《机器学习实战》笔记之八——预测数值型数据：回归

第八章 预测数值型数据：回归

8.1 用 回归找到最佳拟合曲线

假设输入数据存放在矩阵**X**中，而回归系数存放在向量**w**中，那么对于给定的数据**X1**，预测结果将会通过**Y1**=**X1**.T×**w**给出。如何找出误差最小的W，一般采用平方误差最小，即最小二乘法。平方误差可以写做：

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

用矩阵表示还可以写做（**y**-**x*****w**）.T*(**y**-**x*****w**)。如果对w求导，得到**x**.T*(**y**-**xw**)，令其等于零，得到：

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

w帽即为当前估计出的w的最优值，表示最佳估计。一般数据矩阵**x**并非有逆，而公式中需要对**x**.T*x求逆，一般指伪逆，python的numpy包中，linalg函数有函数pinv求伪逆，也即**x**.T*x的逆。

拟合数据：

[python]

```python
#!/usr/bin/env python
# coding=utf-8

from numpy import *
from sklearn import linear_model

def loadDataSet(fileName):
    numFeat  = len(open(fileName).readline().split("\t"))-1#-1是因为最后一列为目标值，前面为特征值
    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)                 #准备好数据矩阵
        labelMat.append(float(curLine[-1]))     #准备好目标向量
    return dataMat, labelMat

def standRegres(xArr, yArr):
    xMat = mat(xArr)
    yMat = mat(yArr).T#原始文件为m*n的矩阵，那么特征矩阵dataMat为
```

```
    if linalg.det(xTx) == 0.0:                          #计算行列式是否为0
        print "this matrix is singular, cannot do inverse"
        ws = linalg.pinv(xTx)*(xMat.T*yMat)        #用伪逆来代替逆
        return ws
    ws = xTx.I*(xMat.T*yMat)
    return ws

xArr, yArr = loadDataSet("ex0.txt")
ws   = standRegres(xArr, yArr)#回归系数
print "最小二乘法得到的回归系数：\n",ws

xMat = mat(xArr)
yMat = mat(yArr)
yHat = xMat*ws           #预测值

#===============绘制数据集散点图和最佳拟合直线图=======
import matplotlib.pyplot as plt
fig = plt.figure()
ax  = fig.add_subplot(111)
ax.scatter(xMat[:,1].flatten().A[0], yMat.T[:,0].flatten().A[0])
xCopy = xMat.copy()
yHat  = xCopy*ws
ax.plot(xCopy[:,1], yHat)
plt.show()

#====================皮尔逊相关系数计算两个向量的相关性===
yHat = xMat*ws
print "皮尔逊相关系数计算预测值和真实值之间的相关性：\n",corrcoef(yHat.T, yMat)
#================sklearn里面的线性回归================

clf = linear_model.LinearRegression(fit_intercept=False)
clf.fit(xArr,yArr)
print "sklearn 里面线性回归训练得到的回归系数：\n",clf.coef_
```

得到的系数即预测值与真实值之间的相关性：

Figure 8-1： 预测得到的回归系数

可以看出sklearn里面预测的和计算的回归系数是一样的。

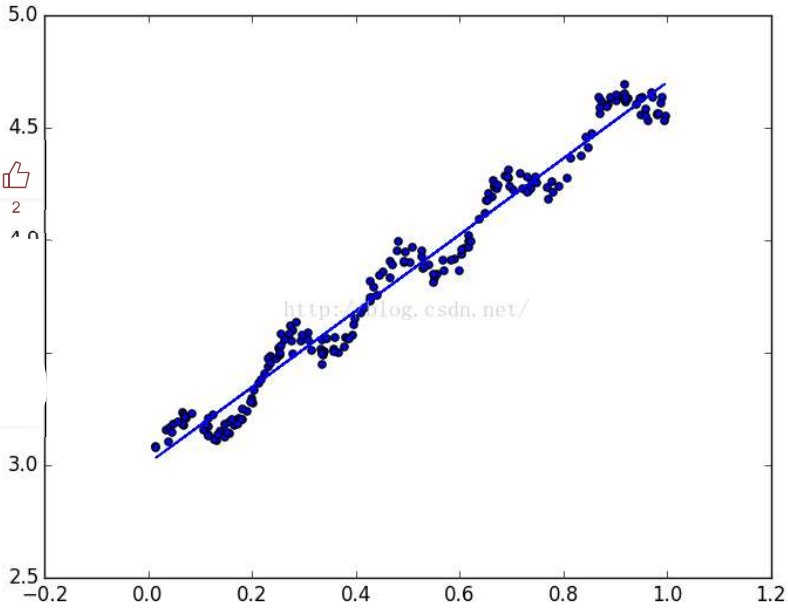


Figure 8-2: 拟合曲线及原始数据的散点图

python sklearn包中有计算线性回归的机器学习算法：

```
In [1]: from sklearn import linear_model

In [2]: import numpy as np

In [3]: x = np.array([[1,6,9],[2,7,7],[3,4,5]])

In [4]: y = np.array([96,90,64])

In [5]: clf = linear_model.LinearRegression(fit_intercept=False)

In [6]: clf.fit(x,y)
Out[6]: LinearRegression(copy_X=True, fit_intercept=False, n_jobs=1, normalize=F
alse)

In [7]: clf.coef_
Out[7]: array([ 3.,  5.,  7.])

In [8]: np.dot(x, clf.coef_)
Out[8]: array([ 96.,  90.,  64.])

In [9]: 1*3+6*5+9*7
Out[9]: 96

In [10]: clf.predict(array([[1,1,1]]))
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-10-3c9a61c0ab0d> in <module>()
----> 1 clf.predict(array([[1,1,1]]))

NameError: name 'array' is not defined

In [11]: clf.predict(np.array([[1,1,1]]))
Out[11]: array([ 15.])
```

http://blog.csdn.net/

Figure 8-3: sklearn 计算线性回归方程

8.2 局部加权线性回归

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

给待预测点附近的每个点赋予一定的权重，然后在这个子集上基于最小均方差来进行普通的回归。算法预测每次均需要事先取出对应的数据子集，解出的回归系数w的形式如下：

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

其中**W**是一个矩阵，用来给每个数据点赋予权重。

LWLR使用“核”来对附近的点赋予更高的权重，该加权模型认为样本点距离越近，越可能符合同一个线性模型。核的类型通常用高斯核。权重计算如下：

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

权重矩阵w为对角阵，点x与x(i)越近，w(i,i)将会越大。

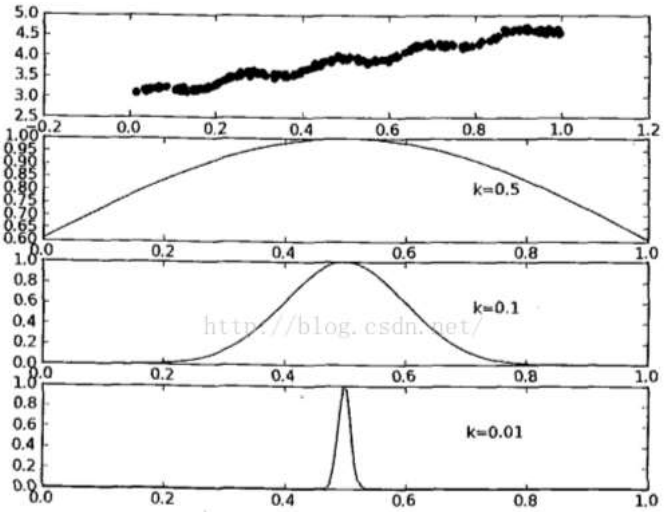


图8-4 每个点的权重图（假定我们正预测的点是x = 0.5），最上面的图是原始数据集，第二个图显示了当k = 0.5时，大部分的数据都用于训练回归模型；而最下面的图显示当k = 0.01时，仅有很少的局部点被用于训练回归模型

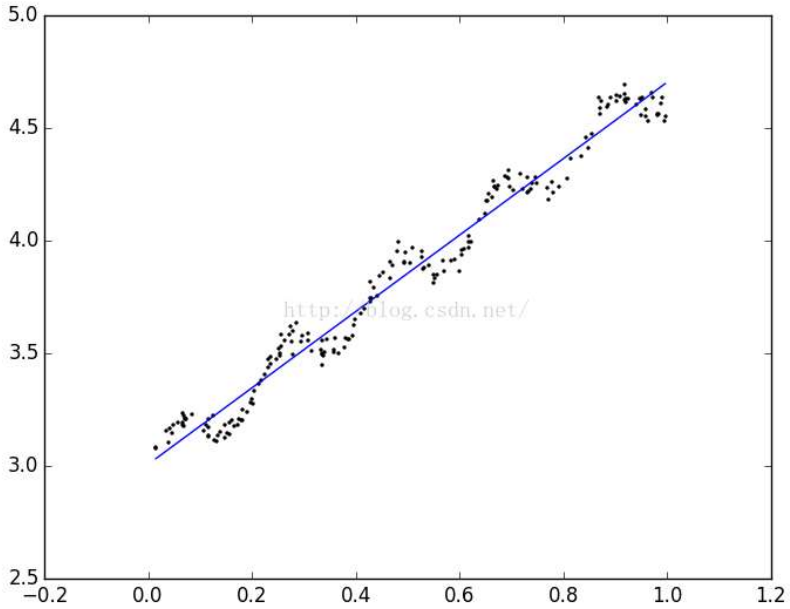
coding:

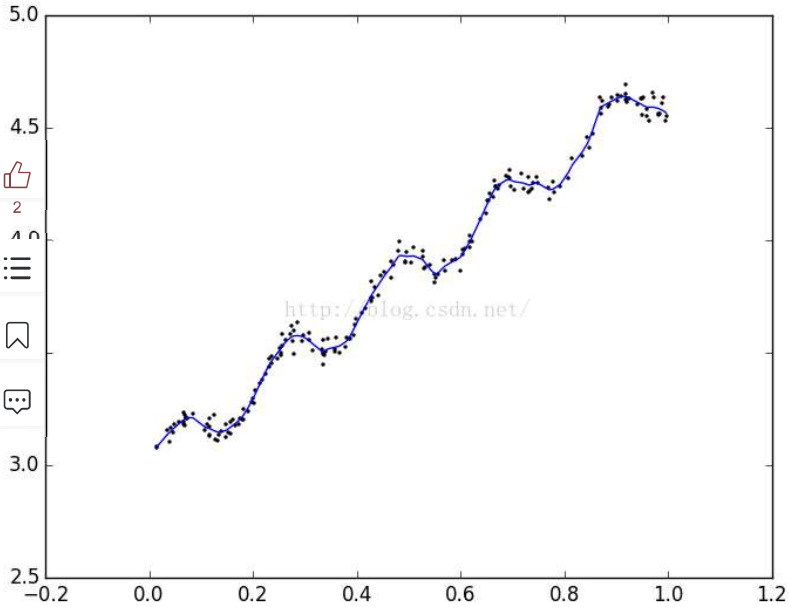
```python
#===============局部加权线性回归函数==================
def lwlr(testPoint, xArr, yArr, k=1.0):
    xMat    = mat(xArr)
    yMat    = mat(yArr).T
    m       = shape(xMat)[0]    #xMat的行数
    weights = mat(eye((m)))     #创建对角阵
    for j in range(m):          #对每个点的计算都用上了整个数据集

        diffMat = testPoint - xMat[j,:]
        weights[j, j] = exp(diffMat*diffMat.T/(-2.0*k**2))#高斯核，对角阵，随样本点与待预测点距离的递增，权重值大小以指数级衰减
    xTx = xMat.T *(weights*xMat)
    if linalg.det(xTx)==0.0:
        print "this matrix is singular, cannot do inverse"
        ws = linalg.pinv(xTx)*(xMat.T*(weights*yMat))
    ws =xTx.I*(xMat.T*(weights*yMat))                    #回归参数
    return testPoint*ws

def lwlrTest(testArr, xArr, yArr, k=1.0): #当对testArr中所有点的估计，testArr=xArr时，即对所以点的全部估计
    m = shape(testArr)[0]
    yHat = zeros(m)
    for i in range(m):
        yHat[i] = lwlr(testArr[i], xArr, yArr, k)
    return yHat

xArr, yArr = loadDataSet("ex0.txt")
```

```
print "估计值：k为0.001",lwlr(xArr[0], xArr, yArr,0.001)
#=========对所有点进行估计,绘图========
yHat = lwlrTest(xArr, xArr, yArr, 0.003)
xMat = mat(xArr)
srtInd = xMat[:,1].argsort(0)
xSort  = xMat[srtInd][:,0,:]

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()
```

```
In [6]: run regression.py
原始值： 3.176513
估计值,k为1.0： [[ 3.12204471]]
估计值：k为0.001 [[ 3.20175729]]
```

Figure 8-5: LWLR对第一个点的估计





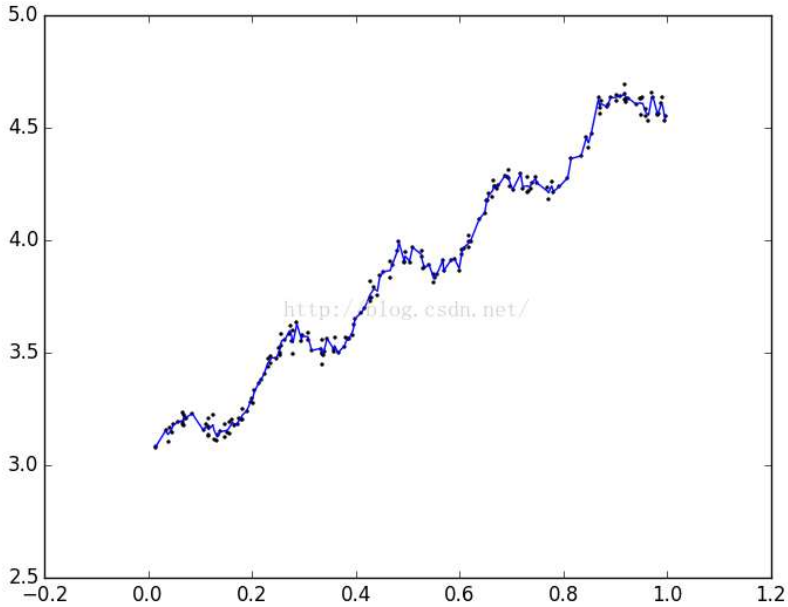


Figure 8-6: k分别为1.0, 0.01, 0.003时的模拟效果，上图欠拟合，下图则过拟合

缺点：LWLR增加了计算量，对每个点做预测时都必需使用整个数据集。在Figure 8-4图中，k=0.01时，对预测点x=0.5估计的时候，大多数据点的权重为0, 就不必用上整个数据集，避免增加计算量。

8.3 示例：预测鲍鱼的年龄

使用真实的数据观察k值对模型的效果，数据来自UCI数据集，预测鲍鱼的年龄。

```
learninginaction_shifeng/chap08$ head -n 10 abalone.txt
1       0.455   0.365   0.095   0.514   0.2245  0.101   0.15    15
1       0.35    0.265   0.09    0.2255  0.0995  0.0485  0.07    7
-1      0.53    0.42    0.135   0.677   0.2565  0.1415  0.21    9
1       0.44    0.365   0.125   0.516   0.2155  0.114   0.155   10
0       0.33    0.255   0.08    0.205   0.0895  0.0395  0.055   7
0       0.425   0.3     0.095   0.3515  0.141   0.0775  0.12    8
1       0.53    0.415   0.15    0.7775  0.237   0.1415  0.33    20
        0.545   0.425   0.125   0.768   0.294   0.1495  0.26    16
        0.475   0.37    0.125   0.5095  0.2165  0.1125  0.165   9
        0.55    0.44    0.15    0.8945  0.3145  0.151   0.32    19
```

http://blog.csdn.net/

Figure 8-7: 鲍鱼数据

[python]

```python
#============预测鲍鱼的年龄================
def rssError(yArr, yHatArr):
    return ((yArr - yHatArr)**2).sum()

#========训练集上的误差
abX, abY = loadDataSet("abalone.txt")
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,训练集上的误差: ",rssError(abY[0:99], yHat01.T)
print "k=1,  训练集上的误差: ",rssError(abY[0:99], yHat1.T)
print "k=10, 训练集上的误差: ",rssError(abY[0:99], yHat10.T)
#========测试集上的误差
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,测试集上的误差: ",rssError(abY[100:199], yHat01.T)
print "k=1,  测试集上的误差: ",rssError(abY[100:199], yHat1.T)
print "k=10, 测试集上的误差: ",rssError(abY[100:199], yHat10.T)

ws = standRegres(abX[0:99], abY[0:99])
yHat = mat(abX[100:199])*ws
print "简单线性回归上的误差和: ",rssError(abY[100:199],yHat.T.A)
```

使用较小的核得到较低的训练误差，但会导致过拟合，也即对新的数据预测的效果比较糟糕。在简单线性回归中，达到了与局部加权线性回归类似的效果，在未知数据中，比较效果才能选取到最佳模型。

8.4 缩减系数来“理解”数据

当数据的特征比样本点还多时，就不能再用线性回归和LWLR，因为计算x.T*x的逆时会出错。为此引入岭回归（ridge regression）的概念。

岭回归

岭回归即在矩阵x.T*x上加一个λI从而使得矩阵非奇异，进而能对x.T*x+λI求逆，此时回归系数的计算公式变为：

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

岭回归最先用来处理特征数多于样本数的情况，在估计中加入偏差，从而得到更好的估计。引入λ来限制了所有w之和，通过引入该乘法项，能够减少不重要的参数，即缩减（shrinkage）技术。

缩减能够去掉不重要的参数，因此能够更好的理解数据，取得更好的预测效果。

使用岭回归和缩减技术，需要对特征做标准化处理，即所有特征都减去各自的均值并除以方差。

[python]

```python
#=========岭回归==========
def ridgeRegres(xMat, yMat, lam=0.2):
    xTx   = xMat.T*xMat
    denom = xTx + eye(shape(xMat)[1])*lam #得到中间结果，eye(shape(xMat)[1])得到xMat列数的单位对称阵
    if linalg.det(denom) ==0.0:
        print "this matrix is singular, cannot do inverse"
        ws = linalg.pinv(denom)*(xMat.T*yMat)
    ws = denom.I*(xMat.T*yMat)
    return ws

def ridgeTest(xArr, yArr):
    xMat       = mat(xArr)
    yMat       = mat(yArr).T
    yMean      = mean(yMat,0)            #按照行取平均，每行平均

    yMat       = yMat - yMean            #对y进行标准化
    xMeans     = mean(xMat, 0)
    xVar       = var(xMat, 0)            #按行取方差
    xMat       = (xMat - xMeans)/xVar    #对数据进行标准化
    numTestPts = 30
    wMat       = zeros((numTestPts, shape(xMat)[1]))
    for i in range(numTestPts):                 #进行numTestPts次计算岭回归，每次的系数向量都放到wMat的一行中。
        ws = ridgeRegres(xMat, yMat, exp(i-10)) #参数lam每次以exp(i-10)变化
        wMat[i,:] = ws.T
    return wMat

abX, abY     = loadDataSet("abalone.txt")
ridgeWeights = ridgeTest(abX, abY)
fig = plt.figure()
ax  = fig.add_subplot(111)
ax.plot(ridgeWeights)       #ridgeWeights,为30*8的矩阵，对矩阵画图，则以每列为一个根线，为纵坐标，横坐标为range(shape(ridgeWeights)[0])也即从0到29,第一行的横坐标为0,最后一行的行坐标为29
plt.show()
```

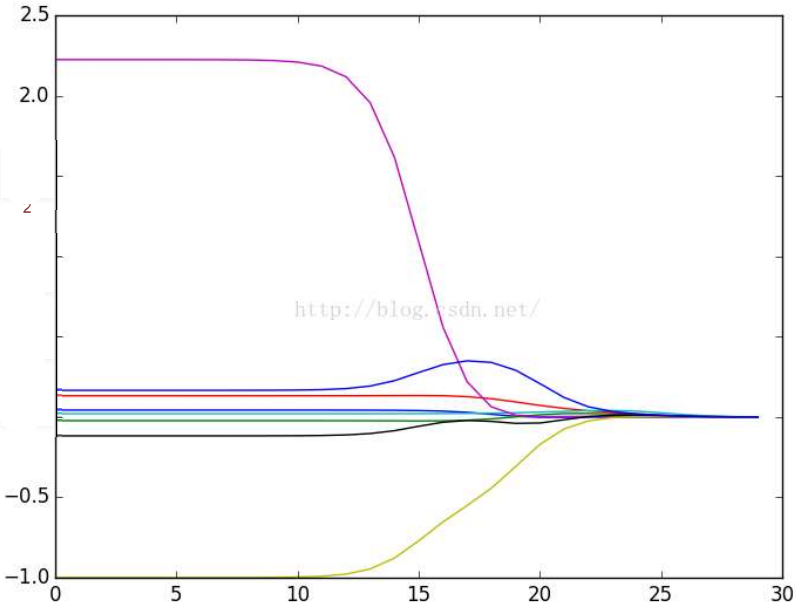


Figure 8-8: 岭回归参数与回归系数的关系

上图显示回归系数与log(λ)的关系，最左边λ最小时，8个系数的原始值与线性回归一样，最右边8个回归系数全部缩减为0，为找到最佳参数λ，还需要将ws乘以测试集比较原始测试集对比误差大小。0对应lam=exp(0-10),30对应lam=exp(30-10)

lasso

在增加约束：所有回归系数的平方和不能大于λ的条件下。普通的最小二乘法回归会得到与岭回归一样的公式。

使用普通的最小二乘法回归在两个或更多的特征相关时，可能会得出一个很大的正系数和一个很大的负系数。在上述约束中，可以约束避免这个问题。

缩减方法lasso也对回归系数做了约束：

在λ足够小的时候，一些系数会被迫缩减到0。约束条件为平方差的时候，可通过求偏导找到最优解，为绝对值形式时，需要使用二次规划，这里使用前向逐步回归。

前向逐步回归

每一步都尽可能减少误差，每一步所做的决策是对某个权重增加或减少一个很小的值。

伪代码：

```
数据标准化，使其分布满足0均值和单位方差
在每轮迭代过程中：
  设置当前最小误差lowestError为正无穷
  对每个特征：
    增大或缩小：
      改变一个系数得到一个新的W
      计算新W下的误差
      如果误差Error小于当前最小误差lowestError：设置Wbest等于当前的W
    将W设置为新的Wbest
```

http://blog.csdn.net/

[python]

```python
#=============前向逐步回归=================
def regularize(xMat):            #按行标准化
    inMat = xMat.copy()
    inMeans = mean(inMat, 0)
    inVar   = var(inMat, 0)
    inMat   = (inMat-inMeans)/inVar
    return inMat

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)        #对xMat进行标准化
    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]:         #分别计算增加或减少该特征对误差的影响
                wsTest     = ws.copy()
                wsTest[j] += eps*sign
                yTest      = xMat*wsTest
                rssE       = rssError(yMat.A, yTest.A)#得到平方误差进行比较
                if rssE < lowestError:
                    lowestError = rssE
                    wsMax       = wsTest
        ws = wsMax.copy()
        returnMat[i,:] = ws.T
    return returnMat

xArr,yArr = loadDataSet("abalone.txt")
print "步长为0.01,迭代次数为200: \n",stageWise(xArr, yArr, 0.01, 200)
print "步长为0.001,迭代次数为5000: \n",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.T
```

```
In [3]: run regression.py
步长为0.01,迭代次数为200：
[[ 0.    0.    0.   ...,  0.    0.    0.  ]
 [ 0.    0.    0.   ...,  0.    0.    0.  ]
 [ 0.    0.    0.   ...,  0.    0.    0.  ]
 ...,
 [ 0.05  0.    0.09 ..., -0.64  0.    0.36]
 [ 0.04  0.    0.09 ..., -0.64  0.    0.36]
 [ 0.05  0.    0.09 ..., -0.64  0.    0.36]]
步长为0.001,迭代次数为5000：
[[ 0.     0.     0.    ...,  0.     0.     0.   ]
 [ 0.     0.     0.    ...,  0.     0.     0.   ]
 [ 0.     0.     0.    ...,  0.     0.     0.   ]
 ...
 [ 0.045 -0.011  0.12  ..., -0.963 -0.105  0.187]
 [ 0.    -0.011  0.12  ..., -0.963 -0.105  0.187]
 [ 0.    -0.011  0.12  ..., -0.963 -0.105  0.187]]
[[ 0.0430442  -0.02274163  0.13214087  0.02075182  2.22403814 -0.99895312
  -0.[illegible]25427  0.16622915]]
```

http://blog.csdn.net/

Figure 8-9: 逐步线性回归迭代得到的回归系数

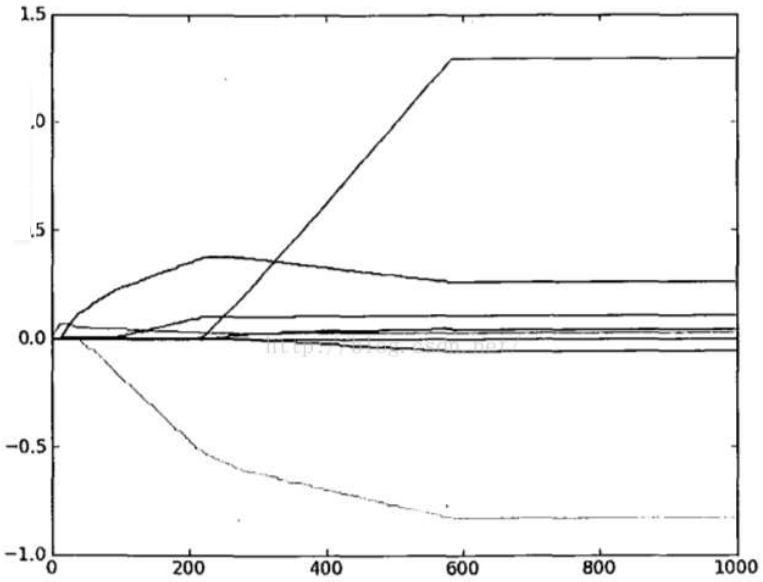


图8-7　鲍鱼数据集上执行逐步线性回归法得到的系数与迭代次数间的关系。逐步线性回归得到了与lasso相似的结果，但计算起来更加简便

逐步线性回归算法可以帮助人们理解现有的模型并做出改进。构建了一个模型后，可以运行算法找出重要的特征，这样就有可能及时停止对那些不重要特征的收集。如上面步长0.01,迭代200次后为w1和w6都为0,表明它们对目标值没有影响，也即这些特征很可能是不需要的。
当应用缩减方法（逐步线性回归或岭回归）时，模型也就增加了偏差（bias），与此同时却减少了模型的方差。

8.5 权衡偏差与方差

方差：模型训练时，模型值之间的差异。

偏差：模型预测值与数据之间的差异。

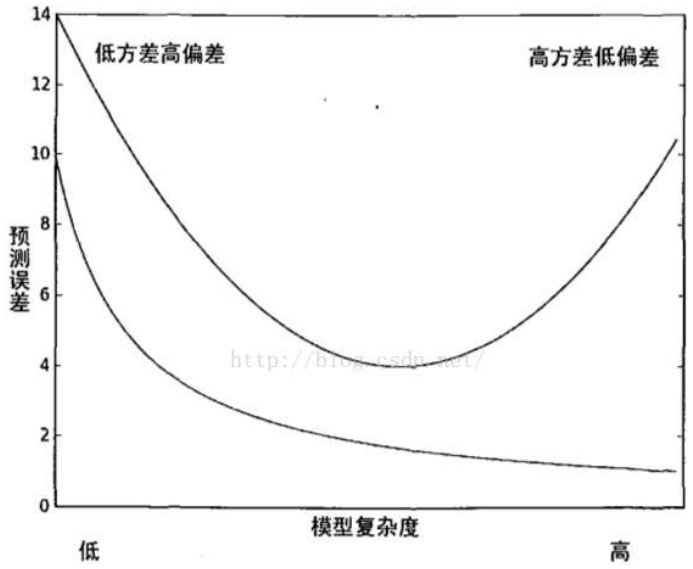


图8-8 偏差方差折中与测试误差及训练误差的关系。上面的曲线就是测试误差，在中间部分最低。为了做出最好的预测，我们应该调整模型复杂度来达到测试误差的最小值

通过缩减法，可以将一些系数缩减成很小的值或者为0，增大模型偏差，同时模型复杂度降低 。

8.6 示例：预测乐高玩具套装的价格

步骤：

1. 收集数据：用Google Shopping的API收集数据。
2. 准备数据：从返回的JSON数据中抽取价格。
3. 分析数据：可视化并观察数据。
4. 训练算法：构建不同的模型，采用逐步线性回归和直接的线性回归模型。
5. 测试算法：使用交叉验证来测试不同的模型，分析哪个效果最好。
6. 使用算法：生成数据模型。

收集数据：使用Google购物的API
待续。。。

8.7 小结

数据集上计算出的回归方程并不一定意味着它是最佳的，可以使用预测值yHat和原始值y的相关性来度量回归方程的好坏。

当数据的样本数比特征数还少时候，矩阵x.T*x的逆不能直接计算。即便当样本数比特征数多时，x.T*x的逆仍有可能无法直接计算，这是因为特征有可能高度相关。这时可以考虑使用岭回归,因为当x.T*x的逆不能计算时，它仍保证能求得回归参数。

岭回归是缩减法的一种，相当于对回归系数的大小施加了限制。 另一种很好的缩减法是lasso。Lasso难以求解，但可以使用计算简便的逐步线性回归方法来求得近似结果。

缩减法还可以看做是对一个模型增加偏差的同时减少方差。偏差方差折中是一个重要的概念，可以帮助我们理解现有模型并做出改进，从而得到更好的模型。

机器学习中的PR曲线和ROC曲线 mingtian715 2016年12月06日 16:03 8332

主要是我对周志华《机器学习》第二章模型估计与选择中一些内容的总结

机器学习之分类器性能指标之ROC曲线、AUC值

机器学习之分类器性能指标之ROC曲线、AUC值 http://www.cnblogs.com/dlml/p/4403482.html 分类器性能指标之ROC曲线、AUC值 一 r...

zdy0_2004 2015年04月08日 23:01 38397

机器学习实战-8预测数值型数据-回归 SoundSlow 2017年11月25日 15:07 227

回归和分类是机器学习两个主要的类别，通常会在面试的时候，被询问这两者之间的区别。对于我，有时候我也无法说清，虽然我感觉我已经弄懂了，但是这里还是说一说自己的理解。首先，我们学过一本书《随机过程》，这里...

机器学习入门第一天，自己对数值预测算法的理解

个人偶然的机器学习，数据预测想法，

aiyomajun 2017年08月14日 15:48 603

机器学习实战——ch8.2 回归之预测乐高玩具价格

基于《机器学习实战》第八章的案例，乐高玩具预测的实例，但是因为原始的Google网页已经失效，所以这里用新的网页进行提取。 完整的代码及注释： #-*- coding: utf-8 -*-...

sun_wangdong 2018年01月14日 14:24 132

机器学习实战---线性回归提高篇之乐高玩具套件二手价预测

作者：崔家华 编辑：王抒伟PS（欢迎访问作者个人网站：www.cuijiahua.com）线性回归零前言：本篇文章讲解线性回归的缩减方法，岭回归以及逐步线性回归，同时熟悉sklearn的岭回归...

u013709270 2017年12月24日 00:00 207

过拟合问题以及解决方法 ff19910203 2015年09月20日 21:28 1840

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！ 登录 注册

LWLR 局部加权线性回归---Python实现

局部加权线性回归 — LWLR算法介绍线性回归的一个问题是可能出现欠拟合，因为它求的是具有最小均方误差的无偏估计，可以通过引入一些偏差，从而降低均方误差。 LWLR： 算法中给待预测点附近的每个点都...

sinat_34022298 2017年08月07日 18:56 957

2

python数据分析（预测性分析与机器学习）

本文涉及主题如下所示： 预处理基于逻辑回归的分类基于支持向量机的分类基于ElasticNetCV的回归分析支持向量回归基于线性传播均值漂移算法遗传算法神经网络决策树算法...

dengjiaxing0321 2017年12月06日 14:04 715

机器学习入门例子--预测浏览量

shenzhuan7 2017年03月08日 11:34 914

下面介绍预测浏览量的例子，构建第一个机器学习的例子，其中需要的数据我放在这里了：http://pan.baidu.com/share/link?shareid=1399332518&uk=351005...

标准机器学习数据集

zhaoyuxia517 2017年09月15日 18:08 937

学好机器学习的关键是用许多不同的数据集来练习。因为对不同的问题，需要有不同的数据准备和建模方法。本文介绍了10个更受欢迎的标准机器学习数据集，可以用作练习的资源。 每个数据集均按照一定的格式介绍，以...

机器学习-采用正态贝叶斯分类器、决策树、随机森林对abalone数据集分类

1.abalone（鲍鱼）数据集描述 http://archive.ics.uci.edu/ml/datasets/Abalone 总共包含4177条数据，每条数据中包含8个特征值，一个分类（...

tiankong_ 2017年11月17日 11:58 300

机器学习算法应用场景

abc52shenghuo 2017年09月15日 13:25 2375

本文整理了60个机器学习算法应用场景实例，含分类算法应用场景20个、回归算法应用场景20个、聚类算法应用场景10个以及关联规则应用场景10个。包含了天池、DataCastle、DataFountain...

机器学习之回归预测

PancrasBohemian 2017年07月18日 14:20 727

回归预测和分类问题的区别在于，前者待预测的目标是连续变量。线性回归器使用的数据集是没过波士顿地区房价数据，在sklearn.datasets中可以找到。 其中使用了线性回归模型LinearRegre...

《机器学习实战》预测数值型数据-回归（Regression）

回归的一般方法： （1）收集数据：采用任意方法收集数据 （2）准备数据：回归需要数值型数据，标称型数据将被转化成二值型数据 （3）分析数据：绘出数据的可视化二维图将有助于对数据做出理解和分析，在采用缩...

Gamer_gyt 2016年05月14日 18:33 8451

量化交易----利用机器学习预测股票价格趋势

联系我们

请扫描二维码联系客服

webmaster@csdn.net

400-660-0108

QQ客服 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录 注册

u013547284 2017年11月04日 15:16 2120

机器学习之模型预估与选择 cqy_chen 2017年08月02日 20:09 413

损失函数与风险函数在监督学习中，对于训练样本(X,Y),其中X表示记录，Y表示对应的实际结果。假设我们学习到的模型为：f(x). 损失函数定义为L(f(x),Y).表示的是模型结果也实际结果之间的误差...

SVM中决策距离（函数）计算 resourse_sharing 2016年05月30日 16:10 3325

sklearn中 VM算法使用了liblinear和libsvm两个包，而且模型参数略有区别。 在sklearn中，SVM有SVC和SVR之分，且有四种核函数如下，所以在SVM的参数中有些模型需要...

联系我们

请扫描二维码联系客服

webmaster@csdn.net

400-660-0108

QQ客服 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！ 登录 注册