Statistical and Sequential Learning for Time Series Forecasting

Bagging, Random Forest, Boosting

Margaux Brégère

Classification And Regression Tree - CART

- Segmentation criteria for classification
- Segmentation criteria for regression
- Algorithm

Bagging

- Bootstrap
- Prediction error diminution

Random Forest

- Algorithm
- Out of bag error and importance

Boosting

- Adaboost
- Gradient boosting

Online approches

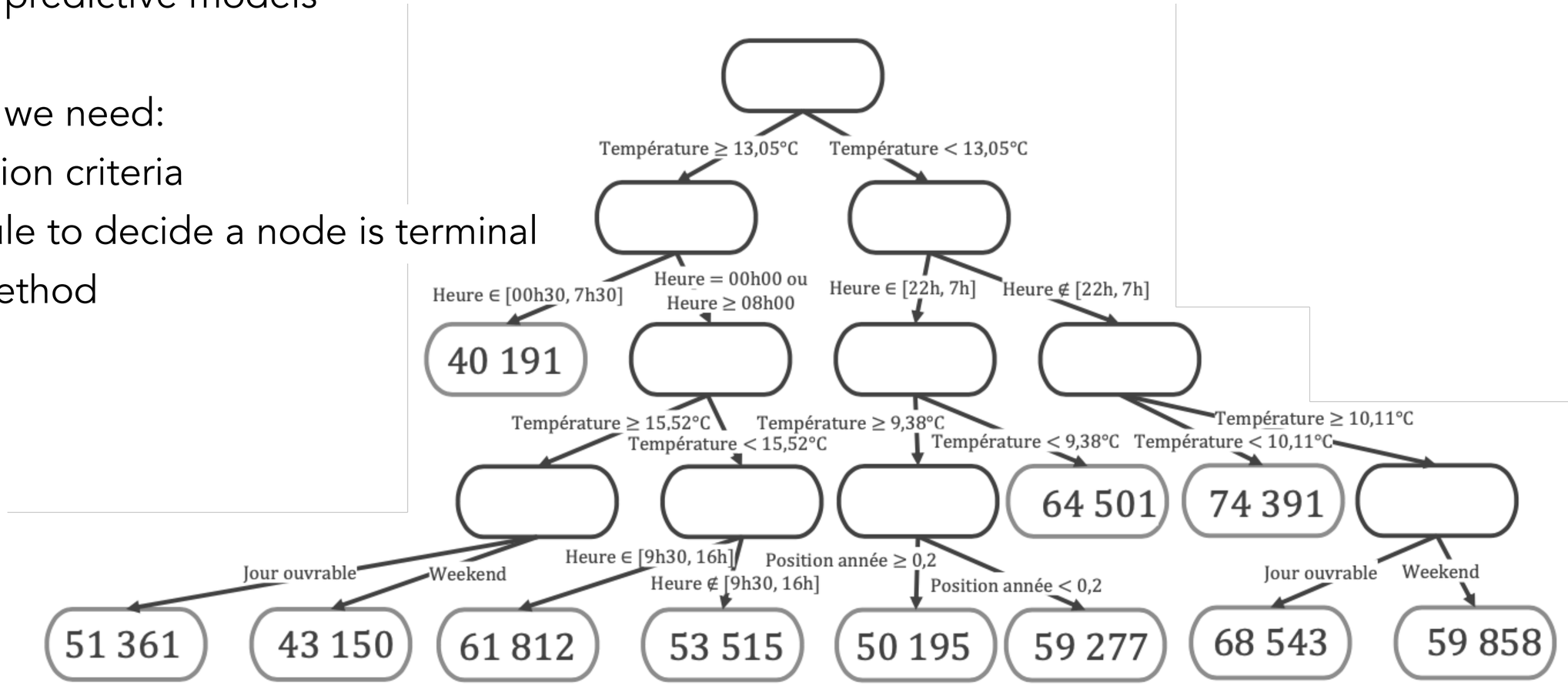
Classification And Regression Tree - CART

Framework

Classification And Regression Tree - CART are explicative and predictive models

To build a tree, we need:

- a segmentation criteria
- a decision rule to decide a node is terminal
- a pruning method



Segmentation criteria for classification

For a node R_k , with M the number of classes, the Gini impurity is:

$$I_{\text{Gini}}(R_k) = \sum_{m=1}^{M} \frac{\sum_{i \in R_k} \mathbf{1}_{Y_i = m}}{|R_k|} \left(1 - \frac{\sum_{i \in R_k} \mathbf{1}_{Y_i = m}}{|R_k|} \right)$$

$\frac{\sum_{i \in R_k} \mathbf{1}_{Y_i = m}}{|R_k|} = \frac{|R_k(m)|}{|R_k|}$ is the proportion of data that are labelled m in the node R_k

For classification, CART algorithm aims to find the leaves $R_1, \ldots R_K$ which minimise

$$\sum_{k=1}^{K} I_{\text{Gini}}(R_k)$$

When is $I_{\text{Gini}}(R_k)$ the higher? And the lower?

Gini Impurity

- $I_{\text{Gini}}(R_k) = 0$ if all the elements of R_k have the same label m
- $I_{\text{Gini}}(R_k) = 1 - \frac{1}{M}$ and is maximal for a uniform repartition of the labels in R_k

Proof:

Gini index is nonnegative since it is the sum of nonnegative terms and using the Cauchy-Schwarz inequality $1 = \left(\sum_{m=1}^{M} \frac{|R_k(m)|}{|R_k|}\right)^2 \leq \sum_{m=1}^{M} 1^2 \sum_{m=1}^{M} \left(\frac{|R_k(m)|}{|R_k|}\right)^2 = M \sum_{m=1}^{M} \left(\frac{|R_k(m)|}{|R_k|}\right)^2$, we get

$$0 \leq I_{\text{Gini}}(R_k) = \sum_{m=1}^{M} \left(\frac{|R_k(m)|}{|R_k|} - \left(\frac{|R_k(m)|}{|R_k|}\right)^2\right) = 1 - \sum_{m=1}^{M} \left(\frac{|R_k(m)|}{|R_k|}\right)^2 \leq 1 - \frac{1}{M}$$

□

Segmentation criteria for regression

For regression, CART algorithm aims to find the leaves $R_1, \ldots R_K$ which minimise the RSS (Residuals Sum Squared) criteria:

$$\mathrm{RSS} = \sum_{k=1}^{K} \sum_{i \in R_k} \left(Y_i - \bar{Y}_k\right)^2 \text{ where } \bar{Y}_k = \frac{1}{|R_k|} \sum_{i \in R_k} Y_i$$

For a leaf R_k, what is $\sum_{i \in R_k} \left(Y_i - \bar{Y}_k\right)^2$?

Segmentation criteria for regression

For regression, CART algorithm aims to find the leaves $R_1, \dots R_K$ which minimise the RSS (Residuals Sum Squared) criteria:

$$\mathrm{RSS} = \sum_{k=1}^{K} \sum_{i \in R_k} \left(Y_i - \bar{Y}_k\right)^2 \text{ where } \bar{Y}_k = \frac{1}{|R_k|} \sum_{i \in R_k} Y_i$$

For a leaf R_k, what is $\sum_{i \in R_k} \left(Y_i - \bar{Y}_k\right)^2$?

$\frac{1}{|R_k|} \sum_{i \in R_k} \left(Y_i - \bar{Y}_k\right)^2$ corresponds to the empirical variance of the leaf R_k! Thus $\mathrm{RSS} = \sum_{k=1}^{K} |R_k| \mathbb{V}(R_k)$

Algorithm

For both classification and regression, find the optimal partition→ $R_1, \dots R_K$ is an NP-hard problem

→ recursive Greedy-approach

Initialisation: all the observation $(Y_i, X_i)_{1,\dots,n}$ are in the same node (the root of the tree)

While « some nodes are not terminal » or « the criteria to stop is false »

For nodes $k = 1,\dots$

Find the best split, i.e the variables $j^\star \in 1,\dots p$ and the threshold $s^\star$ such that

$$(j^\star, s^\star) \in \arg\min_{j,s} |R_{k_L}(j,s)| I_{\text{Gini}}(R_{k_L}(j,s)) + |R_{k_R}(j,s)| I_{\text{Gini}}(R_{k_R}(j,s)) \quad \text{— classification}$$

$$(j^\star, s^\star) \in \arg\min_{j,s} \sum_{i \in R_{k_L}(j,s)} \left(Y_i - \bar{Y}_{k_L}\right)^2 + \sum_{i \in R_{k_R}(j,s)} \left(Y_i - \bar{Y}_{k_R}\right)^2 \quad \text{— regression}$$

where $R_{k_L}(j,s) = \left\{i \in R_k \mid X_{ij} \leq s\right\}$ and $R_{k_R}(j,s) = \left\{i \in R_k \mid X_{ij} > s\right\}$

And cut the node into the two leaves $R_{k_L}(j^\star, s^\star)$ and $R_{k_R}(j^\star, s^\star)$ if it is not terminal

Stopping criteria and pruning

Stoping criteria:

- The node R_k is cut only if it contains more than $n_{\text{min split}}$ observations
- The node R_k becomes terminal if it contains less than $n_{\text{min bucket}}$ observations
- The node R_k is cut only if the segmentation criteria is reduced of at least $\delta > 0$

The tree may be overfitted ($n_{\text{min split}} = n_{\text{min bucket}} = 1$ and $\delta = 0$)

The pruning gathers leaves when the prediction error decreases (cross-validation or data test)

As for Ridge regression, it is also possible to minimise a criteria $R_\lambda(T) = R(T) + \lambda |T|$, where $|T|$ is the depth of the tree T

Take-home messages

- Easy and efficient implementation
- Any assumption on variable distributions
- Can model discontinuous and non-functional phenomena
- Approximate correctly continuous phenomena (with piecewise constant function)
- Adapted to the case where the number of explanatory variables is large (perform variable selection)
- Interpretable
- Robust to outliers

Bagging

Bootstrap (B. Efron, 1979)

General framework:

To estimate a quantity $\theta = T(F)$ from a sample $Y_1, \ldots, Y_n$ of n observations, which are independent and identically distributed according to the unknown distribution law F:

For $b = 1,\ldots, B$

- Draw a sample $Y_1^b, \ldots, Y_n^b$ from $Y_1, \ldots, Y_n$ with replacement
- Estimate θ with $\hat{\theta}^b$

Final estimation = bootstrap aggregation or bagging: $\theta = \frac{1}{B}\sum_{b=1}^{B} \hat{\theta}^b$

It also possible to estimate the standard error of statistic for θ and to obtain some confidence intervals

Bagging and prediction error diminution

Intuition:

Aggregate independent base learners predictions will reduce the prediction error

To obtain independent base learners, they should be trained on disjointed samples

→ too restrictive, so bootstrap samples are used

For a new observation Y_{new}, with $\hat{Y}^b_{\text{new}}$ the prediction made thanks to a base learner trained on the b -bootstrap sample, the final prediction satisfies

$$\left(\hat{Y}_{\text{new}} - Y_{\text{new}}\right)^2 = \left(\frac{1}{B}\sum_{b=1}^{B} \hat{Y}^b_{\text{new}} - Y_{\text{new}}\right)^2 \leq \frac{1}{B}\left(\hat{Y}^b_{\text{new}} - Y_{\text{new}}\right)^2$$

So the prediction error of the bagging model is lower than the mean of the errors of the base learners (this is all more true when base learners are unstable - with high variance)

Bagging and prediction error diminution

Indeed, if the predictions are correlated this way:

$$\mathrm{Cor}\left(\hat{Y}^b, \hat{Y}^{b'}\right) = \begin{array}{lll} 1 & \text{if} & b = b' \\ \rho & \text{if} & b \neq b' \end{array} \quad \text{with } \mathbb{V}(\hat{Y}^b) = \sigma^2$$

$$\text{then } \mathbb{V}(\hat{Y}) = \mathbb{V}\left(\frac{1}{B} \sum_{b=1}^{B} \hat{Y}^b \right) = \frac{1}{B^2}\left(B\sigma^2 + B(1-B)\rho\sigma^2\right) = \rho\sigma^2 + \frac{\sigma^2}{B}(1-\rho)$$

Random Forest

Random Forest (Breiman, 2001)

Algorithm:

Input: sample $(Y_1, X_1), \ldots, (Y_n, X_n)$ of n observations

For $b = 1, \ldots, B$

- Draw a bootstrap sample $(Y_1^b, X_1^b), \ldots, (Y_n^b, X_n^b)$ from $(Y_1, X_1), \ldots, (Y_n, X_n)$ with replacement
- Perform the CART algorithm with the following modification:

  At each node, find the best split among a subset of explanatory variables of size q

Aggregate the B trees

If $q = p$, CARTs are different only because of the bootstrap procedure

If $q = 1$, at each node, the choice of the variable (but not the threshold) is totally random

Generally, we set $q = p/3$ for regression and $q = \sqrt{p}$ for classification

Out Of Bag error and variables importance

For $b = 1,\ldots, B$

- Draw a bootstrap sample $(Y_1^b, X_1^b), \ldots, (Y_n^b, X_n^b)$ and train the CART algorithm
- Evaluate the prediction error $\mathrm{e}_{\mathrm{OBB}}^b$ of the CART on $\left\{(Y_1, X_1), \ldots, (Y_n, X_n) \setminus (Y_1^b, X_1^b), \ldots, (Y_n^b, X_n^b)\right\}$

→ Get an estimation of the prediction error of the random forest: $\mathrm{e}_{\mathrm{OBB}} = \dfrac{1}{B} \mathrm{e}_{\mathrm{OBB}}^b$

To measure the importance of the explanatory variable X_j :

- The values $\left(X_{1j}, \ldots X_{ij}, \ldots X_{nj}\right)$ are randomly permuted according to a permutation π
- The OOB error is computed on both perturbed and non-perturbed data and the importance is defined as:

$$\mathrm{Importance}(X_j) = \mathrm{e}_{\mathrm{OBB}}\left((Y_i, X_{1i}, \ldots, X_{j\pi(i)}, \ldots, X_{pi})_{i=1,\ldots,n}\right) - \mathrm{e}_{\mathrm{OBB}}\left((Y_i, X_{1i}, \ldots, X_{ji}, \ldots, X_{pi})_{i=1,\ldots,n}\right)$$

The greater the importance, the greater the impact on the forecast!

Take-home messages

- Design to avoid overfitting
- Cross-validation generally non necessary
- Possible parallelisation

- Long training
- Lost of interpretability
- Need to calibrate hyper-parameter

Boosting

Intuition

Train iteratively base learners on weighted residuals and adding them to a final strong learner

Adaboost = adaptative boosting for classification

Y. Freund and R. Schapire (1995) - 2003 Gödel Prize

Inputs: Sample $(Y_1, X_1), \ldots, (Y_n, X_n)$ of n observations, with $Y_i \in \{-1, 1\}$

Initialisation: $\omega_1 = \ldots = \omega_n = \frac{1}{n}, \hat{f}^0 = 0$

For $m = 1, \ldots, M$

Find the weak learner $\hat{h}^m \in \arg\min_h \sum_{i=1}^{n} \omega_i \mathbf{1}_{h(X_i) \leq Y_i}$

Set $\alpha^m = \frac{1}{2} \ln \frac{1 - \varepsilon^m}{\varepsilon^m}$, where $\varepsilon^m = \sum_{i=1}^{n} \omega_i \mathbf{1}_{\hat{h}^m(X_i) \leq Y_i}$

Add the weak learner to the strong learner $\hat{f}^m(X) = \hat{f}^{m-1}(X) + \alpha^m \hat{h}^m(X)$

Update weights $\omega_i = \omega_i \exp\left(-\alpha^m \hat{h}^m(X_i) Y_i\right)$ and renormalise them

Output: $\hat{f}^M$

Gradient Boosting for L_2 regression

Inputs: Sample $(Y_1, X_1), \ldots, (Y_n, X_n)$ of n observations, with $Y_i \in \mathbb{R}$

Initialisation: $\hat{f}^0 = \dfrac{1}{n} \displaystyle\sum_{i=1}^{n} Y_i$

For $m = 1, \ldots, M$

Find the base learner $\hat{h}^m \in \arg\min_h \displaystyle\sum_{i=1}^{n} \left(Y_i - \left(\hat{f}^{m-1}(X_i) + h(X_i) \right) \right)^2$

so $\hat{h}^m$ predicts the residuals since it minimises $\displaystyle\sum_{i=1}^{n} \left(\varepsilon_i^{m-1} - h(X_i) \right)^2$ where $\varepsilon_i^{m-1} = Y_i - \hat{f}^{m-1}(X_i)$

Add the base learner to the strong learner $\hat{f}^m(X) = \hat{f}^{m-1}(X) + \hat{h}^m(X)$

Output: $\hat{f}^M$

Why Gradient?

Gradient Descent Algorithm:

Aim: solve $\hat{x} \in \arg\min_x f(x)$

For $m = 1, \ldots, M$

$$x_{m+1} = x_m - \alpha_m \nabla f(x_m)$$

Gradient Boosting:

Aim: solve $\hat{f} \in \arg\min_f \frac{1}{n} \sum_{i=1}^{n} \left(Y_i - f(X_i)\right)^2 = \min_f \frac{1}{n} L(Y_i, f(X_i))$

For $m = 1, \ldots, M$

$$-\left(\frac{\partial L\left(Y_i, f(X_i)\right)}{\partial f(X_i)}\right)_{f(X_i)=\hat{f}^m(X_i)} = 2\left(Y_i - \hat{f}^m(X_i)\right) = 2\varepsilon_i^m$$

→ By trying to predict ε^m, the weak learner $\hat{h}^m$ can be seen has an estimation of $-\nabla L(\hat{f}^m)$ where $L(f) = \frac{1}{n} \sum_{i=1}^{n} \left(Y_i - f(X_i)\right)^2$

$+\hat{h}^m(X_i)$ to the strong learner is a king of gradient step

Synthesis

Ensemble algorithms

General approach: create a model combining several base learners

Bagging: base learners are trained on subsets of the data (bootstrap sample)

- parallel approach
- efficient to reduce overfitting

Random forest: bagging + sampling on the variables at each split

- is generally better than bagging thank to the double sampling

Boosting: each model seeks to correct the weaknesses of the previous one

- iterative approach
- efficient to reduce bias

What to choose? It depends on how sensitive you are to bias or overfitting

Online approaches

Online Random Forest / Online Boosting

First idea: retrain all the model at each time step by eventually weighting the observations

Some concerns:

- Models are complex which need lots of data to be trained so ω_t can not go to fast to 0
- Models are trained to be good on all the data points (for each ω_t is high enough)
  - → Retraining probably won't really change the model…
  - → Need of model which reacts rapidly and locally
- Costly in terms of computing time and memory

Other ideas?

Online Random Forest / Online Boosting

Weighting the base learners:

- Keep the base learners and weight them

$$\text{Forest} = \frac{1}{B}\sum_{b=1}^{B} \text{Tree}^b \;\rightarrow\; \text{Forest}_t = \frac{1}{B}\sum_{b=1}^{B} \omega_t^b \,\text{Tree}^b$$

- Compute at each time step the weights ω_t^b using weighted linear regression

Adding a final base learner:

- Keep the strong learner and add a final base learner train on the weighted residuals

$$\text{Boosting} = \hat{f}^m \quad \rightarrow \quad \text{Boosting}_t = \hat{f}^m + \hat{h}_t^{M+1} \quad \text{where } \hat{h}_t^{M+1} \in \arg\min_h \sum_{s=1}^{t} \omega_t \left(\varepsilon_s^M - h(X_t)\right)^2$$

where $\varepsilon_s^M = Y_s - \hat{f}^M(X_i)$

Remark: explicatives variables may be different since we add a completely new model

That’s all folks!