Bagging

Bootstrap aggregating, also called bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid overfitting. Although it is usually applied to decision tree methods, it can be used with any type of method. Bagging is a special case of the model averaging approach.

Bootstrap 为了得到和原数据同样的分布，因为每次都是按照原来概率选择的。
In statistics, bootstrapping can refer to any test or metric that relies on random sampling with replacement. Why bootstrap? To get the same distribution with the original data, since we resample using the same probability.

Bagging 用 bootstrap 产生 N 组数据，然后用一个模型分别训练这 N 组数据，产生 N 组模型，再用这 N 组模型分别放到 test 上计算，最后 average/vote for 结果。
Bagging generates N new training sets by using bootstrap. Then use a model to train these N training sets and get N models. Finally use average/vote for to blend result.

Works for unstable classifiers

左边是一种模型对一个分布的拟合程度好坏，我用 error 来衡量，右边是 variance+bias，variance 是分歧，对于 stable 模型，stable 模型输入数据稍微变动时候输出结果变动很小，他的 variance 就很小，对于 unstable 模型 variance 就很大。Bias 指的是模型的共识的 error。所以说 bagging 的作用是 average，针对的是 variance，他把 variance 降低到 0？所以最后结果只剩下 bias。这就是为什么 bagging works for unstable model。

The bias of an estimator is the difference between an estimator's expectations and the true value of the parameter being estimated.
Variance measures how far a set of numbers (training result) is spread out.
Bagging mainly reduces the variance, so we need to choose unstable model (high variance) inside.

$$\text{avg}\left(E_{\text{out}}(g_t)\right) = \text{avg}\left(\mathcal{E}(g_t - \bar{g})^2\right) + E_{\text{out}}(\bar{g})$$

expected performance of $\mathcal{A}$ = expected deviation to consensus + performance of consensus

- performance of consensus: called **bias**
- expected deviation to consensus: called **variance**

Uniform blending: 平均后 G 效果好

Theoretical Analysis of Uniform Blending

$$G(\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} g_t(\mathbf{x})$$

$$\begin{aligned}
\text{avg}\left((g_t(\mathbf{x}) - f(\mathbf{x}))^2\right) &= \text{avg}\left(g_t^2 - 2g_t f + f^2\right) \\
&= \text{avg}\left(g_t^2\right) - 2Gf + f^2 \\
&= \text{avg}\left(g_t^2\right) - G^2 + (G - f)^2 \\
&= \text{avg}\left(g_t^2\right) - 2G^2 + G^2 + (G - f)^2 \\
&= \text{avg}\left(g_t^2 - 2g_t G + G^2\right) + (G - f)^2 \\
&= \text{avg}\left((g_t - G)^2\right) + (G - f)^2
\end{aligned}$$

$$\text{avg}\left(E_{\text{out}}(g_t)\right) = \text{avg}\left(\mathcal{E}(g_t - G)^2\right) + E_{\text{out}}(G) \geq + E_{\text{out}}(G)$$

Bias/variance/noise tradeoff of bias and variance:
http://horicky.blogspot.com/2012/06/predictive-analytics-evaluate-model.html

Derivation (continued)

$$\begin{aligned}
&E[\,(h(x^*) - y^*)^2\,] = \\
&\quad = E[\,(h(x^*) - \underline{h(x^*)})^2\,] + \\
&\quad\quad (\underline{h(x^*)} - f(x^*))^2 + \\
&\quad\quad E[\,(y^* - f(x^*))^2\,] \\
&\quad = \text{Var}(h(x^*)) + \text{Bias}(h(x^*))^2 + E[\,\varepsilon^2\,] \\
&\quad = \text{Var}(h(x^*)) + \text{Bias}(h(x^*))^2 + \sigma^2
\end{aligned}$$

Expected prediction error = Variance + Bias^2 + Noise^2

Variance: $E[(h(x^*) - \underline{h(x^*)})^2]$

Describes how much $h(x^*)$ varies from one training set S to another

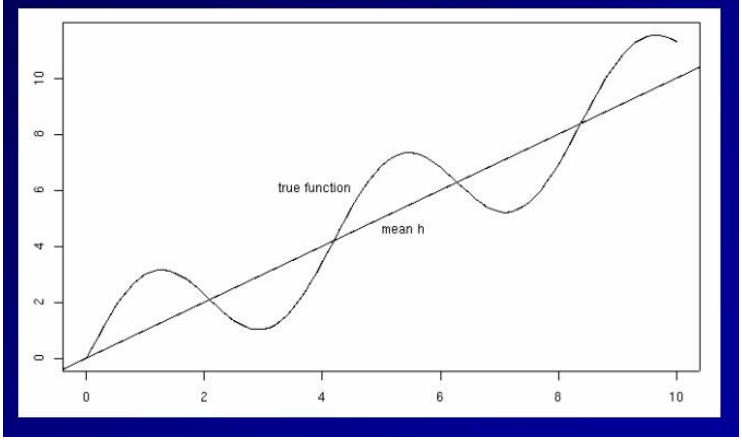
Bias: $[\underline{h(x^*)} - f(x^*)]$

Describes the <u>average</u> error of $h(x^*)$.

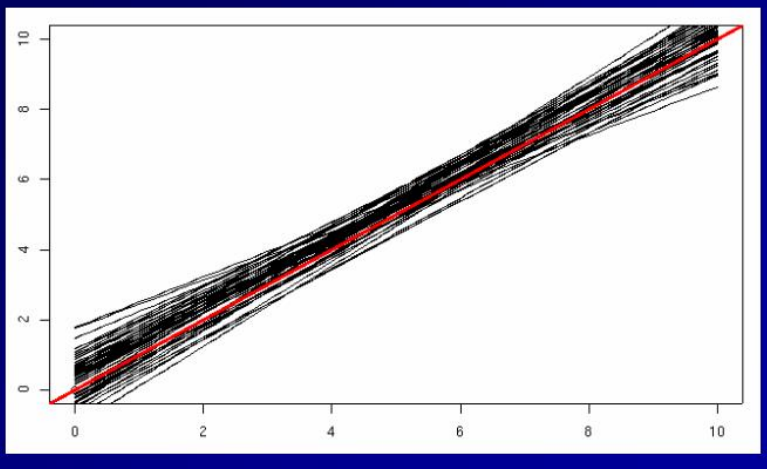
Noise: $E[(y^* - f(x^*))^2] = E[\varepsilon^2] = \sigma^2$

Describes how much y^* varies from $f(x^*)$

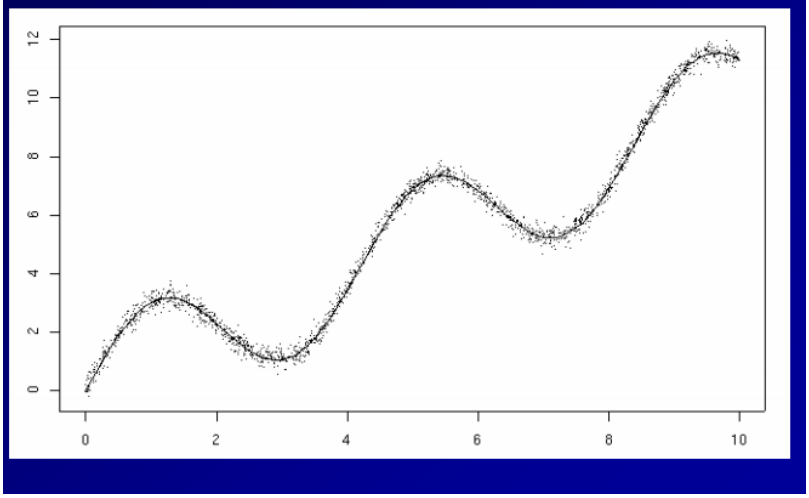
Bias



Variance



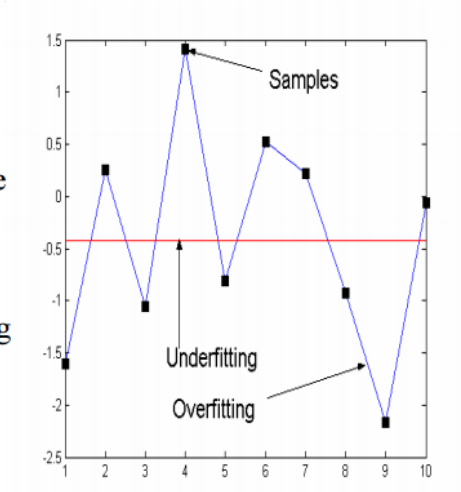
Noise



When Bagging works?

Under-fitting and over-fitting

- **Under-fitting:**
  - High bias (models are not accurate)
  - Small variance (smaller influence of examples in the training set)
- **Over-fitting:**
  - Small bias (models flexible enough to fit well to training data)
  - Large variance (models depend very much on the training set)



When Bagging works

- **Main property of Bagging** (proof omitted)
  - Bagging **decreases variance** of the base model without changing the bias!!!
  - Why? averaging!
- **Bagging typically helps**
  - When applied with an **over-fitted base model**
    - High dependency on actual training data
- **It does not help much**
  - High bias. When the base model is robust to the changes in the training data (due to sampling)

Models that fit the data poorly have high bias: "inflexible models" such as linear regression, regression stumps

Models that can fit the data very well have low bias but high variance: "flexible" models such as nearest neighbor regression, regression trees

This suggests that bagging of a flexible model can reduce the variance while benefiting from the low bias

For regression problems (squared error loss), the expected error rate can be decomposed into

– $\text{Bias}(x^*)^2 + \text{Variance}(x^*) + \text{Noise}(x^*)$

For classification problems (0/1 loss), the expected error rate depends on whether bias is present:

– if $B(x^*) = 1$: $B(x^*) - [V(x^*) + N(x^*) - 2\,V(x^*)\,N(x^*)]$

– if $B(x^*) = 0$: $B(x^*) + [V(x^*) + N(x^*) - 2\,V(x^*)\,N(x^*)]$

– or $B(x^*) + Vu(x^*) - Vb(x^*)$ [ignoring noise]

Dimensionality reduction and feature selection can decrease variance by simplifying models. Similarly, a larger training set tends to decrease variance. Adding features (predictors) tends to decrease bias, at the expense of introducing additional variance. Learning algorithms typically have some tunable parameters that control bias and variance, e.g.:

(Generalized) linear models can be regularized to decrease their variance at the cost of increasing their bias

In artificial neural networks, the variance increases and the bias decreases with the number of hidden units. Like in GLMs, regularization is typically applied.

In k-nearest neighbor models, a high value of k leads to high bias and low variance (see below).

In Instance-based learning, regularization can be achieved varying the mixture of prototypes and exemplars.

In decision trees, the depth of the tree determines the variance. Decision trees are commonly pruned to control variance.

One way of resolving the trade-off is to use mixture models and ensemble learning. For example, boosting combines many "weak" (high bias) models in an ensemble that has lower bias than the individual models, while bagging combines "strong" le arners in a way that reduces their variance.

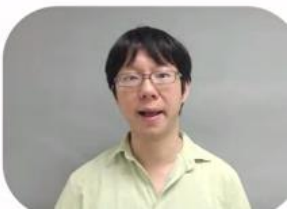
Bootstrap:

When using bootstrapping to re-sample N examples $\tilde{\mathcal{D}}_t$ from a data set $\mathcal{D}$ with N examples, what is the probability of getting $\tilde{\mathcal{D}}_t$ exactly the same as $\mathcal{D}$?

1. $0 \quad / N^N = 0$
2. $1 \quad / N^N$
3. $N! \ / N^N$
4. $N^N / N^N = 1$

Reference Answer: 3

Consider re-sampling in an ordered manner for N steps. Then there are (N^N) possible outcomes $\tilde{D}_t$, each with equal probability. Most importantly, $(N!)$ of the outcomes are permutations of the original $\mathcal{D}$, and thus the answer.



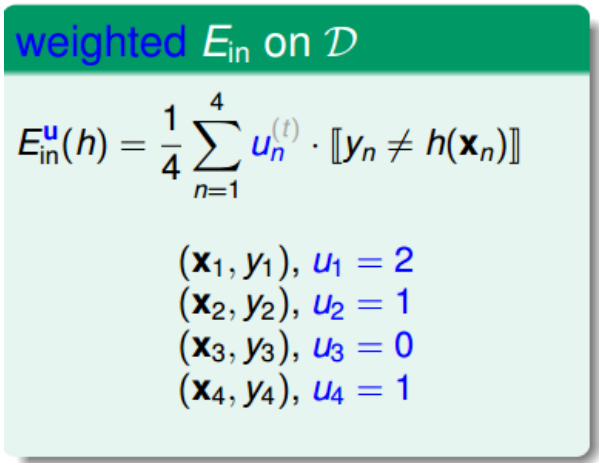
Adaptive boosting

Why match decision tree:

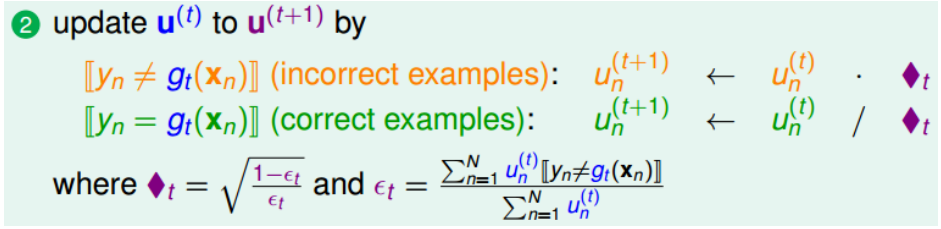
I talked about this in an answer to a related SO question. Decision trees are just generally a very good fit for boosting, much more so than other algorithms. The bullet point/ summary version is this:

1. Decision trees are non-linear. Boosting with linear models simply doesn't work well.
2. The weak learner needs to be consistently better than random guessing. You don't normal need to do any parameter tuning to a decision tree to get that behavior. Training an SVM really does need a parameter search. Since the data is re-weighted on each iteration, you likely need to do another parameter search on each iteration. So you are increasing the amount of work you have to do by a large margin.
3. Decision trees are reasonably fast to train. Since we are going to be building 100s or 1000s of them, thats a good property. They are also fast to classify, which is again important when you need 100s or 1000s to run before you can output your decision.
4. By changing the depth you have a simple and easy control over the bias/variance trade off, knowing that boosting can reduce bias but also significantly reduces variance. Boosting is known to overfit, so the easy nob to tune is helpful in that regard.

Bagging 和 boosting 的第一步都是 resampling。 Bagging 是用 bootstrap。而 boosting 相当于每次都多抽出之前分错的那些 data。Boosting 给每条 data 添加了 weight。 The first step of both bagging and boosting are resampling. Bagging uses bootstrapping, while boosting uses weighted training set.



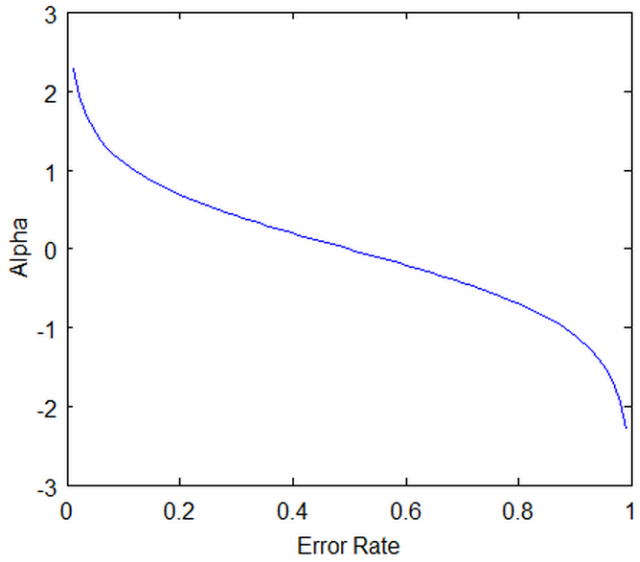
First, initiate data weight $\mathbf{u}^{(1)} = [\frac{1}{N}, \frac{1}{N}, \cdots, \frac{1}{N}]$. Pass the weight into model and train the first model. Increase the weight of wrong data, while decrease the weight of true data. ϵ_t :Classification error rate.



Finally we have many hypotheses, we need to set weight for each hypothesis. For good hypothesis which error rate less than 0.5 the weight is larger than 1, while for weak hypothesis the weight is less than 1.
Like transform original X to hypothesis(x) and do regression.
得到多组 hypothesis 后。给每个 hypothesis 赋予 weight，因为错误率小于 1/2 时，菱形参数大于 1，所以错误率小于 1/2 的 hypothesis 的 weight 大于 1.

3 compute $\alpha_t = \ln(\blacklozenge_t)$

return $G(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t g_t(\mathbf{x})\right)$



- Boosting seeks to find a weighted combination of classifiers that fits the data well
- Prediction: Boosting will primarily act to reduce bias

- In the early iterations, boosting is primary a bias-reducing method
- In later iterations, it appears to be primarily a variance-reducing method

Decision tree

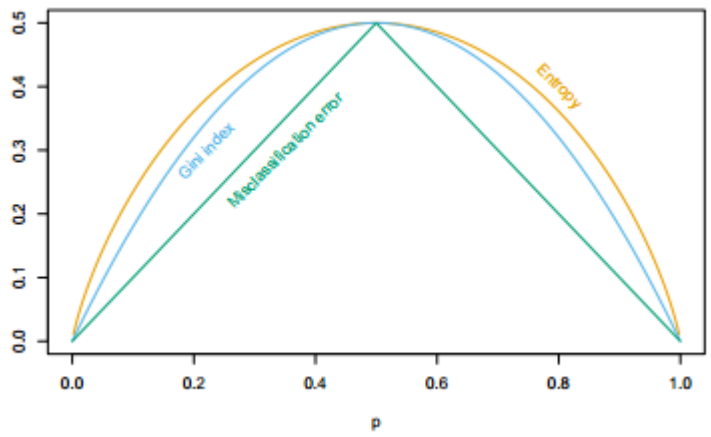
Surrogate branch: CART handle missing values easily.
CART:

two simple choices

- $C = 2$ (binary tree)
- $g_t(\mathbf{x}) = E_{\text{in}}$-optimal constant
  - binary/multiclass classification (0/1 error): majority of $\{y_n\}$
  - regression (squared error): average of $\{y_n\}$

CART 算法是怎样进行样本划分的呢？它检查每个变量和该变量所有可能的划分值来发现最好的划分，对离散值如{x,y,x}，则在该属性上的划分有三种情况({{x,y},{z}},{{x,z},y},{{y,z},x})，空集和全集的划分除外；对于连续值处理引进“分裂点”的思想，假设样本集中某个属性共 n 个连续值，则有 n-1 个分裂点，每个“分裂点”为相邻两个连续值的均值 (a[i] + a[i+1]) / 2。

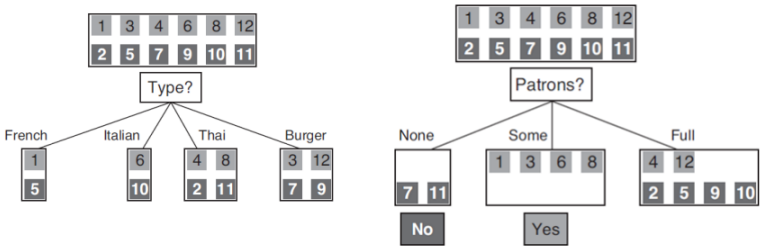
How to select attribute/evaluate purity:



Misclassification error

$$Misclassification(k\text{-th node}) = \frac{1}{N_k = |D_k|} \sum_{i \in D_k} I(y_i \neq C_k)$$

$$\sum_{a \in Values(A)} \frac{|Y_a|}{|Y|} Misclassification(Y_a)$$



$$Misclassification([]) = 0.5$$

$$Misclassification(Type) = \frac{2}{12} \times 0.5 + \frac{2}{12} \times 0.5 + \frac{4}{12} \times 0.5 + \frac{4}{12} \times 0.5 = 0.5$$

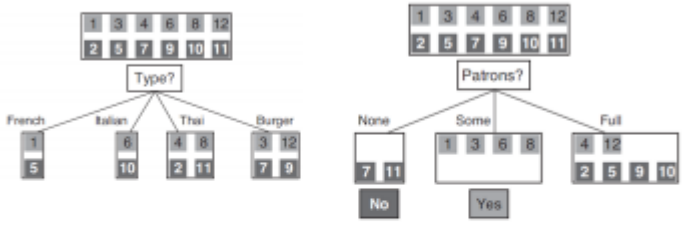
$$Misclassification(Patrons) = \frac{2}{12} \times 0 + \frac{4}{12} \times 0 + \frac{6}{12} \times \frac{2}{6} = \frac{1}{6}$$

Select the Best Attribute III

– IMPORTANCE(a, *examples*)

Impurity $Gini(Y) = 1 - \sum_{y} p(y)^2$

Split a node based on an attribute A $\sum_{a \in Values(A)} \frac{|Y_a|}{|Y|} Gini(Y_a)$



$$Gini([]) = 1 - 0.5^2 - 0.5^2 = 0.5$$

$$Gini(Type) = \frac{2}{12} \times 0.5 + \frac{2}{12} \times 0.5 + \frac{4}{12} \times 0.5 + \frac{4}{12} \times 0.5 = 0.5$$

$$Gini(Patrons) = \frac{2}{12} \times 0 + \frac{4}{12} \times 0 + \frac{6}{12} \times \left(1 - \frac{1}{9} - \frac{4}{9}\right) = \frac{2}{9}$$

Select the Best Attribute II

– IMPORTANCE(a, *examples*)

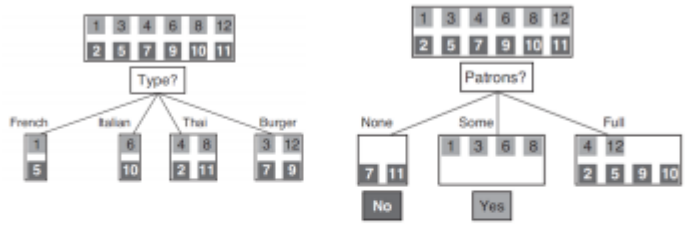
- Entropy – characterizes the amount of uncertainty.

$$Entropy(Y) = H(Y) = \sum_y p(y) \log_2 \frac{1}{p(y)}$$

$$= P("+") \log_2 \frac{1}{P("+")} + P("-") \log_2 \frac{1}{P("-")}$$

- Information Gain

$$Gain(Y, A) = Entropy(Y) - \sum_{a \in Values(A)} \frac{|Y_a|}{|Y|} Entropy(Y_a)$$



$$Gain([]) = 1 - \frac{12}{12} H\left(\frac{1}{2}\right) = 0$$

$$Gain(Type) = 1 - \left[\frac{2}{12} H\left(\frac{1}{2}\right) + \frac{2}{12} H\left(\frac{1}{2}\right) + \frac{4}{12} H\left(\frac{1}{2}\right) + \frac{4}{12} H\left(\frac{1}{2}\right)\right] = 0$$

$$Gain(Patrons) = 1 - \left[\frac{2}{12} H\left(\frac{0}{2}\right) + \frac{4}{12} H\left(\frac{4}{4}\right) + \frac{6}{12} H\left(\frac{2}{6}\right)\right] = 0.54$$

Gini不纯度

$$Gini = 1 - \sum_{i=1}^{n} P(i)^2$$

熵（Entropy）

$$Entropy = -\sum_{i=1}^{n} P(i) * \log_2^{P(i)}$$

错误率

$$Error = 1 - \max\{p(i) | \ i \text{ in } [1, n]\}$$

上面的三个公式均是值越大，表示越 “不纯”，越小表示越“纯”。三种公式只需要取一种即可，实践证明三种公司的选择对最终分类准确率的影响并不大，一般使用熵公式。The higher value, the higher impurity. We usually choose entropy.
纯度差，也称为信息增益（Information Gain），公式如下：

$$\Delta = I(parent) - \sum_{j=1}^{k} \frac{N(v_j)}{N} * I(v_j)$$

其中，I 代表不纯度（也就是上面三个公式的任意一种），K 代表分割的节点数，一般 K = 2。vj 表示子节点中的记录数目。上面公式实际上就是当前节点的不纯度减去子节点不纯度的加权平均数，权重由子节点记录数与当前节点记录数的比例决定。The impurity now subtracts the weighted average of impurity of subtrees.

Prune and stop condition:

Pre-pruning/Post-pruning，CART usually choose post-pruning. 该方法是通过在完全生长的树上剪去分枝实现的，通过删除节点的分支来剪去树节点。最下面未被剪枝的节点成为树叶。

Overfitting is a significant practical difficulty for decision tree models and many other predictive models. Overfitting happens when the learning algorithm continues to develop hypotheses that reduce training set error at the cost of an increased test set error. There are several approaches to avoiding overfitting in building decision trees.

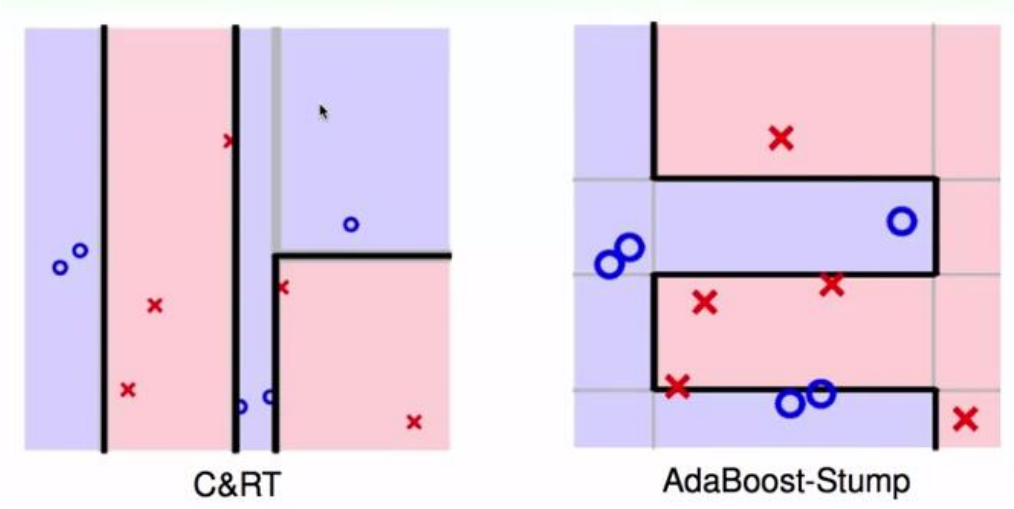
- **Pre-pruning** that stop growing the tree earlier, before it perfectly classifies the training set.
- **Post-pruning** that allows the tree to perfectly classify the training set, and then post prune the tree.

Practically, the second approach of post-pruning overfit trees is more successful because it is not easy to precisely estimate when to stop growing the tree.

The important step of tree pruning is to define a criterion be used to determine the correct final tree size using one of the following methods:

1. Use a distinct dataset from the training set (called validation set), to evaluate the effect of post-pruning nodes from the tree.
2. Build the tree by using the training set, then apply a statistical test to estimate whether pruning or expanding a particular node is likely to produce an improvement beyond the training set.
    - Error estimation
    - Significance testing (e.g., Chi-square test)
3. Minimum Description Length principle : Use an explicit measure of the complexity for encoding the training set and the decision tree, stopping growth of the tree when this encoding size (size(tree) + size(misclassifications(tree)) is minimized.

Adaboost 切出来的一定是横跨整个平面。Adaboost split the whole plane.



- human-explainable
- multiclass easily
- categorical features easily
- missing features easily
- efficient non-linear training (and testing)

—almost no other learning model share all such specialties, except for other decision trees

Reference

https://people.cs.pitt.edu/~milos/courses/cs2750-Spring04/lectures/class23.pdf
bias, variance, noise
http://web.engr.oregonstate.edu/~tgd/classes/534/slides/part9.pdf
Bias–variance tradeoff
https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff
decision tree
http://www.cnblogs.com/bourneli/archive/2013/03/15/2961568.html
prune a tree
http://www.saedsayad.com/decision_tree_overfitting.htm
pretty good decision tree tutorial
http://www.saedsayad.com/decision_tree.htm