Apprentissage supervisé
Arbres de classification.

Marie Chavent

Université de Bordeaux

Introduction

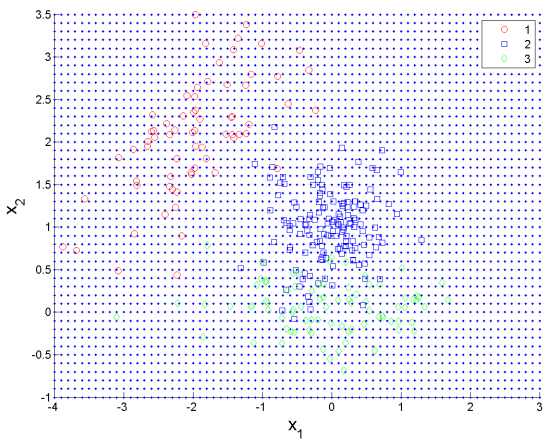
Deux approches possibles pour constuire une règle de classification g.

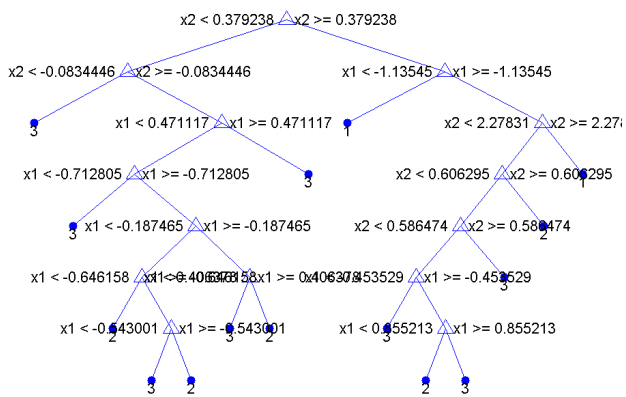
- Approche basée sur un modèle.
  - Apprentissage de la $Loi(Y|X)$ puis déduction de g
  - Exemples : analyse discriminante linéaire, bayésien naïf, régression logistique, etc.
- Approche de type prototype.
  - Apprentissage direct de la règle classification g
  - Exemples : k-plus proches voisins, arbres de classification, forêts aléatoires, etc.
- Dans ce chapitre : arbres de classification
  - CART : Classification and regression trees. L. Breiman, J. H. Friedman, R.A. Olshen, and C. J. Stone, Chapman & Hall, 1984.
  - On plus généralement d'arbres de décision (classification et régression).

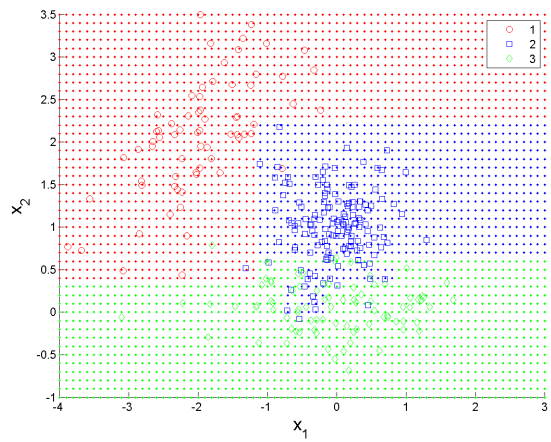
La méthode CART en classification supervisée

- Variables d'entrées quantitatives ou qualitatives $X = (X^1, \ldots, X^p) \in \mathcal{X}$.
- Variable de sortie Y qualitative à K modalités définissant les K classes à prédire.
- La règle de classification $g : \mathcal{X} \to \{1, \ldots, K\}$ est un un arbre de classification construit à partir des données d'apprentissage (X_i, Y_i), $i = 1, \ldots, n$.

Exemple : arbre de classification obtenu sur un jeu de données synthétiques







En résumé

1. La méthode CART construit un arbre de classification T à partir des données d'apprentissage. Chaque feuille (noeud terminal) $t \in T$ correspond à une cellule $\mathcal{X}_t \subset \mathcal{X}$.

2. On estime alors $\mathbb{P}(Y = k|X = x)$ pour $x \in \mathcal{X}_t$ par la proportion de données d'apprentissage appartenant à la classe k dans dans la feuille t.

3. La méthode prédit une nouvelle observation x en la faisant descendre dans l'arbre jusqu'à une feuille t. La prédiction est alors la classe la plus probable à posteriori dans t (pour une matrice de coût 0-1) ou encore la classe la moins risquée à posteriori dans t pour une matrice de coût quelconque.

Construction de l'arbre de classification

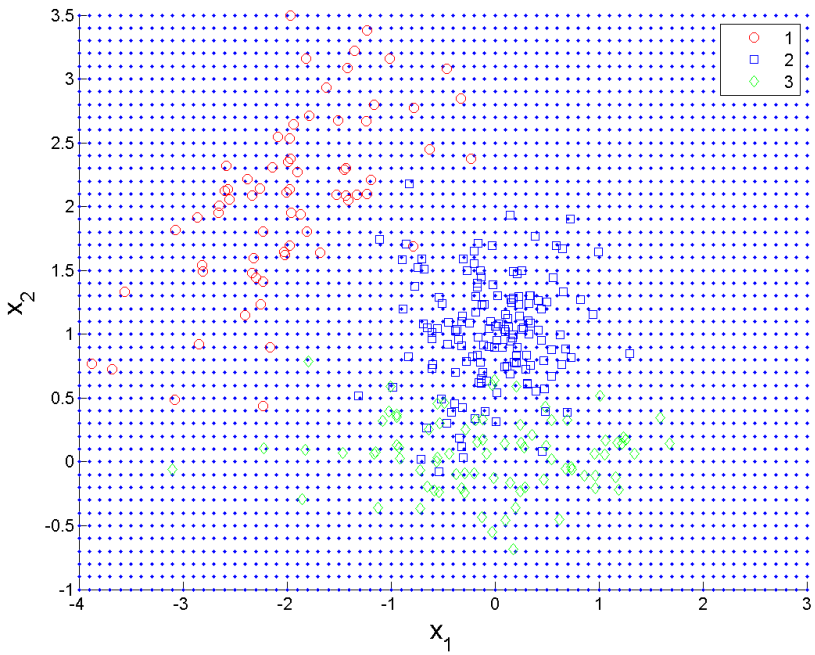
Quel est le principe général ?

1. On part d'un noeud racine contenant toutes les données d'apprentissage.
2. A chaque étape un noeud est divisé en deux sous-noeuds (noeud fils gauche et noeud fils droit) à partir d'une question binaire. La question binaire optimise un critère d'homogénéité des noeuds vis à vis de la variable à expliquer Y.
3. Si aucun noeud ne peut plus être divisé, l'algorithme s'arrête. Un noeud terminal (qui ne peut plus être divisé) est appellé une feuille.
4. Une étiquette est associée à chaque feuille. L'étiquette d'une feuille est la classe la plus fréquente (dans le cas d'une matrice de coût 0-1) ou la classe la moins risquée à posteriori dans la feuille (pour une matrice de coût quelconque).

Comment est divisé un noeud ?

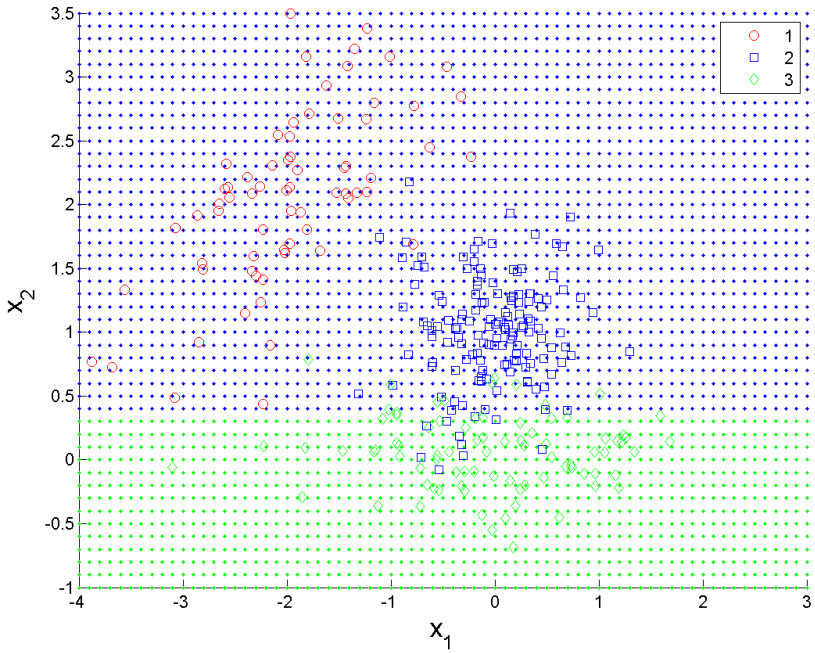
On veut que la question binaire construise des noeuds (ou encore des cellules) homogènes vis à vis de la variable à expliquer Y.

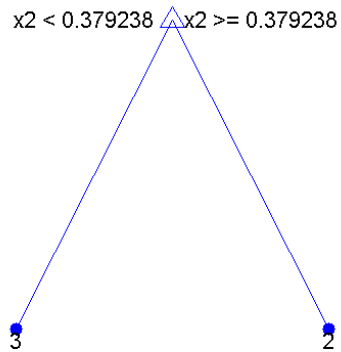
Exemple : proposer une bonne et une mauvaise question binaire pour diviser le noeud racine. Pourquoi l'étiquette de ce noeud racine vaut 2 ?



2

La question binaire trouvée par la méthode CART est la suivante :





CART trouve la meilleure division c'est à dire la division qui maximise la réduction de l'impureté.

L'impureté (l'hétérogénéité) d'un noeud vis à vis de Y se mesure à partir d'une fonction ϕ qui s'applique à la distribution de Y dans le noeud.

La fonction d'impureté ϕ est définie de manière générale sur l'ensemble des K-uplets $(p_1, \ldots, p_K)$ satisfaisants $p_k \geq 0$ pour $k = 1 \ldots, K$ et $\sum_{k=1}^{K} p_K = 1$ avec :

- ϕ admet un unique maximum en $(\frac{1}{K}, \ldots, \frac{1}{K})$
- ϕ est minimum aux points $(1, 0, \ldots, 0)$, $(0, 1, \ldots, 0)$...
- ϕ est une fonction symétrique de $p_1, \ldots, p_K$ c'est à dire que ϕ est constante pour toute permutation de p_k.

On définit alors l'impureté d'un noeud t par :

$$i(t) = \phi(p(1|t), \ldots, p(K|t))$$

où $p(k|t)$ est une notation pour $\mathbb{P}(Y = k|X = x)$ pour $x \in \mathcal{X}_t$.

On estimera ces probabilités sur les données d'apprentissage par la proportion de la classe k dans le noeud t :

$$\hat{p}(k|t) = \frac{n_{t,k}}{n_t}$$

Les deux mesures d'impureté les plus courantes sont :

- l'indice de Gini :

$$i(t) = \sum_{k=1}^{K} p(k|t)(1 - p(k|t)) = 1 - \sum_{k=1}^{K} p(k|t)^2$$

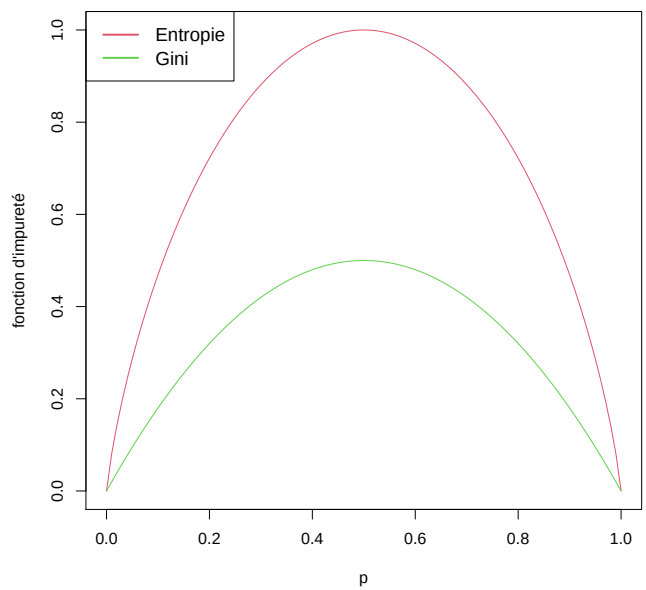
- l'entropie (avec la convention $0 \log(0) = 0$) :

$$i(t) = -\sum_{k=1}^{K} p(k|t) \log_2(p(k|t))$$

L'impureté d'un noeud t est toujours positive ou nulle et :

- nulle si toutes les observations du noeud appartiennent à la même classe de Y. On dira que le noeud est pur.
- maximale lorsque les classes de Y sont équiprobables dans le noeud. On dira que le noeud est impur.

Par exemple, si la variable Y est binaire, en notant $p = p(1|t)$, on a le graphique ci-dessous :



La réduction de l'impureté induite par la division (t_L, t_R) du noeud t est alors définie par :

$$\Delta(t_L, t_R) = i(t) - p_L i(t_L) - p_R i(t_R)$$

où p_L (resp. p_R) est la probabilité qu'une donnée appartienne à la cellule t_L (resp. t_R) sachant qu'elle se trouvait dans la cellule t.

On estimera ces probabilités sur les données par :

$$\hat{p}_L = \frac{n_L}{n_t}, \quad \hat{p}_R = \frac{n_R}{n_t}$$

où n_t est le nombre de données dans le noeud t, n_L dans le noeud t_L et n_R dans le noeud t_R.

Une bonne division occasionera une forte réduction de l'impureté.

Comment est choisie la meilleure question binaire ?

La meilleure question binaire est celle qui maximise la réduction de l'impureté.

- Si $X^j \in \mathbb{R}$ est quantitative, la question binaire sera du type

$$X^j \leq c \, ?$$

  Il existe donc une infinité de valeurs de coupures c possibles mais elles induisent au maximum $n_t - 1$ divisions différentes.

- Si $X^j \in \{1, \ldots, M\}$ est qualitative, la question binaire sera du type

$$X^j \subset A \, ?$$

  où $A \subset \{1, \ldots, M\}$. Il existe $2^{M-1} - 1$ questions binaires et donc au maximum $2^{M-1} - 1$ divisions différentes.

$\Rightarrow$ Il un nombre fini de questions binaires à évaluer.

Attention : lorsque le nombre de modalité M est grand, il y a un problème de complexité et de temps de calcul.

Quand sont arrêtées les divisions ?

Le critère d'arrêt pour obtenir l'arbre le longueur maximale peut-être :

- ne pas découper un noeud pur,
- ne pas découper un noeud qui contient moins de n_{min} données avec souvent n_{min} compris entre 1 et 5.

Nous verrons que cet arbre est généralement élagué.

Attention :

Afin d'éviter cette étape d'élaguage, certaines implémentations proposent d'arrêter les divisions en fonction d'un critère mesurant la pertinence d'une division (voir plus loin la définition du `complexity parameter`).

Comment associer une étiquette à un noeud ?

L'étiquette d'un noeud est la classe prédite dans ce noeud. On notera $\tau(t)$ l'étiquette d'un noeud terminal t.

- $\tau(t)$ est la classe la plus probable à posteriori pour une fonction de coût 0-1 :

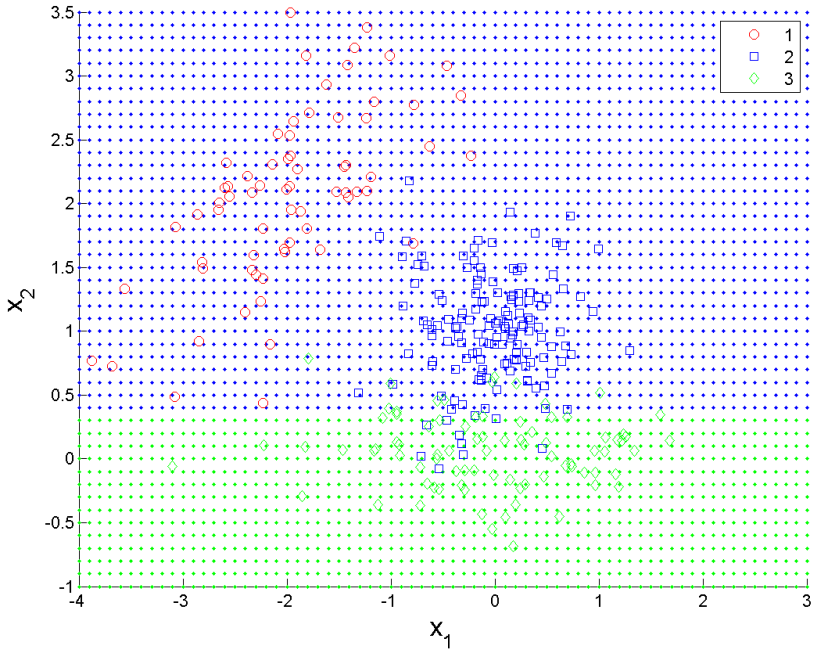
$$\tau(t) = \underset{\ell \in \{1,\dots,K\}}{\arg\max}\ p(k|t)$$

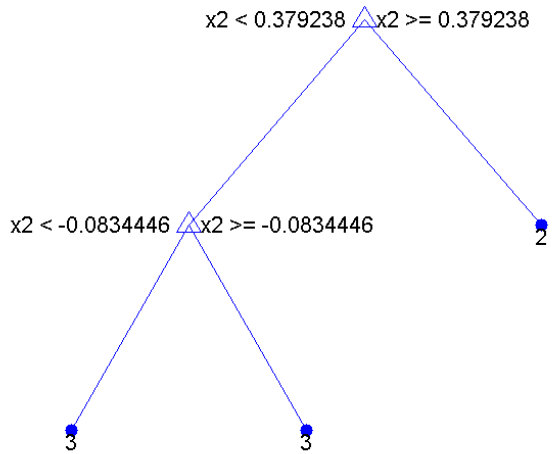
  $\tau(t)$ est alors simplement classe majoritaire.

- $\tau(t)$ est la classe la moins risquée à posteriori pour une fonction de coût quelconque :

$$\tau(t) = \underset{\ell \in \{1,\dots,K\}}{\arg\min} \sum_{k=1}^{K} C_{k\ell}\ p(k|t)$$

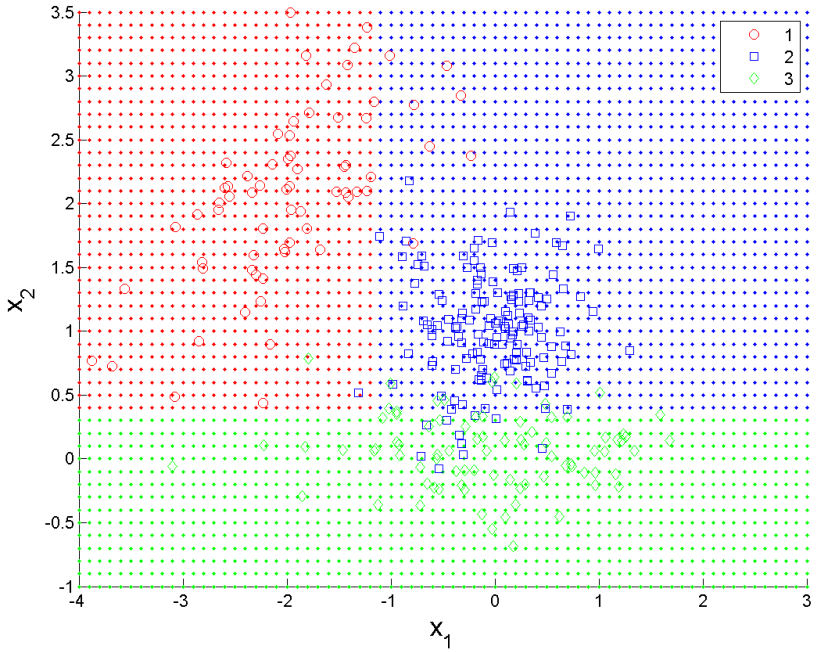
Exemple : deuxième division

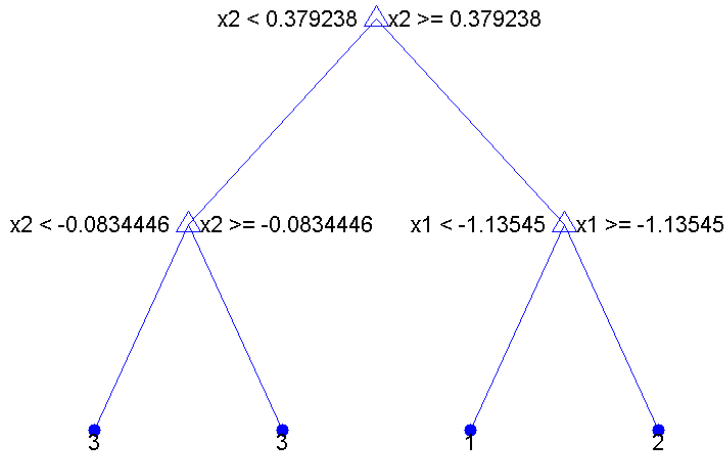




- Tracer un droite verticale ou horizontale sur le graphique de gauche pour visualiser cette seconde division.
- Quelle est le noeud pur à l'issue de cette division ?
- Ce noeud peux-il être redivisé ?

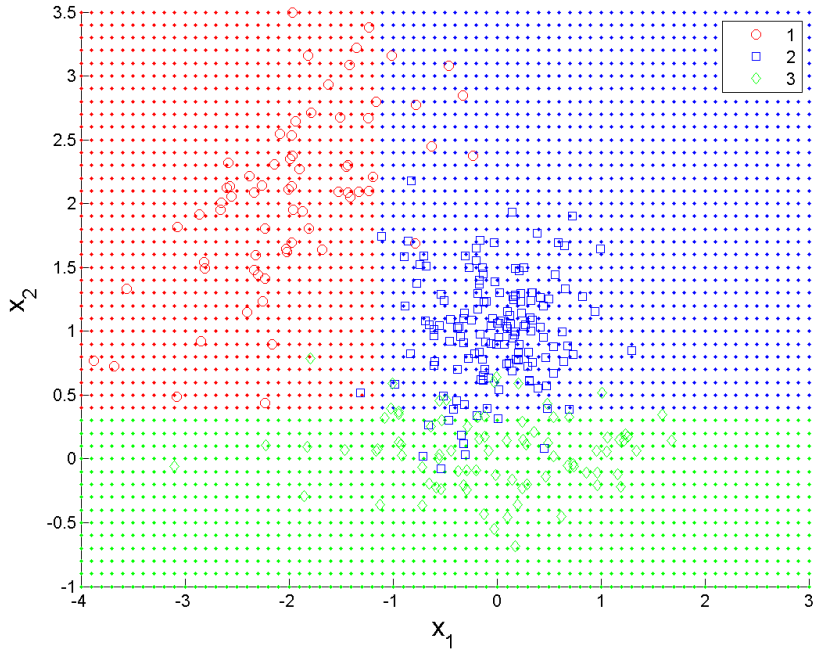
Exemple : troisième division

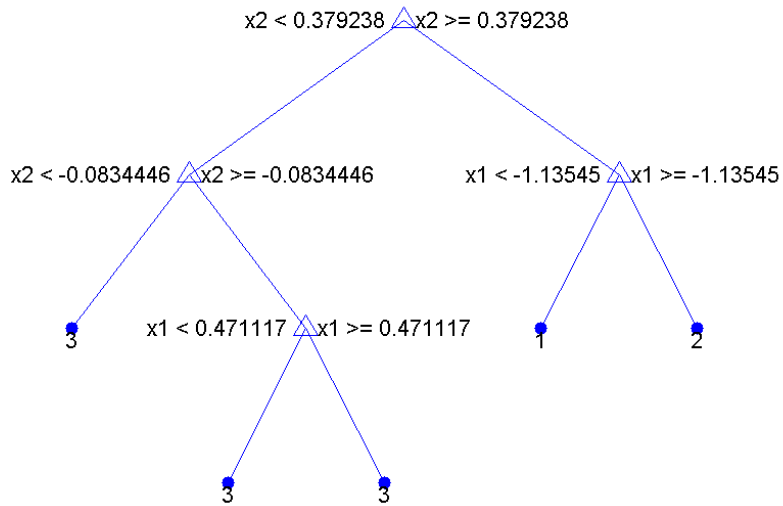




- Tracer sur le graphique de gauche les cellules (zones de $\mathbb{R}^2$ ici) associées aux 4 noeuds terminaux.
- Quels noeuds peuvent être redivisés ?

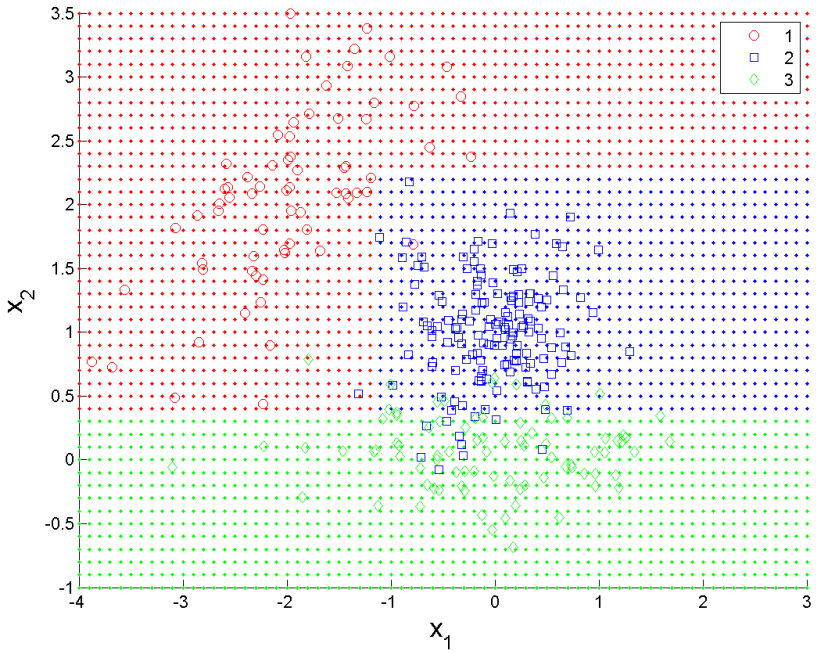
Exemple : quatrième division

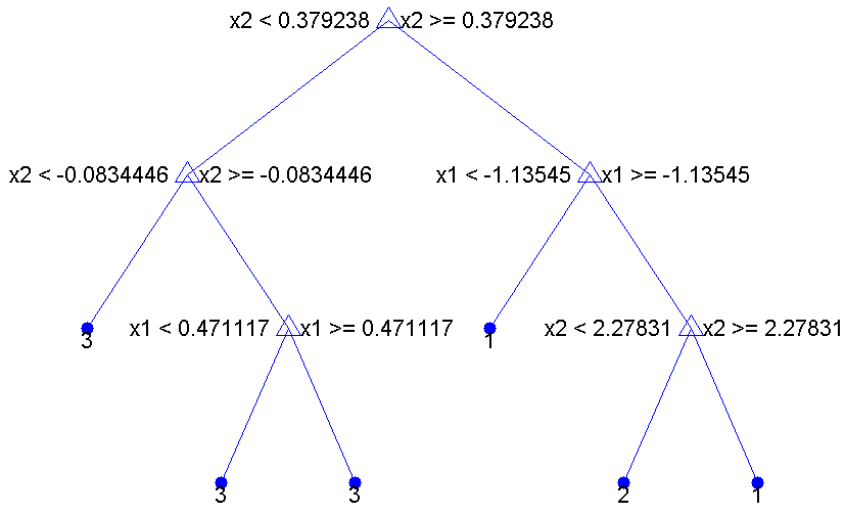




- Tracer sur le graphique de gauche les cellules (zones de $\mathbb{R}^2$ ici) associées aux 5 noeuds terminaux.
- Quelle classe est prédite par cet arbre pour une nouvelle données $\mathbf{x} = (0, 3)$?
- Quelle noeuds peuvent être redivisés ?

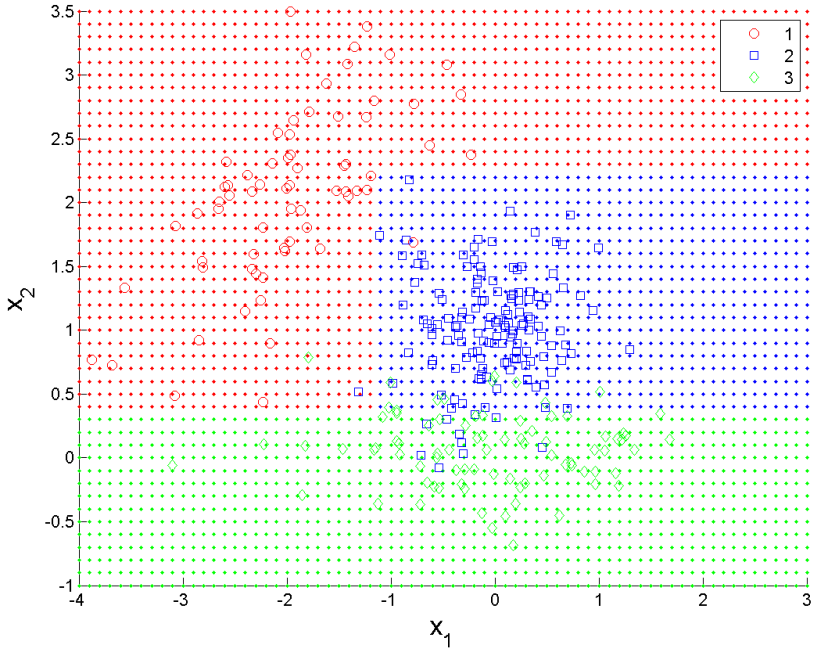
Exemple : cinquième division

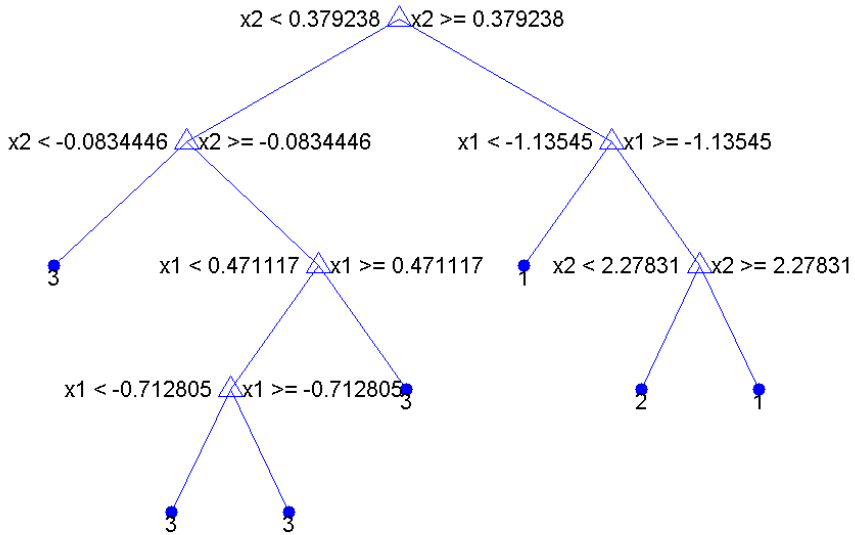




- Tracer sur le graphique de gauche les cellules (zones de $\mathbb{R}^2$ ici) associées aux 6 noeuds terminaux.
- Quelle classe est maintenant prédite par cet arbre pour la nouvelle données $\mathbf{x} = (0, 3)$?
- Quels noeuds peuvent être redivisés ?

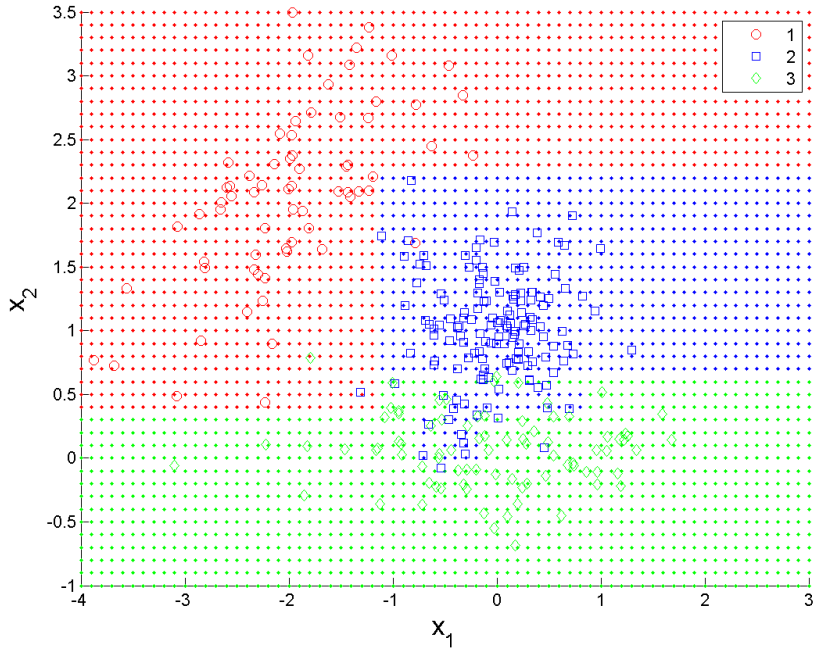
Exemple : sixième division

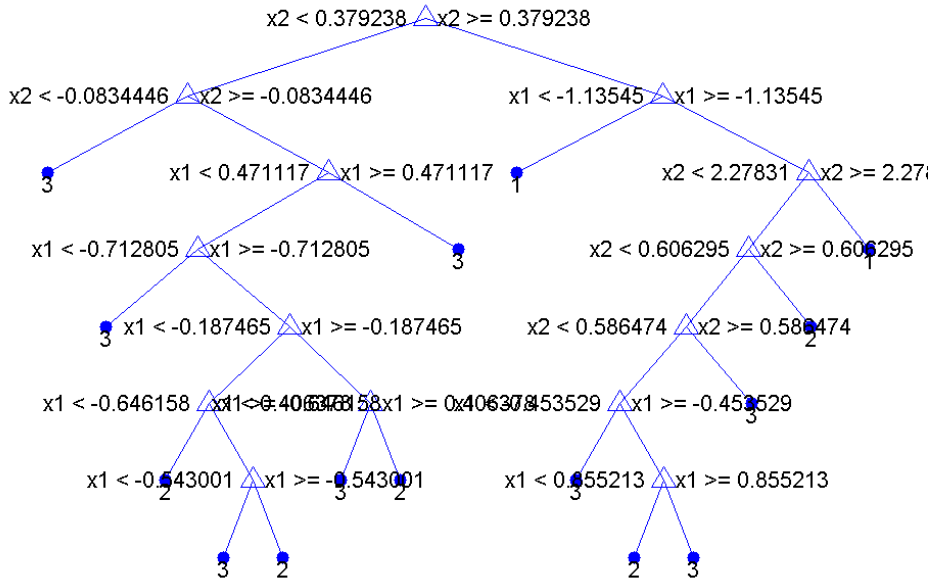




- Tracer sur le graphique de gauche les cellules (zones de $\mathbb{R}^2$ ici) associées aux 7 noeuds terminaux.
- Quels noeuds peuvent être redivisés ?
- Quand arrêter les divisions ?

Exemple : quatorzième division





- D'après vous s'agit-il de l'arbre de longeur maximale ?
- L'arbre de longueur maximale serait-il bon pour prédire de nouvelles observations ?

Evaluer la performance d'un arbre

Le risque théorique d'une feuille t d'étiquette $\tau(t)$ est défini par :

$$r(t) = \sum_{k=1}^{K} C_{k\tau(t)}\ p(k|t)$$

où $C_{k\tau(t)}$ est le coût de mauvaise classification d'une donnée de la classe k dans la classe $\tau(t)$.

Le risque théorique de l'arbre T est alors définit par :

$$R(T) = \sum_{t \in \tilde{T}} p(t) r(t)$$

où $\tilde{T}$ est l'ensemble des feuilles de l'arbre T et $p(t)$ est la probabilité d'appartenir à la feuille t.

On considère maintenant un échantillon d'observations $(X_1, Y_1), \ldots, (X_n, Y_n)$.

Le risque empirique d'une feuille est définie par :

$$\begin{aligned}\hat{r}(t) &= \sum_{k=1}^{K} C_{k\tau(t)} \frac{n_{t,k}}{n_t} \\ &= \frac{1}{n_t} \sum_{x \in t} C_{\tau(x)\tau(t)}\end{aligned}$$

où $\tau(x)$ est la vraie classe de l'observation x.

Dans le cas d'une matrice de coût 0-1, on a :

$$\hat{r}(t) = \frac{1}{n_t} \sum_{x \in t} \mathbb{1}_{\tau(t) \neq \tau(x)}$$

est le taux d'erreur dans la feuille.

Le risque empirique d'un arbre est défini par :

$$\begin{aligned}\hat{R}(T) &= \sum_{t \in \tilde{T}} \frac{n_t}{n} \frac{1}{n_t} \sum_{x \in t} C_{\tau(x)\tau(t)} \\ &= \frac{1}{n} \sum_{t \in \tilde{T}} \sum_{x \in t} C_{\tau(x)\tau(t)}\end{aligned}$$

Dans le cas d'une matrice de coût 0-1, on a :

$$\hat{R}(T) = \frac{1}{n} \sum_{t \in \tilde{T}} \sum_{x \in t} \mathbb{1}_{\tau(t) \neq \tau(x)}$$

est le taux d'erreur de l'arbre.

Remarques :

- $n_t \hat{r}(t)$ est le coût de mauvais classement d'une feuille t.
- $n\hat{R}(T)$ est le coût de mauvais classement d'un arbre T.

Pour une matrice de coût 0-1 :

- $n_t \hat{r}(t)$ est le nombre de mals classés d'une feuille t.
- $n\hat{R}(T)$ est le nombre de mals classés d'un l'arbre T.

En pratique :

Ces mesures de performance (coût moyen de mauvais classement, taux d'erreur....) sont estimées sur des données test différentes des données d'apprentissage utilisées pour construire l'arbre.

Performance d'une division.

Chaque division d'un noeud diminue le coût de mauvais classement (ou nombre de mal classés) de l'arbre. Mais trop de divisions conduisent au surapprentissage. Il faudra donc trouver un compromis en le coût de l'arbre et sa complexité (son nombre de feuilles).

Le coût de mauvais classement dans le noeud t est :

$$LOSS(t) = n_t \hat{r}(t) = \sum_{x \in t} C_{\tau(x)\tau(t)}$$

Diviser le noeud t entraine une diminution du coût de mauvais classement (de l'arbre). Cette amélioration est aussi appelée paramètre de complexité de t :

$$cp(t) = LOSS(t) - LOSS(t_L) - LOSS(t_R)$$

On pourra décider de ne pas diviser un noeud si cette amélioration est trop faible (sachant que chaque division augmente la complexité). Cela permet de gagner en temps de calcul par rapport à l'approche qui consiste à construire l'arbre de longeur maximale puis à l'élaguager pour trouver le meilleur compromis coût-complexité.

Remarque.

Il existe différentes implémentations du paramètre de complexité. Par exemple dans la fonction `rpart` du logiciel R :

$$cp(t) = \frac{LOSS(t) - LOSS(t_L) - LOSS(t_R)}{LOSS(t_1)}$$

avec t_1 le noeud racine.

Dans cette fonction, un noeud t ayant une valeur $cp(t)$ inférieure à 0.01 (par défaut) n'est pas divisé.

Elaguage

Cette procédure consiste à :

1. Construire l'arbre de longeur maximale.
2. En déduire une suite de sous-arbres élagués qui minimisent un critère coût-complexité.
3. Choisir le meilleur en estimant leurs coûts de mauvais classement par validation croisée.

Construction des sous-arbres élagués optimaux.

Le critère de coût-complexité $C_\alpha(T)$ peut être défini par :

$$C_\alpha(T) = n\hat{R}(T) + \alpha|\tilde{T}|,$$

où

- $|\tilde{T}|$ est le nombre de noeuds terminaux de T,
- $n\hat{R}(T)$ est le coût de mauvais classement ou encore le nombre de mal classés pour une fonction de coût 0-1.
- α est appellé le paramètre de complexité.

Pour une valeur fixée du paramètre α, il faudra minimiser $C_\alpha(T)$. Lorsque le paramètre de complexité va augmenter, il deviendra plus intéressant à partir de certaines valeurs de α d'élaguer une branche de l'arbre. On peut montrer qu'il existe un nombre fini de sous-arbres minimisant $C_\alpha(T)$ et ces sous-arbres forment une suite de sous-arbres élagués optimaux de T_{max}.

1. Pour $\alpha = 0$, c'est l'arbre de longueur maximale qui minimise C_α. On notera T_L l'arbre maximal à L feuilles et on note $\alpha_L = 0$.

2. Lorsque α augmente, l'arbre T_L minimise C_α tant que

$$C_\alpha(T_L) < C_\alpha(T_{L-1}),$$

où T_{L-1} est un arbre élagué à $L-1$ feuilles. Cette condition s'écrit alors :

$$n\hat{R}(T_L) + \alpha L < n\hat{R}(T_{L-1}) + \alpha(L-1),$$

ou encore

$$\alpha < \underbrace{n\hat{R}(T_{L-1}) - n\hat{R}(T_L)}_{\alpha_{L-1}}.$$

L'arbre T_{L-1} est obtenu en élaguant le noeud t de T_L qui augmente le moins possible le coût de mauvais classement ou encore qui minimise

$$LOSS(t) - LOSS(t_L) - LOSS(t_R) = cp(t) = \alpha_{L-1}.$$

Pour toute valeur $\alpha \in [0, \alpha_{L-1}[$ c'est donc T_L qui minimise $C_\alpha(T)$.

3. Le procédé est itéré pour obtenir la séquence d'arbres emboités suivante :

$$T_{max} = T_L \supset T_{L-1} \supset \ldots \supset T_1$$

où T_1 est l'arbre réduit au noeud racine.

Les arbres de cette séquence minimisent $C_\alpha(T)$ sur les plages de valeurs de α suivantes :

$$\begin{aligned}
&[0, \alpha_{L-1}[\rightarrow T_L \\
&[\alpha_{L-1}, \alpha_{L-2}[\rightarrow T_{L-1} \\
&\vdots \\
&[\alpha_2, \alpha_1[\rightarrow T_2 \\
&[\alpha_1, \infty[\rightarrow T_1
\end{aligned}$$

Les paramètres de complexités $\alpha_{L-1} \dots \alpha_j \dots \alpha_1$ mesurent l'augmentation du coût de mauvais classement obtenu en élaguant le noeud t de T_{j+1} pour obtenir T_j :

$$\alpha_j = n\hat{R}(T_j) - n\hat{R}(T_{j+1}) = cp(t)$$

En pratique, il suffit de trier les noeuds de l'arbre maximal par ordre croissant de leur paramètre de complexité, puis de supprimer successivement les divisions associées à ces noeuds pour obtenir la séquence des sous-arbres optimaux.

Remarque.

Dans la fonction `rpart` du logiciel R :

$$cp(t) = \frac{LOSS(t) - LOSS(t_L) - LOSS(t_R)}{LOSS(t_1)}$$

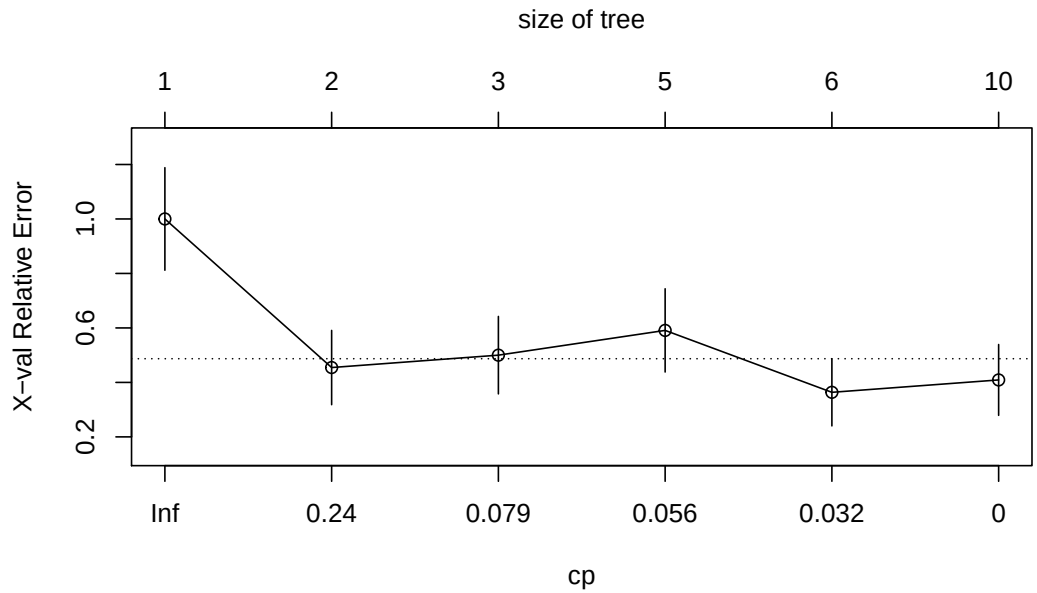
avec t_1 le noeud racine.

Le critère coût-complexité minimisé est alors :

$$C_\alpha(T) = R(T) + \alpha|\tilde{T}|R(T_1).$$

Choix du meilleur sous-arbre

Le meilleur sous-arbre de cette séquence est celui qui minimise le coût de mauvais classement ou encore l'erreur de classement dans le cas d'une matrice de coût 0-1. La difficulté réside dans l'estimation de ces erreurs. La fonction `rpart` implémente une méthode d'estimation par validation croisée (assez compliquée) de ces erreurs.



Ici c'est le sous-arbre à 6 feuilles qui donne l'erreur (relative) de validation croisée la plus petite.

Pour estimer les erreurs des sous-arbres par validation croisée, la fonction `rpart` procède de la manière suivante :

1. De nouveaux paramètres de complexité sont calculés :

$$
\begin{array}{ll}
\beta_L = 0 & \rightarrow T_L \\
\beta_{L-1} = \sqrt{\alpha_{L-1}\alpha_{L-2}} & \rightarrow T_{L-1} \\
\vdots & \\
\beta_2 = \sqrt{\alpha_2\alpha_1} & \rightarrow T_2 \\
\beta_1 = \infty & \rightarrow T_1
\end{array}
$$

   - Chaque paramètre β_j est "typique" de l'intervalle $[\alpha_j, \alpha_{j-1}[$.
   - Le sous-arbre T_j minimise le critère coût-complexité C_{β_j}.

2. Les données sont divisées en I groupes $G_1, \ldots, G_I$ de même taille.

   Pour chaque groupe G_i :

   - L'arbre maximum T_{max}^{-i} est construit à partir des données privées du groupe G_i.
   - Le suite de sous-arbres élagués T_k^{-i} de T_{max}^{-i} est ensuite construite ainsi que les intervalles de paramètres de complexité associés $[\alpha_k^{-i}, \alpha_{k-1}^{-i}[$.
   - Pour chaque β_j :
     - 0n retient le sous-arbre T_k^{-i} associé à l'intervalle $[\alpha_k^{-i}, \alpha_{k-1}^{-i}[$ qui contient β_j.
     - On prédit avec ce sous-arbre la classe des observations du groupe G_i.
     - On calcule le coût de mauvais classement ou encore l'erreur de classement du groupe G_i. Dans `rpart`, c'est l'erreur relative (divisée par l'erreur de prédiction du noeud racine) qui est calculée.

En sortie de cette double boucle (sur les G_i et sur les β_j) on obtient :

	$\beta_1 = \infty \ldots$	β_1	$\ldots \beta_L$
G_1			
$\vdots$		$\vdots$	
G_i	$\ldots$	err_{ij}	$\ldots$
$\vdots$		$\vdots$	
G_I			

où err_{ij} est l'erreur test (calculée sur G_i) du sous-arbre minimisant le critère coût complexité C_{β_j}.

C'est donc une estimation de l'erreur de classement du sous-arbre élagué T_j.

3. Les moyennes et les l'écart-type des erreurs test sont calculées pour chaque β_j.

	$\beta_1 = \infty \ldots$	β_1	$\ldots \beta_L$
G_1			
$\vdots$		$\vdots$	
G_i	$\ldots$	err_{ij}	$\ldots$
$\vdots$		$\vdots$	
G_I			
moyenne		$\overline{err}_j$	
écart-type		std_j	

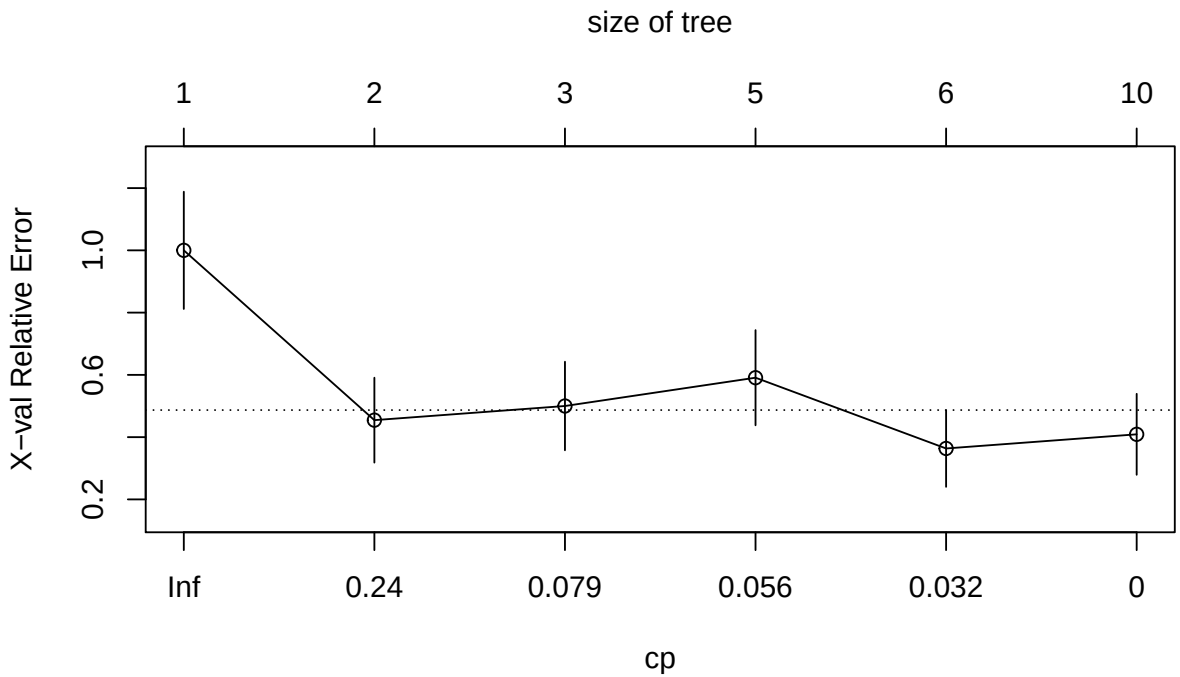
4. Le sous-arbre T_j correspondant au coefficient β_j pour lequel l'erreur moyenne $\overline{err}_j$ est minimum sera retenu. Il s'agira du sous-arbre élagué final.

Dans `rpart`, les erreurs sont relatives (à l'erreur dans le noeud racine). Je pense donc que ces erreurs sont calculées comme suit :

	$\beta_1 = \infty \ldots$	β_1	$\ldots \beta_L$
G_1			
$\vdots$		$\vdots$	
G_i	$\ldots$	$err_{ij}/\overline{err}_1$	$\ldots$
$\vdots$		$\vdots$	
G_I			
moyenne	1	$\overline{err}_j/\overline{err}_1$	
écart-type	$std_1/\overline{err}_1$	$std_j/\overline{err}_1$	

Les moyennes et les écart-types de ce tableau sont appellée `xerror` et `xstd` dans les sorties de la fonction `rpart`.

Reprenons le graphique ci-dessous représentant les erreurs relatives de validation croisées en ordonnée ($\texttt{xerror} \pm \texttt{xstd}$) en fonction des paramètres de complexité β_j en abscisse (cp) :



La règle du 1-SE retient le β_j le plus grand (le plus à gauche ici) dont l'erreur relative est en dessous de la ligne pointillée. Avec cette règle, on choisit le sous-arbre à 2 feuilles et non plus 6 feuilles.