逻辑回归

逻辑回归的概念：

逻辑回归也被称为广义线性回归模型，它与线性回归模型的形式基本上相同，都具有 ax+b，其中 a 和 b 是待求参数，其区别在于他们的因变量不同，多重线性回归直接将 ax+b 作为因变量，即 y = ax+b，而 logistic 回归则通过函数 S 将 ax+b 对应到一个隐状态 p，p = S(ax+b)，然后根据 p 与 1-p 的大小决定因变量的值。这里的函数 S 就是 Sigmoid 函数。

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

通过函数 S 的作用，我们可以将输出的值限制在区间[0， 1]上，p(x)则可以用来表示概率 p(y=1|x)，即当一个 x 发生时，y 被分到 1 那一组的概率。

逻辑回归模型的代价函数：

逻辑回归一般使用交叉熵作为代价函数。关于代价函数的具体细节，请参考代价函数，这里只给出交叉熵公式：

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

m：训练样本的个数；

$h_\theta(x)$：用参数 θ 和 x 预测出来的 y 值；

y：原训练样本中的 y 值，也就是标准答案

上角标(i)：第 i 个样本

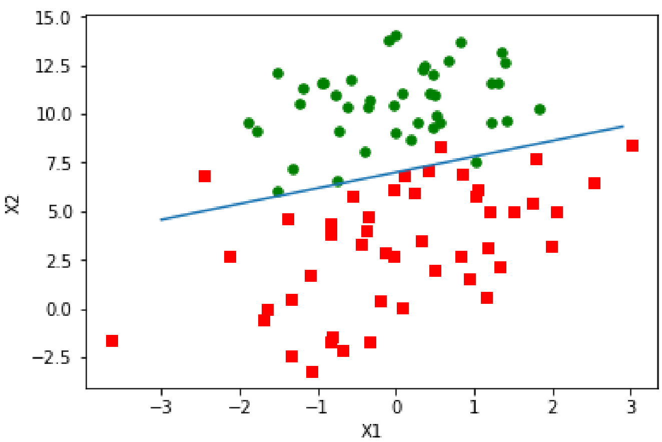
代码实现：

1.py

实验截图：

训练数据：

```
[[1.0, -0.017612, 14.053064], [1.0, -1.395634, 4.662541], [1.0, -0.752157, 6.53862], [1.0,
-1.322371, 7.152853], [1.0, 0.423363, 11.054677], [1.0, 0.406704, 7.067335], [1.0, 0.667394,
12.741452], [1.0, -2.46015, 6.866805], [1.0, 0.569411, 9.548755], [1.0, -0.026632, 10.427743],
[1.0, 0.850433, 6.920334], [1.0, 1.347183, 13.1755], [1.0, 1.176813, 3.16702], [1.0,
-1.781871, 9.097953], [1.0, -0.566606, 5.749003], [1.0, 0.931635, 1.589505], [1.0, -0.024205,
6.151823], [1.0, -0.036453, 2.690988], [1.0, -0.196949, 0.444165], [1.0, 1.014459, 5.754399],
[1.0, 1.985298, 3.230619], [1.0, -1.693453, -0.55754], [1.0, -0.576525, 11.778922], [1.0,
-0.346811, -1.67873], [1.0, -2.124484, 2.672471], [1.0, 1.217916, 9.597015], [1.0, -0.733928,
9.098687], [1.0, -3.642001, -1.618087], [1.0, 0.315985, 3.523953], [1.0, 1.416614, 9.619232],
[1.0, -0.386323, 3.989286], [1.0, 0.556921, 8.294984], [1.0, 1.224863, 11.58736], [1.0,
-1.347803, -2.406051], [1.0, 1.196604, 4.951851], [1.0, 0.275221, 9.543647], [1.0, 0.470575,
9.332488], [1.0, -1.889567, 9.542662], [1.0, -1.527893, 12.150579], [1.0, -1.185247,
11.309318], [1.0, -0.445678, 3.297303], [1.0, 1.042222, 6.105155], [1.0, -0.618787,
10.320986], [1.0, 1.152083, 0.548467], [1.0, 0.828534, 2.676045], [1.0, -1.237728, 10.549033],
[1.0, -0.683565, -2.166125], [1.0, 0.229456, 5.921938], [1.0, -0.959885, 11.555336], [1.0,
0.492911, 10.993324], [1.0, 0.184992, 8.721488], [1.0, -0.355715, 10.325976], [1.0, -0.397822,
8.058397], [1.0, 0.824839, 13.730343], [1.0, 1.507278, 5.027866], [1.0, 0.099671, 6.835839],
[1.0, -0.344008, 10.717485], [1.0, 1.785928, 7.718645], [1.0, -0.918801, 11.560217], [1.0,
-0.364009, 4.7473], [1.0, -0.841722, 4.119083], [1.0, 0.490426, 1.960539], [1.0, -0.007194,
9.075792], [1.0, 0.356107, 12.447863], [1.0, 0.342578, 12.281162], [1.0, -0.810823,
-1.466018], [1.0, 2.530777, 6.476801], [1.0, 1.296683, 11.607559], [1.0, 0.475487, 12.040035],
[1.0, -0.783277, 11.009725], [1.0, 0.074798, 11.02365], [1.0, -1.337472, 0.468339], [1.0,
-0.102781, 13.763651], [1.0, -0.147324, 2.874846], [1.0, 0.518389, 9.887035], [1.0, 1.015399,
7.571882], [1.0, -1.658086, -0.027255], [1.0, 1.319944, 2.171228], [1.0, 2.056216, 5.019981],
[1.0, -0.851633, 4.375691], [1.0, -1.510047, 6.061992], [1.0, -1.076637, -3.181888], [1.0,
1.821096, 10.28399], [1.0, 3.01015, 8.401766], [1.0, -1.099458, 1.688274], [1.0, -0.834872,
-1.733869], [1.0, -0.846637, 3.849075], [1.0, 1.400102, 12.628781], [1.0, 1.752842, 5.468166],
[1.0, 0.078557, 0.059736]] [0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0,
0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1,
1, 1, 1, 0, 1, 1]
[[ 5.262118  ]
 [ 0.60847797]
 [-0.75168429]]
[[0.00490753]
 [0.71262076]
 [0.47239538]
 [0.28512035]
```



测试数据：

```
[[0.99714035]
 [0.04035907]
 [0.12535895]
 [0.99048731]
 [0.98075409]
 [0.97708653]
 [0.09004989]
 [0.97884487]
 [0.28594188]
 [0.00359693]]
```