1. 把最小化叶子节点分数得到，为了得到最小化的叶子节点分数，我们需要计算**损失函数对于叶子节点的一阶到二阶导**
2. 计算Similarity Score
3. 计算子节点的Similarity Score,与父节点相减得到gain(需要最大化gain)
4. 重复以上...
5. 确定树结构后通过lambda,gamma进行剪枝（对应下图计算梯度和分裂节点的部分）

对于回归问题，一般来说我们最常用的损失函数就是

$$L(y_i, p_i) = \frac{1}{2}\sum(y_i - p_i)^2$$

一阶导gi （用到链式法则），可以理解为残差

$$g_i = \frac{d}{dp_i}\frac{1}{2}(y_i - pi)^2 = -(y_i - p_i)$$

二阶导hi （求没了已经） 全部为1

$$h_i = 1$$

写到这里，我们可以惊喜的发现，对于回归问题，XGBoost的最优化叶子节点式子就可以写成：

$$O_{value} = \frac{-(g_1 + g_2 + \ldots g_n)}{h_1 + h_2 + .. h_n + \lambda} = \frac{SumofResiduals}{NumberofResiduals + \lambda}$$

分子就是残差，分母就是叶子节点的样本数量+lambda

相应的目标函数就可以写为：

$$SimilaityScore = \frac{(g_1 + g_2 + \ldots g_n)^2}{h_1 + h_2 + .. h_n + \lambda} = \frac{SumofResiduals^2}{NumberofResiduals + \lambda}$$

分子就是残差和的平方，分母就是叶子节点的样本数量+lambda

无论是分类树还是回归树，CART都要选择**使子节点的GINI值或者回归方差最小**的属性作为分裂的方案。

每一次尝试去对已有的叶子加入一个分割：

$$Gain = \frac{1}{2}[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}] - \gamma$$

加入新叶子节点引入的复杂度代价

左子树分数

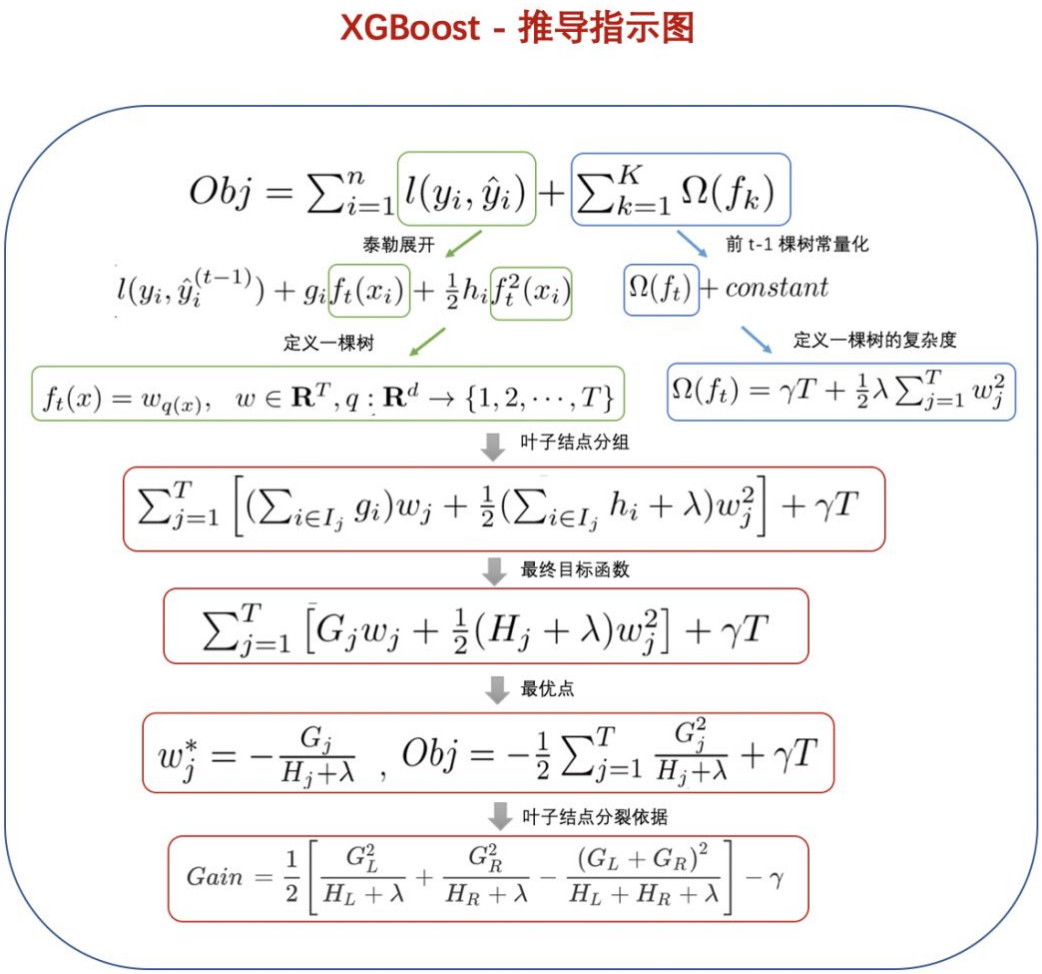
右子树分数

不分割我们可以拿到的分数

对于每次扩展，我们还是要枚举所有可能的分割方案，如何高效地枚举所有的分割呢？我假设我们要枚举所有x < a 这样的条件，对于某个特定的分割a我们要计算a左边和右边的导数和。
我们可以发现对于所有的a，我们只要做一遍从左到右的扫描就可以枚举出所有分割的梯度和GL和GR。然后用上面的公式计算每个分割方案的分数就可以了。

观察这个目标函数，大家会发现第二个值得注意的事情就是引入分割不一定会使得情况变好，因为我们有一个引入新叶子的惩罚项。优化这个目标对应了树的剪枝， 当引入的分割带来的增益小于一个阀值的时候，我们可以剪掉这个分割。大家可以发现，当我们正式地推导目标的时候，像计算分数和剪枝这样的策略都会自然地出现，而不再是一种因为heuristic（启发式）而进行的操作了。

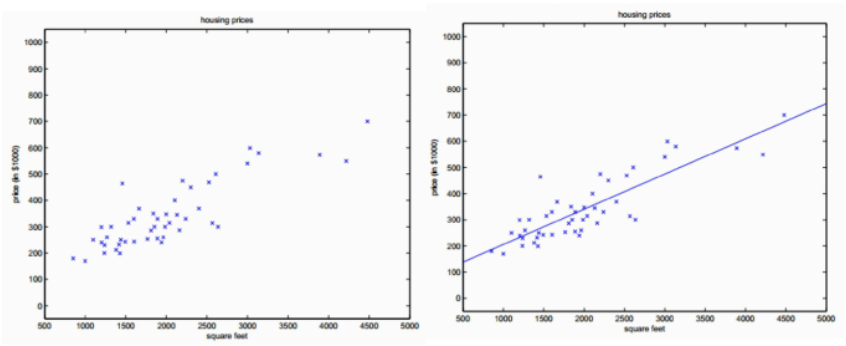
主要三种评价函数：多分类，排序，元素评价，分别定义在三个文件之中



线性回归

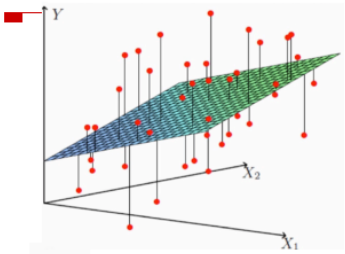
y=ax+b

y=ax+b



考虑两个变量

Living area (feet^2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮



$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

$$h_\theta(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$$

2. 优化方法

监督学习的优化方法=损失函数+对损失函数的优化

3. 损失函数

如何衡量已有的参数 θ 的好坏?

利用损失函数来衡量，损失函数度量预测值和标准答案的偏差，不同的参数有不同的偏差，所以要通过最小化损失函数，也就是最小化偏差来得到最好的参数。

映射函数：$h_\theta(x)$

损失函数：

$$J(\theta_0, \theta_1, ..., \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

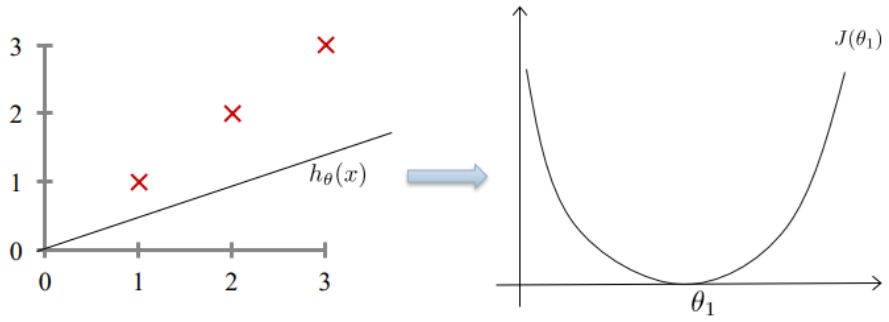
解释：因为有m个样本，所以要平均，分母的2是为了求导方便

损失函数：凸函数

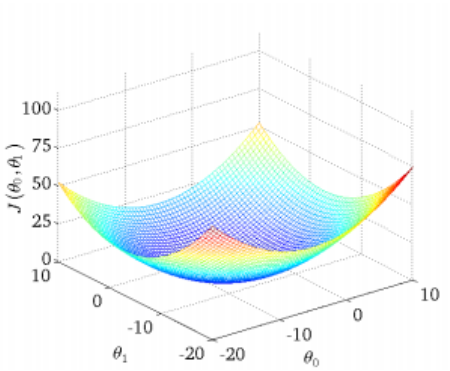
4. 损失函数的优化

损失函数如右图所示，是一个凸函数，我们的目标是达到最低点，也就是使得损失函数最小

凸函数



多元的情况

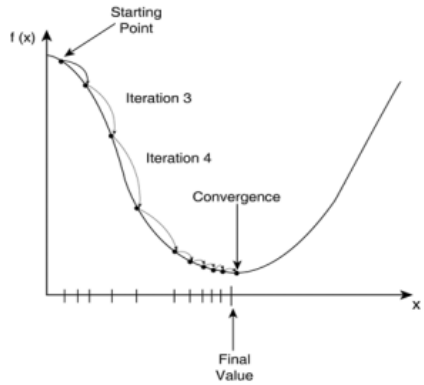


多元容易出现局部极值

求极值的数学思想，对公式求导=0即可得到极值，但是工业上计算量很大，公式很复杂，所以从计算机的角度来讲，求极值是利用梯度下降法。

梯度下降

- 逐步最小化损失函数的过程
- 如同下山，找准方向(斜率)，每次迈进一小步，直至山底



$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

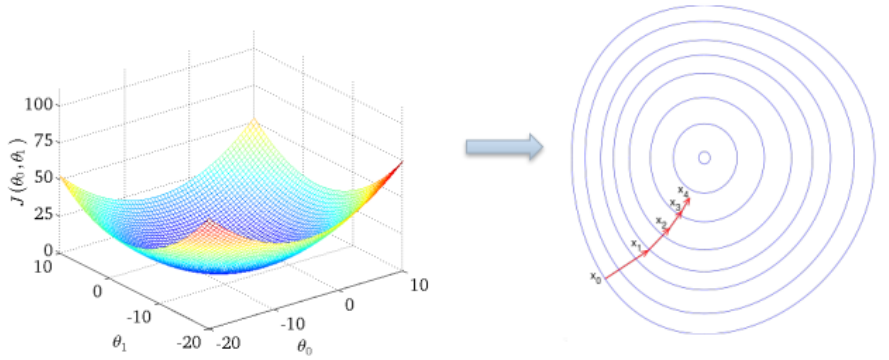
① 初始位置选取很重要

② 负梯度方向更新，二维情况下，函数变换最快的方向是斜率方向，多维情况下就成为梯度，梯度表示函数值增大的最快的方向，所以要在负梯度方向上进行迭代。

③ θ的更新公式如上图，每个参数 θ1,θ2...都是分别更新的

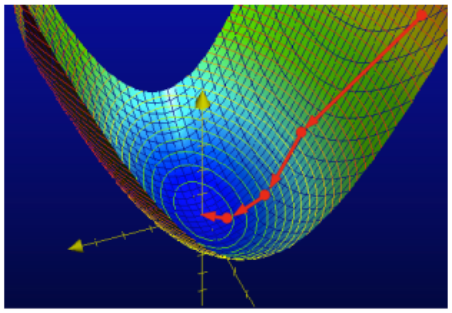
高维情况：梯度方向就是垂直于登高线的方向

- 梯度下降
  - 逐步最小化损失函数的过程
  - 如同下山，找准方向(梯度)，每次迈进一小步，直至山底



参数更新

- 梯度下降
  - 逐步最小化损失函数的过程
  - 如同下山，找准方向(梯度)，每次迈进一小步，直至山底



$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

repeat until convergence {

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

$$\theta_1 := \theta_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)\cdot x^{(i)}$$

}

对每个 θ 都更新:

- 梯度下降
  - 假如现在有n个特征/变量x_j(j=1…n)

repeat until convergence {

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1)$$

}

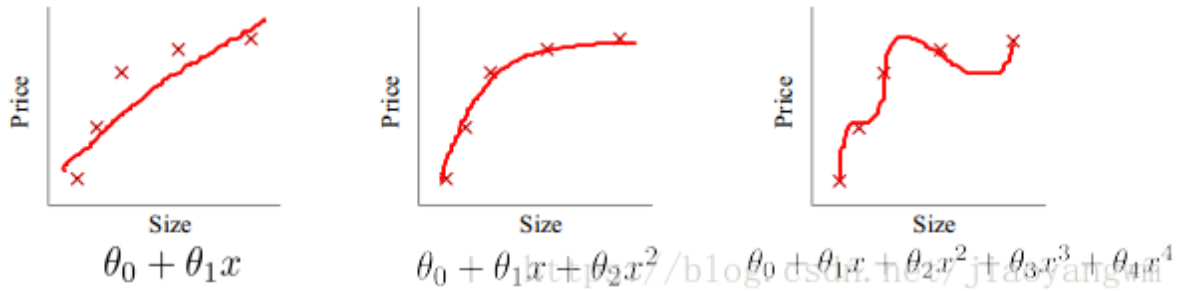
$$\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_2^{(i)}$$

…

5. 过拟合和欠拟合



过拟合的原因：

① 如果我们有很多的特征或模型很复杂，则假设函数曲线可以对训练样本拟合的非常好，学习能力太强了，但是丧失了一般性。
② 眼见不一定为实，训练样本中肯定存在噪声点，如果全都学习的话肯定会将噪声也学习进去。

过拟合造成什么结果：

过拟合是给参数的自由空间太大了，可以通过简单的方式让参数变化太快，并未学习到底层的规律，模型抖动太大，很不稳定，variance变大，对新数据没有泛化能力。

6. 利用正则化解决过拟合问题

正则化的作用：
① 控制参数变化幅度，对变化大的参数惩罚
② 限制参数搜索空间
添加正则化的损失函数：

$$J(\theta_0, \theta_1, ..., \theta_n) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

m：样本有m个
n：n个参数，对n个参数进行惩罚
λ \lambda λ：对误差的惩罚程度，λ \lambda λ越大对误差的惩罚越大，容易出现过拟合，λ \lambda λ越小，对误差的惩罚越小，对误差的容忍度越大，泛化能力好。

逻辑回归
监督学习，解决二分类问题。

分类的本质：在空间中找到一个决策边界来完成分类的决策

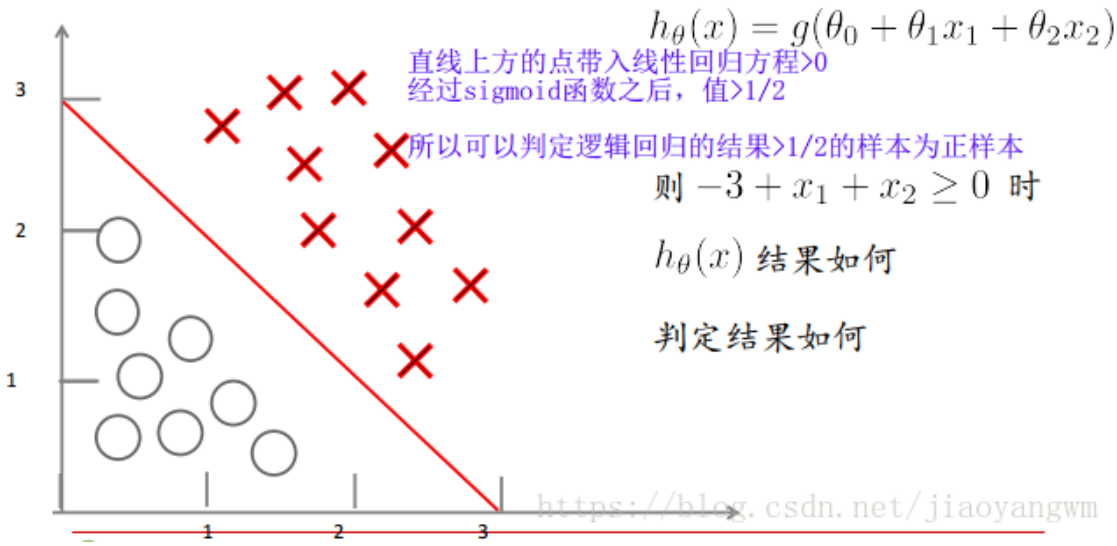
逻辑回归：线性回归可以预测连续值，但是不能解决分类问题，我们需要根据预测的结果判定其属于正类还是负类。所以逻辑回归就是将线性回归的 $(-\infty, +\infty)$结果，通过sigmoid函数映射到 (0,1) 之间。

线性回归决策函数：$h_\theta(x) = \theta^T x$

将其通过sigmoid函数，获得逻辑回归的决策函数：$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

sigmoid函数的作用：

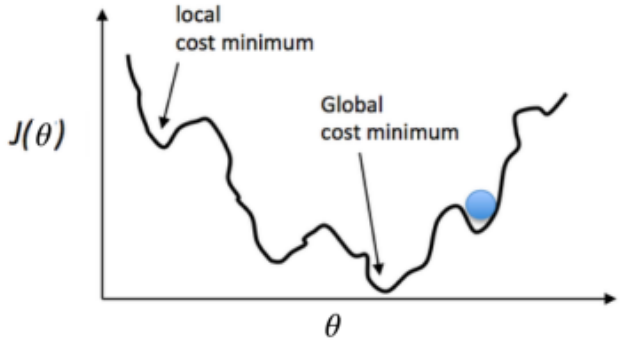
1. 将($-\infty$, $+\infty$)结果映射到（0,1）之间，作为概率；
2. x < 0 , sigmoid(x) < 1/2 ；x > 0 , sigmoid(x) > 1/2 ,可以将1/2作为决策边界；
3. 数学特性号，求导容易：g'(z) = g(z) * (1-g(z))。



3.1 逻辑回归的损失函数

线性回归的损失函数为平方损失函数，如果将其用于逻辑回归的损失函数，则其数学特性不好，有很多局部极小值，难以用梯度下降法求最优。

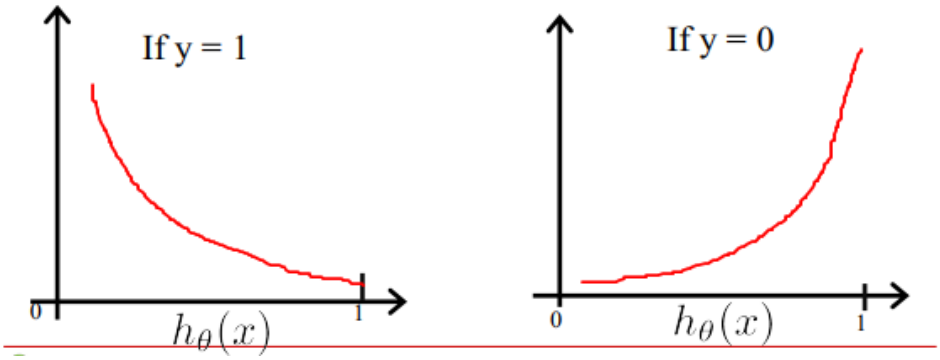
$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$



逻辑回归损失函数：对数损失函数

损失函数

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



解释：如果一个样本为正样本，那么我们希望将其预测为正样本的概率p越大越好，也就是决策函数的值越大越好，则logp越大越好，逻辑回归的决策函数值就是样本为正的概率；

如果一个样本为负样本，那么我们希望将其预测为负样本的概率越大越好，也就是(1-p)越大越好，即log(1-p)越大越好。

为什么要用log：

样本集中有很多样本，要求其概率连乘，概率为(0,1)间的数，连乘越来越小，利用log变换将其变为连加，不会溢出，不会超出计算精度。

逻辑回归损失函数：

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

别忘了正则化项目

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\log(h_\theta(x^{(i)}) + (1 - y^{(i)})\log 1 - h_\theta(x^{(i)})\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

- 梯度下降求最小值

$$J(\theta) = \left[-\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log \left(h_\theta(x^{(i)}) + (1 - y^{(i)}) \log 1 - h_\theta(x^{(i)}) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2 \right.$$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

3.2 逻辑回归实现多分类

构建多个分类器，sigmoid-> softmax

四、LR的特点

可解释性高，工业中可控度高。

- 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

五、 为什么逻辑回归比线性回归好

虽然逻辑回归能够用于分类，不过其本质还是线性回归。它仅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（非线性）映射，即先把特征线性求和，然后使用sigmoid函数来预测。

这主要是由于线性回归在整个实数域内敏感度一致，而分类范围，需要在[0,1]之内。而逻辑回归就是一种减小预测范围，将预测值限定为[0,1]间的一种回归模型，其回归方程与回归曲线如下图所示。逻辑曲线在z=0时，十分敏感，在z>>0或z<<0处，都不敏感，将预测值限定为(0,1)。

1.LR在线性回归的实数范围输出值上施加sigmoid函数将值收敛到0~1范围, 其目标函数也因此从差平方和函数变为对数损失函数, 以提供最优化所需导数（sigmoid函数是softmax函数的二元特例, 其导数均为函数值的f*(1-f)形式）。请注意, LR往往是解决二元0/1分类问题的, 只是它和线性回归耦合太紧, 不自觉也冠了个回归的名字(马甲无处不在). 若要求多元分类,就要把sigmoid换成大名鼎鼎的softmax了。

2.首先逻辑回归和线性回归首先都是广义的线性回归，其次经典线性模型的优化目标函数是最小二乘，而逻辑回归则是似然函数，另外线性回归在整个实数域范围内进行预测，敏感度一致，而分类范围，需要在[0,1]。逻辑回归就是一种减小预测范围，将预测值限定为[0,1]间的一种回归模型，因而对于这类问题来说，逻辑回归的鲁棒性比线性回归的要好。

3.逻辑回归的模型本质上是一个线性回归模型，逻辑回归都是以线性回归为理论支持的。但线性回归模型无法做到sigmoid的非线性形式，sigmoid可以轻松处理0/1分类问题。

count:poisson

泊松回归是最常用的计数回归模型，它是广义线性回归模型的一种，其因变量服从泊松分布。泊松回归常用来对非负整数随机变量建模。

泊松分布

设随机变量$Y \sim Po(\mu)$,则Y的概率密度函数为：

$$f(y|\mu) = \frac{e^{-\mu}\mu^y}{y!}, y = 0, 1, 2, \cdots$$

泊松分布一个重要的特征是：唯一的参数μ，即是均值又是方差，即

$$\mu = E(Y) = Var(Y)$$

泊松回归

在泊松回归中，解释变量对响应变量的平均值进行建模。因为响应变量的均值必须是正的但解释变量的线性组合

$$\eta = \beta' X_i = \sum_{j=0}^{p} \beta_{ij} x_{ij}$$

其中$x_{i0} = 1, i = 1, 2, \cdots, n, j = 0, 2, \cdots, p$，可取任何值，因此我们需要寻找参数$\mu$的联结函数保证回归等式成立。标准的联结函数是自然对数，即

$$\log(\mu) = \eta$$

将上述关系带入泊松分布的概率密度函数中，其对数似然函数为：

$$l(\beta) \propto \sum_{i=1}^{n} (y_i \beta' X_i - e^{\beta' X_i})$$

虽然对β求导后不能获得关于β的解析表达式，但$-l(\beta)$为凸函数，因此我们可以采用梯度下降的方法获得β的最优值。

任务最终需要什么样格式的数据，包括训练数据，验证数据，标签映射表等等。

1. 数据打包格式

```
line1: 算子1特征值+算子1运行时间
line2:
......
```

2. 数据形式

单算子输入，有利于增加训练数据量和样本丰富度。
输入特征的取值是**缓存、数学计算、DMA次数和卷积次数**，**输出单算子运行的时间**

3. 标签映射表

（用于增加模型的可解释性）

```
0                2dconv
1                dwconv
2                add
......
```