2017 年春季学期数学建模课程论文

题目： ATM 交易状态特征分析和异常检测

作者信息：

编号	学号	姓名	专业年级	分工
1	16060012014	黄佳城	16 海洋渔业科学与技术	论文写作
2	16120014008	韩天宇	16 经济学	建模
3	16120014032	奚东宇	16 经济学	编程

评分指标及分值分布说明：

指标	摘要	写作及整体	模型	解法及结果	分析检验	总分
分布	10 分	20 分	30 分	30 分	10 分	100 分
得分						

ATM 交易状态特征分析与异常检验

摘要

ATM 机为我们的生活提供了十分大的便利。同样的，ATM 应用系统的异常检验也十分重要。在论文中，我们对 ATM 交易状态特征分析与异常检验做出了深入探讨。

第一问，为找出影响业务量的因素，先绘制每天业务量与时间的散点图，可以得出：每天的业务量与时间呈现双峰高斯分布关系。而工作日和非工作日在数值上取得双峰值的时间又有所不同。同样的，可以得到业务量与交易响应时间负相关，而业务量与交易成功率也呈现双峰高斯分布的关系。

为了更好地研究不同时间段内交易状态区别，我们定义两种交易状态：分钟状态和自然日状态。分钟状态是指我们将每一分钟的交易情况均视为一种交易状态。之后采取主成分分析(PCA)对数据进行特征值的提取。但是不满足 PCA 算法中对于数据服从高斯分布的假设。故改用非线性 PCA 算法，即核主成分分析（KPCA）来进行特征值的提取。

为了第二问的的故障识别服务，设立自然日状态，并分为两种：普通自然日状态和标准自然日状态。普通自然日状态是指数据中给出的 91 天中每一天的交易状态；标准自然日状态则指理论上没有出现故障的一天所应该具有的普通自然日交易状态。将每一天总的交易情况视为一种交易状态。在这里，我们采用每一天业务量、成功交易率、交易响应时间的均值和方差所组成的 3*2 阶的矩阵作为普通自然日状态的特征值。

第二问，我们对每一分钟的状态进行检测。运用分类器，结合第一问中得到的标准自然日交易特征等标准数据进行数据的预处理，分类器则反馈出状态是否异常，或者异常的种类是哪一种。首先从第一问所提取的分钟状态特征值中找寻规律，将已有的数据分为正常和异常两组，这是非监督学习的部分，也就是聚类部分。对于聚类，选取 K-means 算法，考虑到粒子群算法（PSO）是一种有效的全局寻优算法，我们采用基于粒子群算法的 K-means 算法来对分钟交易状态的特征值进行聚类。在得到聚类结果后，我们就对数据打上了标签。之后我们进行监督学习的部分，把已经打上标签的数据分成训练集和检验集，运用 REF 神经网络进行学习。之后在聚类过程中，利用我们得到的普通自然日状态特征矩阵来手工判断故障的类型，做到对于故障种类的识别。

第三问，我们希望加上的数据是 91 天内的 ATM 应用维修记录，这类记录应该包括：维修期间所用的时间和故障的类型。加上这组数据之后，对于第（1）问，我们可以得到更为精准的标准自然日状态的特征值。我们还可以提高第（2）问中对于故障类型判断的准确性和对于“应该持续故障几分钟才报警”这个问题的判断。

关键词：ATM 应用系统、故障检测、特征参数、KPCA、K-means、PSO、REF 神经网络

目录

一、 问题重述

某商业银行的 ATM 应用系统包括前端和后端两个部分。前端是部署在银行营业部和各自助服务点的 ATM 机（系统），后端是总行数据中心的处理系统。前端的主要功能是和客户直接交互，采集客户请求信息，然后通过网络传输到后端，再进行数据和账务处理。持卡人从前端设备提交查询或转账或取现等业务请求，到后台处理完毕，并将处理结果返回到前端，通知持卡人业务处理最终状态，我们称这样完整的一个流程为一笔交易。

商业银行总行数据中心监控系统为了实时掌握全行的业务状态，每分钟对各分行的交易信息进行汇总统计。汇总信息包括业务量、交易成功率、交易响应时间三个指标，各指标解释如下：

1、业务量：每分钟总共发生的交易总笔数；

2、交易成功率：每分钟交易成功笔数和业务量的比率；

3、交易响应时间：一分钟内每笔交易在后端处理的平均耗时(单位：毫秒)。

交易数据分布存在以下特征：工作日和非工作日的交易量存在差别；一天内，交易量也存在业务低谷时间段和正常业务时间段。当无交易发生时，交易成功率和交易响应时间指标为空。

商业银行总行数据中心监控系统通过对每家分行的汇总统计信息做数据分析，来 捕捉整个前端和后端整体应用系统运行情况以及时发现异常或故障。常见的故障场景包括但不限于如下情形：

1、分行侧网络传输节点故障，前端交易无法上送请求，导致业务量陡降；

2、分行侧参数数据变更或者配置错误，数据中心后端处理失败率增加，影响交易成功率指标；

3、数据中心后端处理系统异常（如操作系统 CPU 负荷过大）引起交易处理 缓慢，影响交易响应时间指标；

4、数据中心后端处理系统应用进程异常，导致交易失败或响应缓慢。

附件是某商业银行 ATM 应用系统某分行的交易统计数据。

你的任务是：

（1） 选择、提取和分析 ATM 交易状态的特征参数；

（2） 设计一套交易状态异常检测方案，在对该交易系统的应用可用性异常情况下能做到及时报警，同时尽量减少虚警误报；

（3） 设想可增加采集的数据。基于扩展数据，你能如何提升任务（1）（2）中你达到的目标？

二、 问题的分析

第一问，我们先将附件中每一天的数据都按照天数分开，考察每天业务量与时间的关系。通过绘制两者的散点图可以得出：每天的业务量与时间呈现双峰高斯分布关系。而工作日和非工作日在数值上取得双峰值的时间又有所不同。同样的可以得到业务量与交易响应时间负相关，而业务量与交易成功率也呈现双峰高斯分布的关系。所以，数据中的业务量、交易成功率、交易响应时间三者之间存在一定的关联，而且三者在工作日和非工作日时具有一定差别。

在这里，我们定义两种交易状态：分钟状态（Minute State）和自然日状态（Day State）。

分钟状态（Minute State）是指我们将每一分钟的交易情况均视为一种交易状态。并且分钟状态的特征值提取算法是第一问任务所要求的答案。选择、提取分钟状态的特征参数就是用一组特定的数据来描述这一分钟的状态。考虑到业务量、交易成功率、交易响应时间三者存在相关性，但之间的关联关系却又不明确；三种数据的数量级和量纲也不尽相同。因而我们采取主成分分析（Principal Component Analysis，PCA）对数据进行特征值的提取。虽然附件中的数据满足 KMO（Kaiser-Meyer-Olkin）检验与 BartlettP 检验[检验要求 KMO 值大于 0.5、BartlerttP 值小于 0.01]，但是不满足 PCA 算法中对于数据服从高斯分布的假设。故改用非线性 PCA 算法，即核主成分分析（Kernel Principal Component Analysis， KPCA）来进行特征值的提取。在得到成分矩阵之后，我们将对成分矩阵进行分析，得到分钟状态的特征值的特点。

自然日状态（Day State）则是将每一天总的交易情况视为一种交易状态，设立这个状态的目的是为了第二问的的故障识别服务。自然日状态分为两种：普通自然日状态和标准自然日状态。普通自然日状态是指数据中给出的 91 天中每一天的交易状态。在这里，我们采用每一天业务量、成功交易率、交易响应时间的均值和方差所组成的 3*2 阶的矩阵作为普通自然日状态的特征值。标准自然日状态则指理论上没有出现故障的一天所应该具有的普通自然日交易状态，标准自然日状态又有工作日和非工作日之分，我们取剔除了异常数据的普通自然日状态的特征值的均值作为标准自然日状态的特征值。

第二问，问题要求设计一套交易状态的异常检验方案。我们将对每一分钟的状态进行检测。后台系统反馈的每一分钟的交易量、交易成功率、交易响应时间，借助第一问中得到的标准自然日交易特征等标准数据进行数据的预处理，并输入分类器，分类器则反馈出状态是否异常，或者是异常的种类是哪一种。

分类器的设计事实上是非监督学习和监督学习的结合，即先进行聚类，再设计分类器。我们需要先从第一问所提取的分钟状态特征值中找寻规律，将已有的数据分为正常和异常两组，这是非监督学习的部分，也就是聚类部分。对于聚类，我们决定选取 K-means 算法：该算法原理简单，容易解释，运行速度快；但是容易陷入局部最小值解的结果。考虑到粒子群算法（Particle Swarm Optimization，PSO）是一种有效的全局寻优算法，我们采用基于粒子群算法的 K-means 算法来对分钟交易状态的特征值进行聚类。在得到聚类结果后，我们就对数据打上了标签。之后我们进行监督学习的部分，把已经打上标签的数据分成训练集和检验集，运用 REF 神经网络进行学习。之后再进行进一步扩展，在聚类过程中，我们将加大聚类数，以便分别出更多的类别，再借助自然日状态的特征值进行分析，做到对于故障种类的识别。当然，为了减少虚警误报的情况，我们在检测的时候设定连续故障 4 分钟后才报警。

第三问，任务是增加数据以完善（1）、（2）题的结果。对于第（1）问来说，数据的不足导致结果会有比较大变化的应该是对于标准自然日状态的特征值；对于第（2）问来说，数据是否完美则会是影响聚类结果的关键。

所以，我们希望加上的数据是 91 天内的 ATM 应用维修记录，这类记录应该包括：维修期间所用的时间和故障的类型。加上这组数据之后，对于第（1）问，我们可以得到更为精准的标准自然日状态的特征值。我们还可以少去第（2）问中非监督学习的部分。因为不管聚类方法选择得再好，终究与现实情况有一定的差别。如果有维修记录，我们就可以将非监督学习与监督学习的过程转化为完全的监督学习的过程。这会形成更好的分类器。

三、 模型假设

1. 方便表示，我们约定数据中的 1 月 23 号设为第 1 日，1 月 24 号设为第 2 日， ……类推，则有 4 月 23 号为第 91 日。
2. 银行的交易量数据不受经营情况的影响。
3. 附件中的数据真实可靠。
4. ATM 机满足日常的规定，例如一笔取款不能超过 3000 元，即不会因为大额数值增加反应时间。

四、 符号与约定

序号	符号	含义
1	v	交易量
2	r	成功率
3	t	响应时间
4	$\text{mean}(v_i^{min})$	第 i 天交易量的均值
5	$\text{mean}(r_i^{min})$	第 i 天成功率的均值
6	$\text{mean}(v_i^{min})$	第 i 天响应时间的均值
7	$\text{var}(v_i^{min})$	第 i 天交易量的方差
8	$\text{var}(r_i^{min})$	第 i 天成功率的方差
9	$\text{var}(t_i^{min})$	第 i 天响应时间的方差

10	X_{i}	第ｉ天自然日交易状态的特征值矩阵
11	$X_{\mathrm{i}}^{\mathrm{ND}}$	第ｉ天普通自然日交易状态的特征值矩阵
12	X^{SWD}	标准工作自然日交易状态的特征值矩阵
13	X^{SOD}	标准非工作自然日交易状态的特征值矩阵
14	C	协方差矩阵
15	U	特征向量矩阵
16	$\phi(x_j)\phi(x_k)$	核函数
17	K	核矩阵
18	$\alpha_{\mathrm{k}}^{\mathrm{i}}$	计算非线性主成成分时用到的特征向量
19	Z_{j}（Ⅰ）	聚类中心
20	$D(\mathrm{x_i},\ Z_{\mathrm{j}}(\mathrm{Ⅰ}))$	每个数据与聚类中心的距离
21	m	粒子群粒子总数
22	$\mathrm{x}_i=(x_i^1,x_i^2,\cdots,x_i^D)$	粒子位置
23	P_{i}=（$\mathrm{p_i^1}$，$\mathrm{p_i^2}$，…，$\mathrm{p_i^D}$）	个体粒子经过的最优位置
24	P_{g}=（$\mathrm{p_g^1}$，$\mathrm{p_g^2}$，…，$\mathrm{p_g^D}$）	整体粒子群所的最优位置
25	S_{i}=（$\mathrm{s_i^1}$，$\mathrm{s_i^2}$，…，$\mathrm{s_i^D}$）	第ｉ 粒子的速度
26	ω	惯性因子
27	$\mathrm{c_1}$、$\mathrm{c_2}$	加速常数
28	α	约束因子
29	N_{pi}	粒子群连续状态无法得到改善的累积次数
30	$\mathrm{thre_p}$	粒子群连续状态无法得到改善的累积次数的阈值
31	N_{g}	粒子群最优解无法得到改善的累积次数
32	$\mathrm{thre_g}$	粒子群最优解无法得到改善的累积次数的阈值

五、 模型的建立和求解

5.1 选择、提取和分析 ATM 交易状态的特征参数

5.1.1 绘制和分析与数据相关的图表与统计学特征的计算

首先我们先对附件中给出的 91 天的数据进行分析。我们现将数据分为以天为单位，然后利用 Matlab 画出时间与交易量的折线图：我们可以发现，交易量在每一天都呈现双峰高斯分布的关系，而且工作日与非工作日存在着交易量达到峰值在不同的时间段的区别，一般是在工作日中，交易量达到峰值的时间总是要比非工作日来得早。

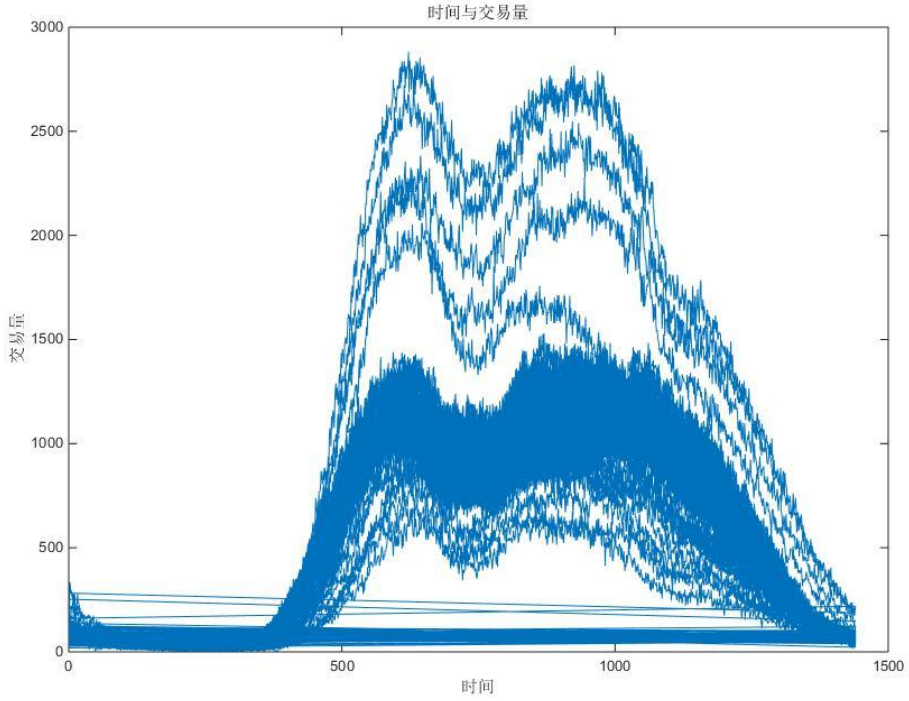

图 1 时间与交易量的折线图

接着，我们同样利用 Matlab 绘制了响应时间时间与交易量的关系，我们可以发现相应时间与交易量基本成负相关，也即交易量越少，则 ATM 应用系统后端的反应时间越长。

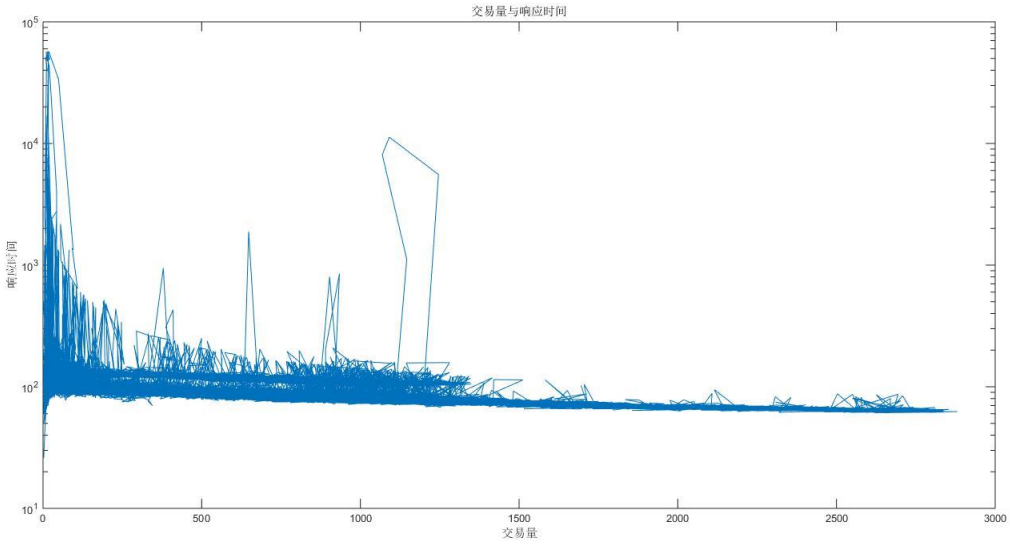

图 2 响应时间与交易量的折线图

然后，我们考虑成功率与时间的关系，利用 Matlab 绘制成功率与时间的散点图，可以发现两者呈现一种单峰分布状态。其中，成功率一般在 95%左右浮动。在一天中的午夜到早晨之间的成功率则是处于较低的水平，且波动较大。

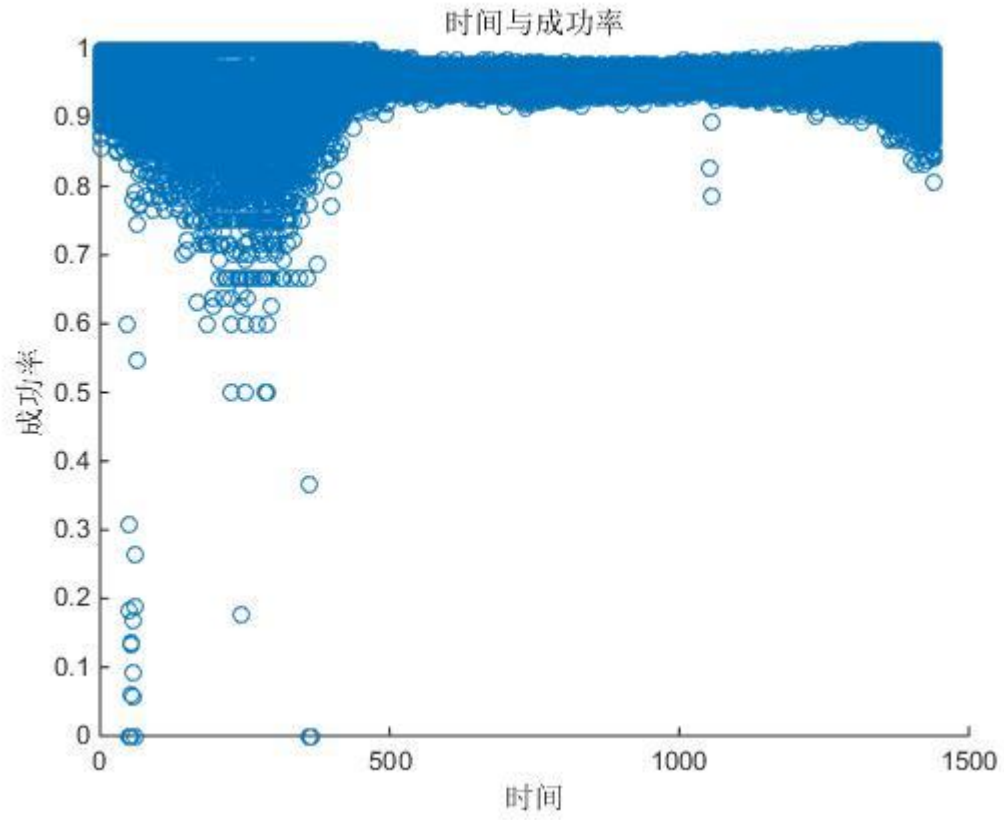

图 3 成功率与时间的散点图

通过上面三张图片的分析，我们可以得知交易量、成功率、响应时间之间存在着关联性，但是这种相关性并不明确，可以做到对其中两者之间的函数拟合，但是效果必然不佳。

最后我们绘制了总成交量与时间的折线图。我们可以发现在前 4 天的数据不断上涨，在第 5 天数据突然下落，然后接下来的七天里缓慢回升值正常水平。由此我们可以推断附件中所提供的数据应该是来自 2017 年 1 月 23 日至 4 月 23 号。之前造成的数据波动收到的是传统节日春节的影响。

图 4 总交易量和日期的折线图

在考察完交易量、成功率、响应时间之间的关系之后，我们接着考虑数据中所具有的统计学特征。我们将数据分为以天为单位，考察每一天中的交易量、成功率、响应时间的均值与方差，得到相对应的数据（附表 1）。

通过对数据的观察可得：交易量、成功率、响应时间关于分钟的均值与方差均呈现集群分布。在数值上有以下特点：每天交易量关于分钟的平均值基本在 490-600 笔之间波动；每天交易量关于分钟的方差则总是普遍偏大；每天成功率关于分钟的平均值基本保持在 0.95 以上，而其方差一般保持在小于 0.001 的水平；对于响应时间来说，由于理论上来说应该波动不大，所以它关于分钟的均值一直处在 100 毫秒左右，而方差较为稳定，一般小于 500。

5.1.2 交易状态的定义

通过对交易量、成功率、响应时间的关联分析与一些统计学特征的计算，我们意识到每一分钟表现出来的特征对于异常检验非常重要，一些一天之中表现出来的特征同样也不容忽视。

所以我们定义了两种交易状态，一种是分钟状态（Minute State），另一种是自然日状态（Day State）。分钟状态是把每一分钟的交易情况当做一种交易状态；而自然日状态是指把每一天的交易情况作为一种交易状态。其中，自然日状态还分为两种——普通自然日状态和标准自然日状态。普通自然日状态是指数据中每一天的真实

自然日状态；标准自然日状态是指理论上没有故障的一天中所应具有的自然日状态。标准自然交易日状态又分为工作日标准自然日状态和非工作日标准自然日状态。

5.1.3 自然日交易状态的特征值提取

1. 自然日交易状态特征值选取

对于自然日状态的特征值的，我们采用统计学特征——即每一天业务量、成功交易率、交易响应时间的关于时间的均值和方差所组成的 3*2 阶的矩阵作为普通自然日状态的特征值，于是，第 i 天自然日交易状态的特征值矩阵可以表示为：

$$X_i = \begin{bmatrix} mean(v_i^{min}) & mean(r_i^{min}) & mean(v_i^{min}) \\ var(v_i^{min}) & var(r_i^{min}) & var(t_i^{min}) \end{bmatrix}$$

选择每一天业务量、成功交易率、交易响应时间的的均值和方差所组成的 3*2 阶的矩阵作为普通自然日状态的特征值的原因是这些统计特征数据反应出每一天的交易状态。以每天交易相应时间关于时间的方差为例：因为 ATM 应用系统后端服务器一般运行较为稳定的 Linux 系统，所以反应时间不会有太大的波动。如果 ATM 应用系统后端出现故障，则会使反应时间大大加大，这造成了反应时间方差的扰动。选用数据的方差可以准确捕捉这样的异常。对于成功率来说也是同样的道理。

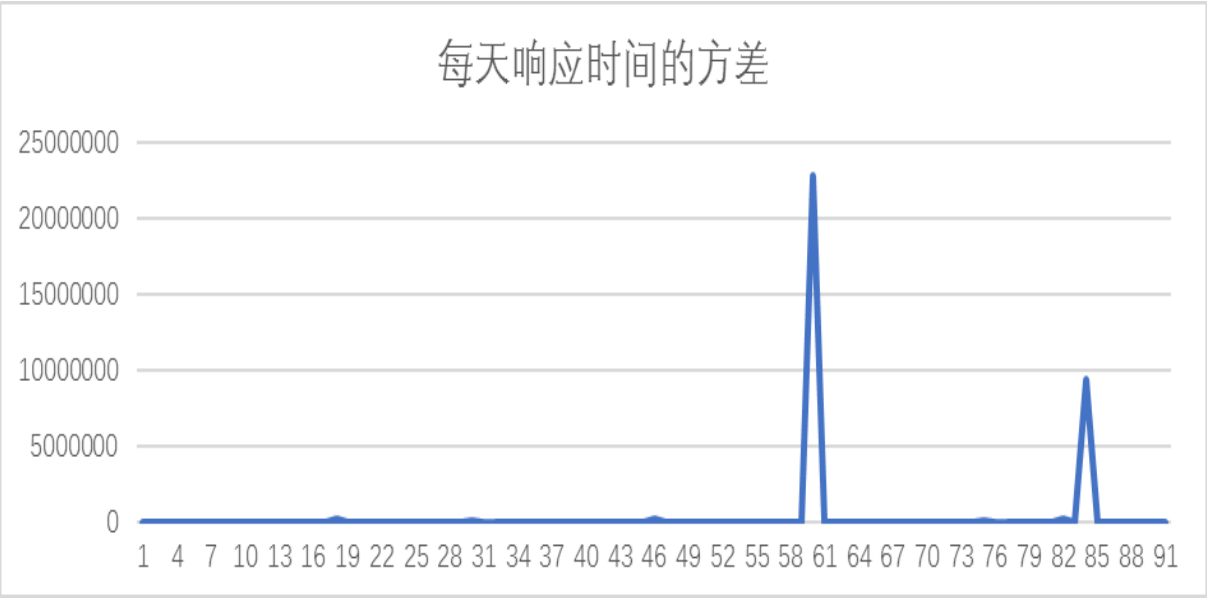

图 5 每天响应时间的方差

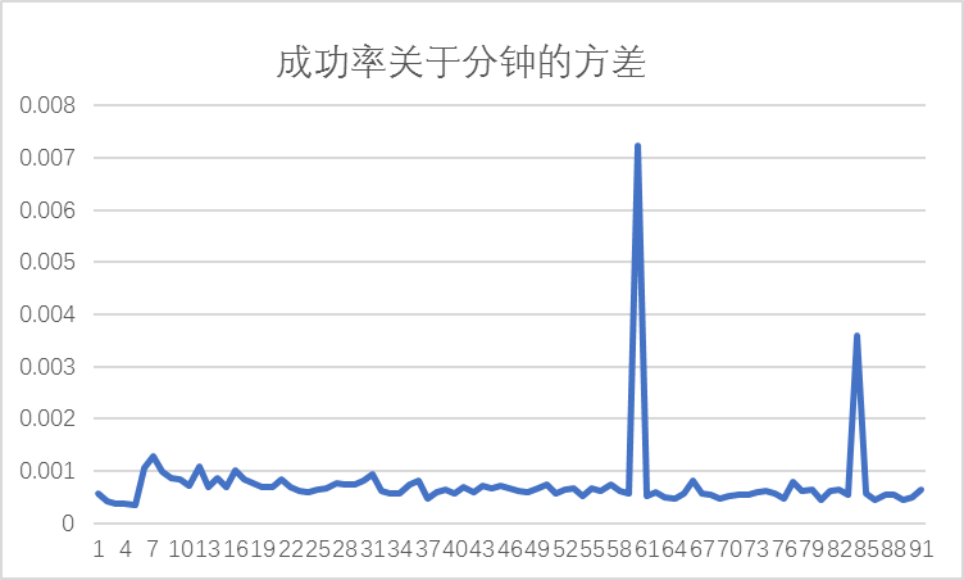

图 6 每天成功率的方差

下面，我们将细化自然日交易状态的特征值。

2. 普通自然日交易状态的特征值矩阵

有上面讨论出的结果，第 i 天普通自然日交易状态的特征值矩阵可以表示为

$$X_i^{ND}=\begin{bmatrix} mean(v_i^{min}) & mean(r_i^{min}) & mean(v_i^{min}) \\ var(v_i^{min}) & var(r_i^{min}) & var(t_i^{min}) \end{bmatrix}$$

3. 标准自然日交易状态的特征值矩阵

在求出标准自然日交易状态的时候，最重要的是如何找出一个理论上没有出现故障的时间。这在在现在的分析手段中是不能，所以我们采用排除了异常值的业务量、成功交易率、交易响应时间的关于时间的均值和方差的数据，再取平均值来避免误差。

在对异常值的排除是，我们设置了对部分数据设置了阈值：

$var(t_i^{min}) \leqslant 500$、 $var(r_i^{min}) \leqslant 0.001$

对于交易量的差别，我们将所有日期分成了工作日和非工作日，在进行去均值。可以得到以下结论：

标准工作自然日交易状态的特征值矩阵可以表示为：

$$X^{SWD}=\begin{bmatrix} 631 & 0.9588 & 101 \\ 264{,}102 & 0.0007 & 222 \end{bmatrix}$$

标准非工作自然日交易状态的特征值矩阵可以表示为：

$$X^{\mathrm{SOD}}=\begin{bmatrix} 516 & 0.9579 & 99 \\ 177{,}169 & 0.000794848 & 232 \end{bmatrix}$$

5.1.4 分钟交易状态的特征值提取

1. 分钟交易状态特征值的选取

从 5.1.1 中对于数据的分析我们可以知道，在以分钟为维度观察下的数据，考虑到业务量、交易成功率、交易响应时间三者存在相关性，但之间的关联关系却又不明确；三种数据的数量级和量纲也不尽相同。因而我们采取主成分分析（Principal Component Analysis，PCA）对数据进行分钟交易状态的特征值提取。虽然附件中的数据满足 KMO（Kaiser-Meyer-Olkin）检验与 BartlettP 检验[检验要求 KMO 值大于 0.5、BartlerttP 值小于 0.01]，但是不满足 PCA 算法中对于数据服从高斯分布的假设。故改用非线性 PCA 算法，即核主成分分析（Kernel Principal Component Analysis，KPCA）来进行特征值的提取。

2. 主成分分析

主成分分析(PCA）是一种数学降维方法，他要做的就是将原先具有一定相关性的数据从新提取成一组新的互相无关的综合变量。

假设有一组数据，$X=\left[x_1,\ x_2\cdots,\ x_{\mathrm{n}}\right]$ ，是一组 $\mathrm{d}\times\mathrm{n}$ 阶的数据矩阵，$\mathrm{d}、\mathrm{n}\in N^*$ 。数据满足中心化，即 $\sum_{\mathrm{i}=1}^{\mathrm{n}} x_{\mathrm{i}}=0$ 。PCA 算法的目的就是寻找一组 p 维向量 $X'=\left[x'_1,\ x'_2\cdots,\ x'_{\mathrm{p}}\right]$，$\mathrm{p}\leqslant\mathrm{d}$ ，用来取代原有的数矩阵，使得它能代表尽可能多的原来数据矩阵的信息。

定义 X 的协方差矩阵 C ：

$$C=\frac{1}{\mathrm{n}}x_{\mathrm{i}}x_{\mathrm{i}}^{T}=\frac{1}{\mathrm{n}}XX^{T}\ .$$

对其做特征值分解可以得到：

$$CU=U\Lambda\Rightarrow C=U\Lambda U^{T}=\sum_{i=1}^{n}\lambda_i\mathrm{u}_i\mathrm{u}_i^{T}$$

其中，$U=\left[u_1,u_2,\ \cdots,\ u_{\mathrm{d}}\right]$ ，$\Lambda=\mathrm{diag}（\lambda_1,\ \lambda_2,\ \cdots,\ \lambda_{\mathrm{d}}）$ 。

当我们做降维操作时，利用前 p 个特征向量 $U_{\mathrm{p}}=\left[u_1,u_2,\ \cdots,\ u_{\mathrm{p}}\right]$，求出 d 维数据 X 在 p 维主成分方向上的投影 $X'=\left[x'_1,\ x'_2\cdots,\ x'_p\right]$。其中，$x'_{\mathrm{i}}=\mathrm{U}_{\mathrm{k}}^{T}x_i$ 。这个投影就是 PCA 的结果。

2. 核主成分分析

主成分分析中要求数据要线性可分，即服从高斯分布。但是通过我们对数据的分析中很容易看出某些数据是不服从高斯分布的，比如说交易量这一项数据。所以我们使用非线性 PCA 算法，即核主成分分析。

同样，我们假设有一组数据，$X=[x_1, x_2\cdots, x_n]$ ，是一组 $d\times n$ 阶的数据矩阵，$d、n\in N^*$ 。它在 d 维空间上是不满足线性可分的。

现在我们有一个非线性函数 ϕ ，将数据矩阵 X 的值映射到一个高维空间 F ，即 $\phi: R^n \to F, x\mapsto \phi(x)$ ，使得数据有线性不可分转化为线性可分。接着我们要在高维空间 F 使用 PCA 算法。

仍旧假设现在 F 中的数据矩阵是中心化的，即 $\sum_{i=1}^{n}\phi(x_i)=0$ 。则其协方差矩阵 C 就转换为:

$$C=\frac{1}{n}\phi(x_i)\phi(x_i)^T=\frac{1}{n}\phi(X)\ \phi(X)^T.$$

我们需要求出协方差矩阵的特征值 $\lambda\geqslant 0$ 以及对应的特征向量 $U\in F\setminus\{0\}$,使其满足特征值分解要求的等式：$CU=\lambda U$ 。由于大于等于零的特征向量都落在数据的张集上，也即满足 $U\in \mathrm{span}\{\phi(x_1), \phi(x_2)\cdots, \phi(x_n)\}$ ，所以一定存在一组系数 α_k（k=1，…，n），使得：

$$U=\sum_{k=1}^{n}\alpha_k\phi(x_k)$$

从这里可推出

$$\lambda\sum_{k=1}^{n}\alpha_i(\phi(x_i)\phi(x_k))=\frac{1}{n}\sum_{i=1}^{n}\alpha_i(\phi(x_i)\sum_{j=1}^{n}\phi(x_j))\ (\phi(x_j)\phi(x_k)), i=1, \cdots, n\ .$$

此时我们定义一个 $n\times n$ 阶的核矩阵 K ，$K_{j,k}=(\phi(x_j)\phi(x_k))=k(x_j, x_k)$。则上式可以写为

$$K\alpha=n\lambda\alpha$$

因此，我们在高维空间进行的 PCA 算法其实完全依赖于 $k(x_j, x_k)$,我们称其为核函数。

常用的核函数有

（1）线性核函数　　k（x, x_k）=x·x_k

（2）p 阶多项式核函数　k（x, x_k）=[($x \cdot x_k$)+1]^p

（3）高斯径向基函数（RBF）函数　　k（x, x_k）= exp($-\frac{\|x - x_k\|}{\sigma^2}$)

（4）多层感知器（MLP）核函数　　k（x, x_k）=tanh[v（$x \cdot x_k$）+c]

对于原始数据矩阵 X ，在高维空间 F 的映射点 ϕ(X）,则我们最后得到的非线性主成分可以表示为：

$$X'=（U^n \cdot \phi(X）)=\sum_{k=1}^{n} \alpha_k^i(\phi(x_j)\phi(x_k))$$ 。

3. 分钟交易状态的特征值矩阵

我们运用 Matlab 编程（代码附在附录中）实现 KPCA 算法。本次核函数选择高斯径向基函数（RBF）函数。对被分成以天为单位的数据进行降维处理。整个算法的主要流程为：

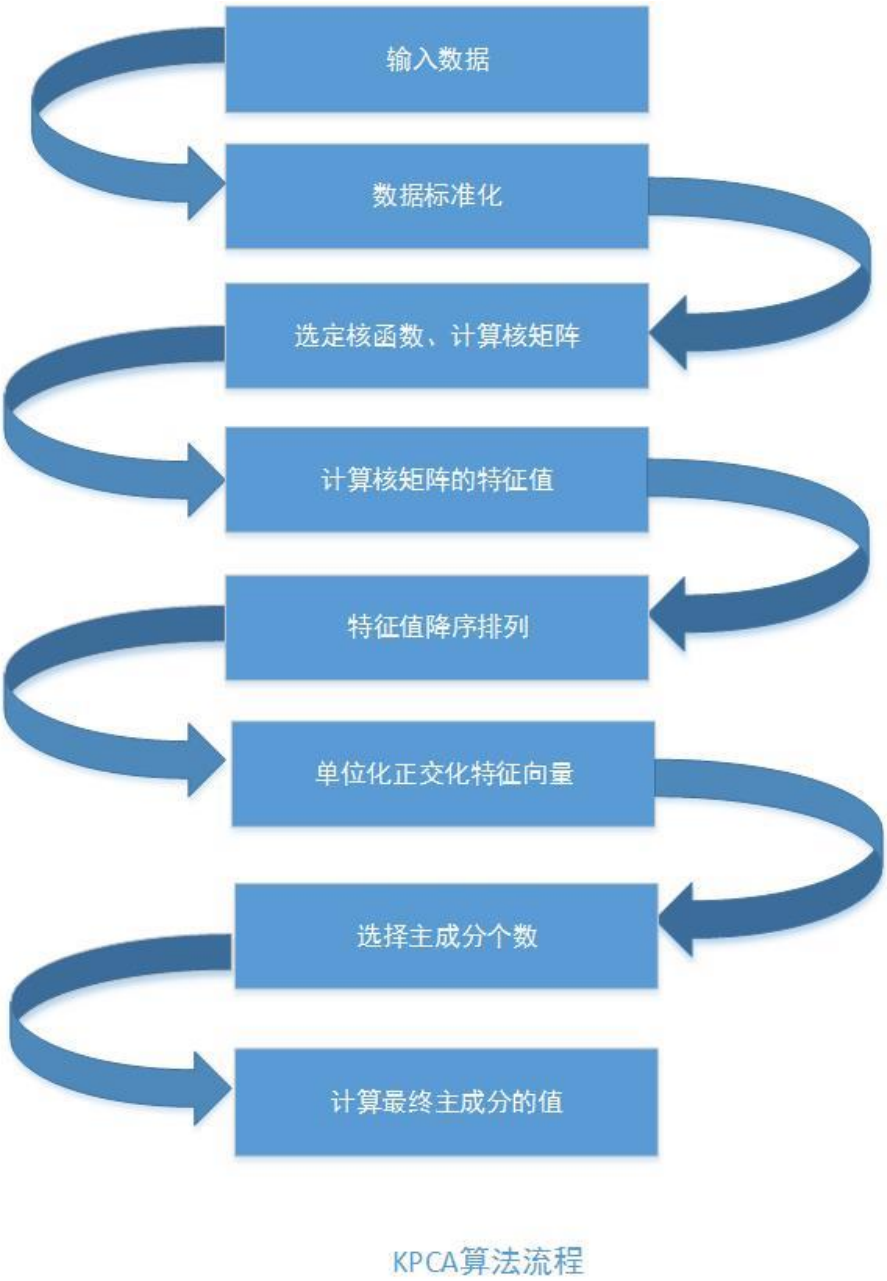

图 7 KPCA 算法流程

在运行了代码之后，我们得到了 131013 组 3*2 阶的特征向量（具体数据会在附录中列出），每分钟的状态均由两组主成分来表示，这两组组成分大概保留了原数据矩阵 80%的信息。

然后我们绘制了两组主成分与时间的关系图。

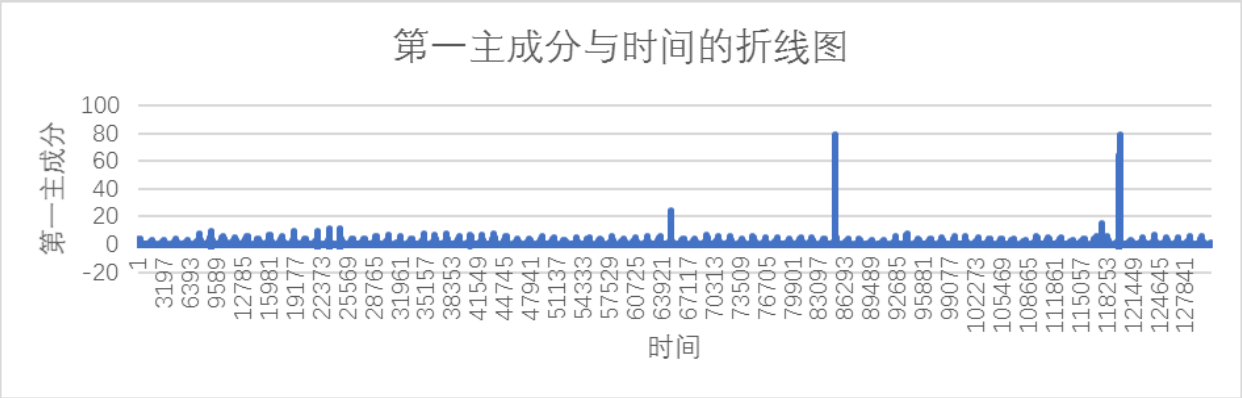

图 8 第一主成分与时间的折线图

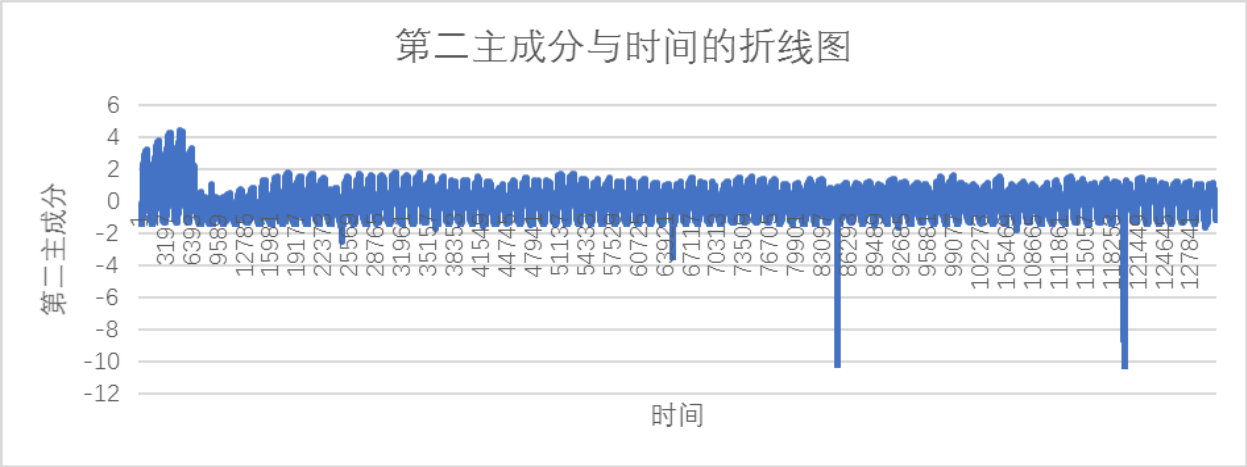

图 9 第二主成分与时间的折线图

通过观察两组主成分与时间的折线图，我们可以得出两组主成分仍然符合周期性的规律。第一组主成分更偏向于表现成功率与响应时间的数据，而第二主成分更偏向于表现交易量和响应时间的数据。

5.1.5 对于第一问的总结

在第一问里，我们先做到对数据的整体的分析。然后定义了两种类型的交易状态——分钟状态和自然日状态。并选取了每一天业务量、成功交易率、交易响应时间的均值和方差所组成的 3*2 阶的矩阵作为普通自然日状态的特征值。运用 KPCA 算法进行降维分析，提取了分钟交易状态的特征值。在 KPCA 算法中，我们还得到了用来计算非线性主成分的特征向量 α_k^i。

5.2 交易状态异常检测的方案

5.2.1 交易状态检测方案的给出

我们设计了一种交易状态检测方案，可以用下面的流程图给出。

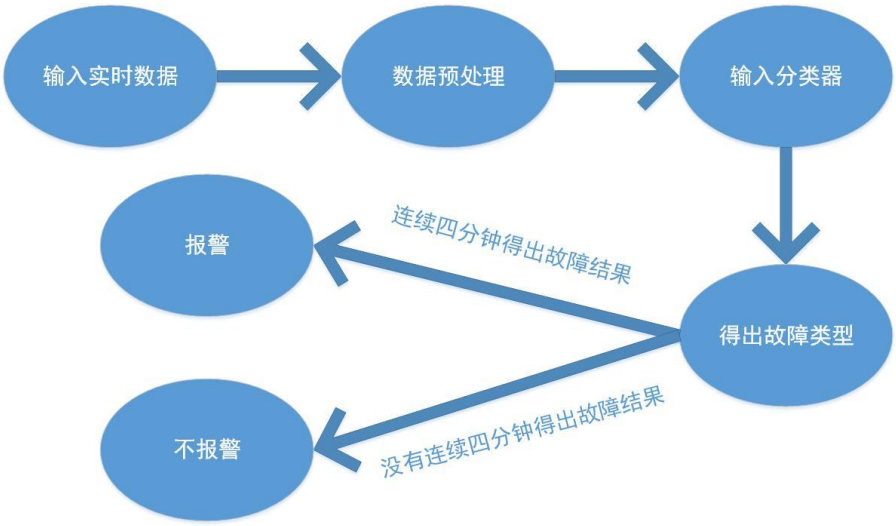

图 10 交易状态检测方案

现在我们稍微解释一下这整个交易状态检测方案的运行机制：我们计划对每一分钟的状态进行检测，我们输入的实时数据是从银行总行数据中心反馈的数据，应该包括日期、时间、交易量、成功率与响应时间。然后我们对数据进行预处理，即提取输入的实时数据代表的那一分钟的分钟状态特征值，然后输入分类器。通过分类器的辨别，得出故障类型。如果连续四分钟得出故障的状态，那么就选择报警；反之则不报警。

下面，我们将对整个交易状态检测方案中的数据预处理、分类器的设计还有如何降低虚警误报。

5.2.2 数据预处理

1. 对数据预处理问题的分析和对第（1）问的回顾

由于我们从银行的总行数据中心所得到的数据是包括了日期、时间、交易量、成功率和响应时间的五组分钟数据，我们要将它化成和第一问中得到的分钟状态特征值一致的形式，这样子才有利于后面输入分类器中的分类。

现在我们考虑第一问中我们对于 KPCA 算法：在 KPCA 算法的最后，先对数据进行归一化，即作出如下变换：

$$X_{ij} = \frac{x_{ij} - \bar{x}_j}{\sqrt{Var(x_j)}}$$

使得数据满足中心化条件，即 $\sum_{i=1}^{n} x_i = 0$。

然后通过对于核函数的选取、核矩阵的变换等等步骤得出最终的非线性主成分可以被表示为：

$$X' = (U^n \cdot \phi(X)) = \sum_{k=1}^{n} \alpha_k^i (\phi(x_j)\phi(x_k))$$

我们对实时数据的预处理也是包括两项，数据标准化和非线性成分的计算。

2.数据标准化

我们对输入的实时数据中的交易量、成功率还有响应时间作出如下变换：

$$X_{ij}' = \frac{x_{ij} - \bar{x}_j}{\sqrt{Var(x_j)}}$$

其中，我们考虑到对于实时数据来说，他一天的均值 $\bar{x}_j$、方差 $Var(x_j)$ 尚未清楚， 我们用第（1）中得到的标准自然自然工作日的特征矩阵和标准自然非工作日特征矩阵中的数据来代替，即下面的这两个矩阵：

$$X^{SWD} = \begin{bmatrix} 631 & 0.9588 & 101 \\ 264,102 & 0.0007 & 222 \end{bmatrix}$$

$$X^{SOD} = \begin{bmatrix} 516 & 0.9579 & 99 \\ 177,169 & 0.000794848 & 232 \end{bmatrix}$$

3.非线性主成分的计算

我们对已经标准化的数据通过以下线性变换，得到它对应的非线性主成分：

$$X' = (U^n \cdot \phi(X)) = \sum_{k=1}^{n} \alpha_k^i (\phi(x_j)\phi(x_k))$$

其中，核函数 $\phi(x_j)\phi(x_k)$ 采用高斯径向基（REF）函数。α_k^i 的值采用第（1）问中我们从 KPCA 算法中得到的特征向量。总之各种预设与前面保持一致。

5.2.3 分类器的设计

1.分类器的设计思路

第一问中我们得到了分钟交易状态的特征值、普通自然日交易状态的特征值。现在我们要设计一个分类器，使得我们能够通过输入的经过预处理的的数据得出某一分钟数据的状态。首先我们必须先把原有数据的异常值找出来，然后再进行分类器的设计。这其实是非监督学习和监督学习的结合。非监督学习即是聚类，把异常值与正常值分开。然后分类器的过程属于监督学习的过程。

在非监督学习中，我们选择 K-means 算法作为聚类算法的主体。这个算法原理

简便，速度快。但是这个算法有着显著的缺点：容易陷入局部极值解。对于这个问题，我们使用粒子群（PSO）算法来解决这个问题，粒子群算法是搜寻全局最优的好办法。

在进行聚类之后，我们就对数据打上了一个标签。将已经打上标签的数据分成训练集和检验集，接下来我们使用人工神经网络对聚类之后的数据进行学习。在本问中，我们采取最为经典的 BP 神经网络算法。

2.K-means 算法

K-means 算法是一种简单使用的聚类算法。

假定我们要进行聚类的的一共有 n 个对象，构成一个样本数据集 $X=\{x_1, x_2\cdots, x_n\}$. 我们的目标是将这 n 个对象分为 k 个类。具体算法步骤如下：

①先把原始数据输入进去，为 k 个聚类选取 k 个初始聚类中心，每个聚类中心记为 $Z_j(I)$， j=1,2，…,k。

②计算数据集 $X=\{x_1, x_2\cdots, x_n\}$ 中每个数据与聚类中心的距离 $D(x_i, Z_j(I))$，i=1,2，…，n，j=1,2，…，k。现在做出如下判定：如果 $D(x_i, Z_t(I))=\min\limits_{j=1,2,\cdots k}\{D(x_i, Z_j(I)), i=1,2, \cdots, n\}$，则 $x_i \in \omega_c$。

③将每个聚类中的样本均值作为 k 个新的聚类中心，即

$$Z_j(I+1)=\frac{1}{n}\sum_{k=1}^{n_j} x_k^j \quad , \quad j=1,2, \cdots, k \quad 。$$

④现在再做一次判定，若 $Z_j(I+1)\neq Z_j(I)$，j=1,2，…k，则 $I=I+1$，返回第②步；否则算法结束。

⑤最后得到的 k 个聚类中心就是我们想要的聚类结果。

3.粒子群（PSO）算法

粒子群是模拟群鸟觅食的模型，它在寻找全局最优的应用上具有十分广泛的作用。在群鸟觅食的模型中。我们把每一只鸟儿都看作一个粒子，那么整个鸟群可以被看成一个粒子群。设在某一个 D 维的目标搜索空间中，有 m 个粒子所组成的粒子群，其中第 i （i=1,2，…，m）个粒子正处于位置 $x_i=(x_i^1, x_i^2, \cdots, x_i^D)$，这个位置可能正好是一个最优解，将 X_i 带入目标函数就可以得出其适应值，通过适应值的大小可以判断出这个位置的优劣。假设个体粒子经过的最好位置是 $P_i=(p_i^1, p_i^2, \cdots, p_i^D)$，整体粒子群所经过最好的位置是 $P_g=(p_g^1, p_g^2, \cdots, p_g^D)$。第 i 粒子的速度记为

S_i=（s_i^1，s_i^2，…，s_i^D）。

在粒子群算法运行中，用以下公式对粒子的位置进行不断的更新：

$$s_i^d=\omega s_i^d+c_1r_1(p_i^d-x_i^d)+c_2r_2(p_g^d-x_i^d)$$

$$x_i^d=x_i^d+\alpha s_i^d$$

其中，i=1，2，…，m；d=1,2，…，D；惯性因子$\omega\geq 0$；加速常数c_1、$c_2\geq 0$；r_1、r_2是在$[0,1]$内变化的随机数；约束因子α控制速度的权重；$s_i^d\in[-s_{max}^d，s_{max}^d]$，表明粒子的速度被最大速度限制，这个最大速度可以人为设定。最后，粒子群位置的改变，也即迭代次数可以被人为设定。

4.基于 PSO 算法改进的 K-means 算法

通过 5.2.2-5.2.3 对于 K-means 算法和 PSO 算法的描述，我们在这里给出整个基于 PSO 算法改进的 K-means 算法的流程。

步骤 1：输入数据与粒子群的初始化操作

①输入带聚类的数据集S，最终聚类的数目k，粒子群中的粒子总数m，最大的迭代次数t_{max}，并设置粒子群的状态连续无法得到改善的累积次数N_{pi}的阈值$thre_p$和粒子群最优解无法得到改善的累积次数N_g的阈值$thre_g$。

步骤二：利用 PSO 算法对粒子群进行迭代搜索

①对粒子群初始化操作。从数据集S中随机选取k个中心点作为粒子的位置X_i的处置的初值，记为Z_j，j=1,2，…,k。并初始化粒子的速度s_i、粒子个体最优位置P_i及其极值f（P_i），粒子群体最优位置P_g及其极值f（P_g）。其中，f（x）=$\sum_{j=1}^{k}\sum_{s\in C_j}\|S_i-Z_j\|$，是粒子的适应度函数。这一过程一共循环进行 m 次，m 次循环后就完成了粒子群的初始化构造。

②根据$\omega(t)=\omega_{max}-(\omega_{max}-\omega_{min})t/t_{max}$调整动态权重。其中，$\omega_{max}$是最大的惯性权重，一般取值为 1.4；$\omega_{min}$是最小惯性权重，一般取值为 0；t 是当前的迭代次数；t_{max}是粒子群的最大迭代次数。

③根据$s_i^d=\omega s_i^d+c_1r_1(p_i^d-x_i^d)+c_2r_2(p_g^d-x_i^d)$、$x_i^d=x_i^d+\alpha s_i^d$调整粒子的速。其中，i=1，2，…，m；d=1,2，…，D；惯性因子$\omega\geq 0$；加速常数c_1、$c_2\geq 0$；r_1、r_2是在$[0,1]$内变化的随机数；约束因子α控制速度的权重。；$s_i^d\in[-s_{max}^d，s_{max}^d]$。

④根据 $x_i(t+1)=x_i(t)+H_0(1-\frac{t}{t_{maz}})\ s_i(t+1)$ 调整粒子的位置。其中，飞行常数 H_0 通常取值为 1.5；t 是当前的迭代次数；t_{max} 是粒子群的最大迭代次数。

⑤根据最近邻原则来划分数据集，并计算粒子的适应度。

⑥若粒子的适应度值小于个体的极值，则更新粒子的个体最有位置 P_i 和个体极值。否则就在表示粒子群最优解无法得到改善的累积次数 N_{pi} 值上加 1.

⑦若所有粒子个体的及时的极值小于粒子的全局极值，则更新粒子群的最优位置 P_g 和全体极值。否则就在粒子群最优解无法得到改善的累积次数 N_g 值上加 1.

⑧如果 N_{pi} 大于 $thre_p$，则对粒子 i 执行变异操作，即对变异粒子执行一次 K-means 操作，使粒子的速度、位置、个体最优位置获得重新的初始化。

⑨如果 N_g 大于 $thre_g$，则对适应度值较小的前几个粒子执行变异操作。

⑩根据群体适应度方差 $\sigma^2=\frac{1}{m\sum_{i=1}^{m}\left[f(x_i)-f_{avg}\right]}$ 来判断粒子群是否达到收敛或者循环达到最大迭代次数 t_{max}，若达到，则终止粒子群的迭代并使用粒子群最优位置 P_g 作为步骤三聚类中心的初值；否则再次进行步骤 2 的循环。

步骤三： K-means 算法的运行

①将步骤二中 PSO 算法得到的粒子群最优位置 P_g 作为，为 k 个聚类选取 k 个初始聚类中心，每个聚类中心记为 $Z_j(I)$， j=1,2，…,k。

②计算每个样本与聚类中心的距离 $D(x_i, Z_j(I))$ ， i=1,2，…，n ，j=1,2，…，k。如果 $D(x_i, Z_t(I))=\min_{j=1,2,\cdots k}\left\{D(x_i, Z_j(I)), i=1,2,\cdots, n\right\}$，则 $x_i\in\omega_c$。

③将每个聚类中的样本均值作为k个新的聚类中心，即

$$Z_j(I+1)=\frac{1}{n}\sum_{k=1}^{n_j}x_k^j \quad , \quad j=1,2,\cdots, k \quad 。$$

④ 若 $Z_j(I+1)\neq Z_j(I)$， j=1,2，…k，则 $I=I+1$ ，返回第②步；否则算法结束。

5.聚类的实施和故障类型的判断

我们在 Python 3.5 中编写基于 PSO 算法改进的 K-means 算法程序，对于一天中的每一分钟的 91 组数据进行聚类操作。我们设置$k=2$，即暂时只判断出一个分钟特征数据是否是正常的，而先不考虑故障的数据的故障类型是哪一种。

在运行完程序之后，我们得到了一天中 1440 分钟中按每一分钟分开的故障分钟和非故障分钟的数据。接下来，我们要借助我们第一问中得到的普通自然日状态的特征值来判断故障数据的故障类型。

具体怎么判断，我们在这里简单说明一下。

对于属于故障状态的数据，我们需要查询它所在日期的普通自然日状态特征值。假设他的普通自然日状态的特征矩阵如下：

$$X_i^{ND}=\begin{bmatrix} mean(v_i^{min}) & mean(r_i^{min}) & mean(v_i^{min}) \\ var(v_i^{min}) & var(r_i^{min}) & var(t_i^{min}) \end{bmatrix}$$

然后我们从没有出现故障的数据中查询不同日期的统一分钟数的的普通自然日状态特征矩阵。假设其普通自然日状态特征矩阵为：

$$X_i^{ND'}=\begin{bmatrix} mean(v_i^{min})' & mean(r_i^{min})' & mean(v_i^{min})' \\ var(v_i^{min})' & var(r_i^{min})' & var(t_i^{min})' \end{bmatrix}$$

如果两个矩阵中的关于交易量、反应时间、成功率的方差有较大的差别，那么我们就可以将他们分为题干中出现的四种故障之一。例如，如果$var(t_i^{min})' << var(t_i^{min})$，则是交易的响应时间太长，那么可能是题干中提到的“数据中心后端处理系统异常（如操作系统 CPU 负荷过大）引起交易处理 缓慢，影响交易响应时间指标；”故障或者“数据中心后端处理系统应用进程异常，导致交易失败或响应缓慢。”故障。

在进行完手工分类后，为了方便表示，我们做出对于故障状态的如下定义：

表 1 故障类型与故障代码

故障代码	故障类型
0	没有故障；
1	分行侧网络传输节点故障，前端交易无法上送请求，导致业务量陡降；
2	分行侧参数数据变更或者配置错误，数据中心后端处理失败率增加，影响交易成功率指标；
3	数据中心后端处理系统异常（如操作系统 CPU 负荷过大）引起交易处理缓慢，影响交易响应时间指标；
4	数据中心后端处理系统应用进程异常，导致交易失败或响应缓慢。

到现在，我们对于每一组数据都得到了一组故障代码。接下来我们要利用 BP 神经网络来学习数据的特征，做到监督学习的目的。

6. REF 神经网络

REF 神经网络是神经网络的一种，它模仿视网膜的感受功能而产生。具体操作如下：

用以下的函数对隐含层进行建模：

$$\varphi(x)=\mathrm{G}(\frac{\|x-c_i\|^2}{\sigma_i})$$

其中，x 为输入样本，c 为感受视野中心，即中心点；σ_i 称为宽度。$G(\mathrm{x})$ 为径向基函数，也被称为激活函数。$\|x-c_i\|$ 是距离函数。

7. REF 神经网络的设计

整个神经网络设计为最简单的三层结构。输入层和输出层都采用由于我们的分钟状态特征值有两个，所以我们的输入层有两个节点。隐含层我们使用 Guassian 函数，即：

$$\varphi(x)=\exp(-\frac{\|x-c_i\|^2}{\sigma_i^2})$$

5.2.4 如何减少虚警误报

为了减少虚警误报的情况，我们对是否报警做出如下限制：如果分类器给出的结果是连续四分钟故障，我们便采取报警；如果没有四分钟连续故障，那么我们选择不报警。

关于四分钟是怎么得出来的，我们观察到附件该出的数据中有以下几组数据：

0558	359	92	0.9239	1374.64	126.4669	116.8428
0559	360	95	0.9053	1332.24	126.5628	114.5773
0600	361	49	0.3673	33627.39	1647.742	605.2157
0601	362	18	0.0000	57174.78	1029.146	0
0602	363	18	0.0000	57177.44	1029.194	0
0603	364	17	0.0000	57067.59	970.149	0
0622	383	3	1.0000	225	0.675	0.675
0623	384	26	0.9231	103.62	2.69412	2.486942
0624	385	85	0.9765	117.47	9.98495	9.750304

图 11 连续故障示意

这是一组出现在 4 月 16 日的故障数据，可以看出，成功率的数据在 05:59 之后连续下降 4-5 分钟之后被发现，然后整个系统开始进行维修，直到 06:22 分后系统才恢复正常。

所以根据上述的情况，我们采取 4 分钟为警戒时间线。

5.3 采集数据的增加

5.3.1 对前两问的总结与分析

在第（1）问中，我们具体完成了对状态的定义，还有各种状态的特征值提取。其中，在数据一切正常的假设下，我们能对整个过程改进的就是对于标准自然工作日和标准自然非工作日的特征矩阵的求取。

在第（2）问中，我们对数据先进行聚类，为每一组数据打上正常和故障的标签。接着我们通过对于普通自然日状态特征矩阵的分许来判断故障的类别，然后运用 BP 神经网络对数据进行学习。最后我们减少虚警误报的方式是如果连续 4 分钟发出故障的判断结果，那么就选择报警。在这第（2）问里，能过改进的就是对于故障类型的判断，还有报警满足条件的改善，也即应该连续几分钟判断出故障才应该报警。

5.3.2 采集数据增加的种类与益处

对于第（3）问，我们选择添加的采集数据是 ATM 应用系统的维修维修记录，这份记录应该包括了每次 ATM 系统故障的故障类型和修复故障应该有的时间。

添加了每次 ATM 系统的故障类型，则我们就不需要通过聚类算法和手动判断来确认故障的类型，这提高了数据的准确度。然后是通过维修记录的观察，我们更容易的能找出那个理论上没有故障的自然日，这有利于我们对标准自然工作日状态特征矩阵和标准自然非工作日状态特征矩阵的求取，这是我们能对实时数据做出更好的数据预处理。

添加了每次故障修复应有的时间，这有利于我们对于“应该连续几分钟判断出故障才应该报警”这个问题中的报警条件的确定。

六、 模型检验

对于前文建立的模型，其实最需要检测的就是第（2）问中建立的故障检测方案。为此，我们在训练 BP 神经网络之前，把 1440 组数据中的 91 个数据随机分成训练集和检验集。训练集运用在 BP 神经网络权值的训练上，而检验集则作为检验模型准确性的数据，充当输入的实时数据。

七、 模型评价

7.1 模型的优点

第（1）问中建立的模型，运用核主成分分析法可以减少变量的维数，为后面的计算节约时间。

第（2）问中建立的模型，运用 PSO 算法改进的 K-means 算法有效避免了陷入区域极值的情况；同时使用 REF 神经网络用来对分类的数据进行学习，有比较好的效果。

7.2 模型的缺点

第（1）问中，我们运用和主成分分析法得出的两组主成分的意义尚不明确，这对于我们对于这个方法的推广有所不利。而且，通过主成分分析法提取的信息，在本题中约有 20%的遗失，这些遗失的信息对于题目的影响我们尚未明确。

参考文献

【1】 韦振中，基于核主成分分析的特征值提取方法，广西工学院学报，17（4）：27-31,2006。

【2】 谢秀华、李陶深，一种基于改进 PSO 的 K-means 优化聚类算法，计算机技术与发展，24（2）：34-38，2014.

【3】 卓金武，Matlab 在数学建模中的应用，北京：北京航天航空大学，2014。

【4】 月韵悠扬，核主成分分析 KPCA matlab 程序，http://blog.sina.com.cn/s/blog_7671b3eb01012d9s.html，2017.6.29。

附录

附录 1 普通自然日状态特征矩阵

日期	mean(v_i^{min})	var(v_i^{min})	mean(r_i^{min})	var(r_i^{min})	mean(v_i^{min})	var(t_i^{min})
1	974	643959.3374	0.9563	0.000561567	85	290.031893
2	1,115	822969.4003	0.9574	0.000428171	84	317.870217
3	1,275	1042877.757	0.9573	0.000378693	83	380.6562336
4	1,303	1104323.92	0.9566	0.00036928	83	324.2045719
5	772	523994.2306	0.9553	0.000348806	89	263.2616169

6	284	54431.25569	0.9590	0.001061359	99	1817.140573
7	264	51537.82279	0.9586	0.001279456	96	136.2727117
8	319	75863.36979	0.9605	0.000993733	95	145.1349115
9	356	99032.65479	0.9607	0.000863356	95	195.2724213
10	388	115969.3471	0.9605	0.00084761	94	254.2564261
11	482	180800.6498	0.9600	0.000729619	91	158.2584864
12	543	232740.9226	0.9588	0.001095345	90	174.8566621
13	611	279241.7188	0.9604	0.000704556	90	224.2085968
14	584	244062.1616	0.9586	0.000858335	89	250.7399056
15	615	281326.8694	0.9590	0.000696972	88	176.9151989
16	559	225321.8543	0.9589	0.001008228	89	146.8042316
17	426	125794.4125	0.9601	0.000847094	94	2706.192144
18	572	237432.9799	0.9599	0.000765005	127	217825.0423
19	613	270072.3869	0.9602	0.000692133	89	186.6498902
20	591	249021.0641	0.9575	0.000685178	89	230.4591431
21	607	251906.743	0.9566	0.000839073	90	742.7883869
22	667	298459.8399	0.9590	0.000690778	87	202.4849058
23	633	256156.8112	0.9589	0.00062259	88	195.4599644
24	667	278989.9597	0.9592	0.000587781	88	239.236424
25	615	237204.2222	0.9585	0.000652035	89	639.0887131
26	585	220244.1216	0.9592	0.000679765	92	7620.800333
27	563	205529.6468	0.9574	0.000757986	90	330.4393948
28	563	206134.2539	0.9563	0.000737924	89	177.7345266
29	577	219004.0551	0.9587	0.000752033	88	166.4043006
30	544	196382.086	0.9580	0.000820681	93	10351.8538
31	493	152541.7994	0.9598	0.000936508	91	169.4849003
32	565	198973.777	0.9608	0.000617792	91	298.8404873
33	608	228150.3972	0.9587	0.000575487	90	238.3492468
34	582	207370.7798	0.9582	0.000563543	91	2434.627353
35	557	202379.1834	0.9584	0.00073911	90	469.4147445
36	676	292029.813	0.9572	0.000817526	89	165.1506123
37	695	292198.2951	0.9607	0.000468963	89	199.4426613
38	614	236313.1928	0.9596	0.000604012	90	505.7939653
39	610	223537.216	0.9589	0.000637507	89	167.3347169
40	602	212764.1538	0.9599	0.00056691	90	401.2809243
41	559	185244.9591	0.9575	0.000707455	90	202.5407874
42	569	202658.6772	0.9575	0.000608345	90	1027.202151
43	599	218262.2853	0.9585	0.000712727	89	167.1492261
44	555	182225.2329	0.9585	0.00067562	90	153.0609137
45	563	183268.6965	0.9590	0.000725733	90	180.6312595
46	563	184082.9161	0.9596	0.000659206	103	198223.8179
47	629	230390.6382	0.9588	0.000612193	89	166.6816622
48	558	182820.6924	0.9581	0.000602908	89	211.1630042

49	479	152636.1814	0.9585	0.000658933	90	153.3404874
50	552	185127.8035	0.9590	0.000736641	90	189.455183
51	619	228850.4031	0.9600	0.000572352	89	191.6003728
52	648	243516.741	0.9588	0.000635602	88	177.2611718
53	621	221376.1835	0.9588	0.000670273	89	390.8032241
54	602	210461.3985	0.9593	0.000529507	89	175.5706517
55	528	164878.7658	0.9571	0.000664991	90	163.6484755
56	516	166510.3736	0.9573	0.000615743	113	172.2239512
57	561	195199.6606	0.9598	0.000735233	119	219.6517626
58	627	229903.4518	0.9580	0.000632436	120	504.6471848
59	456	126898.6014	0.9590	0.000574415	120	225.2675024
60	556	186568.0309	0.9522	0.007225117	562	22778461.58
61	556	175145.2217	0.9605	0.000522539	91	255.3867554
62	575	183127.0653	0.9574	0.00060422	90	170.3846206
63	534	165916.525	0.9590	0.000489728	92	2106.780572
64	622	225161.0961	0.9590	0.000485181	92	301.8733367
65	571	186848.795	0.9595	0.000575632	95	6787.966797
66	563	179397.468	0.9581	0.000821381	91	521.6437591
67	495	147052.9472	0.9591	0.000573091	92	131.4242323
68	650	239481.6621	0.9604	0.000539013	91	210.8954486
69	653	231697.9249	0.9590	0.000474632	88	220.345283
70	550	157206.3609	0.9565	0.000533478	90	608.5005529
71	514	144642.6989	0.9577	0.000559055	90	498.0855549
72	504	155327.5381	0.9574	0.000536569	90	309.1819095
73	650	254358.393	0.9591	0.00060597	88	225.2940289
74	505	146110.7622	0.9581	0.000617148	91	623.4462684
75	578	189872.3981	0.9577	0.000560463	97	16027.25001
76	510	152114.8393	0.9600	0.000467378	92	592.6648352
77	491	153419.2493	0.9586	0.000794848	93	1617.122881
78	524	180451.3182	0.9592	0.000627286	90	671.2330532
79	661	250513.7513	0.9608	0.000634279	90	550.6758816
80	602	199784.2662	0.9604	0.000457565	91	270.5685982
81	566	173876.2782	0.9581	0.00061267	90	305.144426
82	601	195220.2611	0.9571	0.000639456	107	152261.6974
83	579	173004.8388	0.9581	0.000549598	88	149.6987117
84	539	173670.5304	0.9509	0.003600638	321	9332820.61
85	645	243310.5932	0.9602	0.000570391	119	190.9437804
86	611	198857.7738	0.9593	0.000444501	119	149.395138
87	570	173340.4126	0.9586	0.000539928	119	225.1966192
88	585	182716.3301	0.9593	0.000559499	90	201.8358846
89	575	177887.443	0.9595	0.000460908	91	121.2433191
90	538	153619.2743	0.9567	0.000500215	92	1965.838038
91	536	157393.0639	0.9583	0.00065344	92	3730.668152

附录 2 第（1）问程序

```
function[train_kpca,test_kpca] = kpcaFordata(train,test,threshold,rbf_var)
%% Data kpca processing
%% 函数默认设置
if nargin <4
rbf_var=10000;%?
end
if nargin <3
threshold = 90;
end
%%  数据处理
patterns=zscore(train); %训练数据标准化
test_patterns=zscore(test); %测试数据标准化
train_num=size(patterns,1); %train_num 是训练样本的个数
test_num=size(test_patterns,1);%test_num 是测试样本的个数
cov_size = train_num; %cov_size 是训练样本的个数
%%  计算核矩阵
for i=1:cov_size,
for j=i:cov_size,
K(i,j)  =  exp(-norm(patterns(i,:)-patterns(j,:))^2/rbf_var);  % 核 函 数
rbf_var ? ?
K(j,i) = K(i,j);
end
end
unit = ones(cov_size, cov_size)/cov_size;%cov_size 是样本的个数
%%  中心化核矩阵
K_n = K - unit*K - K*unit + unit*K*unit;% 中心化核矩阵
%%  特征值分解
[evectors_1,evaltures_1] = eig(K_n/cov_size);
[x,index]=sort(real(diag(evaltures_1))); %sort 每行按从小到大排序，x 为排
序后结果，index 为索引
evals=flipud(x) ;% flipud 函数实现矩阵的上下翻转
index=flipud(index);
%%  将特征向量按特征值的大小顺序排序
evectors=evectors_1(:,index);
%%  单位化特征向量
% for i=1:cov_size
% evecs(:,i) = evectors(:,i)/(sqrt(evectors(:,i)));
% end
train_eigval = 100*cumsum(evals)./sum(evals);
index = find(train_eigval >threshold);

train_kpca = zeros(train_num, index(1)); %train_num 是训练样本的个数
```

```
%% evecs 单位化后的特征矩阵,K_n 训练数据的中心化核矩阵
train_kpca=[K_n * evectors(:,1:index(1))];
%% 重建测试数据
unit_test = ones(test_num,cov_size)/cov_size;%cov_size 是训练样本的个数
K_test = zeros(test_num,cov_size); %test_num 是测试样本的个数， cov_size
是训练样本的个数
for i=1:test_num, %test_num 是测试样本的个数
for j=1:cov_size,%cov_size 是训练样本的个数
K_test(i,j) = exp(-norm(test_patterns(i,:)-patterns(j,:))^2/rbf_var);
end
end
K_test_n = K_test - unit_test*K - K_test*unit + unit_test*K*unit;
test_kpca = zeros(test_num, index(1));%test_num 是测试样本的个数
test_kpca = [K_test_n * evectors(:,1:index(1))];
```

附录 3 第（2）问程序

```
% 本程序添加了动量因子
% 本程序是基于梯度训练算法的 RBF 网络
% 所以添加动量因子以便不会轻易陷入局部极小值
function main()
clc;
close all;
clear all;
warning off;
% 初始化参数
% 该网络有三层，输入层和输出层都是线性函数，隐含层由距离函数和激活函数构
成
SamNum = 120;
TargetSamNum = 60;
% 样本输入维度
InDim = 1;
% 隐含层神经元数量
UnitNum = 10;
MaxEpoch = 10000;
% 目标误差
E0 = 0.09;

% 输入置于[1,60]区间的随机数
% 理论样本
SamIn = sort(59*rand(1,SamNum)+1);
SamOut = 0.5447*SamIn.^0.1489;

% 实际样本
TargetIn = 1:60;
```

```
TargetOut = […
…
…
…
…
…
…
…
]';

figure;
hold on;
% 添加边框和网格线
box on;
grid on;
plot(SamIn,SamOut,'kO');
plot(TargetIn,TargetOut,'b-');
xlabel('x');
ylabel('y');
title('训练和测试图');

% 原始样本对（输入和输出）初始化
[SamIn,minp,maxp,SamOut,mint,maxt] = premnmx(SamIn,SamOut);

% 利用原始输入数据的归一化参数对新数据进行归一化；
TargetIn = tramnmx(TargetIn,minp,maxp);
TargetOut = tramnmx(TargetOut,mint,maxt);

% 初始化数据中心
Center = 8*rand(InDim,UnitNum)-4;
% 初始化宽度
SP = 0.2*rand(1,UnitNum)+0.1;
% 初始化权值，这里的权值是指隐含层与输出层之间的权值
% 跟 BP 网络不同的是，这里没有输入层与隐含层之间的权值
% 所以单从网络结构上讲，RBF 网其实更简单
W
= 0.2*rand(1,UnitNum)-0.1;

% 数据中心学习速度（速率）
lrCent = 0.02;
% 宽度学习速度（速率）
lrSP = 0.001;
% 权值学习速度（速率）
lrW = 0.001;
```

```
% 动量因子系数
arf = 0.001;

% 用来存储误差
ErrHistory = [];

for epoch = 1:MaxEpoch
% 计算书中输出样本与数据中心之间的距离（这里是欧式距离）
% 相当于书中的||x-c||表达式，其具体表达式为 sum((x-y).^2).^0.5
    AllDist = dist(Center',SamIn);
SPMat = repmat(SP',1,SamNum);
% 以高斯函数作为激活函数，高斯函数表达式为 exp(-n^2)
UnitOut = radbas(AllDist./SPMat);
% 隐含层神经元数据经过加权后在输出层输出
% 输出层是线性激活函数所以直接加权输出即可
    NetOut = W*UnitOut;
    Error = SamOut-NetOut;

    SSE = sumsqr(Error)
    ErrHistory = [ErrHistory SSE];

    if SSE<E0,break,end

        % 初始化用来存储前一次调整量的变量，全部用 0 矩阵填充
        CentGrad = zeros(size(Center(:,1)));
        SPGrad = zeros(size(SP(1)));
        WGrad = zeros(size(W(1)));

    for i = 1:UnitNum

       % 存储前一次训练的调整量，以下是数据中心前一次调整量赋给变量
CenterPre
       % 以便后续动量因子会用到
        CenterPre = CentGrad ;
       % 宽度前一次调整量
        SPPre = SPGrad ;
       % 隐含层与输出层之间的连接权值矩阵前一次调整量
        WPre = WGrad ;

        % 数据中心的网络反向调整
        % 原理仍然是链式偏导
        % 根据求导，数据中心的调整量 CentGrad 等于 （样本-数据中心）*误差*
隐含层网络输出
% *权值/宽度平方
```

```
% 以下宽度调整量和权值调整量类似
CentGrad=(SamIn-repmat(Center(:,i),1,SamNum))*(Error.*UnitOut(i,:)*W(i)/
(SP(i)^2))';
    SPGrad = AllDist(i,:).^2*(Error.*UnitOut(i,:)*W(i)/(SP(i)^3))';
        WGrad = Error*UnitOut(i,:)';

        % 更新网络的数据中心、宽度以及权值
        Center(:,i) = Center(:,i)+lrCent*CentGrad;
        SP(i) = SP(i)+lrSP*SPGrad;
        W(i) = W(i)+lrW*WGrad;

        % 梯度法训练 RBF 网络同样不能幸免极易陷入局部极小值的缺陷，所以这
里添加动量因子
        Center(:,i) = Center(:,i)+ arf*CenterPre ;
        SP(i) = SP(i)+ arf*SPPre ;
        W(i) = W(i) + arf*WPre ;
    end
end

% 测试网络的性能如何
TestDistance = dist(Center',TargetIn);
TestSpreadsMat = repmat(SP',1,TargetSamNum);
TestHiddenUnitOut = radbas(TestDistance./TestSpreadsMat);
TestNNOut = W*TestHiddenUnitOut;
% 训练样本进行了归一化，所以需要还原
TestNNOut = postmnmx(TestNNOut,mint,maxt);
TargetIn = postmnmx(TargetIn,minp,maxp);
plot(TargetIn,TestNNOut,'r-');
axis tight;
legend('理论样本','实际样本','测试样本');

figure;
hold on;
grid;
box;
[xx,Num] = size(ErrHistory);
plot(1:Num,ErrHistory,'k-');
xlabel('训练次数');
ylabel('误差大小');
title('训练误差图');
```