哈爾濱工業大學

概率论与数理统计课程论文

题目：线性回归与梯度下降算法

班　号	**2103602**
学　号	**2021110683**
姓　名	**徐柯炎**
日　期	**2021.11.1**
成　绩	

摘　要

本文介绍了线性回归问题及其求解方法：最小二乘法、正规方程法以及梯度下降算法，并着重介绍了梯度下降算法，将其与其他=两种方法进行了比较，阐述了其在计算机编程中的优势。之后还介绍了三种常用的梯度下降算法，以及其在机器学习中的具体应用以及优化过程。

关键词：线性回归；梯度下降；逻辑回归；正则化；

正文部分

1. 准备知识

1.1 回归分析

在统计学中，回归分析（regression analysis)指的是确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法。回归分析按照涉及的变量的多少，分为一元回归和多元回归分析；按照因变量的多少，可分为简单回归分析和多重回归分析；按照自变量和因变量之间的关系类型，可分为线性回归分析和非线性回归分析。[1]

1.2 线性回归

以一元线性回归为例，假定回归模型为

$$f(x) = ax + b$$

那么线性回归的目标就是如何让 f(x)和 y 之间的差异最小，换句话说就是 a，b 取什么值的时候 f(x)和 y 最接近。

这里我们得先解决另一个问题，就是如何衡量 f(x)和 y 之间的差异。在回归问题中，均方误差是回归任务中最常用的性能度量。记 J(a,b)为 f(x)和 y 之间的差异，即

$$\mathrm{J}(a,b) = \sum_{i=1}^{\mathrm{n}} (f(x^{(i)}) - y^{(i)})^2 = \sum_{i=1}^{\mathrm{n}} (ax^{(i)} + b - y^{(i)})^2$$

i 表示数据的第 i 组，J(a, b)为损失函数（Loss Function），损失函数极小化，意味着拟合程度最好，对应的模型参数即为最优参数，而代价函数（Cost Function）度量全部样本集的平均误差，本质上二者是一样的。

下文将具体介绍线性回归的参数求解问题。

2. 线性回归问题的一般求解

2.1 最小二乘法

因为损失函数为二次函数，即凸函数，则存在一个最小值，使得对应模型参数为最优参数。此时，损失函数关于各参数的偏导为 0，即

$$\frac{\partial J(a,b)}{\partial a} = 2\sum_{i=1}^{n} x^{(i)}\left(ax^{(i)} + b - y^{(i)}\right) = 0$$

$$\frac{\partial J(a,b)}{\partial b} = 2\sum_{i=1}^{n}\left(ax^{(i)} + b - y^{(i)}\right) = 0$$

求解此方程组即可得到 a，b 的值。[1]

2.2 正规方程法

正规方程一般用在多元线性回归中，所以这里不再用一元线性回归举例子了。

同样，假设有 n 组数据，其中目标值（因变量）与特征值（自变量）之间的关系为：

$$y = \sum_{i=0}^{n}\theta_i x_i$$

其中

$$x_0 = \begin{pmatrix}1\\ \dots \\ 1\end{pmatrix}, \quad x_i = \begin{pmatrix}x_i^{(1)}\\ \dots \\ x_i^{(n)}\end{pmatrix} \ (i = 1,2,\dots,n), \quad y = \begin{pmatrix}y^{(1)}\\ \dots \\ y^{(n)}\end{pmatrix}$$

也就是全为 1 的列向量，θ_i为第 i 个参数，于是损失函数为

$$J(\theta) = \sum_{i=1}^{n}\left(f\left(x^{((i))}\right) - y^{((i))}\right)^2 = (X\theta - y)^T(X\theta - y)$$

令损失函数对θ的导数为 0，得

$$\theta = (X^TX)^{-1}X^Ty$$

到此，就求出了所有系数θ。不过正规方程需要注意的是在实际中可能会出现是奇异矩阵，往往是因为特征值之间不独立。这时候需要对特征值进行筛选，剔除那些存在线性关系的特征值。[2]

3. 梯度下降算法

3.1 算法实现

以下将着重介绍梯度下降算法。

梯度下降是一个用来求函数最小值的算法，梯度下降背后的思想是：开始时随机选择一个参数的组合$(\theta 0,\ \theta 1,\dots,\ \theta n)$，计算代价函数，然后寻找下一个能让代价函数值下降最多的参数组合。持续执行此操作直到到一个局部最小值（local minimum），因为没有尝试完所有的参数组合，所以不能确定我们得到的局部最小值是否便是全局最小值（global minimum），选择不同的初始参数组合，

可能会找到不同的局部最小值。

批量梯度下降（batch gradient descent）算法的公式为：

$$\text{repeat until convergence } \{$$
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{for } j = 0 \text{ and } j = 1)$$
$$\}$$

其中a是学习率（learning rate），在批量梯度下降中，我们每一次都同时让所有的参数减去学习速率乘以代价函数的导数。

3.2 梯度下降算法的优势

梯度下降法是通用的，包括更为复杂的逻辑回归算法中也可以使用，并且对于多维变量而言，梯度下降算法的速度更快，使用随机梯度下降时更容易跳出局部最优解，有利于寻找全局最优解；

相较于正规方程而言，正规方程的速度往往更快，但是当数量级达到一定的时候，还是梯度下降法更快，因为正规方程中需要对矩阵求逆，而求逆的时间复杂的是 n 的 3 次方，时间复杂度高，速度慢；

最小二乘法一般比较少用，虽然它的思想比较简单，在计算过程中需要对损失函数求导并令其为 0，从而解出系数 θ 。但是对于计算机来说很难实现，所以一般不使用最小二乘法。

综上所述，梯度下降算法更加适合计算机，对于计算机来说更容易处理，运行速度更快。

3.3 梯度下降算法的分类及优缺点

（1）批量梯度下降（BGD）：

$$\theta_0 := \theta_0 - a\frac{1}{m}\sum_{i=1}^{m}（h_\theta（x^{(i)}）- y（i））$$
$$\theta_1 := \theta_1 - a\frac{1}{m}\sum_{i=1}^{m}（h_\theta（x^{(i)}）- y（i））x^{(i)}$$

在梯度下降的每一步中，我们都用到了所有的训练样本，我们需要进行求和运算，在梯度下降中，在计算微积分时，每一个样本都需要计算，会导致运算速度比较慢。

（2）随机梯度下降法（SGD）

$$\theta_i := \theta_i - \alpha((h_\theta(x^{(i)} - y^{(i)}) * x^{(i)}))$$

随机梯度下降法和批量梯度下降法是两个极端，批量梯度下降每次采用所有

数据下降，随机梯度下降每次用一个样本来梯度下降。

对于训练速度来说，随机梯度下降法由于每次仅仅采用一个样本来迭代，训练速度很快。

对于精准度来说，随机梯度下降法每次训练仅仅用一个样本决定梯度的方向，可能得到局部最小值，精准度不高。

对于收敛速度来说，由于随机梯度下降法一次迭代一个样本，导致迭代方向变化很大，不能很快的收敛到局部最优解。

（3）小批量梯度下降（MBGD）

$$\theta_1 := \theta_1 - \alpha \frac{1}{t} \sum_{i=1}^{t} x((h_\theta(x^{(i)} - y^{(i)}) * x^{(i)}))$$

小批量梯度下降每次迭代使用一个以上又不是全部的样本

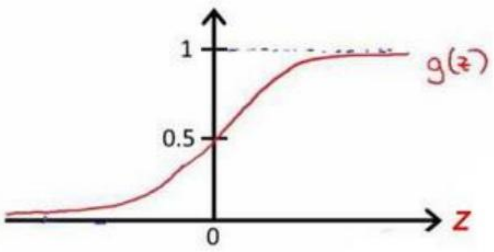
MBGD 是集中了 SGD 和 BGD 的优点。使用多个样本相比 SGD 提高了梯度估计的精准度，小批量的估计。

缺点是同 SGD 一样，每次梯度的方向不确定，可能需要很长时间接近最小值点，不会收敛通常在使用 MBGD 之前先将数据随机打乱，然后划分 Mini-batch，所以 MBGD 有时也称 SGD。

4. 梯度下降算法在机器学习中的应用

4.1 分类问题

分类问题常用逻辑回归模型，常用 sigmoid 函数来进行分类，该函数的图像为：



其取值永远在 0 和 1 之间。其函数为

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

对于线性回归模型，笔者定义的代价函数是所有模型误差的平方和。理论上来说，笔者也可以对逻辑回归模型沿用这个定义，但是问题在于，当$h_\theta(x)$带入到这样定义了的代价函数中时，得到的代价函数将是一个非凸函数。这意味着代

价函数有许多局部最小值，这将影响梯度下降算法寻找全局最小值。因此，笔者将重新定义逻辑回归的代价函数为：

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

这样构建的$Cost(h\theta\ (x),\ y)$函数的特点是：当实际的 $y\ =1$ 且 $h_\theta(x)$ 也为 1 时误差为 0，当 $y\ =1$ 但 $h_\theta(x)$ 不为 1 时误差随着 $h_\theta(x)$ 变小而变大；当实际的 $y\ =0$ 且 $h_\theta(x)$ 也为 0 时代价为 0，当 $y\ =0$ 但 $h_\theta(x)$ 不为 0 时误差随着 $h_\theta(x)$的变大而变大。在得到这样一个代价函数以后，便可以用梯度下降算法来求得能使代价函数最小的参数了。此算法具体为：

Repeat { $\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(\mathrm{x}^{(i)}\right) - \mathrm{y}^{(i)}\right)\mathrm{x}_j^{(i)}$

(simultaneously update all θ_j **)**

}

可以看到得到的梯度下降算法表面上看上去与线性回归的梯度下降算法一样，但是这里的$h_\theta(x) =\ g(\theta^T X)$与线性回归中不同，所以实际上是不一样的。

4.2 正则化

线性回归和逻辑回归能够有效地解决许多问题，但是当将它们应用到某些特定的机器学习应用时，会遇到过拟合(over-fitting)的问题，可能会导致它们效果很差。在这里，笔者将介绍一种解决过拟合的方法——正则化技术，即保留所有的特征，但是减少参数的大小。

因为过拟合的产生是高次项导致的，所以正则化要做的事就是尽量让高次项的系数接近于 0，但由于不知道具体是哪一项高次项导致的过拟合，于是正则化就让所有参数θ在一定程度上减小，形象的说，就是给所有参数设置一个“惩罚”。此时，经过正则化后的线性回归方程为：

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

其中λ又称为正则化参数（Regularization Parameter）。

如果选择的正则化参数 λ 过大，则会把所有的参数都最小化了，导致模型变成 $h_\theta(x) =\ \theta_0$，也就是上图中红色直线所示的情况，造成欠拟合。

但若 λ 的值太大了，那么θ（不包括θ_0）都会趋近于 0，这样得到的只能是一条平行于x轴的直线。

所以对于正则化，λ 要取一个合理的值，这样才能更好的应用正则化。回顾一下代价函数，为了使用正则化，让我们把这些概念应用到线性回归和逻辑回归中去，那么我们就可以让他们避免过度拟合了。[3]

参考文献:

[1].陈希孺, 概率论与数理统计. 2009: 中国科学技术大学出版社.

[2].周志华, 机器学习. 2016: 清华大学出版社.

[3]. Andrew Ng，斯坦福大学 2014 机器学习教程