模型-机器学习-回归-局部加权线性回归【hxy】

1. 模型名称

局部加权线性回归（Locally Weighted Linear Regression，LWLR）

2. 模型评价

2.1 优点

- 实现简单，容易理解
- 理论上可以拟合任意的连续曲线

2.2 缺点

- 每次都要计算权值，效率很低

3. 基本算法

局部加权线性回归的目标函数为

$$J(\theta)=\sum_i w^{(i)}(y^{(i)}-\theta^T x^{(i)})^2$$

$w^{(i)}=w(i,i)$为权重矩阵，$y^{(i)}$为因变量矩阵，$\theta^{(i)}$为系数矩阵，$x^{(i)}$为自变量矩阵

对θ求导并令导数为零可以得到：

$$\theta=(X^TWX)^{-1}X^TWy$$

其中，$LWLR$使用的核可以自由选择，最常用的核就是高斯核，其对应的权重为：

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

此权重是一个均值为x，标准差为k的高斯函数，故与测试样本x越近的样本点，能够得到更高的权重

4. 实例

4.1 数据介绍

第1列和第2列是自变量矩阵，第3列是因变量矩阵

```
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
1.000000  0.556486  3.851484
1.000000  0.901144  4.621899
1.000000  0.958476  4.580768
1.000000  0.274561  3.620992
1.000000  0.394396  3.580501
1.000000  0.872480  4.618706
1.000000  0.409932  3.676867
1.000000  0.908969  4.641845
1.000000  0.166819  3.175939
1.000000  0.665016  4.264980
1.000000  0.263727  3.558448
```

```
1.000000  0.231214  3.436632
1.000000  0.552928  3.831052
1.000000  0.047744  3.182853
1.000000  0.365746  3.498906
1.000000  0.495002  3.946833
1.000000  0.493466  3.900583
1.000000  0.792101  4.238522
1.000000  0.769660  4.233080
1.000000  0.251821  3.521557
1.000000  0.181951  3.203344
1.000000  0.808177  4.278105
1.000000  0.334116  3.555705
1.000000  0.338630  3.502661
1.000000  0.452584  3.859776
1.000000  0.694770  4.275956
1.000000  0.590902  3.916191
1.000000  0.307928  3.587961
1.000000  0.148364  3.183004
1.000000  0.702180  4.225236
1.000000  0.721544  4.231083
1.000000  0.666886  4.240544
1.000000  0.124931  3.222372
1.000000  0.618286  4.021445
1.000000  0.381086  3.567479
1.000000  0.385643  3.562580
1.000000  0.777175  4.262059
1.000000  0.116089  3.208813
1.000000  0.115487  3.169825
1.000000  0.663510  4.193949
1.000000  0.254884  3.491678
1.000000  0.993888  4.533306
1.000000  0.295434  3.550108
1.000000  0.952523  4.636427
1.000000  0.307047  3.557078
1.000000  0.277261  3.552874
1.000000  0.279101  3.494159
1.000000  0.175724  3.206828
1.000000  0.156383  3.195266
1.000000  0.733165  4.221292
1.000000  0.848142  4.413372
1.000000  0.771184  4.184347
1.000000  0.429492  3.742878
1.000000  0.162176  3.201878
1.000000  0.917064  4.648964
1.000000  0.315044  3.510117
1.000000  0.201473  3.274434
1.000000  0.297038  3.579622
1.000000  0.336647  3.489244
1.000000  0.666109  4.237386
```

```
1.000000  0.583888  3.913749
1.000000  0.085031  3.228990
1.000000  0.687006  4.286286
1.000000  0.949655  4.628614
1.000000  0.189912  3.239536
1.000000  0.844027  4.457997
1.000000  0.333288  3.513384
1.000000  0.427035  3.729674
1.000000  0.466369  3.834274
1.000000  0.550659  3.811155
1.000000  0.278213  3.598316
1.000000  0.918769  4.692514
1.000000  0.886555  4.604859
1.000000  0.569488  3.864912
1.000000  0.066379  3.184236
1.000000  0.335751  3.500796
1.000000  0.426863  3.743365
1.000000  0.395746  3.622905
1.000000  0.694221  4.310796
1.000000  0.272760  3.583357
1.000000  0.503495  3.901852
1.000000  0.067119  3.233521
1.000000  0.038326  3.105266
1.000000  0.599122  3.865544
1.000000  0.947054  4.628625
1.000000  0.671279  4.231213
1.000000  0.434811  3.791149
1.000000  0.509381  3.968271
1.000000  0.749442  4.253910
1.000000  0.058014  3.194710
1.000000  0.482978  3.996503
1.000000  0.466776  3.904358
1.000000  0.357767  3.503976
1.000000  0.949123  4.557545
1.000000  0.417320  3.699876
1.000000  0.920461  4.613614
1.000000  0.156433  3.140401
1.000000  0.656662  4.206717
1.000000  0.616418  3.969524
1.000000  0.853428  4.476096
1.000000  0.133295  3.136528
1.000000  0.693007  4.279071
1.000000  0.178449  3.200603
1.000000  0.199526  3.299012
1.000000  0.073224  3.209873
1.000000  0.286515  3.632942
1.000000  0.182026  3.248361
1.000000  0.621523  3.995783
1.000000  0.344584  3.563262
```

```
1.000000  0.398556  3.649712
1.000000  0.480369  3.951845
1.000000  0.153350  3.145031
1.000000  0.171846  3.181577
1.000000  0.867082  4.637087
1.000000  0.223855  3.404964
1.000000  0.528301  3.873188
1.000000  0.890192  4.633648
1.000000  0.106352  3.154768
1.000000  0.917886  4.623637
1.000000  0.014855  3.078132
1.000000  0.567682  3.913596
1.000000  0.068854  3.221817
1.000000  0.603535  3.938071
1.000000  0.532050  3.880822
1.000000  0.651362  4.176436
1.000000  0.901225  4.648161
1.000000  0.204337  3.332312
1.000000  0.696081  4.240614
1.000000  0.963924  4.532224
1.000000  0.981390  4.557105
1.000000  0.987911  4.610072
1.000000  0.990947  4.636569
1.000000  0.736021  4.229813
1.000000  0.253574  3.500860
1.000000  0.674722  4.245514
1.000000  0.939368  4.605182
1.000000  0.235419  3.454340
1.000000  0.110521  3.180775
1.000000  0.218023  3.380820
1.000000  0.869778  4.565020
1.000000  0.196830  3.279973
1.000000  0.958178  4.554241
1.000000  0.972673  4.633520
1.000000  0.745797  4.281037
1.000000  0.445674  3.844426
1.000000  0.470557  3.891601
1.000000  0.549236  3.849728
1.000000  0.335691  3.492215
1.000000  0.884739  4.592374
1.000000  0.918916  4.632025
1.000000  0.441815  3.756750
1.000000  0.116598  3.133555
1.000000  0.359274  3.567919
1.000000  0.814811  4.363382
1.000000  0.387125  3.560165
1.000000  0.982243  4.564305
1.000000  0.780880  4.215055
1.000000  0.652565  4.174999
```

```
1.000000  0.870030  4.586640
1.000000  0.604755  3.960008
1.000000  0.255212  3.529963
1.000000  0.730546  4.213412
1.000000  0.493829  3.908685
1.000000  0.257017  3.585821
1.000000  0.833735  4.374394
1.000000  0.070095  3.213817
1.000000  0.527070  3.952681
1.000000  0.116163  3.129283
```

4.2 实验目的

探究不同k选择下拟合结果

4.3 代码实现

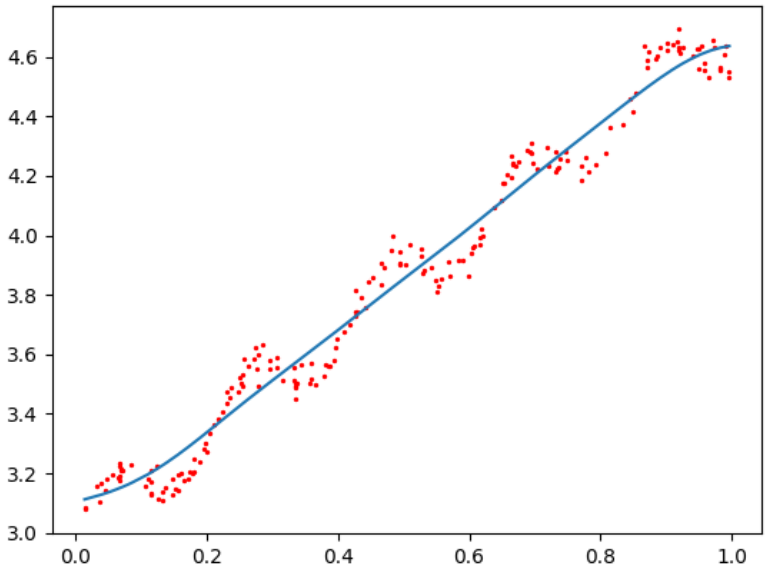
LWLR.py

代码：

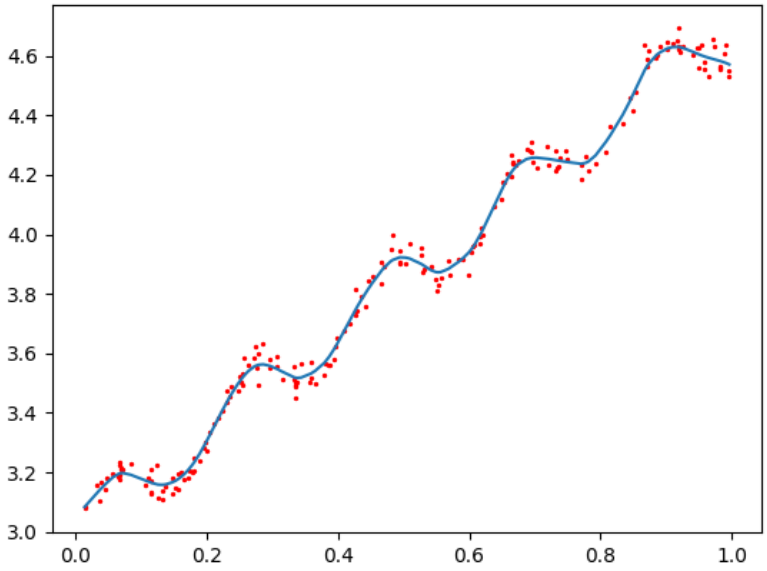
```
from numpy import *
import matplotlib.pyplot as plt

def loadDataSet(fileName):      #general function to parse tab -delimited floats
    numFeat = len(open(fileName).readline().split('\t')) - 1 #get number of fields
    dataMat = []; labelMat = []
    fr = open(fileName)
    for line in fr.readlines():
        lineArr =[]
        curLine = line.strip().split('\t')
        for i in range(numFeat):
            lineArr.append(float(curLine[i]))
        dataMat.append(lineArr)
        labelMat.append(float(curLine[-1]))
    return dataMat,labelMat

def lwlr(testPoint,xArr,yArr,k=1.0):
    xMat = mat(xArr); yMat = mat(yArr).T
    m = shape(xMat)[0]
    weights = mat(eye((m)))
    for j in range(m):                      #next 2 lines create weights matrix
        diffMat = testPoint - xMat[j,:]     #
        weights[j,j] = exp(diffMat*diffMat.T/(-2.0*k**2))
    xTx = xMat.T * (weights * xMat)
    if linalg.det(xTx) == 0.0:
        print("This matrix is singular, cannot do inverse")
        return
    ws = xTx.I * (xMat.T * (weights * yMat))
    return testPoint * ws
```

```
def lwlrTest(testArr,xArr,yArr,k=1.0):  #loops over all the data points and applies
lwlr to each one
    m = shape(testArr)[0]
    yHat = zeros(m)
    for i in range(m):
        yHat[i] = lwlr(testArr[i],xArr,yArr,k)
    return yHat

def lwlrTestPlot(xArr,yArr,k=1.0):  #same thing as lwlrTest except it sorts X first
    yHat = zeros(shape(yArr))       #easier for plotting
    xCopy = mat(xArr)
    xCopy.sort(0)
    for i in range(shape(xArr)[0]):
        yHat[i] = lwlr(xCopy[i],xArr,yArr,k)
    return yHat,xCopy

def rssError(yArr,yHatArr): #yArr and yHatArr both need to be arrays
    return ((yArr-yHatArr)**2).sum()

def draw(xSort, yHat, xMat, yMat):
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.plot(xSort[:,1],yHat)
    ax.scatter(xMat[:,1].flatten().A[0], yMat.T.flatten().A[0], s=2, c='red')
    plt.show()

abX,abY = loadDataSet('/Users/xinyuanhe/Desktop/working/2021美赛/软件/Python机器学
习/kNN/machinelearninginaction/Ch08/ex0.txt')
xMat = mat(abX); yMat = mat(abY)
yHat, xCopy = lwlrTestPlot(abX,abY,0.003)
draw(xCopy, yHat, xMat, yMat)
```

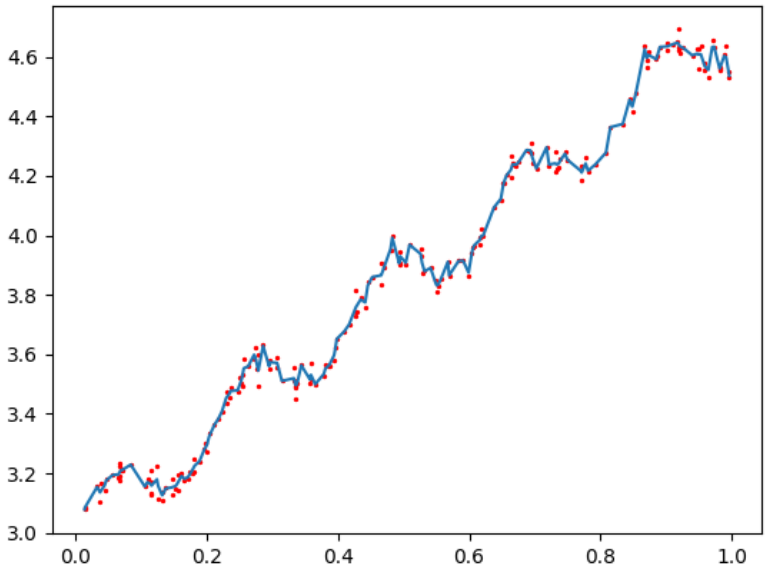
$k = 0.1$（加权效果差）



$k = 0.02$



$k = 0.003$（过拟合）



5. 参考资料

1. 《机器学习实战》P70-P75 机器学习实战.pdf
2. 局部加权线性回归
3. 机器学习-局部加权线性回归