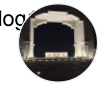
Logistic回归及梯度上升算法

原创 2016年12月06日 20:36:26 标签：机器学习 (http://so.csdn.net/so/search/s.do?q=机器学习&t=blog)

1975

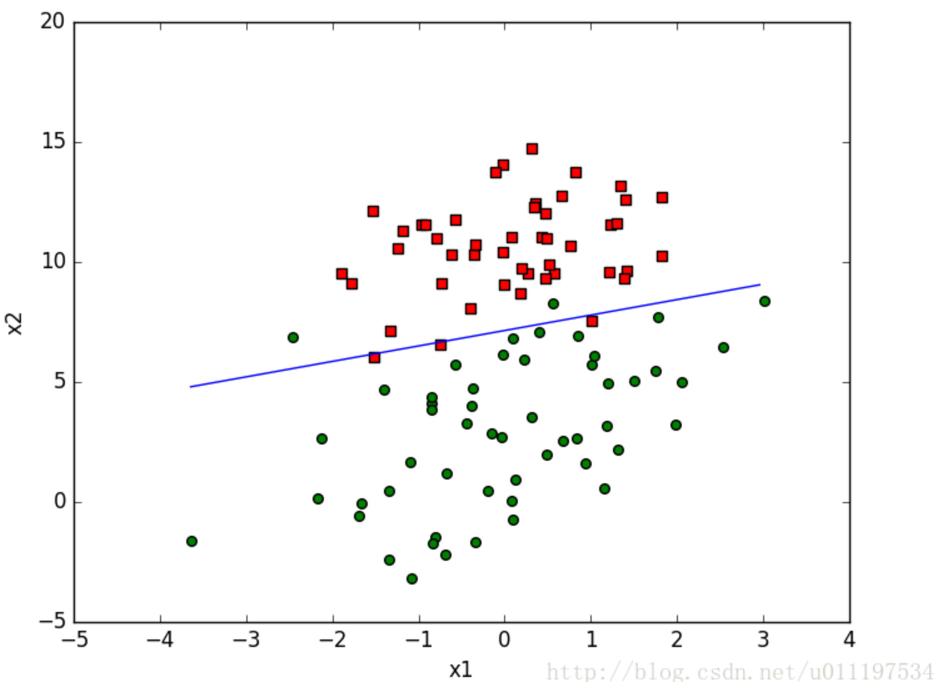


Summit_Yue (http://blo...
+关注
(http://blog.csdn.net/u011197534)

开始学习机器学习算法的时候就接触了Logistic回归及梯度下降法，但是当时并没有深入去自己推导一下公式，写一写代码，现在学习xgboost的时候又碰到Logistic回归相关的知识，干脆自己推一遍，写一下代码吧。。

逻辑回归

逻辑回归可以用来进行回归与分类，两者仅有略微不同，主体算法是一样的，本文以分类进行讲解。如下图二分类问题，我们希望找到一个直线（高维空间为超平面）来将数据划分开。



这样的线性边界可以表示为：$\theta_0 x_1 + \theta_1 x_2 + \ldots + \theta_m x_m = \theta^T x$
上式右边x为向量。
我们取预测函数为Sigmoid函数，Sigmoid函数有一个很棒的特点是它的导数
则预测函数可表示为：

$$P(y=1|x;\theta) = h_\theta(x)$$
$$P(y=0|x;\theta) = 1 - h_\theta(x)$$

将这两个式子合并一下：

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

显然：
当y=0的时候上式等价于$P(y=0|x;\theta) = 1-h_\theta(x)$
当y=1的时候上式等价于$P(y=1|x;\theta) = h_\theta(x)$

取似然函数
我们的目的就是求解似然函数的最大值，为了方便求解，我们取对数似然函数如下：

$$logL(\theta) = \sum_{i=1}^{m}(y_i log h_\theta(x^{(i)}) + (1-y_i)log(1-h_\theta(x^{(i)})))$$

如此，我们就可以使用如下的式子进行梯度上升算法迭代更新θ的取值：

$$\theta_j = \theta_j + \alpha \frac{\partial logL(\theta))}{\partial \theta_j}$$

下面求解$\frac{\partial logL(\theta))}{\partial \theta}$

$$\frac{\partial logL(\theta))}{\partial \theta_j} = \sum_{i=1}^{m}(y_i \frac{1}{h_\theta(x^{(i)})} + (1-y_i)\frac{1}{1-h_\theta(x^{(i)})})\frac{\partial h_\theta(x^{(i)})}{\partial \theta_j}$$

$$= \sum_{i=1}^{m}(y_i \frac{1}{g(\theta^T x^{(i)})} + (1-y_i)\frac{1}{1-g(\theta^T x^{(i)})})g(\theta^T x^{(i)})(1-g(\theta^T x^{(i)}))x^{(i)}$$

$$= \sum_{i=1}^{m}(y_i(1-g(\theta^T x^{(i)})) - (1-y_i)g(\theta^T x^{(i)}))x^{(i)}$$

$$= \sum_{i=1}^{m}(y_i - g(\theta^T x^{(i)}))x^{(i)}$$

$$= \sum_{i=1}^{m}(y_i - h_\theta(x^{(i)}))x^{(i)}$$

所以权重的迭代更新式为：

$$\theta_j = \theta_j + \alpha \sum_{i=1}^{m}(y_i - h_\theta(x^{(i)}))x^{(i)}$$

其中α为更新率

梯度上升

有了以上的逻辑回归的理论基础，下面我们编程实现这一步骤。就以第一张图的样本为例进行，样本维数为2维，采用梯度上升算法进行迭代。
迭代步数自己选择

批量梯度上升
批量梯度上升每进行一次迭代更新就会计算所有样本，因此得到的模型正确率比较高，但同时计算复杂度高，算法耗时。计算过程如下：
1.首先根据权重和训练样本计算估计值

$$h = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} \\ \cdots & \cdots & \cdots \\ x_1^{(m)} & x_2^{(m)} & x_3^{(m)} \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{pmatrix} = \begin{pmatrix} \hat{y_1} \\ \hat{y_2} \\ \cdots \\ \hat{y_m} \end{pmatrix}$$

2.计算误差

$$error = y - \hat{y} = \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_m \end{pmatrix} - \begin{pmatrix} \hat{y_1} \\ \hat{y_2} \\ \cdots \\ \hat{y_m} \end{pmatrix} = \begin{pmatrix} e_1 \\ e_2 \\ \cdots \\ e_m \end{pmatrix}$$

3.迭代更新

$$w = w + \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(m)} \\ x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(m)} \\ x_3^{(1)} & x_3^{(2)} & \cdots & x_3^{(m)} \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ \cdots \\ e_m \end{pmatrix} \rightarrow \begin{pmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \Delta\theta_3 \end{pmatrix}$$

随机梯度上升

根据样本数量进行迭代，每计算一个样本就进行一次更新，过程如下：

1.计算$x^{(i)}$样本对应的估计值

$$h = \begin{pmatrix} x_1^{(i)} & x_2^{(i)} & x_3^{(i)} \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{pmatrix}$$

2.计算误差

$$error = y_i - h(number)$$

注意，此处的误差是个数，不再是个向量

3.迭代更新

$$w = w + \alpha \begin{pmatrix} x_1^{(i)} \\ x_2^{(i)} \\ x_3^{(i)} \end{pmatrix} error$$

以上步骤更新m次。

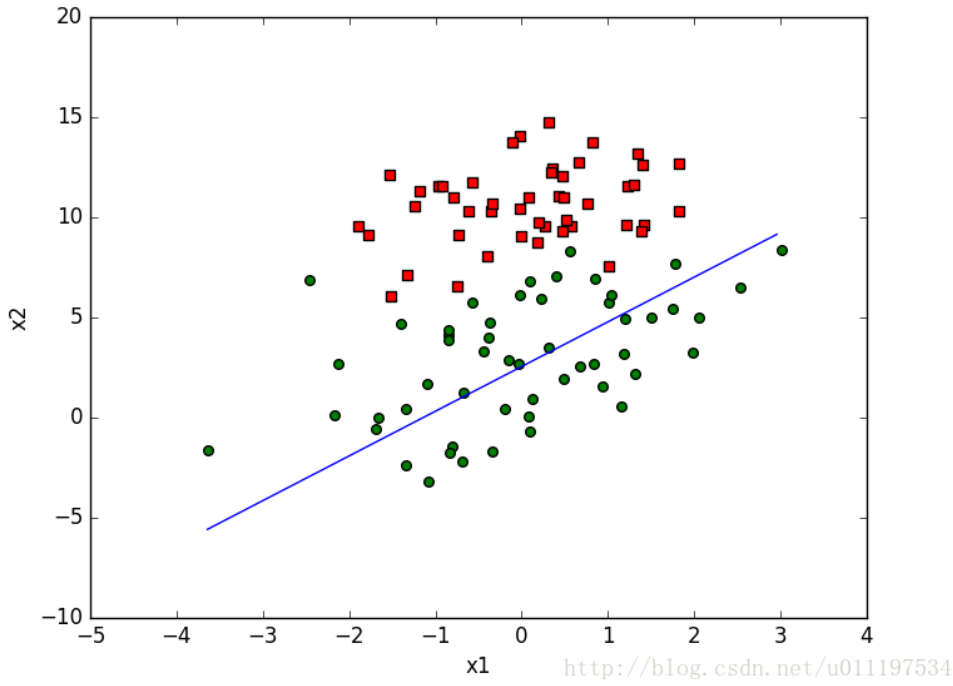
代码如下

```
import numpy as np
import re
from pandas import DataFrame
import time as time
import matplotlib.pyplot as plt
import math

def get_data(filename):                 #读取数据
    f = open(filename)
    data = DataFrame(columns=['x0','x1','x2','label']) #构造DataFrame存放数据，列名为x与y
    line = f.readline()
    line = line.strip()
    p = re.compile(r'\s+')              #由于数据由若干个空格分隔，构造正则表达式分隔
    while line:
        line = line.strip()
        linedata = p.split(line)
        data.set_value(len(data),['x0','x1','x2','label'],[1,float(linedata[0]),float(linedata[1]),int(lined
ata[2])]) #数据存入DataFrame
        line = f.readline()
    return np.array(data.loc[:,['x0','x1','x2']]),np.array(data['label'])
def sigmoid(x):
    return 1.0/(1+np.exp(-x))
def stocGradAscent(dataMat,labelMat,alpha = 0.01):   #随机梯度上升
    start_time = time.time()                         #记录程序开始时间
    m,n = dataMat.shape
    weights = np.ones((n,1))                         #分配权值为1
    for i in range(m):
        h = sigmoid(np.dot(dataMat[i],weights).astype('int64')) #注意：这里两个二维数组做内积后得到的dtype是
object,需要转换成int64
        error = labelMat[i]-h                        #误差
        weights = weights + alpha*dataMat[i].reshape((3,1))*error #更新权重
    duration = time.time()-start_time
    print('time:',duration)
    return weights
def gradAscent(dataMat,labelMat,alpha = 0.01,maxstep = 1000): #批量梯度上升
    start_time = time.time()
    m,n = dataMat.shape
    weights = np.ones((n,1))
    for i in range(maxstep):
        h = sigmoid(np.dot(dataMat,weights).astype('int64'))  #这里直接进行矩阵运算
        labelMat = labelMat.reshape((100,1))                  #label本为一维,转成2维
        error = labelMat-h                                    #批量计算误差
        weights = weights + alpha*np.dot(dataMat.T,error)     #更新权重
    duration = time.time()-start_time
    print('time:',duration)
    return weights
def betterStoGradAscent(dataMat,labelMat,alpha = 0.01,maxstep = 150):
    start_time = time.time()
    m,n = dataMat.shape
    weights = np.ones((n,1))
    for j in range(maxstep):
        for i in range(m):
            alpha = 4/(1+i+j) + 0.01                          #设置更新率随迭代而减小
            h = sigmoid(np.dot(dataMat[i],weights).astype('int64'))
            error = labelMat[i]-h
            weights = weights + alpha*dataMat[i].reshape((3,1))*error
    duration = time.time()-start_time
    print('time:',duration)
    return weights
def show(dataMat, labelMat, weights):
    #dataMat = np.mat(dataMat)
    #labelMat = np.mat(labelMat)
    m,n = dataMat.shape
```

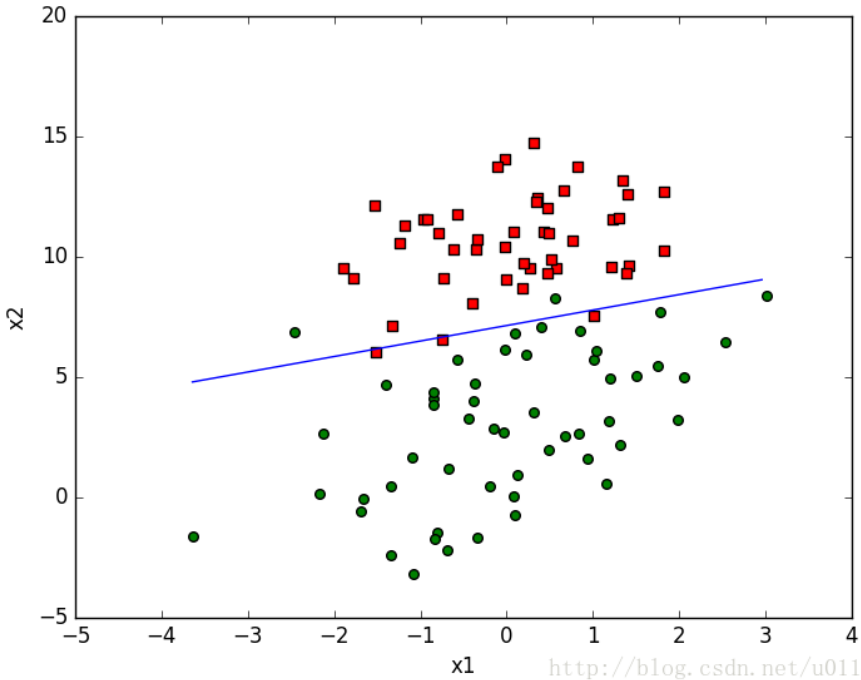
```
    min_x = min(dataMat[:, 1])
    max_x = max(dataMat[:, 1])
    xcoord1 = []; ycoord1 = []
    xcoord2 = []; ycoord2 = []
    for i in range(m):
        if int(labelMat[i]) == 0:
            xcoord1.append(dataMat[i, 1]); ycoord1.append(dataMat[i, 2])
        elif int(labelMat[i]) == 1:
            xcoord2.append(dataMat[i, 1]); ycoord2.append(dataMat[i, 2])
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.scatter(xcoord1, ycoord1, s=30, c="red", marker="s")
    ax.scatter(xcoord2, ycoord2, s=30, c="green")
    x = np.arange(min_x, max_x, 0.1)
    y = (-float(weights[0]) - float(weights[1])*x) / float(weights[2])
    ax.plot(x, y)
    plt.xlabel("x1"); plt.ylabel("x2")
    plt.show()

if __name__=='__main__':
    dataMat,labelMat = get_data('data1.txt')
    weights = gradAscent(dataMat,labelMat)
    show(dataMat,labelMat,weights)
```

效果

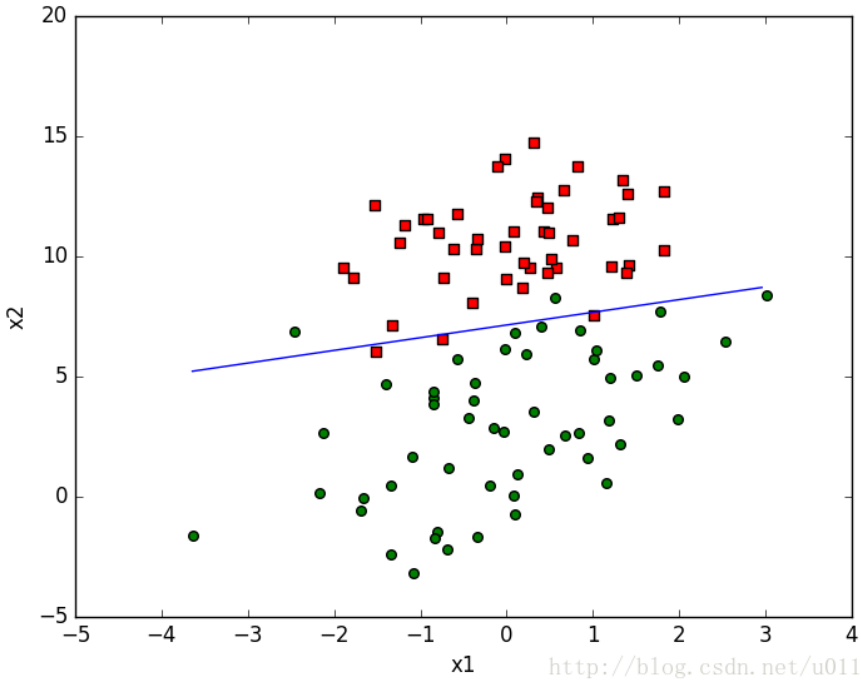
随机梯度：



批量梯度：



改进的随机梯度：
实际就是进行多次随机梯度，学习率随着迭代步数而减小。



发表你的评论

(http://my.csdn.net/forever2008250)

ssfaker (/ssfaker) 2017-09-28 13:52 1楼

(/ssfaker)求导的前两行计算错误，修改一下

回复

0

相关文章推荐

ubuntu下MySql部分配置 (http://blog.csdn.net/u011197534/article/details/53408739)

安装与卸载ubuntu 14.04下安装mysqlsudo apt-get install mysql-serverubuntu 14.04下完全卸载mysqlsudo rm /var/lib/mys...

u011197534 (http://blog.csdn.net/u011197534) 2016年11月30日 15:27 250

在phpmyadmin中创建存储过程并在php中调用 (http://blog.csdn.net/coder9999/articl...

语法： CREATE PROCEDURE p() BEGIN END CREATE PROCEDURE productpricing() BEGIN ...

coder9999 (http://blog.csdn.net/coder9999) 2017年04月16日 11:12 1174

内容举报

返回顶部

[机器学习]Logistic回归梯度上升法与改进的随机梯度上升算法 (http://blog.csdn.net/sinat...

http://sbp810050504.blog.51cto.com/2799422/1608064/这个网址解释了多维空间的sigmoid函数与梯度上升算法的原理，大家可以参考一下。 from n...

sinat_33487968 (http://blog.csdn.net/sinat_33487968) 2017年04月25日 19:14 136

logistics回归--梯度上升算法以及改进--用于二分类 (http://blog.csdn.net/Gentle_Guan/a...

1.sigmoid函数应用 logistics回归是用来分类的，并且属于监督学习，分类也是仅限于二分类，就是结果非0即1 (这种函数通常称作跃阶函数) 这个时候就出现问题了 01之间的分界点怎么处理？...

Gentle_Guan (http://blog.csdn.net/Gentle_Guan) 2017年07月30日 22:15 443

Logistic回归梯度上升分类法 (http://blog.csdn.net/qq_34352010/article/details/5174...

#coding=utf-8 from numpy import * import re def load_data(): """加载数据""" data=[];label=[] ...

qq_34352010 (http://blog.csdn.net/qq_34352010) 2016年06月23日 13:52 190

【机器学习】Logistic回归的梯度上升法 (http://blog.csdn.net/HerosOfEarth/article/det...

现实生活中有一种问题，输出值只有两种情况：yes or no.这类问题常见有：电子邮箱中的垃圾邮件分类(spam or not spam)，肿瘤为良性或者恶性等。在这些问题中，我们想预测的变量y，可以...

HerosOfEarth (http://blog.csdn.net/HerosOfEarth) 2016年07月21日 22:43 3321

梯度上升算法迭代过程和数学原理 (http://blog.csdn.net/berg_weald/article/details/701...

梯度上升算法在统计学习中使用广泛，比如Logistic 使用梯度上升优化每一个属性的对应的权值参数 即 Z = W0X0 + W1X1 + W2X2+ ..... +WnXn 中的w参数 ...

berg_weald (http://blog.csdn.net/berg_weald) 2017年04月12日 17:41 75

机器学习算法（优化）之一：梯度下降算法、随机梯度下降（应用于线性回归、Logistic回归...

本文介绍了机器学习中基本的优化算法—梯度下降算法和随机梯度下降算法，以及实际应用到线性回归、Logistic回归、矩阵分解推荐算法等ML中。...

qq_34531825 (http://blog.csdn.net/qq_34531825) 2016年09月01日 09:14 2412

Logistic回归与梯度下降法 (http://blog.csdn.net/ACdreamers/article/details/44657979)

Logistic回归为概率型非线性回归模型，是研究二分类观察结果与一些影响因素之间关系的一种 多变量分析方法。通常的问题是，研究某些因素条件下某个结果是否发生，比如医学中根据病人的一些症状来判断它是 ...

ACdreamers (http://blog.csdn.net/ACdreamers) 2015年03月26日 22:46 10014

梯度下降法和拟牛顿法（四）：在Logistic回归中的应用 (http://blog.csdn.net/z5718265/...

生活中我们有一个简单的经验：某一件事情发生的概率与另一些事情成正相关关系。比如说，一个开宝马车的人是百万富翁的概率是很大的。使用数学记号可记为： P(Y=1)∝wTXP(Y=1)\propto w^...

z5718265 (http://blog.csdn.net/z5718265) 2016年07月16日 17:30 262

Logistic回归与梯度下降法 (http://blog.csdn.net/haluoluo211/article/details/52029388)

http://blog.csdn.net/acdreamers/article/details/44657979 Logistic回归为概率型非线性回归模型，是研究二分类观察结果与一些影响因素之...

haluoluo211 (http://blog.csdn.net/haluoluo211) 2016年07月25日 23:34 268

Logistic回归原理【似然函数与梯度选择】 (http://blog.csdn.net/TaoTaoFu/article/detai...

0

Logistic回归为概率型非线性回归模型，是研究二分类观察结果与一些影响因素之间关系的一种 多变量分析方法。通常的问题是，研究某些因素条件下某个结果是否发生，比如医学中根据病人的一些症状来判断...

TaoTaoFu (http://blog.csdn.net/TaoTaoFu) 2016年10月16日 18:24 2226

logistic回归|梯度下降|牛顿法总结 (http://blog.csdn.net/u014664226/article/details/5...

1.logistic回归模型logistic回归是用线性模型解决分类问题的算法 考虑现在有一个样本集合，样本特征有两维，要用一条直线作为这两类的分界线，如下图所示 也就是说logistic算法就是...

u014664226 (http://blog.csdn.net/u014664226) 2016年06月16日 11:17 2044

坐标上升算法总结(文档c++代码) (http://download.csdn.net/detail/nu...

RAR

(http://downloa 2012年12月20日 13:57 1.08MB 下载

坐标上升算法（Coordinate Ascent）及C++编程实现 (http://blog.csdn.net/NUPTboyZ...

坐标上升算法（Coordinate Ascent）及C++编程实现 编程实现： #include using namespace std; #define f(x1,...

NUPTboyZHB (http://blog.csdn.net/NUPTboyZHB) 2012年12月20日 13:43 6640

理解Logistic回归算法原理与Python实现 (http://blog.csdn.net/chaipp0607/article/det...

一般的机器学习的实现大致都是这样的步骤： 1.准备数据，包括数据的收集，整理等等 2.定义一个学习模型（learning function model），也就是最后要用来去预测其他数据的那个模型 ...

chaipp0607 (http://blog.csdn.net/chaipp0607) 2017年05月19日 12:19 1178

机器学习--Logistic回归算法 (http://blog.csdn.net/zyl1042635242/article/details/454...

一、基本原理 假设有一些数据点，用一条直线对这些点进行拟合（该线称为最佳拟合直线），这个拟合过程就称作回归。 Sigmoid函数是一种阶跃函数，具体计算公式如下： ...

zyl1042635242 (http://blog.csdn.net/zyl1042635242) 2015年05月04日 21:04 477