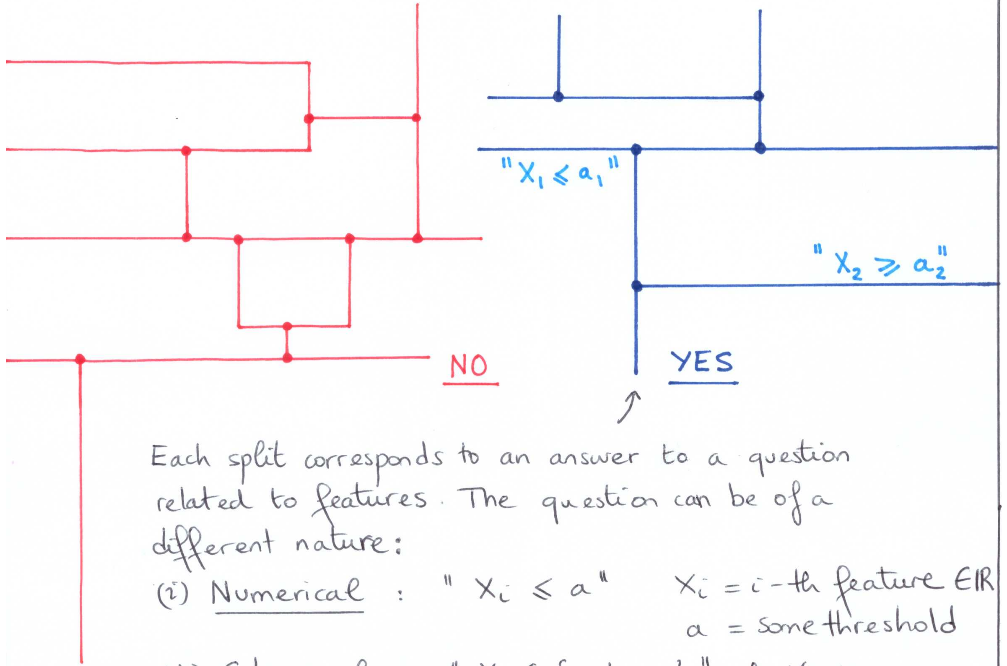
SL = TREES, BAGGING & RANDOM FORESTS

Tree algorithms partition the feature space into a set of rectangular regions. The divisions are performed recursively, and typically consists in splitting existing regions into two smaller regions ("binary splits"), whose edges are aligned with the axes.



Each split corresponds to an answer to a question related to features. The question can be of a different nature:

(i) Numerical : " $X_i \leqslant a$ " $X_i = i$-th feature $\in \mathbb{R}$, a = some threshold

(ii) Categorical : " $X_i \in \{a, b, c\}$ " if X_i is a categorical variable, and (a, b, c) three possible categories for X_i.

We discuss both regression & classification trees; and focus on the CART procedure (Breiman et al '84), and on ID.3 / C4.5 (Quilan '86, '93). We then explain

how the accuracy of tree-based methods can be improved using a general technique known as bootstrap aggregation (bagging), see Section II.

I. TREES

I. 1. CART procedure.

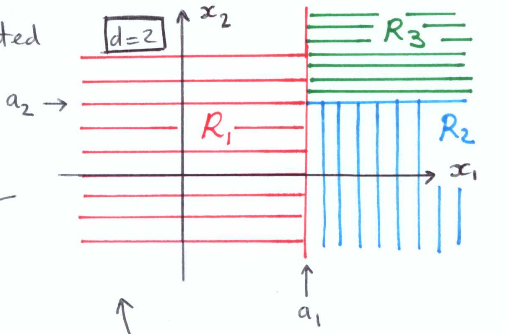
- The binary tree structure is learned in a greedy way: start with the whole dataset $\mathcal{L}_n = \{(x_1, y_1), \dots, (x_n, y_n)\}$, and consider the variable and the threshold which result in the largest decrease of some goodness-of-fit criterion.

The procedure is then repeated in one or both of these regions.

Assume $x \in \mathbb{R}^d$

associated tree :

$$\begin{tikzcd}
 & \boxed{x_1 \leqslant a_1} \arrow[dl] \arrow[dr] & & \\
R_1 & & \boxed{x_2 \leqslant a_2} \arrow[dl] \arrow[dr] & \\
 & R_2 & & R_3
\end{tikzcd}$$



In each region, the predicted response variable is constant (trees are suitable for K-class classification problems).

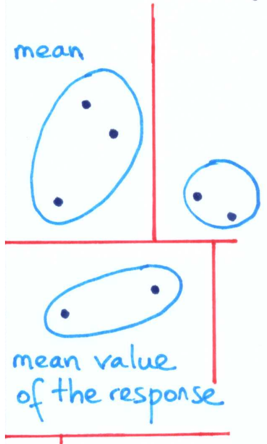
- Decision trees are simple and easily interpretable models.
  ↳ very popular technique.

- We discuss next how to choose the split variable & the split point, based on the nature of the learning problem.

→ Regression Trees

We first answer the simple question related to the prediction of the response variable in each region. Suppose we have constructed J regions $R_1, .., R_J$. Trees predict a constant value in each terminal region:

$$f_n(x) = \sum_{j=1}^{J} \underset{\uparrow}{\hat{y}_j} \, \mathbb{1}(x \in R_j).$$



The value of $\hat{y}_j$ is easily determined once a loss is chosen. For a square loss,

$$\hat{y}_j = \underset{\alpha_j}{\text{argmin}} \sum_{i=1}^{n} (y_i - \alpha_j)^2 \mathbb{1}(x_i \in R_j)$$
$$= \frac{1}{|R_j|} \sum_{i=1}^{n} y_i \, \mathbb{1}(x_i \in R_j)$$

= mean value of the response variable within region R_j.

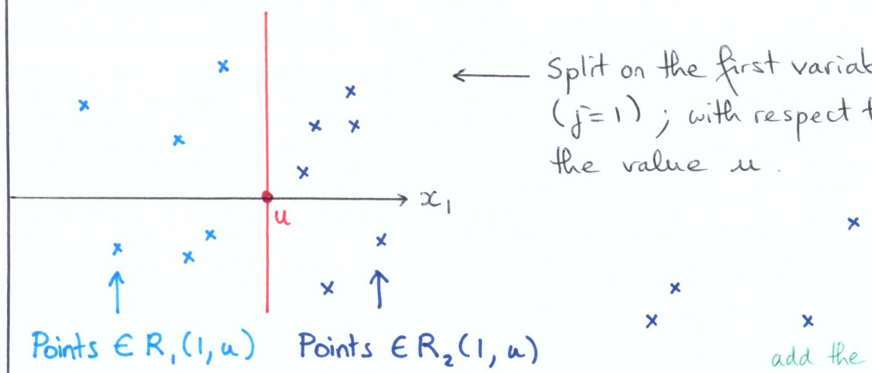
For an absolute loss, $\hat{y}_j$ = median .../...

↳ To choose the split variable & the split point (threshold), we use a greedy procedure = select $(\underset{\nearrow}{j}, \underset{\nwarrow}{u})$ which yields the largest decrease in the residual sum of squares (assuming a square loss → can be easily generalized to more general loss functions). Starting with all data, consider:

j-th variable
$1 \leq j \leq d$
$X = (X_1, .., X_d) \in \mathbb{R}^d$

threshold
$u \in \mathbb{R}$

- $R_1(j, u) = \{x_1, .., x_n \mid \underset{\uparrow}{x_{kj}} \leq u\}$
  $x_k = (x_{k1}, .., x_{kd}) \in \mathbb{R}^d$
- $R_2(j, u) = \{x_1, .., x_n \mid x_{kj} > u\}$.


Split on the first variable (j=1); with respect to the value u.
x₁
u
Points ∈ R₁(1,u)
Points ∈ R₂(1,u)

- We are looking for (j, u) which minimizes =

add the RSS of other regions as well

$$\boxed{\min_{\alpha_1} \sum_{i=1}^{n} (y_i - \alpha_1)^2 \mathbb{1}(x_i \in R_1(j,u)) + \min_{\alpha_2} \sum_{i=1}^{n} (y_i - \alpha_2)^2 \mathbb{1}(x_i \in R_2(j,u))}$$

↳ For any candidate (j, u), the inner minimizer is already known: mean response in $R_1(j,u)$ and $R_2(j,u)$, respectively.

- This optimization problem can be efficiently done with an exhaustive search on (j, u), since there are finitely many candidate values:
  - features lie in $\mathbb{R}^d$, $d < \infty$
  - finite number of training examples.
  (→ consider splits in the 'middle' of two consecutive points, along each dimension).

- The procedure is stopped once the decrease in RSS falls below some threshold, or when the number of observations left in each region becomes to small.

$\oplus$: . Highly interpretable
. Easy to grow
. Visual tree representation.

$\ominus$: . High variance : a small change in the learning sample may lead to different splits, and a different tree structure
. A bad split may result in subsequent less useful splits, resulting in a tree which is far from the optimal one (nature of greedy algorithms).
. Growing large trees might overfit the data.

$\to$ Classification Trees. To grow a classification tree, the procedure is similar : start with a single node (the root), whose label corresponds to the class majority over the whole sample :

$$\boxed{\underset{1\leq k\leq K}{\text{argmax}} \ \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(y_i = k)}$$

Consider K-class classification with $\{(x_i, y_i)\}_{i=1,..,n}$, where $y_i = k$ iff x_i is of class k, $1\leq k\leq K$.

Then, at each node of the tree, split the associated region R into two subregions $R_1(j,u)$ & $R_2(j,u)$ according to some criteria : choose the pair (j,u) = (split variable, split point) which reduces the <u>NODE IMPURITY</u> the most, according to some measure of impurity Q.

x <u>Recall</u>: in regression trees, we make use of the residual sum of squares : . $Q(R) := \frac{1}{|R|}\sum_{i\in R}(y_i - \hat{y})^2$

$$\hat{y} = \frac{1}{|R|}\sum_{i\in R} y_i$$

and . $|R_1|\,Q(R_1) + |R_2|\,Q(R_2)$

$$= \sum_{i\in R_1}(y_i - \hat{y}_1)^2 + \sum_{i\in R_2}(y_i-\hat{y}_2)^2 ,$$

$$\hat{y}_j = \frac{1}{|R_j|}\sum_{i\in R_j} y_i \ ; \ j=1,2.$$

& Select (j,u) which maximizes the difference :

$$\boxed{\underbrace{|R|\,Q(R)}_{\text{initial RSS}} - \underbrace{\{|R_1(j,u)|\,Q(R_1(j,u)) + |R_2(j,u)|\,Q(R_2(j,u))\}}_{\text{new RSS, after the split}}}$$

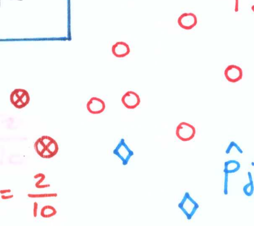
We introduce 3 measures of impurity, adapted to the problem of K-class classification.

(a) <u>Misclassification Error</u> : For a node / terminal region R_j,

let $\hat{p}_{jk} = \frac{1}{|R_j|}\sum_{x_i\in R_j}\mathbb{1}(y_i = k)$
= proportion of points in R_j that belong to class j.

Then

$$\boxed{Q(R_j) = 1 - \max_{1\leq k\leq K}\hat{p}_{jk}}$$



$\hat{p}_{j\circ} = \frac{6}{10}$

$\hat{p}_{j\otimes} = \frac{2}{10}$

$\hat{p}_{j\diamond} = \frac{2}{10}$

$\hookrightarrow$ misclassification error is $\frac{4}{10}$

x <u>Remark</u>: the misclassification error can be rewritten

$\frac{1}{|R_j|}\sum_{i\in R_j}\mathbb{1}(y_i \neq k(j))$,

where

$k(j) = \underset{1\leq k\leq K}{\text{argmax}}\ \hat{p}_{jk}$, and generalizes well to the problem of classification with weighted observations.

Indeed, suppose that (x_i, y_i) has weight $w_i \geqslant 0$ (the previous case corresponds to $w_i = 1 \quad \forall i$).

Then

$$\hat{p}_{jk} = \frac{\sum_{i \in R_j} w_i \mathbb{1}(y_i = k)}{\sum_{i \in R_j} w_i}$$

= weighted proportion of points in R_j that belong to class k.

The expression of the misclassification error remains unchanged.

(b) Gini Index :
$$\boxed{Q(R_j) = \sum_{k=1}^{K} \hat{p}_{jk}(1-\hat{p}_{jk})}$$

Used in ecology as a measure of diversity of species: – great diversity when presence of many rare species – Rare species are thus given a higher weight in the computation of a diversity indicator.
$\Rightarrow$ If a species is present in proportion p, weight it by $(1-p)$.

Gini Index is used by CART.

(c) Entropy :
$$\boxed{Q(R_j) = -\sum_{k=1}^{K} \hat{p}_{jk} \log_2 \hat{p}_{jk}}$$

Similar to Gini impurity index, a large weight is given to rare species, but use $\log(1/p)$ instead of $(1-p)$. The weight is now unbounded.

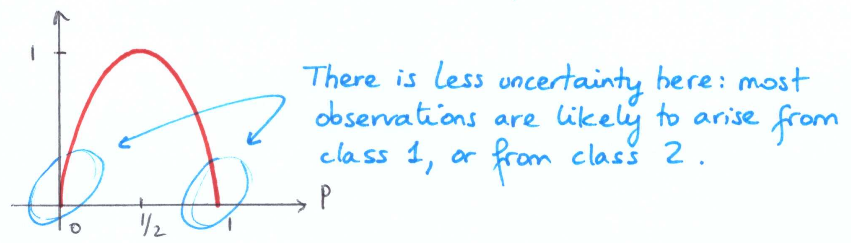
$\rightarrow$ In thermodynamics, entropy is a measure of order.

$\rightarrow$ In information theory, entropy is a measure of uncertainty: with discrete RVs, entropy is maximized for the uniform distribution $p_k = 1/K \quad \forall k = 1, .., K$, and is reduced when one of the p_k dominates the other.

This is best illustrated in the case of binary classification:

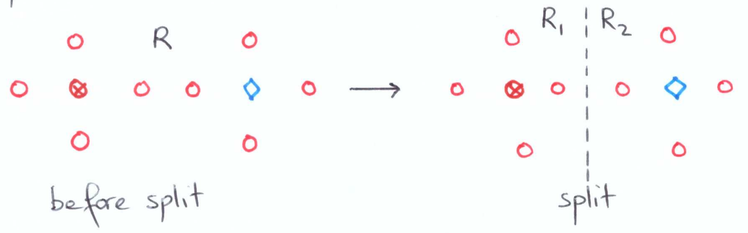
Entropy is $-p \log_2 p - (1-p)\log_2(1-p)$.

$K=2, \quad p_1 = p, \quad p_2 = 1-p$



x Remark: Comparison of Impurity Measures.

The misclassification error may not decrease strictly after a split, compared to Gini Index or Entropy. Consider the following simple scenario:



(a) Misclassification:

$$|R|\,Q(R) = 10\left(1 - \frac{8}{10}\right) = 2$$

$$|R_1|\,Q(R_1) + |R_2|\,Q(R_2) = 5\left(1 - \frac{4}{5}\right) + 5\left(1 - \frac{4}{5}\right) = 2$$

(b) Gini:

$$|R|\,Q(R) = 10\left(\frac{8}{10}\left(1-\frac{8}{10}\right) + \frac{1}{10}\left(1-\frac{1}{10}\right) + \frac{1}{10}\left(1-\frac{1}{10}\right)\right) = 3.4$$

$$|R_1|\,Q(R_1) + |R_2|\,Q(R_2) = 2 \times 5 \times \left(\frac{4}{5}\left(1-\frac{4}{5}\right) + \frac{1}{5}\left(1-\frac{1}{5}\right)\right)$$
$$= 3.2 < 3.4$$

(c) Entropy:

$$|R|\,Q(R) = -10\left(\frac{8}{10}\log_2\frac{8}{10} + 2 \times \frac{1}{10}\log_2\frac{1}{10}\right) = 9.22$$

$$|R_1|Q(R_1) + |R_2|Q(R_2) = -2\times 5\left(\frac{4}{5}\log_2\frac{4}{5} + \frac{1}{5}\log_2\frac{1}{5}\right)$$

$$= 7.22 < 9.22.$$

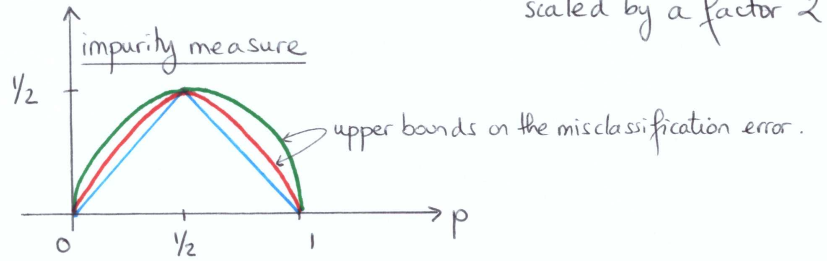
"$0\log_2 0 = 0$" by convention

↳ The reason is that both the Gini index & Entropy are strictly concave functions of p, while the misclassification error is not strictly concave: consider the binary classification case:

(a) Misclassification error is $1-\max(p,(1-p))$ [in blue]

(b) Gini index is $2p(1-p)$ [in red]

(c) Entropy is $-p\log_2 p - (1-p)\log_2(1-p)$ [in green] ↙ scaled by a factor 2.



$\Rightarrow |R|\,Q(R) \geq |R_1|Q(R_1) + |R_2|Q(R_2)$ for misclassification error

$|R|\,Q(R) > |R_1|Q(R_1) + |R_2|Q(R_2)$ for Gini + Entropy.

Ex: $$\underbrace{|R|\sum_{k=1}^{K} p_k(1-p_k)}_{Q(R)} > \sum_{j=1}^{2}|R_j|\,Q(R_j) = \sum_{j=1}^{2}|R_j|\underbrace{\sum_{k=1}^{K} p_{jk}(1-p_{jk})}_{Q(R_j)},$$

where $|R| = |R_1| + |R_2|$.

↳ Gini + Entropy are usually preferred to the misclassification criterion to grow a classification tree.

× Remark: Overfitting & VC dimension.

Large trees can overfit the data. In particular, a tree with n terminal regions, where n denotes the sample size, achieves zero training error, but generalizes poorly.

If $\mathcal{F}$ = class of trees with unlimited number of terminal nodes,

$\mathcal{F}$ can shatter any training set $\forall n$, and we see that $VC(\mathcal{F}) = +\infty$.

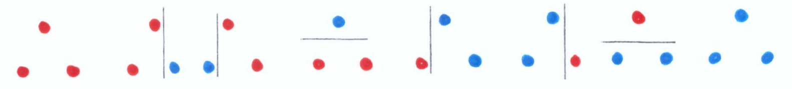
↳ to reduce the complexity of trees, we may limit its size, or its number of terminal nodes. We derive next an upper bound on the VC dimension of trees with $(J+1)$ terminal regions.

Suppose $x\in\mathbb{R}^2$, and put

$$\mathcal{G} = \{x\mapsto \mathbb{1}(x_i\leq u) \text{ or } x\mapsto\mathbb{1}(x_i\geq u),\ u\in\mathbb{R}\}$$

= class of hyperplanes that are aligned with the coordinate axis.

$\mathcal{G}$ can shatter 3 points, but not 4 (even hyperplanes in $\mathbb{R}^2$ cannot shatter 4 points).



$\Rightarrow VC(\mathcal{G}) = 3$

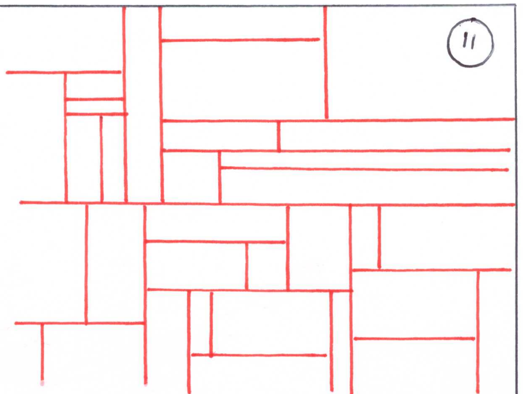
In $\mathbb{R}^d$, $\mathcal{G}\subset$ class of hyperplanes

$\Rightarrow VC(\mathcal{G}) \leq VC(\text{hyperplanes in } \mathbb{R}^d) \underset{\uparrow}{=} d+1$

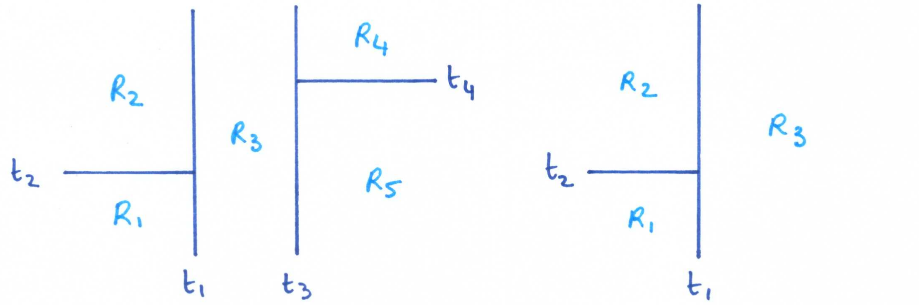
see chapter SL: VC THEORY.

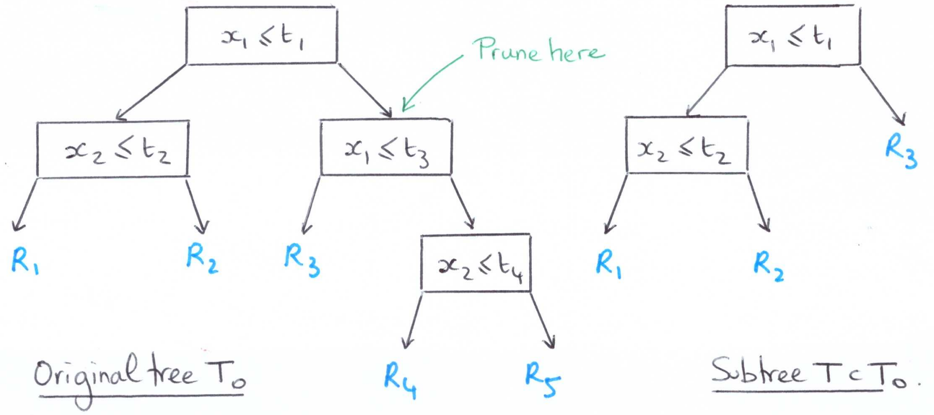
↳ With $(J+1)$ terminal nodes, we need to consider J hyperplanes. These J hyperplane can shatter a maximum number of $(d+1)$ observations each $\Rightarrow$ VC dimension of binary classification trees with $(J+1)$ terminal nodes is $\leq J(d+1)$.

- To avoid overfitting, a common strategy is to first grow a large tree (call it T_0), and then prune it to obtain a subtree T, obtained by collapsing a number of its internal nodes, according to some criteria. This strategy, employed by CART, is known as <u>COST-COMPLEXITY PRUNING</u>.



<u>Example:</u>





<u>Original tree T_0</u>

<u>Subtree $T \subset T_0$.</u>

To prune T_0, we introduce a criterion based on the sum of an empirical error (RSS or impurity measure), and a complexity term, expressed in terms of the size / number of leaves of a tree T:

$$\boxed{C_\lambda(T) := \underbrace{\sum_{j=1}^{|T|} |R_j|\, Q(R_j)}_{\text{impurity measure}} + \lambda \underbrace{|T|}_{|T| = \#\text{ of terminal nodes in } T},}$$

$\lambda > 0$ = tuning parameter

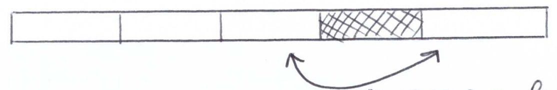
where $Q(R_j) = \frac{1}{|R_j|} \sum_{x_i \in R_j} (y_i - \hat{y})^2$; $\hat{y} = \frac{1}{|R_j|} \sum_{x_i \in R_j} y_i$

for the regression case,

and where $Q(R_j)$ = Misclassification error / Gini Index / Entropy in classification problems (p. 6/7).

$\Rightarrow$ For each $\lambda > 0$, compute the subtree $T_\lambda \subset T_0$ which minimizes $C_\lambda(T)$: $T_\lambda \in \underset{T \subset T_0}{\operatorname{argmin}}\, C_\lambda(T)$. The subtree T_λ is computed using <u>WEAKEST LINK PRUNING</u>: for a given λ, successively collapse internal nodes that result in the smallest increase of $\sum |R_j|\, Q(R_j)$, until you reach a single node tree. This defines a finite sequence of trees, and it can be shown that this sequence must contain T_λ, and that T_λ is unique.

$\hookrightarrow$ The value of λ is usually selected using cross-validation.



(ii) Apply cost complexity pruning to this tree to obtain a sequence of trees $\{T_\lambda\}_\lambda$ for values of λ on a grid.

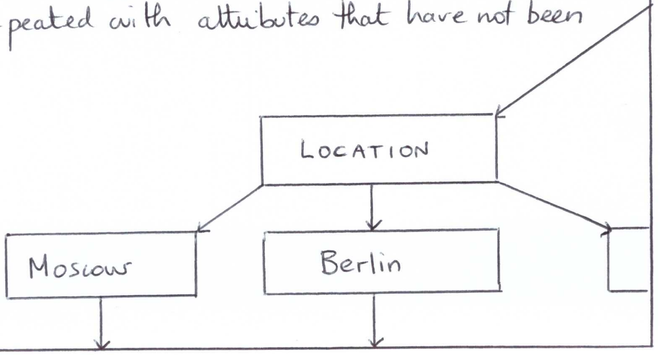
(iii) Evaluate the performance of each subtree on the hold out set.

(iv) Repeat this for each fold, and average the performance.

(v) Select the value $\hat{\lambda}$ of λ which has the smallest error, and return $T_{\hat{\lambda}}$.

I.2. I.D.3 Algorithm.

ID.3 (or Iterative Dichotomiser 3) is a simple decision tree algorithm introduced by Quinlan (1983). In its original form, the algorithm works for categorical variables only. Successive versions of the algorithm (C4.5 and C5.0) can manage continuous features. ID.3, like CART, constructs a decision tree by considering a greedy search, based on a criterion called information gain. Unlike CART, ID.3 does not necessarily perform binary splits. Instead, each node of the tree corresponds to a (categorical) attribute, and the children of this node correspond to all possible categorical values that this attribute can take. The procedure is then repeated with attributes that have not been selected before.



We illustrate the procedure on the following dataset:

COLOR	SIZE	DOTS	EATABILITY
red	large	no	toxic
red	large	yes	toxic
brown	large	no	eatable
green	small	no	eatable
brown	small	yes	eatable
red	large	no	toxic
red	small	no	eatable
green	small	no	eatable
red	small	yes	eatable
brown	large	yes	eatable
brown	small	no	eatable
green	large	yes	toxic
green	small	yes	toxic
green	large	no	eatable

Mushroom data; 3 features; n = 14.

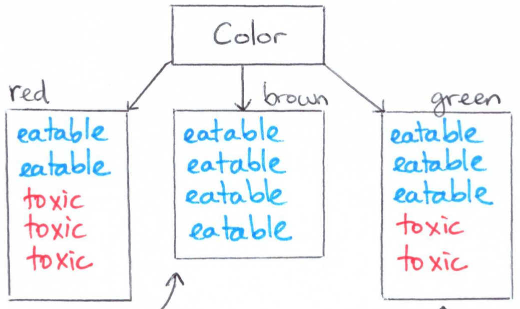
→ ID.3 uses the entropy as a measure of node purity.

$\hat{P}_{toxic} = \frac{5}{14}$; $\hat{P}_{eat} = \frac{9}{14}$

The entropy of the entire dataset is

$$H = -\frac{5}{14}\log_2\frac{5}{14} - \frac{9}{14}\log_2\frac{9}{14} = \underline{0.940}.$$

→ Entropy of subsets of the original training sample, created by splitting on the attribute 'Color'.



$H(\text{red}) =$
$-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = \underline{0.971}$

$H(\text{brown}) = -1\log_2 1 - 0\log_2 0 = \underline{0}$

$H(\text{green}) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = \underline{0.971}$

The average entropy of the attribute 'color' is:

$$H(\text{'Color'}) = \frac{|R_1|}{|R|} H(\text{red}) + \frac{|R_2|}{|R|} H(\text{brown}) + \frac{|R_3|}{|R|} H(\text{green})$$

$$= \frac{5}{14} \times 0.971 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.971$$

$$= \underline{0.693}.$$

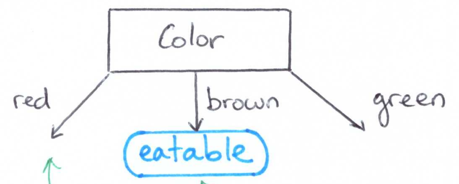
The <u>INFORMATION GAIN</u> is defined as the difference between the entropy of the original dataset, and the average entropy obtained after splitting on the attribute 'Color'.

$$G(\text{'Color'}) = H - H(\text{'Color'}) = \underline{0.247}.$$

→ This calculation is repeated for the attributes 'Size' and 'Dots': Gain ('Size') = $\underline{0.151}$

Gain ('Dots') = $\underline{0.048}$.

→ Select the attribute with the largest information gain; here 'Color'.



Pure node: classify as eatable. <u>END</u>.

For impure nodes, repeat the procedure by splitting the subsets on attributes 'Size' and / or 'Dots' (aka attributes that have not been considered before), until 'pure' nodes are found.

* <u>Remarks</u> (i) The information gain is more likely to select attributes with a large number of values, since subsets are more likely to be pure → ID.3 tends to overfit the data,

by fragmenting it. A possible remedy, suggested by Quinlan, is to introduce a second criterion, known as 'Split Information', which quantifies the purity of an attribute.

<u>Ex</u>: The attribute 'Color' has 5 red / 4 brown / 5 green.

$$\text{SplitInfo}(\text{'Color'}) = -\left(\frac{5}{14}\log_2\frac{5}{14} + \frac{4}{14}\log_2\frac{4}{14} + \frac{5}{14}\log_2\frac{5}{14}\right)$$

$$= \underline{1.577}$$

↖ The smaller, the better.

Quinlan (1986) suggests that rather than choosing the attribute with the largest gain, to select the one with the largest 'Gain Ratio':

$$\text{GainRatio}(\text{'Color'}) = \frac{G(\text{'Color'})}{\text{SplitInfo}(\text{'Color'})} = \underline{0.157}.$$

(ii) <u>Missing Attribute Values</u>.

→ Treat 'Unknown' as a possible category.

→ If $A_1, \dots, A_J$ are possible values for attribute A,

Let • $n_{i,0}$ (resp. $n_{i,1}$) = # observations with attribute A_i belonging to class 0 (resp. to class 1)

Binary Classification

• u_0 (resp. u_1) = # observations of class 0 (resp. of class 1) with unknown attribute.

Quilan suggests to compute the information gain using

$$n_{i,0} + u_0 \left\{ \frac{n_{i,0} + n_{i,1}}{\sum_{j=1}^{J} (n_{j,0} + n_{j,1})} \right\}$$ instead of $n_{i,0}$ (& similarly for $n_{i,1}$).

↖ proportion of observations with attribute A_i.

(iii) Continuous Attributes.

The original ID.3 algorithm does not handle continuous attributes. An extended version, called C4.5, was proposed by Quinlan (1993) to incorporate both categorical & continuous variables.

↳ C4.5 creates a threshold based on all observed values of a continuous attribute. Denote these observations (sorted in increasing value) by $x_1 \leqslant \dots \leqslant x_n$. ($n$ = sample size)

↳ $(n-1)$ thresholds are considered:

$$t_i = \frac{x_{i+1} - x_i}{2} \quad ; \quad i = 1, \dots, n-1 .$$

For each threshold t_i, the information gain obtained by splitting the data set according to values of the attribute $\leqslant t_i$ & $> t_i$ is computed. The threshold with the largest information gain is chosen.

For more information about C4.5, see [Quinlan, Ross. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993].

II _ BOOTSTRAP AGGREGATION

II.1. The bootstrap principle.

The bootstrap is a statistical tool used to quantify the uncertainty associated with an estimator. It was introduced by Efron (1979), and since then successfully applied to many real life problems, due to its very general and universal nature.

Suppose we wish to quantify the uncertainty of a learner $f_n = f(\mathcal{L}_n)$ constructed from a training sample $\mathcal{L}_n = \{(X_i, Y_i)\}_{i=1}^{n}$

To evaluate the bias / variance of $f(\mathcal{L}_n)$, we ideally collect more data, and perform the estimation several times.

$$\mathcal{L}_n^{(1)} = \{(X_1^{(1)}, Y_1^{(1)}), \dots, (X_n^{(1)}, Y_n^{(1)})\} \longrightarrow f(\mathcal{L}_n^{(1)})$$

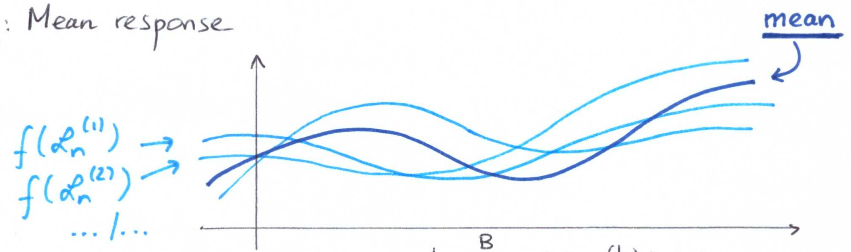
$$\vdots$$

$$\mathcal{L}_n^{(B)} = \{(X_1^{(B)}, Y_1^{(B)}), \dots, (X_n^{(B)}, Y_n^{(B)})\} \longrightarrow f(\mathcal{L}_n^{(B)})$$

↑ B independent replicates of the original dataset: $(X_i^{(b)}, Y_i^{(b)}) \sim \mathbb{P}_{X,Y}$ $\left(\begin{array}{l} i = 1, \dots, n \\ b = 1, \dots, B \end{array}\right.$

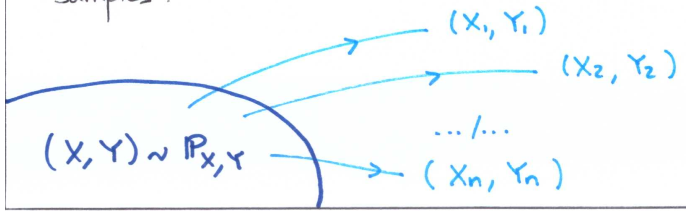
↓ These B independent estimates can be aggregated to learn about some aspects of the distribution of $f(\mathcal{L}_n)$.

Ex: Mean response



$$\text{mean} = \frac{1}{B} \sum_{b=1}^{B} f(\mathcal{L}_n^{(b)})(\cdot)$$

→ In practice however, we cannot repeat the experiment B times. The BOOTSTRAP approach allows us to artificially generate new samples, in order to mimic the process of obtaining new training samples $\mathcal{L}_n^{(1)}, \dots, \mathcal{L}_n^{(B)}$, and thus estimate properties of the distribution of the learner without collecting additional samples.



In particular, we are interested in

(a) the mean prediction $\mathbb{E}_{\mathbb{P}}\{f(\mathcal{L}_n)\}$

(b) the prediction variance $\mathbb{E}_{\mathbb{P}}\{(f(\mathcal{L}_n) - \mathbb{E}_{\mathbb{P}} f(\mathcal{L}_n))^2\}$

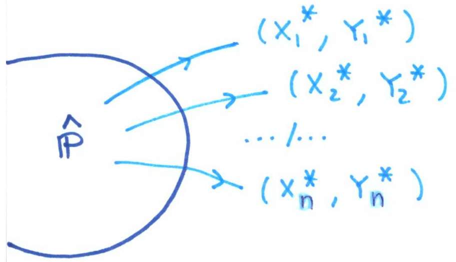
We emphasize that $\mathbb{E}\{\dots\}$ is computed under the true underlying distribution $\mathbb{P} = \mathbb{P}_{X,Y}$.

↳ In both cases, we need an estimate of $\mathbb{E}_{\mathbb{P}}\{\dots\}$.

• The <u>BOOTSTRAP ESTIMATE</u> of $\mathbb{E}_{\mathbb{P}}\{f(\mathcal{L}_n)\}$ is $\mathbb{E}_{\hat{\mathbb{P}}}\{f(\mathcal{L}_n^*)\}$,

where $\hat{\mathbb{P}}$ = Empirical distribution of $\mathcal{L}_n$.

$\mathcal{L}_n^*$ = Additional sample drawn from $\hat{\mathbb{P}}$ (given $\mathcal{L}_n$).



$\mathcal{L}_n^* = \{(X_1^*, Y_1^*), \dots, (X_n^*, Y_n^*)\}$

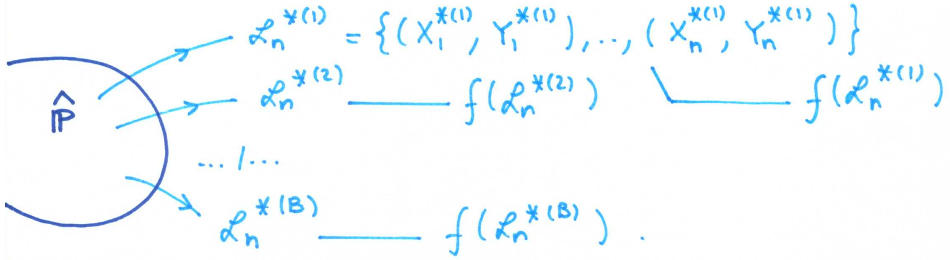
where

$\mathbb{P}\left((X_i^*, Y_i^*) = (X_j, Y_j) \mid \mathcal{L}_n\right) = \frac{1}{n}$

• The <u>BOOTSTRAP PRINCIPLE</u> states that the original $\mathbb{E}_{\mathbb{P}}\{f(\mathcal{L}_n)\}$ can be well approximated by its bootstrap version $\mathbb{E}_{\hat{\mathbb{P}}}\{f(\mathcal{L}_n^*)\}$

This version of the bootstrap is <u>non-parametric</u>. In a <u>parametric</u> context, $\hat{\mathbb{P}}$ is replaced by $\mathbb{P}_{\hat{\theta}}$, where the parameter $\hat{\theta}$ is computed using $\mathcal{L}_n$. Theoretical support of the bootstrap principle is given in the book <u>'The Bootstrap & Edgeworth Expansion'</u> by Peter Hall, in the context of interval & curve estimation.

→ $\mathbb{E}_{\hat{\mathbb{P}}}\{f(\mathcal{L}_n^*)\}$ cannot be computed directly. We need to proceed with a Monte Carlo (MC) approximation to the bootstrap estimate $\mathbb{E}_{\hat{\mathbb{P}}}\{f(\mathcal{L}_n^*)\}$. The simplest MC method is <u>uniform resampling</u>: draw B independent samples from $\hat{\mathbb{P}}$, and approximate $\mathbb{E}_{\hat{\mathbb{P}}}\{\dots\}$ by its empirical mean:


$\hat{\mathbb{P}}$
$\mathcal{L}_n^{*(1)} = \{(X_1^{*(1)}, Y_1^{*(1)}), \dots, (X_n^{*(1)}, Y_n^{*(1)})\}$
$f(\mathcal{L}_n^{*(1)})$
$\mathcal{L}_n^{*(2)}$
$f(\mathcal{L}_n^{*(2)})$
$\dots/\dots$
$\mathcal{L}_n^{*(B)}$
$f(\mathcal{L}_n^{*(B)})$.

$$\boxed{\mathbb{E}_{\hat{\mathbb{P}}}\{f(\mathcal{L}_n^*)\} \approx \frac{1}{B}\sum_{b=1}^{B} f(\mathcal{L}_n^{*(b)}).}$$

Monte-Carlo approximation of the bootstrap version of $\mathbb{E}_{\mathbb{P}}\{f(\mathcal{L}_n)\}$.

→ Proceed similarly to estimate the prediction variance:

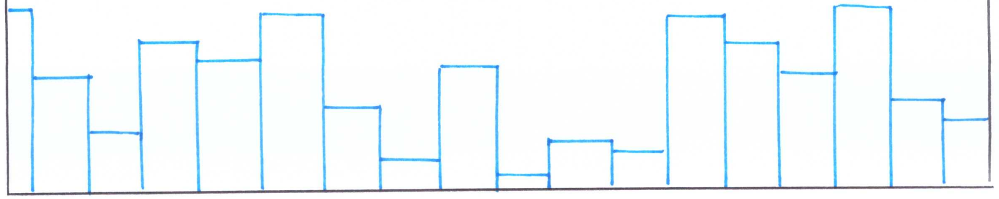
$$\mathbb{E}_{\mathbb{P}}\{(f(\mathcal{L}_n) - \mathbb{E}_{\mathbb{P}} f(\mathcal{L}_n))^2\}$$

bootstrap version

$$\approx \mathbb{E}_{\hat{\mathbb{P}}}\{(f(\mathcal{L}_n^*) - \mathbb{E}_{\hat{\mathbb{P}}} f(\mathcal{L}_n^*))^2\}$$

MC approximation.

$$\approx \frac{1}{B-1}\sum_{b=1}^{B}\left\{f(\mathcal{L}_n^{*(b)}) - \frac{1}{B}\sum_{b'=1}^{B} f(\mathcal{L}_n^{*(b')})\right\}$$



→ The bootstrap replicates can be used to estimate the test error as well : use the bootstrap replicates to fit the model, and the original dataset as the test sample :

$$\mathbb{E}\{R(f(\mathcal{L}_n))\} = \mathbb{E}\{\mathbb{E}[\ell(Y, f(\mathcal{L}_n)(X)) \mid \mathcal{L}_n]\}$$

$$\approx \frac{1}{B}\sum_{b=1}^{B} \mathbb{E}\{\ell(Y, f(\mathcal{L}_n^{*(b)})(X)) \mid \mathcal{L}_n\}$$

$$\approx \frac{1}{Bn}\sum_{b=1}^{B}\sum_{i=1}^{n} \ell(y_i, f(\mathcal{L}_n^{*(b)})(x_i)).$$

Huge overlap between the bootstrap resamples and the original dataset ⇒ underestimation of the risk.

$$\mathbb{P}(\text{observation } i \in \text{bootstrap sample } b)$$
$$= 1 - \mathbb{P}(\text{obs } i \notin \text{bootstrap sample } b)$$
$$= 1 - \left(1 - \frac{1}{n}\right)^n$$
$$\approx 1 - e^{-1}$$
$$\approx 0.632.$$

⇒ About $2/3$ of the original data can be found in any bootstrap replicate.

↓

A better approach is to predict the x_i that did not occur in the current bootstrap resample.

Let $C^{-i} \subset \{1, \dots, B\}$
$=$ set of indices of be bootstrap replicates that do not contain observation i.

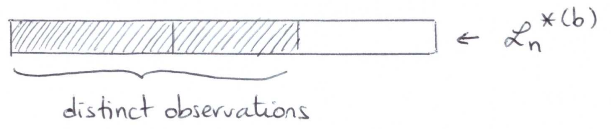
$|C^{-i}| = \#$ elements in C^{-i}

The Leave One Out Bootstrap estimate is

$$\boxed{LOO := \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}} \ell(y_i, f(\mathcal{L}_n^{*(b)})(x_i))}$$

Predict y_i with functions constructed from bootstrap replicates that do not contain x_i.

× Remark: The bootstrap procedure is similar to K-fold CV, with K = 3 since the average number of distinct observations in each bootstrap sample is $\approx 0.632\,n$. → suffers from a bias.



Since $\mathbb{P}((X_i, Y_i) \in \mathcal{L}_n^{*(b)} \mid \mathcal{L}_n) \approx 0.632$,

put $Y_i = 1$ if $(X_i, Y_i) \in \mathcal{L}_n^{*(b)}$, and $= 0$ otherwise.

Then $Y_i \sim Bi(1, 0.632)$.

⇒ $Z = \#$ distinct observations in $\mathcal{L}_n^{*(b)}$

$= \sum_{i=1}^{n} Y_i \sim Bi(n, 0.632)$. ⇒ $\mathbb{E} Z = 0.632n$.

II.2. Bagging

Bagging was introduced by Breiman (1996) in his paper 'Bagging Predictors'. 'Bagging' stands for 'Bootstrap Aggregation'. In the previous section, we learned about the bootstrap principle, and we used the bootstrap to assess the accuracy of an estimator. When bagging predictors, the bootstrap is used to improve the prediction itself, by reducing the variance of the learner.

↳ Each bootstrap sample is used to construct a predictor, and the B predictors are aggregated to produce a final learner. The aggregation averages over the B predictions in the context of regression, and proceeds with a plurality vote in classification problems.

Idea: Averaging reduces the variance : $X_1, \dots, X_n$ iid $\operatorname{Var} X_i = \sigma^2$, then $\operatorname{Var} \bar{X} = \sigma^2/n \ll \sigma^2$, where $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$.

→ Regression Problems let $f(\mathcal{L}_n)$ = predictor constructed from $\mathcal{L}_n$.

Instead of $f(\mathcal{L}_n)$, we would prefer to consider $\mathbb{E}_{\mathbb{P}}\{f(\mathcal{L}_n)\}$.

Bootstrap Version is $\mathbb{E}_{\hat{\mathbb{P}}}\{f(\mathcal{L}_n^*)\}$

MC Approximation is $\frac{1}{B}\sum_{b=1}^{B} f(\mathcal{L}_n^{*(b)})$.

The aggregate estimator is defined as the MC approximation of the bootstrap version of $\mathbb{E}_{\hat{\mathbb{P}}}\{f(\mathcal{L}_n^*)\}$:

$$\boxed{f_{agg}(x) := \frac{1}{B}\sum_{b=1}^{B} f(\mathcal{L}_n^{*(b)})(x).}$$

↓ A simple (and sloppy) justification of why is it a good idea to prefer f_{agg} over $f(\mathcal{L}_n)$:

$$\mathbb{E}\{R(f(\mathcal{L}_n))\} = \mathbb{E}\{\mathbb{E}[[Y - f(\mathcal{L}_n)(X)]^2 \mid \mathcal{L}_n]\}$$

Assume a square loss

$$= \mathbb{E}Y^2 - 2\,\mathbb{E}\{\mathbb{E}[Y f(\mathcal{L}_n)(X) \mid \mathcal{L}_n]\} + \mathbb{E}[f(\mathcal{L}_n)(X)]^2$$

$$= \mathbb{E}_{X,Y}\{\mathbb{E}[Y f(\mathcal{L}_n)(X) \mid X, Y]\}$$

$$= \mathbb{E}_{X,Y}\{Y\, \mathbb{E}[f(\mathcal{L}_n)(X) \mid X, \cancel{Y}]\}$$

$$= \mathbb{E}_{X,Y}\{Y f_{ag\,id}(X)\},$$

Jensen:

$$\geq \mathbb{E}_{X,Y}\{f^2_{ag.id}(X)\}$$

where

$f_{ag\,id} := \mathbb{E}_{\mathbb{P}}\{f(\mathcal{L}_n)\}$ the "ideal" aggregator estimate

$$\Rightarrow \mathbb{E}\{R(f(\mathcal{L}_n))\} \geq \mathbb{E}(Y - f_{ag\,id}(X))^2 = \mathbb{E}\{R(f_{ag\,id})\}$$

(*)

We see from (*) that the ideal aggregate estimator $f_{ag\,id}$ has smaller risk than the original estimator $f(\mathcal{L}_n)$.
In addition, the gain is not substancial when $\geq$ becomes $\approx$; i.e. when Jensen inequality is tight. This happens in particular in cases where $f(\mathcal{L}_n)$ does not fluctuate much around $f_{ag\,id}$ ⇒ Expect bagging to increase the performance of predictors with predictors with high variance (such as trees).

→ Classification problems. Proceed with a majority vote:

$$\boxed{f_{agg}(x) = \underset{1\leq k\leq K}{\operatorname{argmax}} \; \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}\left(f(\mathcal{L}_n^{*(b)})(x) = k\right)}$$

"wisdom of crowds"

↳ The bagged predictor is a vector with K components; $(\pi_1(x), \dots, \pi_K(x))$; where $\pi_k(x)$ = proportion of classifiers predicting class k at x.

aka "voting probabilities".

⚠ Voting probabilities are not estimated class conditional probabilities. Alternatively, when constructing the B classifiers, keep track of the predicted class probabilities $\hat{p}_k^{*(b)}(x)$, and use

$$\hat{p}_{agg,k}(x) := \frac{1}{B}\sum_{b=1}^{B} \hat{p}_k^{*(b)}(x)$$

$$f_{agg}(x) := \underset{1\leq k\leq K}{\operatorname{argmax}} \; \hat{p}_{agg,k}(x)$$

In the classification setting as well, bagging classifiers work best with unstable learners, such as trees. According to Breiman:
« The vital element is the instability of the prediction method. If perturbing the learning set can cause significant changes in the predictor constructed, then bagging can improve accuracy. »

× Remark: Use "Out Of Bag" (OOB) observations to estimate the test error
↑ obs. not included in the bootstrap samples.

Let $C^{-i} \subset \{1, -, B\}$
$= $ indices of bootstrap samples $\mathcal{L}_n^{*(b)}$ that do not contain observation i.

→ Regression: $\hat{y}_i^* := \frac{1}{|C^{-i}|} \sum_{b \in C^{-i}} f(\mathcal{L}_n^{*(b)})(x_i)$

→ Classification: $\hat{y}_i^* := \underset{1 \le k \le K}{\operatorname{argmax}} \sum_{b \in C^{-i}} \mathbb{1}(f(\mathcal{L}_n^{*(b)})(x_i) = k)$

Then $\boxed{OOB = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \hat{y}_i^*)}$.

× Remark: "Bagging a good classifier can improve its prediction accuracy; while bagging a bad one can seriously degrade its performance". We illustrate this on a simple example.
- Binary classification; with two classes $\{0, 1\}$.
- B independent classifiers f_b, $b = 1, -, B$; and each have a misclassification rate equal to $\alpha \in (0,1)$

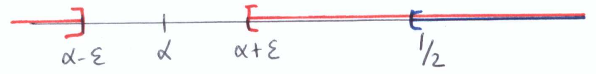
The bagged classifier is $f_{agg} = \underset{k \in \{0,1\}}{\operatorname{argmax}} \sum_{b=1}^{B} \mathbb{1}(f_b(x) = k)$.

Let $B_0 = \#$ votes for class 0.
$= \sum_{b=1}^{B} \mathbb{1}(f_b(x) = 0) \sim Bi(B, \alpha)$
↑ Assuming that the true label of x is 1; so that $\mathbb{P}(f_b(x) = 0) = \alpha$.

The misclassification rate of f_{agg} is $\mathbb{P}(f_{agg}(x) = 0)$
$= \mathbb{P}(B_0 \geqslant \frac{B}{2})$.

- If $\alpha = 0.4$, then $\mathbb{P}(B_0 \geqslant \frac{B}{2}) \to 0$; so that the aggregate classifier has a perfect predictive accuracy as $B \to +\infty$.
- If $\alpha = 0.6$, then $\mathbb{P}(B_0 \geqslant \frac{B}{2}) \to 1$; and the bagged classifier becomes perfectly inaccurate.

(Note that $\mathbb{P}(B_0 \geqslant B/2) \to 0$ when $\alpha = 0.4$ e.g. follows from the WLLN; since $\frac{B_0}{B} \to \alpha = 0.4$ as $B \to \infty$.
Thus $\forall \varepsilon > 0$, $\mathbb{P}(|\frac{B_0}{B} - \alpha| > \varepsilon) \to 0$ as $B \to +\infty$;
and $\mathbb{P}(\frac{B_0}{B} \geqslant \frac{1}{2}) \leqslant \mathbb{P}(|\frac{B_0}{B} - \alpha| > \varepsilon) \to 0$)



II. 3. Random Forests.

Random Forests were introduced by Breiman (2001). They are a simple modification of bagging in the context of tree predictors.

Some heuristics: Let $X_1, \dots, X_B$ identically distributed, with pairwise correlation ρ. Put $\bar{X} = \frac{1}{B} \sum_{i=1}^{B} X_i$.

$$\operatorname{Var} \bar{X} = \frac{1}{B^2} \operatorname{Var}\left(\sum X_i\right) = \frac{1}{B^2} \sum_{i,j} \operatorname{Cov}(X_i, X_j)$$
$$= \frac{1}{B^2} \left\{ \sum_{i=1}^{B} \operatorname{Var} X_i + \sum_{i \neq j} \operatorname{Cov}(X_i, X_j) \right\}$$
$$= \rho \sigma^2 + \frac{1-\rho}{B} \sigma^2$$

If $\rho = 0$, then $\operatorname{Var} \bar{X} = \frac{\sigma^2}{B} \to 0$ as $B \to +\infty$.
If $\rho \neq 0$, then $\operatorname{Var} \bar{X} \not\to 0$.
$\Rightarrow$ Correlation amongst variables (trees) reduces the benefits of bagging.

Idea: When growing a tree, a random selection of m predictors is chosen as split variables. Trees are grown until a minimum node of size n_{min} is reached.

↳ RF have two tuning parameters: m & n_{min}.

Rule of Thumb: [Regression] $m = \lfloor d/3 \rfloor$, $n_{min} = 5$

[Classification] $m = \lfloor \sqrt{d} \rfloor$, $n_{min} = 1$.

These are the default values in the R package randomForest.

See for example Liaw & Wiener (2002), and Díaz-Uriarte & de Andrés (2006).

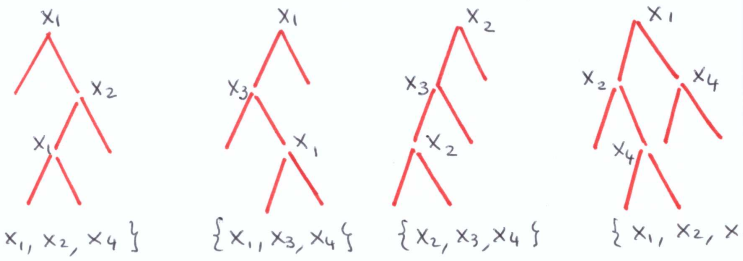
— Not supported by solid theory —

The effect of m is investigated in D-U & dA, who show that this parameter has little importance in the accuracy of the method, m should just not be taken too large.

→ No need of a validation / hold-out set to tune these parameters: OOB observations may be used to assess the accuracy of the method (see page 25).

• Illustration.

d = 4 (4 features).



Selection of $m = 3$ features ↗

→ The default value of B (the number of bootstrap samples) in the R package RandomForest is set to 500. Moreover, selecting a large B does not overfit the data (see Breiman (2001), Biau & Scornet (2016)). The choice of B results from a trade-off between computational cost (B not too large), and accuracy (B large enough to produce a stable estimate).

× Remark = Variable Importance.

When bagging trees, we gain in accuracy, but we loose in interpretability. The relative contribution of each predictor to the final estimate can be evaluated using a Variable Importance indicator.

For a single tree, define

$$I_j^2(T) = \sum_{m=1}^{M} \Delta_m^2 \, \mathbb{1}(\text{split variable at node } m = j).$$

Importance of the j-th predictor, $1 \leq j \leq d$

Sum over the M interior nodes.

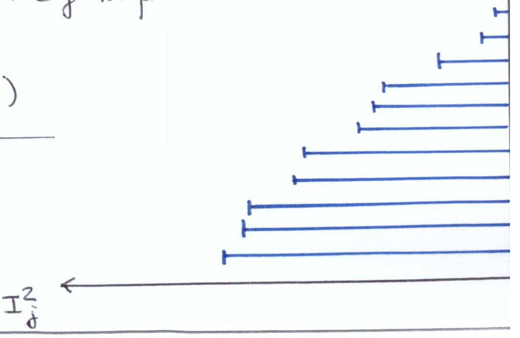
Decrease of the criterion used to grow the tree (Squared Error or Gini Index).

↓

Average the importance of the j-th predictor over all trees:

$$I_j^2 = \frac{1}{B} \sum_{b=1}^{B} I_j^2(T_b)$$

How much the j-th variable reduces the goodness-of-fit criterion / impurity measure



Alternatively, the importance of the j-th variable can be evaluated using a reshuffling procedure:

(i) Grow the b-th tree

(ii) Use OOB observations to estimate the prediction accuracy of the b-th tree

(iii) Repeat the procedure, when randomly shuffling the j-th predictor in the OOB sample & record the decrease in accuracy, compared to (ii)

(iv) Average out over all B trees.

- References

TREES

- L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone. Classification and Regression Trees. Chapman & Hall (1984)
- J.R. Quinlan. Induction of Decision Trees. Machine Learning 1, p. 81-106 (1986)
- J.R. Quinlan. C4.5 Programs for Machine Learning, San Mateo, CA: Morgan Kaufmann (1993)

BAGGING

- B. Efron. Bootstrap Methods: Another look at the Jacknife. Ann. Stats, Vol 7, No 1, p. 1-26 (1979)
- P. Hall. The bootstrap & Edgeworth Expansion. Springer Verlag (1992)
- L. Breiman. Bagging Predictors. Machine Learning, Vol 24. p. 123-140 (1996)

RANDOM FORESTS

- L. Breiman. Random Forests. Machine Learning, Vol 45, p. 5-32 (2001)
- A. Liaw & M. Wiener. Classification and Regression by randomForest. R news, Vol 2, p. 18-22 (2002)
- R. Díaz-Uriarte & S. Alvarez de Andrés. Gene Selection & Classification of Microarray Data using Random Forest. BMC Bioinformatics, Vol 7, p. 1-13 (2006)
- G. Biau & E. Scornet. A Random Forest Guided Tour. TEST. Vol 25, Issue 2, pp. 197-227 (2016)