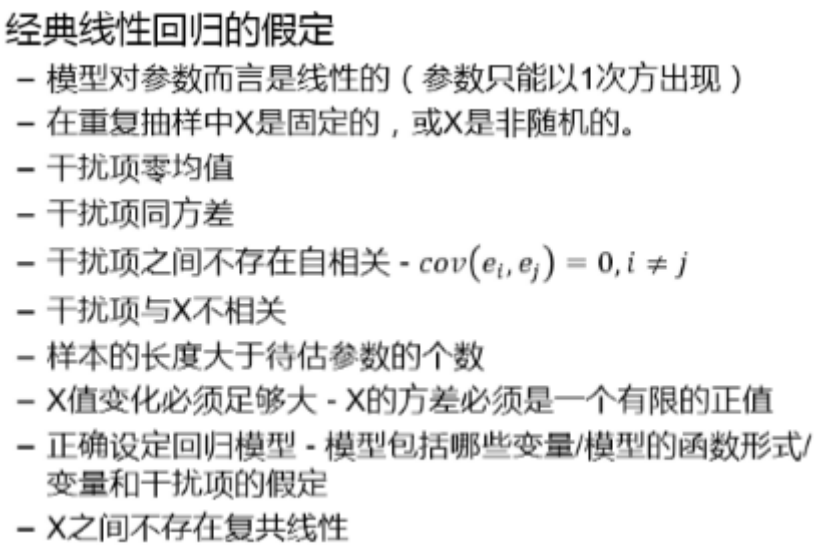
线性回归和逻辑回归原理

一、线性回归

- **定义**：线性回归是回归问题的基础，“线性”限制的是参数（parameter），而不是自变量(feature).线性回归分析是确定两种或者两种以上变量间相互依赖的定量关系的一种统计分析方法。本质上说，变量间相互依赖关系就是一种线性相关性，线性相关性是线性回归模型的理论基础。
- **线性回归的目的**：找到一个数学公式能相对较完美的把所有自变量组合（加减乘除）起来，得到的结果与目标相近。
- **经典线性回归假设**：



- **一元线性回归**：通俗的解释就是只用一个x来预测y,找一个直线来拟合数据。方程如下所示，其中x为自变量，y为自变量：

$$y = \beta_0 + \beta_1 x$$

通过方程得到的y值不是真实观测到的，而是估计值。现实世界中的数据就像下面的散点图，只能尽可能地在杂乱中寻找规律。用数学的模型去拟合现实数据，就是统计。统计会将理论与实际的差别表现出来，也就是“误差”—用u表示，即：

$$y = \beta_0 + \beta_1 x + \mu$$

- **多元线性回归：**

一元线性回归可以看成多元线性回归的一个简单情况，直接引出多元线性回归模型

1. **建立模型基本形式**

数据集：ps.下面公式中加粗的为向量或者矩阵

$$D=\{(\boldsymbol{x_1},y_1),(\boldsymbol{x_2},y_2),...,(\boldsymbol{x_N},y_N)\},\boldsymbol{x_i}\in\mathbb{R}^p,y_i\in\mathbb{R},i=1,2,...,N$$

$$\boldsymbol{X}=(\boldsymbol{x_1},\boldsymbol{x_2},...,\boldsymbol{x_N})^T=\begin{vmatrix}\boldsymbol{x_1}^T\\\boldsymbol{x_2}^T\\\vdots\\\boldsymbol{x_N}^T\end{vmatrix}=\begin{vmatrix}x_{11}&x_{12}&\cdots&x_{1P}\\x_{21}&x_{22}&\cdots&x_{2P}\\\vdots&\cdots&\ddots&\vdots\\x_{N1}&x_{N2}&\cdots&x_{NP}\end{vmatrix}_{N*P}$$

$$\boldsymbol{Y}=\begin{vmatrix}y_1\\y_2\\\vdots\\y_N\end{vmatrix}_{N*1}$$

多元线性模型试图学习到一个通过属性的线性组合来进行预测，预测函数为：

$$f(\boldsymbol{x_i})=\beta_0+\beta_1x_{i1}+\beta_2x_{i2}+...+\beta_Px_{iP}+\mu$$

一般用向量形式写成: $f(x_i)=\boldsymbol{W^T}x_i+b$,

其中b为偏置参数，类似于线性函数中的截距，在线性模型中补偿了目标值的平均值（在训练集上的）于基函数的加权平均值之间的差距。即打靶打歪了，但是允许通过平移固定向量的方法移动到目标点上。为了方便起见可以设置： $b=\beta_0x_i0$ ，其中 x_i0 为1，将其归纳到W中。即 $f(\boldsymbol{x_i})=W^T\boldsymbol{x_i}$ ，其中

$$\boldsymbol{W^T}=\begin{vmatrix}\beta_0\\\beta_1\\\vdots\\\beta_P\end{vmatrix}^T=(\beta_0,\beta_1,...,\beta_P),\boldsymbol{x_1}=\begin{vmatrix}x_{i0}\\x_{i1}\\\vdots\\x_{iP}\end{vmatrix}$$

2. 衡量预测值与真实值之间的距离

- 最终目标：模型预测出来的值和真实值无限接近，即： $f(\boldsymbol{x_i})\approx y_i$
- 衡量预测的准确性：介绍以下四种办法：

a. 误差平方和（SSE判别）

$$dist(Pi,Pj)=\sum_{k=1}^{n}(Pik-Pjk)^2$$

问题：

1> 使用平方后会放大(差>1)部分的误差，同时缩小(-1<差<1)部分的误差

2>当不同维度之间的度量差异很大时无法处理。例如：衡量一个人有年龄和收入两个维度，两个维度相差100倍以上，模型会严重受到收入大小的影响，要求在建模前对数据进行归一化处理

b.欧式距离判别

$$dist(Pi,Pj)=\sqrt{\sum_{k=1}^{n}(Pik-Pjk)^2}$$

问题：

1> 带根号，求解麻烦

2> 当不同维度之间的度量差异很大时无法处理

c.曼哈顿距离判别

$$dist(Pi,Pj)=\sum_{k=1}^{n}|Pik-Pjk|$$

问题：不是连续函数，求导很麻烦，计算不方便，只能计算垂直、水平距离

使用场景：数据稀疏（自带归一化处理）

d.马氏距离判别

$$dist(Pi,Pj)=\sqrt{(Pi-Pj)^TS^{-1}(Pi-Pj)}$$

问题：带根号，求解麻烦

使用场景：多维数据（流形学习），解决不同维度之间的度量差异很大时的问题

3. 建立损失函数

使用均方误差MSE来判定距离，并限定损失函数为平方损失函数，**也就是普通的最小二乘法**。损失函数为：

$L(\boldsymbol{W})=\sum_{i=1}^{N}(\boldsymbol{W^T x_i}-y_i)^2$,其目标为$minL(\boldsymbol{W})$

延伸问题：为什么可以用误差平方和来表示线性回归问题的损失函数

线性回归有这样的假设：对于给定的 $y^{(i)}$ 总能找到 $\varepsilon^{(i)}$ 使得这个等式成立：

$$y^{(i)} = h_\theta(x^{(i)}) + \varepsilon^{(i)}$$

$\varepsilon^{(i)}$ 代表真实值和预测值之间的误差且服从正态分布 $\varepsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$

为什么要加 ϵ ：房价有可能由100种特征共同组成，但是通常我们建模的时不可能把所有可能性都考虑完全，所以把剩下的特征及噪声统一放到 ϵ 来考虑

为什么误差服从正态分布：误差的产生有很多种因素的影响，误差可以看作是这些因素(随机变量)之和共同作用而产生的，由中心极限定理可知随机变量和的分布近似的服从正态分布

随机变量 $\varepsilon^{(i)}$ 的概率密度函数为：

$$p(\varepsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\varepsilon^{(i)})^2}{2\sigma^2}}$$

代入 $\epsilon^i = y^{(i)} - h_\theta(x^{(i)})$ 则：

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\theta^T x^{(i)} - y^{(i)})^2}{2\sigma^2}}$$

- **解释：**有无数种参数组合，每种参数组合都会对应一组 $p(y^{(i)}|x^{(i)})$ 的概率，现在要选一个 θ ，使得数据集中每个数据点的 $p(y^{(i)}|x^{(i)})$ 概率都最大。为了使所有数据点的 $p(y^{(i)}|x^{(i)})$ 的概率最大，则将每个点的概率求乘积，即选出一个 θ 使得数据集所有点 $p(y^{(i)}|x^{(i)})$ 的积最大。该方法即：**最大似然估计**

这里的 $p(y^{(i)}|x^{(i)};\theta)$ 并不代表条件概率，只是一个记号它表示给定 $x^{(i)}$ ， $y^{(i)}$ 和一组 θ 后的概率密度函数。由**最大似然估计**可知：

$$
\begin{aligned}
L(\theta) &= \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) \\
&= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\theta^T x^{(i)} - y^{(i)})^2}{2\sigma^2}}
\end{aligned}
$$

对 $L(\theta)$ 取对数从而得到对数化最大似然估计函数

$$
\begin{aligned}
\mathcal{L}(\theta) &= \log(L(\theta)) \\
&= \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\theta^T x^{(i)} - y^{(i)})^2}{2\sigma^2}} \\
&= \sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\theta^T x^{(i)} - y^{(i)})^2}{2\sigma^2}} \\
&= m\log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)})^2
\end{aligned}
$$

求解最大化对数似然函数可得：

$$
\begin{aligned}
\arg\max_{\theta} L(\theta) &\Leftrightarrow \arg\max_{\theta} \mathcal{L}(\theta) \\
&\Leftrightarrow \arg\max_{\theta} \left\{ m\log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)})^2 \right\} \\
&\Leftrightarrow \arg\max_{\theta} -\frac{1}{2\sigma^2} \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)})^2 \\
&\Leftrightarrow \arg\min_{\theta} \frac{1}{2\sigma^2} \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)})^2 \\
&\Leftrightarrow \arg\min_{\theta} \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)})^2 \\
&\Leftrightarrow \arg\min_{\theta} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2
\end{aligned}
$$

可以发现两种方法推出来的的损失函数都是一样的，把这种损失函数叫做最小二乘法

4. 求**W**的参数

最小二乘法得到的损失函数是一个凸函数，可以看做一个一元二次方程（自变量为**W**），目标是求该方程的最小值，即求一元二次方程导数逼近0的点。有两种方法：**正规方程法、梯度下降法**

- **正规方程法**

$$L(\boldsymbol{W}) = \sum_{i=1}^{N} (\boldsymbol{W^T x_i} - y_i)^2 \Big|$$

$$= (\boldsymbol{W^T x_1} - y_1 \quad \boldsymbol{W^T x_2} - y_2 \quad \dots \quad \boldsymbol{W^T x_N} - y_N) \begin{pmatrix} \boldsymbol{W^T x_1} - y_1 \\ \boldsymbol{W^T x_1} - y_1 \\ \dots \\ \boldsymbol{W^T x_N} - y_N \end{pmatrix}$$

$$= ((\boldsymbol{W^T x_1} \quad \boldsymbol{W^T x_2} \quad \dots \quad \boldsymbol{W^T x_N}) - (y_1 \quad y_2 \quad \dots \quad y_N))(\boldsymbol{XW} - \boldsymbol{Y})$$

$$= \boldsymbol{W^T}(\boldsymbol{x_1} \quad \boldsymbol{x_2} \quad \dots \quad \boldsymbol{x_N}) - (y_1 \quad y_2 \quad \dots \quad y_N)(\boldsymbol{XW} - \boldsymbol{Y})$$

$$= (\boldsymbol{W^T X^T} - \boldsymbol{Y^T})(\boldsymbol{XW} - \boldsymbol{Y}) \quad \text{展开}$$

常数

$$= \boldsymbol{W^T X^T XW} - \boldsymbol{Y^T XW} - \boldsymbol{W^T X^T Y} + \boldsymbol{Y^T Y}$$

$$= \boldsymbol{W^T X^T XW} - 2\boldsymbol{W^T X^T Y} + \boldsymbol{Y^T Y}$$

$$\widehat{\boldsymbol{W}} = \boldsymbol{argmin} L(\boldsymbol{W})$$

矩阵求导：

$$\frac{\partial(\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x})}{\partial \boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{A}^T \boldsymbol{x}$$

$$\frac{\partial L(\boldsymbol{W})}{\partial \boldsymbol{W}} = 2\boldsymbol{X^T XW} - 2\boldsymbol{X^T Y} \triangleq 0$$

$$\frac{\partial(\boldsymbol{x}^T \boldsymbol{a})}{\partial \boldsymbol{x}} = \frac{\partial(\boldsymbol{a}^T \boldsymbol{x})}{\partial \boldsymbol{x}} = \boldsymbol{a}$$

$$\Rightarrow \boldsymbol{X^T XW} = \boldsymbol{X^T Y}$$

$$\Rightarrow \widehat{\boldsymbol{W}} = (\boldsymbol{X^T X})^{-1} \boldsymbol{X^T Y}$$

$X^T X$ 现实任务中往往不是满秩矩阵，所以无法求解矩阵的逆。

非满秩矩阵：3个变量，但是只有2个方程

解决办法：引入正则化，将矩阵补成满秩

正则化框架：$argmin[L(\boldsymbol{W}) + \lambda P(\boldsymbol{W})]$

举例：L2正则化（岭回归）

$$L(\boldsymbol{W}) = \sum_{i=1}^{N} (\boldsymbol{W^T x_i} - y_i)^2 + \lambda \boldsymbol{W^T W}$$

$$= \boldsymbol{W^T X^T X W} - 2\boldsymbol{W^T X^T Y} + \boldsymbol{Y^T Y} + \lambda \boldsymbol{W^T W}$$

$$= \boldsymbol{W^T}(\boldsymbol{X^T X} + \lambda \boldsymbol{I})\boldsymbol{W} - 2\boldsymbol{W^T X^T Y} + \boldsymbol{Y^T Y}$$

$$\widehat{\boldsymbol{W}} = \boldsymbol{argmin} L(\boldsymbol{W})$$

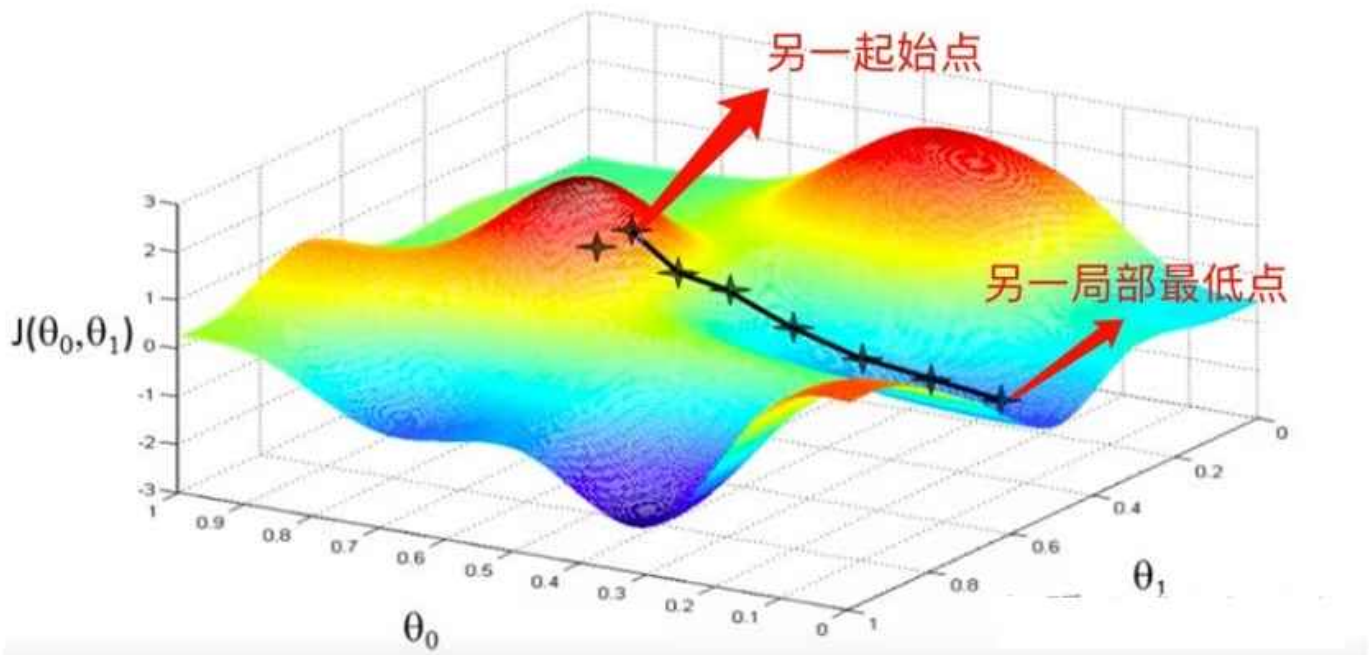
$$\frac{\partial L(\boldsymbol{W})}{\partial \boldsymbol{W}} = 2(\boldsymbol{X^T X} + \lambda \boldsymbol{I})\boldsymbol{W} - 2\boldsymbol{X^T Y} \triangleq 0$$

$$\Rightarrow \widehat{\boldsymbol{W}} = (\boldsymbol{X^T X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X^T Y}$$

- **梯度下降法**

非线性的最小二乘可以通过牛顿高斯迭代、LM算法和梯度下降求解

梯度下降：随机初始化**W,**通过逼近的方式来求解



回想以下多元线性回归的预测函数，损失函数为：

$$f(\boldsymbol{x_i}) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_P x_{iP} + \mu$$

$$L(W) = \sum_{i=1}^{N} (f(\boldsymbol{x_i}) - y_i)^2$$

损失函数的偏导数为：

$$\frac{\partial L(W)}{\partial \beta_j} = \sum_{i=1}^{N} 2\left(f(\boldsymbol{x_i}) - y_i\right)x_{ij} = 2\sum_{i=1}^{N}\left(\sum_{j=0}^{P}\beta_j x_{ij} - y_i\right)x_{ij}$$

每次更新参数的操作为：

$$\beta_j = \beta_j - \alpha\frac{\partial L(W)}{\partial \beta_j} = \beta_j - 2\alpha\sum_{i=1}^{N}\left(f(\boldsymbol{x_i}) - y_i\right)x_{ij} \quad (j = 0,1,2,\ldots,P)$$

其中 α 为学习率。注意：更新参数时必须同步更新所有参数，不能先更新 θ_0 在更新 θ_1

- **两种参数求解方法对比**

1. 正规方程法：

   a.不需要选择 α ，不需要iterations

   b.需要计算 $(X^T X)^{-1}$ ，求逆的算法时间复杂度为 $O(n^3)$,n为特征个数，如果特征很大，就会非常慢

   c.不适用于更加复杂的算法，像逻辑回归

2. 梯度下降法：一种优化方法，需要多次迭代来收敛到全局最小值

   a.需要选择 α ，需要很多次iterations

   b.在feature很多的情况下运行良好（million级别的特征）

二、逻辑回归

- 定义：逻辑回归的原理是用逻辑函数把线性回归的结果 $(-\infty, +\infty)$ 映射到（0,1），主要运用于两分类问题，研究某些事情发生的概率
- 逻辑回归的优缺点

优点：

1>速度快，适合二分类问题

2>简单易于理解，直接看到各个特征的权重

3>能容易的更新模型吸收新的数据

缺点：

1>对数据好场景的适应能力有局限性，不如决策树算法适应性强

- 逻辑回归和多元线性回归的区别

最大区别：因变量不同，根据因变量不同可以选择不同的回归模型

1> 如果是连续的，就用多元线性回归

2> 如果是二项分布，就用逻辑回归

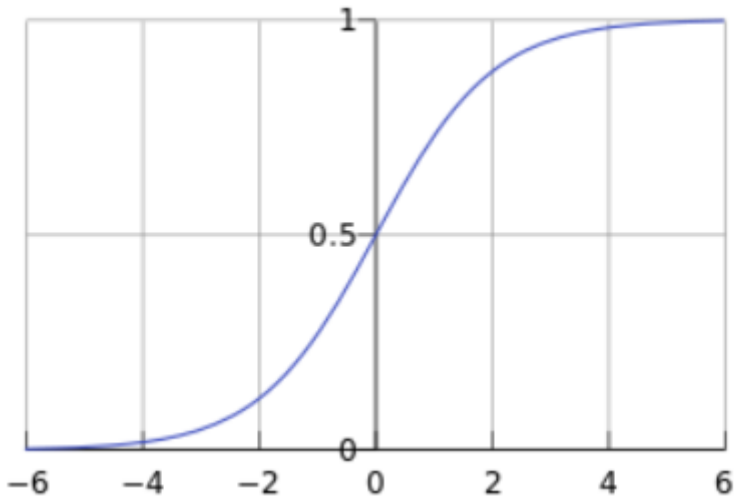
3> 如果是poisson分布，就用poisson回归

4> 如果是负二项分布，就用负二项回归

- 构造预测函数

Logistic函数（或称为Sigmoid函数）,函数形式：

$$g(z)=\frac{1}{1+e^{-z}}$$



其中，

$$z=\beta_0 x_i 0+\beta_1 x_{i1}+\beta_2 x_{i2}+\ldots+\beta_P x_{iP}=\sum_{j=0}^{P}\beta_j x_{ij}=W^T x_i$$

构造预测函数：

$$f_W(x_i)=g(W^T x_i)=\frac{1}{1+e^{-W^T x_i}}$$

函数 $f_{\boldsymbol{W}}(x_i)$ 的值有特殊含义，它表示结果取1的概率，对于输入 x_i 分类结果为类别1和类别0的概率分别为：

$$P\ (y_i = 1|x_i; W)\ = f_{\boldsymbol{W}}(x_i)$$

$$P\ (y_i = 0|x_i; W)\ = 1 - f_{\boldsymbol{W}}(x_i)$$

- 构造损失函数L（N个样本，每个样本具有P个特征）

Cost函数：

$$Cost(f_{\boldsymbol{W}}(x), y) = \begin{cases} -\log\big(f_W(x)\big) & if\ y = 1 \\ -\log\big(1 - f_W(x)\big) & if\ y = 0 \end{cases}$$

合并后的Cost函数为：

$$Cost(f_{\boldsymbol{W}}(x), y) = -y\log\big(f_W(x)\big) - (1-y)\log\big(1 - f_W(x)\big)$$

损失函数则为：

$$L(\boldsymbol{W}) = \sum_{i=1}^{N} Cost(f_{\boldsymbol{W}}(x_i), y_i) = -\sum_{i=1}^{N} y_i\log\big(f_W(x_i)\big) + (1-y_i)\log\big(1 - f_W(x_i)\big)$$

- 使用梯度下降法求解**W**

损失函数的偏导数为：

$$\frac{\partial L(W)}{\partial \beta_j} = \frac{\partial(-\sum_{i=1}^{N} y_i \log\left(f_W(x_i)\right) + (1-y_i)\log\left(1-f_W(x_i)\right))}{\partial \beta_j}$$

$$= -\sum_{i=1}^{N} y_i \frac{1}{g(\boldsymbol{W}^T x_i)} g(\boldsymbol{W}^T x_i)' - (1-y_i)\frac{1}{1-g(\boldsymbol{W}^T x_i)} g(\boldsymbol{W}^T x_i)'$$

$$= -\sum_{i=1}^{N} (y_i \frac{1}{g(\boldsymbol{W}^T x_i)} - (1-y_i)\frac{1}{1-g(\boldsymbol{W}^T x_i)}) g(\boldsymbol{W}^T x_i)'$$

$$= -\sum_{i=1}^{N} (y_i \frac{1}{g(\boldsymbol{W}^T x_i)} - (1-y_i)\frac{1}{1-g(\boldsymbol{W}^T x_i)}) g(\boldsymbol{W}^T x_i)(1-g(\boldsymbol{W}^T x_i))(\boldsymbol{W}^T x_i)'$$

$$= -\sum_{i=1}^{N} (y_i(1-g(\boldsymbol{W}^T x_i)) - (1-y_i)g(\boldsymbol{W}^T x_i))(\boldsymbol{W}^T x_i)'$$

$$= -\sum_{i=1}^{N} (y_i - g(\boldsymbol{W}^T x_i)) x_{ij}$$

$$= \sum_{i=1}^{N} (f_{\boldsymbol{W}}(x_i) - y_i) x_{ij}$$

每次更新参数的操作为：

$$\beta_j = \beta_j - \alpha \frac{\partial L(W)}{\partial \beta_j} = \beta_j - \alpha \sum_{i=1}^{N} \left(f_{\boldsymbol{W}}(x_i) - y_i\right) x_{ij} \quad (j = 0,1,2,\dots,P)$$

将其向量化：

$$\boldsymbol{A} = \boldsymbol{XW} = \begin{bmatrix} x_{10} & \cdots & x_{1P} \\ \vdots & \ddots & \vdots \\ x_{N0} & \cdots & x_{NP} \end{bmatrix} \begin{pmatrix} \beta_0 \\ \dots \\ \beta_N \end{pmatrix} = \begin{pmatrix} \beta_0 x_{10} + \beta_1 x_{11} + \cdots + \beta_N x_{1P} \\ \dots \\ \beta_0 x_{N0} + \beta_1 x_{N1} + \cdots + \beta_N x_{NP} \end{pmatrix}$$

$$\boldsymbol{E} = f_{\boldsymbol{W}}(X) - Y = \begin{pmatrix} g(A_1) - y_1 \\ \dots \\ g(A_N) - y_N \end{pmatrix} = \begin{pmatrix} e_0 \\ \dots \\ e_N \end{pmatrix} = g(A) - Y$$

参数更新的过程可以改为：

$$\beta_j = \beta_j - \alpha \sum_{i=1}^{N} e_i x_{ij} \quad (j = 0,1,2,\dots,P)$$

$$= \beta_j - \alpha \boldsymbol{X}^T \boldsymbol{E}$$

则向量化后的，参数更新的步骤如下：

1. 求**A**=**XW**
2. 求**E**=g(**A**)-**Y**
3. 求：

$$\beta_j = \beta_j - \alpha \boldsymbol{X}^T \boldsymbol{E}$$

2.2 FATE 逻辑回归实现原理

在应用中为了简化运算，将逻辑回归将标签 {0,1} 转化为标签为 {-1,1} 求解，转化后的损失函数和梯度为：

损失函数：

$$J(w) = \frac{1}{n}\sum_{i=1}^{n} ln(1 + e^{-y_i w^T x_i})$$

梯度：

$$\frac{\partial(J(w)}{\partial(w_j)} = \frac{1}{n}\sum_{i=1}^{n}(\frac{1}{1+e^{-y_i w^T x_i}} e^{-y_i w^T x_i}(-y_i x_{ij})) \quad (1)$$

$$= \frac{1}{n}\sum_{i=1}^{n}(\frac{1}{1+e^{-y_i w^T x_i}} - 1) y_i x_{ij} \quad (2)$$

在式子（1）中，求梯度时需要知道 w， y_i ， x_i 和 x_{ij}，其中 w 是逻辑回归模型的系数，是需要模型迭代确定的参数； y_i 是样本数据的标签值； x_i 是数据第 i 个样本数据， x_{ij} 是第 i 个样本数据的第 j 个特征数据。

在横向联邦学习中按参与方数据集数量做横向加权聚合，各参与方都拥有特征值和标签值，在求梯度时 y_i , x_i , x_{ij} 参与各方无需交换得到，参与方计算本地的参数 w 信息，后将参数 w 进行安全聚合，使用聚合后的 w 计算损失函数和下一次迭代的梯度。

参考信息

- 隐私计算算法系列 之 横向逻辑回归算法