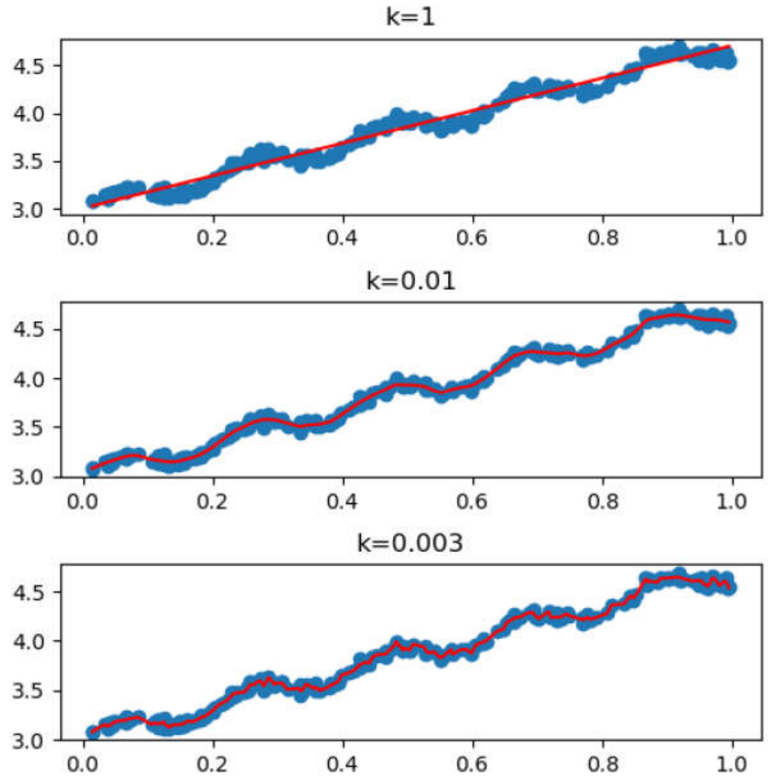
Python编程与人工智能实践



算法篇：局部加权线性回归

Local weighted Linear Regress

于泓
鲁东大学
信息与电气工程学院

2021.9.24

局部加权线性回归

在普通的线性回归中，所有参与训练的样本点，权重都是相同的，
在局部加权线性回归的算法中，在**测试过程中**，
根据测试数据的不同，每个样本点都被赋予**不同的权重**，
然后再计算**w**
即：对每个不同的测试样本计算不同的w

函数为：

$$\hat{y}^{(i)} = f(\mathbf{x}^{(i)}; \mathbf{w}^{(i)}, \mathbf{X}_{train}, \mathbf{y}_{train})$$

损失函数

线性回归

$$\mathrm{L}_{\mathrm{MSE}} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

局部加权线性回归 （上标表示测试数据的序号，下标表示训练数据的序号）

$$\mathrm{L}^{(i)}{}_{\mathrm{MSE}} = \sum_{i=1}^{N} c^{(i)} (y_i - \hat{y}_i)^2 = \mathbf{C}^{(i)} (\mathbf{y} - \hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}})$$

随着测试数据的不同 $\mathbf{C}^{(i)}$**也会发生变化** （上标表示测试数据的序号，下标表示训练数据的序号）

$$\mathbf{C}^{(i)} = \begin{bmatrix} c_1^{(i)} & & & \\ & c_2^{(i)} & & \\ & & \cdots & \\ & & & c_N^{(i)} \end{bmatrix}$$

$$c_j^{(i)} = \exp\left(\frac{\left\| \mathbf{x}^{(i)} - \mathbf{x}_j \right\|_2}{-2k^2} \right)$$

与测试样本较**接近**的
训练样本会被赋予较大的
权重来参与**w的计算**

k越大参与计算的训练样本越多
K越小参与计算的训练样本越少

$$\mathrm{L}^{(i)}{}_{\mathrm{MSE}} = \sum_{i=1}^{N} c^{(i)} (y_i - \hat{y}_i)^2 = \mathbf{C}^{(i)} (\mathbf{y} - \hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}})$$

$$= \mathbf{C}^{(i)} \mathbf{y}^{\mathrm{T}} \mathbf{y} - \mathbf{C}^{(i)} \mathbf{y} \mathbf{X} \mathbf{w}^{(i)} - \mathbf{C}^{(i)} \mathbf{w}^{(i)\mathrm{T}} \mathbf{X}^{\mathrm{T}} \mathbf{y} + \mathbf{C}^{(i)} \mathbf{w}^{(i)\mathrm{T}} \mathbf{X}^{\mathrm{T}} \mathbf{X} \mathbf{w}$$

$$\frac{\partial \mathrm{L}^{(i)}{}_{\mathrm{MSE}}}{\partial \mathbf{w}^{(i)}} = 2\mathbf{X}^T \mathbf{C}^{(i)} \mathbf{X} \mathbf{w} - 2\mathbf{X}^T \mathbf{C}^{(i)} \mathbf{y} = 0$$

$$\mathbf{w}^{(i)} = \left(\mathbf{X}^T \mathbf{C}^{(i)} \mathbf{X} \right)^{-1} \mathbf{X}^T \mathbf{C}^{(i)} \mathbf{y}$$

代码实现：

```python
def local_weight_LR(test_point,train_X,train_Y,k=1.0):
    xMat = mat(train_X)
    yMat = mat(train_Y)
    N,D = np.shape(xMat)
    # 构建weights 矩阵
    diff_mat = np.tile(test_point,[N,1])-train_X
    weights = np.exp(np.sum(diff_mat**2,axis=1)/(-2*k**2))
    weights= mat(np.diag(weights))
    xTx = xMat.T*(weights*xMat)
    if np.linalg.det(xTx) == 0.0:
        print("数据错误， 无法求逆矩")
        return
    ws = xTx.I * xMat.T*weights*yMat
    return test_point*ws
```

$$c_j^{(i)} = \exp\left(\frac{\left\|\mathbf{x}^{(i)} - \mathbf{x}_j\right\|_2}{-2k^2}\right)$$

$$\mathbf{w}^{(i)} = \left(\mathbf{X}^T\mathbf{C}^{(i)}\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{C}^{(i)}\mathbf{y}$$

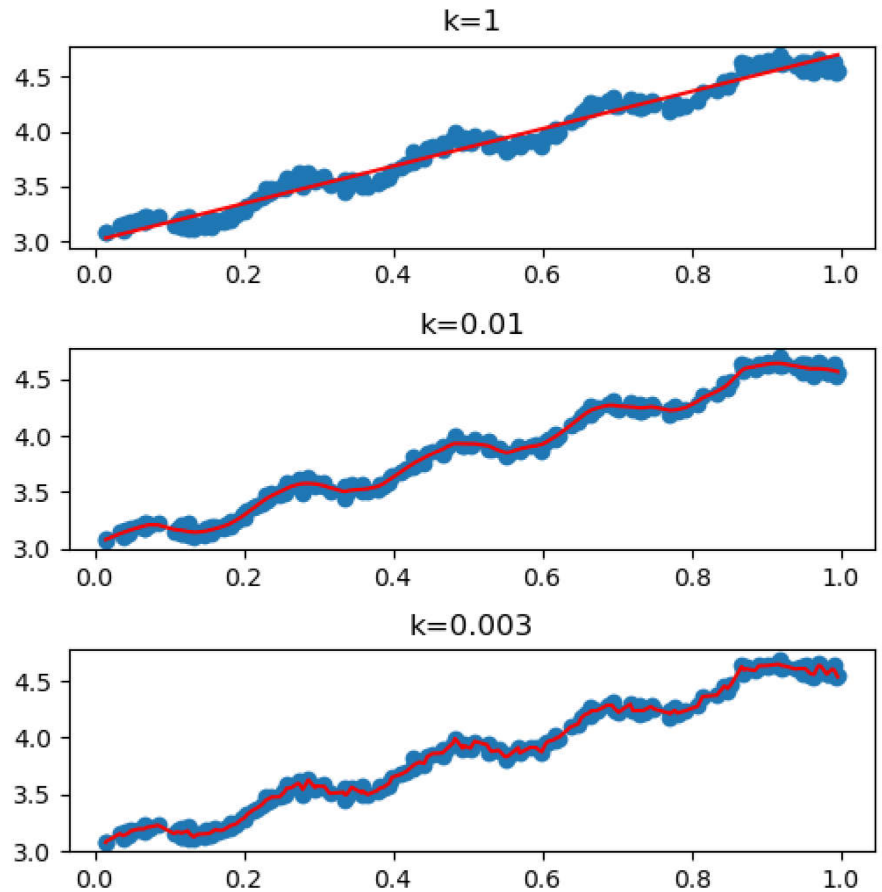
测试

```python
def test_local_weight_LR(test_points,train_X,train_Y,k=1.0):
    N,D = test_points.shape
    Y_hat = np.zeros((N,1))
    for i in range(N):
        Y_hat[i]=local_weight_LR(test_points[i],train_X,train_Y,k=k)
    return Y_hat
```

测试:

```python
if __name__ == "__main__":
    X,Y = load_DataSet('ex0.txt')
    N,D= X.shape

    Y_hat_1 = test_local_weight_LR(X,X,Y,k=1.0)
    Y_hat_2 = test_local_weight_LR(X,X,Y,k=0.01)
    Y_hat_3 = test_local_weight_LR(X,X,Y,k=0.003)
```




```python
# 绘图
index=np.argsort(X[:,1])
X_copy= X[index,:]

fig = plt.figure() # 创建绘图对象
fig.subplots_adjust(hspace=0.5)
# 子图1
ax1 = fig.add_subplot(3,1,1)
ax1.scatter(X[:,1],Y)
Y_hat = Y_hat_1[index]
ax1.plot(X_copy[:,1],Y_hat,color=(1,0,0))
ax1.set_title("k=0.1")

# 子图2
ax2 = fig.add_subplot(3,1,2)
ax2.scatter(X[:,1],Y)
Y_hat = Y_hat_2[index]
ax2.plot(X_copy[:,1],Y_hat,color=(1,0,0))
ax2.set_title("k=0.01")

# 子图 3
ax3 = fig.add_subplot(3,1,3)
ax3.scatter(X[:,1],Y)
Y_hat = Y_hat_3[index]
ax3.plot(X_copy[:,1],Y_hat,color=(1,0,0))
ax3.set_title("k=0.003")

plt.show()
```