AI学习笔记--skLearn--Linear Regression

- 概述

线性回归是利用数理统计中回归分析，来确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法，运用十分广泛。其表达形式为y = w'x+e，e为误差服从均值为0的正态分布。[1]

回归分析中，只包括一个自变量和一个因变量，且二者的关系可用一条直线近似表示，这种回归分析称为一元线性回归分析。如果回归分析中包括两个或两个以上的自变量，且因变量和自变量之间是线性关系，则称为多元线性回归分析。

- 算法解析
- 最小二乘法

一般来说，线性回归都可以通过最小二乘法求出其方程，可以计算出对于y=bx+a的直线。

虽然不同的统计软件可能会用不同的格式给出回归的结果，但是它们的基本内容是一致的。以STATA的输出为例来说明如何理解回归分析的结果。在这个例子中，测试读者的性别（gender），年龄（age），知识程度（know）与文档的次序（noofdoc）对他们所觉得的文档质量(relevance)的影响。

- 输出

这个输出包括以下几部分。左上角给出方差分析表，右上角是模型拟合综合参数。下方的表给出了具体变量的回归系数。方差分析表对大部分的行为研究者来讲不是很重要，不做讨论。在拟合综合参数中， R-squared 表示因变量中多大的一部分信息可以被自变量解释。

- 输出回归系数

一般地，要求这个值大于5%。对大部分的行为研究者来讲，最重要的是回归系数。年龄增加1个单位，文档的质量就下降 -.1020986个单位，表明年长的人对文档质量的评价会更低。这个变量相应的t值是 -2.10，绝对值大于2，p值也<0.05，所以是显著的。结论是，年长的人对文档质量的评价会更低，这个影响是显著的。相反，领域知识越丰富的人，对文档的质量评估会更高，但是这个影响不是显著的。这种对回归系数的理解就是使用回归分析进行假设检验的过程。

- 回归方程误差

离差平方和

$y = bx + a + \varepsilon$，$\varepsilon \sim N(0, \sigma^2)$，$\sigma^2 = \frac{SSE}{n-2} = \frac{S_{yy}}{n-2}(1-r^2)$

其中 $S_{yy} = \sum (y - \overline{y})^2$，代表y的平方和；$r^2$ 是相关系数，代表变异被回归直线解释的比例；$S_{yy}(1-r^2)$ 就是不能被回归直线解释的变异，即SSE。

根据回归系数与直线斜率的关系，可以得到等价形式：

$$r = \frac{\sigma_{xy}^2}{\sigma_x \cdot \sigma_y} = \frac{\frac{\sum(x-\overline{x})(y-\overline{y})}{n}}{\sqrt{\frac{\sum(x-\overline{x})^2}{n}} \times \sqrt{\frac{\sum(y-\overline{y})^2}{n}}} = \frac{\sum(x-\overline{x})(y-\overline{y})}{\sqrt{\sum(x-\overline{x})^2} \cdot \sqrt{\sum(y-\overline{y})^2}}$$

其中b为直线斜率，预测值 $\sigma^2 = \frac{(Y_i - y_i)^2}{n-2}$，其中 y_i 是实际测量值，Y_i 是根据直线方程算出来的预测值。

- 不确定度

斜率b

方法1：用σ来取得

$$u(b) = \frac{\sigma}{\sqrt{\sum (x - \overline{x})^2}}$$

方法2：把斜率b带入

$$u(b) = \sqrt{\frac{S_{yy} - bS_{xy}}{(n-2)S_{xx}}} = \sqrt{\frac{\frac{S_{yy}}{S_{xx}} - b^2}{(n-2)}}$$

截距a

$$u(a) = u(b)\sqrt{\frac{\sum x_i^2}{n}}$$

SKLearn 提供了很多关于线性回归的方法。举几个例子说明。

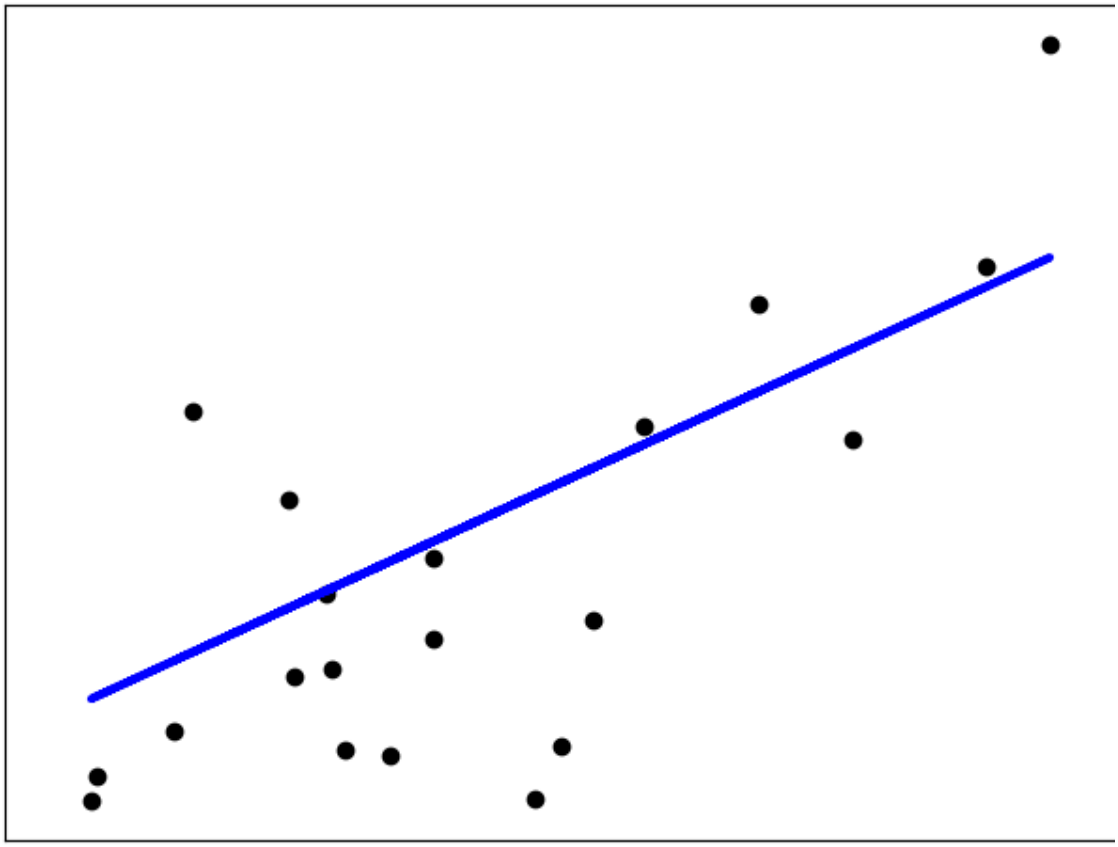
- 算法历程

```
print(__doc__)


# Code source: Jaques Grobler
# License: BSD 3 clause


import matplotlib.pyplot as plt
import numpy as np
```

```
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

# Load the diabetes dataset
diabetes = datasets.load_diabetes()


# Use only one feature
diabetes_X = diabetes.data[:, np.newaxis, 2]

# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

# Split the targets into training/testing sets
diabetes_y_train = diabetes.target[:-20]
diabetes_y_test = diabetes.target[-20:]

# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)

# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)

# The coefficients
print('Coefficients: \n', regr.coef_)
# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(diabetes_y_test, diabetes_y_pred))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(diabetes_y_test, diabetes_y_pred))

# Plot outputs
plt.scatter(diabetes_X_test, diabetes_y_test,  color='black')
plt.plot(diabetes_X_test, diabetes_y_pred, color='blue', linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()
```

运行结果：

```
('Coefficients: \n', array([938.23786125]))
Mean squared error: 2548.07
Variance score: 0. 47
```



再来看一个原始算法的历程：

```
# -*- coding:utf-8 -*-

from numpy import *
import matplotlib.pyplot as plt

def loadDataSet(filename):
    numFeat = len(open(filename).readline().split('\t'))-1#读取第一行并且计算长度
    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 standRegres(xArr,yArr):
    xMat = mat(xArr)
    yMat = mat(yArr).T
    xTx=xMat.T*xMat
    if linalg.det(xTx)==0:#计算行列式
```

```
        print('this matrix is singular,can do inverse')
        return
    ws = xTx.I*(xMat.T*yMat)
    return ws


def show(xArr,yArr,ws):
    xMat = mat(xArr)
    yMat = mat(yArr)
    fig=plt.figure()
    ax = fig.add_subplot(111)
    ax.scatter(xMat[:,1].flatten().A[0],yMat.T[:,0].flatten().A[0])
    xCopy = xMat.copy()
    xCopy.sort(0)
    yHat= xCopy*ws
    ax.plot(xCopy[:,1],yHat,c='red')
    plt.show()

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):
        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):
    m=shape(testArr)[0]
    yHat = zeros(m)
    for i in range(m):
        yHat[i] = lwlr(testArr[i],xArr,yArr,k)
    return yHat

def rssError(yArr,yHatArr):
    return ((yArr-yHatArr)**2).sum()

#岭回归
def ridgeRegres(xMat,yMat,lam=0.2):
    xTx = xMat.T*xMat
    denom = xTx+eye(shape(xMat)[1])*lam
    if linalg.det(denom) == 0.0:
        print('This matrix is singular,cannot do inverse')
        return
    ws = denom.I*(xMat.T*yMat)
    return ws

def ridgeTest(xArr,yArr):
```

```
    xMat = mat(xArr)
    yMat = mat(yArr).T
    yMean = mean(yMat,0)
    yMat = yMat- yMean
    xMeans = mean(xMat,0)
    xVar = var(xMat,0)
    xMat = (xMat - xMeans)/xVar
    numTestPts = 30
    wMat = zeros((numTestPts,shape(xMat)[1]))
    for i in range(numTestPts):
        ws = ridgeRegres(xMat,yMat,exp(i-10))
        wMat[i,:] = ws.T
    return wMat

def regularize(xMat):
    inMat = xMat.copy()
    inMeans = mean(inMat,0)
    inVar = var(inMat,0)
    inMat = (inMat -inMeans)/inVar
    return inMat

#逐步向前线性回归
def stageWise(xArr,yArr,eps=0.01,numIt=100):
    xMat = mat(xArr)
    yMat = mat(yArr).T
    yMean = mean(yMat,0)
    yMat = yMat-yMean
    xMat = regularize(xMat)
    m,n=shape(xMat)
    returnMat = zeros((numIt,n))
    ws = zeros((n,1))
    wsTest = ws.copy()
    wsMax = ws.copy()
    for i in range(numIt):
        print(ws.T)
        lowestError = inf
        for j in range(n):
            for sign in [-1,1]:
                wsTest = ws.copy()
                wsTest[j] += eps*sign
                yTest = xMat*wsTest
                rssE = rssError(yMat.A,yTest.A)
                if rssE<lowestError:
                    lowestError = rssE
                    wsMax = wsTest
        ws = wsMax.copy()
        returnMat[i,:] = ws.T
    return returnMat




if __name__ =='__main__':
    xArr,yArr= loadDataSet('ex0.txt')
```

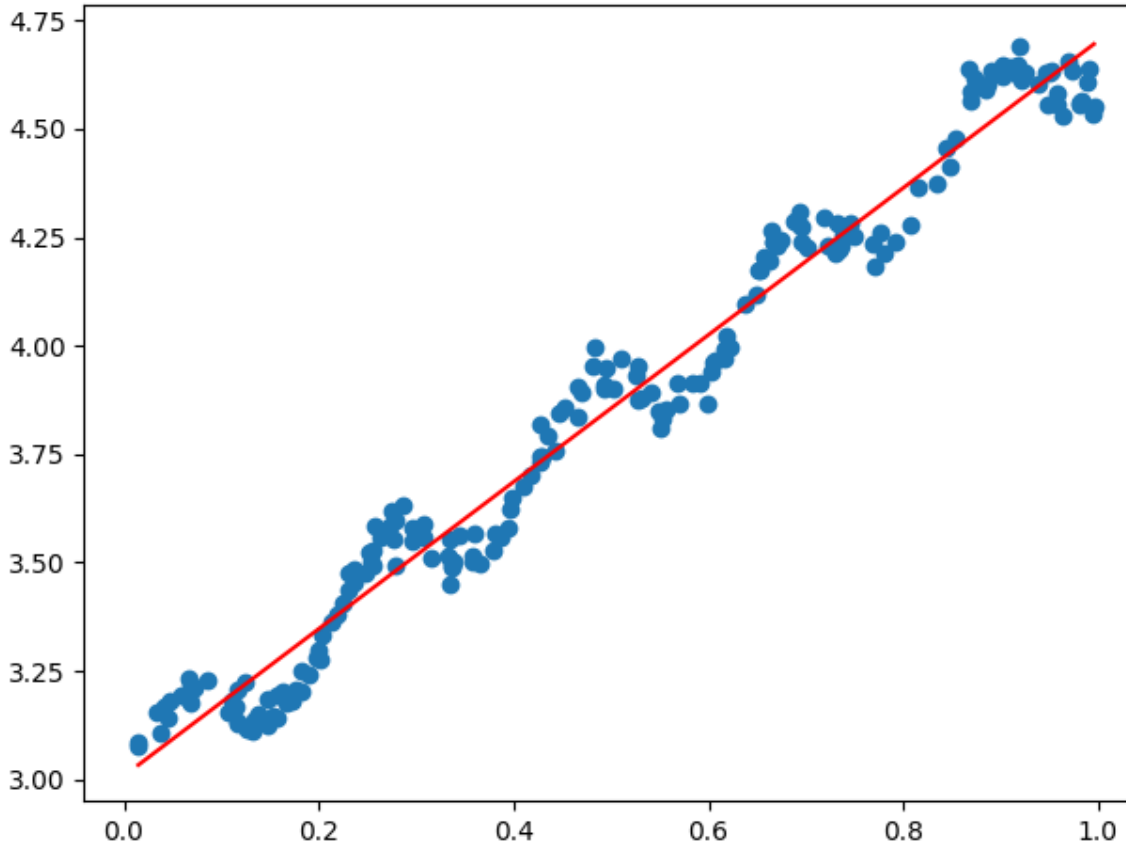
```
    ws=standRegres(xArr,yArr)
    show(xArr,yArr,ws)
    xMat = mat(xArr)
    yHat = lwlrTest(xArr,xArr,yArr,0.003)
    srtInd = xMat[:,1].argsort(0)
    xSort = xMat[srtInd][:,0,:]
    #画出最后的结果
    fig=plt.figure()
    ax = fig.add_subplot(111)
    ax.plot(xSort[:,1],yHat[srtInd])

ax.scatter(xMat[:,1].flatten().A[0],mat(yArr).T.flatten().A[0],s=2,c='red')
    plt.show()

    # #预测鲍鱼年龄
    # abX,abY = loadDataSet('abalone.txt')
    # ridgeWeights = ridgeTest(abX,abY)
    # fig = plt.figure()
    # ax =fig.add_subplot(111)
    # ax.plot(ridgeWeights)
    # plt.show()
    # xArr,yArr = loadDataSet('abalone.txt')
    # stageWise(xArr,yArr,0.001,5000)
    # xMat = mat(xArr)
    # yMat = mat(yArr).T
    # xMat=regularize(xMat)
    # yM = mean(yMat,0)
    # yMat = yMat - yM
    # weights = standRegres(xMat,yMat.T)
    # print('weights:',weights.T)
```

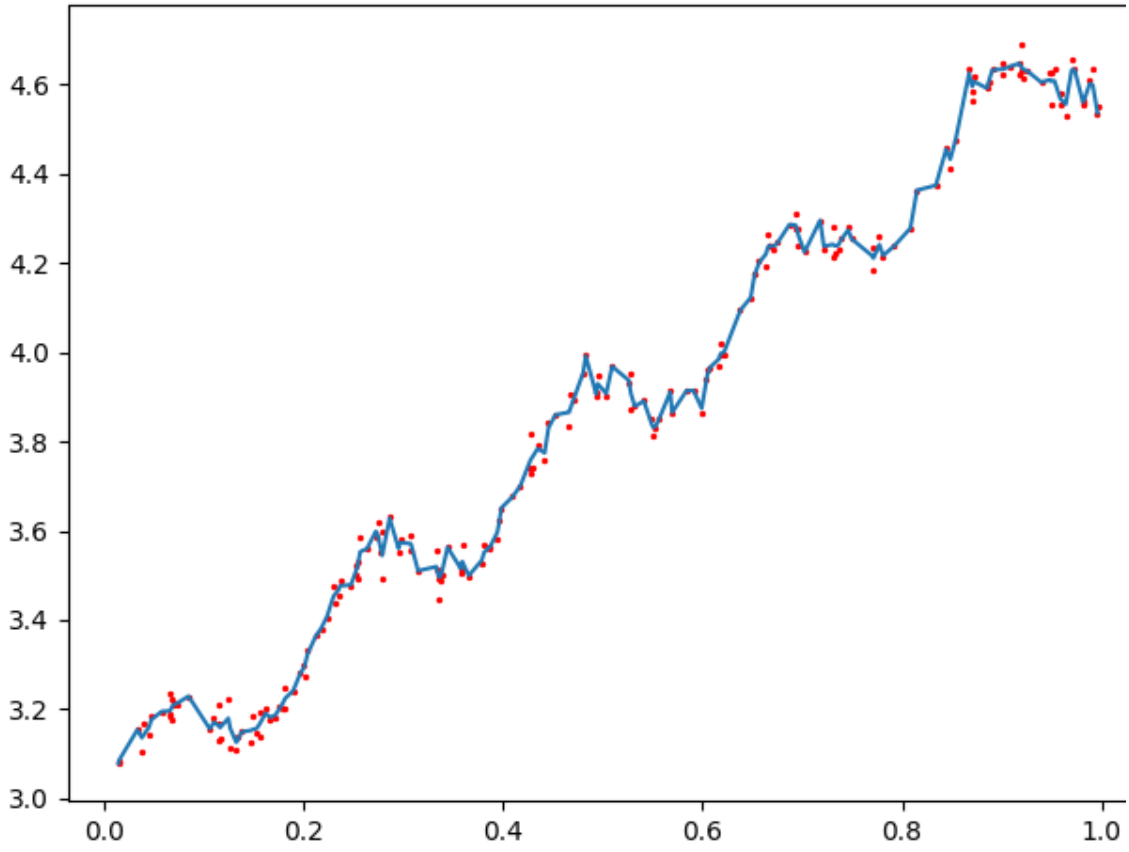
实验数据：

ex0.txt

abalone.txt

ex1.txt


4.75
4.50
4.25
4.00
3.75
3.50
3.25
3.00
0.0
0.2
0.4
0.6
0.8
1.0


4.6
4.4
4.2
4.0
3.8
3.6
3.4
3.2
3.0
0.0
0.2
0.4
0.6
0.8
1.0