哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2020 年春季学期

计算学部《机器学习》课程

Lab 1 实验报告

姓名	王丁子睿
学号	1183710211
班号	1803104
电子邮件	1183710211@stu.hit.edu.cn
手机号码	19845178018

目录

1 问题概述

给定正弦曲线上的一系列点，施加噪声后，用多项式函数来拟合原曲线，拟合的标准为均方误差最小。

即给定点集

$$\{(x_i, y_i)\}$$

求多项式

$$f(x) = \begin{pmatrix} 1 & \dots & x_1^n \\ \vdots & \ddots & \vdots \\ 1 & \cdots & x_m^n \end{pmatrix} \begin{pmatrix} w_0 \\ \vdots \\ w_n \end{pmatrix} = Xw$$

使得

$$MSE = \frac{1}{m}\sum_{i=1}^{m}\left(y_i - f(x_i)\right)^2$$

最小。

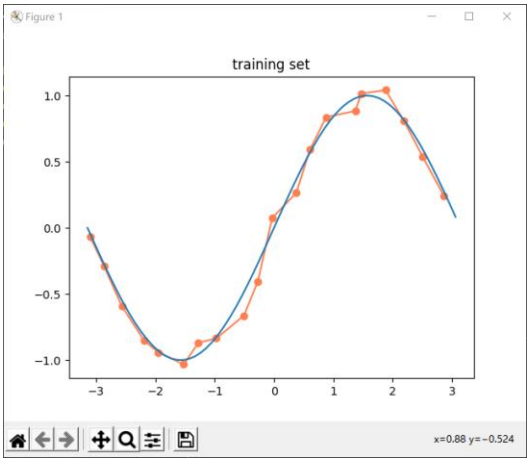
2 数据生成

（代码见 data.DatasetManager.load_random）

为了得到训练集，首先给定一个区间，表示随机生成的x的范围，然后求出每个x对应的y，之后对于每对(x, y)，施加一定的噪声。

这里取$x \in (-\pi, \pi)$，施加噪声的方式为：对于每个点对，先根据标准正态分布生成两个随机数。由于y的值相对较小，所以x和y分别加上生成随机数的$\frac{1}{20}$。

一个规模为 20 的训练集如图所示：



此外，为了保证结果可重复，以下未特殊说明的部分，均取测试集大小为 20，随机数种子为 0。

3 问题求解

3.1 无正则项解析解

（代码见 process. analysis_without_regular）

由于损失函数取的是 MSE，因此可以用最小二乘法来直接进行拟合。

根据微积分相关的知识，函数取最小值的点为梯度等于 0 的点。

误差函数为

$$E(w)=\frac{1}{2}\sum_{i=1}^{n}(f_w(x_i)-y_i)^2=\frac{1}{2}(Xw-y)^T(Xw-y)$$

对其求导得

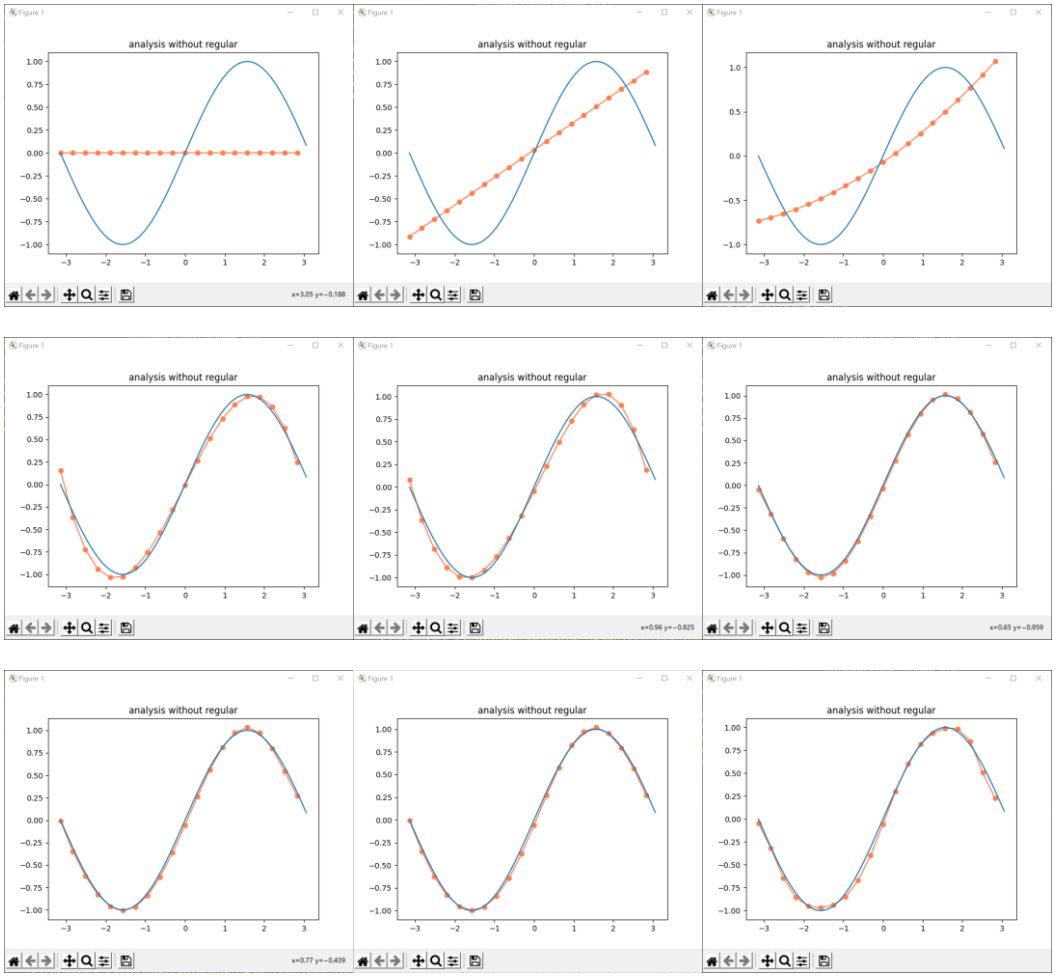
$$\frac{\partial E(w)}{\partial w}=X^TXw-X^Ty=0$$

解得

$$w=(X^TX)^{-1}X^Ty$$

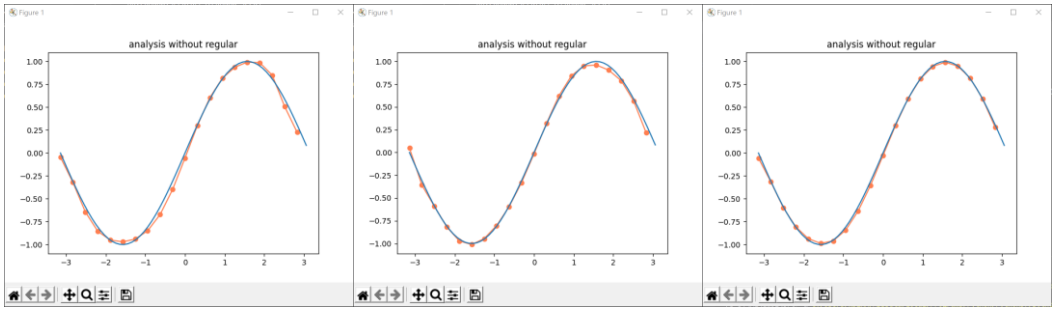
故可以直接求得w的值。

取训练集规模为 20，依次取多项式的阶数为 0 到 8，结果如下图所示：



可以看到在阶数为 5 之前，拟合的效果随着阶数的增大而逐渐改善，但之后拟合的效果逐渐变差，拟合曲线和正弦曲线开始发生偏离，表明发生了过拟合的现象。

取多项式的阶数为 8，依次取训练集的规模为 20、60、100，结果如下图所示：



可以看到随着训练集规模的增大，过拟合的现象有所改善。

3.2 有正则项解析解

（代码见 process. analysis_with_regular）

为了改善多项式阶数增加带来的过拟合的影响，我们引入正则项，误差函数变为：

$$E(w)=\frac{1}{2m}\left(\sum_{i=1}^{n}(f_w(x_i)-y_i)^2+\lambda\sum_{i=1}^{n}w_i\right)$$

其中λ为正则系数，下取 0.05。

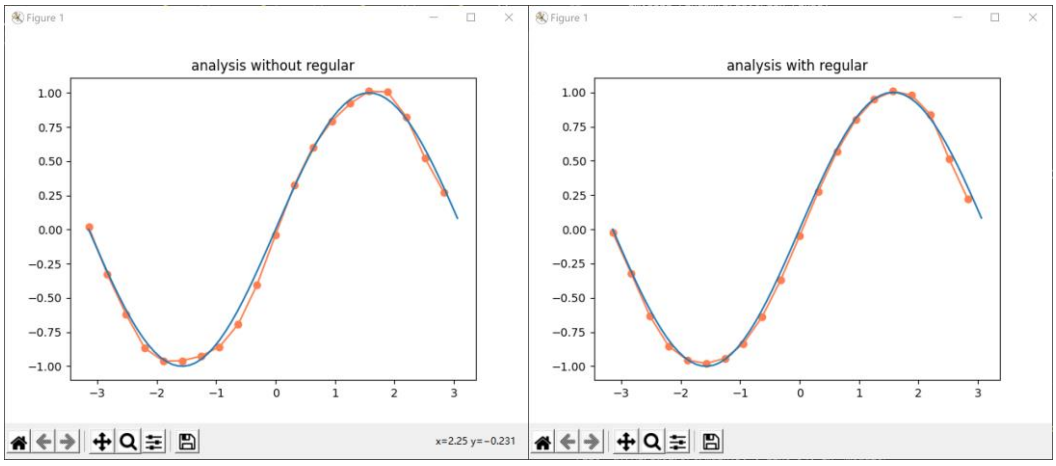
对误差函数求导得

$$\frac{\partial E(w)}{\partial w}=\frac{1}{m}\left((X^TX+\lambda E_m)w-X^Ty\right)=0$$

解得

$$w=(X^TX+\lambda E_m)^{-1}X^Ty$$

取训练集规模为 20，多项式阶数为 10，比较引入正则项前后的拟合效果如图所示：



可以看到，拟合效果有一定改善。

3.3 梯度下降法

（代码见 process. gradient_descent）

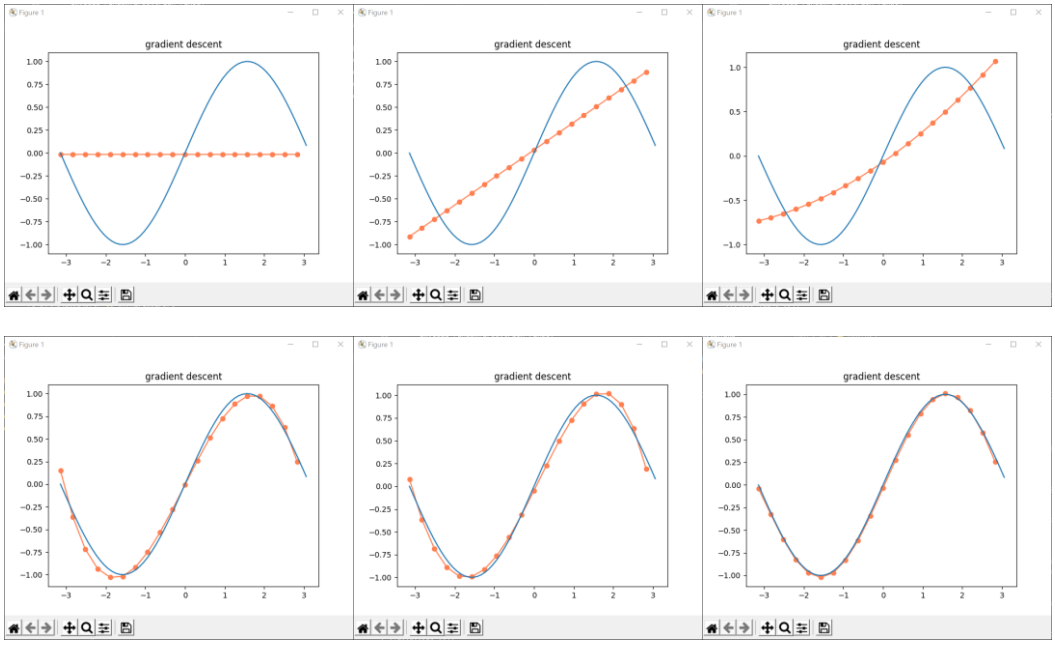
在上述解析解直接求解的过程中，由于需要进行矩阵求逆的运算，所以在训练集或多项式阶数较大时，可能会有运算效率低下的问题。

我们知道，函数上任意一点的梯度指向函数下降最快的方向，而我们原本的目标，就是求损失函数的最小值。

因此我们可以先随机生成一个w作为起点，每次沿着梯度的方向前进一定的距离，并更新w，如此反复，直到函数的梯度小于一定的阈值，即已经位于最小值附近。

我们每次前进的距离为$lr \times \nabla E(w)$，lr称为学习率。

取训练集规模为 20，依次取多项式的阶数为 0 到 5，学习率为 0.0001，梯度阈值为 0.0001，结果如下图所示：



可以发现，梯度下降得到的结果和通过解析式直接求解得到的结果基本是一致的。

但是由于该方法每一步都是沿着这一点的梯度走，实际上是贪心的思想，所以会有陷入局部最优解的风险。

此外，由于迭代的次数并没有保证，所以在阶数较大时，很有可能出现计算效率低下的问题。

各阶数对应迭代次数如下表所示：

阶数	0	1	2	3	4	5
迭代次数	31	345	7265	81760	115462	178654

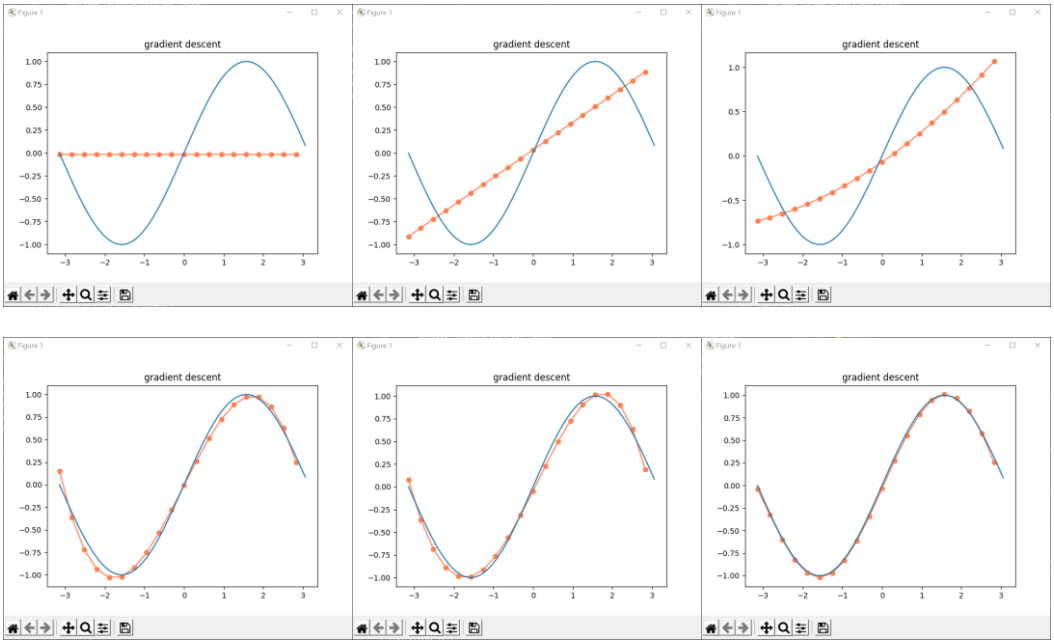
3.4 共轭梯度法

（代码见 process. conjugate_gradient）

由于在梯度下降法中，为了达到目标精度，可能需要大量的迭代次数，所以在效率上依然存在较大的问题。

而共轭梯度法每次沿着全局下降最快的一个方向前进，所以理论上只需要迭代n步就能走到最优解，其中n为多项式的阶数。

取训练集规模为 20，依次取多项式的阶数为 0 到 5，结果如下图所示：



可以看到结果与前述方法基本一致，且计算速度极快，是一种特别优秀的方法。