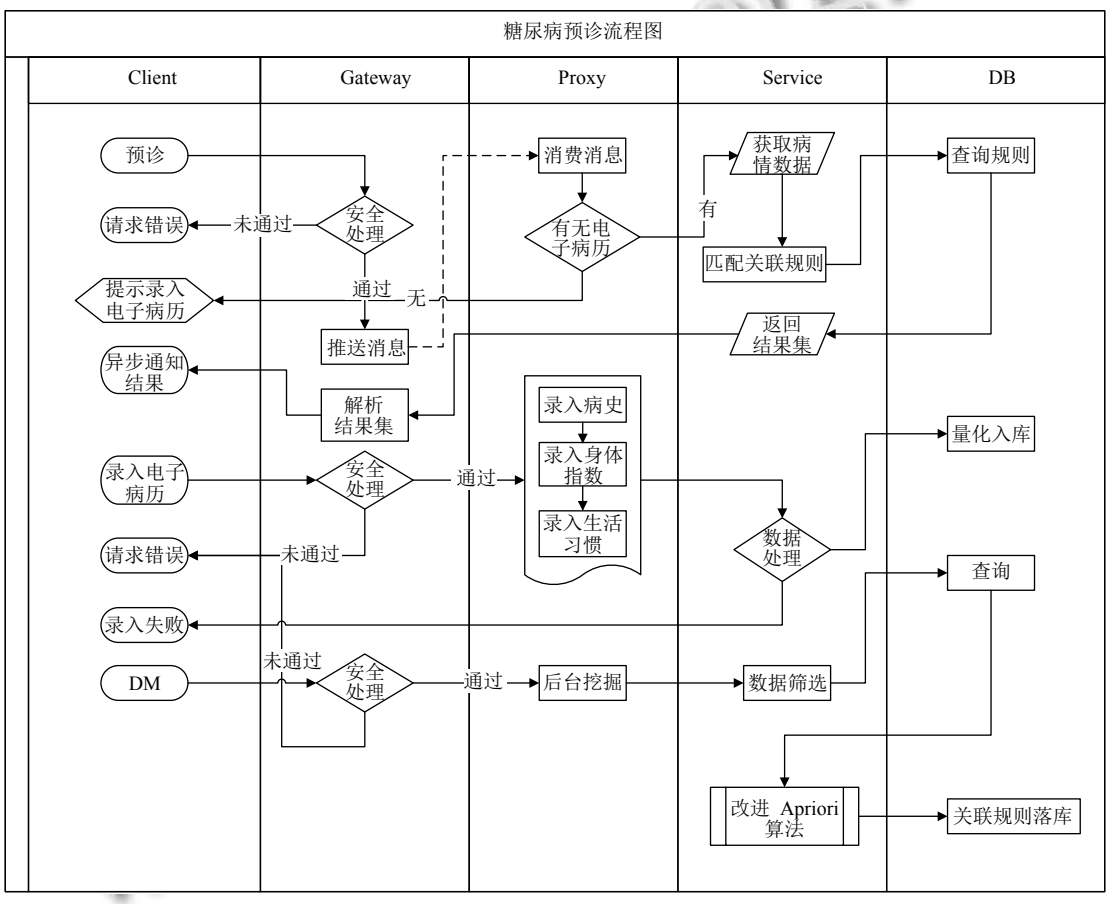


图 4 糖尿病预诊系统流程图

① 当用户提交自查请求, 网关层会对请求做权限认证、接口限流令牌校验、安全处理等. 随后会生产一条异步消息推送至 RabbitMQ, 由代理层消费消息并路由至具体服务, 实现规则匹配和异步结果返回.

② 当用户电子病历记录为空时, 医护人员录入用户各项相关数据. 数据提交后, 代理层调用具体服务处理数据, 生成电子病历并存储入库.

③ 当管理员提交关联规则挖掘请求时, 网关层对权限进行校验, 随后代理层路由至具体服务, 使用改进 Apriori 算法在后台对数据集进行筛选和挖掘. 在关联规则落库后, 以站内信和其他特定方式通知管理员关联规则的挖掘结果.

### 3.4 系统核心配置和 UI 设计

HIS 子系统服务提供者配置文件 provide.xml 核心



© 中国科学院软件研究所 http://www.c-s-a.org.cn

内容如下, 其中包括暴露的服务接口及注册地址:

```
<dubbo:application logger="slf4j"
  name="${dubbo.application-name}"/>
<dubbo:protocol name="dubbo" charset="UTF-8"
  port="${dubbo.port}" serialization="java"/>
<dubbo:registry username="root"
  address="${dubbo.registry-address}"/>
<!-- 声明需要暴露的服务接口 -->
<dubbo:service version="${dubbo.service-version}"
  ref="hisRemoteService"
  interface="com.heren.ois.rpc.
  hisInterface.service.HisRemoteService"/>
```

服务消费者配置文件 consumer.xml 核心内容如下, 同样包含其注册地址等信息:

```
<!-- 服务接口 -->
<dubbo:reference id="hisRemoteService"
  version="${dubbo.service-version}"
  interface="com.heren.ois.rpc.
  hisInterface.service.HisRemoteService"/>
```

接口 HisRemoteService 定义如下, 其中包含根据病历号或姓名性别查找患者、根据患者查找电子病历和诊断结果等方法:

```
public interface HisRemoteService {
  Patient findPatientBy(String MedicalNo);
  List<Patient> findListByNameAndGender(
    String name, Integer gender
  );
  Map<String, List<MedicalRecord>>
  findMedicalRecordListBy(Patient patient);
  Map<String, List<Diagnosis>>
    findDiagnosisListBy(Patient patient);
}
```

系统用雷达图展示各指标危险临界点与自身指标情况; 用饼图展示糖尿病患者某指标异常的比例; 最后辅以诊断分析和医嘱建议等文字, 效果如图 5 所示.

### 3.5 预诊结果分析

截取部分电子病历的核心数据, 如图 6 所示, 其中年龄、病史、BMI、WHR、吸烟、饮酒、过度饮食、运动量达标为八项相关因素.

将八项高危因素按照 3.1 节规则量化, 得到如图 7 所示的项 A 到项 H. 运算得出结果即项 I, 可以发现与真实诊断结果无异.

图 5 糖尿病预诊分析界面

| 性别 | 年龄 | 高血压高血脂病史 | bmi | whr | 吸烟 | 饮酒 | 过度饮食 | 运动量达标 | 患糖尿病 |
|---|---|---|---|---|---|---|---|---|---|
| 男 | 32 | 无 | 25.1 | 0.74 | 是 | 否 | 否 | 是 | 否 |
| 女 | 35 | 有 | 28.2 | 0.82 | 否 | 否 | 是 | 否 | 是 |
| 男 | 46 | 无 | 32.2 | 0.91 | 是 | 是 | 是 | 否 | 是 |
| 男 | 29 | 有 | 24.6 | 0.81 | 否 | 是 | 否 | 是 | 否 |
| 女 | 24 | 无 | 23.5 | 0.71 | 否 | 否 | 是 | 是 | 否 |

图 6 电子病历部分数据

| A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

图 7 量化后的数据

## 4 结束语

针对 Apriori 经典算法存在的缺陷, 本研究进行了改进, 并应用于糖尿病与其高危因素间的关联规则挖掘. 通过实验对算法进行对比, 结果表明改进 Apriori 算法性能得到了大幅度提高. 基于以上工作, 本研究设计了一款糖尿病预诊分析系统, 随着挖掘样本数量的逐步增加其准确率也逐步提升. 此系统为用户自诊和医护人员辅助诊断提供了更加便捷的方式.

### 参考文献


1 Yıldırım EG, Karahoca A, Uçar T. Dosage planning for diabetes patients using data mining methods. Procedia Computer Science, 2011, 3: 1374–1380. [doi: 10.1016/j.procs.2011.01.018]

2 李武成, 王官权, 金科. 2 型糖尿病并发高血压的危险因素分析. 实用医学杂志, 2010, 26(17): 3180–3181. [doi: 10.3969/j.issn.1006-5725.2010.17.045]

3 董宁. 基于数据挖掘的 Apriori 算法研究与改进. 自动化与仪器仪表, 2016, (9): 232–234.

4 张伟科. 一种改进的 AprioriTid 算法. 沈阳工业大学学报, 2016, 38(3): 314–318.




© 中国科学院软件研究所 http://www.c-s-a.org.cn

5 苗苗苗, 王玉英. 基于矩阵压缩的 Apriori 算法改进的研究. 计算机工程与应用, 2013, 49(1): 159–162. [doi: 10.3778/j.issn.1002-8331.1107-0579]
6 王蒙, 邹书蓉, 方睿. 一种基于矩阵的 Apriori 改进算法. 信息技术, 2018, (3): 150–154, 158.
7 周发超, 王志坚, 叶枫, 等. 关联规则挖掘算法 Apriori 的研究改进. 计算机科学与探索, 2015, 9(9): 1075–1083.
8 李超, 余昭平. 基于矩阵的 Apriori 算法改进. 计算机工程, 2006, 32(23): 68–69. [doi: 10.3969/j.issn.1000-3428.2006.23.024]
9 杨志刚, 何月顺. 基于压缩事务矩阵相乘的 Apriori 改进算法. 中国新技术新产品, 2010, (6): 57–58. [doi: 10.3969/j.issn.1673-9957.2010.06.045]
10 冯玉欣, 赵希兵. 2 型糖尿病家系发病特征与危险因素调查. 糖尿病新世界, 2017, 20(12): 19–20.
11 韦哲, 叶广健. 一种 Apriori 改进算法在 2 型糖尿病危险因素分析中的应用. 电子测试, 2015, (17): 63–65, 84. [doi: 10.3969/j.issn.1000-8519.2015.17.026]
12 陈晓栋. 基于 Dubbo 分布式框架的信用卡无卡大额分期系统设计. 信息与电脑, 2017, (7): 132–135. [doi: 10.3969/j.issn.1003-9767.2017.07.051]
13 温晓丽, 苏浩伟, 陈欢, 等. 基于 SpringBoot 微服务架构的城市一卡通手机充值支撑系统研究. 电子产品世界, 2017, 24(10): 59–62.
14 鱼朝伟, 詹舒波. 基于 RabbitMQ 的异步全双工消息总线的实现. 软件, 2016, 37(2): 139–146. [doi: 10.3969/j.issn.1003-6970.2016.02.032]
15 曾超宇, 李金香. Redis 在高速缓存系统中的应用. 微型机与应用, 2013, 32(12): 11–13. [doi: 10.3969/j.issn.1674-7720.2013.12.004]

www.c-s-a.org.cn
www.c-s-a.org.cn



© 中国科学院软件研究所 http://www.c-s-a.org.cn