逻辑回归(Logistic Regression)

songsong_H 关注
0.409 2019.02.10 17:59:33 字数 1,806 阅读 464

逻辑回归（Logistic Regression）是一种用于分类问题的算法。

建立一个Logistic Regression模型，需要以下三部分工作：

- 构造预测函数
- 构建损失函数
- 最小化损失函数，求解θ

1.构造预测函数

逻辑回归和线性回归都是广义的线性模型，而逻辑回归的本质就是由线性回归变化而来的。这里简单介绍下线性回归模型：

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n = \theta^T x (x_0 = 1)$$

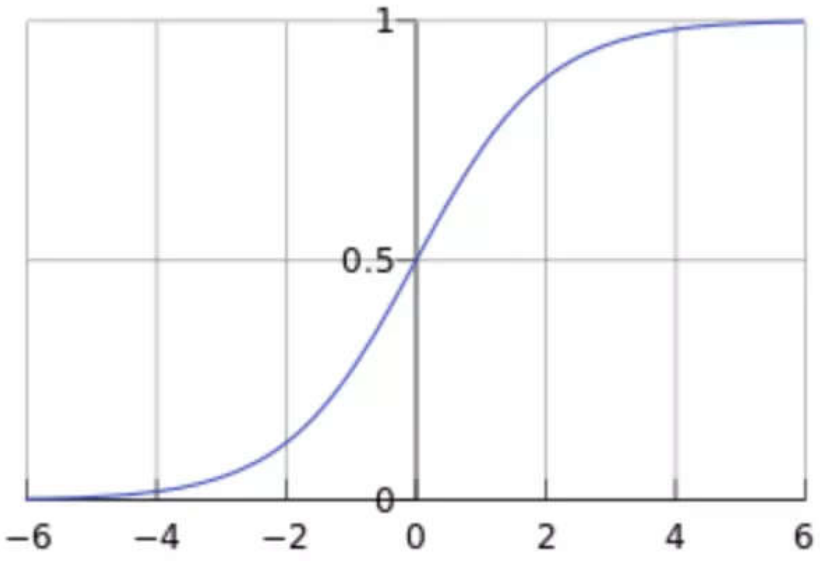
1-1.gif

θ统称为模型的参数，其中θ0为截距，θ1-θn是模型的系数。
线性回归的任务就是构造一个预测函数z来映射输入的特征矩阵x和标签y的线性关系，而构造预测函数的核心就是找出模型的参数θ，线性回归模型通常用最小二乘法来求解参数。

但是在二分类问题中，我们想要函数输出0或1，那么就需要用到联系函数（link function），将线性回归方程z变换为g(z)，并且使得g(z)的值分布在（0，1）之间，且当g(z)接近0时样本的标签为类别0，当g(z)接近1时样本的标签为类别1，这样就得到了一个分类模型。这个联系函数对于逻辑回归来说就是Sigmoid函数：

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

1-2.gif



songsong_H 关注
总资产3 (约0.38元)

AdaBoost算法介绍
阅读 357

梯度下降算法的Python实现
阅读 2,166

推荐阅读

聊天机器人-对话系统技术原理
阅读 228

Qlearning教你的机器人认错
阅读 68

深度神经网络的图像分类应用
阅读 411

开篇：风控评分卡知识总结
阅读 73

《特征工程入门与实践》学习笔记
阅读 264

> Sigmoid函数是一个S型的函数，当自变量z趋于正无穷时，因变量g(z)趋近于1，当自变量z趋于负无穷时，g(z)趋近于0，它可以将任意实数映射到(0,1)区间。

通过将线性回归模型与Sigmoid函数相结合，就可以得到逻辑回归模型的一般形式：

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

1-3.gif

其中，g(z)就是逻辑回归返回的标签值，此时y(x)的取值都在(0,1)之间。令y(x)除以1-y(x)就得到了形似几率odds(一个事件的odds指的是该事件发生的概率与不发生的概率之比)，在此基础上取对数可以得到：

$$\begin{aligned} ln\frac{y(x)}{1-y(x)} &= ln(\frac{\frac{1}{1+e^{-\theta^T x}}}{1-\frac{1}{1+e^{-\theta^T x}}}) \\ &= ln(\frac{\frac{1}{1+e^{-\theta^T x}}}{\frac{e^{-\theta^T x}}{1+e^{-\theta^T x}}}) \\ &= ln(\frac{1}{e^{-\theta^T x}}) \\ &= ln(e^{\theta^T x}) \\ &= \theta^T x \end{aligned}$$

1-4.gif

从公式1-4可看出g(z)的形似几率取对数的本质就是线性回归z，实际上我们是对线性回归模型的预测结果取对数几率来让其结果无限逼近0和1。因此，这个模型也被称为"对数几率回归"（logistic regression）,也就是逻辑回归。

2.构建损失函数

yθ(x)函数的值有特殊的含义，它表示预测结果为1的概率。因此对于输入项x，预测分类结果为1和0的概率分别为：

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

2-1.gif

公式2-1可以合并为：

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

2-2.gif

取似然函数为：

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{m} P(y^{(i)}|x^{(i)};\theta) \\ &= \prod_{i=1}^{m} (y_\theta(x^{(i)}))^{y^{(i)}}(1-y_\theta(x^{(i)}))^{1-y^{(i)}} \end{aligned}$$

2-3.gif

2-4.gif

最大似然函数就是求使*l(θ)*取最大值时的θ，这里可以通过梯度上升算法求解，得到的θ就是最佳参数。但是在Andrew Ng的课程中将损失函数*J(θ)*取为下式：

$$J(\theta) = -\frac{1}{m}l(\theta)$$

2-5.gif

因为乘了系数-1/m，所以使*J(θ)*取最小值的θ就为最佳参数。

3.最小化损失函数，求解θ

使用***梯度下降算法****(Gradient descent)*求解使得损失函数*J(θ)*取最小值的θ，由梯度下降算法可得θ的更新过程如下：

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta), (j = 0, 1, ..., n)$$

3-1.gif

其中j表示样本的第j个特征，α表示步长(也称为学习率)，后半部分对*J(θ)*的参数求偏导并以向量形式表示，就是损失函数的梯度。
对J(θ)求偏导：

$$\begin{aligned}
\frac{\partial}{\partial\theta_j}J(\theta) &= -\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}\frac{1}{y_{\theta(X^{(i)})}}\frac{\partial}{\partial\theta_j}y_\theta(X^{(i)}) - (1-y^{(i)})\frac{1}{1-y_{\theta(X^{(i)})}}\frac{\partial}{\partial\theta_j}y_\theta(X^{(i)})) \\
&= -\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}\frac{1}{g(\theta^T x^{(i)})} - (1-y^{(i)})\frac{1}{1-g(\theta^T x^{(i)})})\frac{\partial}{\partial\theta_j}g(\theta^T x^{(i)}) \\
&= -\frac{1}{m}\sum_{i=1}^{m}(y^{(i)}\frac{1}{g(\theta^T x^{(i)})} - (1-y^{(i)})\frac{1}{1-g(\theta^T x^{(i)})})g(\theta^T x^{(i)})(1-g(\theta^T x^{(i)}))\frac{\partial}{\partial\theta_j}\theta^T x^{(i)} \\
&= -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}(1-g(\theta^T x^{(i)})) - (1-y^{(i)})g(\theta^T x^{(i)})\right)x_j^{(i)} \\
&= -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - g(\theta^T x^{(i)})\right)x_j^{(i)} \\
&= \frac{1}{m}\sum_{i=1}^{m}\left(y_{\theta(X^{(i)})} - y^{(i)}\right)x_j^{(i)}
\end{aligned}$$

3-2.gif

其中

$$\begin{aligned}
\frac{\partial}{\partial\theta_j}g(\theta^T x^{(i)}) &= \frac{0 - \frac{\partial}{\partial\theta_j}(1+e^{-\theta^T x^{(i)}})}{(1+e^{-\theta^T x^{(i)}})^2} \\
&= \frac{e^{-\theta^T x^{(i)}}\frac{\partial}{\partial\theta_j}(\theta^T x^{(i)})}{(1+e^{-\theta^T x^{(i)}})^2} \\
&= (\frac{1}{1+e^{-\theta^T x^{(i)}}}(1 - \frac{1}{1+e^{-\theta^T x^{(i)}}}))\frac{\partial}{\partial\theta_j}(\theta^T x^{(i)}) \\
&= g(\theta^T x^{(i)})(1-g(\theta^T x^{(i)}))\frac{\partial}{\partial\theta_j}(\theta^T x^{(i)})
\end{aligned}$$

3-3.gif

3-4.gif

对该公式进行迭代，直到收敛（J(θ)在每一步的迭代中都会减小，如果某一步J(θ)减小的值小于一个很小的值ε(例如0.001)，则判定收敛）或者达到某个停止条件为止（比如达到指定的迭代次数）

3.1 梯度下降算法的实例

3.1.1 单变量函数的梯度下降

假设单变量函数J(θ)=θ^2，J'(θ)=2θ即函数的梯度，初始化起点为θ0=1，步长α为0.4，那么根据梯度下降的迭代公式进行计算可得到：

θ0=1

θ1=θ0-α*J'(θ0)=1-0.4*2=0.2

θ2=θ1-α*J'(θ1)=0.2-0.4*0.4=0.04

θ3=θ2-α*J'(θ2)=0.04-0.4*0.08=0.008

θ4=θ3-α*J'(θ3)=0.008-0.4*0.016=0.0016

...

我们发现进行了四次迭代，已经接近函数的最小值点0了。

3.1.2 多变量函数的梯度下降

假设目标函数为J(θ)= θ1^2 + θ2^2，通过梯度下降算法计算该函数的最小值。函数的梯度为<2θ1,2θ2>，假设初始起点为(1,3)，初始化步长为0.1，进行迭代：

θ0=(1,3)

θ1=(1,3)-0.1*(2,6)=(0.8,2.4)*

θ2=(0.8,2.4)-0.1(0.16,4.8)=(0.64,1.92)

θ3=(0.64,1.92)-0.1*(1.28,3.84)=(0.512,1.536)

...

随着迭代的不断进行，已经越来越接近该函数的最小值点(0,0)了。

3.2 批量梯度下降法(Batch Gradient Descent)

批量梯度下降算法是梯度下降算法的最常用形式，即在更新参数时利用所有的样本来进行更新，上面公式3-4就是逻辑回归的批量梯度下降算法，由于1/m是一个常量，α也是一个常量，所以可省略1/m，故参数的更新过程可写成：

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} \left(y_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}, (j = 0, 1, ..., n)$$

BGD.gif

由于有m个样本，所以求梯度的时候就用了所有m个样本的梯度数据。

此外，梯度下降算法还包括随机梯度下降法(Stochastic Gradient Descent)、小批量梯度下降法(Mini-batch Gradient Descent)，这里就不展开了，下次再比较一下这几个算法的区别吧。

4.逻辑回归的代码实现

接下来用python实现一个简单的逻辑回归算法。

首先加载数据：

```
def loaddata(filename):
    dataSet = pd.read_table(filename, header=None)
```

```
    return data_x,data_y
```

构造预测函数和损失函数，最后利用批量梯度下降法求参数θ：

```
#sigmoid函数
def sigmoid(y):
    s = 1.0/(1.0+np.exp(-y))
    return s

def cost(xMat,weights,yMat):
    m, n = xMat.shape
    hypothesis = sigmoid(np.dot(xMat, weights))  # 预测值
    cost = (-1.0 / m) * np.sum(yMat.T * np.log(hypothesis) + (1 - yMat).T * np.log(1 - hypothesis
    return cost

def BGD_LR(data_x,data_y,alpha=0.1,maxepochs=10000,epsilon=0.0001):
    xMat = np.mat(data_x)
    yMat = np.mat(data_y)
    m,n = xMat.shape
    weights = np.ones((n,1)) #初始化模型参数
    epochs_count = 0
    while epochs_count < maxepochs:
        loss = cost(xMat,weights,yMat)
        hypothesis = sigmoid(np.dot(xMat,weights))
        error = hypothesis -yMat #预测值与实际值误差
        print(loss)
        grad = (1.0/m)*np.dot(xMat.T,error) #损失函数的梯度
        last_weights = weights #上一轮迭代的参数
        weights = weights - alpha*grad #参数更新
        if (abs(cost(xMat, weights, yMat) - cost(xMat,last_weights,yMat))) < epsilon: #终止条件
            break
        epochs_count += 1
    print('迭代到第{}次，结束迭代！'.format(epochs_count))
    return weights
```

计算模型预测准确率：

```
def acc(weights, test_x, test_y):
    xMat_test = np.mat(test_x)
    m, n = xMat_test.shape
    result = []
    for i in range(m):
        proba = sigmoid(np.dot(xMat_test[i,:], weights))
        if proba < 0.5:
            preict =  0
        else:
            preict = 1
        result.append(preict)
    test_x_ = test_x.copy()
    test_x_['predict'] = result
    acc = (test_x_['predict']==test_y['label']).mean()
    return acc
```

result.png

当设置步长为0.1，损失精度为0.0001时，最终当迭代至第571次时，损失精度已经小于0.0001了，训练集的准确率达到了0.96。至此，一个简单的逻辑回归模型就完成了~

本文代码地址：https://github.com/DaHuangTyro/Daily_Machine_Learning

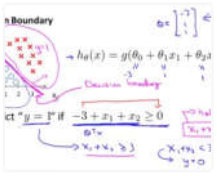
参考

https://blog.csdn.net/xiaoxiangzi222/article/details/55097570
https://www.jianshu.com/p/c7e642877b0e

逻辑回归（Logistic Regression）

逻辑回归的定义 简单来说， 逻辑回归（Logistic Regression）是一种用于解决二分类（0 or 1）...

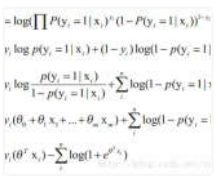
李亚鑫 阅读 3,556 评论 3 赞 16

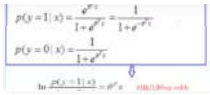


Python实践之逻辑回归（Logistic Regression）

机器学习算法与Python实践这个系列主要是参考《机器学习实战》这本书。因为自己想学习Python，然后也想对一些...

MapleLeaff 阅读 1,306 评论 0 赞 6

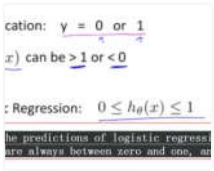




逻辑回归（logistic regression）

1、 2、逻辑回归我们要求h(theta)的值在0和1之间，g（z）是s型函数或者叫逻辑函数，令z = theta...

mugtmag 阅读 395 评论 0 赞 0



逻辑回归（Logistic regression）

参考1参考2 0.前言 Logistic regression （逻辑回归）是当前业界比较常用的机器学习方法，用于...

听风1996 阅读 312 评论 0 赞 6