# 《机器学习》课程系列

逻辑回归*

武汉纺织大学数学与计算机学院

杜小勤

2020/04/11

## Contents

**1** 从线性回归到二分类逻辑回归：**Sigmoid** 函数 **2**

**2** 二分类模型的似然函数与损失函数 **4**

**3** 二分类模型中梯度的计算 **7**

**4** 代理损失函数 **8**

**5** **Logistic** 分布 **8**

**6** **Sigmoid** 函数与 **Tanh** 函数 **10**

**7** 对数几率与线性决策面 **11**

**8** 从二分类到多分类：**Sigmoid** 与 **Softmax** 函数 **12**

**9** 多分类模型的似然函数与损失函数 **13**

**10** 多分类模型中梯度的计算 **15**

*本系列文档属于讲义性质，仅用于学习目的。Last updated on: April 23, 2020。

**11** 神经网络的视角 **16**

**12** 广义线性模型与指数族分布 **17**

12.1 广义线性模型 . . . . . 17

12.2 指数族分布 . . . . . 19

12.3 二分类 (伯努利分布) 的指数族形式 . . . . . 19

12.4 K 分类 (One-Hot 形式) 的指数族形式 . . . . . 20

12.5 高斯分布的指数族形式 . . . . . 22

**13** 逻辑回归与高斯朴素贝叶斯 **23**

**14** 附录 **26**

14.1 损失函数的绘制 . . . . . 26

14.2 Logistic 函数的绘制 . . . . . 27

14.3 Sigmoid 与 Tanh 函数的绘制 . . . . . 27

14.4 逻辑回归实验 . . . . . 29

**15** 参考文献 **36**

# 1 从线性回归到二分类逻辑回归：Sigmoid 函数

回归（Regression）问题指的是在给定变量 $\boldsymbol{x}$ 的情况下，预测连续因变量 $y = f(\boldsymbol{x})$ 的值[1]。在回归模型中，线性回归（Linear Regression）模型具有形式简单、易于建模且易于理论分析的特点。注意，此处的线性指的是函数 $y = f(\boldsymbol{x})$ 关于模型参数 $\boldsymbol{w}$ 是线性的，而函数 $y = f(\boldsymbol{x})$ 关于自变量 $\boldsymbol{x}$ 未必是线性的。它的定义如下：

$$y = f(\boldsymbol{x}; \boldsymbol{w}) = \sum_{i=1}^{M} w_i \phi_i(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) \tag{1}$$

其中，$M$ 表示模型中参数的个数，$\phi_j(\boldsymbol{x})$ 被称为基函数（Basis Function）。该模型被称为线性基函数模型（Linear Basis Function Model），它表示了一大类线性回归模型。

[1] 当然，回归问题也包括同时对多个函数值进行预测的情况。

线性回归模型的最简单形式是，函数 $f(\boldsymbol{x})$ 是输入变量 $\boldsymbol{x}$ 的属性 $x_i$ 的线性组合：

$$y = f(\boldsymbol{x}; \boldsymbol{w}) = \sum_{i=1}^{M} w_i x_i = \boldsymbol{w}^T \boldsymbol{x} \tag{2}$$

从概率的角度看，线性回归模型估算的是连续因变量 $y = f(\boldsymbol{x})$ 的条件期望：

$$\mathbb{E}(y|\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} \tag{3}$$

如果因变量不是一个连续的数值，而是一个只有二值的离散数值，且要求模型在给出预测结果的同时，也能够同时给出每个预测结果发生的概率，那么上述线性回归模型不再适用。此时，所对应的问题被称为二分类问题，能够解决该问题的模型被称为二分类模型。

对于二值离散预测，为讨论方便，不妨设 $y = \{c_1, c_2\}$，那么 $p(y = c_1|\boldsymbol{x})$ 表示 $y = c_1$ 的概率，则 $y = c_2$ 的概率为 $p(y = c_2|\boldsymbol{x}) = 1 - p(y = c_1|\boldsymbol{x})$。于是，需要建立一个关于观察样本 $\boldsymbol{x}$ 的预测模型，模型 $f(\boldsymbol{x}; \boldsymbol{w})$ 的输出是关于 $y$ 的条件概率 $p(y = c_1|\boldsymbol{x})$，即：

$$f(\boldsymbol{x}; \boldsymbol{w}) = \mathbb{E}(y = c_1|\boldsymbol{x}) = p(y = c_1|\boldsymbol{x}) \tag{4}$$

现在的问题是，能否像最简形式的线性回归 (公式 (2)) 那样，利用 $\boldsymbol{w}^T \boldsymbol{x}$ 进行预测呢？然而，在线性回归模型中，$f(\boldsymbol{x}; \boldsymbol{w})$ 的值域是 $(-\infty, +\infty)$，而二分类模型 (公式 (4)) 所需要的值域为 $[0, 1]$，二者并不直接匹配。

显然，需要在 $\boldsymbol{w}^T \boldsymbol{x}$ 与 $p(y = c_1|\boldsymbol{x})$ 之间进行值域变换。对数函数 $\log(\cdot)$ 是选项之一[2]，有诸多优良特性，它也能够在这两者之间建立变换关系——其定义域 $(0, +\infty)$ 上对应的值域是 $(-\infty, +\infty)$。为应用对数函数 $\log(\cdot)$，可令其自变量为 $\frac{p(y=c_1|\boldsymbol{x})}{1-p(y=c_1|\boldsymbol{x})}$：由于 $p(y = c_1|\boldsymbol{x}) \in [0, 1]$，则 $\frac{p(y=c_1|\boldsymbol{x})}{1-p(y=c_1|\boldsymbol{x})}$ 在范围 $[0, +\infty)$ 内，正好符合要求[3]。

上述变换被称为 Logit 变换，而 $\frac{p(y=c_1|\boldsymbol{x})}{1-p(y=c_1|\boldsymbol{x})}$ 被称为几率，Logit 函数也被称为对数几率函数。

于是，得到如下的模型：

$$\log \frac{p(y = c_1|\boldsymbol{x})}{1 - p(y = c_1|\boldsymbol{x})} = \boldsymbol{w}^T \boldsymbol{x} \tag{5}$$

[2]没有显式写出对数的底，如无特别说明，一般指自然对数。另外，也可以依据上下文做出判断。

[3]注意，为应用到 $\log(\cdot)$ 函数，需要在两端点处进行特别处理。

继续变形，得到：

$$p(y=c_1|\boldsymbol{x}) = \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}} \tag{6}$$

从而得到了二分类逻辑回归模型 (Binary Logistic Regression)。令 $z=\boldsymbol{w}^T\boldsymbol{x}$，则上式右边为：

$$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z} \tag{7}$$

这就是 Sigmoid 函数。一般地，如前所述，$z$ 使用权值与特征向量的点积形式表示：

$$z = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n = \sum_{i=0}^{n} w_ix_i = \boldsymbol{w}^T\boldsymbol{x} \tag{8}$$

其中，$x_i$ 表示样本 $\boldsymbol{x}$ 的第 $i$ 个分量，$w_i$ 表示该分量所对应的权重系数。从 $z$ 的表达式可以看出，为了计算方便，令偏置 $w_0$ 对应的分量 $x_0=1$，从而可以使用向量的点积形式来表示 $z$。

从变换效果来看，Sigmoid 函数的作用是，将线性回归 $\boldsymbol{w}^T\boldsymbol{x}$ 从区间 $(-\infty,+\infty)$ 变换到了区间 $(0,1)$ 上，从而实现了二分类的概率表示。因此，Logit 函数与 Sigmoid 函数之间是一种变换与反变换的关系。

Sigmoid 函数的一个优点是，其导数形式较为简单：

$$\sigma'(z) = \sigma(z)\,(1-\sigma(z)) \tag{9}$$

可以看出，导数公式包含 Sigmoid 函数自身。

## 2 二分类模型的似然函数与损失函数

现在，假设使用 Sigmoid 函数进行二分类预测，其预测模型可表示为：

$$\begin{aligned} &p(y=c_1|\boldsymbol{x}) = \sigma(z) = \tfrac{1}{1+e^{-z}} \\ &p(y=c_2|\boldsymbol{x}) = 1 - p(y=c_1|\boldsymbol{x}) = 1-\sigma(z) = \tfrac{e^{-z}}{1+e^{-z}} = \tfrac{1}{1+e^{z}} \end{aligned} \tag{10}$$

为了便于表达公式，可令 $c_1=1$，$c_2=0$，即将 $y$ 的取值限定于 $\{0,1\}$，则上式合并为：

$$p(y|\boldsymbol{x}) = \sigma(z)^y(1-\sigma(z))^{1-y} \tag{11}$$

其中，$z=\boldsymbol{w}^T\boldsymbol{x}$，$y=\{0,1\}$。

给定一个数据集：

$$\boldsymbol{T} = \{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\ldots,(\boldsymbol{x}_N,y_N)\} \tag{12}$$

其中，$N$ 表示数据集中样本点的个数。于是，可以写出二分类逻辑回归模型的似然函数：

$$\mathcal{L}(\boldsymbol{w}) = \prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i;\boldsymbol{w}) \tag{13}$$

其中，$p(y_i|\boldsymbol{x}_i;\boldsymbol{w})$ 表示第 $i$ 个样本点的条件概率，具有公式 (11) 形式；$\boldsymbol{w}$ 表示整个数据集的逻辑回归参数。将公式 (11) 代入，可得：

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}) &= \prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i;\boldsymbol{w}) \\ &= \prod_{i=1}^{N} \sigma(z_i)^{y_i}(1-\sigma(z_i))^{1-y_i}\end{aligned} \tag{14}$$

其中，$z_i = \boldsymbol{w}^T\boldsymbol{x_i}$。对似然函数取对数，得到对数似然函数：

$$\log \mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{N} \log p(y_i|\boldsymbol{x}_i;\boldsymbol{w}) = \sum_{i=1}^{N} [y_i \log \sigma(z_i) + (1-y_i)\log(1-\sigma(z_i))] \tag{15}$$

于是，损失函数可以使用如下形式的负对数似然函数来表示：

$$\begin{aligned}L(\boldsymbol{w}) &= -\frac{1}{N}\log \mathcal{L}(\boldsymbol{w}) \\ &= -\frac{1}{N}\sum_{i=1}^{N} [y_i \log \sigma(z_i) + (1-y_i)\log(1-\sigma(z_i))]\end{aligned} \tag{16}$$

上式就是二分类逻辑回归模型的交叉熵损失函数。其中，$N$ 表示样本点个数，$y_i$ 表示样本点 $\boldsymbol{x_i}$ 的理想输出，取值为 $\{0,1\}$，$\sigma(z_i)$ 表示样本点 $\boldsymbol{x_i}$ 的实际输出，$z_i = \boldsymbol{w}^T\boldsymbol{x_i}$。

损失函数可以变换为其它的形式。例如，将 $\sigma(z_i) = \frac{1}{1+e^{-z_i}}$ 代入公式 (16)，得到：

$$\begin{aligned}L(\boldsymbol{w}) &= -\frac{1}{N}\sum_{i=1}^{N} [y_i \log \sigma(z_i) + (1-y_i)\log(1-\sigma(z_i))] \\ &= -\frac{1}{N}\sum_{i=1}^{N} [y_i \log \frac{1}{1+e^{-z_i}} + (1-y_i)\log(\frac{e^{-z_i}}{1+e^{-z_i}})] \\ &= -\frac{1}{N}\sum_{i=1}^{N} [\log \frac{1}{1+e^{z_i}} + y_i \log \frac{1}{e^{-z_i}}]\end{aligned} \tag{17}$$

令 $\log = \ln$，则可进一步化简：

$$\begin{aligned}L(\boldsymbol{w}) &= \frac{1}{N}\sum_{i=1}^{N} [-y_i z_i + \ln(1+e^{z_i})] \\ &= \frac{1}{N}\sum_{i=1}^{N} [-y_i \boldsymbol{w}^T\boldsymbol{x_i} + \ln(1+e^{\boldsymbol{w}^T\boldsymbol{x_i}})]\end{aligned} \tag{18}$$

公式的最后一行，将 $z_i = \boldsymbol{w}^T\boldsymbol{x_i}$ 代入而得。实际上，在公式 (17) 的推导过程中，也可以直接利用对数几率 (Log Odds) 或 Logit 函数[4]：

$$\text{logit}\left(p(y=1|\boldsymbol{x})\right) = \log\left(\frac{p(y=1|\boldsymbol{x})}{1-p(y=1|\boldsymbol{x})}\right) = \log\frac{\sigma(z)}{1-\sigma(z)} = \boldsymbol{w}^T\boldsymbol{x} \tag{19}$$

得到相同的结果。

另外，观察到公式 (10) 中 $z$ 的正负性特点，如果令 $y$ 的取值为 $\{-1,1\}$，则公式 (10) 可重新表示为：

$$\begin{aligned} &p(y=1|\boldsymbol{x}) = \sigma(z) = \tfrac{1}{1+e^{-z}} \\ &p(y=-1|\boldsymbol{x}) = 1-p(y=1|\boldsymbol{x}) = 1-\sigma(z) = \tfrac{e^{-z}}{1+e^{-z}} = \tfrac{1}{1+e^{z}} \end{aligned} \tag{20}$$

直接观察上式，得到如下关系：

$$\sigma(-z) = 1-\sigma(z) \tag{21}$$

于是，利用该关系式，可以重新将公式 (20) 合并为：

$$p(y|\boldsymbol{x}) = \sigma(yz) \tag{22}$$

再次写出数据集的似然函数，可得：

$$\begin{aligned} \mathcal{L}(\boldsymbol{w}) &= \prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i;\boldsymbol{w}) \\ &= \prod_{i=1}^{N} \sigma(y_i z_i) \\ &= \prod_{i=1}^{N} \frac{1}{1+e^{-y_i z_i}} \end{aligned} \tag{23}$$

相应的损失函数为：

$$\begin{aligned} L(\boldsymbol{w}) &= -\frac{1}{N}\log\mathcal{L}(\boldsymbol{w}) \\ &= \frac{1}{N}\sum_{i=1}^{N}\log\left(1+e^{-y_i z_i}\right) \\ &= \frac{1}{N}\sum_{i=1}^{N}\log\left(1+e^{-y_i\boldsymbol{w}^T\boldsymbol{x_i}}\right) \end{aligned} \tag{24}$$

注意，上式中，$y_i$ 的取值为 $\{-1,1\}$。

[4]如果事件发生的概率是 $p$，那么该事件的几率为 $\frac{p}{1-p}$，该事件的对数几率是 $\text{logit}(p) = \log\frac{p}{1-p}$。

# 3 二分类模型中梯度的计算

为描述方便，令 $\log = \ln$。[5]针对公式 (16)，将 $z_i = \boldsymbol{w}^T\boldsymbol{x_i}$ 代入并求解偏导数：

$$\begin{aligned}\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} &= -\frac{1}{N}\sum_{i=1}^{N}\left[y_i\frac{\sigma(\boldsymbol{w}^T\boldsymbol{x_i})(1-\sigma(\boldsymbol{w}^T\boldsymbol{x_i}))\boldsymbol{x_i}}{\sigma(\boldsymbol{w}^T\boldsymbol{x_i})} + (1-y_i)\frac{-\sigma(\boldsymbol{w}^T\boldsymbol{x_i})(1-\sigma(\boldsymbol{w}^T\boldsymbol{x_i}))\boldsymbol{x_i}}{1-\sigma(\boldsymbol{w}^T\boldsymbol{x_i})}\right] \\ &= -\frac{1}{N}\sum_{i=1}^{N}\left[y_i(1-\sigma(\boldsymbol{w}^T\boldsymbol{x_i}))\boldsymbol{x_i} - (1-y_i)\sigma(\boldsymbol{w}^T\boldsymbol{x_i})\boldsymbol{x_i}\right] \\ &= -\frac{1}{N}\sum_{i=1}^{N}\left[y_i - \sigma(\boldsymbol{w}^T\boldsymbol{x_i})\right]\boldsymbol{x_i}\end{aligned} \tag{25}$$

其中，$y_i$ 表示第 $i$ 个样本的理想输出，$\sigma(\boldsymbol{w}^T\boldsymbol{x_i})$ 表示第 $i$ 个样本的实际输出，$\boldsymbol{x_i}$ 表示第 $i$ 个样本的输入。

实际上，从简化的公式 (18) 出发，可以较简便地得出相同的结论：

$$\begin{aligned}\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} &= \frac{1}{N}\sum_{i=1}^{N}\left[-y_i\boldsymbol{x_i} + \frac{e^{\boldsymbol{w}^T\boldsymbol{x_i}}}{1+e^{\boldsymbol{w}^T\boldsymbol{x_i}}}\boldsymbol{x_i}\right] \\ &= \frac{1}{N}\sum_{i=1}^{N}\left[-y_i\boldsymbol{x_i} + \sigma(\boldsymbol{w}^T\boldsymbol{x_i})\boldsymbol{x_i}\right] \\ &= -\frac{1}{N}\sum_{i=1}^{N}\left[y_i - \sigma(\boldsymbol{w}^T\boldsymbol{x_i})\right]\boldsymbol{x_i}\end{aligned} \tag{26}$$

其中，$y_i$ 的取值为 $\{0, 1\}$。

针对公式 (24)，求解偏导数：

$$\begin{aligned}\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} &= \frac{1}{N}\sum_{i=1}^{N} -\frac{e^{-y_i\boldsymbol{w}^T\boldsymbol{x_i}}}{1+e^{-y_i\boldsymbol{w}^T\boldsymbol{x_i}}}y_i\boldsymbol{x_i} \\ &= -\frac{1}{N}\sum_{i=1}^{N}\sigma(-y_i\boldsymbol{w}^T\boldsymbol{x_i})y_i\boldsymbol{x_i}\end{aligned} \tag{27}$$

其中，$y_i$ 的取值为 $\{-1, 1\}$。

[5]若 $y = \log_a x$，则 $y' = \frac{1}{x\ln a}$，这表明 log 与 ln 的导数只相差一个常数因子。

# 4 代理损失函数

在二分类学习器中，若 $y$ 的取值为 $\{-1,1\}$，则支持向量机、逻辑回归、AdaBoost 方法的损失函数分别具有如下形式：

$$
\begin{aligned}
L(\boldsymbol{w}) &= \frac{1}{N}\sum_{i=1}^{N}\left[1-y_i\boldsymbol{w}^T\boldsymbol{x_i}\right]_+ + \lambda\|\boldsymbol{w}\|^2 = \frac{1}{N}\sum_{i=1}^{N}\left[1-y_i f\left(\boldsymbol{x_i}\right)\right]_+ + \lambda\|\boldsymbol{w}\|^2 \\
L(\boldsymbol{w}) &= \frac{1}{N}\sum_{i=1}^{N}\log\left(1+e^{-y_i\boldsymbol{w}^T\boldsymbol{x_i}}\right) = \frac{1}{N}\sum_{i=1}^{N}\log\left(1+e^{-y_i f(\boldsymbol{x_i})}\right) \\
L(\boldsymbol{w}) &= \frac{1}{N}\sum_{i=1}^{N}e^{-y_i\sum\limits_{m=1}^{M}\alpha_m G_m(\boldsymbol{x_i})} = \frac{1}{N}\sum_{i=1}^{N}e^{-y_i f(\boldsymbol{x_i})}
\end{aligned}
\tag{28}
$$

其中，支持向量机的损失函数中，含有正则项 $\|\boldsymbol{w}\|^2$；前 2 项公式中，$f(\boldsymbol{x})=\boldsymbol{w}^T\boldsymbol{x}$；AdaBoost 方法中，$f(\boldsymbol{x})=\sum\limits_{m=1}^{M}\alpha_m G_m(\boldsymbol{x})$，它是 AdaBoost 的最终分类器（加法模型），$G_m(\boldsymbol{x})$ 为基本分类器，$\alpha_m$ 为相应的权重。

下面从单样本的角度考察这些损失函数：

$$
\begin{aligned}
l(\boldsymbol{w}) &= \left[1-yf(\boldsymbol{x})\right]_+ \\
l(\boldsymbol{w}) &= \log\left(1+e^{-yf(\boldsymbol{x})}\right) \\
l(\boldsymbol{w}) &= e^{-yf(\boldsymbol{x})}
\end{aligned}
\tag{29}
$$

这 3 种损失函数被分别称为合页损失函数、逻辑回归损失函数与指数损失函数，它们都是 0-1 损失函数的上界，具有非常相似的曲线形状[6]，如图4-1所示。

因此，从损失函数的角度来看，支持向量机、逻辑回归与 AdaBoost 方法使用了不同的代理损失函数 (Surrogate Loss Function) 来定义经验风险函数或结构风险函数，用来表示分类损失，从而实现了分类任务。支持向量机本身具有正则项，而逻辑回归没有正则项，但是可以添加正则项。AdaBoost 方法也没有正则项，可以通过早期停止 (Early Stopping) 技术起到正则项的作用。

# 5 Logistic 分布

Logistic 分布 (Logistic Distribution) 指的是连续随机变量 $X$ 服从如下的分布函数：

$$
F(x)=p(X\leqslant x)=\frac{1}{1+e^{-(x-\mu)/\gamma}} \tag{30}
$$

[6]注意：此处，逻辑回归损失函数的 log 以 2 为底。绘制程序参见附录14.1。

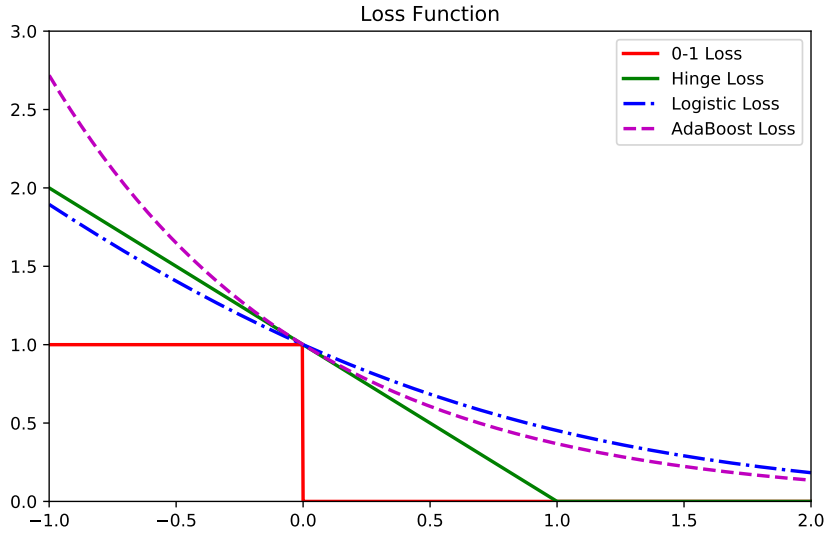



图 4-1: 代理损失函数

它是 Sigmoid 函数的更一般形式。其中，$\mu$ 决定了曲线的位置，$\gamma$ 决定了曲线的形状。其密度函数为：

$$f(x) = F'(x) = \frac{e^{-(x-\mu)/\gamma}}{\gamma\left(1+e^{-(x-\mu)/\gamma}\right)^2} \tag{31}$$

两者的关系为：

$$f(x) = \frac{1}{\gamma}F(x)\left[1-F(x)\right] \tag{32}$$

曲线如图5-2所示[7]。

分析 Logistic 分布 (即 $F(x)$ 函数)，可以发现，当 $x \to +\infty$ 时，$F(x) \to 1$；而当 $x \to -\infty$ 时，$F(x) \to 0$，并且当 $x = \mu$ 时，$F(x) = 0.5$，这是一种较为理想的分类决策函数。例如，若 $y = \{c_1, c_2\}$，则当 $F(x) \geqslant 0.5$ 时，可令 $y = c_1$；否则，$y = c_2$。另外，也可以把 $F(x)$ 看作 $y = c_1$ 或 $y = c_2$ 的概率。因此，在进行决策时，既可以把 $F(x)$ 当做条件概率，也可以把它当作决策函数。

为了应用 $F(\cdot)$ 函数进行二分类，可引入输入特征向量 $\boldsymbol{x}$ 及其权值向量 $\boldsymbol{w}$，令 $z = \boldsymbol{w}^T\boldsymbol{x}$ 作为分类器的基本运算，并且令公式 (30) 中的 $(x-\mu)/\gamma = z$，从而得到如下的公式：

$$p(y = c_1|\boldsymbol{x}) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}} \tag{33}$$

[7]绘制程序参见附录14.2

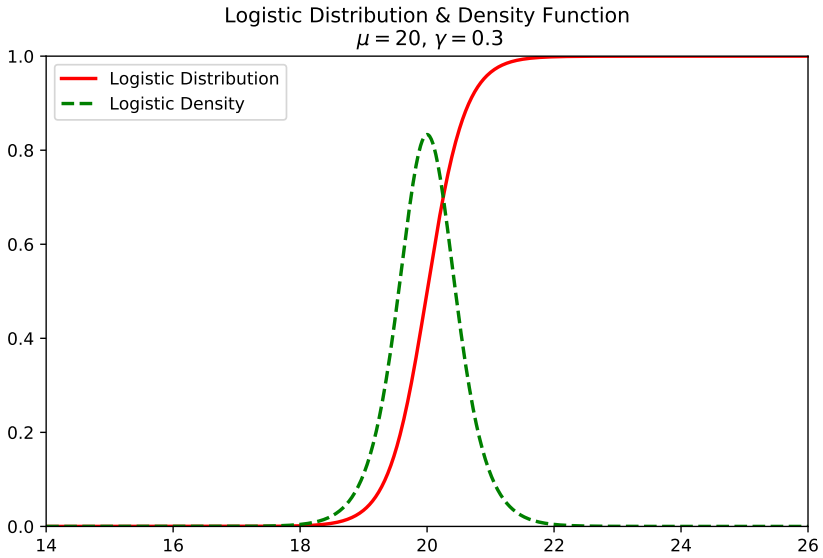



图 5-2: Logistic 分布函数与密度函数

这就是前述的二分类逻辑回归决策模型。

需要注意的是，虽然公式 (33) 没有显式的位置控制变量 $\mu$ 和形状控制变量 $\gamma$，但是，由于 $\boldsymbol{w}$ 具有可学习性，这使得 $F(\cdot)$ 函数具有调节自身位置与形状的能力。

# 6 Sigmoid 函数与 Tanh 函数

Tanh 函数被称为双曲正切函数：

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{34}$$

其导数与原函数之间也具有较为简单的关系：

$$\tanh'(z) = 1 - [\tanh(z)]^2 \tag{35}$$

Sigmoid 函数与 Tanh 函数之间具有如下关系：

$$\tanh(z) = 2\sigma(2z) - 1 \tag{36}$$

或：

$$\sigma(z) = \frac{1 + \tanh(\frac{z}{2})}{2} \tag{37}$$

它们的曲线如图6-3所示[8]。两者都可以作为神经网络中神经元的激活函数。

[8]绘制程序参见附录14.3

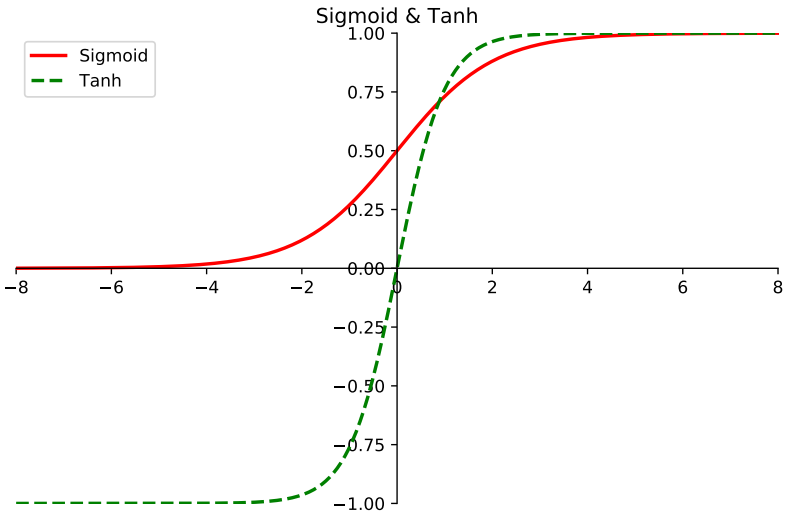



图 6-3: Sigmoid 函数与 Tanh 函数

# 7 对数几率与线性决策面

一个事件的几率 (Odds) 是指该事件发生的概率与事件不发生概率的比值：

$$\frac{p}{1-p} \tag{38}$$

其中 $p$ 表示事件发生的概率。定义事件的对数几率（Log Odds）或 Logit 函数为：

$$\text{logit}(p) = \log \frac{p}{1-p} \tag{39}$$

在此定义下，逻辑回归的对数几率为：

$$\log \frac{p(y=c_1|\boldsymbol{x})}{1-p(y=c_1|\boldsymbol{x})} = \boldsymbol{w}^T\boldsymbol{x} \tag{40}$$

这表明，逻辑回归模型中，对数几率是关于 $\boldsymbol{x}$ 的线性函数或线性模型。

如果 $p(y=c_1|\boldsymbol{x}) \geqslant (1-p(y=c_1|\boldsymbol{x}))$(即 $p(y=c_2|\boldsymbol{x})$)，相应的对数几率值 $\boldsymbol{w}^T\boldsymbol{x} \geqslant 0$，则可令实例 $\boldsymbol{x}$ 所对应的类别为 $y=c_1$；否则，可令 $y=c_2$。也就是说，决策时，将实例 $\boldsymbol{x}$ 分到最大概率所对应的类别。因此，$\boldsymbol{w}^T\boldsymbol{x}$ 形成了一个线性决策面。

由此，可以定义如下的线性决策函数：

$$y = f(\boldsymbol{x}) = \begin{cases} c_1 & \boldsymbol{w}^T\boldsymbol{x} \geqslant 0 \\ c_2 & else \end{cases} \tag{41}$$

另一方面，如果直接将对数几率代入公式 $p(y=c_1|\boldsymbol{x})$ 中，则得到了实例 $\boldsymbol{x}$ 被划分为类别 $y=c_1$ 的概率。从上面的分析可以看出，决策函数与条件概率之间是可以相互转换的。

实际上，可以使用更加通用的归一化方法[9]，并利用 $p$ 与 $1-p$ 之间的几率定义：$\frac{p}{1-p}=e^{\boldsymbol{w}^T\boldsymbol{x}}$，将决策函数转换为条件概率：

$$\begin{aligned} p(y=c_1|\boldsymbol{x}) = p &= \frac{p}{p+(1-p)} \\ &= \frac{(1-p)e^{\boldsymbol{w}^T\boldsymbol{x}}}{(1-p)e^{\boldsymbol{w}^T\boldsymbol{x}}+(1-p)} \\ &= \frac{e^{\boldsymbol{w}^T\boldsymbol{x}}}{1+e^{\boldsymbol{w}^T\boldsymbol{x}}} = \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}} \end{aligned} \tag{42}$$

# 8 从二分类到多分类：Sigmoid 与 Softmax 函数

前面的推导，使用了同一个权值向量 $\boldsymbol{w}$ 来表示 $p$ 和 $1-p$，或者说，利用了 $p$ 与 $1-p$ 之间的耦合关系来表达 2 个概率——虽然有 2 个概率，但自由度却只有 1 个。下面的推导，从另一个角度来建模类别 $y=c_1$ 的概率 $p$ 和类别 $y=c_2$ 的概率 $1-p$，并由此导出 Softmax 函数。

设 $\boldsymbol{w}_0$ 表示类别 $y=c_1$ 的权值参数，$\boldsymbol{w}_1$ 表示类别 $y=c_2$ 的权值参数。不论是二分类，还是多分类，分类决策函数将实例 $\boldsymbol{x}$ 划分为所有分类概率中最大分类概率所对应的类别。据此，根据前述的通用归一化公式（即将每个类别的权值输入 $\boldsymbol{w}_i^T\boldsymbol{x}$ 转换成各自的概率），可得：

$$\begin{aligned} p(y=c_1|\boldsymbol{x}) = p &= \frac{e^{\boldsymbol{w}_0^T\boldsymbol{x}}}{e^{\boldsymbol{w}_0^T\boldsymbol{x}}+e^{\boldsymbol{w}_1^T\boldsymbol{x}}} \\ &= \frac{1}{1+e^{(\boldsymbol{w}_1-\boldsymbol{w}_0)^T\boldsymbol{x}}} \end{aligned} \tag{43}$$

如果令 $\boldsymbol{w}=\boldsymbol{w}_1-\boldsymbol{w}_0$，即只使用一个权值 $\boldsymbol{w}$ 来同时建模二分类，那么上式将还原为：

$$p(y=c_1|\boldsymbol{x}) = p = \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}} \tag{44}$$

该式就是用于二分类的条件概率公式或决策函数，即 Sigmoid 函数。

[9]归一化方法使用非常广泛，常见于诸如 Softmax 函数、赌轮、贝叶斯规则等公式中，它是一种通用的概率表达与转换方法。后文将表明，Sigmoid 函数是 Softmax 函数的特例，当然也能够使用归一化方法获得。

归一化方法可以直接推广到 $K$ 分类。设每个类别所对应的权值分别为 $\boldsymbol{w}_j$，$j=1,\ldots,K$，则第 $j$ 个类别的概率为：

$$p(y=c_j|\boldsymbol{x})=\frac{e^{\boldsymbol{w}_j^T\boldsymbol{x}}}{\sum\limits_{k=1}^{K}e^{\boldsymbol{w}_k^T\boldsymbol{x}}} \tag{45}$$

上式就是用于多分类的 Softmax 函数，Sigmoid 函数是 Softmax 函数的特例。

理论上，$K$ 个权值向量 $\boldsymbol{w}_j(j=1,\ldots,K)$ 与 $K-1$ 个权值向量 $\boldsymbol{w}_j(j=2,\ldots,K$，即假定权值向量 $\boldsymbol{w}_1$ 是冗余的)[10]的预测结果是一致的：

$$\begin{aligned}p(y=c_j|\boldsymbol{x})&=\frac{e^{\boldsymbol{w}_j^T\boldsymbol{x}}}{\sum\limits_{k=1}^{K}e^{\boldsymbol{w}_k^T\boldsymbol{x}}}=\frac{e^{\boldsymbol{w}_j^T\boldsymbol{x}}e^{-\boldsymbol{w}_1^T\boldsymbol{x}}}{\left(\sum\limits_{k=1}^{K}e^{\boldsymbol{w}_k^T\boldsymbol{x}}\right)e^{-\boldsymbol{w}_1^T\boldsymbol{x}}}\\&=\frac{e^{(\boldsymbol{w}_j-\boldsymbol{w}_1)^T\boldsymbol{x}}}{1+\sum\limits_{k=2}^{K}e^{(\boldsymbol{w}_k-\boldsymbol{w}_1)^T\boldsymbol{x}}}\\&=\frac{e^{\boldsymbol{\theta}_j^T\boldsymbol{x}}}{1+\sum\limits_{k=2}^{K}e^{\boldsymbol{\theta}_k^T\boldsymbol{x}}}\end{aligned} \tag{46}$$

上式表明，消除或增加冗余权值向量，不会改变预测结果。但是，在实际应用中，$K$ 个权值向量含有冗余参数，将会有更多的参数参与预测，会增加过拟合的风险。在神经网络中，当输出层为 Softmax 层时，每个类别都有自己的权值，就属于这种情况。通过添加正则项，起到抑制过拟合的作用。

# 9 多分类模型的似然函数与损失函数

现在，假设使用 Softmax 函数进行 $K$ 分类预测，其预测模型可表示为：

$$p(y=c_j|\boldsymbol{x};\boldsymbol{w_j},\boldsymbol{W})=\frac{e^{\boldsymbol{w}_j^T\boldsymbol{x}}}{\sum\limits_{k=1}^{K}e^{\boldsymbol{w}_k^T\boldsymbol{x}}}=\sigma(\boldsymbol{w_j}^T\boldsymbol{x};\boldsymbol{W})=\sigma(\boldsymbol{w_j}^T\boldsymbol{x}) \tag{47}$$

其中，每个分类对应一个权值参数向量 $\boldsymbol{w}_j$，它们形成一个权值矩阵 $\boldsymbol{W}$，仍然采用函数 $\sigma(\cdot)$ 来表示 Softmax 函数。为简便，从 $\sigma(\cdot)$ 符号中去掉了 $\boldsymbol{W}$。

在二分类中，$y$ 的取值是 $\{c_1,c_2\}$。在 $K$ 分类中，$y$ 的取值为 $\{c_1,c_2,\ldots,c_K\}$，其中 $K$ 是类别个数。与二分类情形一样，为了便于表达概率，一般令 $y=\{0,1\}^K$，

[10]此处，为讨论方便，假定权值向量 $\boldsymbol{w}_1$ 是冗余的。实际上，任一权值向量均可。

且 $\sum_{k=1}^{K} y^{(k)} = 1$，即把 $y$ 表达成 One-Hot 形式的向量。例如，在手写体数字识别问题中，$K = 10$，数字 3 和 4 被分别编码成 $y = (0,0,0,1,0,0,0,0,0,0)$ 和 $y = (0,0,0,0,1,0,0,0,0,0)$，其余数字的编码，依此类推。

于是，在上述记号表示下，$K$ 分类公式 (47) 合并为：

$$p(y|\boldsymbol{x};\boldsymbol{W}) = \prod_{k=1}^{K} \sigma(\boldsymbol{w_k}^T \boldsymbol{x})^{y^{(k)}} \tag{48}$$

其中，$y^{(k)}$ 表示 One-Hot 向量 $y$ 的第 $k$ 个分量[11]。

给定一个数据集：

$$\boldsymbol{T} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\} \tag{49}$$

其中，$N$ 表示数据集中样本点的个数。于是，可以写出多分类逻辑回归模型的似然函数：

$$\mathcal{L}(\boldsymbol{W}) = \prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i;\boldsymbol{W}) \tag{50}$$

其中，$p(y_i|\boldsymbol{x}_i;\boldsymbol{W})$ 表示第 $i$ 个样本点的条件概率，具有公式 (48) 形式；$\boldsymbol{W}$ 表示整个数据集的逻辑回归参数。将公式 (48) 代入，可得：

$$\begin{aligned} \mathcal{L}(\boldsymbol{W}) &= \prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i;\boldsymbol{W}) \\ &= \prod_{i=1}^{N}\prod_{k=1}^{K} \sigma(\boldsymbol{w_k}^T \boldsymbol{x_i})^{y_i^{(k)}} \end{aligned} \tag{51}$$

对似然函数取对数，得到对数似然函数：

$$\log \mathcal{L}(\boldsymbol{W}) = \sum_{i=1}^{N} \log p(y_i|\boldsymbol{x}_i;\boldsymbol{W}) = \sum_{i=1}^{N}\sum_{k=1}^{K} y_i^{(k)} \log \sigma(\boldsymbol{w_k}^T \boldsymbol{x_i}) \tag{52}$$

于是，损失函数可以使用如下形式的负对数似然函数来表示：

$$\begin{aligned} L(\boldsymbol{W}) &= -\frac{1}{N} \log \mathcal{L}(\boldsymbol{W}) \\ &= -\frac{1}{N} \sum_{i=1}^{N}\sum_{k=1}^{K} y_i^{(k)} \log \sigma(\boldsymbol{w_k}^T \boldsymbol{x_i}) \end{aligned} \tag{53}$$

上式就是多分类逻辑回归模型的交叉熵损失函数。

[11] 为了描述方便，仍然混用“$y = c_k$”这样的表示来描述“$y$ 属于类别 $c_k$”这样的事实，虽然 $y$ 同时也是一个 One-Hot 向量。

# 10 多分类模型中梯度的计算

在推导过程中，需要用到 Softmax 函数的偏导数。将 Softmax 函数重写如下：

$$\sigma(\boldsymbol{w_k^T x_i}) = \frac{e^{\boldsymbol{w_k^T x_i}}}{\sum\limits_{j=1}^{K} e^{\boldsymbol{w_j^T x_i}}} \tag{54}$$

求解偏导数 $\frac{\partial \sigma(\boldsymbol{w_k^T x_i})}{\partial \boldsymbol{w_t}}$：

- 如果 $t = k$，那么：

$$\frac{\partial \sigma(\boldsymbol{w_k^T x_i})}{\partial \boldsymbol{w_t}} = \frac{e^{\boldsymbol{w_t^T x_i}}}{\sum\limits_{j=1}^{K} e^{\boldsymbol{w_j^T x_i}}} \boldsymbol{x_i} - \left( \frac{e^{\boldsymbol{w_t^T x_i}}}{\sum\limits_{j=1}^{K} e^{\boldsymbol{w_j^T x_i}}} \right)^2 \boldsymbol{x_i} = \sigma(\boldsymbol{w_t^T x_i}) \left(1 - \sigma(\boldsymbol{w_t^T x_i})\right) \boldsymbol{x_i} \tag{55}$$

- 如果 $t \neq k$，那么：

$$\frac{\partial \sigma(\boldsymbol{w_k^T x_i})}{\partial \boldsymbol{w_t}} = -\frac{e^{\boldsymbol{w_k^T x_i}} e^{\boldsymbol{w_t^T x_i}}}{\left( \sum\limits_{j=1}^{K} e^{\boldsymbol{w_j^T x_i}} \right)^2} \boldsymbol{x_i} = -\sigma(\boldsymbol{w_k^T x_i}) \sigma(\boldsymbol{w_t^T x_i}) \boldsymbol{x_i} \tag{56}$$

于是，针对多分类交叉熵损失函数 (公式53)，求解偏导数：

$$\begin{aligned}
\frac{\partial L(\boldsymbol{W})}{\partial \boldsymbol{w_t}} &= -\frac{1}{N} \sum_{i=1}^{N} \left[ y_i^{(t)} \frac{\sigma(\boldsymbol{w_t^T x_i}) \left(1 - \sigma(\boldsymbol{w_t^T x_i})\right) \boldsymbol{x_i}}{\sigma(\boldsymbol{w_t^T x_i})} - \sum_{k \neq t} y_i^{(k)} \frac{\sigma(\boldsymbol{w_k^T x_i}) \sigma(\boldsymbol{w_t^T x_i}) \boldsymbol{x_i}}{\sigma(\boldsymbol{w_k^T x_i})} \right] \\
&= -\frac{1}{N} \sum_{i=1}^{N} \left[ y_i^{(t)} \left(1 - \sigma(\boldsymbol{w_t^T x_i})\right) \boldsymbol{x_i} - \sum_{k \neq t} y_i^{(k)} \sigma(\boldsymbol{w_t^T x_i}) \boldsymbol{x_i} \right] \\
&= -\frac{1}{N} \sum_{i=1}^{N} \left( y_i^{(t)} - \sigma(\boldsymbol{w_t^T x_i}) \right) \boldsymbol{x_i}
\end{aligned} \tag{57}$$

在推导过程中，利用了公式 $y_i^{(t)} + \sum\limits_{k \neq t} y_i^{(k)} = \sum\limits_{j=1}^{K} y_i^{(j)} = 1$。上式中，$y_i^{(t)}$ 表示第 $t$ 个类别的理想输出，$\sigma(\boldsymbol{w_t^T x_i})$ 表示第 $t$ 个类别的实际输出。比较公式 (25) 和公式 (57)，可以发现，二分类与多分类的偏导数非常相似，也具有非常类似的解释。

# 11 神经网络的视角

[12]在神经网络中，当输出层使用 Sigmoid 作为神经元的激活函数时，如果代价函数使用均方误差 (Mean Squared Error, MSE) 损失函数[13]：

$$L(\boldsymbol{w}) = \frac{1}{2N}\sum_{i=1}^{N}\left[y_i - \sigma\left(\boldsymbol{w}^T\boldsymbol{x_i}\right)\right]^2 \tag{58}$$

则输出层就存在学习缓慢的问题。该问题被称为神经元饱和问题，它是制约反传算法效率与有效性的一个重要因素。直观来看，Sigmoid 曲线在两端非常平坦，梯度很小，这会导致权值的调整量也很小。

为讨论方便，以单样本损失函数 $l(\boldsymbol{w})$ 为例，且假设输出层只有 1 个神经元，令输出层的权值输出 $z = \boldsymbol{w}^T\boldsymbol{x}$，输出层的激活输出 $a = \sigma(z)$，则中间量偏导数[14]为：

$$\begin{aligned}\frac{\partial l(\boldsymbol{w})}{\partial z} &= \frac{\partial l(\boldsymbol{w})}{\partial a}\cdot\frac{\partial a}{\partial z}\\ &= -\left[y - \sigma(z)\right]\cdot\sigma'(z)\end{aligned} \tag{59}$$

上式中，右端表达式含有梯度项 $\sigma'(z)$，它是造成学习缓慢的直接原因。

一个解决办法是，定义交叉熵 (Cross Entropy) 损失函数：

$$L(\boldsymbol{w}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i\log\sigma\left(\boldsymbol{w}^T\boldsymbol{x_i}\right) + (1-y_i)\log\left(1-\sigma\left(\boldsymbol{w}^T\boldsymbol{x_i}\right)\right)\right] \tag{60}$$

其中，$N$ 是训练数据的个数，$y_i$ 是第 $i$ 个样本的理想输出，$\sigma\left(\boldsymbol{w}^T\boldsymbol{x_i}\right)$ 是第 $i$ 个样本的实际输出。

在输出层神经元使用 Sigmoid 激活函数的情况下，使用交叉熵作为其损失函数是非常自然的。原因在于，正如第2节表明的那样，逻辑回归模型的负对数似然函数就是交叉熵损失函数 (公式16)。下面的分析将表明，使用交叉熵损失函数，可以消除输出层中存在的神经元饱和问题。

还是以单样本为例，针对交叉熵损失函数：

$$\frac{\partial l(\boldsymbol{w})}{\partial a} = -(\frac{y}{a} - \frac{1-y}{1-a}) = \frac{a-y}{a(1-a)} = \frac{\sigma(z)-y}{\sigma(z)(1-\sigma(z))} = \frac{\sigma(z)-y}{\sigma'(z)} \tag{61}$$

[12]更多细节，详见《人工智能》课程系列之《人工神经网络基础》，Chapter8-CN.pdf。

[13]下文将分析，这是一种不正确的使用方法。

[14]引入中间量偏导数，主要是为了方便地表示损失函数对各层权值的偏导数，详见《人工智能》课程系列之《人工神经网络基础》，Chapter8-CN.pdf。

将上式代入公式（59），得到：

$$\frac{\partial l(\boldsymbol{w})}{\partial z}=\frac{\partial l(\boldsymbol{w})}{\partial a}\cdot\frac{\partial a}{\partial z}=\frac{\sigma(z)-y}{\sigma'(z)}\cdot\sigma'(z)=\sigma(z)-y \tag{62}$$

上式表明，交叉熵损失函数的引入，消去了输出层中 Sigmoid 梯度项 $\sigma'(z)$。

上述结论，对于 Softmax 函数也适用[15]。一般而言，对于二分类情形，输出层使用 Sigmoid 激活函数，损失函数要使用交叉熵损失函数；对于多分类问题，输出层使用 Softmax 激活函数，相应地，损失函数就要使用多分类的交叉熵损失函数：

$$L(\boldsymbol{w})=-\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}y_i^{(k)}\log\sigma\left(\boldsymbol{w}_k^T\boldsymbol{x_i}\right) \tag{63}$$

其中，$N$ 表示样本个数，$K$ 表示分类个数，$\sigma(\cdot)$ 为 Softmax 函数。

# 12 广义线性模型与指数族分布

## 12.1 广义线性模型

在前面，使用 Sigmoid 函数将线性回归 $\boldsymbol{w}^T\boldsymbol{x}$ 的值域从 $(-\infty,+\infty)$ 变换到 $(0,1)$ 上，得到了二分类逻辑回归模型，即 Sigmoid 函数是通过 Logit 对数几率函数“凑出来”的。

实际上，从广义线性模型的视角看，在假设预测变量 $y$ 服从伯努利分布的情况下，使用 Sigmoid 函数或 Logit 对数几率函数作为变换函数，是一种必然的选择。

如果预测变量 $y$ 不是二值离散值，而是多值离散值，那么相应的变换函数是什么形式呢？另外，为什么线性回归要使用 $\boldsymbol{w}^T\boldsymbol{x}$ 这种公式形式呢？

下文将表明，在广义线性模型的框架下，针对不同的预测变量类型，可以选择不同的分布类型，从而形成了不同形式的模型：二分类逻辑回归的 Sigmoid 函数对应着伯努利分布 (二分类)；多分类逻辑回归的 Softmax 函数对应着 $K$ 分类 (One-Hot 形式)；线性回归 $\boldsymbol{w}^T\boldsymbol{x}$ 对应着正态分布。

下面给出广义线性模型的定义，它由三个部分组成：

1. 随机成分 (Random Component)：$p(y;\eta)$[16]

[15]具体分析，详见《人工智能》课程系列之《人工神经网络基础》，Chapter8-CN.pdf。

[16]缺省情况下，指数族概率密度函数是以样本 $\boldsymbol{x}$ 为条件的。为简化公式，不列出 $\boldsymbol{x}$ 项。

它是广义线性模型的随机部分——指数族 (Exponential Family) 分布，定义了预测变量 $y$ 在给定输入变量 $\boldsymbol{x}$ 时的条件分布。一般情况下，常见的分布都属于指数族分布，例如高斯/正态 (Gaussian/Normal) 分布、伯努利 (Bernoulli) 分布、二项 (Binomial) 分布、多项 (Multinomial) 分布、泊松 (Poisson) 分布、伽玛 (Gamma) 分布等。下文将给出指数族分布的定义；

2. 线性预测器 (Linear Predictor)：$\eta$

   它是广义线性模型的线性回归 $\eta = \boldsymbol{w}^T\boldsymbol{x}$ 部分；

3. 链接函数 (Link Function)：$g\left(\mathbb{E}(y)\right)$[17]

   链接函数 $g(\cdot)$ 是一个光滑且可逆的函数，它对预测变量 $y$ 的期望 $\mathbb{E}(y)$ 进行变换，使得变换后的 $g\left(\mathbb{E}(y)\right)$ 与线性预测器 $\eta$ 匹配，即满足如下条件：

$$g\left(\mathbb{E}(y)\right) = \eta = \boldsymbol{w}^T\boldsymbol{x} \tag{64}$$

   例如，二分类逻辑回归中的 Logit 函数就是该模型的链接函数。

从上述定义可以看出，广义线性模型是预测变量 $y$ 的期望输出 $\mathbb{E}(y)$ 变换后的线性模型。

在广义线性模型框架中，模型的求解顺序如下：

1. 假设预测变量 $y$ 服从的分布；
2. 根据指数族分布通式，写出该分布的指数族分布形式；
3. 令链接函数等于线性预测器并进行拟合；

例如，在二分类逻辑回归模型中，首先需要假设预测变量 $y$ 服从的分布——伯努利分布；然后，写出伯努利分布的指数族分布形式；通过指数族分布通式，可以确定链接函数的形式为 Logit 对数几率函数，并令其等于线性预测器 $\eta$[18]；最后，使用各种迭代优化算法求解参数 $\boldsymbol{w}$。

[17] 对回归问题，$y$ 是预测变量的值，$\mathbb{E}(y)$ 是预测变量的期望值。对分类问题，此处应把 $y$ 理解成 $y = c_i$ 事件或 $y$ 属于类别 $c_i$ 事件的发生与否：0，没有发生；1，发生。$\mathbb{E}(y)$ 应理解成 $\mathbb{E}(y = c_i)$，即事件 $y = c_i$ 发生的期望概率。

[18] 它具有固定的形式，即权值向量与输入特征向量的点积 $\boldsymbol{w}^T\boldsymbol{x}$

## 12.2 指数族分布

指数族分布指的是一类分布，它几乎涵盖了统计中绝大多数重要的分布，其等价的通用形式有多种：

$$\begin{aligned} p(y;\eta) &= h(y)\exp\left[\eta^T T(y) - A(\eta)\right] \\ p(y;\eta) &= h(y)g(\eta)\exp\left[\eta^T T(y)\right] \\ p(y;\eta) &= \exp\left[\eta^T T(y) - A(\eta) + B(y)\right] \end{aligned} \tag{65}$$

下面将以第 1 个公式为例展开讨论。在第 1 个公式中，$\eta$ 是自然参数：对于线性回归与逻辑回归，它是一个实数且 $\eta = \boldsymbol{w}^T\boldsymbol{x}$；对于多分类逻辑回归，$\eta$ 是一个向量且 $\eta_i = \boldsymbol{w_i}^T\boldsymbol{x}$。$T(y)$ 是充分统计量，对于线性回归与逻辑回归，$T(y) = y$；对于多分类逻辑回归，$T(y) = (1\{y = c_1\}, 1\{y = c_2\}, \ldots, 1\{y = c_K\})^T$，它是一个向量[19]。$A(\eta)$ 是一个对数配分函数，$e^{-A(\eta)}$ 起到归一化作用，确保概率密度函数在 $y$ 上的积分为 1。$h(y)$ 也是已知函数。这 3 个函数共同确定了分布的具体类型。

## 12.3 二分类 (伯努利分布) 的指数族形式

伯努利分布的指数族形式为：

$$\begin{aligned} p(y;\eta,\phi) &= \phi^y(1-\phi)^{1-y} \\ &= \exp\left(y\log\phi + (1-y)\log(1-\phi)\right) \\ &= \exp\left(\log\left(\frac{\phi}{1-\phi}\right)y + \log\left(1-\phi\right)\right) \end{aligned} \tag{66}$$

其中 $y = \{0, 1\}$，$\phi$ 为伯努利分布的参数。对照指数族分布的第一种形式，有：

$$\begin{aligned} &h(y) = 1 \\ &\eta = \log\left(\frac{\phi}{1-\phi}\right) = \boldsymbol{w}^T\boldsymbol{x} \Rightarrow \phi = \frac{1}{1+e^{-\eta}} = \frac{1}{1+e^{-\boldsymbol{w}^T\boldsymbol{x}}} \\ &T(y) = y \\ &A(\eta) = -\log(1-\phi) = \log\left(1+e^{\eta}\right) \end{aligned} \tag{67}$$

可以看出，上式第 2 行分别对应 Logit 函数与 Sigmoid 函数。

[19] $1\{y = c_k\}$ 是示性函数 (Characteristic Function) 或指示函数 (Indicator Function)，只有当条件 $y = c_k$ 成立时，函数取值为 1，否则为 0。

## 12.4 K 分类 (One-Hot 形式) 的指数族形式

在二分类中，$y$ 的取值是 $\{c_1, c_2\}$[20]。在 $K$ 分类中，$y$ 的取值为 $\{c_1, c_2, \ldots, c_K\}$，其中 $K$ 是分类个数。与二分类情形一样，为了便于表达概率，一般令 $y = \{0,1\}^K$，且 $\sum_{k=1}^{K} y_k = 1$，即把 $y$ 表达成 One-Hot 形式的向量。例如，在手写体数字识别问题中，$K = 10$，数字 3 和 4 被分别编码成 $y = (0,0,0,1,0,0,0,0,0,0)$ 和 $y = (0,0,0,0,1,0,0,0,0,0)$，其余数字的编码，依此类推[21]。

设 $p(y = c_k; \phi_k) = \phi_k$ 表示预测变量 $y$ 取分类 $c_k$ 的概率，则 $K$ 分类的概率可表示为：

$$p(y;\phi) = \phi_1^{1\{y=c_1\}} \phi_2^{1\{y=c_2\}} \ldots \phi_K^{1\{y=c_K\}} \tag{68}$$

上式中，$\sum_{k=1}^{K} \phi_k = 1$。

为了将 $K$ 分类模型表达成指数族分布，先引入指数族分布所需要的 $T(y)$。与二分类情形不同，它是一个向量。由于 $K$ 个概率 $\phi_i$ 之间存在冗余性，即只有 $K-1$ 个自由度。因此，令 $T(y)$ 为一个 $K-1$ 维向量，令 $T(y = c_k)(k = 1, 2, \ldots, K-1)$ 为 One-Hot 向量，而令 $T(y = c_K)$ 为一个 $K-1$ 维 $\mathbf{0}$ 向量，它们分别表示为：

$$T(y=c_1) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, T(y=c_2) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \cdots, T(y=c_{K-1}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, T(y=c_K) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{69}$$

在此定义下，可以将示性函数 $1\{y = c_k\}(k = 0, 1, \ldots, K-1)$ 转换为 $T(y)$ 的元素表示：

$$1\{y = c_k\} = (T(y))_k \tag{70}$$

其中，$(T(y))_k$ 表示该向量的第 $k$ 个元素[22]。而对于 $1\{y = c_K\}$：

$$1\{y = c_K\} = 1 - \sum_{k=1}^{K-1} (T(y))_k \tag{71}$$

[20]在多数情况下，为了便于表达概率，一般令 $y = \{0,1\}$ 或 $y = \{-1,1\}$，这 2 种形式的表示在前文中均已出现。

[21]在一些情况下，为了描述简便，还是会采取 $y = c_k$ 这种形式。一般可以依据上下文，判断其含义。

[22]有了示性函数，就可以在概率与期望之间建立联系：$E[1\{y = c_k\}] = p(y = c_k; \phi_k) = \phi_k$。于是，$E[(T(y))_k] = \phi_k$。

下面，将 $K$ 分类的概率公式表示为指数族形式：

$$\begin{aligned}
p(y;\phi) &= \phi_1^{1\{y=c_1\}}\phi_2^{1\{y=c_2\}}\cdots\phi_K^{1\{y=c_K\}} \\
&= \phi_1^{(T(y))_1}\phi_2^{(T(y))_2}\cdots\phi_{K-1}^{(T(y))_{K-1}}\phi_K^{1-\sum_{k=1}^{K-1}(T(y))_k} \\
&= \exp\left((T(y))_1\log(\phi_1)+(T(y))_2\log(\phi_2)+\right. \\
&\quad \left.\cdots+(T(y))_{K-1}\log(\phi_{K-1})+\left(1-\sum_{k=1}^{K-1}(T(y))_k\right)\log(\phi_K)\right) \\
&= \exp\left((T(y))_1\log(\phi_1/\phi_K)+(T(y))_2\log(\phi_2/\phi_K)+\right. \\
&\quad \left.\cdots+(T(y))_{K-1}\log(\phi_{K-1}/\phi_K)+\log(\phi_K)\right) \\
&= h(y)\exp\left(\eta^T T(y)-A(\eta)\right)
\end{aligned} \tag{72}$$

对照指数族分布的第 1 种形式，可得：

$$\begin{aligned}
\eta &= \begin{bmatrix} \log(\phi_1/\phi_K) \\ \log(\phi_2/\phi_K) \\ \vdots \\ \log(\phi_{K-1}/\phi_K) \end{bmatrix} \\
A(\eta) &= -\log(\phi_K) \\
h(y) &= 1
\end{aligned} \tag{73}$$

求解 $\eta_k$ 与 $\phi_k(k=1,2,\cdots,K-1)$ 之间的关系：

$$\eta_k = \log\frac{\phi_k}{\phi_K} \tag{74}$$

对上式两边同时取指数，得到：

$$\begin{aligned}
&e^{\eta_k}=\frac{\phi_k}{\phi_K} \quad\Rightarrow\quad \phi_K e^{\eta_k}=\phi_k \quad\Rightarrow\quad \phi_K\sum_{k=1}^{K}e^{\eta_k}=\sum_{k=1}^{K}\phi_k=1 \quad\Rightarrow \\
&\phi_K=\frac{1}{\sum_{k=1}^{K}e^{\eta_k}} \quad\Rightarrow\quad \phi_k=\frac{e^{\eta_k}}{\sum_{i=1}^{K}e^{\eta_i}}
\end{aligned} \tag{75}$$

其中，$\eta_k=\boldsymbol{w}_k^T\boldsymbol{x}(k=1,2,\cdots,K-1)$；对于 $K$，$\eta_K=0$。因此，上式最后一项为[23]：

[23]根据前面的讨论，可知，公式 (76) 的等价形式为：$\phi_k=\frac{e^{\boldsymbol{\theta}_k^T\boldsymbol{x}}}{\sum_{i=1}^{K}e^{\boldsymbol{\theta}_i^T\boldsymbol{x}}}$，即在原式的分子分母上同时乘以 $e^{\boldsymbol{w_K^T x}}$ 而得。其中，$\boldsymbol{w_K}$ 表示第 $K$ 个分类的权值向量 (冗余权值向量)。

$$\phi_k = \frac{e^{\eta_k}}{1+\sum_{i=1}^{K-1} e^{\eta_i}} = \frac{e^{\boldsymbol{w}_k^T \boldsymbol{x}}}{1+\sum_{i=1}^{K-1} e^{\boldsymbol{w}_i^T \boldsymbol{x}}} \tag{76}$$

由此，便得到 $K$ 分类情形下的 Softmax 函数。

## 12.5 高斯分布的指数族形式

高斯分布的指数族分布形式如下：

$$\begin{aligned} p(y;\eta,\phi) &= \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right) \\ &= \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{y^2-2y\mu+\mu^2}{2\sigma^2}\right) \\ &= \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{y^2}{2\sigma^2}\right)\cdot\exp\left(\frac{\mu}{\sigma^2}\cdot y-\frac{\mu^2}{2\sigma^2}\right) \end{aligned} \tag{77}$$

其中，$\phi=(\mu,\sigma)$。对照指数族分布的第 1 种形式，可得：

$$\begin{aligned} h(y) &= \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{y^2}{2\sigma^2}\right) \\ \eta &= \frac{\mu}{\sigma^2} \\ T(y) &= y \\ A(\eta) &= \frac{\mu^2}{2\sigma^2} \end{aligned} \tag{78}$$

对于上式的第 2 行：

$$\frac{\mu}{\sigma^2} = \eta = \boldsymbol{w}^T\boldsymbol{x} \quad \Rightarrow \quad \mu = \sigma^2\boldsymbol{w}^T\boldsymbol{x} \quad \Rightarrow \quad \mu = \boldsymbol{\theta}^T\boldsymbol{x} \tag{79}$$

可以看出，$\sigma^2$ 只是对权值参数 $\boldsymbol{w}$ 进行了整体缩放，对其优化不会产生任何影响；或者，使用新权值参数 $\boldsymbol{\theta}$ 取代原权值参数 $\boldsymbol{w}$，再进行优化，两者的效果是一样的。因此，仍然可以使用 $\mu=\boldsymbol{w}^T\boldsymbol{x}$ 来表示线性拟合。于是，便得到了高斯分布假设下的线性回归公式。

在数据集的情况下，将 $\mu=\boldsymbol{w}^T\boldsymbol{x}$ 代入高斯分布，并列出数据集的最大似然函数，可以证明，最大似然估计与最小二乘法是等价的，即[24]：

$$\boldsymbol{w}_{MLE} = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N}\left(y^{(i)}-\boldsymbol{w}^T\boldsymbol{x}^{(i)}\right)^2 = \boldsymbol{w}_{LSE} \tag{80}$$

[24]从损失函数的角度，相当于使用平方误差 (Quadratic Error) 损失函数，它与均方误差 (Mean Squared Error, MSE) 损失函数只相差 1 个常数因子 $\frac{1}{N}$，其中 $N$ 表示数据集中样本点的个数。

# 13 逻辑回归与高斯朴素贝叶斯

逻辑回归是一种判别式模型 (Discriminative Model)，而 (高斯) 朴素贝叶斯是一种生成式模型 (Generative Model)。通过使用贝叶斯定理，可以将高斯朴素贝叶斯转换成类似逻辑回归那样的公式形式。尽管如此，但是需要注意的是，两者属于不同的模型，对参数的求解方式也是不一样的。事实上，参数求解方式的差异性，来源于判别式模型与生成式模型之间的差异性。

高斯朴素贝叶斯 (Gaussian Naive Bayes, GNB) 是朴素贝叶斯分类器中的一种，它具有如下特点：

- 在 $m$ 维输入特征向量 $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)$ 中，它的每个特征 $x_i$ 均为连续随机变量，且特征之间满足条件独立性；
- 因为特征之间的条件独立性，所以可以把每个特征的分布当作一维高斯分布，即：

$$p(x_i|y = c_k) = \frac{1}{\sqrt{2\pi\sigma_{ik}^2}} exp(-\frac{(x_i - \mu_{ik})^2}{2\sigma_{ik}^2}) \tag{81}$$

其中，在二分类中，$k = 1, 2$。它的 2 个模型参数，可以使用频率学派的最大似然方法从训练集中进行估计：

$$\begin{aligned} \mu_{ik} &= \frac{\sum_{i=1}^{N} x_{ik}}{N} \\ \sigma_{ik}^2 &= \frac{\sum_{i=1}^{N} (x_{ik} - \mu_{ik})^2}{N} \end{aligned} \tag{82}$$

它们分别表示某个类别 $c_k$ 中第 $i$ 个特征 $x_i$ 的均值与方差。

注意，在下面的推导中，假设 $\sigma_{ik}$ 与类别 $c_k$ 无关，于是令 $\sigma_i = \sigma_{ik}$。为描述方便，令 $c_1 = 1$，$c_2 = 0$。

在逻辑回归模型中，模型直接对 $p(y|\boldsymbol{x})$ 建模，用来表示给定输入 $\boldsymbol{x}$ 划分到某

个类别的概率。在高斯朴素贝叶斯模型中，需要利用贝叶斯定理来表达分类概率：

$$\begin{aligned} p(y=1|\boldsymbol{x}) &= \frac{p(y=1)p(\boldsymbol{x}|y=1)}{p(y=1)p(\boldsymbol{x}|y=1)+p(y=0)p(\boldsymbol{x}|y=0)} \\ &= \frac{1}{1+\exp\left(\ln\frac{p(y=0)p(\boldsymbol{x}|y=0)}{p(y=1)p(\boldsymbol{x}|y=1)}\right)} \\ &= \frac{1}{1+\exp\left(\ln\frac{p(y=0)}{p(y=1)}+\ln\frac{p(\boldsymbol{x}|y=0)}{p(\boldsymbol{x}|y=1)}\right)} \\ &= \frac{1}{1+\exp\left(\ln\frac{1-p(y=1)}{p(y=1)}+\sum_{i=1}^{m}\ln\frac{p(x_i|y=0)}{p(x_i|y=1)}\right)} \end{aligned} \tag{83}$$

利用公式：

$$p(x_i|y=c_k)=\frac{1}{\sqrt{2\pi\sigma_{ik}^2}}exp(-\frac{(x_i-\mu_{ik})^2}{2\sigma_{ik}^2}) \tag{84}$$

以及 $\sigma_i=\sigma_{ik}$，可得公式 (83) 中最后一行求和式的简化形式：

$$\begin{aligned} \sum_{i=1}^{m}\ln\frac{p(x_i|y=0)}{p(x_i|y=1)} &= \sum_{i=1}^{m}\left(\frac{(x_i-\mu_{i2})^2}{2\sigma_i^2}-\frac{(x_i-\mu_{i1})^2}{2\sigma_i^2}\right) \\ &= \sum_{i=1}^{m}\left(\frac{\mu_{i1}-\mu_{i2}}{\sigma_i^2}x_i+\frac{\mu_{i2}^2-\mu_{i1}^2}{2\sigma_i^2}\right) \end{aligned} \tag{85}$$

将上式代入公式 (83)，继续变形：

$$\begin{aligned} p(y=1|\boldsymbol{x}) &= \frac{1}{1+\exp\left(\ln\frac{1-p(y=1)}{p(y=1)}+\sum_{i=1}^{m}\frac{\mu_{i2}^2-\mu_{i1}^2}{2\sigma_i^2}+\sum_{i=1}^{m}\frac{\mu_{i1}-\mu_{i2}}{\sigma_i^2}x_i\right)} \\ &= \frac{1}{1+\exp\left(-(w_0+\sum_{i=1}^{m}w_ix_i)\right)} \\ &= \frac{1}{1+\exp\left(-\boldsymbol{\theta}^T\boldsymbol{x}\right)} \end{aligned} \tag{86}$$

其中：

$$\begin{aligned} w_0 &= -\left(\ln\frac{1-p(y=1)}{p(y=1)}+\sum_{i=1}^{m}\frac{\mu_{i2}^2-\mu_{i1}^2}{2\sigma_i^2}\right) \\ w_i &= -\left(\frac{\mu_{i1}-\mu_{i2}}{\sigma_i^2}\right) \\ \boldsymbol{\theta} &= (w_0,w_1,\ldots,w_m)^T \\ \boldsymbol{x} &= (1,x_1,x_2,\ldots,x_m)^T \end{aligned} \tag{87}$$

因此，形式上看，高斯朴素贝叶斯的分类条件概率 $p(y|\boldsymbol{x})$ 与二分类逻辑回归的 Sigmoid 函数是一致的。

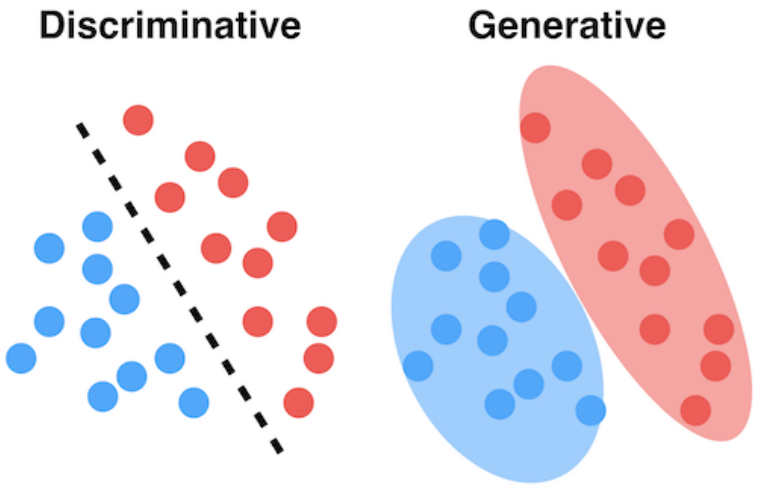



图 13-4: 判别式模型与生成式模型示意图

但是，需要注意的是，它们只是形式上的一致。在高斯朴素贝叶斯模型表达为逻辑回归模型前，参数 $\boldsymbol{\theta}$ 已经可以利用最大似然法来确定——使用相应的公式，直接利用训练集估计 $p(y=c_k)$、$\mu_{i1}$、$\mu_{i2}$ 以及 $\sigma_i^2$，而不需要通过迭代优化的方式求解参数 $\boldsymbol{\theta}$。逻辑回归模型则不同，需要以迭代优化的方式求解权值参数。

从条件概率 $p(y|\boldsymbol{x})$ 的获取方式来看，逻辑回归直接利用模型估计 $p(y|\boldsymbol{x})$，而高斯朴素贝叶斯则需要先估计 $p(y)$ 和 $p(\boldsymbol{x}|y)$，然后再利用贝叶斯定理推导出 $p(y|\boldsymbol{x})$。虽然上面的推导过程已经表明，最后的 $p(y|\boldsymbol{x})$ 公式有一致的形式，但是高斯朴素贝叶斯模型还存在一个非常强的假设——特征分量间的条件独立性。直觉上，就分类效果而言，逻辑回归要优于高斯朴素贝叶斯。

图13-4展示了判别式模型与生成式模型的区别，图中红色与蓝色圆点属于不同类别的训练集样本。判别式模型的目标是找到一个“最好”地区分它们的边界，而不关心样本点的分布规律；生成式模型首先需要对样本点的分布规律建模，即求解分布 $p(y)$ 和 $p(\boldsymbol{x}|y)$，然后再利用贝叶斯定理求解 $p(y|\boldsymbol{x})$，从而找到分类边界。图13-5展示了两类模型分类边界的确定方法示意图。图中黑色实线与黑色空心线表示两类样本点的真实分布曲线，绿色实线与绿色空心线表示生成式模型确定的两类样本点的分布曲线。红色竖直线表示判别式模型找到的分类边界，绿色点虚竖线表示生成式模型找到的分类边界。可以看出，判别式模型直接依据数据样本点确定分类边界，而生成式模型需要依据求解出的分布来确定分类边界。

事实上，逻辑回归模型没有使用特征分量的条件独立性假设，而是依据最大

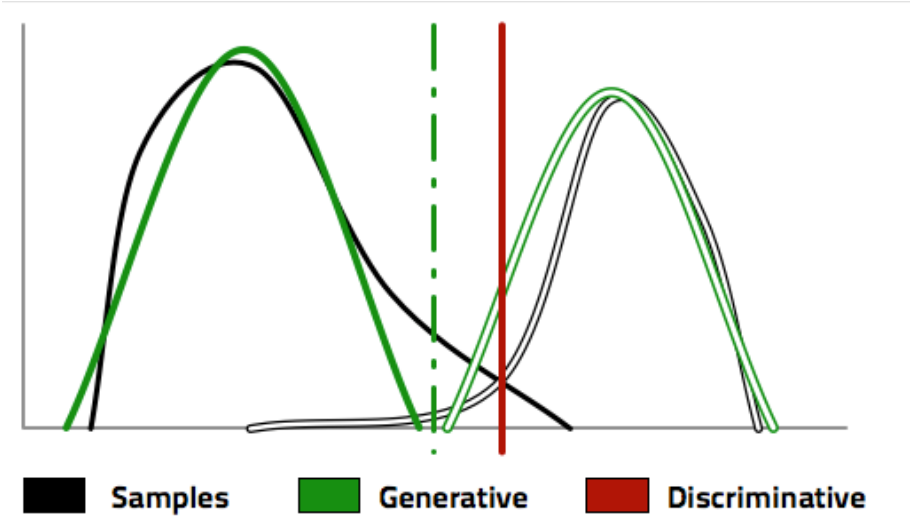



图 13-5: 两类模型分类边界的确定方法

似然方法，根据数据样本点的真实分布，直接最优化权值参数。在训练集合适时，其分类效果往往要优于朴素贝叶斯模型。

理论与实验表明，当训练集较小时，朴素贝叶斯模型因为收敛较快，其样本外的分类精度会高于逻辑回归模型；而随着训练集样本数的增多，逻辑回归将取得更高的分类精度。

# 14 附录

## 14.1 损失函数的绘制

```
# -*- coding: utf-8 -*-
"""
Created on Sun Apr 12 12:30:41 2020
@author: duxiaoqin
"""
import numpy as np
import matplotlib.pyplot as plt

plt.figure(figsize = (8, 5))

x = np.linspace(start = -1, stop = 2, num = 1000)

y_01 = x < 0
y_hinge = 1.0 - x
y_hinge[y_hinge < 0] = 0
y_logistic = np.log2(1 + np.exp(-x))
y_adaboost = np.exp(-x)
plt.plot(x, y_01, 'r-', label = '0-1 Loss', lw = 2)
plt.plot(x, y_hinge, 'g-', label = 'Hinge Loss', lw = 2)
plt.plot(x, y_logistic, 'b-.', label = 'Logistic Loss', lw = 2)
```

```
plt.plot(x, y_adaboost, 'm--', label = 'AdaBoost Loss',lw = 2)
plt.xlim(-1.0, 2.0)
plt.ylim(0.0, 3.0)
plt.legend(loc = 'upper right')
plt.title('Loss Function')
plt.show()
```

## 14.2 Logistic 函数的绘制

```
# -*- coding: utf-8 -*-
"""
Created on Mon Apr 13 14:53:42 2020
@author: duxiaoqin
"""
import numpy as np
import matplotlib.pyplot as plt

plt.figure(figsize = (8, 5))

mu, gamma = 20, 0.3

x = np.linspace(start = 14, stop = 26, num = 1000)

y_logistic = 1/(1+np.exp(-(x-mu)/gamma))
y_density = y_logistic*(1-y_logistic)/gamma
plt.plot(x, y_logistic, 'r-', label = 'Logistic Distribution', lw = 2)
plt.plot(x, y_density, 'g--', label = 'Logistic Density', lw = 2)
plt.xlim(14, 26)
plt.ylim(0, 1.0)
plt.legend(loc = 'upper left')
plt.title('Logistic Distribution & Density Function\n $\mu=20$, $\gamma=0.3$')
plt.show()
```

## 14.3 Sigmoid 与 Tanh 函数的绘制

```
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 14 15:14:32 2020
@author: duxiaoqin
"""
import numpy as np
import matplotlib.pyplot as plt

plt.figure(figsize = (8, 5))

x = np.linspace(start = -8, stop = 8, num = 1000)

y_sigmoid = 1/(1 + np.exp(-x))
y_2sigmoid = 1/(1 + np.exp(-2 * x))
y_tanh = 2 * y_2sigmoid - 1

ax=plt.gca()
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.spines['bottom'].set_position(('data',0))
ax.spines['left'].set_position(('data',0))

plt.plot(x, y_sigmoid, 'r-', label = 'Sigmoid', lw = 2)
plt.plot(x, y_tanh, 'g--', label = 'Tanh', lw = 2)
```

```
27 plt.xlim(-8, 8)
28 plt.ylim(-1.0, 1.0)
29 plt.legend(loc = 'upper left')
30 plt.title('Sigmoid & Tanh')
31 plt.show()
```

## 14.4 逻辑回归实验

### 0.1 二分类逻辑回归分类器的实现

```
In [1]: import numpy as np
```

```
In [2]: class BinLogisticRegression:
            def __init__(self, max_step = 80, learning_rate = 0.01):
                self.max_step = max_step
                self.learning_rate = learning_rate

            def sigmoid(self, z):
                return 1/(1+np.exp(-z))

            # 可用训练方案：批训练、mini-batch 训练与随机梯度下降训练
            def fit(self, X, Y):
                self.weights = np.zeros((1, len(X[0])), dtype = np.float32)

                self.paras = [] # 存放中间权值的结果，用于生成动画
                for step in range(self.max_step):
                    results = self.sigmoid(np.dot(X, np.transpose(self.weights)))
                    self.weights = self.weights + np.sum(self.learning_rate *
                     (Y - results) * X, axis = 0)
                    self.paras.append(self.weights)
                print('Weights:', self.weights)
                print('Binomial logistic regression model training completed!')

            # 直接利用对数几率（logit）函数进行预测：wx>=0, 1；否则，0
            def predict(self, X):
                return [[1] if z >= 0 else [0] for z in
                 np.dot(X, np.transpose(self.weights))]
```

载入预存的鸢尾花数据集

```
In [3]: iris_npz = np.load('iris.npz')
        data = iris_npz['data']
        X = iris_npz['X']
        Y = iris_npz['Y']
```

数据集预处理

```
In [4]: from sklearn.model_selection import train_test_split

        X = np.insert(X, 0, values = 1, axis = 1) # 将偏置并入权值中，其对应的 x=1
        Y = Y.reshape(-1, 1)

        XTRAIN, XTEST, YTRAIN, YTEST = train_test_split(X, Y, test_size = 0.3)
```

使用鸢尾花数据集训练二分类逻辑回归模型

```
In [5]: blr = BinLogisticRegression()
        blr.fit(XTRAIN, YTRAIN)
```

```
Weights: [[-0.5860826   4.06423765 -6.89826648]]
Binomial logistic regression model training completed!
```

使用鸢尾花测试数据集进行测试

```
In [6]: from collections import Counter

        results = blr.predict(XTEST)
        scores = (results == YTEST)
        scores = [score[0] for score in scores]
        print('Accuracy = {:3.2f}%'.format(Counter(scores)[True]/len(YTEST)
         * 100))
```

```
Accuracy = 96.67%
```

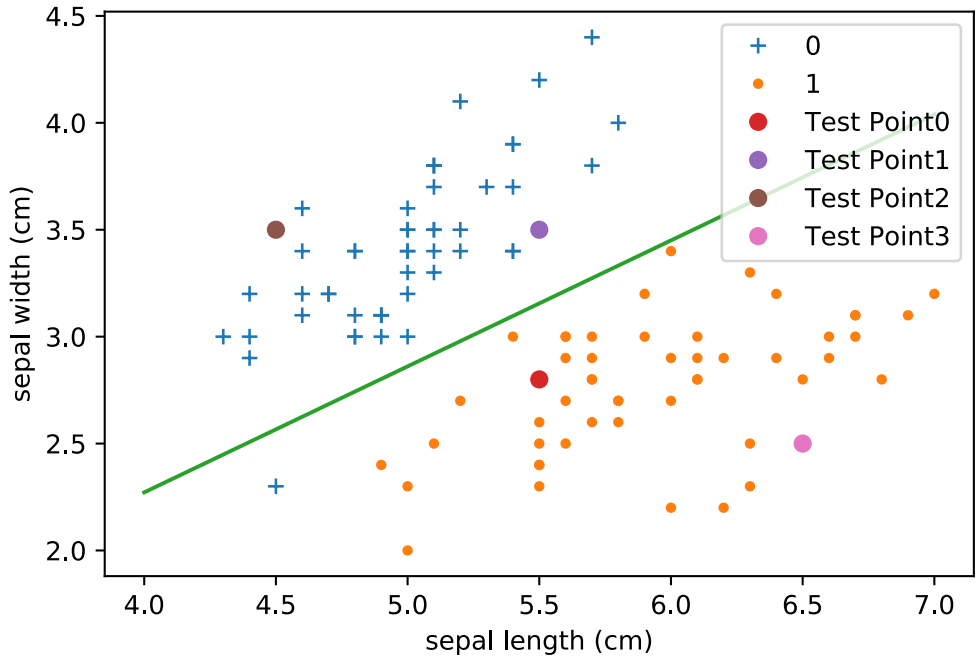
简单的预测测试

```
In [7]: XSAMPLES = [(1, 5.5, 2.8), (1, 5.5, 3.5), (1, 4.5, 3.5), (1, 6.5, 2.5)]
        results = blr.predict(XSAMPLES)
        print(results)
```

```
[[1], [0], [0], [1]]
```

绘制数据集、决策面与测试样例

```
In [8]: import matplotlib.pyplot as plt
        %matplotlib inline
        %config InlineBackend.figure_format = 'svg'

        plt.plot(data[:50, 0], data[:50, 1], '+', label='0')
        plt.plot(data[50:100, 0], data[50:100, 1], '.', label='1')

        # 绘制决策面
        x_ponits = np.arange(4, 8)
        y_points = -(blr.weights[0][1] * x_ponits + blr.weights[0][0])/
         blr.weights[0][2]
        plt.plot(x_ponits, y_points)

        for i, point in enumerate(XSAMPLES):# 绘制测试样例点
            plt.plot(point[1], point[2], 'o', label='Test Point{0}'.format(i))
        plt.xlabel('sepal length (cm)')
        plt.ylabel('sepal width (cm)')
        plt.legend()
        plt.savefig('LR_OUTPUT1.pdf', bbox_inches='tight')
```




绘制训练动画

```
In [9]: import matplotlib.animation as animation
        from IPython.display import HTML
        # 动画播放，如果是%matplotlib inline，则会嵌入图片
        %matplotlib notebook

        fig = plt.figure()
        ax = plt.axes(xlim=(3, 7), ylim=(2, 5))

        ax.plot(data[:50, 0], data[:50, 1], '+', label='0')
        ax.plot(data[50:100, 0], data[50:100, 1], '.', label='1')
        ax.set_xlabel('sepal length (cm)')
        ax.set_ylabel('sepal width (cm)')
        ax.set_title('Logistic Regression')
        ax.legend(loc = 'upper left')

        line, = ax.plot([], [], lw = 2)
        def update(para):
            b = para[0][0]
            W0 = para[0][1]
            W1 = para[0][2]

            X_points = np.linspace(4, 7,10)
            Y_points = -(W0 * X_points + b)/W1
```

```
            line.set_data(X_points, Y_points)
        ani = animation.FuncAnimation(fig, update, blr.paras)
        # 视频播放
        # 安装 FFMPEG 库
        # 在 anaconda 环境下，需要执行“conda install -c conda-forge ffmpeg”
        # 需要 FFMPEG.exe，直接下载 BUILD，解压到目录，例如 E:\ffmpeg\bin
        # 然后在环境变量 PATH 中添加“E:\ffmpeg\bin”，路径名称需要根据实际情况
        # 做相应的调整
        HTML(ani.to_html5_video())
```

```
In [10]: ani.save('Fitting-duxiaoqin.mp4')
```

## 0.2 多分类逻辑回归分类器的实现

```
In [11]: class MulLogisticRegression:
             def __init__(self, max_step = 80, learning_rate = 0.01):
                 self.max_step = max_step
                 self.learning_rate = learning_rate

             # 支持 X 为多样本情形（即矩阵形式）
             def softmax(self, X):
                 Y = np.exp(np.dot(X, np.transpose(self.weights)))
                 return Y / np.sum(Y, axis = 1).reshape(-1, 1)

             # 可用训练方案：批训练、mini-batch 训练与随机梯度下降训练
             def fit(self, X, Y):
                 K = len(Y[0]) #Y:One-hot 编码格式，K 表示类别个数
                 self.weights = np.zeros((K, len(X[0])), dtype = np.float32)

                 for step in range(self.max_step):
                     results = self.softmax(X)
                     errors = Y - results
                     # 配置 K 个权值向量（通常情况下，配置 K-1 个权值向量。
                     # 可以证明 2 者理论上等价）
                     for k in range(K):
                         # 可以证明，多项逻辑回归与二项情形非常类似：每个类别的
                         # 权值分量只与本类别的误差及 X 的相应分量有关
                         self.weights[k] = self.weights[k] + np.sum(
                          self.learning_rate * errors[:, k].reshape(-1, 1) * X,
                          axis = 0)
                 print('Weights:', self.weights)
                 print('Multinomial logistic regression model training completed!')

             def predict(self, X):
                 return np.argmax(self.softmax(X), axis = 1)
```

使用所有的鸢尾花数据集

```
In [12]: iris_npz_full = np.load('iris_full.npz')
         data_full = iris_npz_full['data']
         XFULL = iris_npz_full['X']
         YFULL = iris_npz_full['Y']
```

数据集预处理

```
In [13]: from sklearn.preprocessing import OneHotEncoder
         # 将偏置并入权值中，其对应的 x=1
         XFULL_POST = np.insert(XFULL, 0, values = 1, axis = 1)
         YFULL_POST = YFULL.reshape(-1, 1)

         encoder = OneHotEncoder()
         encoder.fit(YFULL_POST)
         YFULL_POST = encoder.transform(YFULL_POST).toarray()

         XTRAINFULL, XTESTFULL, YTRAINFULL, YTESTFULL = train_test_split(
          XFULL_POST, YFULL_POST, test_size = 0.3)
```

使用所有的鸢尾花数据集训练多项逻辑回归模型

```
In [14]: mlr = MulLogisticRegression(max_step = 280)
         mlr.fit(XTRAINFULL, YTRAINFULL)
```

```
Weights: [[  1.39598751   3.13638735   6.89958143  -8.52016354  -4.02731323]
 [  2.73833251   4.59755087   0.27354085  -3.89949369  -4.9549365 ]
 [ -4.13431931  -7.73394251  -7.17312574  12.41965485   8.98224831]]
Multinomial logistic regression model training completed!
```

使用鸢尾花测试数据集进行测试

```
In [15]: results = mlr.predict(XTESTFULL)
         scores = (results == np.argmax(YTESTFULL, axis = 1))
         print('Accuracy = {:3.2f}%'.format(Counter(scores)[True]/
          len(YTESTFULL) * 100))
```

```
Accuracy = 97.78%
```

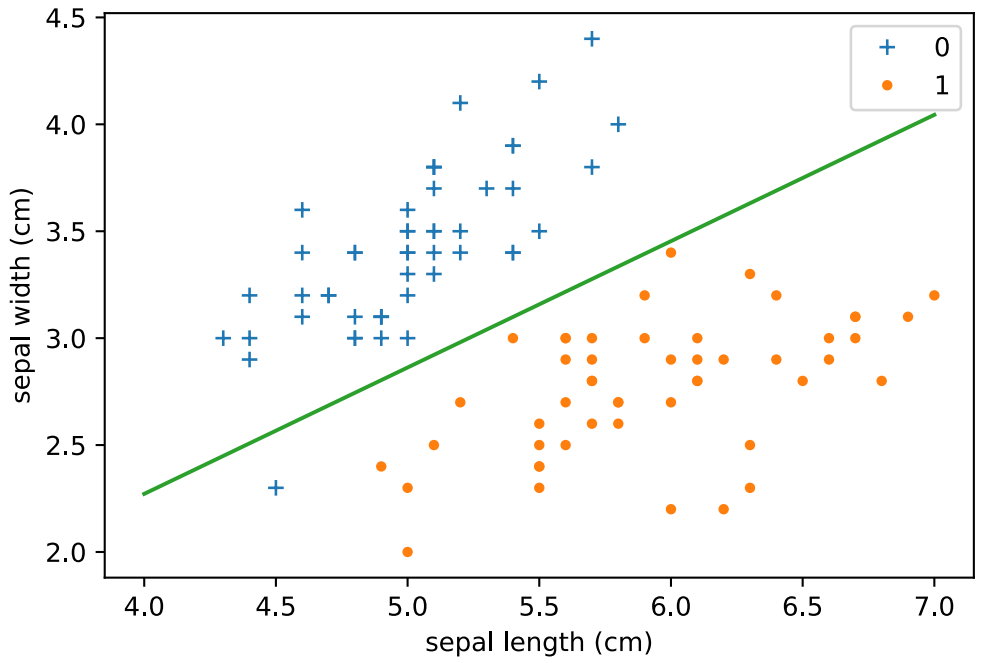
sklearn.linear_model.LogisticRegression

该类中的 solver 参数定义了四种优化方法，它们分别是：- liblinear：使用开源 liblinear 库的实现，内部使用坐标轴下降法来迭代优化损失函数；- lbfgs：拟牛顿法的一种，利用近似海森矩阵来迭代优化损失函数；- newton-cg：牛顿法的一种，利用海森矩阵来迭代优化损失函数；- sag：随机平均梯度下降，梯度下降法的变种。与普通梯度下降法的区别在于，每次迭代仅仅用一部分的样本来计算梯度，适用于样本数据较多的情形；

```
In [16]: from sklearn.linear_model import LogisticRegression
```

使用鸢尾花的二分类数据进行训练与测试

```
In [17]: clf = LogisticRegression(max_iter = 200)
         # 去掉 XTRAIN 中附加的 1（第 0 列）
         clf.fit(XTRAIN[:, 1:], YTRAIN.reshape(-1))

Out[17]: LogisticRegression(C=1.0, class_weight=None, dual=False,
         fit_intercept=True, intercept_scaling=1, max_iter=200, multi_class='ovr',
         n_jobs=1, penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
         verbose=0, warm_start=False)

In [18]: clf.score(XTEST[:, 1:], YTEST.reshape(-1))

Out[18]: 0.96666666666666667

In [19]: %matplotlib inline
         %config InlineBackend.figure_format = 'svg'

         plt.plot(data[:50, 0], data[:50, 1], '+', label='0')
         plt.plot(data[50:100, 0], data[50:100, 1], '.', label='1')

         # 绘制决策面
         x_ponits = np.arange(4, 8)
         y_points = -(clf.coef_[0][0]*x_ponits + clf.intercept_)/clf.coef_[0][1]
         plt.plot(x_ponits, y_points)

         plt.xlabel('sepal length (cm)')
         plt.ylabel('sepal width (cm)')
         plt.legend()
         plt.savefig('LR_OUTPUT2.pdf', bbox_inches='tight')
```



使用鸢尾花的多分类数据进行训练与测试

```
In [20]: clf = LogisticRegression(max_iter = 200)
         # 去掉 XTRAIN 中附加的 1（第 0 列）
         clf.fit(XTRAINFULL[:, 1:], np.argmax(YTRAINFULL, axis = 1))

Out[20]: LogisticRegression(C=1.0, class_weight=None, dual=False,
         fit_intercept=True, intercept_scaling=1, max_iter=200, multi_class='ovr',
         n_jobs=1, penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
         verbose=0, warm_start=False)

In [21]: clf.score(XTESTFULL[:, 1:], np.argmax(YTESTFULL, axis = 1))

Out[21]: 0.97777777777777775
```

# 15 参考文献


1. Christopher M. Bishop. Pattern Recognition and Machine Learning. Springer, 2006.
2. 史春奇、卜晶祎、施智平。《机器学习：算法背后的理论与优化》，清华大学出版社，2019 年 7 月第 1 版。
3. 周志华。《机器学习》，清华大学出版社，2016 年 1 月第 1 版。
4. 李航。《统计学习方法》，清华大学出版社，2019 年 5 月第 2 版。
5. Ng, A. Y. and M. I. Jordan (2002). On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes. In T. G. Dietterich, S. Becker and Z. Ghahramani (Eds), Advances in Neural Information Processing Systems, Vol. 14, MIT Press, 841 -848.
6. 逻辑回归与高斯朴素贝叶斯。https://www.gfedu.cn/aqf/content_26984.shtml.
7. Exponential Family. https://en.wikipedia.org/wiki/Exponential_family.
8. 逻辑回归：损失函数与梯度下降。https://blog.csdn.net/jediael_lu/article/details/77852060.
9. 指数族分布、广义线性模型、线性回归、logistic 回归。https://zhuanlan.zhihu.com/p/57757965.
10. 广义线性模型. https://zhuanlan.zhihu.com/p/22876460.
11. Softmax Regression. https://www.cnblogs.com/BYRans/p/4905420.html.
12. https://github.com/wzyonggege/statistical-learning-method.