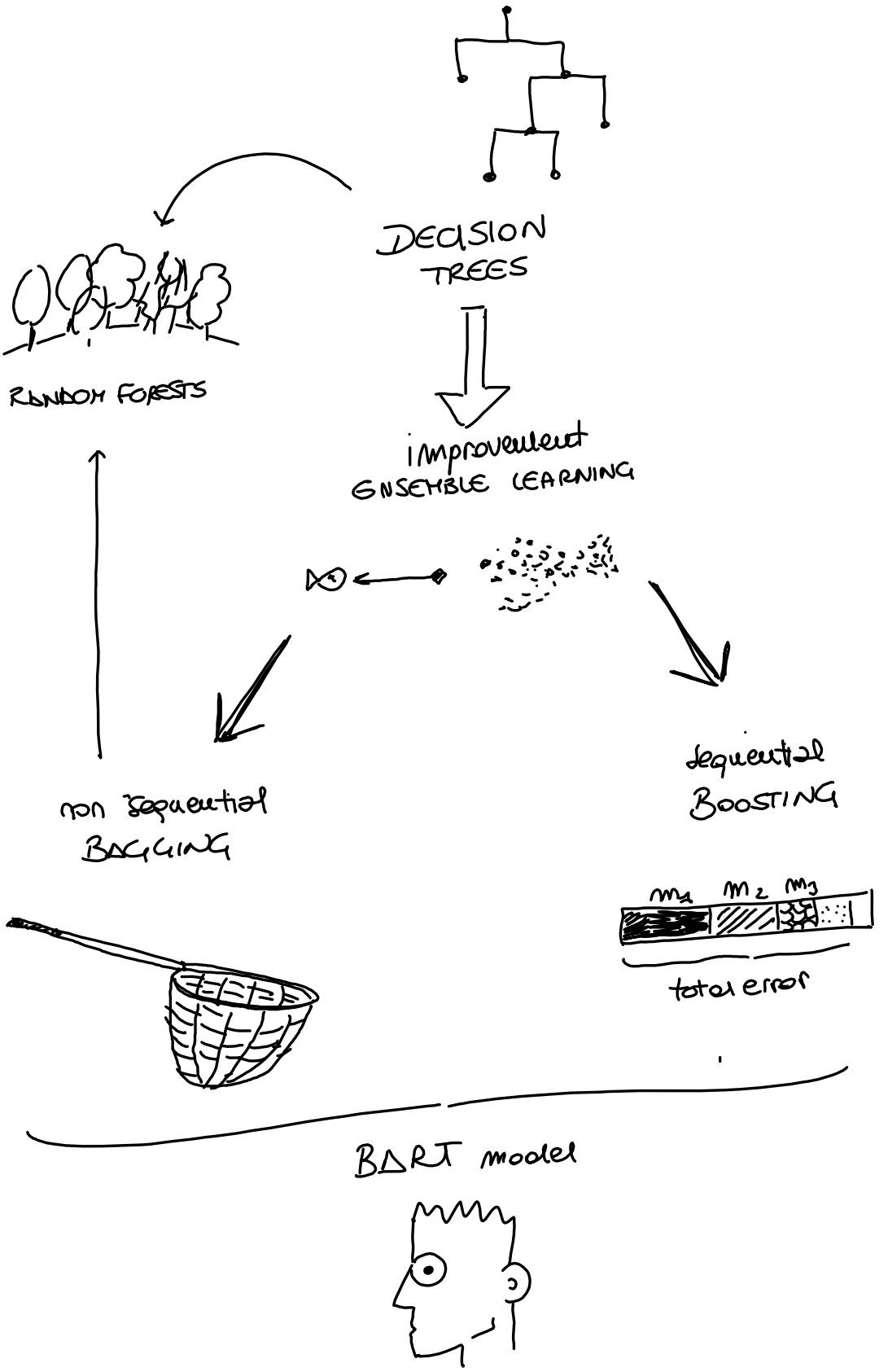
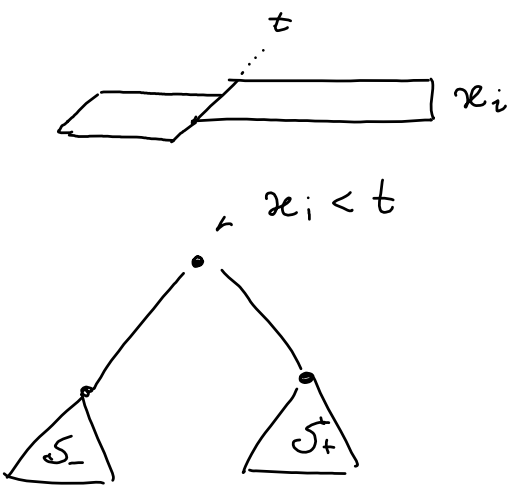

DECISION TREES
RANDOM FORESTS
improvement
ENSEMBLE LEARNING
non sequential
BAGGING
sequential
BOOSTING
m1 m2 m3
total error
BART model

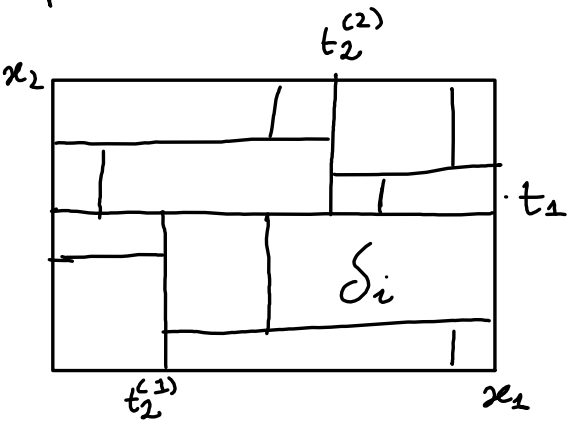
DECISION TREES

$x_1, \dots, x_m$ features $\quad y = \{C_1, \dots, C_m\}$. class

Recursive Definition



Each split corresponds to a subdivision of the feature space:

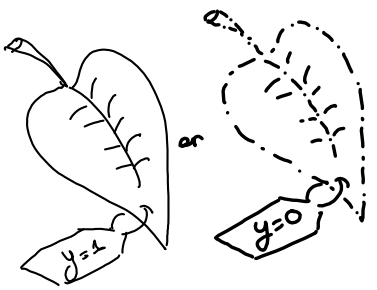


At each level we try to find best pair of (x_i, t) to divide the data in order to:

- minimize RSS (REGRESSION)
- maximize IG (CLASSIFICATION)

Prediction Formula: In every hyperplane S_i of the feature space the same formula for prediction is used.

final leaves



$$f_S := \begin{cases} \frac{1}{|S|} \sum_{i \in S} y_i = \bar{y}_S & \text{in Regression} \\ \frac{1}{|S|} \sum_{(\underline{x}, y) \in S} y & \text{in CLASSIFIC.} \end{cases}$$

LIKE: inverted tree.

DECISION TREES (REGRESSION)

$$f_S = \bar{y}_S = \frac{1}{|S|} \sum_{y \in S} y$$

(prediction $\forall$ S hyp. in feature space)

GOAL FUNCTION:

$$\min_{J} \underbrace{\sum_{j=1}^{J} \sum_{i \in S_j} (y_i - \bar{y}_{S_j})^2}_{RSS}$$

J = # of subsets S we find.

The goal is finding the best set of $S_1, \dots, S_J$ to divide the feature space, so that the predictions we make minimize the goal function.

SOLUTION:

Top-down building of the tree.

At each level we seek the best pair (x_j, t) such that splitting the feature space into:

leads to biggest reduction in RSS.

DECISION TREES (CLASSIFICATION)

mathematical form of the optimization problem.

the threshold t is found as the value that, each split, maximizes the information gain:

$$IG_t(S) = H(S) - \sum_{i=1}^{2} \frac{|S_i|}{|S|} H(S_i)$$

this means that at every split we seek the t that divides the training set in the most balanced way:

$$\min IG_t(S) = \max \sum_{i=1}^{2} \frac{|S_i|}{|S|} H(S_i)$$

where $S_r := \{$ record in the root $\}$

Eventually the IG we can achieve by splitting the node will be very small.

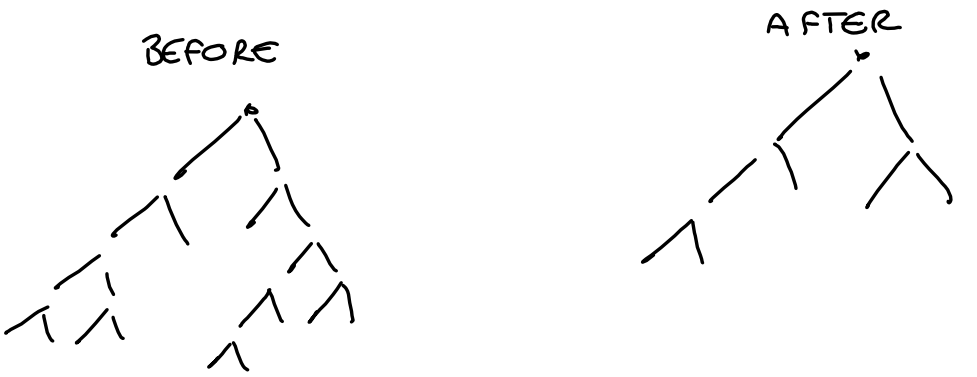
$$|IG| < \varepsilon$$

This means that the leaf we arrived at during the splitting process cannot be divided into 2 balanced subsets of records. So the leaf is already very imbalanced: its records belong mostly to the same class. So:

$$H(S_r) = -\sum_{i=1}^{m} f_i \ln f_i \simeq -1 \ln 1 \simeq \emptyset$$

DECISION TREES (PRUNING)

PROBLEM: decision trees tend to overfit the data. They become too complex and the feat. space (X) subd. too specific.



Pruning is the operation of undoing the splits (rejoining 2 subspaces of the feature space) that grant the less advancement wrt the optimization problem

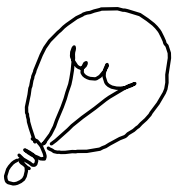
PRO:

- reducing overfitting tendency.

→ not ↓ RSS (Regression).

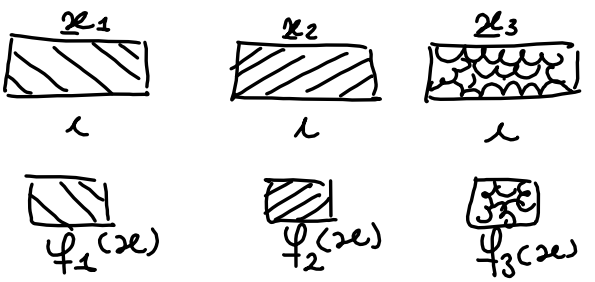
→ not ↑ TIG (classif.)

BAGGING ..



models that are good will tend to agree on the same prediction, while bad models tend to disagree on different predictions

B bootstrap samples from the training data



train a weak model on each one of them

prediction happens through voting of weak models

$$y \leftarrow f(x) = \frac{1}{B} \sum_{i=1}^{B} f_i(x)$$

BAGGING

theory.

Bagging is a method used for reducing the variance of a statistical learning method.

$$f(x) = \frac{1}{B} \sum_{i=1}^{B} f_i(x)$$

The model prediction is computed as average output of B other models, trained on Bootstrap sets.

Infact Bagging stands for Bootstrap Aggregation.
Since we output an avg., from the CLT:

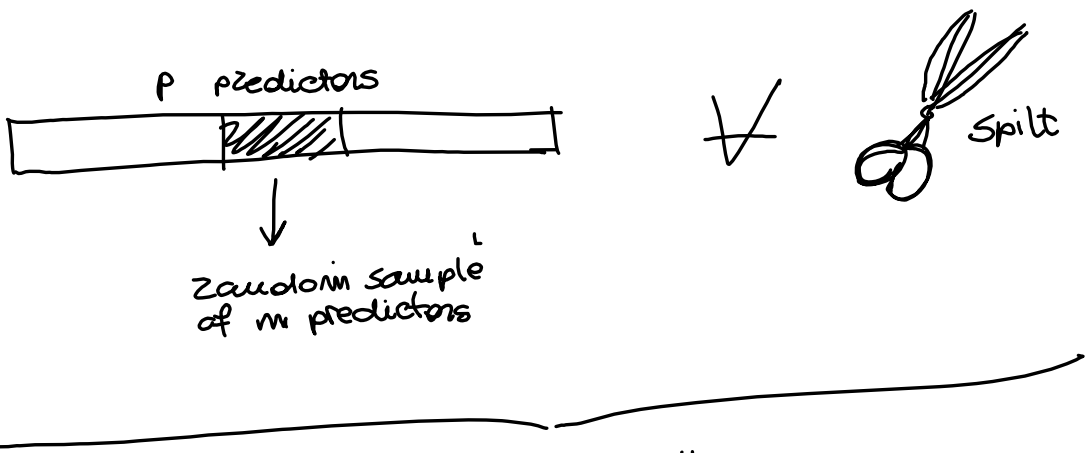
$$f(x) \overset{d}{\sim} \mathcal{N}\left(\bar{f}(x) = y;\ \frac{\mathbb{V}(f(x))}{B}\right)$$

As B increases the final model variance decreases.
This is why we operate bagging with high variance learning methods.

Bagging improves prediction accuracy at the expenses of readibility.

RANDOM FORESTS

We apply a bagging procedure to a decision tree. Each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates.



Avoiding collinearity between bagged trees

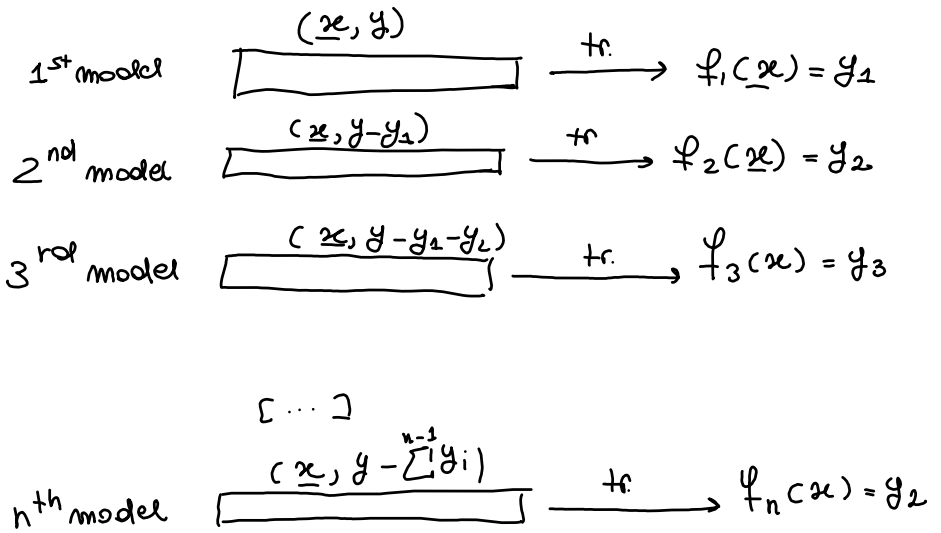
RANDOM FOREST

Having correlated trees doesn't lead to a reduction in variance as having many un-correlated trees. For this reason RANDOM FORESTS represent a quick improvement of bagged trees

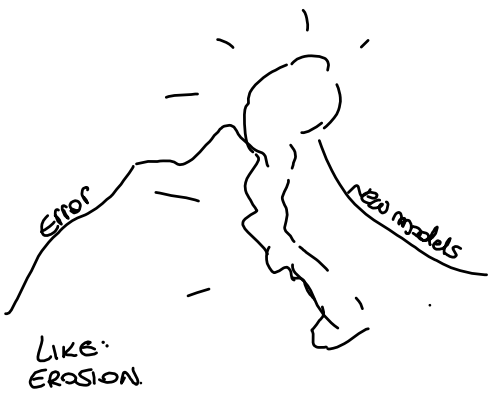
NOTE: usually $m \simeq \sqrt{p}$.

Boosting:

Create models to predict the errors the other ones make.



boosted model: $f(\underline{x}) = \sum_{i=1}^{n} \alpha f_i(x)$



Each new model is trained having as labels the error of the previous ones:

$$y^{(k)} = y - \sum_{i=1}^{k-1} f_i(\underline{x})$$

NOTE: Similarity to Grad. Desc. At each step the overall error is implicitly reduced by training a new mod. on the past error.

STEP UPD.) $x_{k+1} = x_k - \alpha \nabla f(x_k)$

RESID. UPD.) $y_{k+1} = y_k - \alpha f_k(x)$

For this reason, Boosting is also referred as Gradient Boosting.

Boosting: hyperparameters

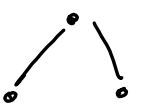
1) B := nº of trees to train

chosen through cross validation

2) α := learning rate (step of GD)

Typical values: $\alpha = 10^{-2}, 10^{-3}$

3) d := nº of splits in each tree
this controls the complexity of the boosted ensemble.
Often: $d = 1$

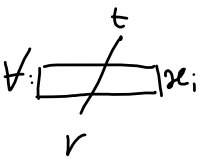


BOOSTING:

example algorithm.

Training data are weighted:

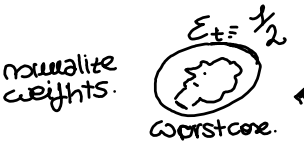
NOTE: $y_i = \begin{cases} 1 \\ -1 \end{cases}$



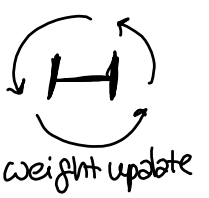
AdaBoost:

FOR $t = 1, \dots, T$:

- choose weak classifier:
  - find weak learner $h_t(x)$ that minimize the weighted sum of errors: $\varepsilon_t = \sum_i \omega_{i,t} \leq 0,5$
  - choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$
- ensemble:
  - $F_t(x) = F_{t-1}(x) + h_t(x) \Big] = h_1(x) + h_2(x) + \dots$
- update weights:
  - $w_{i,t+1} = (w_{i,t}) e^{-y_i \alpha_t h_t(x_i)} \quad \forall i \Big]$
  - Renormalize weights.



$$-y_i \alpha_t h_t(x_i) = \begin{cases} > 0 & \text{WRONG PRED. } h_t(x_i) \neq y_i \rightarrow \uparrow \omega_i \\ < 0 & \text{RIGHT PRED. } h_t(x_i) = y_i \rightarrow \downarrow \omega_i \end{cases}$$



BART (for regression)

Bayesian Additive Regression trees

K = # regression trees

B = # iterations

$f_k^{(b)}(x)$ = prediction at x of the k-th regression tree at iteration b

n = # train. records

1. $f_1^{(1)}(x) = \cdots = f_K^{(1)}(x) = \frac{1}{nK} \sum_{i=1}^{n} y_i$] like bagging

2. $\varphi^{(1)}(x) = \frac{1}{n} \sum_{i=1}^{n} f_i^{(1)}(x)$

3. for $b = 2, \ldots, B$

   a) for $k = 1, \ldots, K$:

      I) for $i = 1, \ldots, n$ compute partial residual] like boosting

      $$r_i = y_i - \sum_{k' \neq k} f_{k'}^{(b)}(x)$$

      II) Fit a new tree $\varphi_k^{(b)}(x)$ by rand. perturbating $\varphi_k^{(b-1)}(x)$.

   b) compute $f^{(b)}(x) = \sum_{k=1}^{K} \varphi_k^{(b)}(x)$

4. Compute the mean after L burn-in samples]

$$\varphi(x) = \frac{1}{B-L} \sum_{b=L+1}^{B} \varphi^{(b)}(x)$$

In words:

1, 2: initialization of K regression trees. $\varphi^{(1)}(x)$ is computed as avg. output of the K trees

3.: B iterations. At each iteration everyone of the K trees is updated from its previous version, by a random modification

4.: the model is an avg. of the not-burnt prediction models.

BART (for regression)

Key insights

3. At this step we do not fit a new tree to the current partial residual error but a modification of the previous one

   - AVOID overfitting: improving the older tree prevents from fitting «hard» the data at each iteration

4. The burn-in period (the first L iterations) corresponds to the number of models we do not consider in the final version model. Generally the first models found $(f^{(1)}(x), \dots, f^{(L)}(x))$ tend to not provide very good results.

4. As in bagging the final model is an average of the best models we have found:

$$f(x) = \frac{1}{B-L} \sum_{i=L+1}^{B} f^{(i)}(x)$$

$$= \frac{1}{B-L} \sum_{i=L+1}^{B} \sum_{k=1}^{K} f_k^{(i)}(x)$$

Having 2 ang in the final model, maximally leverages the variance reduction effect of the mean.

As CLT states:

$$\bar{X} \overset{d}{\sim} \mathcal{N}\left(E[\tilde{X}], \frac{\mathbb{V}[\tilde{X}]}{n}\right)$$

BART
hyperparameters setting

For how it is thought: the BART:

- prevents overfitting
- is good out of the box

Standard value for its hyperparameters are:

$K = 200$ (n° of trees)

$B = 1000$ (n° of iterations)

$L = 100$ (burn-in period)