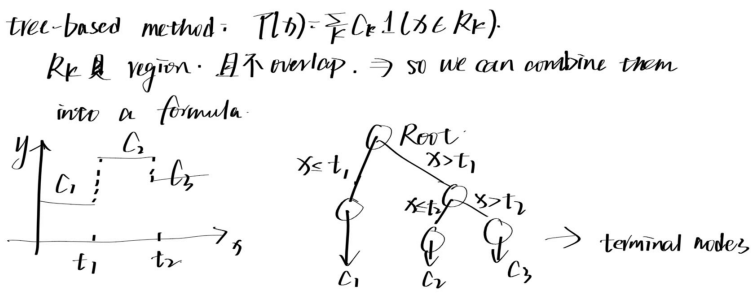
Tree-based method

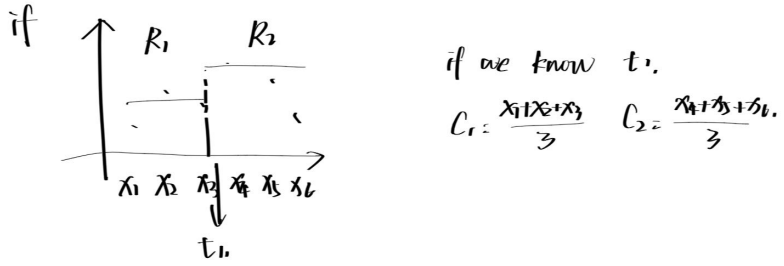
- tree based method --> interaction detection: x1 在有无x2存在的情况下对Y的effect不同，即effect of x1 on Y depends on x2
- 什么是树模型: a piece wise constant model
  - function



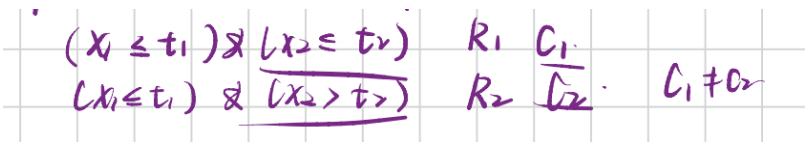
  - loss function:如果我们知道在哪里split，就很容易计算loss function，但是怎么找到split point呢?

$$\mathrm{RSS}_{new} = \sum_{i \in R_1(j,s)} (y_i - \hat{y}_1)^2 + \sum_{i \in R_2(j,s)} (y_i - \hat{y}_2)^2$$

    - example



    - one dimension: 找到每个区间之间的middle points ,然后寻找loss function最小的
    - multi dimension: 对每个dimension都要执行one dimension的操作，每个dimension都有一个middle points，选择loss最小的维度的split point
  - why interaction:



  - when to stop?
    - 一个自然的想法就是converge，但如果，我们的第一个结点和第二个结点并没有capture the interaction effect，那很可能loss function不会变，直到capture ine interaction，loss function会大幅度减少，那怎么办呢

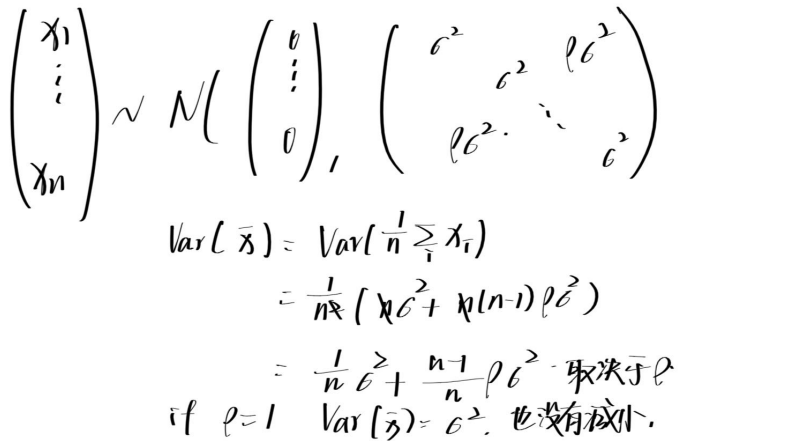
        - 刚开始grow a very large tree, 然后prune the tree，也就是从后向前剪枝，如果变化很大，那说明这个节点interaction，不能删掉，否则，有没有这个节点不重要，prune
    - 但是我们目前并不用prune，因为我们并不用一个single tree，而是用很多trees，所以我们不涉及如何剪枝的问题
- impurity measure:
    - classification error rate: E=1-max{p_k}
        - 不好，只用到了最大的p，并不能区分最大值相同但其他值不同的两个模型
    - gini:

$$G = \sum_{k=1}^{K} \hat{p}_k(1 - \hat{p}_k)$$

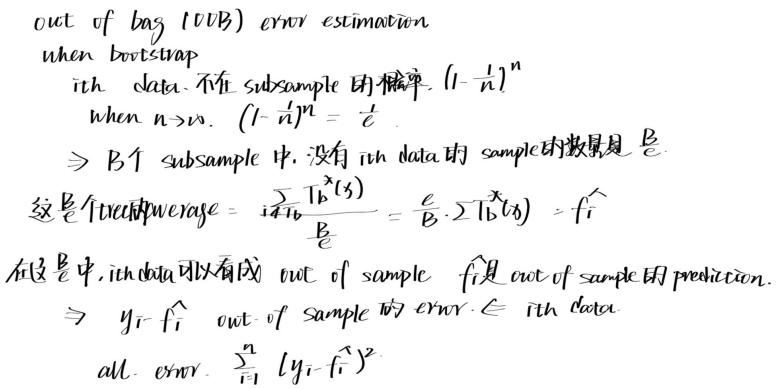
    - cross-entropy:

$$D = -\sum_{k=1}^{K} \hat{p}_k \log(\hat{p}_k)$$

- pros:
    - easy to explain(老师不这么认为)
    - computational cost， 只需要每个dimension sort， 复杂度为nplog(n),而ridge，如果我们用naive的方法，需要求逆，也就是p^3
    - can handle non-linearity
    - good at handling different format, 可以是连续的或者discrete，男女、大学名称等均可
    - good at handling missing data, nan 并不影响fit the tree，但之前的linearity model不可以有nan
- cons: 不稳定，有非常大的variance
- 那怎么解决方差太大的问题？bootstrap（还记得吗，如果bootstrap，那么var是之前的var/sqrt（n），但问题是x必须iid）
    - 可以用bagging的方法提高Lasso嘛?
    - 不行，对Lasso来说，用bias换variance，variance并不是大问题了，更大的问题是bias，所以会有relaxed lasso
- bootstrap就可以吗？注意bootstrap的前提的iid，但显然，我们抽取sample的时候，有很多的data是相同的，也就是说，correlation是很大的，如果有correlation，那么减少的variance是有上限的

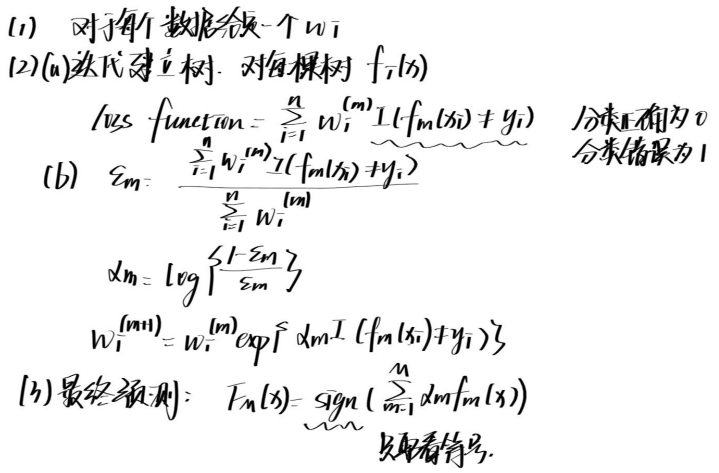


- OOB error



- 如何减少correlation呢？如果我们确实找到了一种方法是的correlation降低，这个时候variance确实减少，但是bias同时也会增加，有没有一种两全其美的方法
- 每次不全部抽取了，而是随机抽取m dimension，p=100，m=10，每次都随机抽取，这就是random forest
  - 如果只在第一次随机抽取，也就是随机抽10dimension之后一直用这个呢？我们知道100个里面可能只有10个有用，随机抽取的这10个可能并不合有用的10个重合，这样就会有很大的bias
  - 注意，m/p越大，也就是每次随机抽取的variable越大，search越多，df越多
  - Random forest把m限制在一个规定范围内，这样相当于regularization，会降低过拟合的概率
  - 一般来说，rf是比bagging好点的，但并不一定，因为我们之前也说过best subset并不一定是最差的，因为要看signal noise ratio
- Bagging trees: need a large tree and no pruning
- Random forest:need a large tree and no pruning
- Gradient Boosting: combine many small trees,use depth（每棵树的depth，通常1和2就已经很好了，即stump 的depth），combine many weak learner，每次都是训练之前的residual, not parallel structure but sequential structure.
  - 对于随机森林，是一颗大的树，这个树上的每一个分支等权，也是linear function， 但是对于boosting来说，每一颗树可能不是linear的，且每棵树的权重不一样

- Adaboost:
  - 通常是二分类，每次将分类错误的点的权重提高
  - 流程



    - 似乎不会overfit，由于在足够多迭代之后，training error为0，再加入新的树，test error也没有增长，所以认为没有overfit
    - 为什么好?
      - additive model
      - logistic regression
    - 首先来证明为什么是这么迭代的，前提，adaboost是最小化exponential loss function
  - 之后我们对Adaboost进行拓展
    - forward stagewise additive model(这个是拓展了什么？把Loss拓展成了square loss？)
    - Friedman's Gradient Boost Algorithm
    - Stochastic Gradient Boosting Tree:和之前不同的是用的是batch，不再用每个数据每个数据的计算了，既提高了速度，还提高了表现（像一些regularization），同时，由于没有square loss，只需要计算rho，而且存在一个lambda作为penalty，实现了一定的shrinkage，同时，还记得我们在Friedman中用的是square loss，但为什么要用square loss呢？并没有一个准确的依据，所以这里是假设区域划分不变，调整每个系数，从而实现不同的loss function
- Expectation-Maximization argoritnm
  - 当我们想要最大化$\max \log P(x|\theta)$时，等于$\sum_z P(x,z|\theta)$并不是那么容易，第二个是没办法直接对$P(x|\theta)$求导，这时就要考虑EM