[原创] 用人话解释机器学习中的Logistic Regression（逻辑回归）

转载请注明出处：**http://www.codelast.com/**
友情提示：如果觉得页面中的公式显示太小，可以放大页面查看（不会失真）。

Logistic Regression（或Logit Regression），即逻辑回归，简记为LR，是机器学习领域的一种极为常用的算法 / 方法 / 模型。
你能从网上搜到十万篇讲述Logistic Regression的文章，也不多我这一篇，但是，就像我写过的最优化系列文章一样，我仍然试图用“人话”来再解释一遍——可能不专业，但是容易看得懂。那些一上来就是几页数学公式什么的最讨厌了，不是吗？
所以这篇文章是写给完全没听说过Logistic Regression的人看的，我相信看完这篇文章，你差不多可以从无到有，把逻辑回归应用到实践中去。

Logistic Regression是一种分类算法。分类，也就是把一个群体（或问题，或数据）分为几个类别，例如，男/女/人妖；爱她的人/不爱她的人；今天会下雨/今天不会下雨。
Logistic Regression最常用于处理“二分类”问题，也就是说分类只有两个，像“爱她的人/不爱她的人”就是二分类，而“男/女/人妖”就不是二分类。当然，Logistic Regression也可以用于处理多分类问题，即所谓的“多分类逻辑回归”（Multiclass Logistic Regression），但本文并不涉及这个方面。
所以，说得简单点就是，给你一条数据，用Logistic Regression可以判断出这条数据应该被分到两个类别中的哪个中去。
文章来源：http://www.codelast.com/
Logistic Regression在现实世界中非常有用。例如，可以用它来判断一个用户是否会点击一个广告（会点击 / 不会点击），可以用Logistic Regression来判断两类人是否会相爱（会相爱 / 不会相爱），等等。

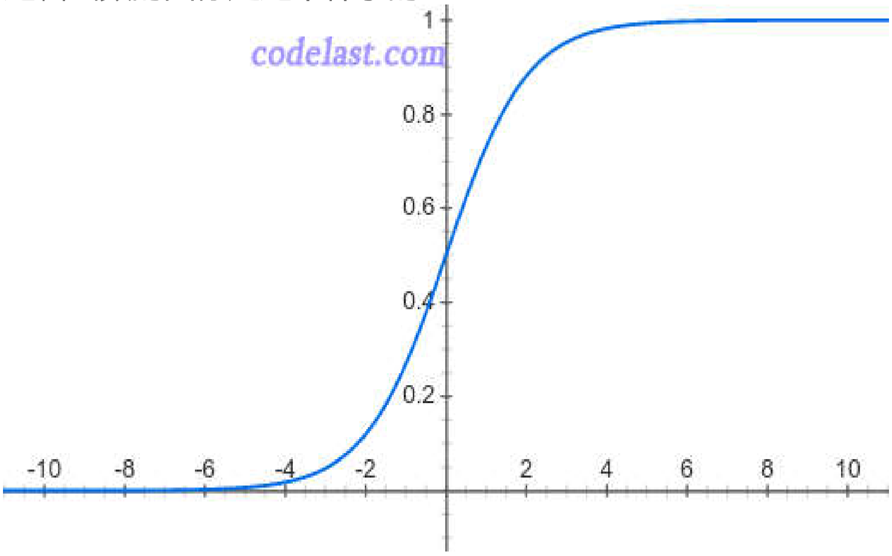
机器学习的主旨就是通过对历史数据的计算（即“学习”），得到一些未知参数的值，从而可以推断出新数据会有什么结论。
例如一个非常简单的函数： $y = ax + b$ ，在已知几组 (x, y) 历史数据的情况下：

> (1, 5.5)
> (1.5, 7)
> (2, 6.5)

我们怎样能够预测一个未知的自变量 $x = 3$ 会对应什么样的因变量 y 呢？也就是说， $x = 3$ 时 $y = ?$
显然我们的任务就是计算出两个未知参数 a 和 b 的值，有了这两个值，那么任意给定一个 x ，我们都能通过函数 $y = ax + b$ 计算出 y 的值了，这就是所谓的“预测”。
文章来源：http://www.codelast.com/
Logistic Regression也是类似，我们有一个函数 $y = f(X)$ ，里面包含若干个未知参数 $\theta_0, \theta_1, \theta_2, \cdots, \theta_n$ 。
由于现实世界是复杂的，因变量 y 通常会跟很多因素（自变量 x ）有关系，即 $x_0, x_1, x_2, \cdots, x_n$ ，所以这里自变量是一个向量，这里用大写的 X 来表示。同理，那一堆未知的参数也是一个向量，用一个字母 θ 来表示。
现在给我们一堆 (x, y) 的历史数据，我们要想办法计算出所有未知参数的值，然后就可以拿来预测新的 x 值所对应的 y 值了。
但是这个函数是什么呢？如下：
$f(X) = \frac{1}{1+e^{-\theta^T X}}$ ...... (1)
其中， θ 是参数向量， X 是自变量（向量）。
文章来源：http://www.codelast.com/
那么，这个略显奇怪的函数是怎么来的呢？
首先我们看 $\theta^T X$ 这部分：这是参数向量与自变量（向量）的点积，这个式子想要表达的含义是：计算某个事件发生的可能性，可以把跟这个事件相关的所有特征加权求和。例如，要求今天下雨的可能性，可以把今天所有和下雨相关的概率加权求和，例如梅雨季节权重为9（每天都很可能下雨），有台风经过权重为6，等等，每一个因素都影响着“下雨的可能性”，即：
$$s = \sum_{i=0}^{n} \theta_i x_i = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n = \theta^T X$$
但是这个加权求和的结果是在 $(-\infty, +\infty)$ 范围内的，为了能表示预测的概率，我们希望把输出值限制在 $(0, 1)$ 之间，而不是 $(-\infty, +\infty)$ 。所以，这时，逻辑函数就出场了。

文章来源：http://www.codelast.com/
通过这个WiKi页面你可以知道，其实所谓的逻辑函数，就是这

样的一个函数：
$P(t) = \frac{1}{1+e^{-t}}$
这个函数是由 Pierre François Verhulst（皮埃尔·弗朗索瓦·韦吕勒）在1844～1845年的时候给它起的名字。而我们上面的函数(1)，就是这个形式。
逻辑函数的图像是这个样子的：



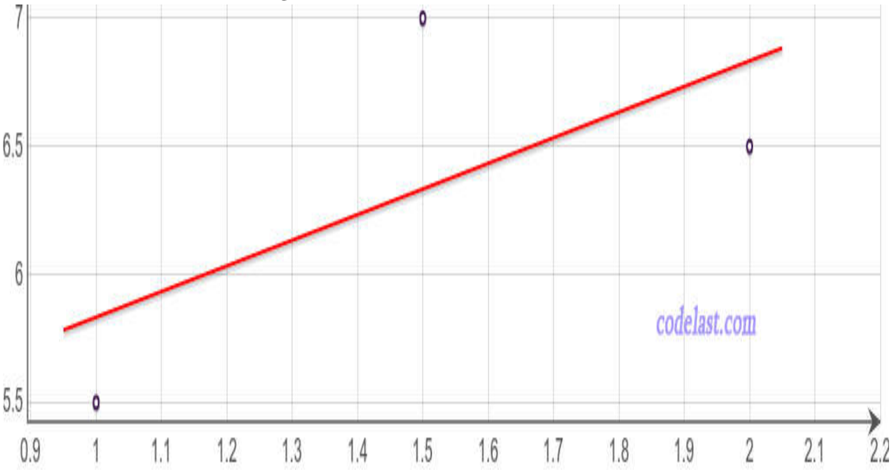
它的函数值刚好就是在(0,1)之间。
所以，我们通过逻辑函数，就可以计算出一个事件的概率了（(0,1)之间）。但是不要忘了，我们前面说要处理二分类问题，得到一个(0,1)之间的任意值并不能归到两个分类中的一个里去，所以还要把这个概率值“归类”。其实这里很简单，我们可以在 $f(X) > 0.5$ 的时候，把它归到类别1中，$f(X) \leq 0.5$ 的时候，把它归到类别2中就可以了（概率值的“分水岭”可以根据实际情况调整）。用数学公式来表达这段话的含义就是：

$$y' = \begin{cases} 0, f(X) > 0.5 \\ 1, f(X) \leq 0.5 \end{cases}$$

在各种机器学习的文章中，你都会看到，它们给了逻辑函数一个常用的名字：Sigmoid函数。sigmoid，意为“S形的”，这正符合其函数图像特点，所以大家记住就行了。
文章来源：http://www.codelast.com/
现在，我们已经有了函数，下一步任务就是求出函数表达式中的未知参数向量 θ 了。这个过程是机器学习中最为核心的计算步骤。
以前面讲过的函数 $y = ax + b$ 为例：
你会发现，当已知几组 (x, y) 数据的情况下：

> (1, 5.5)
> (1.5, 7)
> (2, 6.5)

你无论如何也不可能找到一对 a 和 b 的值，使得以上3组数据能精确地满足方程 $y = ax + b$，正如下面的图像所示：



这条直线如果要精确地通过其中的两个点，那么就不能通过第三个点。所以，最终求出来的 a 和 b 的值，并不是方程的解析解，而是"最优解"。

因此，问题在于，我们如何画一条直线，使得其是"最优"的？"最优"的评判标准是什么？

文章来源：http://www.codelast.com/

为了理解"最优"，我们需要先了解一些概念。

- 损失函数 / Loss Function / 代价函数 / Cost Function

很多文章说，这几个名词的含义是一样的。但是也有文章说，Loss Function和Cost Function不是一回事，例如这篇文章。但通常认为，这二者是一回事。我觉得嘛，大家就按通常的概念来接受就好了。

按WiKi的定义：

> In mathematical optimization, statistics, decision theory and machine learning, a loss function or cost function is a function that maps an event or values of one or more variables onto a real number intuitively representing some "cost" associated with the event. An optimization problem seeks to minimize a loss function.

以及：

> The loss function quantifies the amount by which the prediction deviates from the actual values.

我们可以知道，损失函数用于衡量预测值与实际值的偏离程度，如果预测是完全精确的，则损失函数值为0；如果损失函数值不为0，则其表示的是预测的错误有多糟糕。使得损失函数值最小的那些待求参数值，就是"最优"的参数值。

所以现在问题来了，损失函数的表达式又是什么？
在探讨损失函数的表达式之前，我们先来看一下损失函数有哪些种类。
损失函数有很多种，例如下面几个：
（1）0-1损失函数：可用于分类问题，即该函数用于衡量分类错误的数量，但由于此损失函数是非凸（non-convex）的，因此在做最优化计算时，难以求解，所以，正因为如此，0-1损失函数不是那么“实用”（如果这句话有误，请指正）。
（2）平方损失函数（Square Loss）：常用于线性回归（Linear Regression）。
（3）对数损失（Log Loss）函数：常用于其模型输出每一类概率的分类器（classifier），例如逻辑回归。
（4）Hinge损失函数：常用于SVM（Support Vector Machine，支持向量机，一种机器学习算法）。中文名叫“合页损失函数”，因为hinge有“合页”之意。这个翻译虽然直白，但是你会发现，99%的文章都不会用它的中文名来称呼它，而是用“Hinge损失”之类的说法。

这些都是人们的经验总结，当然，说每一种损失函数常用于什么机器学习算法，也都是有数学依据的。但是在这里，我们讲的是Logistic Regression，所以只看对数损失函数。对数损失函数通常用于衡量分类器（classifier）的精度，这里的“分类器”也就是指机器学习的模型，它对每一个类别输出一个概率值。从前面的文章中，我们已经知道了，逻辑回归就是这样一种分类器，所以才用对数损失函数来衡量其精度。
有时候，对数损失函数（Log Loss）也被叫作交叉熵损失函数（Cross-entropy Loss）。交叉熵这个名字比较拗口，在信息理论中，熵用于衡量某种事件的“不可预测性”，而交叉熵=事件的真实分布+不可预测性，所以交叉熵可以用于度量两个概率分布（真实分布&预测分布）之间的差异性，即：交叉熵损失函数（对数损失函数）可以衡量一个模型对真实值带来的额外噪音，通过最小化交叉熵损失函数（对数损失函数），我们就可以最大化分类器（模型）的精度。
上面这一大段话试图用简单的描述让你相信，为什么要用Log Loss来衡量Logistic Regression的误差，但是没有给出证明。有人可能会说，为什么不能用其他的方法来衡量，例如用平方损失函数（Square Loss）。事实上，这是有数学依据的——它会导致损失函数是一个关于参数向量 θ 的非凸函数，而用对数损失函数就没有这种问题。凸函数的性质为我们后面求解参数向量 θ 提供了极大便利，非凸函数有很多局部最优解，不利于求解 θ 的计算过程。

到这里为止，我们还是没有提到损失函数的数学表达式，但是如果要计算损失函数的值，我们是回避不了的，必须要知道。
所以，这里用 L 来表示损失函数（取Loss之意），则对数损失

函数的表达式为：

$$L=-\frac{1}{n}\sum_{i=1}^{n}\left[y_i\log(\hat{y}_i)+(1-y_i)\log(1-\hat{y}_i)\right] \ldots\ldots \ (2)$$

其中，y_i 是第i个真实值（$y_i\in\{0,1\}$），$\hat{y}_i$ 是第i个预测值。

这个对数损失函数的表达式中并没有出现我们要求解的参数 θ，所以我们把 $\hat{y}=f(X)=\frac{1}{1+e^{-\theta^T X}}$ 代到（2）式中去：

$$L=-\frac{1}{N}\sum_{i=1}^{n}\left[y_i\log\left(\frac{1}{1+e^{-\theta^T X_i}}\right)+(1-y_i)\log\left(1-\frac{1}{1+e^{-\theta^T X_i}}\right)\right]$$

再来仔细看一下这个式子：N 为数据集的条数（有多少组 (X,y)，N就是多少），已知；y_i 是真实值，已知；X_i 是输入的向量，也已知。所以整个式子里只有 θ 是未知的，可以记为 $L(\theta)$，称之为目标函数：

$$L(\theta)=-\frac{1}{N}\sum_{i=1}^{n}\left[y_i\log\left(\frac{1}{1+e^{-\theta^T X_i}}\right)+(1-y_i)\log\left(1-\frac{1}{1+e^{-\theta^T X_i}}\right)\right]$$

因此，我们只要找到一个参数向量 θ，能使得此式的值最小，那么这个参数向量 θ 就是"最优"的参数向量。

求得了这个最优的 θ 之后，把它代入式（1），则对任一个未知的 X，我们都可以计算出 $f(X)$ 值，然后再根据一个阈值把它调整到 0 或 1，就得到了这个 X 所属的分类，这样，我们就完成了一次"预测"的过程。

文章来源：http://www.codelast.com/

- 求解方法

所以现在问题来了，这个"最优"的参数向量 θ 怎么求解？
在大的方向上，你可以选择不使用搜索方向的算法（例如信赖域算法），也可以选择众多使用搜索方向的算法（例如梯度下降法）。
在是否计算目标函数的导数这个方面，你可以使用不用求目标函数导数的算法（例如Powell共轭方向集方法），也可以使用要求目标函数导数的算法（例如梯度下降法）。由于某些目标函数形式特别复杂，计算其导数特别麻烦，所以在这种时候，不用计算导数的算法可能大有帮助。
求解的过程就是一个最优化的过程，本文无法用一两句话描述清楚，请大家移步链接进行阅读。

事实上，在现在各种机器学习library百花齐放的今天，我们基本上不需要自己编写这些算法的具体实现，只需要调用它们即可。例如，通过Spark的Machine Learning Library (MLlib)，我们可以直接使用Stochastic gradient descent (SGD)，Limited-memory BFGS (L-BFGS)等实现。但是对这背后的原理有所了解，对工作学习是有帮助的。

Tagged on: cross entropy Logistic Regression Logit Regression machine learning optimization 交叉熵 最