

2018 年 12 月

哈爾濱工業大學

# 大数据计算基础

题    目： 时间序列数据的多种工作状态分类  
与模式识别

专    业： 大数据

学    号： 1160300814

姓    名： 姜思琪

课程类别： 必修

# 1. 问题描述

题号：必修 1.

题目名称：时间序列数据的多种工作状态分类与模式识别【X = 1.2】

问题描述：

## 1.1 问题的背景/意义

进入 21 世纪以来，智能制造概念不断深化。近年来，工业 4.0、工业互联网、中国制造 2025 将智能制造进一步推向公众视野。物联网、大数据、云计算等新一代信息技术的快速发展及应用，使得智能制造被赋予了新的内涵，即新一代信息技术下的智能制造。在数据驱动的制造模式下，工业 4.0 时代下的制造业正演变成一项由软件技术主导的系统性工程。在智能制造系统中，高素质、高智能的人将发挥更好的作用，机器智能和人的智能真正集成在一起。

时间序列是工业大数据中一种常见的重要类型，而生产过程中机器传感器采集的不同类型时间序列对应着不同工作状态，表现为不同类型的序列模式。对于这些工作状态的有效识别，能够帮助提高工业数据的异常检测效果，使得异常检测算法能够有的放矢，提高靶向性。因此，本实验的目标为设计并实现时间序列数据上工作状态分类与模式识别算法。

## 1.2 实验内容

### 1.2.1 设计部分：

本实验要求结合具体数据，首先给出时间序列上不同工作状态(人工定义或者由算法学习得到)的形式化定义:然后，制定有效的状态识别流程:最后结合制定的识别流程，设计具体的识别算法，并能对所用算法的类型、思想、步骤和创新点进行合理的论证。

### 1.2.2 实验部分

应用一种大数据计算系统，实现“设计部分”所要求的算法，利用实验数据，设计对比实验，采用合适的实验指标，测试本实验提出的算法与已有算法的性能(包括效果、效率)对比结果，并能够对实验结果进行合理分析。

### 1.2.3 数据来源

本实验的数据来源于真实数据集：国内某火力发电厂(温控)数据：引风机的数据。共 40 项，分别为：引风机 3A 总有功功率、引风机 3A 入口烟气压力、引风机 3A 出口烟气压力、引风机 3A 出口烟气温度、引风机 3A 动叶调节挡板控制指令、引风机 3A 电流、引风机 3A 入口烟气流量、引风机 3A 入口烟气流量、引风机 3A 动叶调节挡板阀位、引风机 3A 驱动端轴承温度 1、引风机 3A 驱动端轴承温度 2、引风机 3A 驱动端轴承温度 3、引风机 3A 推力轴承温度 1、引风机 3A 推力轴承温度 2、引风机 3A 推力轴承温度 3、引风机 3A 支撑轴承温度 1、引风机 3A 支撑轴承温度 2、引风机 3A 支撑轴承温度 3、引风机 3A 非驱动端轴承温度 1、引风机 3A 非驱动端轴承温度 2、引风机 3A 非驱动端轴承温度 3、引风机

3A 电机非驱动端轴承温度 1、引风机 3A 电机非驱动端轴承温度 2、引风机 3A 电机驱动端轴承温度 1、引风机 3A 电机驱动端轴承温度 2 、引风机 3A 电机线圈绕组温度 U1、引风机 3A 电机线圈绕组温度 U2、引风机 3A 电机线圈绕组温度 V1、引风机 3A 电机线圈绕组温度 V2、引风机 3A 电机线圈绕组温度 W1、引风机 3A 电机线圈绕组温度 W2、引风机 3A 轴承振动 X 向 1、引风机 3A 轴承振动 X 向 2、引风机 3A 轴承振动 Y 向 1、引风机 3A 轴承振动 Y 向 2、引风机 3A 液压油压力、引风机 3A 润滑油压力、引风机 3A 油泵出口滤网差压、引风机 3A 润滑油进油管油温、引风机 3A 轴承润滑油回油管油温。

本实验就引风机 3A 轴承润滑油回油管油温的 400000 个实验数据做具体分析。

## 2. 基于的系统/算法

### 2.1 k-Means 算法

K-MEANS 算法是输入聚类个数  $k$ ，以及包含  $n$  个数据对象的数据库，输出满足方差最小标准  $k$  个聚类的一种算法。

$k$ -means 算法接受输入量  $k$ ；然后将  $n$  个数据对象划分为  $k$  个聚类以便使得所获得的聚类满足：同一聚类中的对象相似度较高；而不同聚类中的对象相似度较小。

聚类相似度是利用各聚类中对象的均值所获得一个“中心对象”（引力中心）来进行计算的。具体实现过程见算法设计。

K-Means 聚类算法具有以下优点：

- (1) 该算法简单、快速，原理易于理解；
- (2) 对处理大数据集，该算法相对是可伸缩和高效率的，其算法复杂度大约是  $O(nkt)$ ，其中  $n$  是所有对象的数目， $k$  是簇的数目， $t$  是迭代的次数；
- (3) 当数据集满足球状密集性或者团状密集性时，其聚类效果很好。但是这个算法也有一些待改进的地方，比如，事先要确定  $k$  值，即要求用户事先知道数据的一些特点，而且该算法经常以局部最优结束，有时很难达到全局最优；该算法对初始聚类的中心比较敏感，对于不同的初始值，其聚类结果也可能有很大的差异；而且该算法只能发现球状簇，其他形状的簇比较难发现；另外，噪声数据对聚类的结果影响也比较大。针对上面种种不足和缺陷，可对该算法提出一些改进。
- (4) 算法能根据较少的已知聚类样本的类别对树进行剪枝确定部分样本的分类；为克服少量样本聚类的不准确性，该算法本身具有优化迭代功能，在已经求得的聚类上再次进行迭代修正剪枝确定部分样本的聚类，优化了初始监督学习样本分类不合理的地方

### 2.2 Baum-Welch 算法

baum-welch 算法是一种对 hmm 模型做参数估计的方法，是 EM 算法的一个特例。EM 算法包含两步：

1:expectation, 计算隐变量的概率分布，并得到可观察变量与隐变量联合概率的最大似然估计在前面求得的隐变量概率分布下的期望。

2:maximization 求得使上述期望最大的新的模型参数。若达到收敛条件则退出，否则回到步骤 1。

此算法可以用于未知隐藏序列的，只知观察序列和转移矩阵等。

### 2.3 高斯分布

正态分布 (Normal distribution)，也称“常态分布”，又名高斯分布 (Gaussian distribution)。

若随机变量  $X$  服从一个数学期望为  $\mu$ 、方差为  $\sigma^2$  的正态分布，记为  $N(\mu, \sigma^2)$ 。其概率密度函数为正态分布的期望值  $\mu$  决定了其位置，其标准差  $\sigma$  决

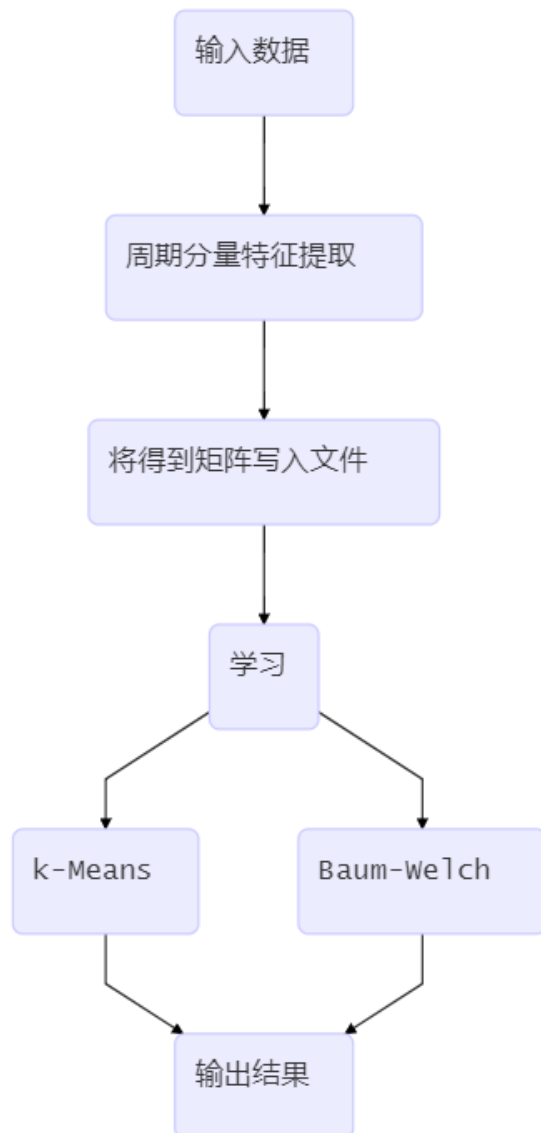
定了分布的幅度。当  $\mu = 0$ ,  $\sigma = 1$  时的正态分布是标准正态分布。

正态分布有极其广泛的实际背景，生产与科学实验中很多随机变量的概率分布都可以近似地用正态分布来描述。例如，在生产条件不变的情况下，产品的强力、抗压强度、口径、长度等指标；同一种生物体的身長、体重等指标；同一种种子的重量；测量同一物体的误差；弹着点沿某一方向的偏差；某个地区的年降水量；以及理想气体分子的速度分量，等等。一般来说，如果一个量是由许多微小的独立随机因素影响的结果，那么就可以认为这个量具有正态分布（见中心极限定理）。从理论上讲，正态分布具有很多良好的性质，许多概率分布可以用它来近似；还有一些常用的概率分布是由它直接导出的，例如对数正态分布、t 分布、F 分布等。

### 3.系统设计/算法设计

#### 3.1 框架

我设计的整个实验流程如下图：



#### 3.2 周期分量特征提取

设计的周期分量方法如下：时间序列中周期分量的特征提取主要为时间序列周期的确定，将根据时间序列周期特点，结合大数据处理方法，来确定周期  $L$ 。

首先根据式 (1) 对时间序列  $X\{x_1, x_2, \dots, x_M\}$  ( $x_i$  范围为  $V_1 - V_2$ ) 进行差分计算得到矩阵  $A$  (其中  $m > 30 \times 48$ ,  $n > 50$ ), 如下所示:

$$A = \begin{Bmatrix} x_2 - x_1 & x_3 - x_2 & \dots & x_{n-1} - x_{n-2} \\ x_3 - x_1 & x_4 - x_2 & \dots & x_n - x_{n-2} \\ \dots & \dots & \dots & \dots \\ x_m - x_1 & x_{m+1} - x_2 & \dots & x_{n+m-3} - x_{n-2} \end{Bmatrix} \quad (1)$$

对矩阵  $A$  的每一行进行线性拟合, 参数分别记为

$$(a_1, a_2, \dots, a_{\{m-1\}}), (b_1, b_2, \dots, b_{\{m-1\}}),$$

将  $A$  的每一行下标分别代入对应的  $Y = aN + b$  中得到对应的  $A'$ , 如式 (2) 所示。

$$\begin{Bmatrix} a_1 \\ a_2 \\ \dots \\ a_{m-1} \end{Bmatrix} \begin{Bmatrix} 1 & 2 & \dots & n-2 \\ 1 & 2 & \dots & n-2 \\ \dots & \dots & \dots & \dots \\ 1 & 2 & \dots & n-2 \end{Bmatrix} + \begin{Bmatrix} b_1 \\ b_2 \\ \dots \\ b_{m-1} \end{Bmatrix} \quad (2)$$

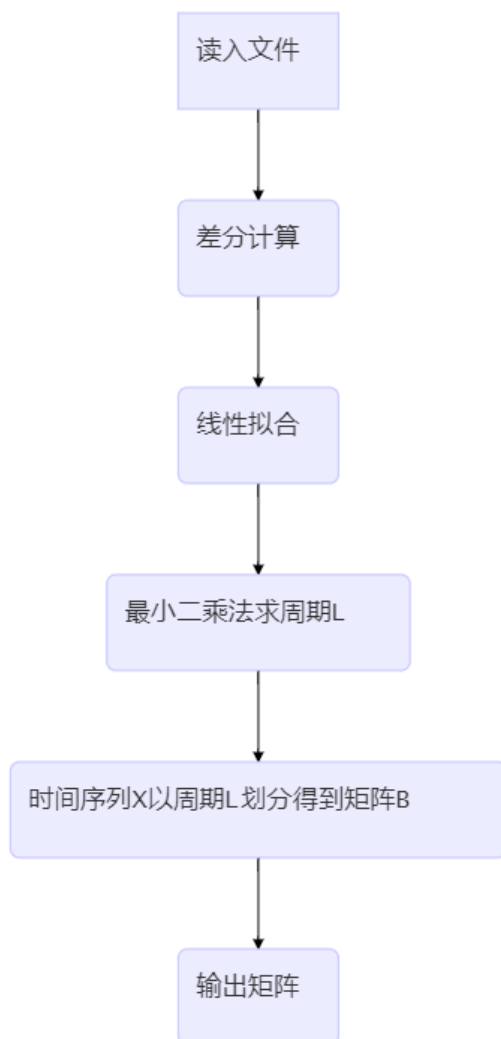
采用最小二乘法 (见式 (3)) 计算  $A$  与  $A'$  每行最小误差, 其中首次出现最小误差的行数即为周期  $L$ 。

$$\left\{ \begin{array}{l} (x_2 - x_1 - a_1 - b_1)^2 + (x_3 - x_2 - 2a_1 - b_1)^2 + \dots + (x_{n-1} - x_{n-2} - (n-2)a_1 - b_1)^2 \\ (x_3 - x_1 - a_2 - b_2)^2 + (x_4 - x_2 - 2a_2 - b_2)^2 + \dots + (x_n - x_{n-2} - (n-2)a_2 - b_2)^2 \\ \dots \\ (x_m - x_1 - a_{m-1} - b_{m-1})^2 + (x_{m+1} - x_2 - 2a_{m-1} - b_{m-1})^2 + \dots + (x_{n+m-3} - x_{n-2} - (n-2)a_{m-1} - b_{m-1})^2 \end{array} \right\} \quad (3)$$

通过周期分量特征提取得到周期  $L$ , 将时间序列  $X$  以周期  $L$  进行划分得到矩阵  $B$ , 其中  $x_M$  为时间序列  $X$  最后一个数据,  $x_M$  之后数据均为空值 NA。

$$A = \begin{Bmatrix} x_1 & x_2 & \dots & x_L \\ x_{L+1} & x_{L+2} & \dots & x_{2L} \\ \dots & \dots & \dots & \dots \\ \dots & x_M & NA & \dots \end{Bmatrix} \quad (4)$$

流程图如下:



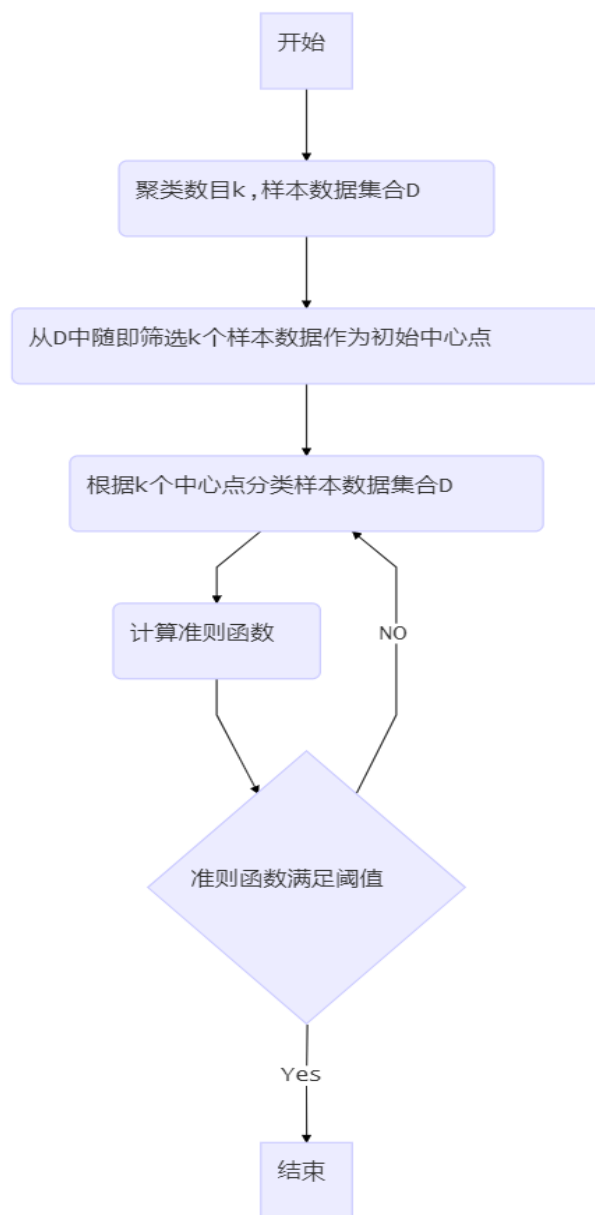
### 3.3 k-Means 算法

思路：

- 1、适当选择  $c$  个类的初始中心
- 2、在第  $K$  次迭代中, 对任意一个样本, 求其到  $c$  各中心的距离, 将该样本归到距离最短的中心所在的类
- 3、利用均值等方法更新该类的中心值
- 4、对于多有的  $c$  个聚类中心, 如果利用 2, 3 的迭代法更新后, 值保持不变。则迭代结束, 否则继续迭代

流程图如下：





### 3.4 Baum-Welch 算法

输入：观测数据  $O=(o_1, o_2, \dots, o_T)$

输出：隐马尔可夫模型参数

(1) 初始化

对  $n=0$ ，选取：

$$a_{\{ij\}}^{(0)}, b_j(k)^{(0)}, \pi_i^{(0)},$$

得到模型：

$$\lambda^{(0)} = (A^{(0)}, B^{(0)}, \pi^{(0)}).$$

(2) 递推，对  $n=1, 2, \dots$ ,

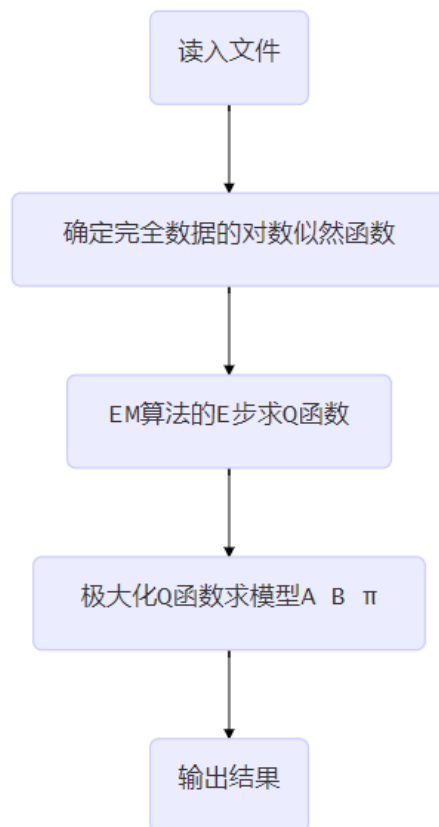
$$a_{ij}^{(n+1)} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$b_j^{(n+1)}(k) = \frac{\sum_{t=1, ot=vk}^{T-1} \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)}$$

$$\pi_i^{(n+1)} = \gamma_1(i)$$

(3) 终止，得到模型参数  $\lambda^{(n+1)} = (A^{(n+1)}, B^{(n+1)}, \pi^{(n+1)})$

流程图如下：



## 4. 实验流程

### 4.1 环境搭建

环境：win10 上的虚拟机 VMware Workstation 中，Spark/Scala + Hadoop 作为 MapReduce 计算框架的计算系统，Maven 为项目版本控制工具，实验 IDE 为 IntelliJ-IDEA(Java/Scala)、包 jahmm-0.6.1

语言：java、Scala

在 Spark 上可以运行多种语言，是一个很强大的软件。它的安装也会有很多困难。以下简单写一些我遇到的值得记录的地方：

1. 始终找不到 Hive 表的问题

原因：官方编译后的 Spark1.2.0+hadoop2.4 与 hadoop2.4.1 不匹配

解决方法有两个：a. 将 Hadoop2.4.1 替换为 2.4.0 版本；b. 重新编译 Spark1.2.0。

2. 部署 Spark 任务，不用拷贝整个架包，只需拷贝被修改的文件，然后在目标服务器上编译打包。

3. 打开 Hive 命令行客户端，观察输出日志是否有打印 “SLF4J: Found binding in [jar:file:/work/poa/hive-2.1.0-bin/lib/spark-assembly-1.6.2-hadoop2.6.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]” 来判断 hive 有没有绑定 spark

4. adoop 系统不可以在 root 用户下开启运行，需要在普通用户下运行 initial-all.sh 以启动 sshd 服务器。

5. 运行 sshd-server 后 X-server 无法定位显示服务到那个位置，原因是多用户登陆导致的，解决方法：在 root 用户下运行 \$xhost \indent+\$，无法运行 UI 程序的问题可以解决。

6. 在 Maven 版本控制方面需要做好 Scala 与 Spark 版本相容性控制，如 Spark 目前 release 版本为基于 Scala2.11 版本编译的 Jar 包，但是 Scala 目前的 release 为 2.12，Scala 不具有向下兼容的性质，因此自版本也要相容，Maven 拥有远程配置文件的库，spark 编译的版本多为基于副版本号但不明确到更详细的版本号，因此，会下载 spark-core-scala2.11，Scala 版本号精确如 2.11.3，但没有 2.11 版本的 core，此时就会出现 Scala 可编译但没有 spark 可用库的情况，

解决方案：选取较近版本号的 Scala-core 如 2.11.1，更改 maven 配置文件为 2.11，并保存到 Scala-core-2.11 的文件夹下，从而就有 scala 的 2.11 版本，并在本地安装 2.11.x 版本的 scala，即可以正确的编译 Scala App。

### 4.3 算法实现

#### 4.3.1 周期分量特征提取

我在文件 App.scala 实现这部分功能，使用的是 Scala 语言。Scala 语言是一个很强大的语言，而且很简洁。

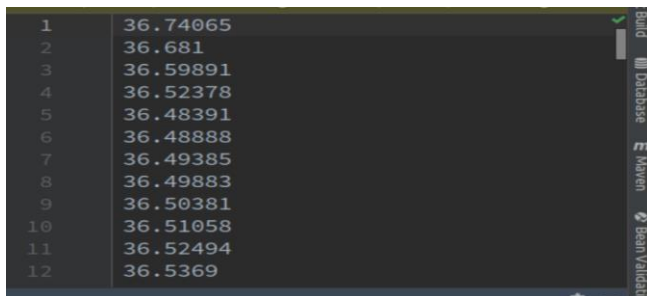
首先，读入文件，将读入的 rdd 用换行分割。再用 DenseVector() 函数将其

转换成向量便于对数据进行处理，利用 `DenseMatrix.tabulate(x,y)` 生成一个  $x*y$  的矩阵进行差分计算。

然后，利用 Spark 中的库 ML 进行线性拟合。其中的参数需要自己尝试，找到最优的。输出所有的拟合结果，并输出最小的那一行第一次出现的行数 L，即可判断周期是 L。

再调用 `produce.java`，以一行一周期的形式生成矩阵 B，并写入文件 `test.seq`。

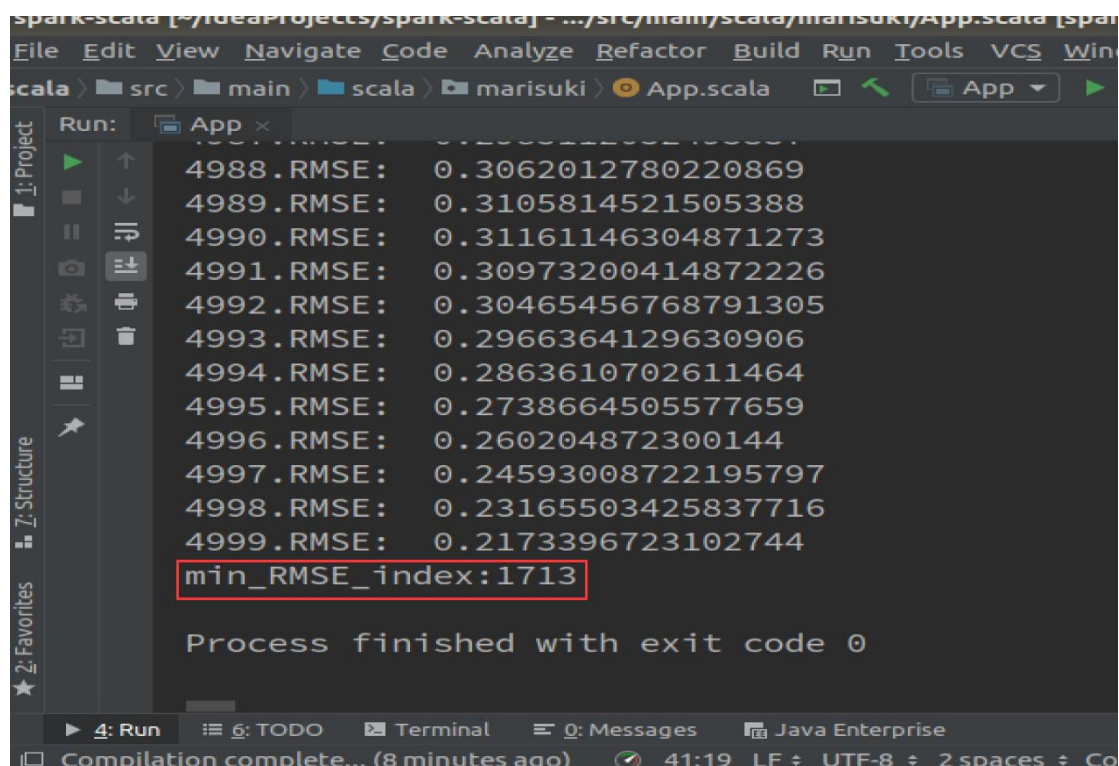
输入示例截图：`test.txt` 是读入的原始数据，下面是部分截图



```
1 36.74065
2 36.681
3 36.59891
4 36.52378
5 36.48391
6 36.48888
7 36.49385
8 36.49883
9 36.50381
10 36.51058
11 36.52494
12 36.5369
```

每一个数据以换行分割开。

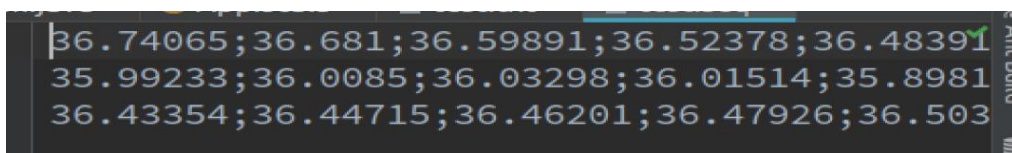
输出是每一行计算出的误差数以及最小误差的行数 L，以及矩阵 B（每行 L 个数据，存储在 `test.seq` 文件中）：



```
4988.RMSE: 0.3062012780220869
4989.RMSE: 0.3105814521505388
4990.RMSE: 0.31161146304871273
4991.RMSE: 0.30973200414872226
4992.RMSE: 0.30465456768791305
4993.RMSE: 0.2966364129630906
4994.RMSE: 0.2863610702611464
4995.RMSE: 0.2738664505577659
4996.RMSE: 0.260204872300144
4997.RMSE: 0.24593008722195797
4998.RMSE: 0.23165503425837716
4999.RMSE: 0.2173396723102744
min_RMSE_index:1713

Process finished with exit code 0
```

`test.seq`:



```
36.74065;36.681;36.59891;36.52378;36.48391
35.99233;36.0085;36.03298;36.01514;35.8981
36.43354;36.44715;36.46201;36.47926;36.503
```

#### 4.3.2 学习 (k-Means/Baum-Welch)

我主要尝试了两种学习方法：k-Means 和 Baum-Welch。将这两种算法进行对比。

将观察序列文件 test.seq 读进来后，首先调用 KMeansLearner() 函数初始生成 3 个状态，然后通过 learn() 算法进行 9 次迭代学习实现 k-Means 算法；利用 iterate() 来为 Baum-Welch 算法做准备，通过利用 4 次 iterate(hmm, seq) 函数进行 Baum-Welch 的实现。最后输出此时每个数据对应的状态。算法的主要实现过程在 jahmm 包里，实现思路是前文的算法设计中的内容，在这里不赘述了。

特别的是在进行 Baum-Welch 算法的实现时，因为我的数据是观察序列的方式，这不符合 Baum-Welch 的要求。所以我先用 Baum-Welch 中 KMeansLearner() 函数来简单生成一个概率矩阵，再通过 Baum-Welch 的多次迭代进行修正。

Baum-Welch 算法的输入是 xxx.hmm 文件输入，格式如下：

```
State
Pi 0.3
A 0.2 0.2 0.2 0.2 0.2
IntegerOPDF [0.5 0.5 ]

State
Pi 0.2
A 0.2 0.2 0.2 0.2 0.2
IntegerOPDF [0.5 0.5 ]

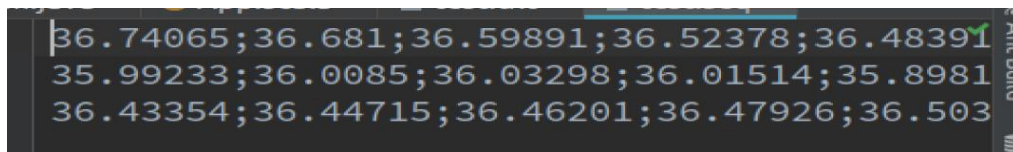
State
Pi 0.2
A 0.2 0.2 0.2 0.2 0.2
IntegerOPDF [0.5 0.5 ]

State
Pi 0.2
A 0.2 0.2 0.2 0.2 0.2
IntegerOPDF [0.5 0.5 ]
```

而 k-Means 算法是以观察序列以；隔离开即可，不区分换行。格式如下图：

```
obs11 ; obs12 ; obs13 ;
obs21 ; obs22 ; obs23 ; obs24 ;
```

在我的实验中，输入文件是 test.seqw 文件，内容是由 produce.java 生成的矩阵，一行 L 个数据（L 此时为 1713），数据之间以“；”隔开，部分截图如下：



k-Means 控制台输出结果：3 个状态，每个状态的概率以及转移概率，和高斯分布方差。

```

-----KML Start !-----
HMM with 3 state(s)

State 0
  Pi: 0.6666666666666666
  Aij: 0.998 0.002 0
  Opdf: Gaussian distribution --- Mean: 36.534 Variance

State 1
  Pi: 0.3333333333333333
  Aij: 0.003 0.996 0.001
  Opdf: Gaussian distribution --- Mean: 35.821 Variance

State 2
  Pi: 0.0
  Aij: 0 0.003 0.997
  Opdf: Gaussian distribution --- Mean: 34.949 Variance

-----KML finish !-----

```

Baum-Welch 控制台输出结果：3 个状态，每个状态的概率以及转移概率，和高斯分布方差。判断哪些数据是什么状态（如果数据和它前一个数据，则不打印；和前一个数据不相同，则打印它是第几个数据和是什么状态）

```

HMM with 3 state(s)

State 0
  Pi: 0.6670254454376476
  Aij: 0.996 0.004 0
  Opdf: Gaussian distribution --- Mean: 36.524 Variance

State 1
  Pi: 0.3329745545623506
  Aij: 0.005 0.994 0.002
  Opdf: Gaussian distribution --- Mean: 35.806 Variance

State 2
  Pi: 0.0
  Aij: 0 0.003 0.997
  Opdf: Gaussian distribution --- Mean: 34.948 Variance

```

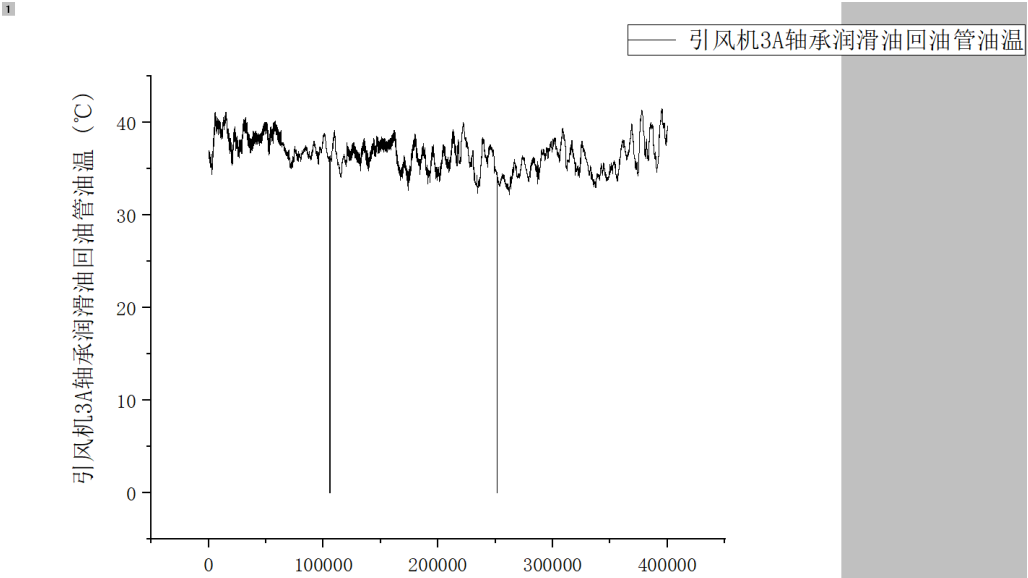
```
0 : 0
715 : 1
731 : 0
769 : 1
852 : 0
941 : 1
959 : 0
993 : 1
1466 : 2
1478 : 1
lnProbability:
-2.912795774786605
Probability:
0.05432364069973653
```

# 5. 实验结果

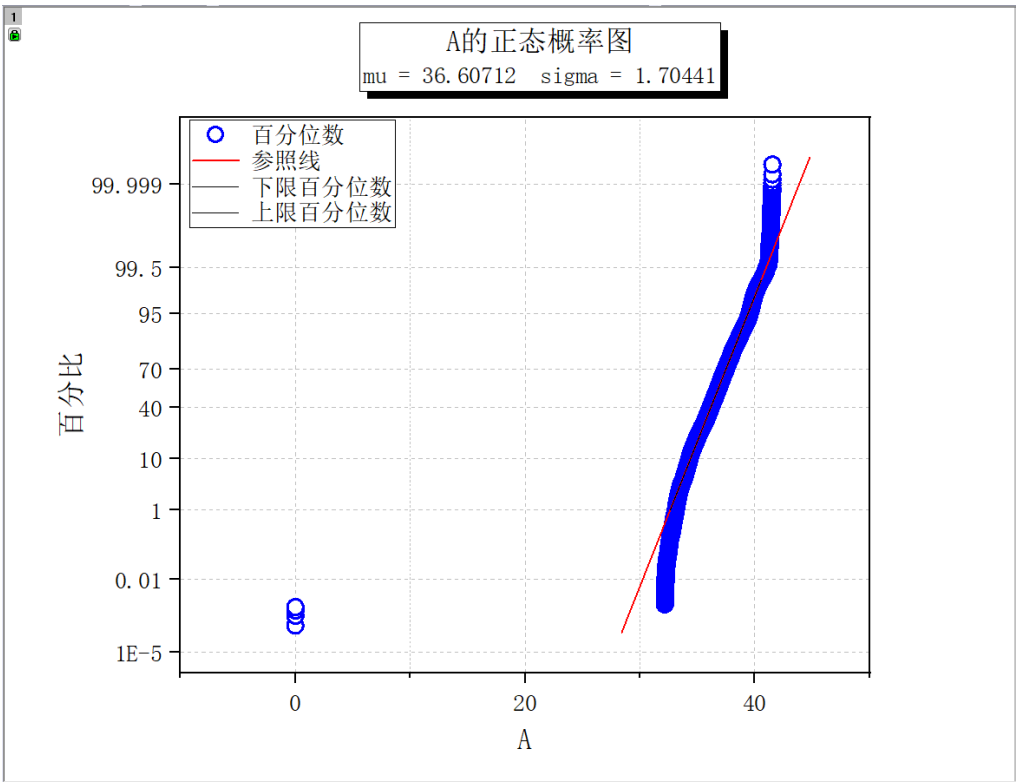
本次实验我选取了原数据一部分数据进行实验，即引风机 3A 轴承润滑油回油管油温这一部分数据。当代码运行 200000 个数据时，提取周期分量是 90 分钟左右，学习是 60 分钟左右。实验结果图如下：

原数据分布折线图：使用绘图软件 Origin 软件绘制。

引风机 3A 轴承润滑油回油管油温

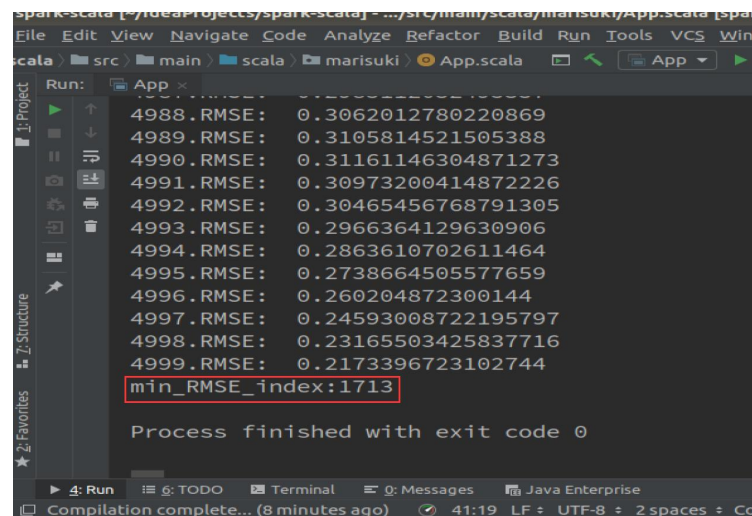


数据服从高斯分布（正态分布）：引风机 3A 轴承润滑油回油管油温





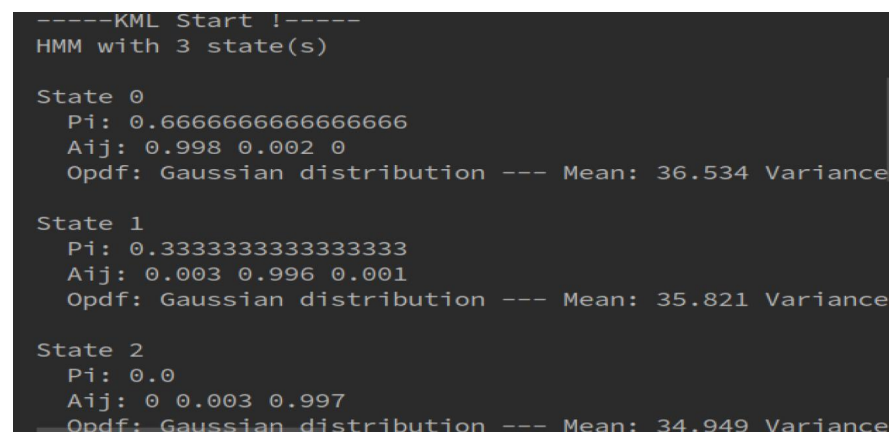
提取周期分量结果:



```
Run: App x
4988.RMSE: 0.3062012780220869
4989.RMSE: 0.3105814521505388
4990.RMSE: 0.31161146304871273
4991.RMSE: 0.30973200414872226
4992.RMSE: 0.30465456768791305
4993.RMSE: 0.2966364129630906
4994.RMSE: 0.2863610702611464
4995.RMSE: 0.2738664505577659
4996.RMSE: 0.260204872300144
4997.RMSE: 0.24593008722195797
4998.RMSE: 0.23165503425837716
4999.RMSE: 0.2173396723102744
min_RMSE_index:1713

Process finished with exit code 0
```

k-Means 算法结果:



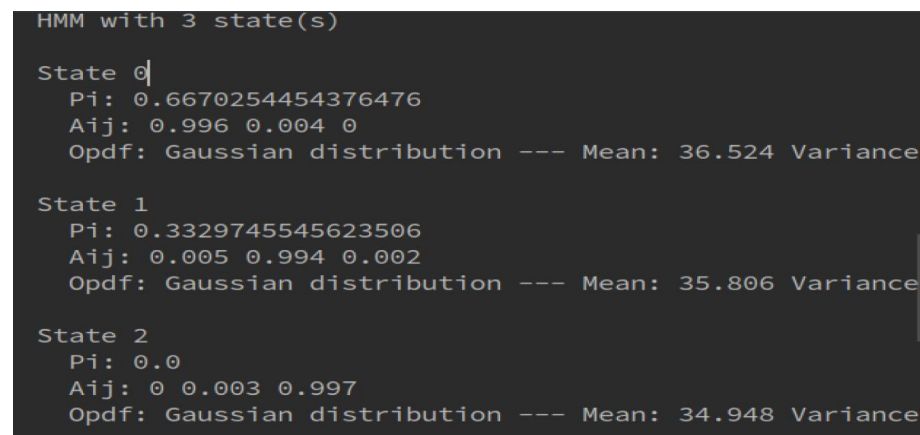
```
-----KML Start !-----
HMM with 3 state(s)

State 0
Pi: 0.6666666666666666
Aij: 0.998 0.002 0
Opdf: Gaussian distribution --- Mean: 36.534 Variance

State 1
Pi: 0.3333333333333333
Aij: 0.003 0.996 0.001
Opdf: Gaussian distribution --- Mean: 35.821 Variance

State 2
Pi: 0.0
Aij: 0 0.003 0.997
Opdf: Gaussian distribution --- Mean: 34.949 Variance
```

Baum-Welch 算法结果:



```
HMM with 3 state(s)

State 0
Pi: 0.6670254454376476
Aij: 0.996 0.004 0
Opdf: Gaussian distribution --- Mean: 36.524 Variance

State 1
Pi: 0.3329745545623506
Aij: 0.005 0.994 0.002
Opdf: Gaussian distribution --- Mean: 35.806 Variance

State 2
Pi: 0.0
Aij: 0 0.003 0.997
Opdf: Gaussian distribution --- Mean: 34.948 Variance
```

## 6. 结论

1. 通过本次实验可知, Baum-Welch 算法要比 k-Means 算法的结果更准确一些。因为数据本身就比较集中, 而 k-Means 算法对于越集中的数据, 区分效果越低。k-Means 算法适合分类比较明显的数据类型。虽然 Baum-Welch 算法结果更优, 但是对于本次实验, 状态的分类还需要找出更适合的算法或进行改进。

2. 在做周期分量提取的过程中, 用线性拟合可以得到比较好的结果, 但是由于引风机在实际过程中, 受到的因素很多, 其他发电设备的工作状态以及季节、温度都会影响引风机的数据, 所以线性拟合不会是最好的结果, 接下来要尝试用不同的拟合方法来计算, 从而得到更准确的周期数据。

3. 调参是很有必要的。此时的参数不一定是最优的, 因为我在调参的过程中发现, 不同参数对结果的影响真的很大, 当参数变化 0.1 时, 周期  $L$  的变化也可能会以 100 的数量级发生变化, 确定最优的参数尤其困难。

4. 通过阅读文献, 我了解到了引风机的工作是十分的复杂的。因为它是整个火力发电厂的一个部份之一, 它会受多种因素影响。

通过阅读国电泰州电厂 2X1000MW MCS 功能说明书(初版), 可以知道引风机用处有维持炉膛压力稳定、防止两台风机出力不平衡引起“倒风”现象、发生 MFT(超驰控制)跳炉时, 引风机的动叶开度减小; 当磨跳闸时, 也要关小引风机动叶等等。在不同的工作模式和条件下, 引风机都会有不同的动作, 这将引起不同数据的变化。在今后的研究中, 我们将重点研究不同数据之间的关联和变化曲线, 在将找到适合的一维模式识别的基础上进行多维时间序列的研究。