曲线拟合之matlab实现

emacsun

目录

1 回忆

在

1. 多项式拟合到模式识别的相关概念
2. 曲线拟合过程中的欠定过定问题
3. 曲线拟合之概率回访

中我们介绍了多项式曲线拟合问题并从多方面对其进行了分析，顺便引入了机器学习领域的一些关键术语。今天我们针对多项式曲线拟合问题做一些matlab试验。

1.1 数学模型

目标模型：

$$y(x,\mathbf{w}) = w_0 + w_1 x + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j \tag{1.1}$$

均方误差函数：

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 \tag{1.2}$$

针对目标模型和均方误差函数的解$\mathbf{w} = \{w_j\}$是一般线性方程：

$$\sum_{j=0}^{M} A_{ij} w_j = T_i, i = 0, \ldots, M \tag{1.3}$$

的解。其中$A_{ij} = \sum_{n=1}^{N}(x_n)^{i+j}$，$T_i = \sum_{n=1}^{N}(x_n)^i t_n$ 关于这个结论的证明过程详见 曲线拟合过程中的欠定过定问题 。

式 (1.3) 可以写成矩阵的形式，如下：

$$\begin{bmatrix} A_{00} & \ldots & A_{0M} \\ \vdots & \ddots & \vdots \\ A_{M0} & \ldots & A_{MM} \end{bmatrix} \begin{bmatrix} w_0 \\ \vdots \\ w_M \end{bmatrix} = \begin{bmatrix} T_0 \\ \vdots \\ T_M \end{bmatrix} \tag{1.4}$$

1.2 正则化过程

针对误差函数 (1.2)出现高阶模型的过拟合问题，对(1.2) 添加一个正则项会产生比较好的效果，如：

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{1.5}$$

其中$\|\mathbf{w}\|^2 = w_0^2 + \Gamma \ldots w_M^2$ ，稍后我们会看到这种正则化方法对于克服过拟合问题非常有效。针对式 (1.5) 和式 (1.1) 的解可以写成如下形式的解：

$$\sum_{j=0}^{M}(A_{ij} + \lambda\delta_{ij})w_j = T_i, i = 0, \ldots, M \tag{1.6}$$

其中：

$$\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{1.7}$$

矩阵形式就是：

$$\begin{bmatrix} A_{00}-\lambda & \dots & A_{0M} \\ \vdots & \ddots & \vdots \\ A_{M0} & \dots & A_{MM}-\lambda \end{bmatrix} \begin{bmatrix} w_0 \\ \vdots \\ w_M \end{bmatrix} = \begin{bmatrix} T_0 \\ \vdots \\ T_M \end{bmatrix} \tag{1.8}$$

式~(1.8)和式 (1.4) 的区别在于，矩阵A的对角线上有一个λ的修正（或者叫做惩罚）。这样做的好处是求得的系数**w**不会太大。

2 matlab实现

2.1 画出$y = \sin(2\pi x)$

[2017-05-21 Sun 15:55] 首先我们画出$y = \sin(2\pi x)$，这个函数的图像:

```
x = 0:0.01:1;%x
y = sin(2* pi * x);
figure
plot(x,y,'-r','linewidth',2),hold on;
```

结果如图1 所示：

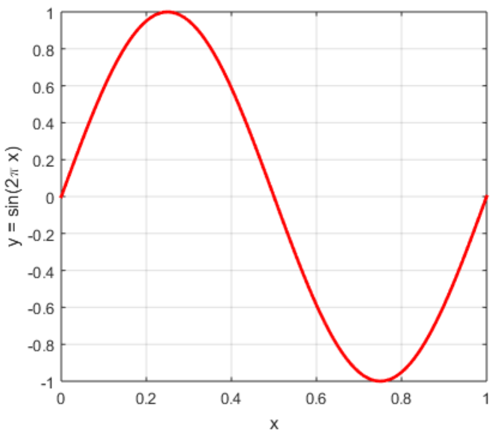

图 1: $y = \sin(2 * \pi x)$

2.2 训练集合

[2017-05-21 Sun 15:55] 然后给出一个训练集合，并画出这个集合的图像：

```
numTraining = 100;
```

```
xTraining = rand(1,numTraining);
noiseVariance = 0.5;
noise = noiseVariance *randn(1,numTraining);
yTraining = sin(2*pi*xTraining) + noise;
plot(xTraining,yTraining,'b+','linewidth',2),hold on;
```

和$y = \sin(2\pi x)$画在一张图上，如图所示：

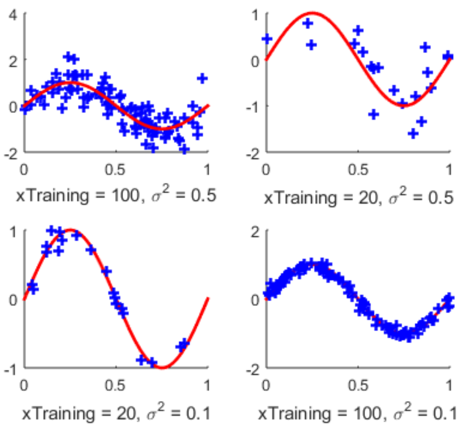

图 2: 训练集合

注意这里有两个地方可以做调整：1. 训练结合的数量 `numTraining` ；2. 噪声的方差 `noiseVariance` .不同的训练集合大小和方差会影响最终的训练效果。当然总体结论是：训练集合越大，方差越小，训练效果越好。

2.3 曲线拟合

曲线拟合的过程就是计算**w**的过程，首先我们观察式~(1.3) 的结果。我们把计算$Ax = B$的过程分为三步：

1. `getA`
2. `getB`
3. `A\B`

这三步的代码：

```
%%find the A and B for Ax = B
A = getA(xTraining,modelOrder);
B = getB(xTraining,yTraining,modelOrder);
w = A\B';
for i = 1:length(x)
```

```
yEstimation(i) = x(i).^(0:modelOrder ) * w  ;
end
```

先是 getA

```
function A = getA(xTraining,modelOrder)
    for i = 1:modelOrder + 1
        for j = 1:modelOrder + 1
            A(i,j) = sum(xTraining.^(i+j - 2));
        end
    end
end
```

然后 getB

```
function B = getB(xTraining,yTraining,modelOrder)
   for i = 1:modelOrder + 1
       B(i) = sum(yTraining .* xTraining.^(i-1));
   end
end
```

我们给出训练集合大小为20，模型阶数为9的训练结果：

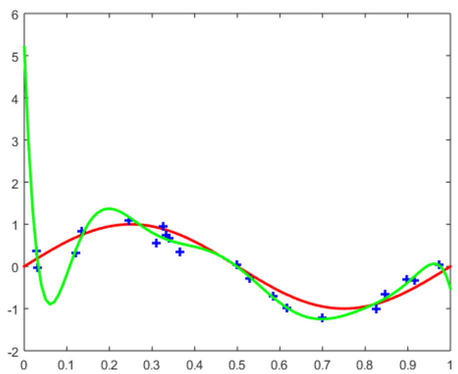

图 3: 训练集合大小为20，模型阶数为9的训练结果

图中绿色曲线就是训练结果，可以看出来这个结果显然是过拟合了。我们给出来训练集合大小为20，模型结束为5的训练结果：

2.4 正则化

按照式 (1.6) 所给结果，设置$\lambda = 0.001$，设置模型阶数为9，训练集合大小为20，训练结果：

从图5 和 3 可以看到，正则化有效的遏制了拟合结果波动过大的情况。

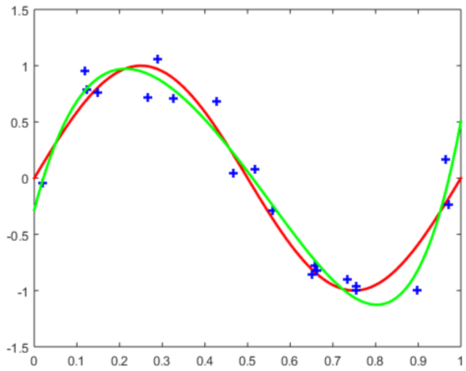

图 4: 训练集合大小为20，模型阶数为5的训练结果

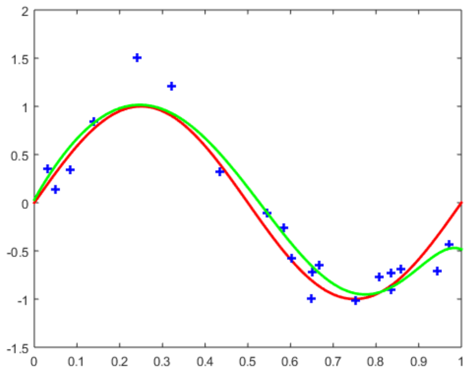

图 5: 训练集合大小为20，模型阶数为9的训练结果,$\lambda = 0.001$

3 下载完整的代码

下载完整的代码。