面试问题

1.公式推导

2.逻辑回归的基本概念　广义线性模型　假设服从伯努利分布

回顾之前我们介绍的一元和多元线性回归，其最终目标都是寻找一个权值向量 $\boldsymbol{w}$ 和常数偏置 b，使每个样本数据 $(\boldsymbol{x}, y)$ 中的 $\boldsymbol{x}$ 代入二者所确立的模型

$$\hat{y} = \boldsymbol{w}^T \boldsymbol{x} + b \tag{1}$$

中所得到的模型预测值 $\hat{y}$ 与真实值 y 能够尽可能地接近。这里因变量 y 与自变量 $\boldsymbol{x}$ 中的每一维都是线性关系，而在现实问题中，绝大多数问题并非简单的线性情况，那么如果 y 的 **衍生** 可以满足与自变量的线性关系，例如取对数以后

$$\ln y = \boldsymbol{w}^T \boldsymbol{x} + b \tag{2}$$

我们把对数函数这类的函数称为 **联系函数**，其作用是：**把真实标记 y 转换成为其对应特征的线性回归值**。从形式上看式(2)仍是线性回归，但本质已经转变成了**寻找从输入空间到输出空间的非线性映射关系。**

更加一般地形式化表示，我们寻找一个单调可微的联系函数 $f(\bullet)$ 令其满足

$$y = f^{-1}(\boldsymbol{w}^T \boldsymbol{x} + b) \tag{3}$$

即联系函数的 **反函数的功能** 在于把输入空间的线性回归值映射到输出空间上。式(3)我们就称之为“广义线性模型”。

其实逻辑回归为什么要用sigmoid函数而不用其他是因为逻辑回归是采用的伯努利分布，伯努利分布的概率可以表示成

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

其中

$$\eta = \log(\phi/(1-\phi)).$$

得到

$$\Phi = \frac{1}{1+e^{\eta}}$$

这就解释了logistic回归时为了要用这个函数。

3.LR和SVM对比

首先，LR和SVM最大的区别在于损失函数的选择，LR的损失函数为Log损失（或者说是逻辑损失都可以）、而SVM的损失函数为hinge loss。

其次，两者都是线性模型。

最后，SVM只考虑支持向量（也就是和分类相关的少数点）

4.LR和随机森林区别

随机森林等树算法都是非线性的，而LR是线性的。LR更侧重全局优化，而树模型主要是局部的优化。

5.常用的优化方法

逻辑回归本身是可以用公式求解的，但是因为需要求逆的复杂度太高，所以才引入了梯度下降算法。

一阶方法：梯度下降、随机梯度下降、mini 随机梯度下降降法。随机梯度下降不但速度上比原始梯度下降要快，局部最优化问题时可以一定程度上抑制局部最优解的发生。

二阶方法：牛顿法、拟牛顿法：

这里详细说一下牛顿法的基本原理和牛顿法的应用方式。

（牛顿法几何意义理解：
https://blog.csdn.net/HouDouZhou/article/details/85988847）

牛顿法其实就是通过切线与x轴的交点不断更新切线的位置，直到达到曲线与x轴的交点得到方程解。在实际应用中我们因为常常要求解凸优化问题，也就是要求解函数一阶导数为0的位置，而牛顿法恰好可以给这种问题提供解决方法。实际应用中牛顿法首先选择一个点作为起始点，并进行一次二阶泰勒展开得到导数为0的点进行一个更新，直到达到要求，这时牛顿法也就成了二阶求解问题，比一阶方法更快。我们常常看到的x通常为一个多维向量，这也就引出了Hessian矩阵的概念（就是x的二阶导数矩阵）。

缺点：牛顿法是定长迭代，没有步长因子，所以不能保证函数值稳定的下降，严重时甚至会失败。还有就是牛顿法要求函数一定是二阶可导的。而且计算Hessian矩阵的逆复杂度很大。

拟牛顿法： 不用二阶偏导而是构造出Hessian矩阵的近似正定对称矩阵的方法称为拟牛顿法。拟牛顿法的思路就是用一个特别的表达形式来模拟Hessian矩阵或者是他的逆使得表达式满足拟牛顿条件。主要有DFP法（逼近Hession的逆）、BFGS（直接逼近Hession矩阵）、 L-BFGS（可以减少BFGS所需的存储空间）。

1.逻辑斯蒂回归介绍

LR可以用来回归，也可以用来分类，主要是二分类。假设样本{x,y},y是0或者1，表示正类或者负类，x是m维样本特征向量，那么这个样本x属于正类，也就是y=1的概率可以通过下面的逻辑函数表示：

$$p(y=1|\mathbf{x};\theta)=\sigma(\theta^{\top}\mathbf{x})=\frac{1}{1+\exp(-\theta^{\top}\mathbf{x})}$$

这里θ是模型参数，也就是回归系数，σ是sigmoid函数。

实际上这个函数是由下面的对数几率（也就是x属于正类的可能性和负类的可能性的比值的对数）变换得到的：

$$\log it(\mathrm{x}) = \ln(\frac{P(\mathrm{y}=1|\mathrm{x})}{P(\mathrm{y}=0|\mathrm{x})})$$

$$= \ln(\frac{P(\mathrm{y}=1|\mathrm{x})}{1-P(\mathrm{y}=1|\mathrm{x})}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_m x_m$$

我们将特征向量x1,x2,...xm对应的权值叫做回归系数，他们相乘的加权和就是我们要的概率结果。

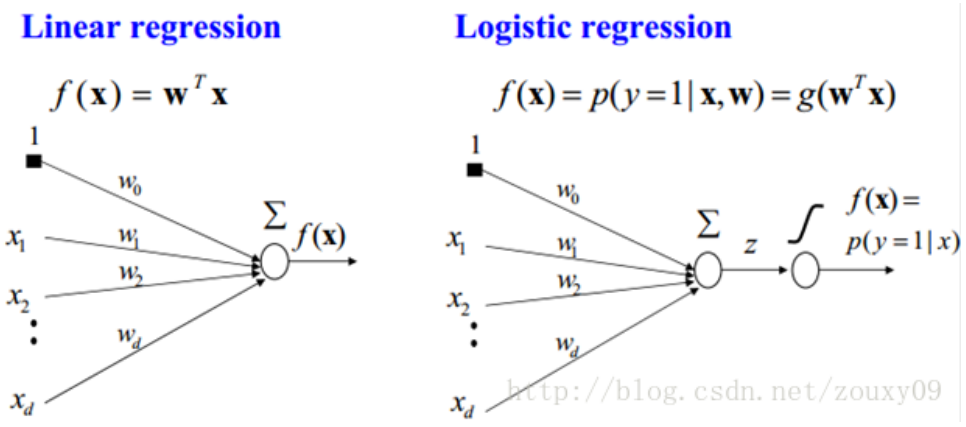
1.1 logistic回归 vs 线性回归

	线性回归	*逻辑回归*
目的	预测	分类
$y^{(i)}$	未知	{0,1}
函数	拟合函数	预测函数
参数计算方式	最小二乘	最大似然估计

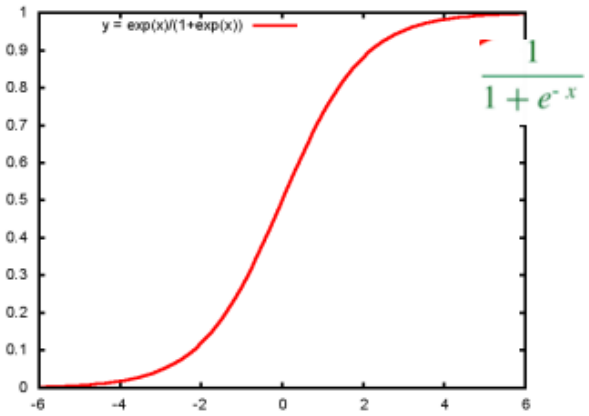
下面具体解释一下：

1. 拟合函数和预测函数什么关系呢？其实就是将拟合函数做了一个逻辑函数的转换，转换后使得$y^{(i)} \in (0,1)$;
2. 最小二乘和最大似然估计可以相互替代吗？回答当然是不行了。我们来看看两者依仗的原理：最大似然估计是计算使得数据出现的可能性最大的参数，依仗的自然是Probability。而最小二乘是计算误差损失。因此两者不可混淆（笑）。

它与线性回归的不同点在于：在线性回归的基础上，在特征到结果的映射中加入了一层sigmoid函数，将线性回归输出的很大范围的数，压缩到（0,1）之间。其目标函数也因此从差平方和函数变为对数损失函数。LR往往解决二分类问题，只是它和线性回归耦合太紧，所以也冠以回归的名字，如果求多分类，可把sigmoid换成softmax。逻辑回归得到一个离散的结果，线性回归得到一个连续的结果。



为什么使用sigmoid函数



① 可以对$(-\infty, +\infty)$结果，映射到$(0, 1)$之间，作为概率。

② $x < 0, sigmoid(x) < \frac{1}{2}; x > 0, sigmoid(x) > \frac{1}{2}$，可以将$\frac{1}{2}$作为决策边界。

③ 数学特性好，求导容易：$g^{'}(z) = g(z) \cdot (1 - g(z))$

1.2 博客加强理解

1、 为什么是逻辑回归?

都说线性回归用来做回归预测，逻辑回归用于做二分类，一个是解决回归问题，一个用于解决分类问题。但很多人问起逻辑回归和线性回归的区别，很多人会大喊一声（也可能是三声）：逻辑回归就是对线性回归做了一个压缩，将y 的阈值从$y \in (+\infty, -\infty)$压缩到$(0, 1)$。那么问题来了，问什么仅仅做一个简单的压缩，就将回归问题变成了分类问题？里面蕴含着本质?

首先要从数据说起，**线性回归的样本的输出，都是连续值，$y \in (+\infty, -\infty)$而，逻辑回归中$y \in \{0, 1\}$，只能取0和1。**对于拟合函数也有本质上的差别：

线性回归：$f(x) = \theta^T X = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

逻辑回归：$f(x) = p(y = 1 \mid x; \theta) = g(\theta^T X)$，其中，$g(z) = \frac{1}{1+e^{-z}}$

可以看出，线性回归的拟合函数，的确是对f(x)的输出变量y的拟合，而逻辑回归的拟合函数是对为1类的样本的概率的拟合。

2、那么，为什么要以1类样本的概率进行拟合呢，为什么可以这样拟合呢?

首先，logstic 函数的本质说起。若要直接通过回归的方法去预测二分类问题， y 到底是0类还是1类，最好的函数是单位阶跃函数。然而单位阶跃函数不连续（GLM 的必要条件），而 logsitic 函数恰好接近于单位阶跃函数，且单调可微。于是希望通过该复合函数去拟合分类问题：

$$y = \frac{1}{1+e^{-\theta^T X}}$$

于是有：

$$ln\frac{y}{1-y} = \theta^T X$$

发现如果我们假设 $y = p(y为1类 \mid x;\theta)$ 作为我们的拟合函数，等号左边的表达式的数学意义就是1类和0类的对数几率（log odds）。这个表达式的意思就是：用线性模型的预测结果去逼近1类和0类的几率比。于是，$\theta^T X = 0$就相当于是1类和0类的决策边界：

当$\theta^T X > 0$，则有$y > 0.5$；若$\theta^T X \to +\infty$，则$y \to 1$，即y 为1类;

当$\theta^T X < 0$，则有$y < 0.5$；若$\theta^T X \to -\infty$，则$y \to 0$，即 y 为0类。

这个时候就能看出区别来了，在线性回归中$\theta^T X$为预测值的拟合函数；而在逻辑回归中$\theta^T X = 0$为决策边界。

1.3 二项逻辑斯蒂回归模型和使用极大似然估计

定义 6.2（逻辑斯谛回归模型） 二项逻辑斯谛回归模型是如下的条件概率分布：

$$P(Y=1|x)=\frac{\exp(w\cdot x+b)}{1+\exp(w\cdot x+b)} \tag{6.3}$$

$$P(Y=0|x)=\frac{1}{1+\exp(w\cdot x+b)} \tag{6.4}$$

这里，$x\in \mathbf{R}^n$ 是输入，$Y\in\{0,1\}$ 是输出，$w\in \mathbf{R}^n$ 和 $b\in \mathbf{R}$ 是参数，w 称为权值向量，b 称为偏置，$w\cdot x$ 为 w 和 x 的内积.

对于给定的输入实例 x，按照式 (6.3) 和式 (6.4) 可以求得 $P(Y=1|x)$ 和 $P(Y=0|x)$. 逻辑斯谛回归比较两个条件概率值的大小，将实例 x 分到概率值较大的那一类.

有时为了方便，将权值向量和输入向量加以扩充，仍记作 w，x，即 $w=(w^{(1)},w^{(2)},\cdots,w^{(n)},b)^{\mathrm{T}}$，$x=(x^{(1)},x^{(2)},\cdots,x^{(n)},1)^{\mathrm{T}}$. 这时，逻辑斯谛回归模型如下：

$$P(Y=1|x)=\frac{\exp(w\cdot x)}{1+\exp(w\cdot x)} \tag{6.5}$$

$$P(Y=0|x)=\frac{1}{1+\exp(w\cdot x)} \tag{6.6}$$

6.1.3 模型参数估计

逻辑斯谛回归模型学习时，对于给定的训练数据集 $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$，其中，$x_i\in\mathbf{R}^n$，$y_i\in\{0,1\}$，可以应用极大似然估计法估计模型参数，从而得到逻辑斯谛回归模型.

设：　　$P(Y=1|x)=\pi(x)$，　$P(Y=0|x)=1-\pi(x)$

似然函数为

$$\prod_{i=1}^{N}[\pi(x_i)]^{y_i}[1-\pi(x_i)]^{1-y_i}$$

对数似然函数为

$$\begin{aligned}L(w)&=\sum_{i=1}^{N}[y_i\log\pi(x_i)+(1-y_i)\log(1-\pi(x_i))]\\&=\sum_{i=1}^{N}\left[y_i\log\frac{\pi(x_i)}{1-\pi(x_i)}+\log(1-\pi(x_i))\right]\\&=\sum_{i=1}^{N}[y_i(w\cdot x_i)-\log(1+\exp(w\cdot x_i)]\end{aligned}$$

对 $L(w)$ 求极大值，得到 w 的估计值.

这样，问题就变成了以对数似然函数为目标函数的最优化问题. 逻辑斯谛回归学习中通常采用的方法是梯度下降法及拟牛顿法.

假设 w 的极大似然估计值是 $\hat{w}$，那么学到的逻辑斯谛回归模型为

$$P(Y=1|x)=\frac{\exp(\hat{w}\cdot x)}{1+\exp(\hat{w}\cdot x)}$$

$$P(Y=0|x)=\frac{1}{1+\exp(\hat{w}\cdot x)}$$

2. 求解逻辑回归模型参数

LR最基本的学习算法实最大似然，假设我们有n个独立的训练样本{(x1, y1) ,(x2, y2), …, (xn, yn)}，y={0, 1}，那每一个观察到的样本(xi, yi)出现的概率是：

$$P(y_i,x_i)=P(y_i=1|x_i)^{y_i}(1-P(y_i=1|x_i))^{1-y_i}$$

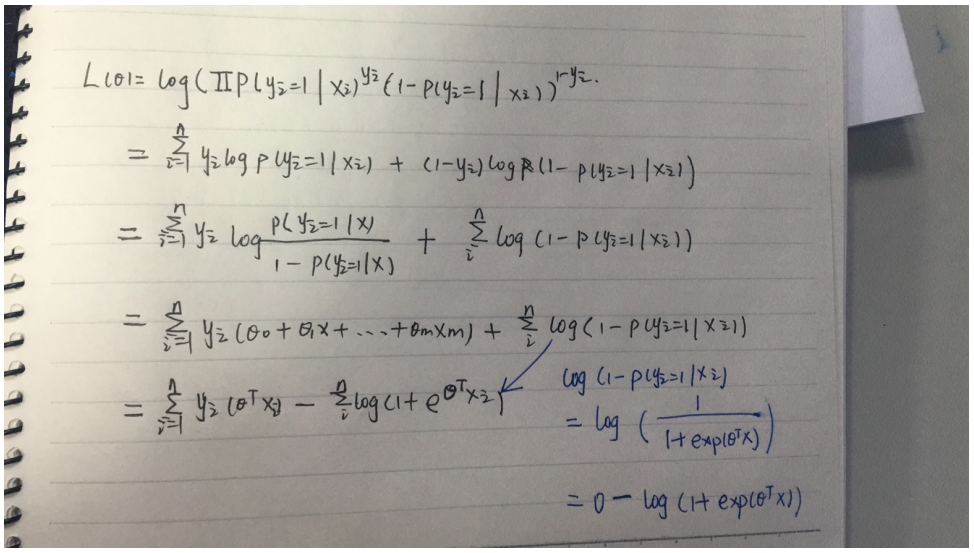
上面为什么是这样呢？当y=1的时候，后面那一项是不是没有了，那就只剩下x属于1类的概率，当y=0的时候，第一项是不是没有了，那就只剩下后面那个x属于0的概率（1减去x属于1的概率）。所以不管y是0还是1，上面得到的数，都是(x, y)出现的概率。那我们的整个样本集，也就是n个独立的样本出现的似然函数为（因为每个样本都是独立的，所以n个样本出现的概率就是他们各自出现的概率相乘）：

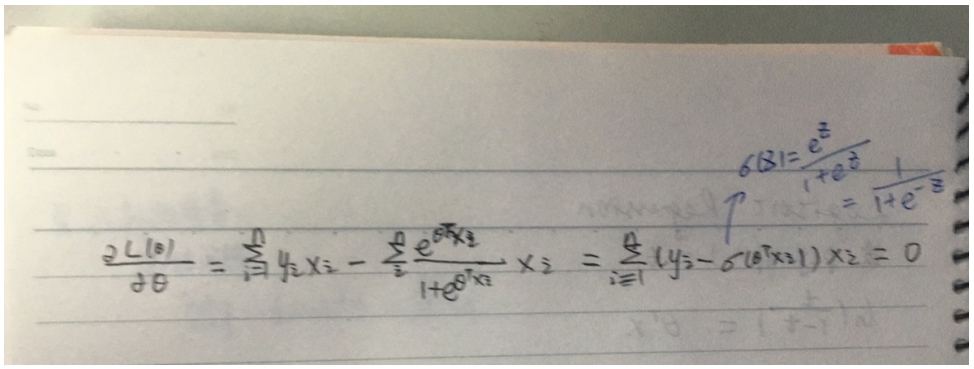
$$L(\theta)=\prod P(y_i=1|x_i)^{y_i}(1-P(y_i=1|x_i))^{1-y_i}$$

最大似然法就是求模型中使得似然函数最大的系数取值θ*，这个最大似然就是我们的代价函数（cost function）。

下一步，求解：

先尝试对上面的代价函数求导，看导数为0的时候可不可以解出来，也就是看有没有解析解。先变换下L(θ)，取自然对数，然后化简，得到：





无法解析求解，只能迭代求解。

2.1 梯度下降

迭代找到函数局部最优解，沿梯度负方向是损失下降最快的方向，所以

$$\theta^{t+1}=\theta^{t}-\alpha\frac{\partial L(\theta)}{\partial\theta}=\theta^{t}-\alpha\sum_{i=1}^{n}(y_i-\sigma(\theta^T x_i))\mathbf{x}_i$$

其中，参数 α 叫学习率，就是每一步走多远，这个参数蛮关键的。如果设置的太多，那么很容易就在最优值附加徘徊，因为你步伐太大了，可能会找不到全局最小值；但如果设置的太小，那收敛速度就太慢了。

注：因为本文中是求解的Logit回归的代价函数是似然函数，需要最大化似然函数。所以我们要用的是梯度上升算法。但因为其和梯度下降的原理是一样的，只是一个是找最大值，一个是找最小值。找最大值的方向就是梯度的方向，最小值的方向就是梯度的负方向。

另一个更普遍的写法：

逻辑回归损失函数：

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

简单回顾一下几个变量的含义：

表1 cost函数解释

x(i)	每个样本数据点在某一个特征上的值，即特征向量x的某个值
y(i)	每个样本数据的所属类别标签
m	样本数据点的个数
hθ(x)	样本数据的概率密度函数，即某个数据属于1类（二分类问题）的概率
J(θ)	代价函数，估计样本属于某类的风险程度，越小代表越有可能属于这类

我们的目标是求出θ，使得这个代价函数J(θ)的值最小，这里就需要用到梯度下降算法。

先来看看梯度下降算法中，自变量的迭代过程。表示如下

$$\theta_j = \theta_j - \alpha\cdot\frac{\partial}{\partial}J(\theta),(j=0,1,\cdots,n)\cdots\cdots\cdots(1)$$

可以看到，这是一个θ值不断迭代的过程，其中α是学习速率，就是θ的移动"步幅"，后面的偏导数数就是梯度，可以理解为cost函数在θ当前位置，对于j位置特征的下降速度。

【梯度的求导】

先来写一下大致的推导过程：

$$\frac{\partial J(\theta)}{\partial\theta_j} = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\frac{1}{h_\theta(x^{(i)})}\frac{\partial h_\theta(x^i)}{\partial\theta_j} - (1-y^{(i)})\frac{1}{1-h_\theta(x^{(i)})}\frac{\partial h_\theta(x^i)}{\partial\theta_j}\right)\cdots\cdots\cdots\cdots(1)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\frac{1}{g(\theta^T x^{(i)})} - (1-y^{(i)})\frac{1}{1-g(\theta^T x^{(i)})}\right)\cdot\frac{\partial g(\theta^T x^{(i)})}{\partial\theta_j}\cdots\cdots\cdots\cdots(2)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\frac{1}{g(\theta^T x^{(i)})} - (1-y^{(i)})\frac{1}{1-g(\theta^T x^{(i)})}\right)\cdot g(\theta^T x^{(i)})(1-g(\theta^T x^{(i)})x_j^{(i)}\cdots(3)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}(1-g(\theta^T x^{(i)}) - (1-y^{(i)})g(\theta^T x^{(i)})\right)\cdot x_j^{(i)}\cdots\cdots\cdots\cdots(4)$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - g(\theta^T x^{(i)})\right)\cdot x_j^{(i)}\cdots\cdots\cdots\cdots(5)$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)\cdot x_j^{(i)}\cdots\cdots\cdots\cdots(6)$$

稍微解释一下推导流程，便于理解。

(1)--->(2)：使用sigmoid函数的形式g(z)替换hθ(x)、提出公因子，放在式子尾

(2)--->(3)：这一步具体推导如下（使用了复合函数的求导公式）

$$\frac{\partial g(\theta^T x^{(i)})}{\partial \theta_j} = \frac{dg(z)}{dz} \cdot \frac{\partial(\theta^T x^{(i)})}{\partial \theta_j} \cdots\cdots\cdots\cdots g(z) = \frac{1}{1+e^{-z}} : 令 z=\theta^T x^{(i)}$$

$$= \frac{e^{-z}}{(1+e^{-z})^2} \cdot \frac{\partial(\theta_0 + \theta_1 x_1 + \cdots + \theta_j x_j + \cdots + \theta_m x_m)}{\partial \theta_j}$$

$$= \frac{e^{-z} - 1 + 1}{(1+e^{-z})^2} \cdot x_j^{(i)}$$

$$= \left[\frac{1}{1+e^{-z}} - \left(\frac{1}{1+e^{-z}}\right)^2\right] \cdot x_j^{(i)}$$

$$= \left(g(z) - g^2(z)\right) \cdot x_j^{(i)}$$

$$= g(\theta^T x^{(i)})\left(1 - g(\theta^T x^{(i)})\right) \cdot x_j^{(i)}$$

如果加上正则化

Cost function:

$$\rightarrow J(\theta) = -\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \quad | \theta_1, \theta_2, \ldots, \theta_n$$

同样使用梯度下降：

Gradient descent

Repeat {

$$\rightarrow \theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

$$\rightarrow \theta_j := \theta_j - \alpha \left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} - \frac{\lambda}{m}\theta_j\right] \leftarrow$$

$(j = \cancel{0}, 1, 2, 3, \ldots, n)$ $\theta_1, \ldots, \theta_n$

}

2.2 随机梯度下降SGD

梯度下降算法在每次更新回归系数的时候都需要遍历整个数据集（计算整个数据集的回归误差），该方法对小数据集尚可，但数据集很大特征很多时，计算复杂度太高。改进的方法是一次只用一个样本点（的回归误差）来更新回归系数，这个方法叫随机梯度下降算法。（**上面公式中的误差是针对于所有训练样本而得到的，而随机梯度下降的思想是根据每个单独的训练样本来更新权值**）。由于可以在新的样本到来的时候对分类器进行增量的更新，所以它属于在线学习算法。与在线学习相对应，一次处理整个数据集的叫“批处理”。

> 随机梯度下降算法的伪代码如下：
> 初始化回归系数为1
>
> 重复下面步骤直到收敛{
>
> 　　对数据集中每个样本
>
> 　　　　计算该样本的梯度

> 使用alpha xgradient来更新回归系数
>
> }
> 返回回归系数值

2.3 改进的随机梯度下降

为了使算法快速稳定收敛到某个值，我们要避免在每次迭代时系数的剧烈改变，（数据集可能并非完全线性可分，对不正常的样本点，在调整系数减少样本的分类误差时，会出现这种情况）。对SGD做两个改进避免波动问题：

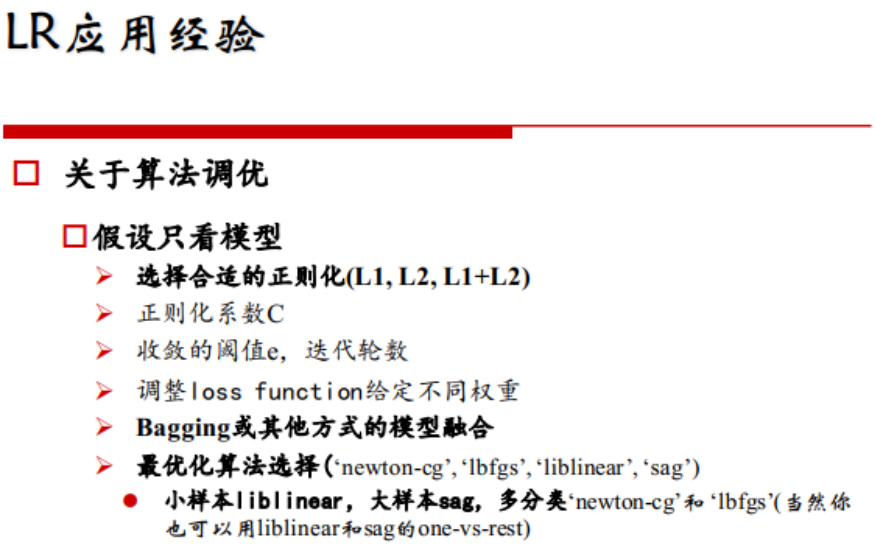
1）每次迭代时，调整更新步长alpha的值。随着迭代的进行，alpha越来越小，这会缓解系数的高频波动（也就是每次迭代系数改变得太大，跳的跨度太大）。为了避免alpha随着迭代不断减小到接近于0，约束其必须大于一个常数。

2）每次迭代，改变样本的优化顺序，也就是随机选择样本来更新回归系数，这样做可以减少周期性的波动，因为样本顺序的改变，使得每次迭代不再形成周期性。

（梯度下降等优化算法这里不讲，应该有专门一篇笔记）

3.LR的特点和应用

- LR < SVM/GBDT/RandomForest ?
  - 并不是，模型本身并没有好坏之分
    - LR能以概率的形式输出结果，而非只是0,1判定
    - LR的可解释性强，可控度高(你要给老板讲的嘛…)
    - 训练快，feature engineering之后效果赞
    - 因为结果是概率，可以做ranking model
    - 添加feature太简单…
  - 应用
    - CTR预估/推荐系统的learning to rank/各种分类场景
    - 某搜索引擎厂的广告CTR预估基线版是LR
    - 某电商搜索排序/广告CTR预估基线版是LR
    - 某电商的购物搭配推荐用了大量LR
    - 某现在一天广告赚1000w+的新闻app排序基线是LR



4.LR与SVM的关系

4.1 共同点

1. LR和SVM都可以处理分类问题，且一般都用于处理线性二分类问题（在改进的情况下可以处理多分类问题）

2. 两个方法都可以增加不同的正则化项，如l1、l2等，所以在很多实验中，两种算法的结果是很接近的。

4.2 不同点

1. LR是参数模型，SVM是非参数模型

2. 从目标函数来看，逻辑回归采用的是softmax loss，SVM采用的是hinge loss（损失函数，见https://blog.csdn.net/u010976453/article/details/78488279）

这两个损失函数的目的都是增加对分类影响较大的数据点的权重，减小于分类关系较小的数据点的权重。

3. SVM的处理方法是只考虑支撑向量，也就是和分类最相关的少数点，去学习分类器；而逻辑回归通过非线性映射，大大减小了离分类平面较远的点的权重。

4. LR模型相对简单，特别是大规模线性分类时比较方便，而SVM的理解和优化相对复杂，SVM转化为对偶问题后，分类只需要计算与少数几个支持向量的距离，在进行复杂核函数计算时优势很明显，能够大大简化模型和计算。