机器学习——线性回归进阶版

对于线性回归的基础知识我们已经实现了一遍，接下来就是要对它进行一定的缩减优化，例如岭回归以及逐步线性回归，同时对于sklearn进行一定的理解。

岭回归

如果我们的数据集特征比样本点还多，显然上一节的方法是不可以的，因为输入矩阵X不是满秩矩阵，非满秩矩阵在求逆时时会出现问题，因此我们需要岭回归。岭回归就是我们说到的L2正则线性回归，在一般的线性回归最小化均方误差的基础上我们添加了一个参数w的L2范数惩罚项，从而最小化残差平方和变成如下：

$$min||Xw-y||_2^2+\lambda||w||_2^2$$

因此我们通过最小化这个函数得到了：

$$\hat{w}=(X^TX+\lambda I)^{-1}X^Ty$$

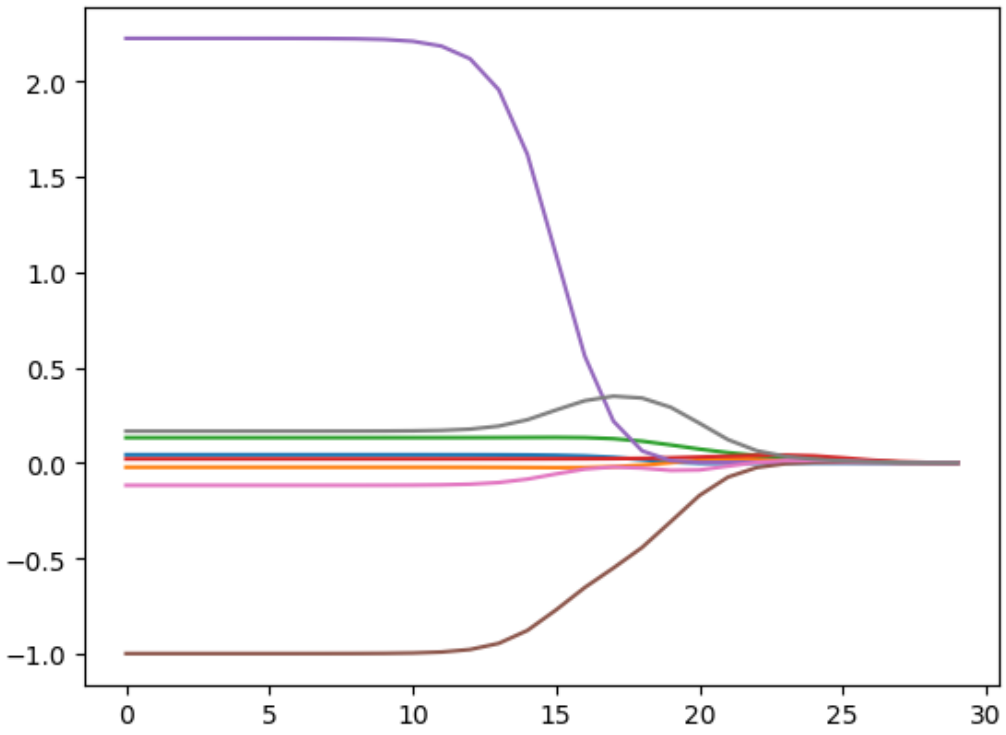
相比较我们之前得到的回归系数的式子主要就是增加了一个防止矩阵奇异的λI的矩阵。岭回归通过λ来限制所有权值的和，通过这样来减少不重要的参数，这就叫缩减。

但是注意一点，就是使用了岭回归以及缩减技术，首先需要对特征进行标准化处理，因为我们需要对每个维度特征处理成具有相同重要性的特征值，一般的处理可以用特征减去各自的均值再除以方差。

我们首先观察岭回归的权值随lamma的变化趋势：

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

def loadDataSet(filename):
    fr=open(filename)
    xArr=[]
    yArr=[]
    numFeat=len(open(filename).readline().split('\t'))-1
    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 rigeRegress(xMat,yMat,lam=0.2):
    xTx=xMat.T*xMat
    denom=xTx+np.eye(np.shape(xMat)[1])*lam
    if np.linalg.det(denom)==0.0:
        print("矩阵为奇异矩阵，不能转置")
        return
    w=denom.I*(xMat.T*yMat)
    return w

def rigeTest(xArr,yArr):
    xMat=np.mat(xArr)
```

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

def plot():
    abX,abY=loadDataSet('abalone.txt')
    w=rigeTest(abX,abY)
    fig=plt.figure()
    ax=fig.add_subplot(111)
    ax.plot(w)
    plt.show()
plot()
```



可以看到在lamma比较小时，所有系数是正常的原始值，随着lamma的增加，系数缩减成0，在中间部分会有比较好的预测结果，要得到这个参数可以用交叉验证获得，我们慢慢来。

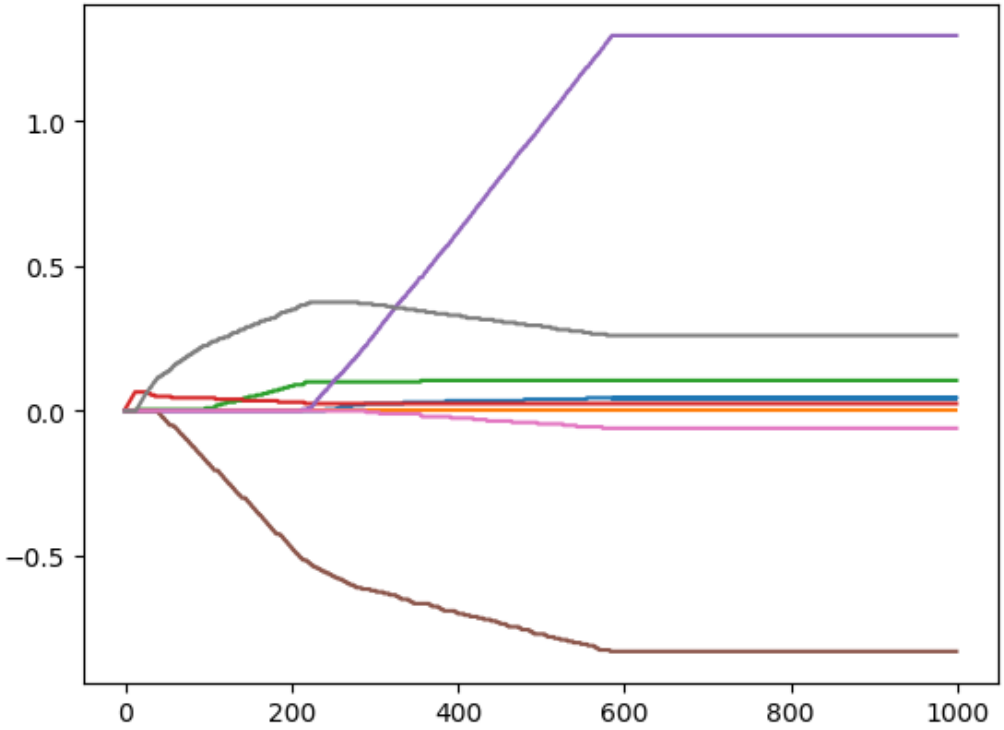
前向逐步线性回归

这是一种贪心算法，每一次都尽可能减少误差，通过每一次微调回归系数，计算预测误差，使得误差最小的一组回归系数就是最佳回归系数。

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

```
def loadDataSet(filename):
    fr=open(filename)
    xArr=[]
    yArr=[]
    numFeat=len(open(filename).readline().split('\t'))-1
    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 regularize(xMat,yMat):
    inxMat=xMat.copy()
    inyMat=yMat.copy()
    yMean=np.mean(yMat,0)
    inyMat=inyMat-yMean
    xMean=np.mean(xMat,0)
    xVar=np.var(xMat,0)
    inxMat=(inxMat-xMean)/xVar
    return inxMat,inyMat

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

def stageWise(xArr,yArr,eps=0.01,numIt=100):
    xMat=np.mat(xArr)
    yMat=np.mat(yArr).T
    xMat,yMat=regularize(xMat,yMat)
    m,n=np.shape(xMat)
    returnMat =np.zeros((numIt,n))
    ws=np.zeros((n,1))
    wsTest=ws.copy()
    wsMax=ws.copy()
    for i in range(numIt):
        lowestError=float('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
def plot():
    abX,abY=loadDataSet('abalone.txt')
    w=stageWise(abX,abY,0.005,1000)
    fig=plt.figure()
    ax=fig.add_subplot(111)
    ax.plot(w)
    plt.show()
```

```
plot()
```



综上，我们可以看到有些系数始终是0，说明他们对于目标并没有影响，这也体现出逐步线性回归可以帮助我们理解模型并改进，可以停止对一些不重要特征的收集。

接下来我们就使用实战演练来试一下：通过对网页的解析进行回归：

```
import numpy as np
from bs4 import BeautifulSoup
import random

def scrapePage(retX,retY,inFile,yr,numPce,origPrc):
    with open(inFile,encoding='utf-8') as f:
        html=f.read()
    soup =BeautifulSoup(html)
    i=1
    currentRow=soup.find_all('table',r="%d" %i)
    while(len(currentRow)!=0):
        currentRow=soup.find_all('table',r="%d" %i)
        title =currentRow[0].find_all('a')[1].text
        lwrTitle=title.lower()
        if(lwrTitle.find('new')>-1) or (lwrTitle.find('nisb')>-1):
            newFlag=1.0
        else:
            newFlag=0.0
        soldUnicde=currentRow[0].find_all('td')[3].find_all('span')
        if len(soldUnicde)==0:
            print("商品 #%d 没有出售" %i)
        else:
            soldPrice=currentRow[0].find_all('td')[4]
            priceStr=soldPrice.text
            priceStr=priceStr.replace('$','')
```

```
            priceStr=priceStr.replace(',','')
            if len(soldPrice)>1:
                priceStr=priceStr.replace('Free shipping','')
            sellingPrice=float(priceStr)
            if sellingPrice>origPrc*0.5:
                print("%d\t%d\t%d\t%f\t%f" %
(yr,numPce,newFlag,origPrc,sellingPrice))
                retX.append([yr,numPce,newFlag,origPrc])
                retY.append(sellingPrice)
        i+=1
        currentRow=soup.find_all('table',r="%d" %i)


def setDataCollect(retX,retY):
    scrapePage(retX,retY,'./lego/lego8288.html',2006,800,49.99)
    scrapePage(retX,retY,'./lego/lego10030.html',2002,3096,269.99)
    scrapePage(retX,retY,'./lego/lego10179.html',2007,5195,499.99)
    scrapePage(retX,retY,'./lego/lego10181.html',2007,3428,199.99)
    scrapePage(retX,retY,'./lego/lego10189.html',2008,5922,299.99)
    scrapePage(retX,retY,'./lego/lego10196.html',2009,3263,249.99)

def regularize(xMat,yMat):
    inxMat=xMat.copy()
    inyMat=yMat.copy()
    yMean=np.mean(yMat,0)
    inyMat=inyMat-yMean
    xMean=np.mean(xMat,0)
    xVar=np.var(xMat,0)
    inxMat=(inxMat-xMean)/xVar
    return inxMat,inyMat

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

def rigeRegress(xMat,yMat,lam=0.2):
    xTx=xMat.T*xMat
    denom=xTx+np.eye(np.shape(xMat)[1])*lam
    if np.linalg.det(denom)==0.0:
        print("矩阵为奇异矩阵，不能转置")
        return
    w=denom.I*(xMat.T*yMat)
    return w

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

```
def crossValidation(xArr,yArr,numVal=10):
    m=len(yArr)
    indexList=list(range(m))
    errorMat=np.zeros((numVal,30))
    for i in range(numVal):
        trainX=[]
        trainY=[]
        testX=[]
        testY=[]
        random.shuffle(indexList)
        for j in range(m):
            if j<m*0.9:
                trainX.append(xArr[indexList[j]])
                trainY.append(yArr[indexList[j]])
            else:
                testX.append(xArr[indexList[j]])
                testY.append(yArr[indexList[j]])
        wMat=rigeTest(trainX,trainY)
        for k in range(30):
            matTestX=np.mat(testX)
            matTrainX=np.mat(trainX)
            meanTrain=np.mean(matTrainX,0)
            varTrain=np.var(matTrainX,0)
            matTestX=(matTestX-meanTrain)/varTrain
            yEst=matTestX*np.mat(wMat[k,:]).T+np.mean(trainY)
            errorMat[i,k]=rssError(yEst.T.A,np.array(testY))
    meanErrors=np.mean(errorMat,0)
    minMean=float(min(meanErrors))
    bestWeights=wMat[np.nonzero(meanErrors==minMean)]
    xMat=np.mat(xArr)
    yMat=np.mat(yArr).T
    meanX=np.mean(xMat,0)
    varX=np.var(xMat,0)
    unReg=bestWeights/varX
    print('%f%+f*年份%+f*部件数量%+f*是否为全新%+f*原价' % ((-1 *
np.sum(np.multiply(meanX,unReg)) + np.mean(yMat)), unReg[0,0], unReg[0,1],
unReg[0,2], unReg[0,3]))

lgX=[]
lgY=[]
setDataCollect(lgX,lgY)
crossValidation(lgX,lgY)
print(rigeTest(lgX,lgY))
```

```
商品 #16 没有出售
2009   3263   1 249.990000   380.000000
2009   3263   1 249.990000   399.000000
2009   3263   1 249.990000   427.990000
2009   3263   0 249.990000   360.000000
商品 #5 没有出售
商品 #6 没有出售
2009   3263   1 249.990000   399.000000
2009   3263   1 249.990000   399.950000
2009   3263   1 249.990000   499.990000
商品 #10 没有出售
2009   3263   0 249.990000   399.950000
```

```
商品 #12 没有出售
2009   3263   1  249.990000   331.510000
67121.846064-33.463940年份-0.002901部件数量+27.205717是否为全新+2.149328原价
[[-1.45288906e+02 -8.39360442e+03 -3.28682450e+00  4.42362406e+04]
 [-1.46649725e+02 -1.89952152e+03 -2.80638599e+00  4.27891633e+04]
 [-1.44450432e+02  8.55488076e+02 -1.35089285e+00  4.00885735e+04]
 [-1.37402474e+02  1.64217093e+03  1.95840783e+00  3.44932120e+04]
 [-1.24750588e+02  1.44326171e+03  7.62540167e+00  2.50647592e+04]
 [-1.10234679e+02  8.81842164e+02  1.40617304e+01  1.43874420e+04]
 [-9.96484167e+01  4.17805568e+02  1.87140361e+01  6.66770425e+03]
 [-9.40345090e+01  1.71289137e+02  2.10844952e+01  2.71206176e+03]
 [-9.11400659e+01  6.57287394e+01  2.20487105e+01  1.03800465e+03]
 [-8.86246985e+01  2.45452725e+01  2.23181664e+01  3.87564774e+02]
 [-8.41447674e+01  9.05861459e+00  2.21495534e+01  1.43313895e+02]
 [-7.44804291e+01  3.31863501e+00  2.14607512e+01  5.27810178e+01]
 [-5.68008473e+01  1.20770663e+00  2.00168153e+01  1.93999701e+01]
 [-3.43546503e+01  4.38238026e-01  1.77836684e+01  7.12719906e+00]
 [-1.62951276e+01  1.59882766e-01  1.48514658e+01  2.62234165e+00]
 [-6.48291858e+00  5.89025383e-02  1.09847950e+01  9.67404902e-01]
 [-2.35268585e+00  2.18391027e-02  6.61152257e+00  3.57478187e-01]
 [-8.35001919e-01  8.09519290e-03  3.19552087e+00  1.32007993e-01]
 [-2.99711902e-01  2.99108326e-03  1.33043325e+00  4.86659524e-02]
 [-1.08982743e-01  1.10249682e-03  5.14449554e-01  1.79199150e-02]
 [-3.99038878e-02  4.05898142e-04  1.92885670e-01  6.59479857e-03]
 [-1.46534283e-02  1.49365062e-04  7.14631760e-02  2.42642878e-03]
 [-5.38707897e-03  5.49542829e-05  2.63587872e-02  8.92679468e-04]
 [-1.98130399e-03  2.02173589e-05  9.70622185e-03  3.28404700e-04]
 [-7.28814363e-04  7.43766021e-06  3.57198872e-03  1.20814188e-04]
 [-2.68106796e-04  2.73617711e-06  1.31423307e-03  4.44451712e-05]
 [-9.86297565e-05  1.00658531e-06  4.83502591e-04  1.63504803e-05]
 [-3.62836944e-05  3.70302314e-07  1.77873811e-04  6.01500768e-06]
 [-1.33480028e-05  1.36226645e-07  6.54365445e-05  2.21279795e-06]
 [-4.91045279e-06  5.01149871e-08  2.40728171e-05  8.14042912e-07]]
```

上面的过程运用了交叉熵的方法进行验证。

好了差不多了，我们就演示一下怎么使用sklearn进行线性回归的使用，同样对上面的数据集进行处理：

```
import numpy as np
from bs4 import BeautifulSoup
import random
from sklearn import linear_model

def scrapePage(retX,retY,inFile,yr,numPce,origPrc):
    with open(inFile,encoding='utf-8') as f:
        html=f.read()
    soup =BeautifulSoup(html)
    i=1
    currentRow=soup.find_all('table',r="%d" %i)
    while(len(currentRow)!=0):
        currentRow=soup.find_all('table',r="%d" %i)
        title =currentRow[0].find_all('a')[1].text
        lwrTitle=title.lower()
```

```
        if(lwrTitle.find('new')>-1) or (lwrTitle.find('nisb')>-1):
            newFlag=1.0
        else:
            newFlag=0.0
        soldUnicde=currentRow[0].find_all('td')[3].find_all('span')
        if len(soldUnicde)==0:
            print("商品 #%d 没有出售" %i)
        else:
            soldPrice=currentRow[0].find_all('td')[4]
            priceStr=soldPrice.text
            priceStr=priceStr.replace('$','')
            priceStr=priceStr.replace(',','')
            if len(soldPrice)>1:
                priceStr=priceStr.replace('Free shipping','')
            sellingPrice=float(priceStr)
            if sellingPrice>origPrc*0.5:
                print("%d\t%d\t%d\t%f\t%f" %
(yr,numPce,newFlag,origPrc,sellingPrice))
                retX.append([yr,numPce,newFlag,origPrc])
                retY.append(sellingPrice)
        i+=1
        currentRow=soup.find_all('table',r="%d" %i)


def setDataCollect(retX,retY):
    scrapePage(retX,retY,'./lego/lego8288.html',2006,800,49.99)
    scrapePage(retX,retY,'./lego/lego10030.html',2002,3096,269.99)
    scrapePage(retX,retY,'./lego/lego10179.html',2007,5195,499.99)
    scrapePage(retX,retY,'./lego/lego10181.html',2007,3428,199.99)
    scrapePage(retX,retY,'./lego/lego10189.html',2008,5922,299.99)
    scrapePage(retX,retY,'./lego/lego10196.html',2009,3263,249.99)

def usesklearn():
    reg=linear_model.Ridge(alpha=.5)
    lgX=[]
    lgY=[]
    setDataCollect(lgX,lgY)
    reg.fit(lgX,lgY)
    print('%f%+f*年份%+f*部件数量%+f*是否为全新%+f*原价' %
(reg.intercept_,reg.coef_[0],reg.coef_[1],reg.coef_[2],reg.coef_[3]))

usesklearn()
```

> 55235.771966-27.550831*年份*-0.026911部件数量-10.883816*是否为全新*+2.576278原价

可以看到使用sklearn的好处就是不用自己写很多相关的实现函数，得到的结果也能和我们自己写的差不多。