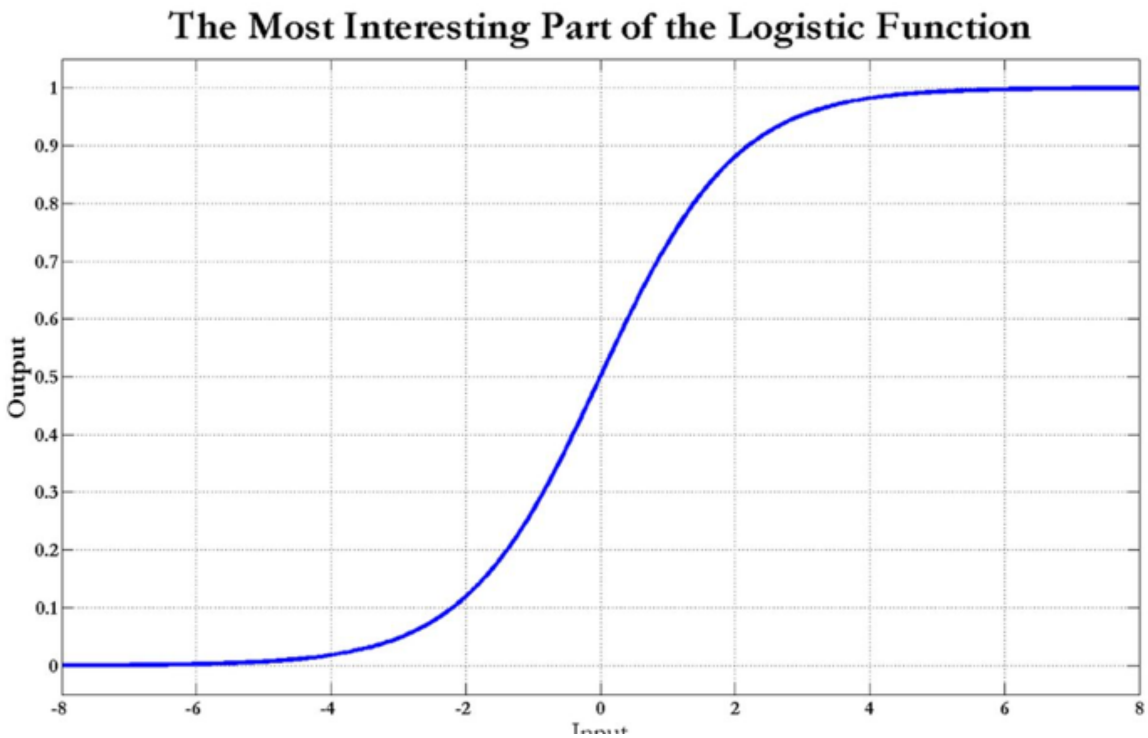
逻辑回归 logistics regression 公式推导



原创，转载请注明出处。

（常规字母代表标量，粗体字母代表向量，大写粗体字母代表矩阵）

逻辑回归虽然名字里面有回归，但是主要用来解决分类问题。

一、线性回归（Linear Regression）

线性回归的表达式：

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$$

线性回归对于给定的输入 $\boldsymbol{x}$，输出的是一个数值 y，因此它是一个解决回归问题的模型。

为了消除掉后面的常数项b，我们可以令

$$\boldsymbol{x}' \quad = [1 \quad \boldsymbol{x}]^T$$

，同时

$$\boldsymbol{w}' \quad = [b \quad \boldsymbol{w}]^T$$

，也就是说给x多加一项而且值恒为1，这样b就到了w里面去了，直线方程可以化简成为：

$$f(\boldsymbol{x}') = \boldsymbol{w}'^T \boldsymbol{x}'$$

在接下来的文章中为了方便，我们所使用的 $\boldsymbol{w}, \boldsymbol{x}$ 其实指代的是

$$w', x'$$

。

二、分类问题（Classification）

二分类问题就是给定的输入 $\boldsymbol{x}$，判断它的标签是A类还是类。二分类问题是最简单的分类问题。我们可以把多分类问题转化成一组二分类问题。比如最简单的是OVA(One-vs-all)方法，比如一个10分类问题，我们可以分别判断输入 $\boldsymbol{x}$ 是否属于某个类，从而转换成10个二分类问题。

因此，解决了二分类问题，相当于解决了多分类问题。

三、如何用连续的数值去预测离散的标签值呢?

线性回归的输出是一个数值，而不是一个标签，显然不能直接解决二分类问题。那我如何改进我们的回归模型来预测标签呢?

一个最直观的办法就是设定一个阈值，比如0，如果我们预测的数值 y > 0，那么属于标签A，反之属于标签B，采用这种方法的模型又叫做**感知机**（Perceptron）。

另一种方法，我们不去直接预测标签，而是去预测标签为A概率，我们知道

概率是一个[0,1]区间的连续数值，那我们的输出的数值就是标签为A的概率。一般的如果标签为A的概率大于0.5，我们就认为它是A类，否则就是B类。这就是我们的这次的主角**逻辑回归模型** (Logistics Regression)。

四、逻辑回归（logistics regression）

明确了预测目标是标签为A的概率。

我们知道，概率是属于[0,1]区间。但是线性模型

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$$

值域是

$$(-\infty, \infty)$$

。

我们不能直接基于线性模型建模。需要找到一个模型的值域刚好在[0,1]区间，同时要足够好用。

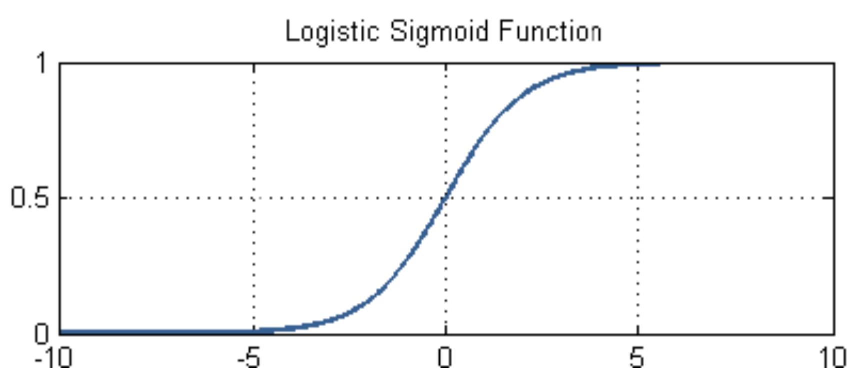
于是，选择了我们的sigmoid函数。

它的表达式为：

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

。

它的图像：



sigmoid函数

这个函数的有很多非常好的性质，一会儿你就会感受到。但是我们不能直接拿了sigmoid函数就用，毕竟它连要训练的参数 w 都没得。

我们结合sigmoid函数，线性回归函数，把线性回归模型的输出作为sigmoid函数的输入。于是最后就变成了逻辑回归模型：

$$y=\sigma(f(\boldsymbol{x}))=\sigma(\boldsymbol{w}^T\boldsymbol{x})=\frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}}$$

假设我们已经训练好了一组权值 $\boldsymbol{w}^T$ 。只要把我们需要预测的 $\boldsymbol{x}$ 代入到上面的方程，输出的y值就是这个标签为A的概率，我们就能够判断输入数据是属于哪个类别。

接下来就来详细介绍，如何利用一组采集到的真实样本，训练出参数w的值。

五、逻辑回归的损失函数（Loss Function）

损失函数就是用来衡量模型的输出与真实输出的差别。

假设只有两个标签1和0，

$$y_n\in\{0,1\}$$

。我们把采集到的任何一组样本看做一个事件的话，那么这个事件发生的概率假设为p。我们的模型y的值等于标签为1的概率也就是p。

$$P_{y=1}=\frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}}=p$$

因为标签不是1就是0，因此标签为0的概率就是：

$$P_{y=0}=1-p$$

。

我们把单个样本看做一个事件，那么这个事件发生的概率就是：

$$P(y|\boldsymbol{x})=\left\{\begin{array}{r}p, y=1 \\ 1-p, y=0\end{array}\right.$$

这个函数不方便计算，它等价于:

$$P(y_i|\boldsymbol{x}_i)=p^{y_i}(1-p)^{1-y_i}$$

。

解释下这个函数的含义，我们采集到了一个样本

$$(\boldsymbol{x_i}, y_i)$$

。对这个样本，它的标签是 y_i 的概率是

$$p^{y_i}(1-p)^{1-y_i}$$

。（当y=1，结果是p；当y=0，结果是1-p）。

如果我们采集到了一组数据一共N个，

$$\{(\boldsymbol{x}_1, y_1),(\boldsymbol{x}_2, y_2),(\boldsymbol{x}_3, y_3) \ldots(\boldsymbol{x}_N, y_N)\}$$

，这个合成在一起的合事件发生的总概率怎么求呢？其实就是将每一个样本发生的概率相乘就可以了，即采集到这组样本的概率：

$$\begin{aligned} P_{\text{总}} &= P(y_1|\boldsymbol{x}_1)P(y_2|\boldsymbol{x}_2)P(y_3|\boldsymbol{x}_3)\ldots.P(y_N|\boldsymbol{x}_N) \\ &= \prod_{n=1}^{N} p^{y_n}(1-p)^{1-y_n} \end{aligned}$$

注意$P_{\text{总}}$ 是一个函数，并且未知的量只有 $\boldsymbol{w}$ （在p里面）。

由于连乘很复杂，我们通过两边取对数来把连乘变成连加的形式，即：

$$F(\boldsymbol{w}) = ln(P_{总}) = ln(\prod_{n=1}^{N} p^{y_n}(1-p)^{1-y_n})$$
$$= \sum_{n=1}^{N} ln(p^{y_n}(1-p)^{1-y_n})$$
$$= \sum_{n=1}^{N} (y_n ln(p) + (1-y_n) ln(1-p))$$

其中，

$$p = \frac{1}{1+e^{-\boldsymbol{w}^T \boldsymbol{x}}}$$

这个函数

$$F(\boldsymbol{w})$$

又叫做它的**损失函数**。损失函数可以理解成衡量我们当前的模型的输出结果，跟实际的输出结果之间的差距的一种函数。这里的损失函数的值等于事件发生的总概率，我们希望它越大越好。但是跟损失的含义有点儿违背，因此也可以在前面取个负号。

六、最大似然估计MLE(Maximum Likelihood Estimation)

我们在真实世界中并不能直接看到概率是多少，我们只能观测到事件是否发生。也就是说，我们只能知道一个样本它实际的标签是1还是0。那么我们如何估计参数 $\boldsymbol{w}$ 跟b的值呢?

最大似然估计MLE(Maximum Likelihood Estimation)，就是一种估计参数 $\boldsymbol{w}$ 的方法。在这里如何使用MLE来估计 $\boldsymbol{w}$ 呢?

在上一节，我们知道损失函数

$$F（\boldsymbol{w}）$$

是正比于总概率 $P_{总}$ 的，而

$$F（\boldsymbol{w}）$$

又只有一个变量 $\boldsymbol{w}$ 。也就是说，通过改变 $\boldsymbol{w}$ 的值，就能得到不同的总概率值 $P_{总}$ 。那么当我们选取的某个 $\boldsymbol{w}^*$ 刚好使得总概率 $P_{总}$ 取得最大值的时候。我们就认为这个 $\boldsymbol{w}^*$ 就是我们要求得的 $\boldsymbol{w}$ 的值，这就是最大似然估计的思想。

现在我们的问题变成了，找到一个 $\boldsymbol{w}^*$ ，使得我们的总事件发生的概率，即损失函数

$$F（\boldsymbol{w}）$$

取得最大值，这句话用数学语言表达就是：

$$\boldsymbol{w^*} = arg \max_{w} F(\boldsymbol{w}) = -arg \min_{w} F(\boldsymbol{w})$$

七、求

$$F（\boldsymbol{w}）$$

的梯度

$$\nabla F（\boldsymbol{w}）$$

梯度的定义

我们知道对于一个一维的标量x，它有导数 x' 。

对一个多维的向量

$$\boldsymbol{x} = (x_1, x_2, x_3, .., x_n)$$

来说，它的导数叫做梯度，也就是分别对于它的每个分量求导数

$$\boldsymbol{x'} = (x'_1, x'_2, x'_3, .., x'_n)$$

。

接下来请拿出纸笔，一起动手来推导出

$$\nabla F（\boldsymbol{w}）$$

的表达式。请尽量尝试自己动手推导出来，如果哪一步不会了再看我的推

导。

七（二）、求梯度的推导过程

为了求出

$$F\left(\boldsymbol{w}\right)$$

的梯度

$$\nabla F\left(\boldsymbol{w}\right)$$

，我们需要做一些准备工作。原谅我非常不喜欢看大串的数学公式，所以我尽可能用最简单的数学符号来描述。当然可能不够严谨，但是我觉得更容易看懂。

首先，我们需要知道向量是如何求导的。具体的推导过程以及原理请参见 [矩阵求导]

我们只要记住几个结论就行了：对于一个矩阵 $\boldsymbol{A}$ 乘以一个向量的方程 $\boldsymbol{Ax}$ ，对向量 $\boldsymbol{w}$ 求导的结果是 $\boldsymbol{A}^T$ 。在这里我们把函数 $\boldsymbol{Ax}$ 对 $\boldsymbol{w}$ 求梯度简单记作

$$\left(\boldsymbol{Ax}\right)'$$

。因此

$$\left(\boldsymbol{Ax}\right)' = \boldsymbol{A}^T$$

, 推论是

$$\left(\boldsymbol{x}^T\boldsymbol{A}\right)' = \boldsymbol{A}$$

，我们把

$$\boldsymbol{x}, \boldsymbol{w}^T$$

代入进去，可以知道

$$(\boldsymbol{w}^T\boldsymbol{x})' = \boldsymbol{x}$$

。

然后求

$$1-p$$

的值：

$$1-p=\frac{e^{-\boldsymbol{w}^T\boldsymbol{x}}}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}}$$

p是一个关于变量 $\boldsymbol{w}$ 的函数，我们对p求导，通过链式求导法则，慢慢展开可以得：

$$\begin{aligned}
p' = f'(\boldsymbol{w}) &= (\frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}})' \\
&= -\frac{1}{(1+e^{-\boldsymbol{w}^T\boldsymbol{x}})^2}\cdot(1+e^{-\boldsymbol{w}^T\boldsymbol{x}})' \\
&= -\frac{1}{(1+e^{-\boldsymbol{w}^T\boldsymbol{x}})^2}\cdot e^{-\boldsymbol{w}^T\boldsymbol{x}}\cdot(-\boldsymbol{w}^T\boldsymbol{x})' \\
&= -\frac{1}{(1+e^{-\boldsymbol{w}^T\boldsymbol{x}})^2}\cdot e^{-\boldsymbol{w}^T\boldsymbol{x}}\cdot(-\boldsymbol{x}) \\
&= \frac{e^{-\boldsymbol{w}^T\boldsymbol{x}}}{(1+e^{-\boldsymbol{w}^T\boldsymbol{x}})^2}\cdot\boldsymbol{x} \\
&= \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}}\cdot\frac{e^{-\boldsymbol{w}^T\boldsymbol{x}}}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}}\cdot\boldsymbol{x} \\
&= p(1-p)\boldsymbol{x}
\end{aligned}$$

上面都是我们做的准备工作，总之我们得记住：

$$p'=p(1-p)\boldsymbol{x}$$

, 并且可以知道

$$(1-p)'=-p(1-p)\boldsymbol{x}$$

。

下面我们正式开始对

$$F(\boldsymbol{w})$$

求导，求导的时候请始终记住，我们的变量只有 $\boldsymbol{w}$ ，其他的什么

$$y_n, \boldsymbol{x}_n$$

都是已知的，可以看做常数。

$$\begin{aligned}
\nabla F\text{（}\boldsymbol{w}\text{）} &= \nabla\text{（}\sum_{n=1}^{N}(y_n ln(p) + (1-y_n) ln(1-p))\text{）} \\
&= \sum (y_n ln'(p) + (1-y_n) ln'(1-p)) \\
&= \sum ((y_n \frac{1}{p} p') + (1-y_n) \frac{1}{1-p} (1-p)') \\
&= \sum (y_n (1-p) \boldsymbol{x}_n - (1-y_n) p \boldsymbol{x}_n) \\
&= \sum_{n=1}^{N} (y_n - p) \boldsymbol{x}_n
\end{aligned}$$

终于，我们求出了梯度

$$\nabla F\text{（}\boldsymbol{w}\text{）}$$

的表达式了，现在我们再来看看它长什么样子：

$$\nabla F\text{（}\boldsymbol{w}\text{）} = \sum_{n=1}^{N} (y_n - p) \boldsymbol{x}_n$$

它是如此简洁优雅，这就是我们选取sigmoid函数的原因之一。当然我们也能够把p再展开，即：

$$\nabla F\text{（}\boldsymbol{w}\text{）} = \sum_{n=1}^{N} (y_n - \frac{1}{1+e^{-\boldsymbol{w}^T \boldsymbol{x}_n}}) \boldsymbol{x}_n$$

八、梯度下降法（GD）与随机梯度下降法（SGD）

现在我们已经解出了损失函数

$$F\text{（}\boldsymbol{w}\text{）}$$

在任意 $\boldsymbol{w}$ 处的梯度

$$\nabla F\text{（}\boldsymbol{w}\text{）}$$

，可是我们怎么算出来 $\boldsymbol{w}^*$ 呢？ 回到之前的问题，我们现在要求损失函数取

最大值时候的$\boldsymbol{w}^*$的值：

$$\boldsymbol{w}^* = arg\max_w F(\boldsymbol{w})$$

梯度下降法(Gradient Descent)，可以用来解决这个问题。核心思想就是先随便初始化一个 $\boldsymbol{w}_0$ ，然后给定一个步长 η ，通过不断地修改

$$\boldsymbol{w}_{t+1}$$

<- $\boldsymbol{w}_t$ ，从而最后靠近到达取得最大值的点，即不断进行下面的迭代过程，直到达到指定次数，或者梯度等于0为止。

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta \nabla F（\boldsymbol{w}）$$

随机梯度下降法（Stochastic Gradient Descent），如果我们能够在每次更新过程中，加入一点点噪声扰动，可能会更加快速地逼近最优值。在SGD中，我们不直接使用

$$\nabla F（\boldsymbol{w}）$$

，而是采用另一个输出为随机变量的替代函数

$$G(\boldsymbol{w})$$

:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta G(\boldsymbol{w})$$

当然，这个替代函数

$$G(\boldsymbol{w})$$

需要满足它的期望值等于

$$\nabla F（\boldsymbol{w}）$$

，相当于这个函数围绕着

$$\nabla F（\boldsymbol{w}）$$

的输出值随机波动。

在这里我先解释一个问题：**为什么可以用梯度下降法?**

因为逻辑回归的损失函数L是一个连续的凸函数（conveniently convex）。这样的函数的特征是，它只会有一个全局最优的点，不存在局部最优。对于GD跟SGD最大的潜在问题就是它们可能会陷入局部最优。然而这个问题在逻辑回归里面就不存在了，因为它的损失函数的良好特性，导致它并不会有好几个局部最优。当我们的GD跟SGD收敛以后，我们得到的极值点一定就是全局最优的点，因此我们可以放心地用GD跟SGD来求解。

好了，那我们要怎么实现学习算法呢？其实很简单，注意我们GD求导每次都耿直地用到了所有的样本点，从1一直到N都参与梯度计算。

$$\nabla F（\boldsymbol{w}） =\sum_{n=1}^{N}(y_n-\frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}_n}})\boldsymbol{x}_n$$

在SGD中，我们每次只要均匀地、随机选取其中一个样本

$$(\boldsymbol{x_i},y_i)$$

,用它代表整体样本，即把它的值乘以N，就相当于获得了梯度的无偏估计值，即

$$E(G(\boldsymbol{w}))=\nabla F(\boldsymbol{w})$$

，因此SGD的更新公式为：

$$\boldsymbol{w}_{t+1}=\boldsymbol{w}_t+\eta N(y_n-\frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}_n}})\boldsymbol{x}_n$$

这样我们前面的求和就没有了，同时 ηN 都是常数， N 的值刚好可以并入 η 当中,因此SGD的迭代更新公式为：

$$\boldsymbol{w}_{t+1}=\boldsymbol{w}_t+\eta(y_n-\frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}_n}})\boldsymbol{x}_n$$

其中

$$(\boldsymbol{x_i},y_i)$$

是对所有样本随机抽样的一个结果。

九、逻辑回归的可解释性

逻辑回归最大的特点就是**可解释性**很强。

在模型训练完成之后，我们获得了一组n维的权重向量 $\boldsymbol{w}$ 跟偏差 b。

对于权重向量 $\boldsymbol{w}$，它的每一个维度的值，代表了这个维度的特征对于最终分类结果的贡献大小。假如这个维度是正，说明这个特征对于结果是有正向的贡献，那么它的值越大，说明这个特征对于分类为正起到的作用越重要。

对于偏差b (Bias)，一定程度代表了正负两个类别的判定的容易程度。假如b是0，那么正负类别是均匀的。如果b大于0，说明它更容易被分为正类，反之亦然。

根据逻辑回归里的权重向量在每个特征上面的大小，就能够对于每个特征的重要程度有一个量化的清楚的认识，这就是为什么说逻辑回归模型有着很强的解释性的原因。

十、决策边界

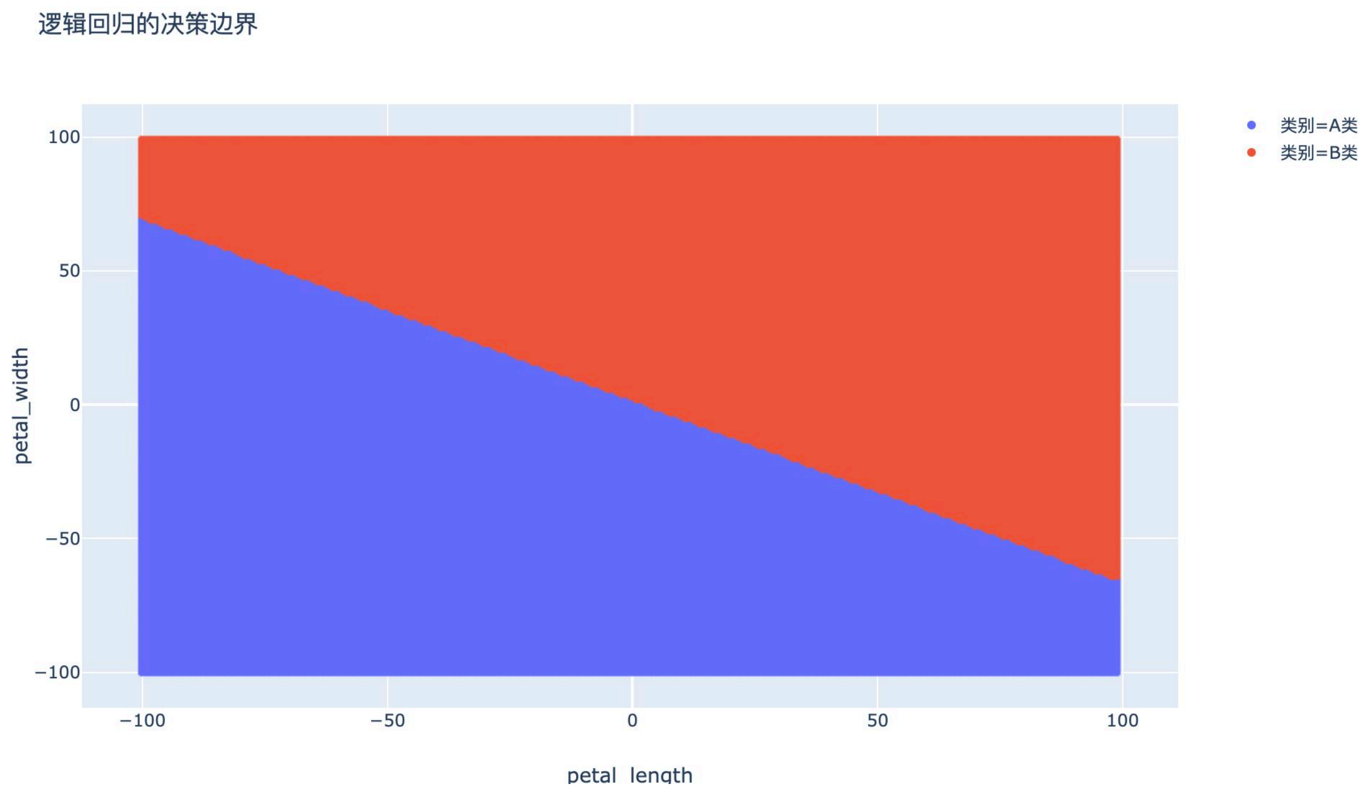
补充评论里的一个问题，逻辑回归的决策边界是否是线性的，相当于问曲线：

$$\frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}}=0.5$$

是不是的线性的，我们可以稍微化简一下上面的曲线公式，得到：

$$\begin{gathered} e^{-\boldsymbol{w}^T\boldsymbol{x}}=1=e^0 \\ \text{即}-\boldsymbol{w}^T\boldsymbol{x}=0 \end{gathered}$$

我们得到了一个等价的曲线，显然它是一个超平面（它在数据是二维的情况下是一条直线）。



十一、总结

终于一切都搞清楚了，现在我们来理一理思路，首先逻辑回归模型长这样：

$$y=\frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}}$$

其中我们不知道的量是 $\boldsymbol{w}$ ，假设我们已经训练好了一个 $\boldsymbol{w}^*$, 我们用模型来判断 $\boldsymbol{x}_i$ 的标签呢？很简单，直接将$\boldsymbol{x}_i$代入y中，求出来的值就是$\boldsymbol{x}_i$的标签是1的概率，如果概率大于0.5，那么我们认为它就是1类，否则就是0类。

那怎么得到 $\boldsymbol{w}^*$ 呢？

如果采用随机梯度下降法的话，我们首先随机产生一个$\boldsymbol{w}$的初始值 $\boldsymbol{w}_0$,然后通过公式不断迭代从而求得$\boldsymbol{w}^*$的值：

$$\boldsymbol{w}_{t+1}=\boldsymbol{w}_t+\eta(y_n-\frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}_n}})\boldsymbol{x}_n$$

每次迭代都从所有样本中随机抽取一个

$$(\boldsymbol{x}_i, y_i)$$

来代入上述方程。

原创，转载请注明出处。

初学者，不可避免出现错误。如果有任何问题，欢迎大家指正，也欢迎大家一起来交流讨论。