第 3 章　连续变量与线性回归

回归分析着重寻求变量之间近似的函数关系

正如 1.1.1 节谈到的，有监督学习根据目标变量y_i是否连续可以进一步分为分类（y_i是离散变量）和回归（y_i是连续变量）。第 2 章详细讨论了 5 种基本的分类模型，本章将对回归模型展开讨论。

根据目标变量y_i是否为关于参数的线性函数，可以将回归模型分为线性回归和非线性回归两大类。本章只对线性回归模型进行介绍，至于非线性回归，这里仅作一个简要的说明。比如以第 2 章介绍的 k-近邻和决策树为例，这两种模型都是非线性模型，也都可以用来解决回归问题（比如 k-近邻通过取均值进行回归），而成为非线性回归模型。本章先介绍基本线性回归和经典的最小二乘法，然后接着介绍岭回归和 LASSO 两种正则化的线性回归模型。

3.1　基本线性回归

回顾 2.3.1 节谈到的线性模型$f(\boldsymbol{x};\boldsymbol{w})=\boldsymbol{w}^T\boldsymbol{x}+b$，这个模型可以直接应用到回归问题上，也就是直接用$\hat{y}=f(\boldsymbol{x};\boldsymbol{w})$来预测连续目标变量$y$的值。这就得到了线性回归模型。图 3.1 给出了一个线性回归的例子（类似图 1.2 给出的例子），直观上可以看到拟合出的直线$y=0.6x+0.52$（图中长实线所示）从所有样本点的中间穿过。实际上，这正是源于基本线性回归的优化目标：使得所有训练样本点（图中圆点所示）与其对应的拟合直线上点的距离（图中用短实线标出了一个距离）在总体上达到最小。这就是所谓的最小二乘法。

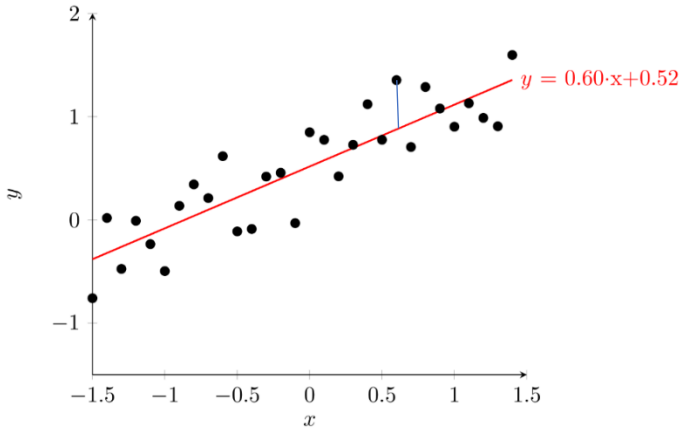


图 3.1　一个线性回归的例子

正式地，为了方便，可以将$f(\boldsymbol{x};\boldsymbol{w})=\boldsymbol{w}^T\boldsymbol{x}+b$写为等价的$f(\boldsymbol{x};\boldsymbol{w})=\boldsymbol{w}^T\boldsymbol{x}$增扩向量形式，其中$\boldsymbol{w}=(w_1,\ldots,w_F,b)^T$，$\boldsymbol{x}=(x_1,\ldots,x_F,1)^T$，$F$表示特征维数。为了度量预测值$\hat{y}$与真实值$y$之间的误差，可以直接将两个值相减（对应图 3.1 中的短实线），然后取平方，从而得到二次损失函数：

$$L(\boldsymbol{w})=\frac{1}{2}\sum_{n=1}^{N}(y^{(n)}-\boldsymbol{w}^T\boldsymbol{x}^{(n)})^2 \qquad (3.1)$$

其中，$\hat{y}=\boldsymbol{w}^T\boldsymbol{x}^{(n)}$为样本点$\boldsymbol{x}^{(n)}$的预测值，$y^{(n)}$为其真实值。式（3.1）在训练集$D$的所有$N$个样本点上计算总的损失，正是“误差在总体上最小” 这一要求的反映。实际上，在 2.5 节已经用到了二次损失函数，用来度量类别预测误差，使得“误差在总体上最小”。对于分类问

题，1.2.3 节也谈到了 0-1 损失函数，这是分类问题最为自然的损失函数：相同则无损失，不同则损失为 1。那么，读者一定会问，对于分类问题，为什么 2.5 节不直接用 0-1 损失函数呢？答案是，2.5 节的梯度下降优化算法需要对损失函数求导，而 0-1 损失函数不可导。因此，采用可导的二次损失函数来替代 0-1 损失函数，并将其称为替代损失函数或者代理损失函数。读者可以进一步思考下，对于回归问题同样采用二次损失函数的合理性在哪里？（习题 3.2）

式（3.1）是向量形式，为了方便，可以将其写为等价的矩阵形式：

$$L(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{w}\|^2 = \frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{w}),$$

$$其中\boldsymbol{y} = \left[y^{(1)}, \dots, y^{(N)}\right]^{\boldsymbol{T}} \in R^N,$$

$$\boldsymbol{X} = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(N)} \\ \vdots & \vdots & \ddots & \vdots \\ x_F^{(1)} & x_F^{(2)} & \cdots & x_F^{(N)} \\ 1 & 1 & \dots & 1 \end{bmatrix} \tag{3.2}$$

其中，$\boldsymbol{y}$为N个训练样本点对应的标签向量；$\boldsymbol{X}$为N个训练样本点的增扩特征矩阵，一列表示一个样本点的增扩特征向量，一行表示一个特征。

为了得到$L(\boldsymbol{w})$的最小值，可以求$L(\boldsymbol{w})$对$\boldsymbol{w}$的导数，得到：

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}) = \frac{1}{2}\frac{\partial\|\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{w}\|^2}{\partial \boldsymbol{w}} = -\boldsymbol{X}(\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{w}) \tag{3.3}$$

式（3.3）就是二次函数求导，容易得到。取$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}) = 0$，就可得到最优的参数$\boldsymbol{w}^*$：

$$\boldsymbol{w}^* = (\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}\boldsymbol{y} \tag{3.4}$$

式（3.4）就是所谓的（线性）最小二乘解[1]。求解基本线性回归的这个方法称为（线性）最小二乘法。

式（3.4）要求方阵$\boldsymbol{X}\boldsymbol{X}^T$可逆，即$\boldsymbol{X}$的行向量（对应特征）之间线性不相关。一种常见的不可逆情况是：样本数N小于特征数$F+1$，这时候相当于有$F+1$个未知数，但只有N个方程，$\boldsymbol{w}^*$的解就有很多个。

3.2 岭回归

如果$\boldsymbol{X}\boldsymbol{X}^T$不可逆，则式（3.4）给出的最小二乘解就无法求解。岭回归的基本思想是：给$\boldsymbol{X}\boldsymbol{X}^T$的主对角元素都加上一个小的正数$\lambda$，使得方阵$(\boldsymbol{X}\boldsymbol{X}^T + \lambda I)$可逆。这样，式（3.4）就成为：

$$\boldsymbol{w}^* = (\boldsymbol{X}\boldsymbol{X}^T + \lambda I)^{-1}\boldsymbol{X}\boldsymbol{y} \tag{3.5}$$

其中，$\lambda > 0$，是一个超参；I是单位阵。λ加在$\boldsymbol{X}\boldsymbol{X}^T$的主对角元素上，形成一个个“山岭”，因此称为岭回归。

实际上，岭回归就是 L2 正则化的基本线性回归：

$$L(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{w}\|^2 + \frac{1}{2}\lambda\|\boldsymbol{w}\|^2 \tag{3.6}$$

所谓的 L2 正则化，实际上就是给参数$\boldsymbol{w}$增加了一个二次约束（$\|\boldsymbol{w}\|^2$），或者说给参数$\boldsymbol{w}$增加了一个二次惩罚项。这样，将使得问题能够得到更稳定和推广能力更好的解。式（3.6）中，$L(\boldsymbol{w})$被称为结构风险函数，其中超参λ用来权衡基于训练数据的经验风险项（式（3.6）的第一项）和正则项（式（3.6）的第二项），被称为正则化系数。λ的值越小，则经验风险项越重要，模型越复杂，更倾向于对训练数据的过拟合；相反，则正则项越重要，模型越简单，更

注 1：中国古代把平方称为二乘，最小二乘由此得名。对应的英文是 Least Square。

倾向于对训练数据的欠拟合。正如 1.2.3 节所介绍的，可以通过交叉验证等方法选取一个合适的λ值，以期达到对训练数据的最佳拟合。回顾一下，在 2.4.4 节中，已经介绍了正则项和正则化系数的概念，在那里超参C起着类似的作用。

更一般地，对于任意向量$\boldsymbol{x}$，有L_p范数正则化的概念，其中L_p范数定义为：

$$\|\boldsymbol{x}\|_p = (\sum_i |x_i|^p)^{\frac{1}{p}} \tag{3.7}$$

当$p=2$时，就是L_2范数，一般省略p不写；当$p=1$时，就得到常用的L_1范数。式（3.6）中的 L2 正则化项$\|\boldsymbol{w}\|^2$实际上是$\boldsymbol{w}$的L_2范数的平方，是一个二次函数。

3.3 基本线性回归的一个改进：局部加权线性回归

基本线性回归存在的主要问题是，对数据的欠拟合。如图 3.2 所示，训练数据（图中圆点所示）虽然整体上是一个线性增长的趋势，但存在着局部的起伏，基本线性回归拟合出的直线（图中斜直线所示）并不能反映出这种局部的变化规律，模型处于欠拟合的状态。

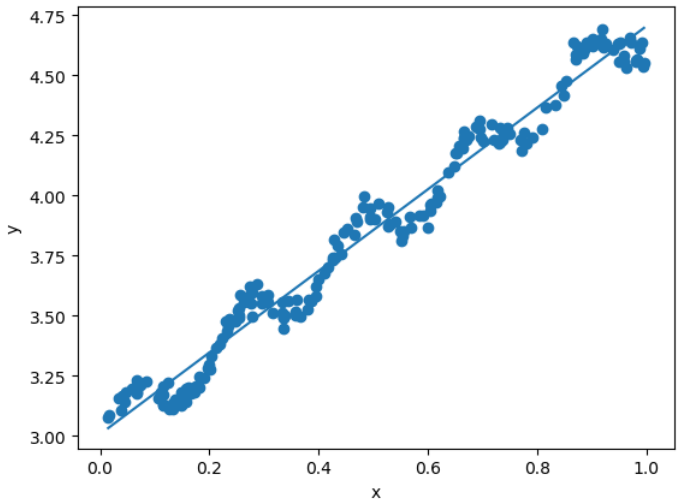


图 3.2 基本线性回归对数据的欠拟合

实际上，局部加权线性回归就是 2.4.7 节介绍的核函数和核方法在回归问题上的一种应用。其基本思想是：引入核函数（比如式(2.4.41)给出的高斯核函数）将基本线性回归模型拓展为对应的非线性模型。以高斯核函数为例，具体来讲，定义$N \times N$的加权矩阵$\boldsymbol{W}$，其主对角元素为：

$$w(i,i) = exp(\frac{-\|\boldsymbol{x}^{(i)} - \boldsymbol{x}\|^2}{2\sigma^2}) \tag{3.8}$$

其他元素均为 0。式（3.7）中，$\boldsymbol{x}^{(i)}$表示第i个训练样本；$\boldsymbol{x}$表示测试样本；参数σ称为带宽，类似高斯分布的方差，用来控制各训练样本相对于中心$\boldsymbol{x}$的散开程度，σ越大，则各训练样本的权重$w(i,i)$越大。对增扩特征矩阵$\boldsymbol{X}$进行加权，则得到：

$$\boldsymbol{w}^* = (\boldsymbol{X}\boldsymbol{W}\boldsymbol{X}^T)^{-1}\boldsymbol{X}\boldsymbol{W}\boldsymbol{y} \tag{3.9}$$

这就是所谓的局部加权线性回归。通过调整高斯核函数的带宽σ，就可以控制模型的复杂程度，σ越大，则模型越简单；反之，则模型越复杂。后面通过具体的实例，可以清楚地看到这一点。

3.4 LASSO 回归

如果将式（3.6）中的 L2 正则化替换为 L1 正则化，就得到了 L1 正则化的基本线性回归，称为 LASSO 回归：

$$L(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|_1 \tag{3.10}$$

其中，$\|\boldsymbol{w}\|_1$就是前面提到的L_1范数。

为了更清楚地看到超参λ在式（3.6）和式（3.10）中的作用，并比较两种正则化方式的不同之处，可以将两个最优化问题分别等价地写为：

$$\boldsymbol{w}^* = argmin\|\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{w}\|^2, 满足\|\boldsymbol{w}\|^2 \le t \tag{3.11}$$

$$\boldsymbol{w}^* = argmin\|\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{w}\|^2, 满足\|\boldsymbol{w}\|_1 \le t \tag{3.12}$$

其中，$t > 0$为超参。设$\boldsymbol{w} = \{w_1, w_2\}$为 2 维向量，可以用图示的方式来帮助理解式（3.11）和式（3.12）的几何意义。

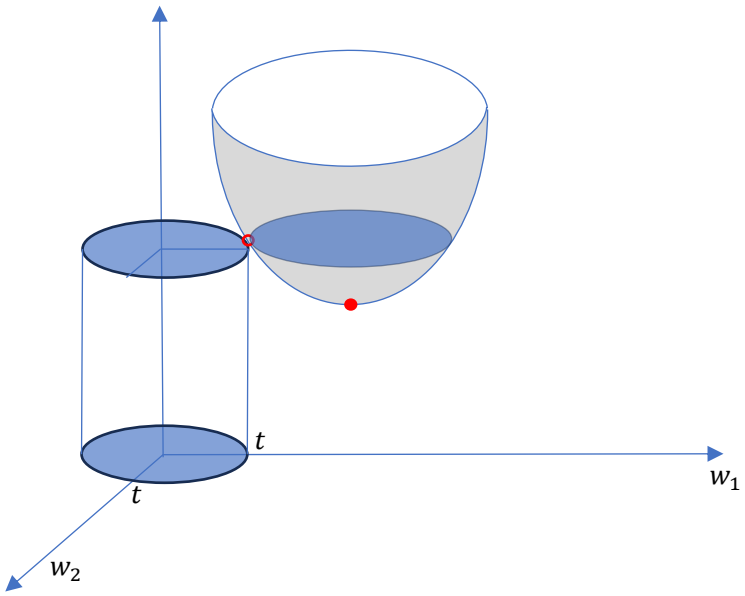


图 3.3 岭回归优化过程的图示

图 3.3 给出了式（3.11）岭回归优化过程的图示。该图中，抛物面表示经验风险项$\|\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{w}\|^2$，其最小值由实心圆点标出，对应式（3.4）给出的最小二乘解。圆柱体表示 L2 正则条件$\|\boldsymbol{w}\|^2 \le t$，其与抛物面的交点（图中空心圆点所示）即为式（3.11）给出的最优解。由该图可见，t变小，则$\|\boldsymbol{w}\|^2$越小，圆柱体与抛物面的交点越向上移，从而距离实心圆点（最小值）越远；相反，如果t变大，则$\|\boldsymbol{w}\|^2$越大，交点越向下移，从而距离实心圆点越近。t变小等价于式（3.6）中λ变大，模型更倾向于欠拟合；相反，t变大等价于λ变小，模型更倾向于过拟合。

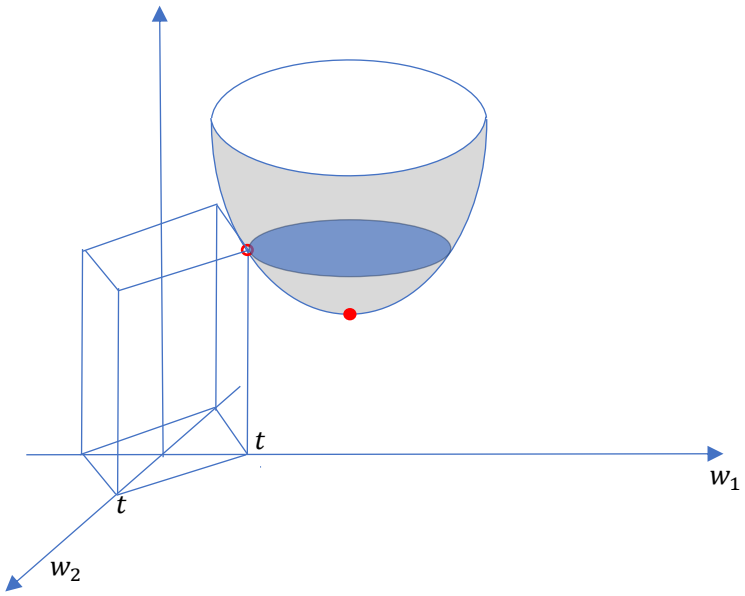


图 3.4 LASSO 回归优化过程的图示

类似地，图 3.4 给出了式（3.12）LASSO 回归优化过程的图示。该图中，长方体表示 L1 正则条件$\|\boldsymbol{w}\|_1 \leq t$，其与抛物面的交点（图中空心圆点所示）即为式（3.12）给出的最优解。该图中，类似地，t变小，则$\|\boldsymbol{w}\|_1$越小；t变大，则$\|\boldsymbol{w}\|_1$越大。

虽然图 3.3 和图 3.4 中，t的减小都会起到缩减权重向量$\boldsymbol{w}$的效果，但两者在解的稀疏性上还是有所不同。如图 3.3 所示，关于原点中心对称的圆柱体与一个任意抛物面的交点不会在任意一个坐标轴（w_1或w_2）上，也就是说，w_1或w_2都不会为 0。相反，图 3.4 中，关于原点中心对称且顶点位于坐标轴上的长方体与一个任意抛物面的交点一般会是长方体的顶点，从而或者w_1为 0，或者w_2为 0。所以，岭回归的解不具稀疏性，而 LASSO 回归的解则具有稀疏性。因此，LASSO 回归也起到了特征选择的作用：权重为 0 的特征被筛选掉。图 3.5 给出了一个图示，为了看得更清楚，该图是将图 3.3 和图 3.4 的三维图形投影到了w_1和w_2的二维平面上。

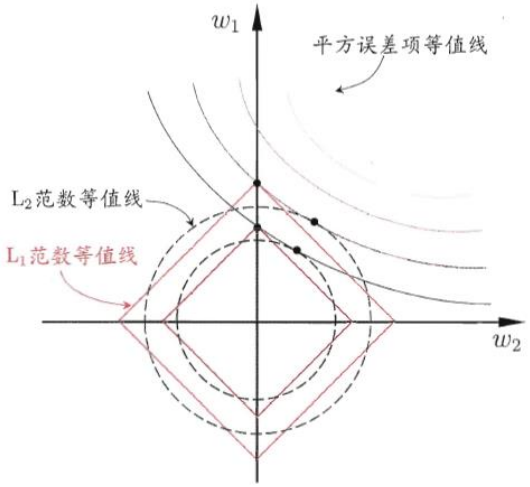


图 3.5 岭回归解与 LASSO 回归解的图示

显然，式（3.6）和式（3.10）的最小化问题都可以通过梯度下降来求解。式（3.10）还可采用坐标下降方法近似求解。

3.5 线性回归的核心代码实现

本节将分别就基本线性回归、局部加权线性回归、岭回归进行代码实现，并在简单数据集上进行实际验证。相信读者在此基础上能够进一步完善和改进代码，解决更具挑战性的实际问题。

3.5.1 基本线性回归

```
def standRegres(xList, yList):
    xMat = np.mat(xList); yMat = np.mat(yList).T
    xTx = xMat.T * xMat
    if np.linalg.det(xTx) == 0.0:
        print("This matrix is singular, cannot do inverse")
        return
    ws = xTx.I * (xMat.T * yMat)
    return ws
```

这个函数很简单，就是套式（3.4）给出的公式直接求解。需要注意如下几点：

（1）输入参数 xList 和 yList 是列表类型，其中 xList 的一个元素就是一个列表，对应一个样本；

（2）xList 转换成 Numpy 矩阵 xMat 之后，仍然是一行一个样本、一列一个特征，这与式（3.4）要求的$\boldsymbol{X}$恰好相反，因此代码里对应的$\boldsymbol{X}$应该为 xMat.T，而$\boldsymbol{X}^T$应该为 xMat；

（3）调用 Numpy 的库函数 np.linalg.det()求 xTx 的行列式，行列式为 0 则说明 xTx 不可逆，从而无法求解；

（4）xTx.I 就是计算 xTx 的逆矩阵；

（5）函数返回权重参数 ws。

图 3.2 就是在简单数据集上运行的结果：横轴为特征，纵轴为真值或预测值。该简单数据集总共有 200 个样本点，一行就是一个样本点，下面给出了前三行的数据：

```
1.000000 0.067732 3.176513
1.000000 0.427810 3.816464
1.000000 0.995731 4.550095
```

其中，每一行的第一列固定为 1，对应增扩特征；第二列是特征；第三列是真值。

图 3.2 的绘图代码如下：

```
xList, yList=reg.loadDataSet('ex0.txt') #装入数据
ws = reg.standRegres(xList,yList) #调用基本线性回归函数
xMat=np.mat(xList)
yMat=np.mat(yList) #真值
yHat = xMat*ws #预测值

fig = plt.figure()
```

```
ax = fig.add_subplot(111)
ax.scatter(xMat[:, 1].flatten().A[0], yMat.T[:, 0].flatten().A[0]) #画出所有样本点

xCopy=xMat.copy()
xCopy.sort(0) #按列排序，方便作图
yHat=xCopy*ws
ax.plot(xCopy[:,1],yHat) #画出拟合直线

plt.xlabel('x'); plt.ylabel('y')
plt.show()
```

需要说明几点：

（1）xMat 是 200 行 2 列的矩阵，xMat[:, 1]是第 2 列，对应特征；

（2）yMat.T 是 200 行 1 列的矩阵，yMat[:, 0]是第 1 列，对应真值；

（3）xMat[:, 1].flatten().A[0]：首先把 200 行 1 列的矩阵 xMat[:, 1]变换为 1 行 200 列的矩阵 xMat[:, 1].flatten()，然后将其变换为 1 行 200 列的 Numpy 数组 xMat[:, 1].flatten().A，最后取其第 1 行 xMat[:, 1].flatten().A[0]；

（4）yMat.T[:, 0].flatten().A[0]：类似（3），需要注意 yMat.T 只有 1 列。

图 3.2 中拟合出的斜直线就是基本线性回归的模型，因此对于新的测试数据，输入特征，就可以得到预测值。

另外，如何定量地评估回归的质量呢？除了 1.2.3 节给出的MAE和MSE这些（MSE其实就对应二次损失函数），还可以采用（皮尔逊）相关系数进行评估，Numpy 对应的库函数为 corrcoef()。

3.5.2 局部加权线性回归

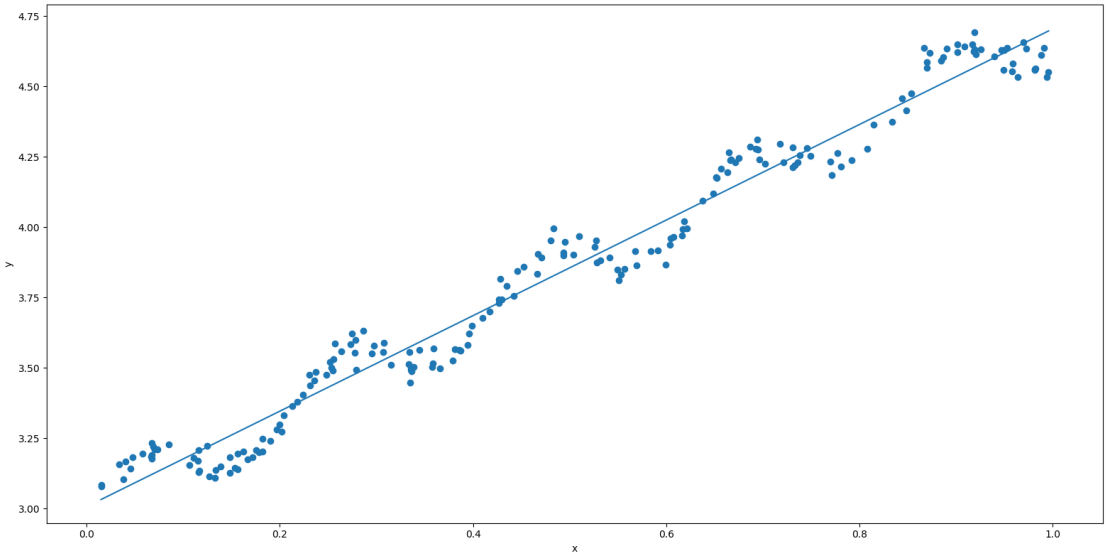
```
def lwlr(testPoint, xList, yList, k=1.0):
    xMat = np.mat(xList); yMat = np.mat(yList).T
    n = np.shape(xMat)[0]
    weights = np.mat(np.eye((n)))
    for j in range(n): #计算权重矩阵
        diffMat = testPoint - xMat[j, :]
        weights[j, j] = np.exp(diffMat * diffMat.T / (-2.0 * k**2))
    xTx = xMat.T * (weights * xMat)
    if np.linalg.det(xTx) == 0.0:
        print("This matrix is singular, cannot do inverse")
        return
    ws = xTx.I * (xMat.T * (weights * yMat))
    return testPoint * ws
```

这个函数稍微复杂点，主要是式（3.9）中增加的加权矩阵$\boldsymbol{W}$。首先得到样本个数 n，然后将加权矩阵$\boldsymbol{W}$初始化为 n 阶单位阵 weights。weights 的主对角元素根据式（3.8）进行设置，其中函数参数 testPoint 对应$\boldsymbol{x}$，xMat[j, :]对应$\boldsymbol{x}^{(i)}$，diffMat * diffMat.T 对应$\left\|\boldsymbol{x}^{(i)}-\boldsymbol{x}\right\|^2$，函数默认参数 k=1.0 对应$\sigma$。得到 weights 后，后面就是套式（3.9）。注意，函数的返回值是 testPoint * ws。

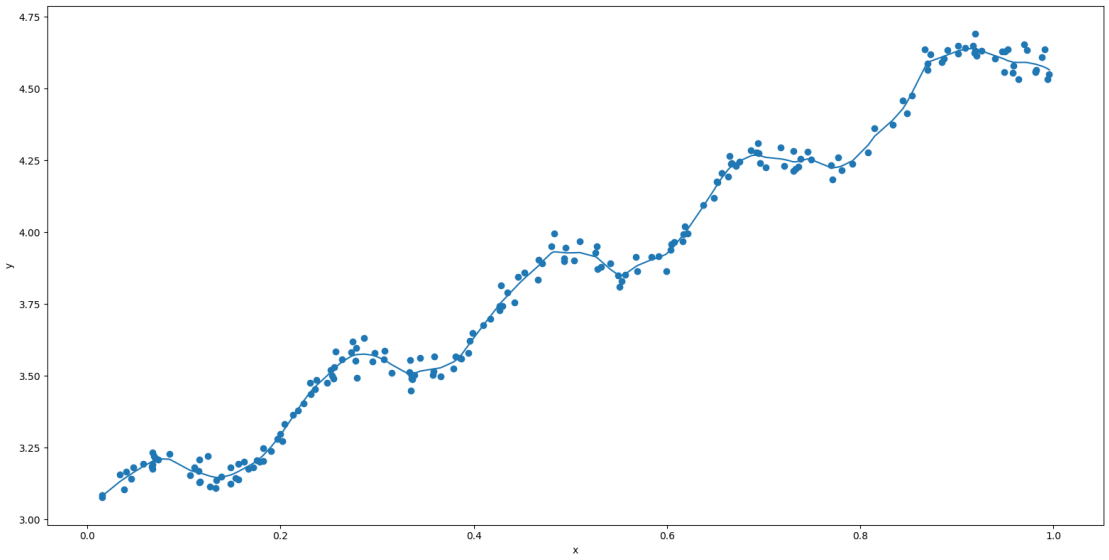
```
def lwlrTestPlot(xList, yList, k=1.0):
    yHat = np.zeros(np.shape(yList))
    xCopy = np.mat(xList)
    xCopy.sort(0)
    for i in range(np.shape(xList)[0]):
        yHat[i] = lwlr(xCopy[i], xList, yList, k)
    return yHat, xCopy
```

这个函数调用 lwlr()函数，完成对每个样本点 xCopy[i]预测值 yHat[i]的计算。注意 yHat = np.zeros(np.shape(yList))这一句将 yHat 初始化为形状与 yList 相同、初值为 0 的 Numpy 数组。这个函数返回 yHat 和 xCopy，用其就可以方便地通过语句 ax.plot(xCopy[:,1],yHat)画出拟合直线。

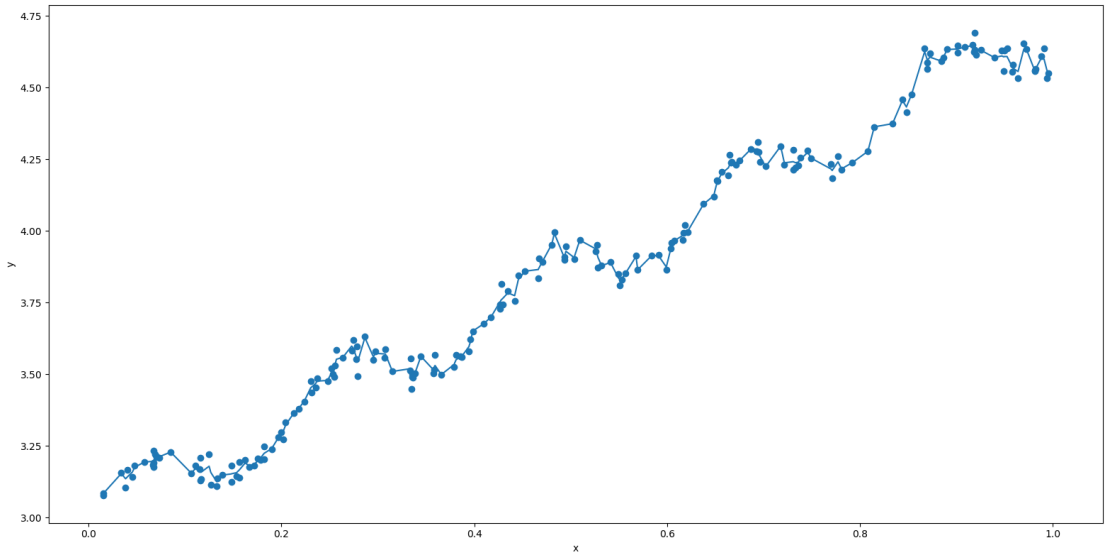
图 3.6 给出了不同 k 值的拟合结果，由图可见：k =1.0 时，拟合结果与基本线性回归（见图 3.2）几乎没有差异，模型处于欠拟合的状态；k = 0.01 时，拟合结果则很好地反映出了数据的局部起伏，同时整体上也较好地反映出了线性增长的趋势，模型处于最佳拟合的状态；而 k = 0.003 时，拟合结果被数据点的随机起伏（噪声）所主导，模型已经处于过拟合的状态。这与 3.3 节给出的理论分析是一致的，即σ决定模型的复杂程度，σ越大，则模型越简单；反之，则模型越复杂。



（a）k = 1.0



（b）k = 0.01



（c）k = 0.003

图 3.6 不同 k 值的拟合结果

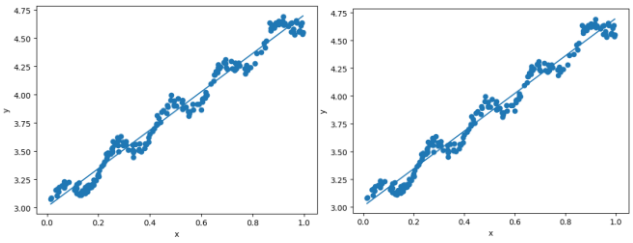
局部加权线性回归虽然能够很好地拟合数据的局部起伏，但是其计算复杂度也更高：对于每个测试数据点，都需要与所有训练样本点进行计算，计算复杂度为$O(N)$。而基本线性回归对于测试过程，则无需与训练样本点进行计算，计算复杂度可视为$O(1)$。当然，对于较小的 k 值（比如 k = 0.01），很多加权值 weights[j, j]都为 0，可以据此减少很大部分计算量。

3.5.3 岭回归

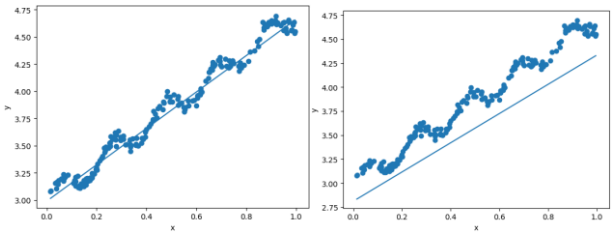
```
def ridgeRegres(xList, yList, lam=0.2):
    xMat = np.mat(xList); yMat = np.mat(yList).T
    xTx = xMat.T * xMat
    denom = xTx + np.eye(np.shape(xMat)[1]) * lam
    if np.linalg.det(denom) == 0.0:
```

```
            print("This matrix is singular, cannot do inverse")
            return
        ws = denom.I * (xMat.T * yMat)
        return ws
```

这个函数也很简单，套式（3.5），得到权重参数 ws。函数的默认参数 lam=0.2 就是式（3.5）中的正则化系数λ。为了看到 lam 对拟合结果的影响，将其依次设为 0.02、0.2、2.0、20.0 四个不同的值，在简单数据集上进行测试。图 3.7 给出了不同 lam 值的拟合结果。



（a）lam=0.02　　（b）lam=0.2



（c）lam=2.0　　（d）lam=20.0

图 3.7 不同 lam 值的拟合结果

由图 3.7 可知，lam 越小，则经验风险项越重要，更倾向于对训练数据的过拟合；相反，则正则项越重要，更倾向于对训练数据的欠拟合。那么，图 3.7 中，哪个 lam 值最优呢？这可以通过 1.2.3 节介绍的MAE和MSE来进行定量评估。评估的结果表明，图 3.7 中，lam=2.0 时能够得到最小的MAE和MSE。

仔细观察图 3.7，还可以发现，随着 lam 逐渐增大，拟合直线逐渐由左上角向右下角移动。实际上，这进一步证实了：lam 越大则 L2 正则化效果越明显。正如图 3.3 所示，随着 lam 增大（t 变小），则权重参数 ws 变小，对应到图 3.7，就是拟合直线的斜率和截距变小，逐渐由左上角向右下角移动。

LASSO 回归的实现及其与岭回归的比较，将留给读者在实验部分来进一步完成。

习题 3

3.1 图 3.1 中用短实线给出的训练样本点（图中圆点所示）与其对应的拟合直线上点的距离可以取为垂直于拟合直线$y = 0.6x + 0.52$（图中长实线所示）吗？为什么？

3.2 对于回归问题同样采用二次损失函数（式（3.1））的合理性在哪里？

3.3 说明式（3.2）中向量和矩阵的形状，及其所进行的具体运算和最终结果。

（$\boldsymbol{y}$是 Nx1 矩阵，$\boldsymbol{X}$是(F+1)xN 矩阵，$\boldsymbol{w}$是(F+1)x1 矩阵，$\boldsymbol{X}^T\boldsymbol{w}$的结果为 Nx1 矩阵，所以$\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{w}$的结果为 Nx1 矩阵，即 N 维列向量，这个列向量和其自身的点积得到一个标量——即损失函数的值）

3.4 推导式（3.3），并给出式中每一项的行列数。

3.5 请用$\boldsymbol{x}^n$和y^n表达式（3.4）。

3.6 式（3.4）要求方阵$\boldsymbol{XX^T}$可逆，那么如果$\boldsymbol{XX^T}$不可逆，你能想到哪些解决办法呢？

3.7 请证实从式（3.6）推导出的最优解就是式（3.5）。

3.8 如果设真值 y 是一个服从均值为$\boldsymbol{f(x;w) = w^T x}$，方差为$\sigma^2$的高斯分布，则式（3.4）给出的最小二乘解也可以通过极大似然估计（见 2.3.3 节）得到，请给出推导过程。提示：构造关于$\boldsymbol{w}$的对数似然函数。

3.9 式（3.6）和式（3.10）的最小化问题都可以通过梯度下降来求解，请分别给出两者的梯度下降公式，并分析各自的下降速度，解释各自解的稀疏性。

3.10 图 3.2 中拟合出的斜直线就是基本线性回归的模型，因此对于新的测试数据，输入特征，就可以得到预测值。请编写测试代码，给出特征为 0.5 时的预测值。

3.11 请用（皮尔逊）相关系数对 3.5.1 节的回归质量进行评估，注意 Numpy 对应的库函数为 corrcoef()，请仔细阅读该函数的使用说明，并对该函数返回的评估结果进行分析和解释。

3.12 针对图 3.6 给出的不同 k 值拟合结果分别计算（皮尔逊）相关系数，并分析该系数的值与模型拟合状态的关系。

3. 13 编写代码，计算图 3.7 中不同 lam 值下的 MAE 和 MSE，并将其与（皮尔逊）相关系数的值进行比较，哪一个更能反映模型对数据的拟合状态？为什么？

3. 14 编写代码，计算图 3.7 中不同 lam 值下的权重参数 ws，分析并解释其变化规律。