统计信号处理基础 第 06 次作业

许凌玮 2018011084

1. 目标

给定特征方程

$$\lambda^3 + a\lambda^2 + b\lambda + c = 0$$

特征方程的系数已知，且$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$。令

$$p_i = \frac{\lambda_i}{\lambda_1 + \lambda_2 + \lambda_3}$$

试利用上述特征多项式的系数近似表示下面的表达式

$$H = -\sum_{i=1}^{3} p_i \log_3 p_i$$

2. 分析与建模

首先推导韦达定理。特征方程用特征根表示，可推得

$$(\lambda-\lambda_1)(\lambda-\lambda_2)(\lambda-\lambda_3)=0 \quad \Rightarrow \quad \lambda^3-(\lambda_1+\lambda_2+\lambda_3)\lambda^2+(\lambda_1\lambda_2+\lambda_2\lambda_3+\lambda_3\lambda_1)\lambda-\lambda_1\lambda_2\lambda_3=0$$

$$\Rightarrow \quad \begin{cases} a=-(\lambda_1+\lambda_2+\lambda_3) \\ b=\lambda_1\lambda_2+\lambda_2\lambda_3+\lambda_3\lambda_1 \\ c=-\lambda_1\lambda_2\lambda_3 \end{cases}$$

由于

$$p_i = \frac{\lambda_i}{\lambda_1+\lambda_2+\lambda_3} = -\frac{\lambda_i}{a} \quad (i=1,2,3)$$

因此可用p_i来替换λ_i，得到关于p的特征方程

$$p^3 - p^2 + \frac{b}{a^2}p - \frac{c}{a^3} = 0$$

它满足$1 \geq p_1 \geq p_2 \geq p_3 \geq 0$且$p_1+p_2+p_3=1$。

由此可见系数的自由度只剩两个了（还有一个自由度被p_i的和为 1 约束住了），可以将如下α、β作为观测值

$$\alpha = \frac{b}{a^2}, \qquad \beta = \frac{c}{a^3}$$

拟合的目标为

$$H = -\sum_{i=1}^{3} p_i \log_3 p_i$$

由于$p_i (i=1,2,3)$与估计量H的任何先验均未知，因此可以采用最小二乘估计进行拟合。依据多元函数的泰勒展开，我们可以将H用如下n阶二元多项式逼近

$$f(\alpha,\beta) = \sum_{i=0}^{n}\sum_{j=0}^{i} C_{ij}\alpha^{i-j}\beta^j$$

其中系数C_{ij}可构成如下$l=\frac{1}{2}(n+1)(n+2)$维的系数矩阵

$$\boldsymbol{w} = [C_{00}, C_{10}, C_{11}, \dots, C_{ij}, \dots, C_{nn}]^T \in \mathbb{R}^{l\times 1} \quad (i=0,\dots,n;\ \ j=0,\dots,i)$$

将每次的观测值记为

$$\boldsymbol{q}=[\alpha,\beta,\alpha^2,\alpha\beta,\beta^2,\dots,\alpha^{i-j}\beta^j,\dots,\alpha^n\beta^n]\in\mathbb{R}^{1\times l}\quad(i=0,\dots,n;\ \ j=0,\dots,i)$$

记第i次观测的噪声为n_i（对应模型拟合的误差），则相应的观测值为

$$H_i=\boldsymbol{q_i}\boldsymbol{w}+n_i\quad(i=1,2,\dots,N)$$

则对于全部观测，可记为

$$\boldsymbol{H}=\boldsymbol{Q}\boldsymbol{w}+\boldsymbol{n}\quad(i=1,2,\dots,N)$$

其中

$$\boldsymbol{H}=[H_1,H_2,\dots,H_n]^T\in\mathbb{R}^{N\times 1}$$

$$\boldsymbol{Q}=[\boldsymbol{q_1}^T,\boldsymbol{q_2}^T,\dots,\boldsymbol{q_N}^T]^T\in\mathbb{R}^{N\times l}$$

利用最小二乘法可以对系数矩阵$\boldsymbol{w}$进行估计，最小化如下均方误差（MSE）代价函数

$$J(\boldsymbol{w})=\|\ \boldsymbol{H}-\boldsymbol{Q}\boldsymbol{w}\ \|^2=(\boldsymbol{H}-\boldsymbol{Q}\boldsymbol{w})^T(\boldsymbol{H}-\boldsymbol{Q}\boldsymbol{w})$$

可以导出计算式如下

$$\boldsymbol{w}=(\boldsymbol{Q}^T\boldsymbol{Q})^{-1}\boldsymbol{Q}^T\boldsymbol{H}$$

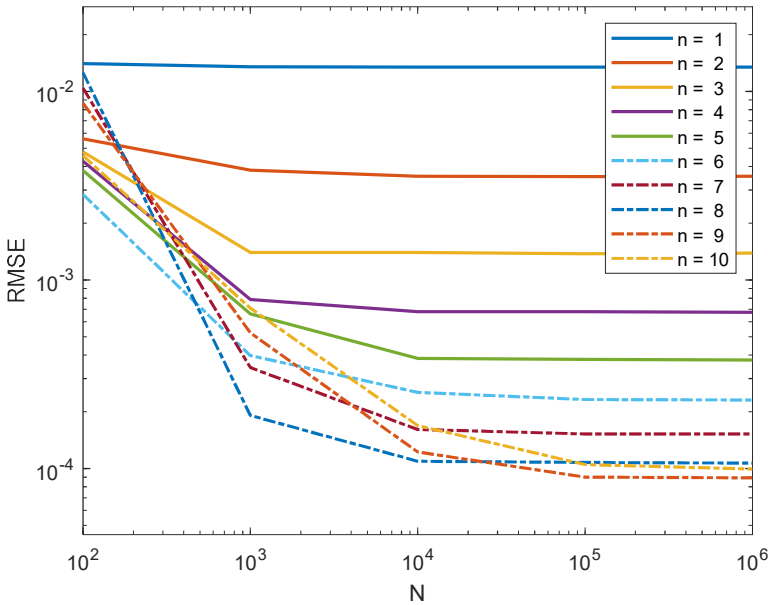
3. 模型训练与结果

为了得到合适的系数矩阵$\boldsymbol{w}$，使用 MATLAB 编程，生成N个随机样本进行最小二乘估计。相应的代码写在了 myLSE.m 中。

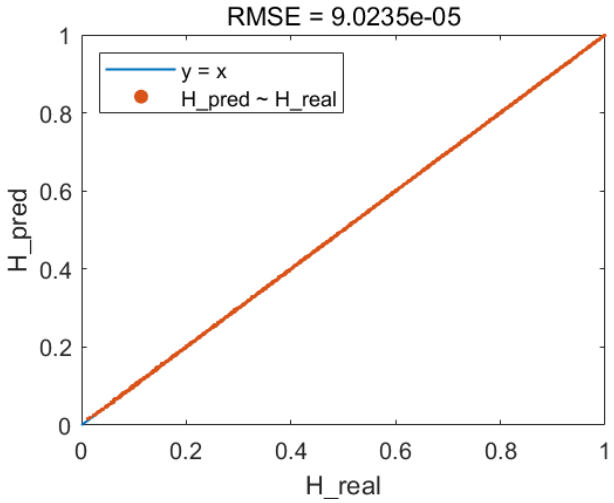
在生成样本时，为了保证$\lambda_1\geq\lambda_2\geq\lambda_3\geq 0$且为实数，采用随机生成$\lambda_1,\lambda_2,\lambda_3$然后用韦达公式反推$a,b,c$，进而计算得到相应的$\alpha,\beta$生成观测值，并计算得到相应的用于生成$p_1,p_2,p_3$真值。

首先采用不同的阶数n以及数据规模N训练模型，作出损失曲线，选取合适的n、N值。

损失函数取 RMSE（与上面的 MSE 本质上是一致的，采用 RMSE 能更直观地看出量级），损失曲线如下图所示，可以看到在$N>10^5$以后随着N的增长 RMSE 基本不再减小，而当$n>7$以后随着n的增大 RMSE 变化的程度非常小，且在$n=9$时达到了最小，而n继续增大时由于可能发生了过拟合，模型性能反而略微有所下降。综合考虑模型复杂度，最终采取$N=10^5,n=9$作为训练的参数。



作出此时模型的预测值-真值曲线，可以看出拟合度非常好，RMSE 略小于10^{-4}数量级。



然后将相应的模型参数封装起来，写为一个函数 function H = predict_H(a, b, c)（代码在 predict_H.m 中），它的作用是输入指定的a、b、c，输出根据拟合的多项式计算得到的H值。

4. 总结

这次大作业通过实践巩固了最小二乘法的相关知识，可以看出这种方法简单而强大，可以很快地将复杂的非线性函数用多项式拟合出来，而且精度也比较高。本题要拟合的函数是三次方程的解经过归一化后通过对数的非线性变化得到的函数，整个过程下来非线性比较强。而我们的观测参数有两个自由度，所以这是多元最小二乘估计。通过损失曲线可以看出，经过最小二乘法拟合后，即使拟合的多项式阶数较低也能达到较小的 RMSE（3 阶就已经收敛到10^{-3}量级了），但随着阶数的增加也不是一直性能变好的，因为有可能出现过拟合的问题，所以要折中选取合适的参数。

在编程时也发现，用 MATLAB 计算高维矩阵的逆时有可能由于内置算法的问题出现矩阵接近奇异值而导致结果不精确的问题，且有可能出现随着N的增大，RMSE 出现一定上下波动的问题，但从结果看来，软件计算的误差在可接受范围内。

5. 文件清单与代码

共有两份代码文件：

① myLSE.m

训练模型，寻找最优参数并保存为 weight.mat 文件。代码如下：

```
clc; clear all; close all;
%% 训练模型
n_max = 10;
N_min = 2;  % min of N = 10^N_min
N_max = 6;  % max of N = 10^N_max
lambda_max = 10000;
J = zeros(n_max, N_max-N_min+1);
w = cell(n_max, N_max-N_min+1);
```

```
% 对 n 与 N 的不同取值进行最小二乘估计
for n = 1:n_max
    for N = logspace(N_min, N_max, N_max-N_min+1)
        % 生成随机数并预处理数据
        lambda = lambda_max*rand(N, 3);
        [alpha, beta, p] = pre_proccess(lambda);
        % 计算观测矩阵 Q 与结果矩阵 H
        [Q, H] = get_Q_H(p, alpha, beta, n, N);
        % 最小二乘估计系数矩阵 w 并计算误差函数值
        w{n, log10(N)-N_min+1} = (Q'*Q)\Q'*H;  % 等价于 inv(Q'*Q)*Q'*H
        J(n, log10(N)-N_min+1) = get_loss(w{n, log10(N)-N_min+1}, lambda_max, n);
    end
end

%% 作图
figure;
for n = 1:n_max/2
    loglog(logspace(N_min, N_max, N_max-N_min+1), J(n,:), 'linewidth', 1.5); hold on;
end
for n = (n_max/2+1):n_max
    loglog(logspace(N_min, N_max, N_max-N_min+1), J(n,:), '-.', 'linewidth',1.5); hold on;
end
legend(strcat("n = ", num2str((1:n_max)')));
xlabel("N"); ylabel("RMSE");
ylim([min(J(:))*0.5, max(J(:))*2]);

%% 保存模型参数
w0 = w{9, 5-N_min+1};
save weight.mat w0

%% 作出模型的预测-真值曲线
n = 9;
N = 10^6;
lambda = lambda_max*rand(N, 3);
[alpha, beta, p] = pre_proccess(lambda);
[Q, H_real] = get_Q_H(p, alpha, beta, n, N);
H_pred = Q*w0;
RMSE0 = norm(H_pred - H_real) / sqrt(N);
figure;
plot(0:0.01:1, 0:0.01:1, 'linewidth', 1);  hold on;
scatter(H_real, H_pred, 3, 'filled');
legend('y = x', 'H\_pred ~ H\_real');
xlabel('H\_real');  ylabel('H\_pred');
xlim([0, 1]);  ylim([0, 1]);
title(strcat("RMSE = ", num2str(RMSE0)));

%%
function [alpha, beta, p] = pre_proccess(lambda) % 由随机生成的 lambda 计算参数
    a = -sum(lambda, 2);
    b = sum(lambda .* lambda(:, [2,3,1]), 2);
    c = -prod(lambda, 2);
    alpha = b./(a.^2);
```

```
    beta = c./(a.^3);
    p = -lambda ./ repmat(a, 1, 3);
end

function [Q, H] = get_Q_H(p, alpha, beta, n, N) % 由参数计算观测矩阵 Q 与结果矩阵 H
    H = -sum(p.*log(p)/log(3), 2);
    Q = zeros(N, (n+1)*(n+2)/2);
    cnt = 1;
    for i = 0:n
        for j = 0:i
            Q(:, cnt) = alpha.^(i-j) .* beta.^j;
            cnt = cnt+1;
        end
    end
end

function RMSE = get_loss(w, lambda_max, n)  % 计算 RMSE
    N = 10^6;
    lambda = lambda_max*rand(N, 3);
    [alpha, beta, p] = pre_proccess(lambda);
    [Q, H] = get_Q_H(p, alpha, beta, n, N);
    RMSE = norm(H - Q*w) / sqrt(N);
end
```

② **predict_H.m**

输入给定的特征方程系数a、b、c，加载 weight.mat 文件，输出根据拟合的多项式计算得到的H值。代码如下：

```
function H = predict_H(a, b, c)
    n = 9;
    alpha = b./(a.^2);
    beta = c./(a.^3);
    Q = zeros(1, (n+1)*(n+2)/2);
    cnt = 1;
    for i = 0:n
        for j = 0:i
            Q(cnt) = alpha.^(i-j) .* beta.^j;
            cnt = cnt+1;
        end
    end
    load('weight.mat', 'w0');
    H = Q*w0;
end
```