机器学习实战教程（十一）：线性回归基础篇之预测鲍鱼年龄

摘要

前面的文章介绍了很多分类算法，分类的目标变量是标称型数据，而本文将会对连续型的数据做出预测。主要讲解简单的线性回归和局部加权线性回归，并通过预测鲍鱼年龄
战演练。

一、前言

好久没有更新了，最近中耳炎，晚上耳鸣，一度影响正常工作，慢慢吃药调理中。在学习之余，记得加强体育锻炼！

前面的文章介绍了很多分类算法，分类的目标变量是标称型数据，而本文将会对连续型的数据做出预测。主要讲解简单的线性回归和局部加权线性回
鲍鱼年龄的实例进行实战演练。

二、什么是回归？

回归的目的是预测数值型的目标值。最直接的办法是依据输入写出一个目标值的计算公式。假如你想预测小姐姐男友汽车的功率，可能会这么计算：

HorsePower = 0.0015 * annualSalary - 0.99 * hoursListeningToPublicRadio

写成中文就是：

小姐姐男友汽车的功率 = 0.0015 * 小姐姐男友年薪 - 0.99 * 收听公共广播的时间

这就是所谓的回归方程（regression equation），其中的0.0015和-0.99称为回归系数（regression weights），求这些回归系数的过程就是回归。
回归系数，再给定输入，做预测就非常容易了。具体的做法是用回归系数乘以输入值，再将结果全部加在一起，就得到了预测值。

说到回归，一般都是指线性回归（linear regression），所以本文里的回归和线性回归代表同一个意思。线性回归意味着可以将输入项分别乘以一些
加起来得到输出。需要说明的是，存在另一种成为非线性回归的回归模型，该模型不认同上面的做法，比如认为输出可能是输入的乘积。这样，上面的功
以写做：

HorsePower = 0.0015 * annualSalary / hoursListeningToPublicRadio

这就是一个非线性回归的例子，本文对此不做深入讨论。

三、揭开回归的神秘面纱

1、用线性回归找到最佳拟合直线

应该怎么从一大堆数据里求出回归方程呢？假定输入数据存放在矩阵X中，结果存放在向量y中：

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}, y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

而回归系数存放在向量w中：

$$\omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$$

那么对于给定的数据x1，即矩阵X的第一列数据，预测结果u1将会通过如下公式给出：

$$u_1 = \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \end{bmatrix}^T * \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$$

现在的问题是，手里有数据矩阵X和对应的标签向量y，怎么才能找到w呢？一个常用的方法就是找出使误差最小的w。这里的误差是指预测u值和真实值，使用该误差的简单累加将使得正差值和负差值相互抵消，所以我们采用平方误差。

平方误差和可以写做：

$$\sum_{i=1}^{m} (y_i - {x_i}^T \omega)^2$$

用矩阵表示还可以写做：

$$(y - X\omega)^T (y - X\omega)$$

为啥能这么变化，记住一个前提：若x为向量，则默认x为列向量，x^T为行向量。将上述提到的数据矩阵X和标签向量y带进去，就知道为何这么变化了

在继续推导之前，我们要先明确一个目的：找到w，使平方误差和最小。因为我们认为平方误差和越小，说明线性回归拟合效果越好。

现在，我们用矩阵表示的平方误差和对w进行求导：

$$\begin{aligned}
&\frac{\partial (y - X\omega)^T (y - X\omega)}{\partial \omega} \\
&= \frac{\partial (y^T y - y^T X\omega - \omega^T X^T y + \omega^T X^T X\omega)}{\partial \omega} \\
&= \frac{\partial y^T y}{\partial \omega} - \frac{\partial y^T X\omega}{\partial \omega} - \frac{\partial \omega^T X^T y}{\partial \omega} + \frac{\partial \omega^T X^T X\omega}{\partial \omega} \\
&= 0 - X^T \mathrm{y} - \frac{\partial (\omega^T X^T y)^T}{\partial \omega} + 2X^T X\omega \\
&= 0 - X^T \mathrm{y} - \frac{\partial y^T X\omega}{\partial \omega} + 2X^T X\omega \\
&= 0 - X^T \mathrm{y} - X^T \mathrm{y} + 2X^T X\omega \\
&= 2X^T (y - X\omega)
\end{aligned}$$

如果对于矩阵求不熟悉的，可以移步这里：点击查看

令上述公式等于0，得到：

$$\hat{\omega} = (X^T X)^{-1} X^T y$$

w上方的小标记表示，这是当前可以估计出的w的最优解。从现有数据上估计出的w可能并不是数据中的真实w值，所以这里使用了一个"帽"符号来表示个最佳估计。

值得注意的是，上述公式中包含逆矩阵，也就是说，这个方程只在逆矩阵存在的时候使用，也即是这个矩阵是一个方阵，并且其行列式不为0。

述的最佳w求解是统计学中的常见问题，除了矩阵方法外还有很多其他方法可以解决。通过调用NumPy库里的矩阵方法，我们可以仅使用几行代码就能。该方法也称作OLS， 意思是“普通小二乘法”（ordinary least squares）。

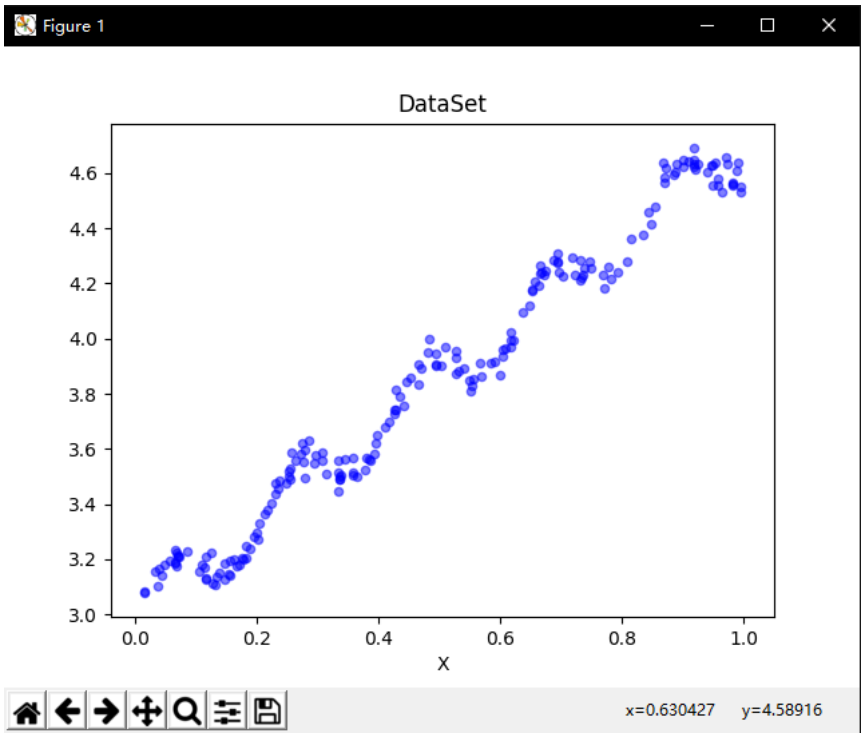
我们先看下数据集，数据下载地址：数据集下载

```
1.000000	0.067732	3.176513
1.000000	0.427810	3.816464
1.000000	0.995731	4.550095
1.000000	0.738336	4.256571
1.000000	0.981083	4.560815
1.000000	0.526171	3.929515
1.000000	0.378887	3.526170
1.000000	0.033859	3.156393
1.000000	0.132791	3.110301
1.000000	0.138306	3.149813
1.000000	0.247809	3.476346
1.000000	0.648270	4.119688
1.000000	0.731209	4.282233
1.000000	0.236833	3.486582
1.000000	0.969788	4.655492
1.000000	0.607492	3.965162
1.000000	0.358622	3.514900
1.000000	0.147846	3.125947
1.000000	0.637820	4.094115
1.000000	0.230372	3.476039
1.000000	0.070237	3.210610
1.000000	0.067154	3.190612
1.000000	0.925577	4.631504
1.000000	0.717733	4.295890
1.000000	0.015371	3.085028
1.000000	0.335070	3.448080
1.000000	0.040486	3.167440
1.000000	0.212575	3.364266
1.000000	0.617218	3.993482
1.000000	0.541196	3.891471
1.000000	0.045353	3.143259
1.000000	0.126762	3.114204
```

第一列都为1.0，即x0。第二列为x1，即x轴数据。第三列为x2，即y轴数据。首先绘制下数据，看下数据分布。编写代码如下：

```python
# -*- coding:utf-8 -*-
import matplotlib.pyplot as plt
import numpy as np

def loadDataSet(fileName):
    """
    函数说明:加载数据
    Parameters:
        fileName - 文件名
    Returns:
        xArr - x数据集
        yArr - y数据集
    Website:
        http://www.cuijiahua.com/
    Modify:
        2017-11-12
    """

    numFeat = len(open(fileName).readline().split('\t')) - 1
    xArr = []; yArr = []
    fr = open(fileName)
    for line in fr.readlines():
        lineArr =[]
        curLine = line.strip().split('\t')
        for i in range(numFeat):
            lineArr.append(float(curLine[i]))
        xArr.append(lineArr)
        yArr.append(float(curLine[-1]))
    return xArr, yArr

def plotDataSet():
    """
    函数说明:绘制数据集
    Parameters:
        无
    Returns:
        无
    Website:
        http://www.cuijiahua.com/
    Modify:
        2017-11-12
    """
    xArr, yArr = loadDataSet('ex0.txt')                                     #加载数据集
    n = len(xArr)                                                           #数据个数
    xcord = []; ycord = []                                                  #样本点
    for i in range(n):
        xcord.append(xArr[i][1]); ycord.append(yArr[i])                     #样本点
    fig = plt.figure()
    ax = fig.add_subplot(111)                                               #添加subplot
    ax.scatter(xcord, ycord, s = 20, c = 'blue',alpha = .5)                 #绘制样本点
    plt.title('DataSet')                                                    #绘制title
    plt.xlabel('X')
    plt.show()

if __name__ == '__main__':
    plotDataSet()
```

运行代码如下：



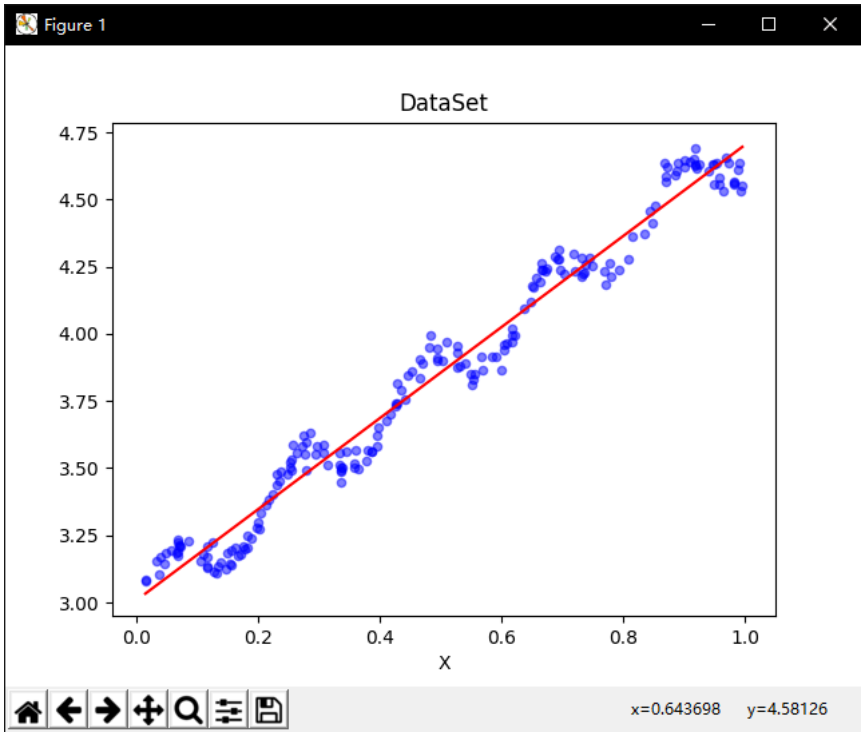
通过可视化数据，我们可以看到数据的分布情况。接下来，让我们根据上文中推导的回归系数计算方法，求出回归系数向量，并根据回归系数向量绘

写代码如下：

```
# -*- coding:utf-8 -*-
import matplotlib.pyplot as plt
import numpy as np

def loadDataSet(fileName):
    """
    函数说明:加载数据
    Parameters:
        fileName - 文件名
    Returns:
        xArr - x数据集
        yArr - y数据集
    Website:
        http://www.cuijiahua.com/
    Modify:
        2017-11-12
    """
    numFeat = len(open(fileName).readline().split('\t')) - 1
    xArr = []; yArr = []
    fr = open(fileName)
    for line in fr.readlines():
        lineArr =[]
        curLine = line.strip().split('\t')
        for i in range(numFeat):
            lineArr.append(float(curLine[i]))
        xArr.append(lineArr)
        yArr.append(float(curLine[-1]))
    return xArr, yArr

def standRegres(xArr,yArr):
    """
    函数说明:计算回归系数w
    Parameters:
        xArr - x数据集
        yArr - y数据集
    Returns:
        ws - 回归系数
    Website:
        http://www.cuijiahua.com/
    Modify:
        2017-11-12
    """
    xMat = np.mat(xArr); yMat = np.mat(yArr).T
    xTx = xMat.T * xMat                             #根据文中推导的公示计算回归系数
    if np.linalg.det(xTx) == 0.0:
        print("矩阵为奇异矩阵,不能求逆")
        return
    ws = xTx.I * (xMat.T*yMat)
    return ws

def plotRegression():
    """
    函数说明:绘制回归曲线和数据点
    Parameters:
        无
    Returns:
        无
    Website:
        http://www.cuijiahua.com/
```

```
    Modify:
        2017-11-12
    """
    xArr, yArr = loadDataSet('ex0.txt')                                    #加载数据集
    ws = standRegres(xArr, yArr)                                          #计算回归系数
    xMat = np.mat(xArr)                                                    #创建xMat矩阵
    yMat = np.mat(yArr)                                                    #创建yMat矩阵
    xCopy = xMat.copy()                                                    #深拷贝xMat矩阵
    xCopy.sort(0)                                                         #排序
    yHat = xCopy * ws                                                      #计算对应的y值
    fig = plt.figure()
    ax = fig.add_subplot(111)                                             #添加subplot
    ax.plot(xCopy[:, 1], yHat, c = 'red')                                 #绘制回归曲线
    ax.scatter(xMat[:,1].flatten().A[0], yMat.flatten().A[0], s = 20, c = 'blue',alpha = .5)                #绘制样本点
    plt.title('DataSet')                                                  #绘制title
    plt.xlabel('X')
    plt.show()

if __name__ == '__main__':
    plotRegression()
```

运行代码如下：



如何判断拟合曲线的拟合效果的如何呢？当然，我们可以根据自己的经验进行观察，除此之外，我们还可以使用corrcoef方法，来比较预测值和真实值

写代码如下：

```
# -*- coding:utf-8 -*-
import numpy as np

def loadDataSet(fileName):
    """
    函数说明:加载数据
    Parameters:
        fileName - 文件名
    Returns:
        xArr - x数据集
        yArr - y数据集
    Website:
        http://www.cuijiahua.com/
    Modify:
        2017-11-12
    """
    numFeat = len(open(fileName).readline().split('\t')) - 1
    xArr = []; yArr = []
    fr = open(fileName)
    for line in fr.readlines():
        lineArr =[]
        curLine = line.strip().split('\t')
        for i in range(numFeat):
            lineArr.append(float(curLine[i]))
        xArr.append(lineArr)
        yArr.append(float(curLine[-1]))
    return xArr, yArr

def standRegres(xArr,yArr):
    """
    函数说明:计算回归系数w
    Parameters:
        xArr - x数据集
        yArr - y数据集
    Returns:
        ws - 回归系数
    Website:
        http://www.cuijiahua.com/
```

```
    Modify:
        2017-11-12
    """
    xMat = np.mat(xArr); yMat = np.mat(yArr).T
    xTx = xMat.T * xMat                     #根据文中推导的公示计算回归系数
    if np.linalg.det(xTx) == 0.0:
        print("矩阵为奇异矩阵,不能求逆")
        return
    ws = xTx.I * (xMat.T*yMat)
    return ws

if __name__ == '__main__':
    xArr, yArr = loadDataSet('ex0.txt')                 #加载数据集
    ws = standRegres(xArr, yArr)                        #计算回归系数
    xMat = np.mat(xArr)                                 #创建xMat矩阵
    yMat = np.mat(yArr)                                 #创建yMat矩阵
    yHat = xMat * ws
    print(np.corrcoef(yHat.T, yMat))
```

运行结果如下：

```
if __name__ == '__main__':
    xArr, yArr = loadDataSet('ex0.txt')
    ws = standRegres(xArr, yArr)
    xMat = np.mat(xArr)
    yMat = np.mat(yArr)
    yHat = xMat * ws
    print(np.corrcoef(yHat.T, yMat))
```

```
[[ 1.          0.98647356]
 [ 0.98647356  1.        ]]
[Finished in 1.8s]
```

可以看到，对角线上的数据是1.0，因为yMat和自己的匹配是完美的，而YHat和yMat的相关系数为0.98。

最佳拟合直线方法将数据视为直线进行建模，具有十分不错的表现。数据当中似乎还存在其他的潜在模式。那么如何才能利用这些模式呢？我们可以修调整预测，下面就会介绍这种方法。

2、局部加权线性回归

线性回归的一个问题是有可能出现欠拟合现象，因为它求的是具有小均方误差的无偏估 计。显而易见，如果模型欠拟合将不能取得好的预测效果。所以许在估计中引入一 些偏差，从而降低预测的均方误差。

其中的一个方法是局部加权线性回归（Locally Weighted Linear Regression，LWLR）。在该方法中，我们给待预测点附近的每个点赋予一定的权重样，这种算法每次预测均需要事先选取出对应的数据子集。该算法解除回归系数W的形式如下：

$$\hat{\omega} = (X^T W X)^{-1} X^T W y$$

其中W是一个矩阵，这个公式跟我们上面推导的公式的区别就在于W，它用来给每个店赋予权重。

LWLR使用"核"（与支持向量机中的核类似）来对附近的点赋予更高的权重。核的类型可以自由选择，最常用的核就是高斯核，高斯核对应的权重如下

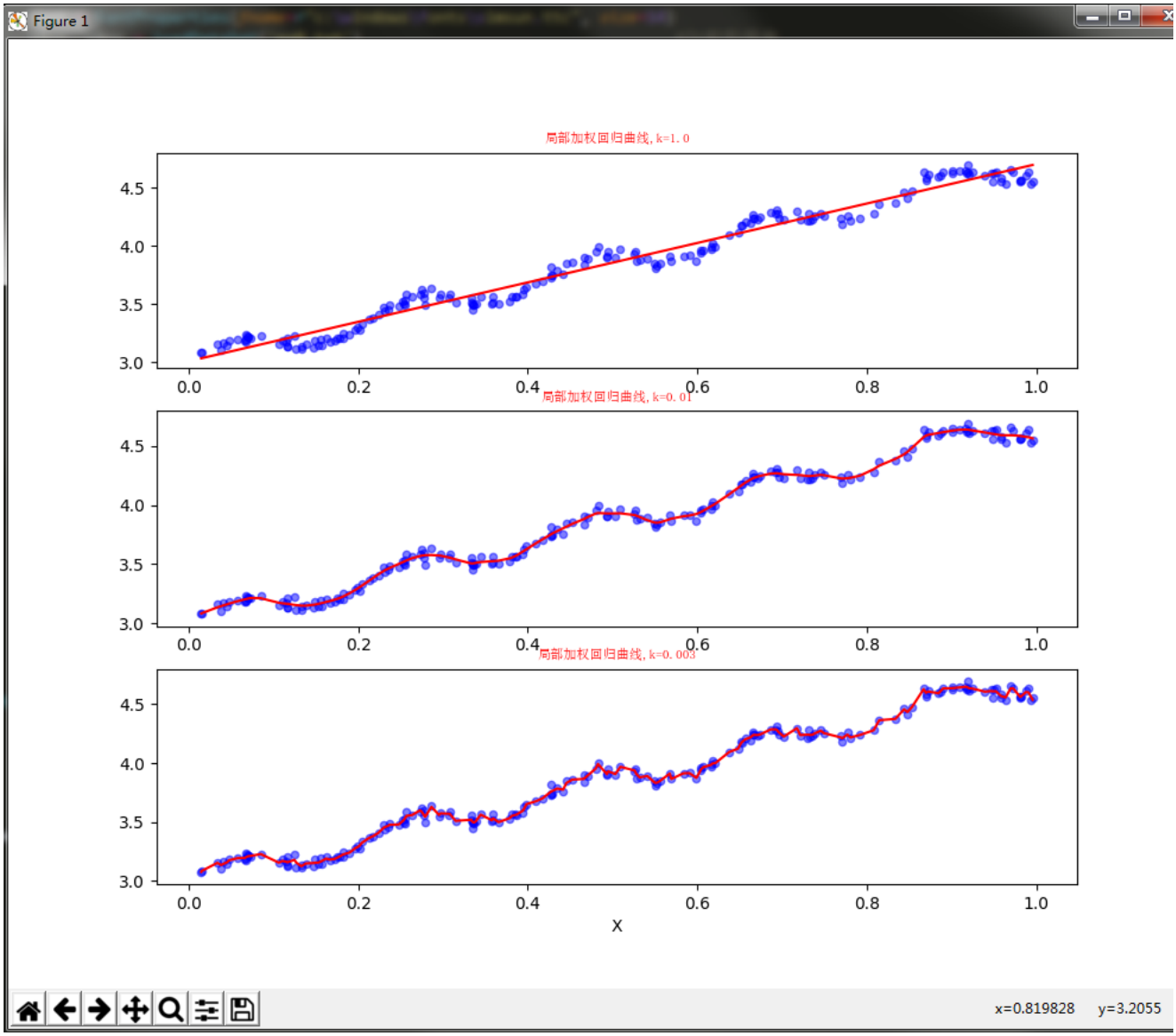
$$w(i,i) = exp\left(\frac{|x^{(i)} - x|}{-2k^2}\right)$$

这样我们就可以根据上述公式，编写局部加权线性回归，我们通过改变k的值，可以调节回归效果，编写代码如下：

```
# -*- coding:utf-8 -*-
from matplotlib.font_manager import FontProperties
import matplotlib.pyplot as plt
import numpy as np
def loadDataSet(fileName):
    """
    函数说明:加载数据
    Parameters:
        fileName - 文件名
    Returns:
        xArr - x数据集
        yArr - y数据集
    Website:http://www.cuijiahua.com/
    Modify:
        2017-11-12
    """
    numFeat = len(open(fileName).readline().split('\t')) - 1
    xArr = []; yArr = []
    fr = open(fileName)
    for line in fr.readlines():
        lineArr =[]
        curLine = line.strip().split('\t')
        for i in range(numFeat):
            lineArr.append(float(curLine[i]))
        xArr.append(lineArr)
```

```
        yArr.append(float(curLine[-1]))
    return xArr, yArr

def plotlwlrRegression():
    """
    函数说明:绘制多条局部加权回归曲线
    Parameters:
        无
    Returns:
        无
    Website:http://www.cuijiahua.com/
    Modify:
        2017-11-15
    """
    font = FontProperties(fname=r"c:\windows\fonts\simsun.ttc", size=14)
    xArr, yArr = loadDataSet('ex0.txt')                                     #加载数据集
    yHat_1 = lwlrTest(xArr, xArr, yArr, 1.0)                               #根据局部加权线性回归计算yHat
    yHat_2 = lwlrTest(xArr, xArr, yArr, 0.01)                               #根据局部加权线性回归计算yHat
    yHat_3 = lwlrTest(xArr, xArr, yArr, 0.003)                              #根据局部加权线性回归计算yHat
    xMat = np.mat(xArr)                                                     #创建xMat矩阵
    yMat = np.mat(yArr)                                                     #创建yMat矩阵
    srtInd = xMat[:, 1].argsort(0)                                          #排序，返回索引值
    xSort = xMat[srtInd][:,0,:]
    fig, axs = plt.subplots(nrows=3, ncols=1,sharex=False, sharey=False, figsize=(10,8))
    axs[0].plot(xSort[:, 1], yHat_1[srtInd], c = 'red')                     #绘制回归曲线
    axs[1].plot(xSort[:, 1], yHat_2[srtInd], c = 'red')                     #绘制回归曲线
    axs[2].plot(xSort[:, 1], yHat_3[srtInd], c = 'red')                     #绘制回归曲线
    axs[0].scatter(xMat[:,1].flatten().A[0], yMat.flatten().A[0], s = 20, c = 'blue', alpha = .5)                #绘制样本点
    axs[1].scatter(xMat[:,1].flatten().A[0], yMat.flatten().A[0], s = 20, c = 'blue', alpha = .5)                #绘制样本点
    axs[2].scatter(xMat[:,1].flatten().A[0], yMat.flatten().A[0], s = 20, c = 'blue', alpha = .5)                #绘制样本点
    #设置标题,x轴label,y轴label
    axs0_title_text = axs[0].set_title(u'局部加权回归曲线,k=1.0',FontProperties=font)
    axs1_title_text = axs[1].set_title(u'局部加权回归曲线,k=0.01',FontProperties=font)
    axs2_title_text = axs[2].set_title(u'局部加权回归曲线,k=0.003',FontProperties=font)
    plt.setp(axs0_title_text, size=8, weight='bold', color='red')
    plt.setp(axs1_title_text, size=8, weight='bold', color='red')
    plt.setp(axs2_title_text, size=8, weight='bold', color='red')
    plt.xlabel('X')
    plt.show()
def lwlr(testPoint, xArr, yArr, k = 1.0):
    """
    函数说明:使用局部加权线性回归计算回归系数w
    Parameters:
        testPoint - 测试样本点
        xArr - x数据集
        yArr - y数据集
        k - 高斯核的k,自定义参数
    Returns:
        ws - 回归系数
    Website:http://www.cuijiahua.com/
    Modify:
        2017-11-15
    """
    xMat = np.mat(xArr); yMat = np.mat(yArr).T
    m = np.shape(xMat)[0]
    weights = np.mat(np.eye((m)))                                           #创建权重对角矩阵
    for j in range(m):                                                      #遍历数据集计算每个样本的权重
        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("矩阵为奇异矩阵,不能求逆")
        return
    ws = xTx.I * (xMat.T * (weights * yMat))                                #计算回归系数
    return testPoint * ws
def lwlrTest(testArr, xArr, yArr, k=1.0):
    """
    函数说明:局部加权线性回归测试
    Parameters:
        testArr - 测试数据集
        xArr - x数据集
        yArr - y数据集
        k - 高斯核的k,自定义参数
    Returns:
        ws - 回归系数
    Website:
http://www.cuijiahua.com/
    Modify:
        2017-11-15
    """
    m = np.shape(testArr)[0]                                                #计算测试数据集大小
    yHat = np.zeros(m)
    for i in range(m):                                                        #对每个样本点进行预测
        yHat[i] = lwlr(testArr[i],xArr,yArr,k)
    return yHat
if __name__ == '__main__':
    plotlwlrRegression()
```

运行结果如下：



可以看到，当k越小，拟合效果越好。但是当k过小，会出现过拟合的情况，例如k等于0.003的时候。

四、预测鲍鱼的年龄

接下来，我们将回归用于真是数据。在abalone.txt文件中记录了鲍鱼（一种水生物→__→）的年龄，这个数据来自UCI数据集合的数据。鲍鱼年龄可以数推算得到。

数据集下载地址：数据集下载

数据集的数据如下：

```
1   1   0.455   0.365   0.095   0.514   0.2245  0.101   0.15    15
2   1   0.35    0.265   0.09    0.2255  0.0995  0.0485  0.07    7
3   -1  0.53    0.42    0.135   0.677   0.2565  0.1415  0.21    9
4   1   0.44    0.365   0.125   0.516   0.2155  0.114   0.155   10
5   0   0.33    0.255   0.08    0.205   0.0895  0.0395  0.055   7
6   0   0.425   0.3     0.095   0.3515  0.141   0.0775  0.12    8
7   -1  0.53    0.415   0.15    0.7775  0.237   0.1415  0.33    20
8   -1  0.545   0.425   0.125   0.768   0.294   0.1495  0.26    16
9   1   0.475   0.37    0.125   0.5095  0.2165  0.1125  0.165   9
10  -1  0.55    0.44    0.15    0.8945  0.3145  0.151   0.32    19
11  -1  0.525   0.38    0.14    0.6065  0.194   0.1475  0.21    14
12  1   0.43    0.35    0.11    0.406   0.1675  0.081   0.135   10
13  1   0.49    0.38    0.135   0.5415  0.2175  0.095   0.19    11
14  -1  0.535   0.405   0.145   0.6845  0.2725  0.171   0.205   10
15  -1  0.47    0.355   0.1     0.4755  0.1675  0.0805  0.185   10
16  1   0.5     0.4     0.13    0.6645  0.258   0.133   0.24    12
17  0   0.355   0.28    0.085   0.2905  0.095   0.0395  0.115   7
18  -1  0.44    0.34    0.1     0.451   0.188   0.087   0.13    10
19  1   0.365   0.295   0.08    0.2555  0.097   0.043   0.1     7
20  1   0.45    0.32    0.1     0.381   0.1705  0.075   0.115   9
21  1   0.355   0.28    0.095   0.2455  0.0955  0.062   0.075   11
22  0   0.38    0.275   0.1     0.2255  0.08    0.049   0.085   10
23  -1  0.565   0.44    0.155   0.9395  0.4275  0.214   0.27    12
24  -1  0.55    0.415   0.135   0.7635  0.318   0.21    0.2     9
25  -1  0.615   0.48    0.165   1.1615  0.513   0.301   0.305   10
26  -1  0.56    0.44    0.14    0.9285  0.3825  0.188   0.3     11
27  -1  0.58    0.45    0.185   0.9955  0.3945  0.272   0.285   11
28  1   0.59    0.445   0.14    0.931   0.356   0.234   0.28    12
29  1   0.605   0.475   0.18    0.9365  0.394   0.219   0.295   15
30  1   0.575   0.425   0.14    0.8635  0.393   0.227   0.2     11
31  1   0.58    0.47    0.165   0.9975  0.3935  0.242   0.33    10
32  -1  0.68    0.56    0.165   1.639   0.6055  0.2805  0.46    15
33  1   0.665   0.525   0.165   1.338   0.5515  0.3575  0.35    18
34  -1  0.68    0.55    0.175   1.798   0.815   0.3925  0.455   19
```

可以看到，数据集是多维的，所以我们很难画出它的分布情况。每个维度数据的代表的含义没有给出，不过没有关系，我们只要知道最后一列的数据是了，最后一列代表的是鲍鱼的真实年龄，前面几列的数据是一些鲍鱼的特征，例如鲍鱼壳的层数等。我们不做数据清理，直接用上所有特征，测试下我们的归。

新建abalone.py文件，添加rssError函数，用于评价最后回归结果。编写代码如下：

```
# -*- coding:utf-8 -*-
from matplotlib.font_manager import FontProperties
import matplotlib.pyplot as plt
import numpy as np
def loadDataSet(fileName):
    """
    函数说明:加载数据
    Parameters:
        fileName - 文件名
    Returns:
        xArr - x数据集
        yArr - y数据集
    Website:
http://www.cuijiahua.com/
    Modify:
        2017-11-19
    """
    numFeat = len(open(fileName).readline().split('\t')) - 1
    xArr = []; yArr = []
    fr = open(fileName)
    for line in fr.readlines():
        lineArr =[]
        curLine = line.strip().split('\t')
        for i in range(numFeat):
            lineArr.append(float(curLine[i]))
        xArr.append(lineArr)
        yArr.append(float(curLine[-1]))
    return xArr, yArr
def lwlr(testPoint, xArr, yArr, k = 1.0):
    """
    函数说明:使用局部加权线性回归计算回归系数w
    Parameters:
        testPoint - 测试样本点
        xArr - x数据集
        yArr - y数据集
        k - 高斯核的k,自定义参数
    Returns:
        ws - 回归系数
    Website:
http://www.cuijiahua.com/
    Modify:
        2017-11-19
    """
    xMat = np.mat(xArr); yMat = np.mat(yArr).T
    m = np.shape(xMat)[0]
    weights = np.mat(np.eye((m)))                                   #创建权重对角矩阵
    for j in range(m):                                              #遍历数据集计算每个样本的权重
        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("矩阵为奇异矩阵,不能求逆")
        return
    ws = xTx.I * (xMat.T * (weights * yMat))                    #计算回归系数
    return testPoint * ws
def lwlrTest(testArr, xArr, yArr, k=1.0):
    """
    函数说明:局部加权线性回归测试
    Parameters:
        testArr - 测试数据集,测试集
        xArr - x数据集,训练集
        yArr - y数据集,训练集
        k - 高斯核的k,自定义参数
    Returns:
        ws - 回归系数
    Website:
http://www.cuijiahua.com/
    Modify:
        2017-11-19
    """
    m = np.shape(testArr)[0]                                    #计算测试数据集大小
    yHat = np.zeros(m)
    for i in range(m):                                            #对每个样本点进行预测
        yHat[i] = lwlr(testArr[i],xArr,yArr,k)
    return yHat
def standRegres(xArr,yArr):
    """
    函数说明:计算回归系数w
    Parameters:
        xArr - x数据集
        yArr - y数据集
    Returns:
        ws - 回归系数
    Website:
http://www.cuijiahua.com/
    Modify:
        2017-11-19
    """
    xMat = np.mat(xArr); yMat = np.mat(yArr).T
    xTx = xMat.T * xMat                                 #根据文中推导的公示计算回归系数
    if np.linalg.det(xTx) == 0.0:
        print("矩阵为奇异矩阵,不能求逆")
        return
    ws = xTx.I * (xMat.T*yMat)
    return ws
def rssError(yArr, yHatArr):
    """
    误差大小评价函数
    Parameters:
        yArr - 真实数据
        yHatArr - 预测数据
    Returns:
        误差大小
    """
    return ((yArr - yHatArr) **2).sum()
if __name__ == '__main__':
    abX, abY = loadDataSet('abalone.txt')
    print('训练集与测试集相同:局部加权线性回归,核k的大小对预测的影响:')
    yHat01 = lwlrTest(abX[0:99], abX[0:99], abY[0:99], 0.1)
    yHat1 = lwlrTest(abX[0:99], abX[0:99], abY[0:99], 1)
    yHat10 = lwlrTest(abX[0:99], abX[0:99], abY[0:99], 10)
    print('k=0.1时,误差大小为:',rssError(abY[0:99], yHat01.T))
    print('k=1  时,误差大小为:',rssError(abY[0:99], yHat1.T))
    print('k=10 时,误差大小为:',rssError(abY[0:99], yHat10.T))
    print('')
    print('训练集与测试集不同:局部加权线性回归,核k的大小是越小越好吗？更换数据集,测试结果如下:')
    yHat01 = lwlrTest(abX[100:199], abX[0:99], abY[0:99], 0.1)
    yHat1 = lwlrTest(abX[100:199], abX[0:99], abY[0:99], 1)
    yHat10 = lwlrTest(abX[100:199], abX[0:99], abY[0:99], 10)
    print('k=0.1时,误差大小为:',rssError(abY[100:199], yHat01.T))
    print('k=1  时,误差大小为:',rssError(abY[100:199], yHat1.T))
    print('k=10 时,误差大小为:',rssError(abY[100:199], yHat10.T))
    print('')
    print('训练集与测试集不同:简单的线性归回与k=1时的局部加权线性回归对比:')
    print('k=1时,误差大小为:', rssError(abY[100:199], yHat1.T))
    ws = standRegres(abX[0:99], abY[0:99])
    yHat = np.mat(abX[100:199]) * ws
    print('简单的线性回归误差大小:', rssError(abY[100:199], yHat.T.A))
```

运行结果如下：

```
    print('k=0.1时,误差大小为:',rssError(abY[100:199], yHat01.T))
    print('k=1  时,误差大小为:',rssError(abY[100:199], yHat1.T))
    print('k=10 时,误差大小为:',rssError(abY[100:199], yHat10.T))

    print('')

    print('训练集与测试集不同:简单的线性归回与k=1时的局部加权线性回归对比:')
    print('k=1时,误差大小为:', rssError(abY[100:199], yHat1.T))
    ws = standRegres(abX[0:99], abY[0:99])
    yHat = np.mat(abX[100:199]) * ws
    print('简单的线性回归误差大小:', rssError(abY[100:199], yHat.T.A))
```

```
训练集与测试集相同:局部加权线性回归,核k的大小对预测的影响:
k=0.1时,误差大小为: 56.8121847316
k=1  时,误差大小为: 429.89056187
k=10 时,误差大小为: 549.118170883

训练集与测试集不同:局部加权线性回归,核k的大小是越小越好吗？更换数据集,测试结果如下:
k=0.1时,误差大小为: 347136.599167
k=1  时,误差大小为: 573.52614419
k=10 时,误差大小为: 517.571190538

训练集与测试集不同:简单的线性归回与k=1时的局部加权线性回归对比:
k=1时,误差大小为: 573.52614419
简单的线性回归误差大小: 518.636315325
[Finished in 2.5s]
```

可以看到，当k=0.1时，训练集误差小，但是应用于新的数据集之后，误差反而变大了。这就是经常说道的过拟合现象。我们训练的模型，我们要保证
高，这样训练出的模型才可以应用于新的数据，也就是要加强模型的普适性。可以看到，当k=1时，局部加权线性回归和简单的线性回归得到的效果差不多
点，必须在未知数据上比较效果才能选取到最佳模型。那么最佳的核大小是10吗？或许是，但如果想得到更好的效果，应该用10个不同的样本集做10次测
果。

本示例展示了如何使用局部加权线性回归来构建模型，可以得到比普通线性回归更好的效果。局部加权线性回归的问题在于，每次必须在整个数据集上
说为了做出预测，必须保存所有的训练数据。

五、总结

- 本文主要介绍了简单的线性回归和局部加权线性回归。
- 训练的模型要在测试集比较它们的效果，而不是在训练集上。
- 在局部加权线性回归中，过小的核可能导致过拟合现象，即训练集表现良好，测试集表现就渣渣了。
- 下篇文章将继续讲解回归，会介绍另一种提高预测精度的方法。
- 如有问题，请留言。如有错误，还望指正，谢谢！

PS： 如果觉得本篇本章对您有所帮助，欢迎关注、评论、赞！

本文出现的所有代码和数据集，均可在我的github上下载，欢迎Follow、Star：点击查看

参考资料：

- [1] 机器学习实战第八章内容

JackCui

关注人工智能及互联网的个人博客

查看熊掌号