Logistic Regression

Shen Hengheng

2017

该笔记是来自 Andrew Ng 的 Machine Learning 课程的第三周: **逻辑回归**的课堂记录，逻辑回归是机器学习分类算法的地一个算法，涵盖信息论的相关内容. 主要讲解了以下几个内容:

- 分类算法的应用
- 二元分类问题
- 边界函数
- 代价函数
- 梯度下降算法

0.1 分类算法的应用

分类的场景在现实社会中无处不在，比如

- Email: 垃圾邮件分类
- 金融交易是否存在欺诈
- 肿瘤是恶性/良性

分类问题和回归问题一样，只是现在要预测的值只占一小部分离散值。现在，将重点讨论二**分类**问题，其中 Y 只能接受两个值，**0 和 1**。(也可以推广到多个类)，例如，如果我们试图为电子邮件构建一个垃圾邮件分类器，那么 $X^{(i)}$ 可能是一封电子邮件的一些特征，如果它是一封垃圾邮件，y 是 1，否则为 0。因此，$Y \in \{0, 1\}$。0 也被称为否定类，1 是正类，它们有时也用符号 **"-"** 和 **"+"** 表示。

0.2 二分类问题

医疗诊断是机器学习在医学方面其中的一个应用，本小节主要借助乳腺癌诊断这个例子引出. 给出一组数据，其中数据特征为肿瘤的大小，标记 y 是 0 或者 1，其中 0 表示恶性，1 表示良性，将这些样本数据做图如下：

我们可以尝试使用线性回归加上一个合适的 **"阈值"** 来实现分类问题，如图1所示，$h_\theta(x) = \theta^T x$，则基于阈值 0.5 的分类器算法如下：

$$h_\theta(x) \geq 0.5 \rightarrow y = 1$$
$$h_\theta(x) < 0.5 \rightarrow y = 0$$

但是这将存在一个很大问题，该算法并不具有鲁棒性，即假设我们有观测到了一个非常大的尺寸的恶性肿瘤，将其作为实例加入到我们的训练集中，这样将会我们之前的

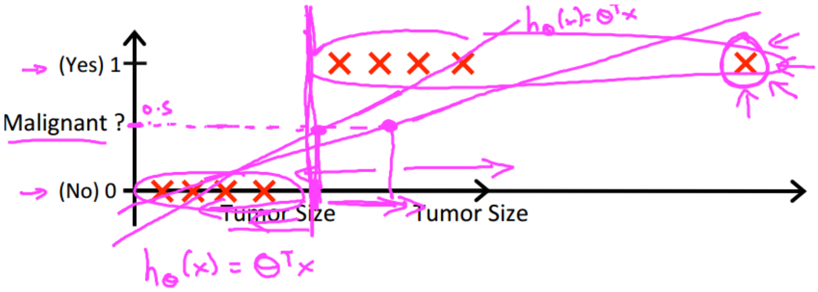


图 1: 乳腺癌数据

假说 h 发生很大变化，如图1中最右侧的那个点加入到训练集中，将对 $h(x)$ 影响很大，这时将 0.5 作为阈值来预测肿瘤是否为恶性就不再合适了。虽然很容易构造出模型来，但用原来的线性回归模型解决分类问题的这种方法表现很差。直观地说，也没有任何道理，通过图1很容易看出，存在 $h_\theta(x) > 1$ 或者 $h_\theta(x) < 0$，但，$Y \in \{0,1\}$。为了解决这个问题，我们改变 $h_\theta(x)$ 的形式，使得输出变量 $0 \le h_\theta(x) \le 1$. 我们将使用 $Sigmoid$ 来表示假说函数，也叫为逻辑函数，又因为逻辑函数的输入为 $\theta^T x$，所以我们的模型又叫做**逻辑回归模型**.

逻辑回归模型为：

$$
\begin{aligned}
&h_\theta(x) = g(\theta^T x) \\
&z = \theta^T x \\
&g(z) = \frac{1}{1+e^{-z}}
\end{aligned}
$$

$g(z)$ 函数图像如图2所示.

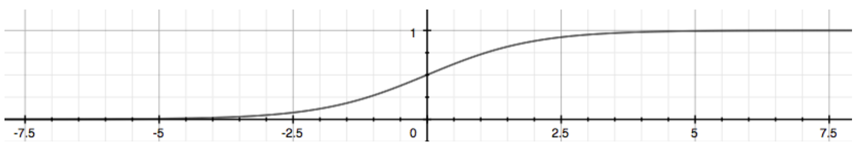


图 2: Sigmoid 函数图像

另外，对于给定的数据 X，根据选择的参数，$h_\theta(x)$ 表示 $y=1$ 发生的可能性。即 $h_\theta(x) = P(y=1|x,\theta)$, 例如，$h_\theta(x) = 0.7$ 告诉我们 $y=1$ 的概率为 70%，那么根据概率基本知识，$y=0$ 的概率为 30%.

基本的概率公式：

$$
\begin{aligned}
&h_\theta(x) = P(y=1|x;\theta) = 1 - P(y=0|x;\theta) \\
&P(y=0|x;\theta) + P(y=1|x;\theta) = 1
\end{aligned}
$$

0.3 边界函数

0.3.1 线性决策边界

根据图2和上面的公式，有以下结论.

$$z=0,e^0=1\Rightarrow g(z)=1/2$$
$$z\rightarrow\infty,e^{-\infty}\rightarrow 0\Rightarrow g(z)=1$$
$$z\rightarrow-\infty,e^{\infty}\rightarrow\infty\Rightarrow g(z)=0$$
$$\theta^Tx\geq 0\rightarrow h_\theta(x)\geq 0.5\rightarrow y=1$$
$$\theta^Tx<0\rightarrow h_\theta(x)<0.5\rightarrow y=0$$

假设有一个模型，模型表示如下

$$\theta=\begin{bmatrix}-3\\1\\1\end{bmatrix}$$
$$y=1\ if\ -3+1x_1+1x_2\geq 0$$
$$y=1\ if\ x_1+x_2\geq 3$$

画出边界，如下图3所示.

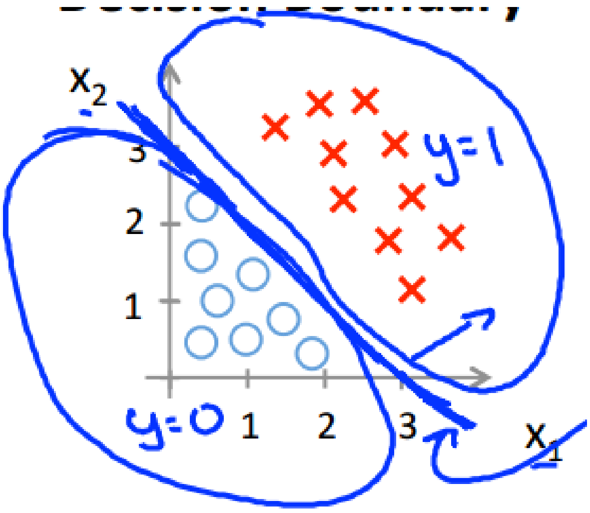


图 3: 线性决策边界

0.3.2 非线性决策边界

往往在实际的数据中，数据往往是线性不可分的！所以在此基础之上，引申出非线性决策边界的概念。如图所示，这个边界是一个圆 $z=\theta_0+\theta_1x_1^2+\theta_2x_2^2$ 或者一个其他的形状将 $y=0$ 的区域和 $y=1$ 的区域分开.

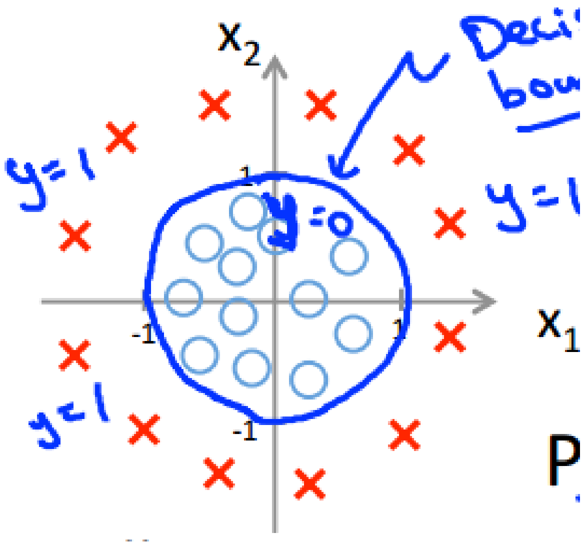


图 4: 非线性决策边界

图4，模型描述为：

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$
$$h_\theta(x) = g(-1 + 0x_1 + 0x_2 + 1x_1^2 + 1x_2^2)$$
$$y = 1\ if\ 1x_1^2 + 1x_2^2 \geq 1$$

我们也可以使用比较复杂的模型来适应非常复杂形状的边界.$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2 + ...)$

0.4 代价函数

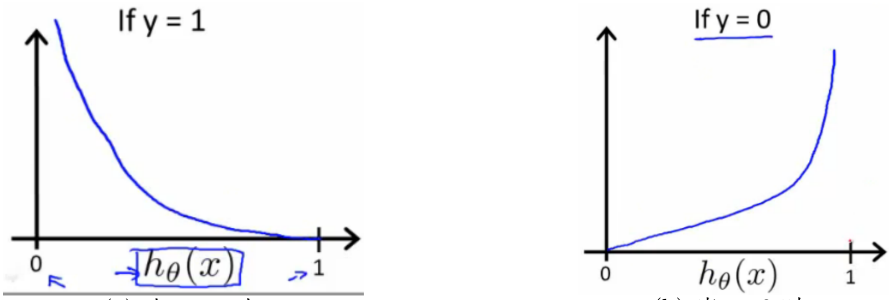
在定义逻辑回归模型的代价函数中，我们不能使用线性回归模型定义的成本函数，因为 *logistic* 函数会导致 y 出现震荡，从而出现许多局部的最优解，此时定义的代价函数不是一个凸函数。

逻辑回归模型的代价函数为

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

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

当 $y = 1$ 和 $y = 0$ 时，分别画出 $J(\theta) = \mathrm{Cost}(h_\theta(x), y)$ 的图像，如图5



(a) 当 y=1 时

(b) 当 y=0 时

图 5: $J(\theta)$ vs h_θ

$\mathrm{Cost}(h_\theta(x), y)$ 的特点为:

$$\mathrm{Cost}(h_\theta(x), y) = 0 \text{ if } h_\theta(x) = y$$
$$\mathrm{Cost}(h_\theta(x), y) \to \infty \text{ if } y = 0 \text{ and } h_\theta(x) \to 1$$
$$\mathrm{Cost}(h_\theta(x), y) \to \infty \text{ if } y = 1 \text{ and } h_\theta(x) \to 0$$

公式反映了一个事实: 如果 $h_\theta = 0$，即 $\mathrm{P}(y = 1|\theta, x) = 0$，但是 $\mathrm{y} = 1$，则我们将会以一个很大的代价（∞）惩罚学习算法!

上面的话简单点: 你预测对了，没事!（OK $\to$ Cost $= 0$)，但是你如果预测错了，抱歉，狠狠教训一顿!（Sorry, Cost $\to \infty$)，其实这是**信息论中熵的应用，信息增益**!

这样写过之后，就保证了代价函数是凸函数!

0.5 简化代价函数形式和梯度下降算法

我们可以将

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

简化为

$$\mathrm{Cost}(h_\theta(x), y) = -y\ \log(h_\theta(x)) - (1-y)\log(1 - h_\theta(x))$$

代入代价函数得到:

$$\begin{aligned} J(\theta) &= -\frac{1}{m}\sum_{i=1}^{m}\mathrm{Cost}(h_\theta(x^{(i)}) - y^{(i)}) \\ &= -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1 - h_\theta(x^{(i)}))] \end{aligned}$$

向量化之后:

$$h = g(X\theta)$$
$$J(\theta) = \frac{1}{m} \cdot \left(-y^T \log(h) - (1-y)^T \log(1-h)\right)$$

在得到这样的一个代价函数之后，就可以使用梯度下降算法求出使得代价函数最小的参数 θ.

梯度下降算法框架如下：

$$Repeat\ \{$$
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$
$$\}$$

将 $J(\theta)$ 代入得

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

向量化：

$$\theta := \theta - \frac{\alpha}{m} X^T (g(X\theta) - \vec{y})$$

注意：表面上看起来和线性回归的梯度下降一样，但是这里的 $h_\theta(x) = g(\theta^T x)$ 与线性回归不同，所以实际上是不一样的。另外在运行梯度下降算法之前，进行特征缩放是非常必要的。

0.6 高级优化

“共轭梯度”、“BFGS” -局部优化法、和 “L-BFGS” -有限内存局部优化，使用更成熟，更快的方法来优化 θ，它们可以用来代替梯度下降。不建议自己编写这些复杂的算法，而是使用库，因为它们已经经过测试并高度优化。Octave 提供它们！

$$\text{Given } \theta$$
$$\rightarrow J(\theta)$$
$$\rightarrow \frac{\partial}{\partial_{\theta_j}} J(\theta) \text{ for } j = 0, 1, 2, 3, ...$$

代码实现：

```
function [jVal, gradient] = costFunction(theta)
  jVal = [...code to compute J(theta)...];
  gradient = [...code to compute derivative of J(theta)...];
end
```