机器学习算法原理

李广创

目 录

[目 录 II](#_Toc10136750)

[第一章 线性回归与逻辑回归 4](#_Toc10136751)

[1.1 线性回归 4](#_Toc10136752)

[1.1.1 模型的公式 4](#_Toc10136753)

[1.1.2 参数求解 4](#_Toc10136754)

[1.1.3 线性回归的正则化 5](#_Toc10136755)

[1.2 二分类逻辑回归 5](#_Toc10136756)

[1.2.1 模型公式 6](#_Toc10136757)

[1.2.2 损失函数 6](#_Toc10136758)

[1.2.3 参数求解 7](#_Toc10136759)

[1.3 多元逻辑回归 7](#_Toc10136760)

[1.4 通用的二分类扩展到多分类 8](#_Toc10136761)

[1.4.1 OVO 8](#_Toc10136762)

[1.4.2 OVR 8](#_Toc10136763)

[1.5 附录 9](#_Toc10136764)

[1.5.1 逻辑回归中，梯度下降法的具体推导 9](#_Toc10136765)

[1.5.2 L1，L2正则化的区别（常问的题目） 9](#_Toc10136766)

[1.5.3 梯度下降法： 11](#_Toc10136767)

[1.5.4 牛顿法： 12](#_Toc10136768)

[2) 牛顿法的解释 13](#_Toc10136769)

[用一维数据比较好解释，对于一维的数据，其迭代公式为：C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps99.png； 13](#_Toc10136770)

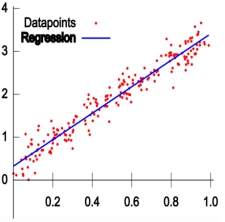
[求极小值点，也就是求时f’(x) = 0，x的取值，因此可以画出f’(x)的图像，初始化的时候x=x0，在x0处做切线，与x轴的交点，得到x1，继续x1点处做切线，得到x2......，这样迭代下去，直到f(xn) = 0 13](#_Toc10136771)

1. 线性回归与逻辑回归
   1. 线性回归

线性回归的思想是：线性回归的模型假设非常简单直观，它想用一条直线或一个超平面来拟合所有的样本点。因此，属于线性模型。（当然也可以理解为每个特征的线性组合）

* + 1. 模型的公式

其中表示参数向量；表示特征向量。



* + 1. 参数求解

确定了需要优化的目标函数以后，模型的训练过程就变成纯粹的函数优化问题，我们希望调节参数，使得上述目标函数的值最小，也即总体样本的损失之和最小；因此可以运用梯度下降法，牛顿法，拟牛顿法等来求解；下面只介绍梯度下降法求解参数的过程：

1. 随机初始化所有参数，。
2. 对于每个参数, 执行下面的公式，每一轮迭代将所有的都进行更新。
3. 如果每个参数的更新值都小于，则算法结束，否则继续执行2）
   * 1. 线性回归的正则化

在优化损失函数的时候，为了防止模型过分拟合，可以在损失函数后面加上正则化项；常见的正则化项有L1、L2正则化。

1. L1正则化为：

带L1正则化的线性回归也称为Lasso回归，Lasso回归除了能减缓过拟合外，还具有特征选择的功能，因为它在做优化的时候，很容易使某些不重要特征的权重为0。

1. L2正则化为：

带L2正则化的线性回归也称为岭回归。

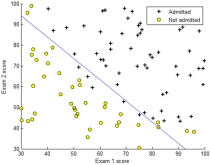
正则化项还可以用于很多以损失函数为优化目标的模型，如逻辑回归，神经网络，xgb等，以后不再赘述。

* 1. 二分类逻辑回归

**逻辑回归的思想是**：为每个特征赋予一个权重，然后线性组合，并经过一个sigmoid函数映射成概率。

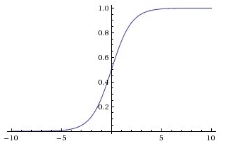
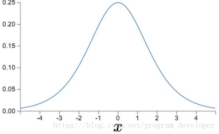
表示预测为正类；表示表示预测为负类；，也即 为分类超平面（因此，本质上还是用一个超平面来对样本点做切分）

* + 1. 模型公式



将概率合写为一个式子，表示为：

Sigmoid函数表示为：；其导数为：

的图像 的图像

* + 1. 损失函数

逻辑回归采用的损失函数为对数似然损失，这个损失函数并不像线性回归采用的RMSE那样直观明了，他是通过“极大似然估计”的思想推导而来的。

极大似然估计是一个非常重要的思想，后面会看到它是很多优化问题的基础；其思想是：假设样本是独立同分布的，也即样本与样本之间没有关联；我们希望找到一个参数，使得出现当前结果的可能性最大；

具体到机器学习问题中，我们希望**每个样本都尽可能地预测到它原来的标签**；也即通过调节参数，使得最大，（注意，是样本原来的标签，式子表示所有样本都预测到它原来标签的概率）。

因此，定义需要**最大化**的目标函数为：

因为后面需要对目标函数求偏导，而连乘的形式求偏导很复杂；为了求导方便，方程两边同时取log，将连乘的形式转化为求和，此时对整个目标函数求导等于对每一项分别求导再求和；也即

因此，对数目标函数为：

为了直观，我们定义了损失函数的概念，将上述对数目标函数取反便是损失函数：

之后，**最小化**损失函数，便可求得参数，也即。

* + 1. 参数求解

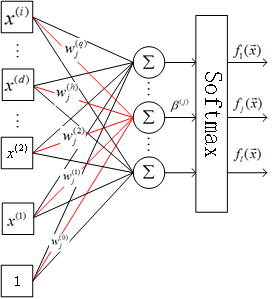
依然用梯度下降法求解参数：

C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps49.png

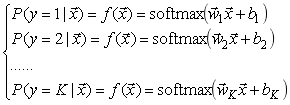
#推导过程见附录

* 1. 多元逻辑回归

多元逻辑回归其实就是一层神经网络，然后用softmax函数作为激活函数，处理多分类问题：



模型的公式为：



模型的损失函数采用交叉熵损失，具体的参数求解过程与神经网络一样，在后面有详细的推导，这里不再赘述。

* 1. 通用的二分类扩展到多分类
     1. OVO

假设有K个类别，对所有的数据进行两两组合取2个类别的数据训练一个分类器，每个组合训练一个分类器，最终会有K(K-1)/2个分类器；

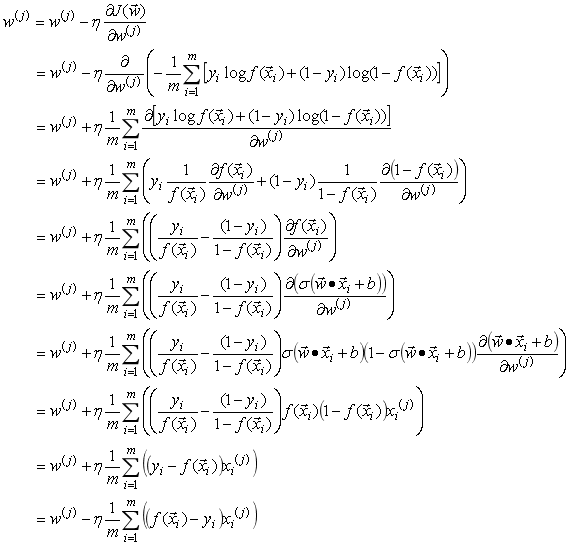
预测时，可以用投票法，也即取预测得最多的类别作为输出；也可以用概率法，也即将预测为每个类别的概率加起来，取概率最大的类别作为输出。

* + 1. OVR

假设有K个类别，取1个类别为正类，其他类别为负类，构建一个分类器；遍历所有的类别，最终构建K个分类器。

预测时，采用概率法，也即取预测概率最大的类别作为输出。

* 1. 附录
     1. 逻辑回归中，梯度下降法的具体推导



* + 1. L1，L2正则化的区别（常问的题目）
* 对应线性回归来说，使用L1正则化称为Lasso回归，使用L2正则化称为岭回归。

Lssso回归：C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps53.png

岭回归：C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps54.png

* 相同点：

毫无疑问，L1，L2正则化都可以用来降低过拟合的风险，

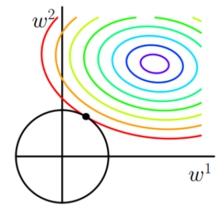
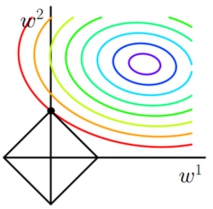
* 不同点：

L2正则化会让模型所有的参数都变小，但一般不会变为0，以此构造一个更简单的模型；

L1正则化很容易获得“稀疏解”，也即w中很多项为0；相当于将不重要的特征的权重置为0，实现了特征选择。

* L1、L2产生这种解的原因解释：

假设特征C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps55.png是二维的，那么对应的参数C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps56.png也是二维的，即C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps57.png和C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps58.png；以C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps59.png和C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps60.png为坐标轴，C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps61.png的等值线是图中的彩色圆，C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps62.png的等值线是黑色的圆，C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps63.png的等值线是一个菱形；整个损失函数要在“平方误差项”与“正则化项”之间折中，即C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps64.png的取值要出现在两个等值线的交点处；从坐标轴可以看出，L1正则化的解很容易出现在坐标轴处，即C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps65.png或C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps66.png为0；而L2正则化的解一般出现在某个象限中，也即C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps67.png和C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps68.png不为0；因此L1正则化具有“稀疏性”

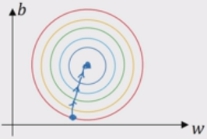
* + 1. 梯度下降法：

1. **梯度：**对多元函数求偏导，并以向量的形式展示出来（注意，多元函数的梯度是向量）；也即：C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps71.png；在梯度的方向上，函数增加得最快，在负梯度的方向上，函数减少得最快。
2. **梯度下降法的描述**：
3. 随机初始化所有参数C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps72.png
4. 对于每个参数：C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps73.png；

注：每次的所有的参数要一起更新，这样才能表示C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps74.png往负梯度的方向更新

1. 重复2），直到所有参数的更新都小于C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps75.png；输出最终的参数。
2. **梯度下降法的解释**：

对于凸函数C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps76.png，C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps77.png往“负梯度”的方向更新，会使函数值C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps78.png下降得最快，因此GD思想为：对于参数C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps79.png，每次往“负梯度”的方向走一小步，直到C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps80.png到达极小值，此时梯度C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps81.png。GD的更新公式为：C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps82.png，其中C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps83.png表示负梯度，C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps84.png表示每次更新的步长。



1. **梯度下降法家族**：

标准梯度下降（GD）：使用全部样本更新参数

随机梯度下降（SGD）：随机使用一个样本更新参数

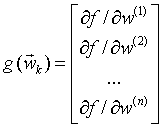
小批量梯度下降（mini\_GD）：使用一小部分样本更新参数

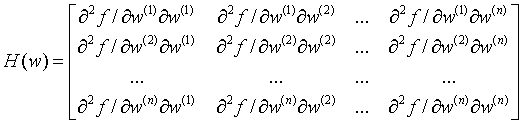
* + 1. 牛顿法：

1. **顿法的推导过程：**
2. 求函数C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps86.png的“极小值点”，可以转化为求C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps87.png的解
3. 令C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps88.png，对C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps89.png进行一阶泰勒展开：C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps90.png
4. C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps91.png，也即：C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps92.png

求得：C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps93.png；写成迭代形式：C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps94.png

其中C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps95.png是梯度向量，C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps96.png是海森矩阵的逆。





1. 牛顿法的解释

用一维数据比较好解释，对于一维的数据，其迭代公式为：C:\Users\李广创\AppData\Local\Temp\ksohtml19992\wps99.png；

求极小值点，也就是求时f’(x) = 0，x的取值，因此可以画出f’(x)的图像，初始化的时候x=x0，在x0处做切线，与x轴的交点，得到x1，继续x1点处做切线，得到x2......，这样迭代下去，直到f(xn) = 0

