中山大學

《机器学习》– 第一次课程作业报告

题目：使用逻辑回归模型对训练集进行分类

小组：20354041 黄标

20354034 郭凯

20354049 金彦宏

20354207 高李毅

专业：智能科学与技术

时间：二〇二一年 十 月

指导老师：彭卫文

一、逻辑回归模型

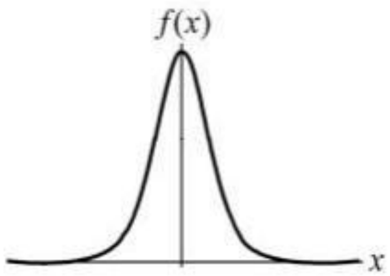
1、 Logistic 分布

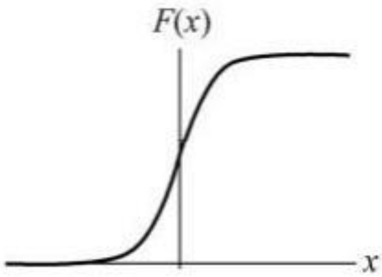
Logistic Regression 虽然被称为回归，但其实际上是分类模型，并常用于二分类。Logistic 回归的本质是：假设数据服从这个分布，然后使用极大似然估计做参数的估计。

Logistic 分布是一种连续型的概率分布，其密度函数和分布函数分别为：

$$f(x)=F'(X\le x)=\frac{e^{-(x-\mu)/\gamma}}{\gamma(1+e^{-(x-\mu)/\gamma})^2}$$

$$\mathrm{F}(x)=P(X\le x)=\frac{1}{1+e^{-(x-\mu)/\gamma}}$$





2、 Logistic 回归

最简单的回归是线性回归，逻辑回归其实仅为在线性回归的基础上，套用了一个逻辑函数。考虑二分类问题，给定数据集：

$$D=(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N),x_i\subseteq R^n,y_i\in 0,1,i=1,2,\cdots,N$$

考虑到$\omega^{\mathrm{T}}x+b$取值是连续的，因此它不能拟合离散变量。可以考虑用它来拟合条件概率 $p(Y=1/x)$，因为概率的取值也是连续的。为了使概率取值为 0 到 1，因此考虑采用对数几率函数。

$$y=\frac{1}{1+e^{-(\omega^T x+b)}}$$

于是有

$$ln\frac{y}{1-y}=w^Tx+b$$

我们将 y 视为 x 为正例的概率，则 1-y 为 x 为其反例的概率。两者的比值称为几率（odds），指该事件发生与不发生的概率比值，若事件发生的概率为 p。则对数几率：

$$ln(odds)=ln\frac{y}{1-y}$$

将 y 视为类后验概率估计，重写公式有：

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

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

也就是说，输出 Y=1 的对数几率是由输入 x 的线性函数表示的模型，这就是逻辑回归模型。

3、 策略（极大似然法）—— ERM

数据：$D=\{(x1,y1),(x2,y2),(x3,y3)\cdots(x_N,y_N)\};x_i\in R^n,y_i\in\{0,1\}$

模型：$p(y=1|x)=\frac{1}{1+e^{-(\omega^T x+b)}},p(y=0|x)=\frac{1}{1+e^{\omega^T x+b}}$

根据我们的数据和选用的模型，若采取线性回归的平方误差，因为模型的原因，求出来的代价函数是“非凸”函数，所以我们考虑采用极大似然法，此时的代价函数如下：

$$E(\overset{\wedge}{\omega})=\begin{cases}-\ln(h_\omega(x)) & y=1\\ -\ln(1-h_\omega(x)) & y=0\end{cases}$$

当模型概率越接近于 1，其代价就趋近于 0；当模型概率越接近于 0，其代价就趋近于∞。这样我们将 $E(\hat{\omega})$ 化简后得到的函数为“凸函数”如下：

$$E(\hat{\omega}) = -y\ln[p(y=1|x)] - (1-y)\ln[1-p(y=1|x)]$$

则 logistic 的代价函数为：

$$E(\omega,b) = \sum_{i=1}^{N}[-y_i\ln(p(y_i=1|x_i)) - (1-y_i)\ln(1-p(y_i=1|x_i))]$$

因此，下一步采用梯度下降法来求最优解。

二、算法（梯度下降法）

我们由上面分析得到的代价函数为：

$$E(\omega,b) = \sum_{i=1}^{N}[-y_i\ln(p(y_i=1|x_i)) - (1-y_i)\ln(1-p(y_i=1|x_i))]$$

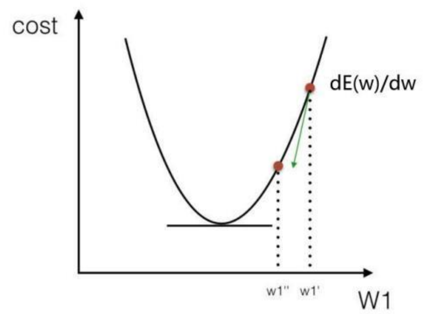
目标通过循环和迭代找到相应的ω1、ω2、b 使代价函数最小或局部最优。梯度下降法的定义为：

$$\hat{\omega}_{(k+1)} = \hat{\omega}_{(k)} - \eta\frac{\partial E(\hat{\omega})}{\partial\hat{\omega}}\Big|_{\hat{\omega}=\hat{\omega}(k)}$$

其中η为步长（学习率），$\frac{\partial E(\hat{\omega})}{\partial\hat{\omega}}$ 为代价函数的导数，最终当上式收敛时，代价函数达到最小值或局部最优。

1、算法原理

以ω含ω1 为例，描述梯度下降法的原理，如图：



$\frac{dE(\omega)}{d\omega}$为曲线在ω1'处切线的斜率，当ω1位于最低点右侧，此时斜率k1>0，由梯度下降公式（$\hat{\omega}_{(k+1)} = \hat{\omega}_{(k)} - \eta \frac{\partial E(\hat{\omega})}{\partial \hat{\omega}}\Big|_{\hat{\omega}=\hat{\omega}(k)}$）可知，ω1将会不断向左移动，直至$\frac{dE(\omega)}{d\omega}$=0，即达到最低点；或小于某一个人为设定的精度ε，此时ω1 在最低点附近震荡。

同理当ω1位于最低点左侧，k<0，ω1会不断向左移动，重复上述过程。

对于η的选取，若η过小，会导致运算时间过长，若过大，则会出现ω1直接跳过最低点，无法趋于最优值。

当ω为多维向量时，结果与上述相同。

2、梯度求导过程

$$E(\hat{\omega}) = \sum_{i=1}^{N}[-y_i \ln(p(y_i=1|x_i)) - (1-y_i)\ln(1-p(y_i=1|x_i))$$

对$E(\hat{\omega})$进行求导，其中$\hat{\omega} = (w_1, w_2, b)$对应题目中的三个参数，$\hat{x}_i = (x_{i1}, x_{i2}, 1)$表示第i个样本，$x_{i1}, x_{i2}$表示样本的两个特征。

$$\begin{aligned} E(\hat{\omega}) &= \sum_{i=1}^{N}[-y_i \ln(p(y_i=1|x_i)) - (1-y_i)\ln(1-p(y_i=1|x_i)) \\ &= \sum_{i=1}^{N}[-y_i \hat{\omega}^T \hat{x}_i + \ln(1+\hat{\omega}^T \hat{x}_i)] \\ &= \sum_{i=1}^{N}[-y_i z_i + \ln(1+e^{z_i})] \qquad （令 z_i = \hat{w}^T \hat{x}_i） \end{aligned}$$

$$\nabla E(\hat{\omega}) = \frac{\partial E}{\partial \omega_1} + \frac{\partial E}{\partial \omega_2} + \frac{\partial E}{\partial b}$$

$$= \sum_{i=1}^{N}[-y_i x_{i1} + \frac{x_{i1}e^{z_i}}{1+e^{z_i}}] + \sum_{i=1}^{N}[-y_i x_{i2} + \frac{x_{i2}e^{z_i}}{1+e^{z_i}}] + \sum_{i=1}^{N}[-y_i + \frac{e^{z_i}}{1+e^{z_i}}]$$

$$= \sum_{i=1}^{N}[-y_i x_{i1} + \frac{x_{i1}}{1+e^{-z_i}}] + \sum_{i=1}^{N}[-y_i x_{i2} + \frac{x_{i2}}{1+e^{-z_i}}] + \sum_{i=1}^{N}[-y_i + \frac{1}{1+e^{-z_i}}]$$

我们可以将上式看作两个矩阵的乘积。

$$x = \begin{bmatrix} x_{11} & x_{12} & 1 \\ \ldots & \ldots & \ldots \\ x_{n1} & x_{n2} & 1 \end{bmatrix}$$

$$D = [\frac{1}{1+e^{-\hat{\omega}\hat{x}_1}} - y_1, \frac{1}{1+e^{-\hat{\omega}\hat{x}_2}} - y_2, \ldots.., \frac{1}{1+e^{-\hat{\omega}\hat{x}_n}} - y_n]$$

x 是 n*3 的矩阵，D 是 1*n 的矩阵，因此，有：

$$\nabla E(\hat{\omega}) = DX = [\frac{\partial E}{\partial \omega_1}, \frac{\partial E}{\partial \omega_2}, \frac{\partial E}{\partial b}]$$

通过 D，x 矩阵相乘可以得到一个 1*3 的矩阵，里面三个数即三个梯度。

3、具体流程步骤

①取初始值$\widehat{\omega}^{(0)} \in \mathbb{R}^{d+1}$，置$k = 0$；

②计算$E\left(\widehat{\omega}^{(k)}\right)$；

③计算梯度$\nabla E\left(\widehat{\omega}^{(k)}\right)$，当$\left\|\nabla E\left(\widehat{\omega}^{(k)}\right)\right\| < \varepsilon$时，令$\widehat{\omega}^{*} = \widehat{\omega}^{(k)}$，

停止迭代；否则，求η_k使$\min\limits_{\eta \geq 0} E\left(\widehat{\omega}^{(k)} + \eta \times \left(-\nabla E\left(\widehat{\omega}^{(k)}\right)\right)\right)$；

④置$\widehat{\omega}^{(k+1)} = \widehat{\omega}^{(k)} + \eta_k \times \left(-\nabla E\left(\widehat{\omega}^{(k)}\right)\right)$，计算$E\left(\widehat{\omega}^{(k+1)}\right)$，

当$\left\|E\left(\widehat{\omega}^{(k+1)}\right) - E\left(\widehat{\omega}^{(k)}\right)\right\| < \varepsilon$或$\left\|\widehat{\omega}^{(k+1)} - \widehat{\omega}^{(k)}\right\| < \varepsilon$时，

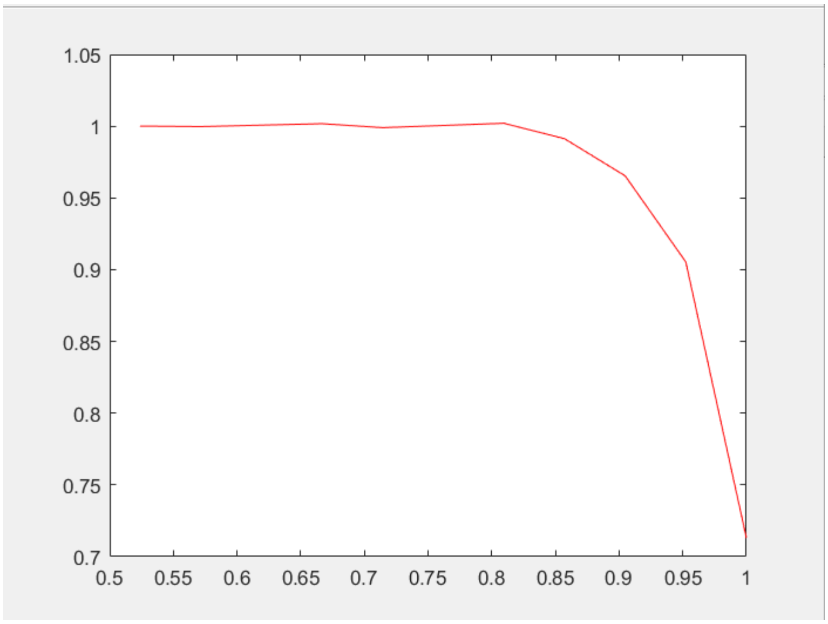
令$\widehat{\omega}^{*} = \widehat{\omega}^{(k)}$，停止迭代；

⑤否则，置$k = k + 1$，转步骤（3）

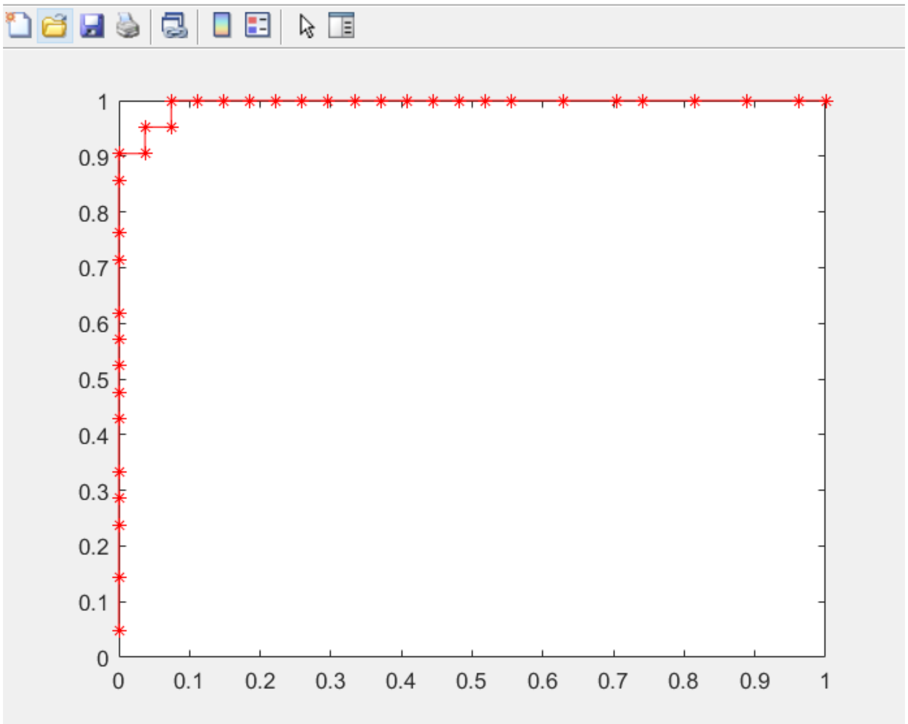
=

三、PR 曲线和 ROC 曲线

p-r 曲线：



Roc 曲线



四、模型训练过程中的收获

学习步长选择

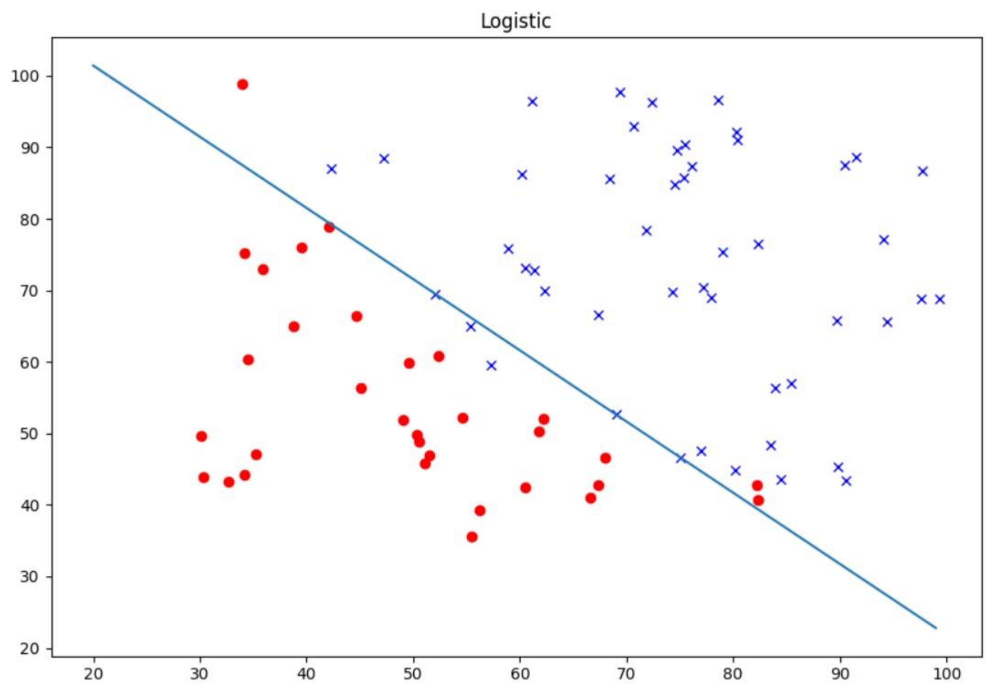
学习步长的选择要适中，既不能过小也不能过大。

步长过大：精度不准，可能会进入无限循环，误差过大

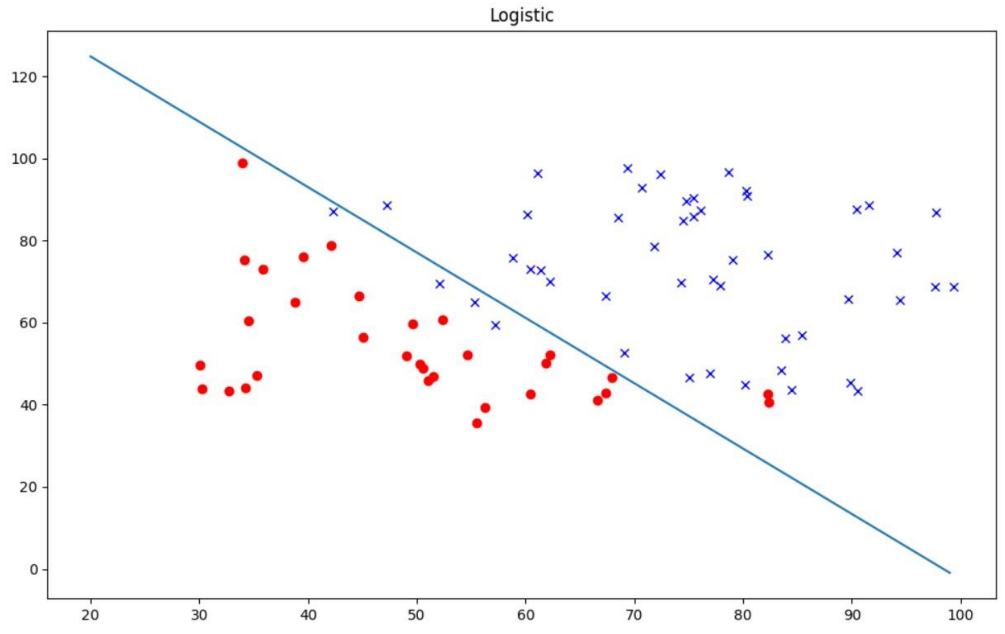
步长过小：运行时间过长，甚至出现得不出结果的情况，并且结果会因判断结束循环的条件而有所不同，可能导致拟合效果不行

例如，在实际代码的运行中，我们选择两个学习步长，a1=0.001、a2=0.0001. 我们将两个步长带入程序中运行，所得结果：

① a1=0.001 时



② a2=0.0001 时



可以看出两个学习步长所得结果相差并不是特别大
但是，a1=0.001 有个致命的缺点，当我们使用 a1=0.001 来计算每个点的 p(yi=1|xi)的值时，会发现数据偏向两个极端，极大的偏向 1 和 0，如图

```
[1.371783590480857e-21, 0.9999999550990699, 1.1212923486021764e-10, 0.43745302765937866, 1.0, 1.0, 2
.331984835257487e-09, 1.0, 1.0, 0.027594536691456603, 1.0, 1.0, 1.0, 4.5707996801279435e-18, 1.0, 1.0, 1.0, 1.0,
5.486550287671253e-13, 0.4327519556111556, 3.24525151970549e-54, 1.0, 0.9998724348790108, 0.9995775157841706,
8.037867778332724e-27, 1.0, 1.0, 3.471322358694825e-17, 0.9999999999999496, 1.0, 0.969299893353901, 1
.0005159201340431e-15, 1.0, 1.0, 1.0, 1.0, 2.0008134552325874e-06, 3.513979970992321e-14, 0.9999999994068647,
0.9999999999996707, 1.0, 1.0, 2.0777321525454264e-51, 1.993086031758904e-24, 5.690853569451237e-25, 0
.16284718766741105, 5.380445184117246e-26, 1.0, 5.408703897584676e-32, 1.0, 1.0293431723956835e-07, 6
.1410082497397355e-50, 5.4751982764943565e-09, 0.9999999999999996, 8.639155693932581e-30, 6.0802768107795025e-12,
5.476336599985653e-28, 0.9999999999988689, 6.30876021556217e-15, 1.137169063165168e-22, 1.0, 0.9999999999999998, 1.0,
 1.0, 0.9999999999999962, 1.1630400601799603e-36, 2.055732208158083e-31, 0.9999999999999984, 2.236096734283849e-56,
1.0, 0.6066307004333837, 7.652340437077604e-47, 4.8397043726568766e-15, 1.0, 0.9999999999999953, 0.9999999999999911,
0.9998437753812879, 1.0, 1.0, 1.0]
```

而使用 0.0001 就不会发生这种情况

```
[0.009807672014879419, 0.9638036347692578, 0.12598288142359854, 0.33729454819177535, 0.7478662201328994, 0
.999456823184814, 0.2982581827604066, 0.9999419209942811, 0.9999803766764439, 0.3634097984532193, 0.9963340917591997,
 0.9806296113845366, 0.9997840198827195, 0.06328463543690414, 0.9992627481776716, 0.9996657395362855, 0
.9999320596309518, 0.9999705014488591, 0.04990893298394953, 0.8081556136253687, 7.88826395362564e-05, 0
.9963402464991158, 0.9197552924616753, 0.8884849023235873, 0.014497887367968306, 0.9967458289840936, 0
.9999981805817997, 0.1851298074885174, 0.35729503365640247, 0.9996655353954946, 0.9109836937394772, 0
.01603320763035042, 0.9998924323624295, 0.9999801577277266, 0.9999567618108498, 0.9996432042996952, 0
.28295199138691673, 0.2662833681421996, 0.41280065051130127, 0.8716229822627043, 0.9961826341272213, 0
.9996658987486018, 0.00013066639371866622, 0.012044070739675778, 0.01612228025247145, 0.15820459473917256, 0
.01576904930829474, 0.9997549950083067, 0.0017091248449830547, 0.9971239661417101, 0.06137127819854772, 0
.00010811395556584272, 0.4303787512862536, 0.8790755658516773, 0.010344354449410132, 0.2261353024247327, 0
.013669829456677126, 0.9780682517785246, 0.015288544783508773, 0.05938976134024226, 0.998953657013751, 0
.9925508663394557, 0.9921223067018383, 0.9996067402578069, 0.9331310796661436, 0.006348777720574884, 0
.01345869561771674, 0.8943970440203675, 4.74928390680687e-05, 0.9999938363339856, 0.7171124802427439, 0
.00026233860342941037, 0.05935891390957956, 0.9994869196415651, 0.991514191817635, 0.8759403636848794, 0
.9318135786215112, 0.9940685042077921, 0.9999902637751416, 0.9965168717298367]
```

这些概率的大小会影响后续画 PR 和 ROC 曲线，因此，上述的 a1、a2 两个学习步长，后者更优

初始参数的选定，对最终结果的大小会产生影响，但是，三个参数总是同比例的增大或者缩小，这对预测结果不会产生影响。

我们发现，用 matlab 可以同时对三个参数进行计算，不需要一个个求导计算。

在进行绘制 pr 曲线时，阈值的选择是任意的，只要合理就行。但是如果依次选择每个样本的预测概率值作为阈值，就可以在绘制 roc 曲线时直接使用数据，不需要再对数据进行划分。

p-r 曲线的阈值：

```
%阈值的步长
basic=0.01;

%从0.01、0.02...0.99依次设定阈值，计算不同阈值下的查准率和查全率
for i=1:99
    values=pridict_function(theta0, x_test, y_test, basic*i,m);
```

r-c 曲线的阈值：

```
for i=1:lengthy
    bound=X(i,5);%依次将样本的预测概率值作为阈值
```