1. 不平衡问题怎么去处理？
    1. 数据层面：上采样（或SMOTE） & 下采样；
    2. 算法层面：训练过程中加入代价敏感矩阵（举例AdaCost）；
    3. 集成层面：easyEnsemble + 增大差异性的集成；

2. GBDT和RF里面都是基于CART的树（二叉树），不是那种最简单的多叉树，同时RF的树不做剪枝；但是必须做列层面的随机抽样，这样做不仅是为了提升效率，也是为了增大差异性、提升集成效果~~（防止过拟合）~~。

3. AUC的定义：某个学习器的ROC曲线下的面积，AUC当然是越大越好，用来衡量偷电比赛那种排序问题的一个排序质量（MAP也是）。同时，AUC的指标也用于不平衡问题下面，因为AUC考虑了少数类的分类精度，对不平衡较为不敏感。

4. 分类和回归任务的常用指标：回归，MSE；分类：错误率和精度、P值R值F值、ROC与AUC、代价敏感错误率。各种线下的评估方法记得一定要做分层采样，保证每一份数据集都是和原来D的标记分布一致。补充一点，留出法和交叉验证法能用分层抽样保证样本分布的一致性，但是自助法是不能保证标记分布的一致性的，这也是自助法的一个缺点。

5. 随机森林里面的树都是CART树、不考虑剪枝操作，好处就是随机森林的随机采样性已经避免了过拟合的可能性，剪枝仅仅是为了降低模型复杂度防止过拟合，再剪枝估计就要欠拟合且训练时间也会变长。RF里面的一些套话要会说：来降低子模型间的相关度、降低整体的一个方差（RF是Bagging的一种）。行采样默认有1/3的样本是采不到的，这里有一个极限公式在里面，面试的时候记得要说一下；列采样的话是默认是log2d的一个大小，这个在Briedmen2001的论文上有。

6. L1和L2正则化的区别？L1为什么能保证稀疏？
    1. https://www.zhihu.com/question/37096933 回答为什么L1范数更容易获得稀疏解？要注意主要去看PRML的解释：主要看知乎自己收藏的三个即可。

2. http://blog.csdn.net/jinping_shi/article/details/52433975 (L1范数和L2范数和L1、L2正则化含义相同)，这个是谷歌排名第一的很好的帖子，很不错的CSDN的帖子。
3. http://blog.sciencenet.cn/blog-253188-968555.html (Rank 3 Google)

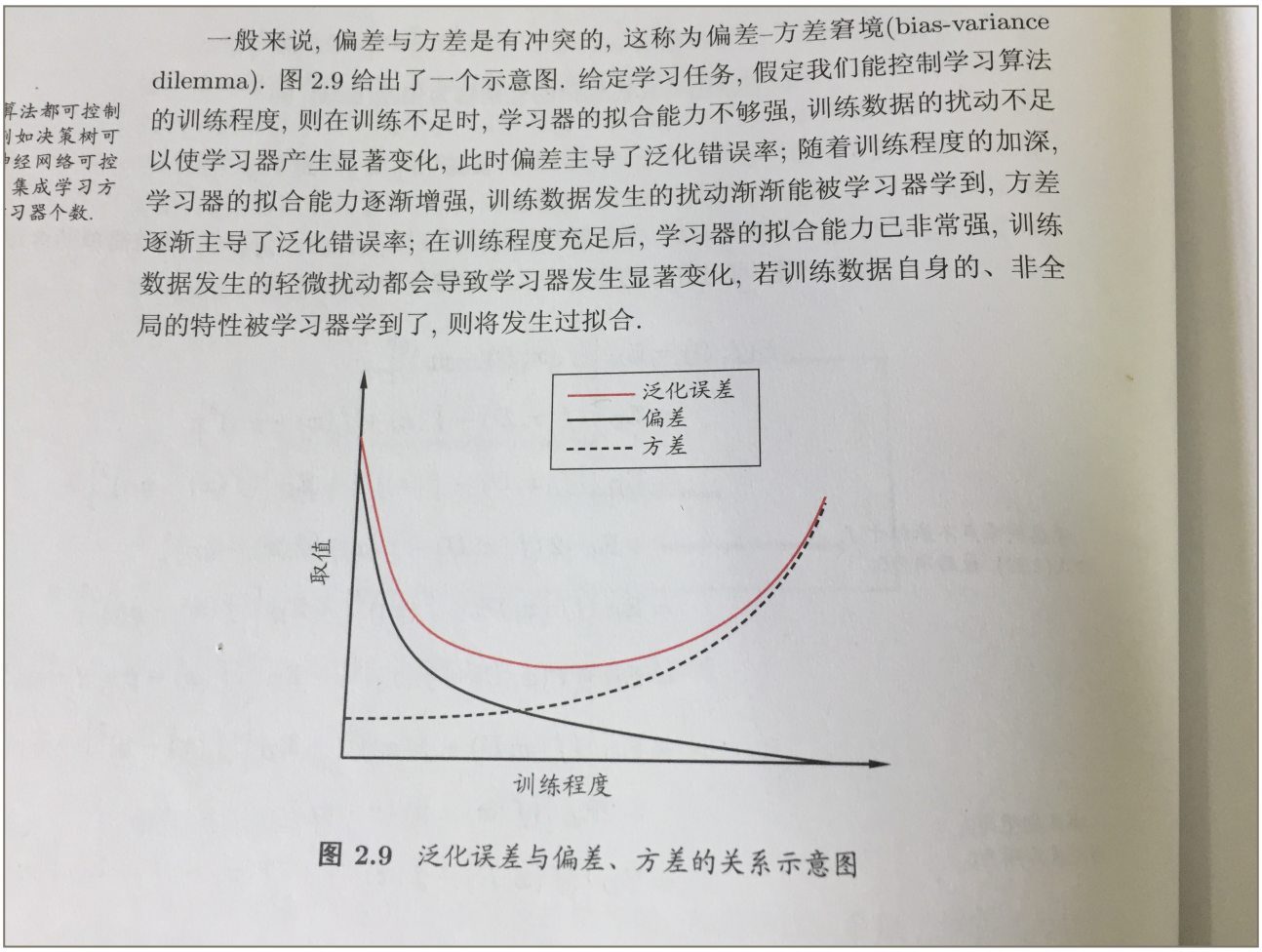
7. 偏差和方差的tradeoff (知乎一问题 + 周志华书图)
   1. https://www.zhihu.com/question/20448464
   2. 天池辛超的天池视频内容

8. Bagging和Boosting的区别 (为什么前者是降低方差、后者是降低偏差)
   1. https://www.zhihu.com/question/26760839
   2. Bagging每次减少了outlier的采样 (原始数据集如果outlier很多的话，会让Model记住太多噪声导致过拟合；bagging随机选取data的subset，outlier因为比例比较低，参与model training的几率也比较低，所以bagging降低了outliers和noise对model的影响，所以降低了variance (这个从outlier的角度回答，所以很新颖)。

9. GBDT、XGBoost、RF的区别与联系 (全家桶)
   1. https://www.zhihu.com/question/41354392 the answer of weapon
   2. ~~http://www.jianshu.com/p/005a4ebac775 讲GBDT的好的文章，尤其是提升树和梯度提升树，跟自己的想法是一样的，GBDT只要去回答拟合残差就行了。~~
   3. XGBoost的目标函数加了(正则化 + 二阶泰勒展开)，特征并行化很快，加入列抽样和对缺失值的处理、节点的分裂方式不同：GBDT还是用的CART树但XGBoost用的是d自己的打分函数做分裂。

10. 机器学习中的树模型有什么优势？为什么做RF + XGB的融合？
    1. 天池wepon和辛超录制的o2o的教学视频里面讲到的；
       1. 数据的特点：混合类型数据、大量缺失值、含有离群点。
       2. 模型的特点：a、善于处理混合类型特征 (对于连续和离散的特征有一套统一的理论)、善于处理缺失值 (自动学习缺失值的分裂方向)、对离群点鲁棒RF；b、工程上来说很容易并行化&可解释性强、有特征选择作用、Bagging树降低方差、Boosting树降低偏差。

2. 后半个问题回答偏差-方差tradeoff（如下西瓜书图）。
    1. 上来要先说泛化误差的构成，然后要对照这个图说。



算法都可控制
如决策树可
经网络可控
集成学习方
习器个数.

一般来说，偏差与方差是有冲突的，这称为偏差-方差窘境(bias-variance dilemma). 图 2.9 给出了一个示意图. 给定学习任务，假定我们能控制学习算法的训练程度，则在训练不足时，学习器的拟合能力不够强，训练数据的扰动不足以使学习器产生显著变化，此时偏差主导了泛化错误率；随着训练程度的加深，学习器的拟合能力逐渐增强，训练数据发生的扰动渐渐能被学习器学到，方差逐渐主导了泛化错误率；在训练程度充足后，学习器的拟合能力已非常强，训练数据发生的轻微扰动都会导致学习器发生显著变化，若训练数据自身的、非全局的特性被学习器学到了，则将发生过拟合.

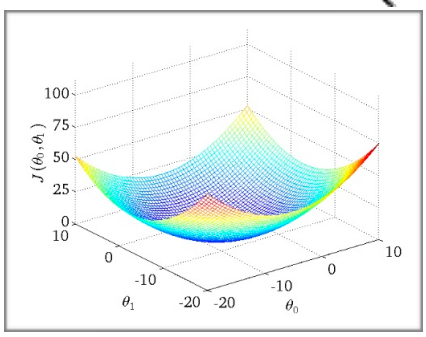
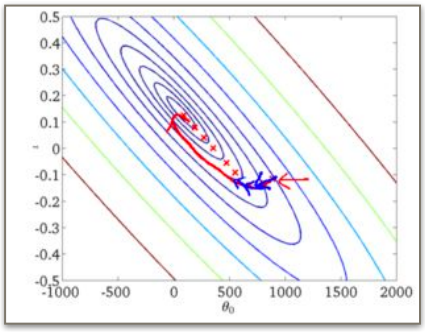
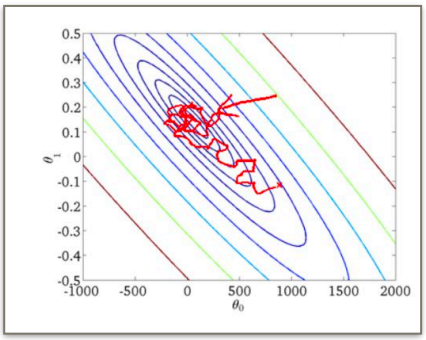
图 2.9 泛化误差与偏差、方差的关系示意图

11. LR逻辑回归里为什么用sigmoid函数？（这个问题下要深入理解指数族分布、和广义线性模型GLM）
    1. https://blog.csdn.net/u011467621/article/details/48197943 GLM和指数族分布讲的特别好，这个也是NG讲义上的所以很靠谱。
    2. 统计中很多熟悉的概率分布都是指数族分布的特定形式，如伯努利分布，高斯分布，多项分布（multionmal），泊松分布等；同时LR属于广义线性模型的一种，广义线性模型有三个假设h(x) = E(y) = p = (1 / e - N) = (1/theata.T.theta)。

12. 关于特征选择
    1. 主要是三种方法：过滤式（特征列的方差大小、标签&特征列的皮尔逊相关系数）、包裹式（用树模型的feature importance）、嵌入式（L1正则化）。

13. 机器学习中常见的损失函数
    1. http://www.csuldw.com/2016/03/26/2016-03-26-loss-function/（还是Google最靠谱 排序很重要）。AdaBoost是指数损失函数，SVM（线性支持向量机）是合页损失函数，LR是交叉熵损失函数。Huber Loss是为了克服MSE对异常值敏感的缺点，底下的参数为1。
    2. https://zhuanlan.zhihu.com/p/25765735 后面总结。

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \le \delta, \\ \delta |y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

14. AdaBoost和GBDT的区别？
    1. https://www.zhihu.com/question/54626685下Frankenstein的回答就完全足够了，都是Boosting的两种典型的机器学习方法，相同点最后他们都是加法模型。

15. 怎么去防止过拟合？
    1. https://www.zhihu.com/question/59201590/answer/166205675（自己还回答过哈哈，这个是很多大牛关注的问题）
    2. 自己总结一些：数据层面上，多加一点数据；模型层面上，模型本身不能太复杂加正则化防止过拟合，做好模型的集成，Bagging操作要做

好；还有就是工程上的：比如选好正确的验证集、做好cv交叉验证、做好特征选择，模型选择等。

16. LR和SVM各自的优缺点和适用场景？（高频、一定要弄会）

1. 看统计学习方法上SVM的那章（主要看每章小结部分），看牛客网关于LR的PPT。
2. 最本质是他们的损失函数不同，LR是对数损失而SVM是合页损失函数；LR可以给出每个点属于每一类的概率，而SVM是非概率的，一个是基于统计的方法，一个基于几何的方法；支持向量机只考虑局部的边界线附近的点（支持向量），而逻辑回归考虑所有的样本点，所以线性SVM不直接依赖于数据分布，分类平面不受一类点影响，LR则受所有数据点的影响对不平衡数据先做balance；同样是线性分类，如果异常点较多无法剔除，首先LR中每个样本都有贡献，最大似然后会自动压制异常的贡献，SVM + 软间隔对异常却比较敏感，因为其训练只需要支持向量，有效样本本来就不高，一旦被干扰，预测结果会难以预料。

17. XGBoost怎么处理缺失值（决策树如何处理缺失值）？如何实现并行化？

1. https://www.zhihu.com/question/34867991/answer/224122110 ~~中TomHall的回答。其实就是很简单的两个分支都去试取Gain比较大的分支作为默认分支。~~
2. 算法首先忽略带缺失值的数据，像正常情况下一样将前两种数据分别计算并导流到左子树与右子树，然后 1.将带缺失值的数据导向左子树，计算出这时候模型的Objective_L；2. 接着将带缺失值的数据导向右子树，计算出这时候模型的Objective_R；3. 最后比较Objective_L和Objective_R。假设Objective_L更小，那么在预测时所有变量A是缺失值的数据在这个节点都会被导向左边当做A <= 0去处理。
3. xgboost的并行不是tree粒度的并行，xgboost也是一次迭代完才能进行下一次迭代的（第t次迭代的代价函数里包含了前面t-1次迭代的预测值）。xgboost的并行是在特征粒度上的。我们知道，决策树的学习最耗时的一个步骤就是对特征的值进行排序（因为要确定最佳分割点），

xgboost在训练之前，预先对数据进行了排序，然后保存为block结构，后面的迭代中重复地使用这个结构，大大减小计算量。这个block结构也使得并行成为了可能，在进行节点的分裂时，需要计算每个特征的增益Gain，最终选增益最大的那个特征去做分裂，那么各个特征的增益计算就可以开多线程进行。（Every Column Block）

4. 关于并行化的问题（回答为什么比传统的GBDT要快）：每个特征的特征增益的Gain计算的并行化 + 每个特征值对应的各样本梯度以及Feature Value的Pre-sorted可以事先计算并保存为Block结构 + 工程上的并行化 + ~~（可并行的近似直方图算法）~~。

18. 机器学习里Loss的各种优化方法及其优缺点

1. BGD收敛的过程很稳，每一步都是朝最优的方向去，所以会收敛到全局最优（如果原函数是凸函数）。但其参数更新很慢，每次参数更新需要用到所有的训练集，并且不支持在线学习（全部训练集梯度均值能代表样本总体）；SGD最后往往不会收敛到全局最优，而是在最优值的附近，同时收敛的过程不稳定，总体的方向会走很多弯路虽然大致是朝着最优方向走的，但是SGD的收敛是要比BGD更快的，能在线学习（单个样本不能代表样本总体的分布，利用不同样本进行梯度更新可能会相互抵消从而SGD在整个训练集上不能收敛：Tencent面试解决这种不能收敛的办法是增大batch Size用MBGD，减小learning rate）。
2. 牛顿法（求零点转化为导数为0去求解，梯度下降是直接优化求解）、拟牛顿法也常用在逻辑回归的优化中，拟牛顿法是对牛顿法的改进，直接用正定矩阵来近似海塞矩阵的逆矩阵因为求逆操作的复杂度很高关于learning rate，如果调太大那么参数就会在最优值附近震荡不能收敛，所以一开始要大一点后面要调小一点但也不可能减为0，否则会有一直不收敛的情况；对SGD来说，如果不收敛还可能是Batch Size太小所以要增大Batch Size。

关于牛顿法和梯度下降法的效率对比：

从本质上去看，牛顿法是二阶收敛，梯度下降是一阶收敛，所以牛顿法就更快。如果更通俗地说的话，比如你想找一条最短的路径走到一个盆地的最底部，梯度下降法每次只从你当前所处位置选一个坡度最大的方向走一步，牛顿法在选择方向时，不仅会考虑坡度是否够大，还会考虑你走了一步之后，坡度是否会变得更大。所以，可以说牛顿法比梯度下降法看得更远一点，能更快地走到最底部。（牛顿法目光更加长远，所以少走弯路；相对而言，梯度下降法只考虑了局部的最优，没有全局思想。）

根据wiki上的解释，从几何上说，牛顿法就是用一个二次曲面去拟合你当前所处位置的局部曲面，而梯度下降法是用一个平面去拟合当前的局部曲面，通常情况下，二次曲面的拟合会比平面更好，所以牛顿法选择的下降路径会更符合真实的最优下降路径。

牛顿法的优缺点总结：

优点：二阶收敛，收敛速度快；

缺点：牛顿法是一种迭代算法，每一步都需要求解目标函数的Hessian矩阵的逆矩阵，计算比较复杂。

拟牛顿法的本质思想是改善牛顿法每次需要求解复杂的Hessian矩阵的逆矩阵的缺陷，它使用正定矩阵来近似Hessian矩阵的逆，从而简化了运算的复杂度。拟牛顿法和最速下降法一样只要求每一步迭代时知道目标函数的梯度。通过测量梯度的变化，构造一个目标函数的模型使之足以产生超线性收敛性。这类方法大大优于最速下降法，尤其对于困难的问题。另外，因为拟牛顿法不需要二阶导数的信息，所以有时比牛顿法更为有效。如今，优化软件中包含了大量的拟牛顿算法用来解决无约束，约束，和大规模的优化问题。

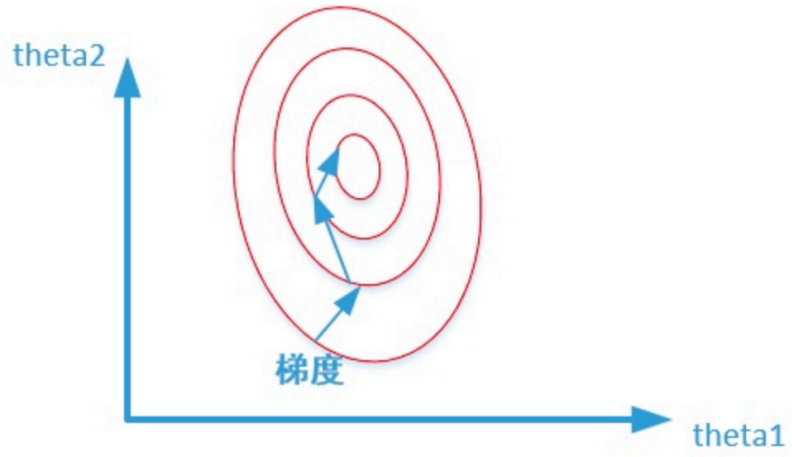
19. 关于线性回归和逻辑回归（抱佛脚资料）
    1. 去看MAC本地自带的<回归.pdf>的东西，这个是关于LR最好的资料。
    2. 《统计学习方法》上的每章小结。
    3. 看打印出来的文档资料（算法整理的资料）。

20. 特征工程中的一些盲点回顾
    1. https://www.zhihu.com/question/29316149（特征工程到底是什么）
    2. 归一化的作用：波士顿地区房价的例子（房间数 + 面积）；归一化的方法：max-min、Z-score；树模型不用做归一化：因为不需要考虑特征的value只需考虑split的位置。
    3. http://www.cnblogs.com/LBSer/p/4440590.html很不错的解释。

数据没有归一化的表达式，可以为：

$$J = (3\theta_1 + 600\theta_2 - y_{correct})^2$$

造成图像的等高线为类似椭圆形状，最优解的寻优过程就是像下图所示：

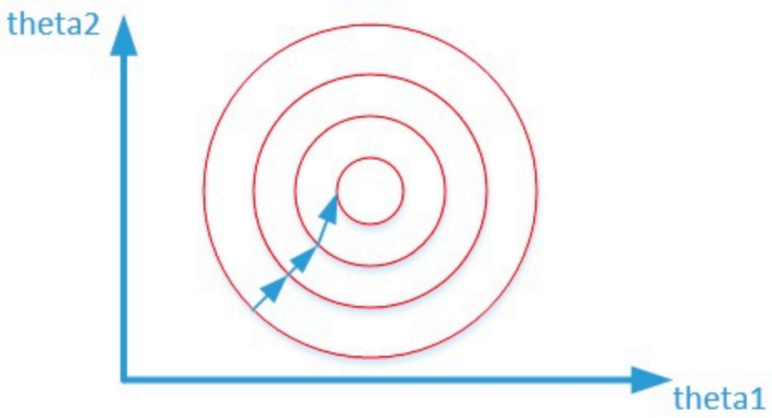


而数据归一化之后，损失函数的表达式可以表示为：

$$J = (0.5\theta_1 + 0.55\theta_2 - y_{correct})^2$$

其中变量的前面系数几乎一样，则图像的等高线为类似圆形形状，最优解的寻优过程像下图所示：

其中变量的前面系数几乎一样，则图像的等高线为类似圆形形状，最优解的寻优过程像下图所示：



从上可以看出，数据归一化后，**最优解的寻优过程明显会变得平缓，更容易正确的收敛到最优解。**

这也是数据为什么要归一化的一个原因。

21. 判别模型和生成模型的区别

29 人赞同了该回答

简单地说，对于监督学习，预测时，一般都是在求 $p(Y|X)$ 生成模型： 从数据中学习联合概率分布 $p(X,Y)$ ，然后利用贝叶斯公式求： $p(Y|X) = \frac{P(X,Y)}{\Sigma P(X,Y_i)}$ ；这类典型的模型包括：朴素贝叶斯、LDA、HMM

判别模型：直接学习 $P(Y|X)$ ， 它直观输入什么特征X，就直接预测出最可能的 Y ；典型的模型包括：LR, SVM,CRF,Boosting,Decision tree....

编辑于 2014-04-25

78 人赞同了该回答

假设你现在有一个分类问题，x是特征，y是类标记。用生成模型学习一个联合概率分布P（x，y），而用判别模型学习一个条件概率分布P（y|x）。

用一个简单的例子来说明这个这个问题。假设x就是两个（1或2），y有两类（0或1），有如下如下样本（1，0）、（1，0）、（1，1）、（2，1）

则学习到的联合概率分布（生成模型）如下：

-------0------1----

--1-- 1/2---- 1/4

--2-- 0 ------1/4

而学习到的条件概率分布（判别模型）如下：

-------0------1----

--1-- 2/3--- 1/3

--2-- 0--- 1

在实际分类问题中，判别模型可以直接用来判断特征的类别情况，而生成模型，需要加上贝耶斯法则，然后应用到分类中。但是，生成模型的概率分布可以还有其他应用，就是说生成模型更一般更普适。不过判别模型更直接，更简单。