非侵入式电力负荷监测
HMM 问题的求解

王嘉诚 徐天乐 孙逸忻
电子信息工程提高 1701
U201713505

摘　要：本次实验通过对于用户用电数据的分析，建立基于隐马尔可夫问题（HMM）的模型，利用可以普遍性求解 HMM 问题的维特比算法（Viterbi algorithm），在不进入负荷内部的情况下，较为准确地计算出了各用电器的状态。

关键词：非侵入式，HMM，MMP 算法，基于事件，维特比算法

1 引言

当今社会，能源问题日益严重，如何尽最大可能做到减少能源的不必要的损耗成为一个至关重要的问题；在实现智能电网过程中，负荷监测是一个重要的组成部分；同时智能电网不仅会给国家带来收益，也会给人们带来便利。。传统的负荷可视化系统只能显示某 家庭总体能耗，而单个电器能耗无法得知；另一种采用分立式传感器监测电器的方法，虽然可以快速准确监控每台设备的运行状态，但系统必须为每个电器配备一个传感器，传感器网络复杂，成本高昂，不利于推广。非侵入式电荷监测技术虽然克服了这些问题，但是如何完善非侵入式电力负荷监测还是一个难题。目前国外针对非侵入的研究，文献[1]应用突变信号检测方法对暂态功率信息进行非侵入式电力负荷监测。 Norford 等人将非侵入式电力负荷监测到的暂态事件分类并与系统辨识技术结合到一起应用于电力系统监测与设备判别[2]。

下面将分为四个部分来介绍我们对于这个问题的研究。

2. 事件检测

事件检测的目的是从电表数据中找出用电器的开关时刻，并记录功率跳变值。事件检测的核心是有效跳变区间的搜索。在完成核心工作的同时，还需要消除数据中的噪声和冲激等影响。事件检测算法主要分为以下几个步骤：（1）对原始数据进行中值滤波，减少噪声干扰；（2）运用 MMP 算法搜索有效跳变区间；（3）对搜索后的结果进行稳态近似，方便后续算法处理；（4）检测冲激点并消除冲激影响。

2.1 中值滤波

原始的数据含有较多的噪声，如图 1-1 所示。我们希望对原始数据进行预先滤波处理，减少噪声的干扰。这里我们采用了一维中值滤波的方式。

中值滤波的原理如下：确定一个窗长 m，将某一点的函数值用该点的一个邻域中各点的中值代替，邻域的长度即为窗长 m。中值能有效消除原始数据中的尖锐冲激，但对于真正的功率跳变沿没有影响，这样就在尽量保证数据细节特征的条件下对噪声进行了有效抑制。

在 MATLAB 中，一维中值滤波可以采用 medfilt1（）函数实现，在本实验中我们取窗长为 5。处理后的数据图像如图 1-2 所示。

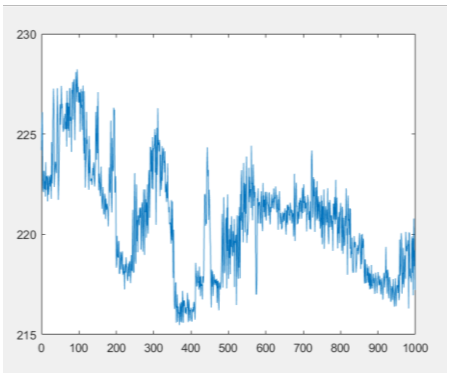

图 1-1

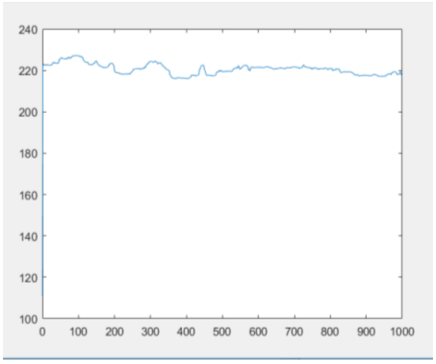

图 1-2

2.2 MMP 算法

MMP 算法[1]的全称为基于极大值极小值点的事件检测算法（event-detection algorithm based on maximum and minimum points），由 Tianqi Lu 于 2017 年提出。该算法的流程归结如下：（1）搜索极大值极小值点；（2）搜索有效跳变区间；（3）确定跳变起始/终止点。

基于数学知识，极大值点和极小值点分别满足公式（1）、（2），可以此为依据搜索极大值点和极小值点。

$$\begin{cases} x_{n-1} \le x_n \\ x_n \ge x_{n+1} \end{cases} \tag{1}$$

$$\begin{cases} x_{n-1} \ge x_n \\ x_n \le x_{n+1} \end{cases} \tag{2}$$

接下来是搜索有效跳变区间，即确定哪些跳变是由真正的用电器开关造成的，而不是由于噪声干扰造成的。在这里我们设定一个噪声阈值th，该阈值大于噪声所造成的跳变值，且小于任一用电器的功率跳变值（包括多个用电器开关的组合功率跳变值）。这样，通过比较相邻极值点的功率差，我们就可以确定有效跳变区间。假设x_n和x_{n+1}为两相邻的极值点，当x_n为极小值点时，可根据公式（3）判定有效上升区间；当x_n为极大值点时，可根据公式（4）判定有效下降区间。本实验中设置$th = 80W$。

$$x_n + th \le x_{n+1} \tag{3}$$

$$x_n \ge x_{n+1} + th \tag{4}$$

仅仅得到一个跳变区间不足以满足实际要求，我们需要确定跳变的起始/终止点，即确定用电器开/关的具体时刻。具体的方法为，在第二部确定的有效跳变区间中，进行正向搜索与反向搜索。正向搜索是从区间左端点开始逐点向右搜索，直到满足公式（5），停止，获得跳变起始点。同样的，反向搜索从区间右端点开始逐点向左搜索，最终得到跳变终止点。

$$\begin{cases} x_{t+1} - x_t \ge th\,, rising\ interval \\ x_t - x_{t+1} \ge th\,, falling\ interval \end{cases} \tag{5}$$

运用 MMP 算法，我们可以确定用电器开关的时刻及功率跳变情况，初步完成了事件检测工作。

2.3 稳态近似

为了便于后续模型的建立与求解，我们在这里需要对 MMP 算法的结果作稳态近似。所谓稳态近似，即忽略两次跳变之间由于噪声所造成的的功率波动，而用一个稳态值（定值）来代替这一时间段的功率值。

在本实验中，我们采用时间段的功率均值作为该时间段的稳态值。在 MATLAB 中，可以用 mean（）函数计算均值。处理后的数据图像如图 2 所示。

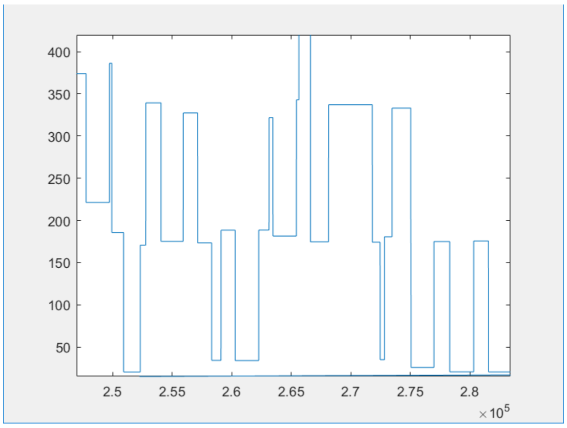

图 2

2.4 消除冲激

数据中的冲激对于事件检测与分析显然是不利的，我们应当极力消除冲激的影响。消除冲激的大体思路是：（1）搜索冲激；（2）判断冲激类型；（3）抹平冲激。

可以利用公式（6）作为冲激的搜索指标。式中，x_{t-1}，x_t，x_{t+1}为三个相邻时刻，th'为冲激阈值。冲激阈值th'可适当大于噪声阈值th，本实验中设置$th' = 100W$。

$$
\begin{cases} x_{t-1} + th' \le x_t \\ x_t \ge x_{t+1} + th' \end{cases} \tag{6}
$$

经过数据分析，我们发现了两种类型的冲激，分别如图 3-1，3-2 所示。其中，图 3-1 中的冲激是有效的，确实代表用电器的开关，这类冲激不应该抹平。图 3-2 中的冲激是无效的，会影响跳变功率的判断，应该抹平。

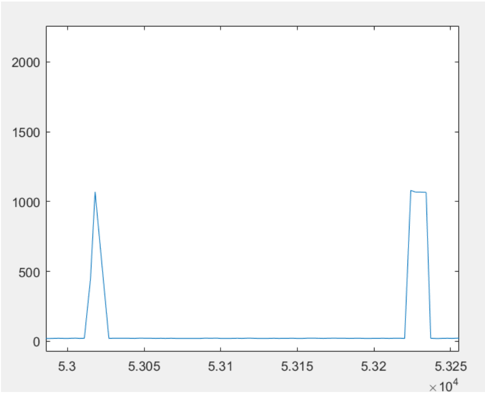

图 3-1

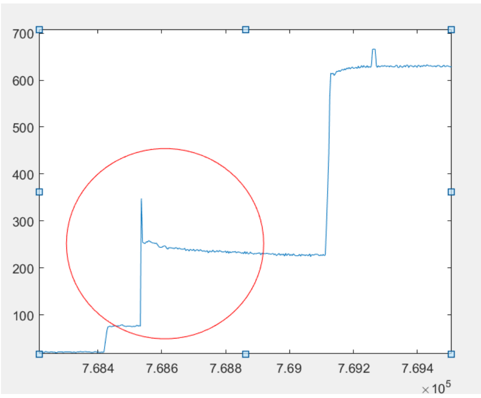

图 3-2

基于上述分析，我们根据公式（7）来识别无效冲激。式中，x_n，x_{n+1}分别为冲激前后的稳态值，当两者的插值大于噪声阈值th时，我们将他识别第二种冲激情况，即无效冲激。

$$
x_n + th \le x_{n+1} \tag{7}
$$

在识别出无效冲激后，我们需要将其抹平，来消除它的影响。抹平冲激的方法为，将冲激点处的功率值用冲击前后稳态功率的均值来代替，如公式（8）所示，其中x_i为冲激点功率值，x_n，x_{n+1}分别为冲激前后的稳态值。

$$
x_i = \frac{x_n + x_{n+1}}{2} \tag{8}
$$

图 3-2 消除冲激后的结果如图 3-3 所示。

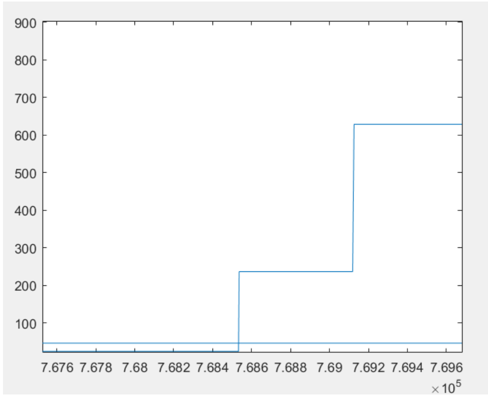

图 3-3

2.5 事件检测结果

事件检测算法的结果如图 4 和表 1 所示。其中，图 4 为原始数据图像，表 1 为事件检测矩阵（表 1 中每两列表示一次跳变，行 1 为跳变时刻，行 2 为上升/下降标识，行 3 为功率值，行 4 计算该跳变的功率变化）。

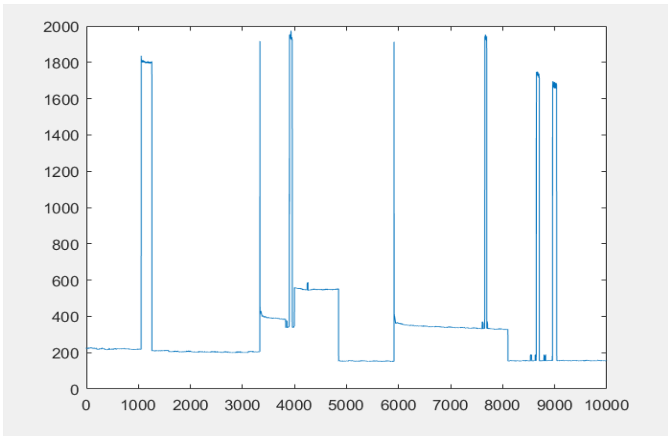

图 4

	1	2	3	4	5	6	7	8	9	10
1	1051	1054	1258	1260	3334	3335	3335	3337	3900	3901
2	1	0	2	0	1	0	2	0	1	0
3	220.5866	1.8019e+...	1.8019e+...	205.2828	205.2828	1.9164e+...	1.9164e+...	388.2723	388.2723	1.2692e+...
4	1.5813e+...	0	-1.5966e...	0	1.7111e+...	0	-1.5281e...	0	880.9377	0

	19	20	21	22	23	24	25	26	27	28
1	4846	4847	5911	5912	5912	5914	7653	7654	7654	7655
2	2	0	1	0	2	0	1	0	2	0
3	549.9502	153.8215	153.8215	1.9132e+...	1.9132e+...	344.4006	344.4006	1.0452e+...	1.0452e+...	949.3600
4	-396.1287	0	1.7594e+...	0	-1.5688e...	0	700.8394	0	-95.8800	0

表 1

从上述检测结果可以看出，事件检测算法具有较高的精度和准确度，能满足实际应用要求。

3 基于状态简约的 vertibi 算法

将整段时间划分为各个稳态区间，根据各区间内的均值和跳变值来求各区间状态。

3.1 状态编码

将用电器状态进行二进制编码再转化为 10 进制，作为状态的编码（例如 3 个用电器，那么状态 0 代表全部都关着，状态 1（001）代表 3 号用电器开，状态 2（010）代表 2 号用电器开）。

3.2 观测矩阵 B(i,t)

B(i,t)= $\{\boldsymbol{b_{it}}\}$, $\boldsymbol{b_{it}=p(X_t=q_i|O_t=\mathbf{O(t)})}$, 是指 t 时刻用电器状态为状态 i 的概率。传统意义的 B=$\{\boldsymbol{b_{ik}}\}$，$\boldsymbol{b_{ik}=p(O_t=v_k|X_t=q_i)}$指的是在状态为 i 时观测值为$\boldsymbol{v_k}$的概率。获得 B(i,t) 的方法：假设功率服从正态分布，将每个状态的均值和方差算出来，得到每个状态功率的正态分布概率密度函数，正态分布概率密度函数在均值取值最大，用概率密度函数在 O(t)(t 时刻的总功率值)的取值除以它在均值处的取值可以得到一个在 0，1 之间的数，把这个作为功率观测值为 O(t)时状态为 i 的概率$\boldsymbol{b_{it}}$。可以很直观地想，观测到的功率越接近状态 i 的功率，此时刻状态越可能是 i。

3.3 状态转移矩阵 A(i,j,t)

A(i,j,t)=$\{a_{ij}\}$, $a_{ij}=p(X_{t+1}=q_j|X_t=q_i,\mathrm{T}=\mathrm{t})$,表示 t 时刻从状态 i 转移至状态 j 的概率。获得方法：各个状态的正态分布均值相减，方差相加，可以获得状态转移时功率跳变值的正态分布的均值与方差，然后类似计算 B(t)的做法，用概率密度函数在 O(t)-O(t-1) 处的值除以在均值处的值即可获得转移概率。

三维的数组太大，于是我把跳变值的限制放在 vertibi 算法里面考虑，预先产生一个 A(i,j)限制至多两个用电器开关。

获得 A(i,j)方法：对 i,j 进行二进制的位异或，获得的值若是 2 的次幂则 A(i,j)=1，若是 2 的 a 次幂加 2 的 b 次幂则概率 A(i,j)=0.02，其他情况 A(i,j)=0。

4 实验结果

5 实验总结

5.1 小结

本次建模利用 MMP 算法，维特比算法，重点在于从用户电荷数据中提取事件以及从各设备中提取特征，寻找到稳定特征后，进行事件比对，基本上实现了只利用总功率求解用电器状态的问题，模型准确度较高，较好的完成了任务。但是对于情况极度复杂的现实生活，还是需要建立在广泛数据上的更多的研究。

小组分工如下：

孙逸忻负责分析和处理数据，编写事件检测算法，获得总功率的开关时间点及跳变功率，并制作实验的验证集（分用电器的开关时间表）

王嘉诚

徐天乐负责编写 Vertibi 算法，根据处理后的数据对各时间段的状态进行预测，并将结果和验证集的结果进行比对获得准确率召回率以及 F-measure。

5.2 不足

事件检测算法：

1. 有的用电器包含待机功率，此功率值较小且

跳变频繁，对于此类功率跳变的检测结果不太理想。

2. 算法的条件判定标准较为单一，如果存在某些尚未被发现的特殊情况，算法无法处理。
3. 稳态近似的算法较为粗略，直接取均值的方式在某些情况下会造成较大误差。

Vertibi 算法：

1）初始状态

由于无法判断初始状态，所以在求解模型的时候，可能与真实结果有较大偏差

2）模型参数

对于状态转移矩阵的处理不是很完善

3）分辨度低

对于两种功率相近的状态，无法有效区分

4）没有完全利用真实情况

例如没有利用用户习惯设计状态转移矩阵，比如早上起床先开灯

5.3 改进

1. 针对冰箱等多档位用电器，设定专门的事件检测算法
2. 更优的数据滤波方法和去噪手段，减少噪声干扰稳态近似的算法较为粗略，直接取均值的方式在某些情况下会造成较大误差。
3. 调整模型算法，减少中间参数，提升其运算速度
4. 使模型参数可调，提升判断准确度

参考文献：

[1] Tianqi Lu et al.，“An Event-Based Nonintrusive Load Monitoring Approach: Using the Simplified Viterbi Algorithm”，IEEE CS，2017，1536-1268

[2] 牛卢璐，贾宏杰.一种适用于非侵入式负荷监测的暂态事件检测 算法[J].电力系统自动化，2011， 35（9）：30-35.

[3] Norford L K，Leeb S B，Non-intrusive electrical load moni－ toring in commercial buildings based on steady-state and tran－ sient load-detection algorithms [J]. Energy & Buildings，1995， 24 （1）：51-64.

程序源码：

```
%%事件检测算法%%

data_power_0 = power{1,3}+power{1,4}+power{1,5}+power{1,6}+power{1,8}+power{1,9}+power{1,10}+power{1,11};
data_time = t{1,3};
%data_power = medfilt1(data_power_0, 5);

y = data_power_0(1:length(data_power_0),1);
x = data_time(1:length(data_time),1);
figure(1)
plot(x,y)

%threshold value
th = 80;


%STEP1
%寻找极值点
A = zeros(1,length(data_power_0));
B = zeros(1,length(data_power_0));
j=1;
for i=2:length(data_power_0)-1
    if ( (y(i-1)<=y(i) && y(i)>y(i+1)) || (y(i-1)<y(i) && y(i)>=y(i+1)))
        A(i) = 2;
        B(j) = i;
        j = j+1;
    else if ( (y(i-1)>=y(i) && y(i)<y(i+1)) || (y(i-1)>y(i) && y(i)<=y(i+1)) )
        A(i) = 1;
        B(j) = i;
        j = j+1;
```

```
        end
    end
end


%STEP2
%寻找有效跳变区间
C = zeros(1,length(data_power_0));
D = zeros(1,length(data_power_0));
m=1;
for k = 1:length(data_power_0)-1
    if (B(k+1)==0)
        break;
    end
    if ( A(B(k))==1 && y(B(k))+th<y(B(k+1)) )
        C(B(k)) = 1;
        C(B(k+1)) = 2;
        D(m) = B(k);
        D(m+1) = B(k+1);
        m = m+2;
    else if ( A(B(k))==2 && y(B(k))>y(B(k+1))+th )
            C(B(k)) = 2;
            C(B(k+1)) = 1;
            D(m) = B(k);
            D(m+1) = B(k+1);
            m = m+2;
        end
    end
end


%STEP3
%确定跳变起始/终止点
F_0 = zeros(5,10000);
q=1;
for p=1:2:length(data_power_0)-1
    if (D(p)==0)
        break;
    end
    diff = abs(y(D(p+1))-y(D(p)));
    if ( C(D(p))==1 )
        for r=0:(D(p+1)-D(p))-1
            if ( y(D(p)+r)+th < y(D(p)+r+1) && y(D(p)+r)+1/6*diff < y(D(p)+r+1) )
                break
            end
        end
        for s=0:(D(p+1)-D(p))-1
            if ( y(D(p+1)-s-1)+th < y(D(p+1)-s) && y(D(p+1)-s-1)+1/6*diff < y(D(p+1)-s))
                break
            end
        end
        if ( r+s<(D(p+1)-D(p)) )
            F_0(5,q) = D(p)+r;
            F_0(5,q+1) = D(p+1)-s;
            F_0(1,q) = x(D(p)+r);
            F_0(1,q+1) = x(D(p+1)-s);
            F_0(2,q) = 1;
            F_0(3,q) = y(D(p)+r);
            F_0(3,q+1) = y(D(p+1)-s);
            q = q+2;
        end
    else if ( C(D(p))==2 )
            for r=0:(D(p+1)-D(p))-1
                if ( y(D(p)+r) > y(D(p)+r+1)+th  && y(D(p)+r) > y(D(p)+r+1)+1/6*diff )
                    break
                end
            end
            for s=0:(D(p+1)-D(p))-1
                if ( y(D(p+1)-s-1) > y(D(p+1)-s)+th && y(D(p+1)-s-1) > y(D(p+1)-s)+1/6*diff )
                    break
                end
            end
            if ( r+s<(D(p+1)-D(p)) )
                F_0(5,q) = D(p)+r;
                F_0(5,q+1) = D(p+1)-s;
                F_0(1,q) = x(D(p)+r);
                F_0(1,q+1) = x(D(p+1)-s);
                F_0(2,q) = 2;
                F_0(3,q) = y(D(p)+r);
```

```
                F_0(3,q+1) = y(D(p+1)-s);
                q = q+2;
            end
        end
    end
end


%STEP4
%稳态近似
F_0(3,1) = mean( y(1:F_0(5,1)) );
for w=1:2:(length(F_0(1,:))-2)
    if ( F_0(1,w+2)==0 )
        break
    end
    temp = mean( y(F_0(5,w+1):F_0(5,w+2)) );
    F_0(3,w+1) = temp;
    F_0(3,w+2) = temp;
    F_0(4,w) = F_0(3,w+1)-F_0(3,w);
end
F_0(3,w+1) = mean( y(F_0(5,w+1):length(data_power_0)) );
F_0(4,w) = F_0(3,w+1)-F_0(3,w);


%STEP5
%消除冲激
for ii=2:length(F_0(1,:))-1
    if ( F_0(1,ii+1)==0 )
        break;
    end
    if ( F_0(1,ii-1)==F_0(1,ii) && F_0(3,ii+1)>F_0(3,ii-2)+th )
        F_0(3,ii-1) = F_0(3,ii+1);
        F_0(4,ii-2) = F_0(3,ii-1)-F_0(3,ii-2);
        F_0(:,ii) = -1;
        F_0(:,ii+1) = -2;
    end
end

F_0( :,all(F_0==-1, 1) ) = [];
F_0( :,all(F_0==-2, 1) ) = [];


figure(2)
plot(F_0(1,:),F_0(3,:))

save('F_0')


%%维特比算法 %%

function A=transferO


%获得各稳态区间的起始时间和结束时间，方便后面还原q(t)

%time：各个稳态区间的起始时间和结束时间

%O(t)为观测到的功率值，t为稳态序号


load('F_0.mat','F_0');


A=F_0(3,:);


[O,iabefore,ic]=unique(A,'stable');

[~,ialast,ic]=unique(A,'last');

ialast=sort(ialast);


L=length(iabefore);

time=zeros(L,2);
```

```
for i=1:L
    time(i,1)=F_0(1,iabefore(i));
    time(i,2)=F_0(1,ialast(i));
end
save time.mat time ; %状态跳变时间点
save iabelast.mat ialast ;
save iabefore.mat iabefore ;
save O.mat O ; %各稳态观测值

function y = getcharacter(k)

%获得各个状态的均值mu(i),各个状态的方差sigma(i),i为状态二进制编码
%k为用电器数量

mu1(1)= 149.7894;%4 lighting
mu1(2)= 163.5772;%9 refrigerator
mu1(3)= 368.8275;%11 disposal
mu1(4)= 406.6092;%5 stove
mu1(5)= 775.6247;%3 kitchen_outlets
mu1(6)= 1056.3; %8 kitchen_outlets
mu1(7)= 1877.2;%6 microwave
mu1(8)= 1198.0;%10 dishwaser noise

% mu1(3)= 248.9097; 10 dishwaser
% mu1(10)= 1983.2; 9 refrigerator noise

sigma1(1)=287.8157;%287.8157 80
sigma1(2)=67.0947;%67.0947 30
sigma1(3)=103.6625;%103.6625 60
sigma1(4)=16.3073;%70.3073 33
sigma1(5)=127.4912;%127.4912 55
sigma1(6)=77;%77 35
sigma1(7)=1100;%1100 90
sigma1(8)=337;%337 157

% sigma1(3)=89.4009;%89.4009 80
% sigma1(10)=9468.7;%9468.7 500

N=2^k;

mu=zeros(N,1);
sigma=zeros(N,1);
```

```
for i=0:N-1
    for j = 1:k
        mu(i+1)=mu(i+1)+bitget(i,j)*mu1(j);
        sigma(i+1) = sigma(i+1)+bitget(i,j)*sigma1(j);
    end
end

sigma(1)=20;

save mu.mat mu ;
save sigma.mat sigma;

function A = getA(k)

%获得"转移概率矩阵"A(i,j)(i状态转移到j状态的概率)
%若两个状态之间有一个用电器的状态不同，则概率为1，若有两个不同，概率为0.02

N=2^k; %状态数

A=zeros(N,N);

for i=0:N-1
    for j=0:N-1

    for m=0:k-1
        if abs(bitxor(i,j))==2^m
            A(i+1,j+1)=1;
            break
        else
        for n=0:m-1
            if (abs(bitxor(i,j))==2^m+2^n)||(abs(bitxor(i,j))==2^m-2^n)
                A(i+1,j+1)=0.02;
                break
            else
                A(i+1,j+1)=0;
            end
        end
        end
    end

    end
end
save A.mat A
```

```
function B = getB(k)

%获得“观测概率矩阵”B(i,t)(t时刻稳态对应状态i的概率)

load('mu.mat','mu');
load('sigma.mat','sigma');
load('O.mat','O');
T=length(O);

N=2^k;
% a=zeros(N);
% b=zeros(N);
B=zeros(N,T);

for t=1:T
%     a(i)=mu(i)-500;
%     b(i)=mu(i)+500;

% [~,m] = min(abs(mu-O(t)));
    for i=1:N
       P=normpdf(mu(i),mu(i),sigma(i));

%         if abs(O(t)-mu(i)) >= 200
%             B(i,t)=0.1;
%         else
          B(i,t)=normpdf(O(t),mu(i),sigma(i))*P;
%         end

    end
%     B(m,t)=B(m,t)*10;
end
save B.mat B

function [delta,psi,q,pprob] = Viterbi(k)

%获得观测到的状态序列q(t),t为稳态序号，q(t)为状态编号
%k为用电器数量

load('B.mat','B');
load('A.mat','A');
load('sigmad.mat','sigmad');
load('mud.mat','mud');
load('O.mat','O');
```

```matlab
load('P.mat','P');

%T: 稳态总数
%mu(i):状态i均值
%sigma(i):状态i方差
%O(t):t时刻观测到的功率值

T=length(O);
N=2^k; %×´Ì¬Êý
delta=zeros(T,N);
psi=zeros(T,N);
pi=ones(N,1)/N;

C=zeros(N,N);

%1. Initialization

for i = 1 : N
    delta(1,i) = pi(i) * B(i,1);  %//O[1]-->1
    psi(1,i) = 0;
end

%2. Recursion
for t = 2:T
    for j = 1:N
        maxval = 0.0;
        maxvalind = 1;
        for i = 1:N
%             if (c(i,j)<O(t)-O(t-1))&&(O(t)-O(t-1)<d(i,j))
%             C(i,j)=0.9*A(i,j);
%             else
%             C(i,j)=0.1*A(i,j);
%             end

            if (O(t)-O(t-1))*(mud(i,j)) < 0
                C(i,j)=0;
            else
                C(i,j)=normpdf(O(t)-O(t-1),mud(i,j),sigmad(i,j))*A(i,j)/P(i,j);
            end
                val = delta(t-1,i)*C(i,j);
            if (val > maxval)
                maxval = val;
                maxvalind = i;
                delta(t,j) = maxval*B(j,t);
```

```
            psi(t,j) = maxvalind;
        end
    end
end
if delta(t,1)<10^-50 && delta(t,1)>0
    for j=1:N
        delta(t,j)=delta(t,j)*10^50;
    end
end
end

%3. Termination
pprob = 0.0;
q(T) = 1;
for i = 1:N
    if (delta(T,i) > pprob)%获得T时刻最大概率路径的概率以及结点
        pprob = delta(T,i);
        q(T) = i;
    end
end

for t = T-1:-1:1
    q(t) = psi(t+1,q(t+1));
end
save Q.mat q ;

function y = DecomposeQ(m)

%该函数将观测到的矩阵q(t)分解，获得各个用电器在各个稳态区间的功率值
%Onew每一行代表一个稳态区间，第一列是总功率预测值，第2~m+1列为各个用电器的功率预测值
load('q.mat','q');
load('F_0.mat','F_0');
load('O.mat','O');
load('time.mat','time');

mu1(1)= 149.7894;%4 lighting
mu1(2)= 163.5772;%9 refrigerator
mu1(3)= 368.8275;%11 disposal
mu1(4)= 409.3255;%5 stove
mu1(5)= 775.6247;%3 kitchen_outlets
mu1(6)= 1056.3; %8 Hkitchen_outlets
mu1(7)= 1877.2;%6 microwave
mu1(8)= 1198.0;%10 dishwaser noise
% mu1(3)= 248.9097;%10 dishwaser
```

```
% mu1(10)= 1983.2;%9 refrigerator noise

T=F_0(1,4242);
L=length(q);
power=zeros(L,m);
Onew=zeros(T,m+1);
s=zeros(m);

for i=1:L
    for j=1:m
        s(j)=bitget(q(i)-1,j);
        power(i,j)=power(i,j)+s(j)*mu1(j);
    end
end

for i=1:L
    for j=1:m
        for k=time(i,1):time(i,2)
            Onew(k,j+1)=power(i,j);
        end
    end
end
for k=1:T
    for j=1:m
    Onew(k,1)=Onew(k,1)+Onew(k,j);
    end
end

save Onew.mat Onew ;

for i=2:m+1
    Judgestate(i);
end


function y=Judgestate(n)

%获得各个用电器的上升沿和下降沿
%输出State矩阵，第一行是时间，第二行是状态，1代表上升沿，-1代表下降沿
%n为用电器编号，n=1是总功率，其他见下
%2 lighting
%3 refrigerator
%4 disposal
%5 stove
```

```
%6 kitchen_outlets
%7 kitchen_outlets
%8 microwave
%9 dishwaser

load('Onew.mat','Onew');

L=length(Onew(:,1));
State=zeros(2,L);

k=1;

for t=1:L-1
    if Onew(t,n)~=0 && Onew(t+1,n)==0
        State(1,k)=-1;
        State(2,k)=t;
        k=k+1;
    else
        if Onew(t,n)==0 &&Onew(t+1,n)~=0
        State(1,k)=1;
        State(2,k)=t;
        k=k+1;
        end
    end
end

switch(n)
    case 1
        save TotalState.mat State
    case 2
        save lightingState.mat State;
    case 3
        save refrigeratorState.mat State;
    case 4
        save disposalState.mat State;
    case 5
        save stoveState.mat State;
    case 6
        save kitchen_outletsState.mat State;
    case 7
        save Hkitchen_outletsState.mat State;
    case 8
        save microwaveState.mat State;
    case 9
```

```
        save dishwaserState.mat State;
    end
```