**再谈线性回归与梯度下降**

线性回归、Logistic回归、Softmax回归、梯度下降、随机梯度下降、批量梯度下降可以说在机器学习领域是基础中的基础。笔者根据网上资料，如[UFLDL](http://ufldl.stanford.edu/wiki/index.php/UFLDL%E6%95%99%E7%A8%8B)等，进行了统一的整理，希望笔者的总结能够帮助大家更好得理解这些基础概念。

在总结之前，对于这三个回归，应该都有所认知。我们都知道，统计机器学习包括三个部分：模型、策略以及算法。模型表示的是所要学习的条件概率分布或者决策函数，模型的假设空间包含所有可能的决策函数。我们的目的就是从模型的假设空间中选择最优的一个作为我们的决策函数。那么怎么选择呢，用什么评价标准来选择呢，这就涉及到了第二个要素策略，策略就是我们依照什么样的规则来从假设空间中选择最优的一个决策函数。有了选择模型的方法也即有了这个策略了，接下来的问题就是如何执行这个策略了，这就是第三个要素算法。那对于我们的回归问题或者分类问题来讲，我们可以选择的模型有很多，我们会选择代价函数最小化的评价标准来将其作为我们的策略，最后会使用梯度下降的算法来执行我们的策略。因此，在这些基础问题中，我们首先需要搞清楚，对于每种模型，我们的代价函数是什么，为什么要选择它们作为代价函数。在接下来的叙述中会一一解答。

1. **线性回归**

线性回归的代价函数是多少呢？如何推导呢？在[Andrew Ng的公开课](http://openclassroom.stanford.edu/MainFolder/CoursePage.php?course=MachineLearning)中，Andrew使用了房价预测的例子来说明什么是线性回归。线性回归比较通用的写法是：

需要记住它是一个连续的模型，而不像分类模型是离散的。那么根据样本怎么去拟合参数呢？我们的策略就是通过最小化代价函数来获取最优的值。代价函数如何定义？这很简单，只需要利用最小二乘法来最小化误差即可。因此，一个样本的代价函数可以写成如下形式：

前面加1/2的原因是在求导的时候可以和2想抵消。因此，所有样本的平均代价函数便为：

注意，这里有的地方是不加m的，最小化的是总的代价，而不是平均代价。因为m是一个常数，所以加不加m对最后的结果是没有影响的。在这篇博文中，我们统一求平均代价。在编程实现上也把m加上。公式中，上标i表示样本的index。这样，我们需要梯度下降这个算法，来最小化。

什么是梯度？梯度可以理解为代价函数的导数，沿着这个方向每次移动一小步，最终肯定可以找到函数的极值点。这和高中学习的通过导数求解函数的极值是类似的。始终要明白的是，我们的参数是向量，即，我们的目标是求解当向量值为多少的时候，代价函数可以取得最小值。我们利用梯度下降是这样做的：对于中的每个参数，求出关于的偏导数（梯度），然后朝着该方向移动一个步长（通常是0.1，0.05，0.01），向量更新完一次为一次iteration。当的变化低于设定的阈值或者iteration达到一定次数的时候，便停止迭代，完成梯度下降的过程，而此时的便认为是拟合程度最好的。那么在此过程中，关于的偏导数是多少呢？我们直接根据求导公式得到如下：

这便是代价函数在处的偏导数，即在的梯度。那这个玩意儿到底是啥呢？其实表示的就是第i个样本的模型输出值和实际标注值得误差。其实，代价函数关于的梯度就是误差矩阵error的转置与样本矩阵x的第j列的乘积，再对m求平均。误差矩阵error的转置是1\*m的矩阵，而样本矩阵x的第j列是m\*1的矩阵，最后相乘便得到一个数值，再除以m便为所求的梯度。该过程的伪代码如下所示：

|  |
| --- |
| for i in MAX\_CYCLE:  output = //x和output均为矩阵  error = y – output //求出误差矩阵  = – α\*xT\*error/m //α为步长，xT\*error可以直接把每个的偏导求出 |

这里之所以使用xT\*error，而不是像上述那样反过来，是因为我们这样计算可以直接得出所有的偏导，这样便可以直接把向量的梯度求出来，这样便直接完成了一次完整的梯度下降了。所以我们可以看出，梯度下降的实现很简单，核心语句就一句代码。看下面Python的代码实现：

|  |
| --- |
| def gradAscent(dataArray, labelArray, alpha, maxCycles):  dataMat=mat(dataArray) #size:m\*n  labelMat=mat(labelArray) #size:m\*1  m,n=shape(dataMat)  weigh=ones((n,1))  for i in range(maxCycles):  h= dataMat\*weigh  error=labelMat-h weigh=weigh+alpha\*dataMat.transpose()\*error/m  return weigh |

关于求的最优解，还有其他的方法。比如使用normal equation的方法，可以直接求出最优解的解析解。推导过程不在这里列出，最后的结果是：

一般来讲，normal equation的方法使用起来简洁、方便，并且不需要feature scaling（即我们常说的特征缩放，可以理解为归一化。因为有些特征的值所处的范围比较大，这样一些数值结果就会被这些特征值所主导）。一般来讲，当特征数量小于100000时，可以直接使用normal equation的方法求解最优值；当特征数量大于100000时，就应当使用梯度下降的方法求解。

另一个比较重要的内容便是正则化（Regularization）。如果看过周志华的《机器学习》那本书，应该知道结构风险最小化这个概念。而最小二乘法表示的样本和实际标注值之间的误差平方和，只是表示的是经验风险。如果仅仅使经验风险最小化，而不考虑正则项，就会造成过拟合现象。正则化的提出是为了解决过拟合的问题。什么是过拟合呢？当的时候，这个时候模型对训练数据的拟合程度达到了几乎完美的程度，但是却对其他测试数据的generalization的能力不够，使得模型产生了很大的误差，这便是过拟合。一般有两种手段可以防止过拟合的发生：1、人工减少特征数量；2、加入正则项，使某些系数的估计非常小甚至为0，从而近似地减少特征数量。正则化的方法主要有两种：L2范数（岭回归）和L1范数（Lasso回归）。我们知道，L1范数可以理解为绝对值，而L2范数可以理解为平方和。那加入了正则项之后，代价函数变成了什么样子呢？

对于岭回归（Ridge Regression），代价函数便为：

这个时候如果用Normal Equation的方法，则的解析解为：

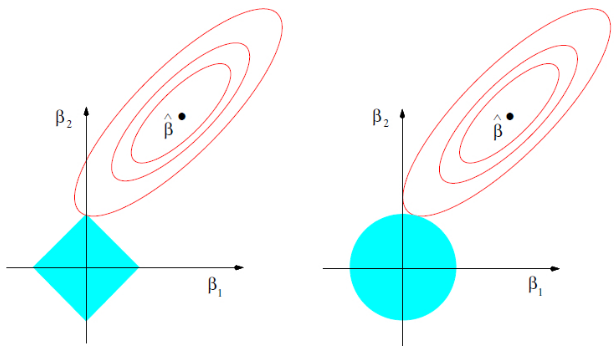
对于Lasso回归（Lasso Regression），代价函数便为：

但是，加入惩罚因子为什么可以使得参数值降低甚至为0呢？这个问题可以使用拉格朗日乘子来解释。比如拿L2范数作为例子，此时的问题可以转为求解最小的，其中：

并且

收敛示意图如下所示，左边是Ridge回归，右边是lasso回归。黑点表示的是最小二乘法代价函数的收敛中心，蓝色区域是加了乘法项的约束，其交点就是用相应regularization得到的系数在系数空间的表示。想一下，为什么交点一定是正则化之后的最优解？其实不管收敛中心在哪里（约束范围之外），交点一定是正则化之后的最优解。这个时候就会发现绝大多数的系数都很小甚至为0，而且L1范数更有效果，更加容易使得系数减小或者为0。

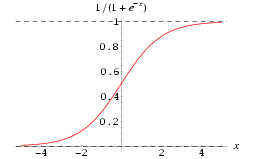
其实还有0范数，关于正则化的历史，可以参考知乎上一位知友的解答：[解答地址](https://www.zhihu.com/question/20924039)。总之，线性回归和梯度下降是基础中的基础，在神经网络的反向传播训练过程中，也需要使用梯度下降来求解最优的网络权重系数。因此，这一部分要深刻地理解，再怎么深刻也不为过。关于正则项，读者可以阅读其他方面的资料。



1. **逻辑斯蒂回归**

说到Logistic回归，大家首先应该明白的是一个函数，那就是Sigmoid函数。Sigmoid函数的形式为：

这个函数的图像如下所示：



从图中我们可以看出，Sigmoid函数是一个很好的阈值函数。连续、光滑、严格单调，并且关于(0,0.5)呈中心对称。另外还有一个很重要的性质，那就是该函数的导数十分容易求解。这为之后使用梯度下降求解最优解打下了坚实的基础，使得该算法可以很容易的实现和完成。Sigmoid函数g(x)的导数如下所示：

现在回到最初的问题上来。对于分类问题，如何构造代价函数？即Logistic的代价函数的形式是什么样的？是否依然可以使用最小二乘法来构建代价函数呢？虽然我们都知道线性回归是连续的，逻辑斯蒂回归是离散的，但是这里还是要重申一下，二者的模型的形式已经不同了。Logistic回归通常使用Sigmoid函数写为：

如果输出大于0.5便为1，小于0.5便为0。如果直接套用线性回归的代价函数，那么得到的代价函数将非凸的（non-convex），利用梯度下降我们可能停留在局部最小值，而不是我们想要的全局最小值。因此我们需要重新定义代价函数。常用的策略就是非线性转线性，利用log来做。因此，代价函数的形式如下：

该公式可以进一步简化为：

至于为什么采用这种形式的代价函数，我想用最大似然法来解释会比较通俗易懂。其实这里就是一个负对数似然损失项，利用交叉熵求出来的。Negative Log-Likelihood Loss的表达式如下：

\mathcal{L}(\theta, \mathcal{D}) =
    \sum_{i=0}^{|\mathcal{D}|} \log P(Y=y^{(i)} | x^{(i)}, \theta)

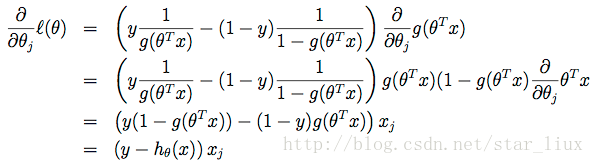
由于二值分类很像二项分布，我们把单一样本的类值假设为发生概率。则其概率公式为：

因此后验分布的概率为：

注意这里是后验概率，即表示已经发生了，需要使用乘法表示同时发生。因此，为了使所有样本的后验概率达到最大，即使样本的似然概率达到最大，我们需要最大化似然概率：

为了方便求解最大值和最优化问题，我们这里对取log对数，得到：

推导到这里，你就会发现，这里的形式和上面Cost函数的和形式是一样的！因此，我们这里的目标就只剩下利用梯度下降算法来求解以上函数的最小值(不带负号则为求最大值)。通过对log求导，你会很惊奇的发现，梯度的形式和线性回归是完全一样的！如下所示：



因此，根据这个推导，我们就明白，逻辑斯蒂回归的梯度下降算法的实现和线性回归几乎是完全一样的，只是在计算error矩阵的时候，需要明白是计算的是sigmoid函数，而不是直接，其他方面和线性回归是一致的。正则化的策略和惩罚项也都是和线性回归一致的，因为模型的参数都是。

1. **Softmax回归**