


原子力显微镜





秋水长天q

[关注](#) [发私信](#)



访问：32290次

积分：637

等级： 3

排名：千里之外

原创：34篇 转载：0篇

译文：0篇 评论：25条

文章搜索

文章分类

- [机器学习](#) (15)
- [Java基础](#) (6)
- [Hadoop](#) (6)
- [Python](#) (1)
- [线性代数](#) (1)
- [Leetcode](#) (4)
- [数据结构](#) (1)
- [JavaWeb](#) (1)

文章存档

- [2017年09月](#) (1)
- [2017年07月](#) (1)
- [2017年04月](#) (2)
- [2017年03月](#) (4)
- [2017年02月](#) (1)

[展开](#)

阅读排行

- [神经网络之BP神经网络 \(Pyt...](#) (12732)
- [迁移学习算法之TrAdaBoost](#) (6125)

图灵赠书——程序员11月书单 [【思考】Python这么厉害的原因竟然是！](#) 感恩节赠书：《深度学习》等异步社区优秀图书和作译者评选启动！ 每周荐书：京东架构、Linux内核、Python全栈

分类和Logistic回归

2017-01-05 20:59 399人阅读 [评论\(0\)](#) [收藏](#) [举报](#)

分类：
[机器学习 \(14\)](#)

版权声明：本文为博主原创文章，未经博主允许不得转载。

在上一章节中，介绍了简单的线性回归，给出一系列的离散点，利用回归模型找到一条最佳的拟合直线，其中在求解最佳拟合直线的过程中利用到了批梯度下降算法和随机梯度下降算法以及最小二乘法。可以看到，回归是一个连续的模型，那么怎样将这样一个连续的模型用在分类问题上呢，这就是这一节中将要介绍的对数回归模型。

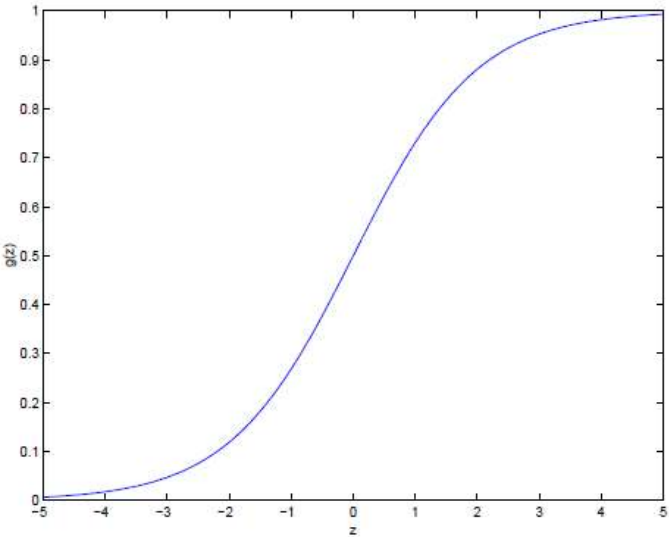
对数回归本质上就是线性回归，只不过在线性回归的基础上加上了一个函数而已，这一点和单层感知器和相似，在单层感知器的输入上加上一个激励函数，就会得到不同的输出。对数回归也是一样，只需要在我们上一节中的回归模型的基础上加上一个函数，就可以将上面的线性回归模型应用到分类问题上。

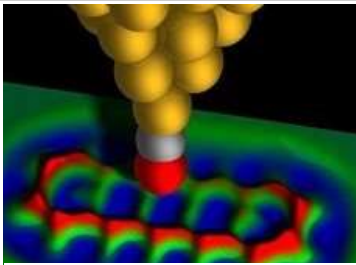
1.Sigmoid函数

Sigmoid函数也叫做逻辑函数，形式如下所示：

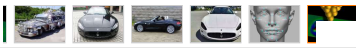
$$g(z) = \frac{1}{1 + e^{-z}}$$

函数对应的图像如下所示：





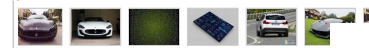
原子力显微镜



- 迁移学习算法之TrAdaBoost (14)
- 神经网络之BP神经网络 (Pyt... (6)
- 范数汇总 (3)
- MapReduce实现KNN (1)
- k-近邻算法 (0)
- 决策树学习 (0)
- 贝叶斯分类算法 (0)
- SSM框架整合 (0)
- equals和hashCode方法 (0)
- Java线程 (0)



二手玛莎拉蒂



最新评论

- 神经网络之BP神经网络 (Python实现)
香香的七仔 : 使用V (输入层节点数×输出层节点数) 作为隐藏层节点数 (88个) 可以将正确率提升到大概95%
- 范数汇总
yujianlan : 你好, 请问F范数和21范数之间怎么转化呢? 谢谢
- 迁移学习算法之TrAdaBoost
confina : 想问下什么样本的权重怎么和基本的学习器结合啊 哪一种学习器会考虑样本的权重呢
- 范数汇总
秋水长天q : @aoqiulei7117机器学习中的很多地方都会有范数的应用, 比如说公式中的一些约束之类的:
- 范数汇总
aoqiulei7117 : 范数有啥应用呢
- 神经网络之BP神经网络 (Python实现)
weixin_39518878 : 博主你好, 请问这段代码的迭代次数在哪里啊
- 神经网络之BP神经网络 (Python实现)
Coding_ForFun : @Fengming1220:用matlab就可以转化为mat格式了~
- 神经网络之BP神经网络 (Python实现)
Fengming1220 : 你好, 请问原始数据怎么转化为.mat格式文件?
- MapReduce实现KNN
binyet : 您好, 可以看一下您的Instance这个类的实现嘛? 谢谢你
- 迁移学习算法之TrAdaBoost
coda469993172 : 后来推导了一下发现和论文里面的式子是一样的 $B^{-1/2}$ 和 $1/2 \ln$

可以看到, 对于不同的输入 z , 函数的输出总是在0到1之间, 当 z 的值为0是, 函数值是0.5。正是因为函数的这样的性质我们可以将线性回归中的输出值作为 z 作用大函数上, 最终的结果就是一个在0到1之间的数字, 以0作为分界线, 当线性回归的结果大于0的时候, 分类结果就是1, 当线性回归的结果小于0的时候, 分类的结果就是0。有了这种思想, 我们就可以将回归函数重新定义如下:

$$h_{\theta}(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

2 模型定义

对数回归可以用来分类0/1问题, 也就是预测结果属于0或者1的二分类问题, 我们假设问题满足伯努利分布, 也就是如下的形式:

$$\begin{aligned} P(y = 1 | x; \theta) &= h_{\theta}(x) \\ P(y = 0 | x; \theta) &= 1 - h_{\theta}(x) \end{aligned}$$



其中The World's #1 Online Equation Editor是我们要求取的参数, 将上述的式子用一个式子表示就是如下

的形式:

$$p(y | x; \theta) = (h_{\theta}(x))^y (1 - h_{\theta}(x))^{1-y}$$

假设所有的样本都是独立分布的, 利用最大似然估计

$$\begin{aligned} L(\theta) &= p(\vec{y} | X; \theta) \\ &= \prod_{i=1}^m p(y^{(i)} | x^{(i)}; \theta) \\ &= \prod_{i=1}^m (h_{\theta}(x^{(i)}))^{y^{(i)}} (1 - h_{\theta}(x^{(i)}))^{1-y^{(i)}} \end{aligned}$$

对式子两边同时取对数, 得到

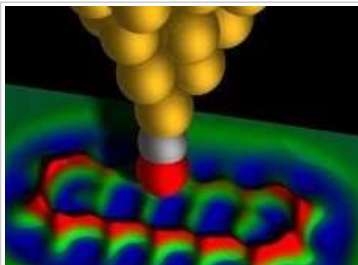
$$\begin{aligned} \ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^m y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)})) \end{aligned}$$



现在式子中只有一个The World's #1 Online Equation Editor是未知的, 现在的目标是要将式子最大化, 采用梯度上升的做法

$$\theta := \theta + \nabla_{\theta} L(\theta)$$

对单一的样本的似然函数进行求导得:



原子力显微镜



$$\begin{aligned}
 \frac{\partial}{\partial \theta_j} \ell(\theta) &= \left(y \frac{1}{g(\theta^T x)} - (1-y) \frac{1}{1-g(\theta^T x)} \right) \frac{\partial}{\partial \theta_j} g(\theta^T x) \\
 &= \left(y \frac{1}{g(\theta^T x)} - (1-y) \frac{1}{1-g(\theta^T x)} \right) g(\theta^T x)(1-g(\theta^T x)) \frac{\partial}{\partial \theta_j} \theta^T x \\
 &= (y(1-g(\theta^T x)) - (1-y)g(\theta^T x)) x_j \\
 &= (y - h_\theta(x)) x_j
 \end{aligned}$$

可以看到，这里的结果和线性回归中的结果是一样的，所以采用梯度上升的算法对

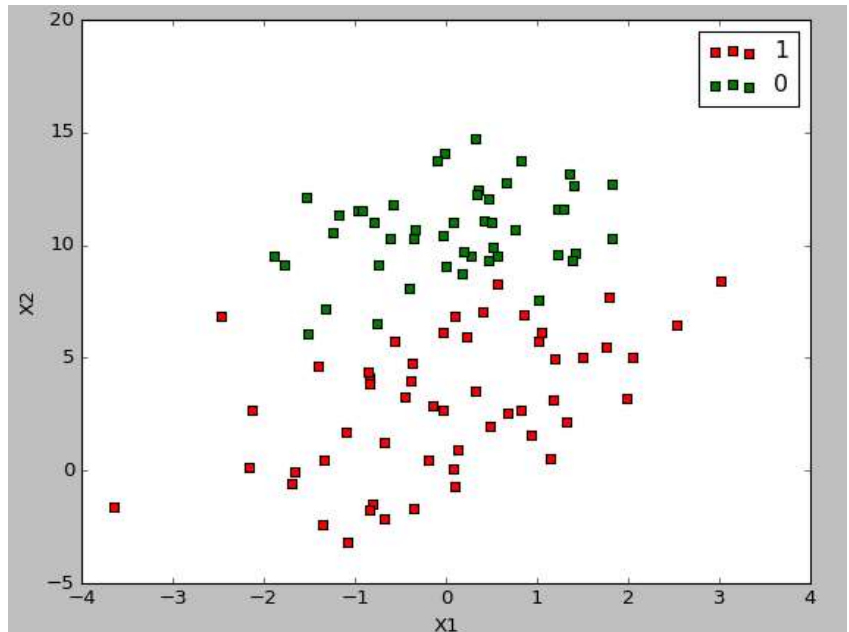


The World's #1 Online Equation Editor 进行更新的过程为：

$$\theta := \theta + \alpha \sum_{i=1}^m (y^i - h_\theta(x^i)) x_j^i$$

3 实验过程

本次实验中采用的数据集如下：

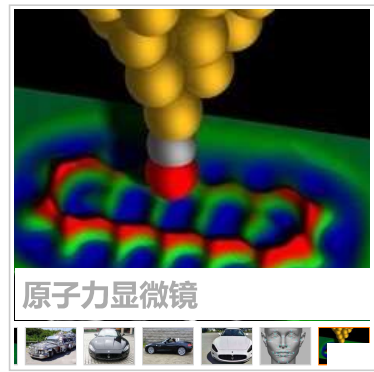


3.1 批梯度上升算法实现过程

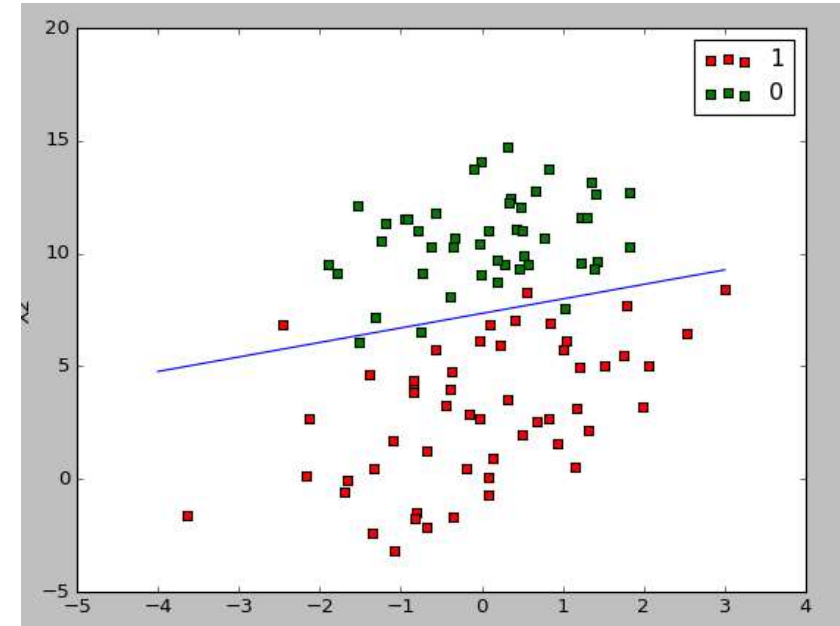
```

[python] view plain copy print ?
01. <span style="font-size:14px;">#批梯度上升算法
02. def gradAscent(X,Y):
03.     X=np.array(X)
04.     m,n=np.shape(X)
05.     Y=np.array(Y).reshape(m,1) #转化为列向量的形式
06.     alpha = 0.01 #定义步长
07.     iteration=400
08.     thea = np.ones((n, 1)) #定义初始的点
09.     for k in range(iteration):
10.         H=sigmoid(X.dot(thea))
11.         error=Y-H
12.         thea=thea+alpha*X.T.dot(error)
13.     return thea

```



实验结果：

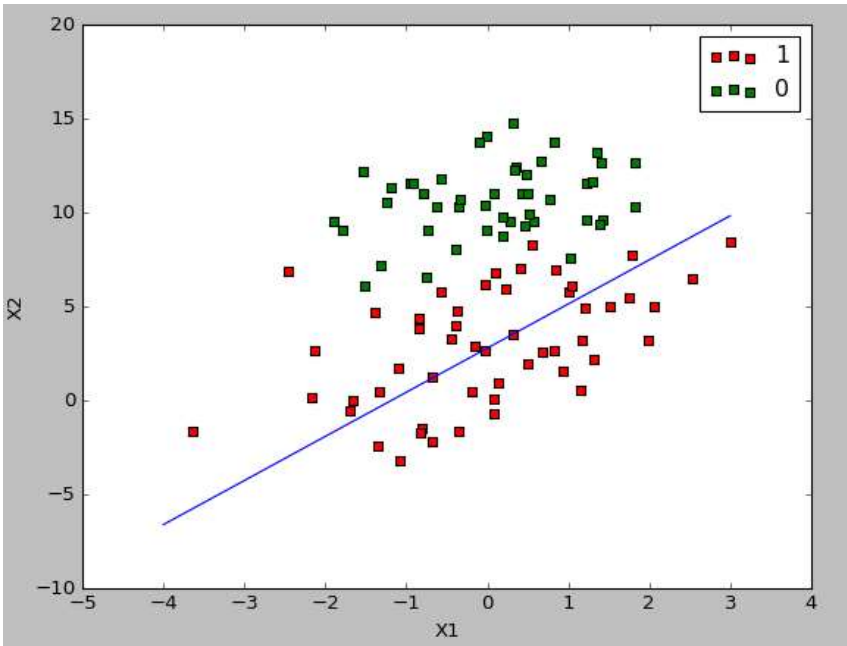


3.2 随机梯度上升算法

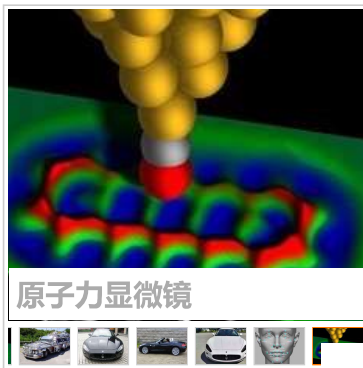
[python] view plain copy print ?

```
01. <span style="font-size:14px;">#随机梯度算法
02. def stocgradAscent(X,Y):
03.     X=np.array(X)
04.     m,n=np.shape(X)
05.     Y=np.array(Y).reshape(m,1)
06.     thea=np.ones((n,1))
07.     alpha = 0.01 # 定义步长
08.     for i in range(m):
09.         H=sigmoid(X[i].dot(thea))
10.         error=Y[i]-H
11.         thea=thea+alpha*error*X[i].reshape(n,1)
12.     return thea</span>
```

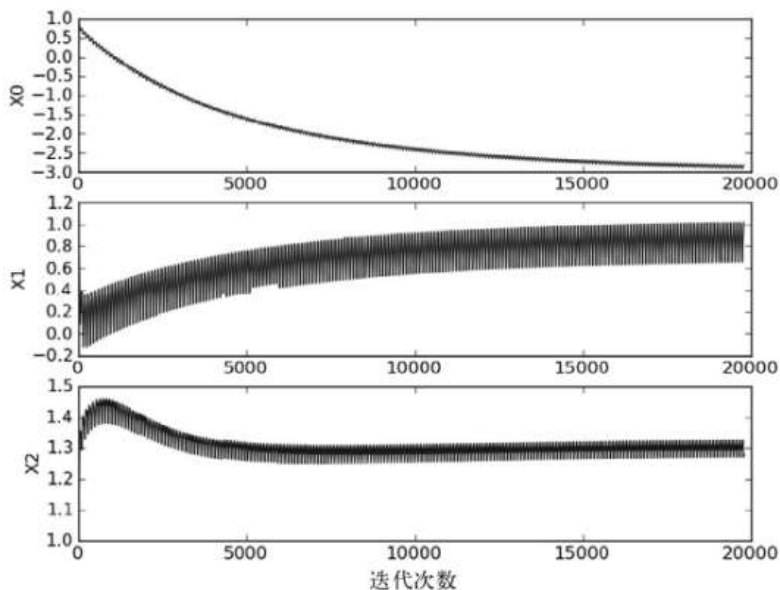
实验结果如下：



可以看到，随机梯度上升算法的结果在本例中并不比批梯度上升算法的结果好，但是只从这样的结果中直接的去比较算法的好坏是不公平的，在批梯度上升中，算法在整个的数据集上经过了 400次的迭代过



程。而判断一个算法好坏的可靠方法是看它是否收敛，也就是在经过有限步骤的迭代之后，参数是否到达了一个稳定值。在《机器学习实战》一书中，针对这个例子给出了系数迭代过程中变化趋势图：



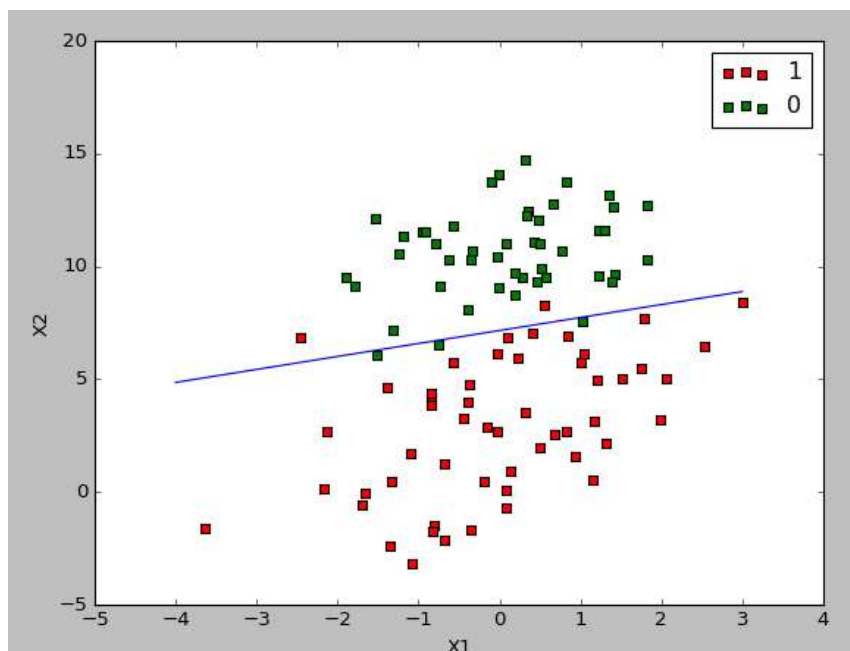
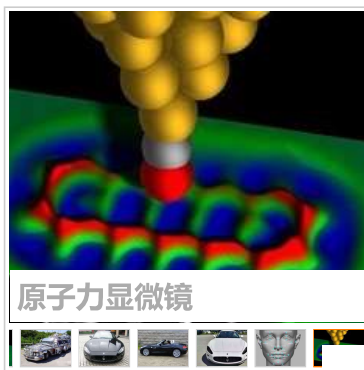
可以看到X的系数在达到稳定的时候经过的次数是不一样的，在大的波动停止后，还会有一些小的周期性波动，产生这种现象的原因是存在着一些不能正确分类的样本点，在每次迭代的时候会引发系数的剧烈改变，我们针对随机梯度上升算法做出如下的改进：

- 1.在每次迭代的过程中，改变步长alpha的值，这样会减小波动性。
- 2.随机的选择样本点进行梯度上升来更新系数值，这样可以避免周期性的波动，在每次迭代完成之后从列表中删除这个已经选择的值。

该改进之后的随机梯度上升算法如下：

```
[python] view plain copy print ?
01. <span style="font-size:14px;">#改进的随机梯度下降上升算法
02. def stocgradAscent1(X,Y):
03.     X=np.array(X)
04.     m,n=np.shape(X)
05.     Y=np.array(Y).reshape(m,1)
06.     iteration = 400
07.     thea=np.ones((n,1))
08.     for j in range(iteration):
09.         dataindex=[]
10.         for k in range(m):
11.             dataindex.append(k)
12.         for i in range(m):
13.             alpha=4/(1.0+i+j)+0.01
14.             randinx=int(random.uniform(0,len(dataindex)))
15.             h=sigmoid(X[i].dot(thea))
16.             error=Y[i]-h
17.             thea=thea+alpha*error*X[i].reshape(n,1)
18.             dataindex.remove(dataindex[randinx])
19.     return thea</span>
```

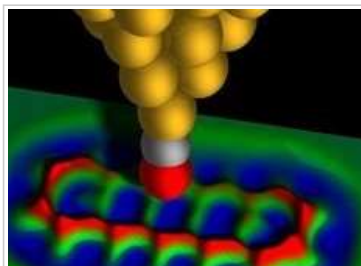
该改进之后的效果如下：



可以看到，在改进之后，随机梯度上升算法一样可以取得很好的效果。

文中实验所有的代码如下：

```
[python] view plain copy print ?
01. <span style="font-size:14px;">#对数回归
02. import numpy as np
03. import matplotlib.pyplot as plt
04. import random
05. #读取文本数据，三维样本
06. def ReadFile(filename):
07.     X=[]
08.     Y=[]
09.     lines=open(filename,encoding='utf-8').readlines()
10.     for i in range(len(lines)):
11.         line=lines[i].strip().split()
12.         X.append([1.0,float(line[0]),float(line[1])])
13.         Y.append(int(line[2]))
14.     return X,Y
15.
16. #定义sigmoid函数
17. def sigmoid(x):
18.     return 1.0/(1+np.exp(-x))
19.
20. #批梯度上升算法
21. def gradAscent(X,Y):
22.     X=np.array(X)
23.     m,n=np.shape(X)
24.     Y=np.array(Y).reshape(m,1) #转化为列向量的形式
25.     alpha = 0.01 #定义步长
26.     iteration=400
27.     thea = np.ones((n, 1)) #定义初始的点
28.     for k in range(iteration):
29.         H=sigmoid(X.dot(thea))
30.         error=Y-H
31.         thea=thea+alpha*X.T.dot(error)
32.     return thea
33. #随机梯度算法
34. def stocgradAscent(X,Y):
35.     X=np.array(X)
```



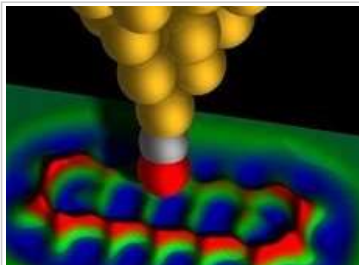
原子力显微镜



```

36.     m,n=np.shape(X)
37.     Y=np.array(Y).reshape(m,1)
38.     thea=np.ones((n,1))
39.     alpha = 0.01 # 定义步长
40.     for i in range(m):
41.         H=sigmoid(X[i].dot(thea))
42.         error=Y[i]-H
43.         thea=thea+alpha*error*X[i].reshape(n,1)
44.     return thea
45. #改进的随机梯度下降上升算法
46. def stocgradAscent1(X,Y):
47.     X=np.array(X)
48.     m,n=np.shape(X)
49.     Y=np.array(Y).reshape(m,1)
50.     iteration = 400
51.     thea=np.ones((n,1))
52.     for j in range(iteration):
53.         dataindex=[]
54.         for k in range(m):
55.             dataindex.append(k)
56.         for i in range(m):
57.             alpha=4/(1.0+i+j)+0.01
58.             randinx=int(random.uniform(0,len(dataindex)))
59.             h=sigmoid(X[i].dot(thea))
60.             error=Y[i]-h
61.             thea=thea+alpha*error*X[i].reshape(n,1)
62.             dataindex.remove(dataindex[randinx])
63.     return thea
64.
65.
66.
67. #作出分类曲线
68. def plot(X,Y,thea):
69.     X = np.array(X)
70.     m, n = np.shape(X)
71.     Y = np.array(Y).reshape(m, 1)
72.     xcord1=[];ycord1=[]
73.     xcord2=[];ycord2=[]
74.     for i in range(m):
75.         if Y[i]==1:
76.             xcord1.append(X[i][1])
77.             ycord1.append(X[i][2])
78.         if Y[i]==0:
79.             xcord2.append(X[i][1])
80.             ycord2.append(X[i][2])
81.     fig=plt.figure()
82.     ax=fig.add_subplot(111)
83.     type1=ax.scatter(xcord1, ycord1, s=30, c='red', marker='s',label='1')
84.     type2=ax.scatter(xcord2, ycord2, s=30, c='green',marker='s',label='0')
85.     plt.legend(loc='upper right')
86.     x=np.arange(-4,4)
87.     y=((-thea[0] - thea[1] * x) / thea[2])
88.     ax.plot(x, y)
89.     plt.xlabel("X1")
90.     plt.ylabel("X2")
91.     plt.show()
92.
93.
94. if __name__=="__main__":

```



原子力显微镜



```
95. X,Y=ReadFile('ex2data1.txt')
96. #thea=stocgradAscent1(X,Y)
97. thea=gradAscent(X,Y)
98. #thea=stocgradAscent(X,Y)
99. print(thea)
100. plot(X,Y,thea)
101.
102. </span>
```

4 利用Logistic解决多分类问题

Logistic主要应用在二分类问题中，我们上述的实验就是在一个二分类的问题上进行的，Logistic同样也可以应用在多分类问题中。在多分类问题中，我们可以将多分类问题看作是多个二分类问题。在本实验中，采用传统的手写字识别的数据集，利用Logistic回归实现手写字识别的思想大致如下：这是一个包含10个类别的分类问题，将整个识别问题看作为多个二分类问题。例如在识别数字0的时候，我们可以将整个问题看作为0和非0处理，将属于类别0的数据项看作是1，将不属于数字0的数据项看作是0，对于其他的类别做同



样的处理。在二分类问题中，求得的 [The World's #1 Online Equation Editor](#) 是一个向量的形式，而在多分类问题中，因为将问题看作是多分类问题，所以训练的结果是多分类器，最终得到的



The World's #1 Online Equation Editor 是一个矩阵，矩阵的每一行代表一个分类器。

实验的结果如下：

9的识别结果是:9

9的识别结果是:9

9的识别结果是:9

9的识别结果是:9

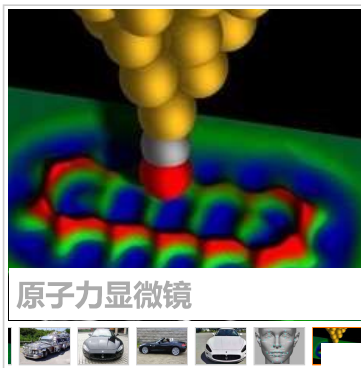
9的识别结果是:9

true rate is: 0.9588

本实验中所有代码如下：

[python] [view plain](#) [copy](#) [print](#) ?

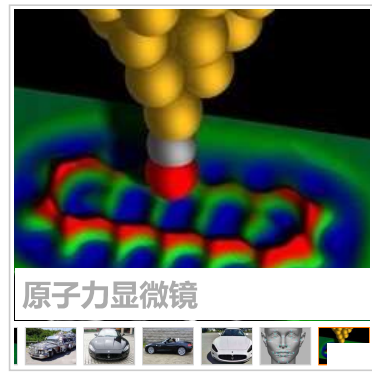
```
01. <span style="font-size:14px;">#利用logistic回归实现多分类问题
02.
03. import numpy as np
04. from os import listdir
05.
06. #定义数据读取函数
07. def LoadData(dir):
08.     filelist=listdir(dir)
09.     m=len(filelist)
10.     X=np.zeros((m,1024))
11.     Y=np.zeros((m,1))
12.     for i in range(m):
13.         arr=np.zeros((1,1024))
14.         filename=filelist[i]
15.         fr=open('%s/%s' %(dir,filename))
16.         for j in range(32):
```

```

17.         line=fr.readline()
18.         for k in range(32):
19.             arr[0, 32 * j + k] = int(line[k])
20.         X[i,:]=arr
21.         name=filename.split('.')[0]
22.         label=name.split('_')[0]
23.         Y[i]=int(label)
24.     return X,Y
25.
26. #定义Sigmoid函数
27. def sigmoid(x):
28.     return .5 * (1 + np.tanh(.5 * x))
29.     #return 1.0/(1+np.exp(-x))
30.
31. #利用批梯度上升算法为每个类别训练一个分类器
32. def gradAscent(X,Y):
33.     m,n=np.shape(X)
34.     num_labels=10 #类别的数目
35.     all_thea=np.ones((num_labels,n)) #定义将要返回的系数矩阵
36.     alpha = 0.01 #定义步长
37.     iteration=400
38.     for i in range(num_labels):
39.         index = []
40.         thea = np.ones((n, 1)) #定义初始的点
41.         for j in range(m):
42.             if Y[j]==i:
43.                 index.append(1)
44.             else:
45.                 index.append(0)
46.         y=np.array(index).reshape(m,1)
47.         for k in range(iteration):
48.             H=sigmoid(X.dot(thea))
49.             error=y-H
50.             thea=thea+alpha*X.T.dot(error)
51.             all_thea[i,:]=thea.reshape(1,n)
52.     return all_thea
53.
54. #定义预测函数
55. def predict(X,thea):
56.     m,n=np.shape(X)
57.     num_labels = 10 # 类别的数目
58.     lables=[]
59.     r=sigmoid(X.dot(thea.T))
60.     for i in range(m):
61.         lables.append(np.argmax(r[i]))
62.     return lables
63.
64. if __name__=="__main__":
65.     TrainX,TrainY=LoadData("trainingDigits")
66.     TestX,TestY=LoadData("testDigits")
67.     all_thea=gradAscent(TrainX,TrainY)
68.     labels=predict(TestX,all_thea)
69.     y=np.array(labels).reshape(np.shape(TestX)[0],1)
70.     k=0
71.     for i in range(np.shape(TestX)[0]):
72.         print('%d的识别结果是:%d'%(TestY[i],y[i]))
73.         if y[i]==TestY[i]:
74.             k=k+1
75.     print('true rate is:', '%.4f' %(k/(np.shape(TestY)[0])))

```



原子力显微镜

76.
77.
78.
79.
80.

参考文献

Ng机器学习讲义
<http://www.cnblogs.com/netuml/p/5727745.html>
<http://blog.csdn.net/stdcoutzyx/article/details/9113681>

顶 0 踩 1

- 上一篇 线性回归
- 下一篇 Python IO编程

相关文章推荐

- 二分类模型性能评价 (R语言 , logistic回归 , RO...
- MySQL在微信支付下的高可用运营--莫晓东
- 支持向量机SVM(一):支持向量机SVM的推倒：从I...
- 容器技术在58同城的实践--姚远
- 使用Logistic回归对MNIST手写字符进行分类识别
- SDCC 2017之容器技术实战线上峰会
- 二分类模型性能评价 (R语言 , logistic回归 , RO...
- SDCC 2017之数据库技术实战线上峰会
- 机器学习：多分类的logistic回归
- 腾讯云容器服务架构实现介绍--董晓杰
- 分类模型的性能评估——以SAS Logistic回归为例...
- 微博热点事件背后的数据库运维心得--张冬洪
- 分类模型的性能评估——以SAS Logistic回归为例
- 机器学习-基于Logistic回归和Sigmoid函数的分类
- Logistic回归 (LR) 分类器
- 分类算法：Logistic回归

人工智能掘金时代 首选Pyt

查看评论

暂无评论

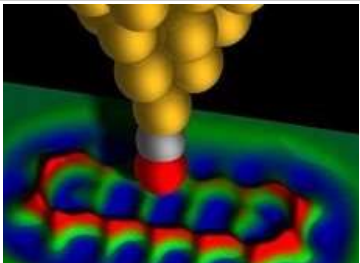
发表评论

用户名：wujuxKkoolerter

评论内容：

提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场



原子力显微镜



京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved