逻辑回归：

PS:逻辑回归本身跟逻辑没什么关系，只是翻译的结果。

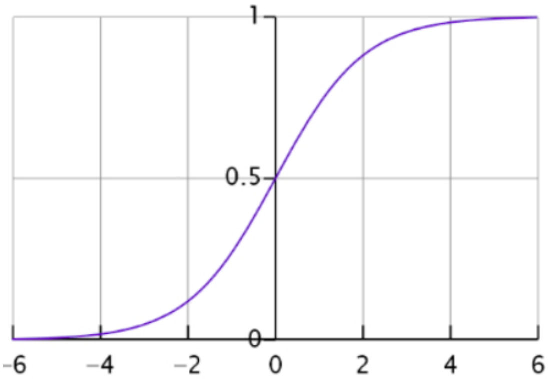
逻辑回归主要是用来分类，我们最熟悉的分类就是以0为界，把数分为正数和负数，但是考虑到实际数据的多维性，单单以一个数来判断显然是不够科学的，介于此，数学家们提出了逻辑回归，即采取一个非线性变换sigmoid函数，对其进行映射，然后就可以再根据数值来分类了。

sigmoid函数：

$$S(t) = \frac{1}{1+e^{-t}}$$

```
def sigmoid(x):
    return 1/(1+np.exp(-x))
```

图形：



可以明显的看到，映射后的值位于[0,1]这个区间，若以0.5作为闸值，大于0.5的为是，小于0.5的为否，那么一个简单的分类器就构建完成了。

则我们一般的模型方程如下：

$$H(a,b) = \frac{1}{1+e^{(aX+b)}}$$

ps:a、b是参数，将负号抵消了。

以概率的角度来看，y的值可以对应预测结果为1的概率（后验概率），换言之，可以把模型改写成如下形式：

$$P(Y=1|x) = \frac{1}{1+e^{-(w^T x+b)}}$$

设:

$$P(Y=1|x) = p(x)$$
$$P(Y=0|x) = 1-p(x)$$

很明显可以把上述概率转化为似然函数的形式，然后最大化似然函数就可得到对应的参数。对应的似然函数：

$$L(w) = \prod [p(x_i)]^{y_i} [1 - p(x_i)]^{1-y_i}$$

对于这种形式，想要直接求是很困难的，所以要对其进行对数化。

损失函数:

- $J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m}(y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$

```
def logistic(X, y, W, b):
    num_train = X.shape[0]
    num_features = X.shape[1]
    a = sigmoid(np.dot(X, W) + b)
    loss = -1/num_train * np.sum(y*np.log(a)+(1-y)*np.log(1-a))
    dW = np.dot(X.T, (a-y))/num_train
    db = np.sum(a-y)/num_train
    loss = np.squeeze(loss)
    return a, loss, dW, db
```

PS:如果直接采用之前回归的损失函数是行不通的，因为将模型带入其中，会得到一个非凸(这里凸指的是能求得最小值的，某些教材凹凸定义与之相反)函数，也就是说有很多局部最小值，这将影响梯度下降算法寻找最小值。

优化方法(一阶方法)：

梯度下降(gradient descent):线性回归采取的优化方法，本质上是遍历所有样本，进行一个更新，学习速度比较慢。

随机梯度下降(stochastic gradient descent):随机选取一个点，根据该点进行梯度下降，样本量大时，效果不是很好。

批梯度下降(batch gradient descent):训练采取一批一批的训练，效果比较好。