哈尔滨工业大学计算机科学与技术学院

实验报告

课程名称：机器学习

课程类型：必修

实验题目：多项式拟合正弦函数

学号：1160300314

姓名：朱明彦

一、实验目的

掌握最小二乘法求解（无惩罚项的损失函数），掌握增加惩罚项（2范数）的损失函数优化，梯度下降法、共轭梯度法，理解过拟合、客服过拟合的方法（如增加惩罚项、增加样本）。

二、实验要求及实验环境

实验要求

1. 生成数据，加入噪声；
2. 用高阶多项式函数拟合曲线；
3. 用解析解求解两种loss的最优解（无正则项和有正则项）
4. 优化方法求解最优解（梯度下降，共轭梯度）；
5. 用你得到的实验数据，解释过拟合。
6. 用不同数据量，不同超参数，不同的多项式阶数，比较实验效果。
7. 语言不限，可以用matlab，python。求解解析解时可以利用现成的矩阵求逆。梯度下降，共轭梯度要求自己求梯度，迭代优化自己写。不许用现成的平台，例如pytorch，tensorflow的自动微分工具。

实验环境

- OS: Microsoft Windows 10.0.17134
- Python 3.6.4

三、设计思想(本程序中用到的主要算法及数据结构)

算法原理

1、生成数据算法

主要是利用$sin(2\pi x)$函数产生样本，其中x均匀分布在$[0,1]$之间，对于每一个目标值$t=sin(2\pi x)$增加一个0均值，方差为0.5的高斯噪声。

2、利用高阶多项式函数拟合曲线(不带惩罚项)

利用训练集合，对于每个新的$\hat{x}$，预测目标值$\hat{t}$。采用多项式函数进行学习，即利用式(1)来确定参数w，假设阶数m已知。

$$y(x,w)=w_0+w_1x+\cdots+w_mx^m=\sum_{i=0}^{m}w_ix^i \tag{1}$$

采用最小二乘法，即建立误差函数来测量每个样本点目标值t与预测函数$y(x,w)$之间的误差，误差函数即式(2)

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\{y(x_i,\mathbf{w}) - t_i\}^2 \tag{2}$$

将上式写成矩阵形式如式(3)

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{Xw} - \mathbf{T})'(\mathbf{Xw} - \mathbf{T}) \tag{3}$$

其中

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & \cdots & x_1^m \\ 1 & x_2 & \cdots & x_2^m \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & \cdots & x_N^m \end{bmatrix}, \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix}, \mathbf{T} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{bmatrix}$$

通过将上式求导我们可以得到式(4)

$$\frac{\partial E}{\partial \mathbf{w}} = \mathbf{X'Xw} - \mathbf{X'T} \tag{4}$$

令 $\dfrac{\partial E}{\partial \mathbf{w}} = 0$ 我们有式(5)即为$\mathbf{w}^*$

$$\mathbf{w}^* = (\mathbf{X'X})^{-1}\mathbf{X'T} \tag{5}$$

3、带惩罚项的多项式函数拟合曲线

在不带惩罚项的多项式拟合曲线时，在参数多时$\mathbf{w}^*$具有较大的绝对值，本质就是发生了过拟合。对于这种过拟合，我们可以通过在优化目标函数式(3)中增加$\mathbf{w}$的惩罚项，因此我们得到了式(6)。

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\{y(x_i,\mathbf{w}) - t_i\}^2 + \frac{\lambda}{2}||\mathbf{w}||^2 \tag{6}$$

同样我们可以将式(6)写成矩阵形式， 我们得到式(7)

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}[(\mathbf{Xw} - \mathbf{T})'(\mathbf{Xw} - \mathbf{T}) + \lambda\mathbf{w'w}] \tag{7}$$

对式(7)求导我们得到式(8)

$$\frac{\partial \widetilde{E}}{\partial \mathbf{w}} = \mathbf{X'Xw} - \mathbf{X'T} + \lambda\mathbf{w} \tag{8}$$

令 $\dfrac{\partial \widetilde{E}}{\partial \mathbf{w}} = 0$ 我们得到$\mathbf{w}^*$即式(9)，其中$\mathbf{I}$为单位阵。

$$\mathbf{w}^* = (\mathbf{X'X} + \lambda\mathbf{I})^{-1}\mathbf{X'T} \tag{9}$$

4、梯度下降法求解最优解

对于$f(\mathbf{x})$如果在$\mathbf{x_i}$点可微且有定义，我们知道顺着梯度 $\nabla f(\mathbf{x_i})$为增长最快的方向，因此梯度的反方向 $-\nabla f(\mathbf{x_i})$ 即为下降最快的方向。因而如果有式(10)对于 $\alpha > 0$ 成立，

$$\mathbf{x_{i+1}} = \mathbf{x_i} - \alpha\nabla f(\mathbf{x_i}) \tag{10}$$

那么对于序列$\mathbf{x_0}, \mathbf{x_1}, ...$我们有

$$f(\mathbf{x_0}) \geq f(\mathbf{x_1}) \geq ...$$

因此，如果顺利我们可以得到一个 $f(\mathbf{x_n})$ 收敛到期望的最小值，**当然对于我们此次实验很大可能性可以收敛到最小值。**

5、共轭梯度法求解最优解

共轭梯度法解决的主要是形如$\mathbf{Ax} = \mathbf{b}$的线性方程组解的问题，其中$\mathbf{A}$必须是对称的、正定的。 求解的方法就是我们先猜一个解$\mathbf{x_0}$，然后取梯度的反方向 $\mathbf{p_0} = \mathbf{b} - \mathbf{Ax}$，在n维空间的基中$\mathbf{p_0}$要与其与的基共轭并且为初始残差。

然后对于第k步的残差 $\mathbf{r_k} = \mathbf{b} - \mathbf{Ax}$，$\mathbf{r_k}$为$\mathbf{x} = \mathbf{x_k}$时的梯度反方向。由于我们仍然需要保证 $\mathbf{p_k}$ 彼此共轭。因此我们通过当前的残差和之前所有的搜索方向来构建$\mathbf{p_k}$，得到式(11)

$$\mathbf{p_k} = \mathbf{r_k} - \sum_{i<k} \frac{\mathbf{p_i}^T \mathbf{A} \mathbf{r_k}}{\mathbf{p_i}^T \mathbf{A} \mathbf{p_i}} \mathbf{p_i} \tag{11}$$

进而通过当前的搜索方向$\mathbf{p_k}$得到下一步优化解$\mathbf{x_{k+1}} = \mathbf{x_k} + \alpha_k \mathbf{p_k}$，其中

$$\alpha_k = \frac{\mathbf{p_k}^T(\mathbf{b} - \mathbf{A}\mathbf{x_k})}{\mathbf{p_k}^T \mathbf{A} \mathbf{p_k}} = \frac{\mathbf{p_k}^T \mathbf{r_k}}{\mathbf{p_k}^T \mathbf{A} \mathbf{p_k}}$$

算法的实现

对于数据生成、求解析解（有无正则项）都是可以利用numpy中的矩阵求逆等库变相降低了算法实现的难度，因此在这里就不再赘述，下面主要讲梯度下降法和共轭梯度法的算法实现。

梯度下降

共轭梯度下降

四、实验结果分析

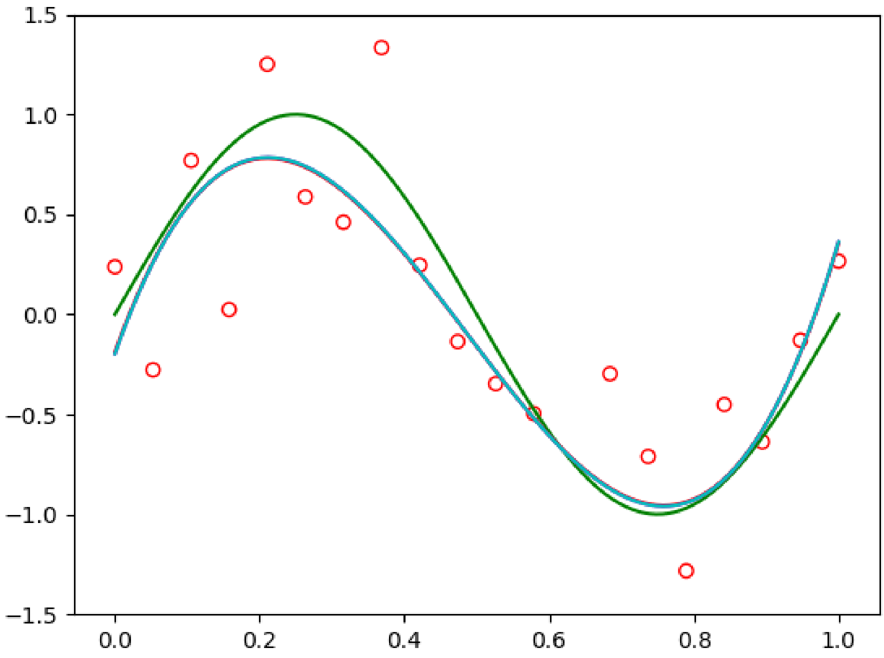

Figure 1.test

五、结论

六、参考文献

- Pattern Recognition and Machine Learning.
- Gradient descent wiki
- Conjugate gradient method wiki

七、附录:源代码(带注释)

```
print("Hello world")

import numpy as np

np.one(10)
```