



Université
de Rennes

istic
Informatique
Electronique



UMR IRISA

Chapitre 2. k plus proches voisins (k -ppv)

Claude Petit, Insee et université de Rennes - claude.petit@univ-rennes.fr

Juin 2026

1. Rappels sur le cadre probabiliste
2. Principe des algorithmes à base de partition
2. Algorithme des k plus proches voisins
3. Arbres de décision et algorithme CART

1. Rappels sur le cadre probabiliste

Apprentissage supervisé -1-

- $\mathcal{D}_n = \{Z_1, \dots, Z_n\}$ n -échantillon de v.a.i.i.d. $Z_i = (X_i, Y_i)$.
- X_i **observations** issues d'une v.a. X : données, variables explicatives.
- Y_i issues d'une v.a. Y , catégories des X_i : **étiquettes** ou labels.
- $X \in \mathbb{X}$, $Y \in \mathbb{Y}$.
- \mathbb{P} proba sur $\mathcal{E} = \mathbb{X} \times \mathbb{Y}$: loi (inconnue) de (X, Y) et des (X_i, Y_i) .

Objectif de l'apprentissage supervisé : déterminer Y sachant X , à partir des seules observations de Z_1, \dots, Z_n .

- $h \in \mathcal{F} = \mathcal{F}(\mathbb{X}, \mathbb{Y})$ **fonction de prédiction** : $h(x) = y$.
- $l : \mathbb{Y}^2 \rightarrow \mathbb{R}_+$ **fonction de perte** pour mesurer la qualité de g .
- $R(h) = R_{\mathbb{P}}(h) = \mathbb{E}[l(Y, h(X))]$ **risque de h** : valeur moyenne de la perte sur toutes les réalisations possibles.

Algorithme d'apprentissage : fonction

$$h : \bigcup_{n=1}^{+\infty} (\mathbb{X} \times \mathbb{Y})^n \longrightarrow \mathcal{F}(\mathbb{X}, \mathbb{Y}) \quad (1)$$

qui à $\mathcal{D}_n \longrightarrow h_n$. Estimateur de la meilleure fonction de prédiction.

$h_n(x) = h_n(x, \mathcal{D}_n)$ dépend de l'échantillon !

$$g_h = h_n(\cdot, \mathcal{D}_n) \in \mathcal{F}(\mathbb{X}, \mathbb{Y}) \quad (2)$$

\mathcal{D}_n fonction des $(X_i, Y_i) \Rightarrow$ aléatoire, $h_n(x) = h_n(x, \mathcal{D}_n)$ également aléatoire, tout comme :

$$R_{\mathbb{P}}(h_n) = \mathbb{E}[l(Y, h_n(X))] = \int_{\mathbb{X} \times \mathbb{Y}} l(y, h_n(x)) \mathbb{P}(dx, dy) \quad (3)$$

Risque empirique

$R_{\mathbb{P}}[l(Y, h(X))]$ inconnu. L'algorithme d'apprentissage doit trouver h de risque le plus petit possible. On estime R par l'estimateur plug-in :

$$R_n(h) = \frac{1}{n} \sum_{i=1}^n l(Y_i, h(X_i)) \quad (4)$$

R_n défini pour une famille de fonctions de prédiction (sous-ensemble \mathcal{G} de \mathcal{F}). La loi forte des grands nombres (LFGN) et le théorème de la limite centrale (TLC) donnent :

$$\lim_{n \rightarrow +\infty} R_n(h) = R(h) \text{ p.s.} \quad (5)$$

$$\sqrt{n}(R_n(h) - R(h)) \rightsquigarrow \mathcal{N}(0, \sigma^2) \quad (6)$$

si $\sigma^2 = \mathbb{V}(l(Y, h(X))) < \infty$

$\Rightarrow R_n(h)$ bonne approximation de $R(h)$ si n suffisamment grand.

Minimisation du risque empirique et capacité du modèle

\mathcal{G} étant fixé, on choisit comme fonction de prédiction un minimiseur $\hat{h}_{n,\mathcal{G}}$ du risque empirique défini par :

$$\hat{h}_{n,\mathcal{G}} = \hat{h}_n(\mathcal{D}_n, \mathcal{G}) = \arg \min_{h \in \mathcal{G}} R_n(h) \quad (7)$$

où $\mathcal{G} \subset \mathcal{F}$ est un sous-ensemble de toutes les fonctions de \mathbb{X} dans \mathbb{Y} .
C'est la classe de fonctions à laquelle on se restreint pour déterminer h_n .

$\mathcal{G} = \mathcal{F} \Rightarrow$ mauvaise idée :

- Souvent infinité de fonctions minimisantes.
- Très loin d'être universellement consistant.
- **Sur-apprentissage** assuré.

Taille de $\mathcal{G} =$ **capacité ou complexité du modèle**.

On doit prendre \mathcal{G} assez grand pour bien approcher toute fonction, mais pas trop pour éviter le sur-apprentissage.

Exemples de dictionnaires

- **Régression linéaire** :

$$\mathcal{G} = \{h : x \mapsto h(x) = b + w^\top x\}.$$

- **Régression logistique** :

$$\mathcal{G} = \{h : x \mapsto h(x) = \sigma(b + w^\top x)\}.$$

- **Classifieur bayésien naïf binaire** :

$$\mathcal{G} = \{h : x \mapsto h(x) = \mathbb{1}_{[\mathbb{P}[Y=1|X=x] > \frac{1}{2}]}\}.$$

- **Classifieur bayésien naïf multinomial** :

$$\mathcal{G} = \{h : x \mapsto h(x) = \mathbb{1}_{[\arg \max_k \mathbb{P}[Y=y_k|X=x]]}\}.$$

- **Réseaux de neurones** : $\mathcal{G} = \{h : x \mapsto h(x)\}$ avec h tel que

$$h(x) = \bigcirc_{i=1}^L \sigma(w_i^\top x + b_i) = \sigma(w_L \sigma(\dots \sigma(w_1^\top x + b_1) \dots))$$

σ fonction logistique.

2. Principe des algorithmes à base de partition

Introduction -1-

- \mathcal{F} ensemble des fonctions de \mathbb{X} dans \mathbb{Y} .
- $\mathcal{G} \subset \mathcal{F}$ dictionnaire.

1 méthode d'apprentissage supervisé = 1 choix de dictionnaire \mathcal{G} .

Choix populaire : fonction constante / morceaux sur 1 partition de \mathbb{X} .

⇒ **algorithme des plus proches voisins** k -ppv ou k -NN (pour k Nearest Neighbors), **arbres de décision** et de régression avec l'**algorithme CART** (Classification and Regression Tree).

$\mathcal{A} = (A_1, \dots, A_M)$ partition de \mathbb{X} :

$$\mathbb{X} = \bigcup_{m=1}^M A_m \quad (8)$$

$\mathcal{G}(\mathcal{A}) \subset \mathcal{F}(\mathbb{X}, \mathbb{Y})$ ensemble des fonctions constantes sur chaque $A_m \in \mathcal{A}$.

Introduction -2-

La partition peut être aléatoire, déterminée par \mathcal{D}_n .

Soit $\hat{g}_n = \hat{g}_n(\cdot, \mathcal{A})$ le minimiseur du risque empirique sur le dictionnaire $\mathcal{G}(\mathcal{A})$ et N_m le nombre d'exemples appartenant à A_m :

$$\hat{g}_n \in \arg \min_{g \in \mathcal{G}(\mathcal{A})} R_n(g) = \arg \min_{g \in \mathcal{G}(\mathcal{A})} \frac{1}{n} \sum_{i=1}^n l(Y_i, g(X_i)) \text{ et } N_m = \sum_{i=1}^n \mathbb{1}_{A_m}(X_i)$$

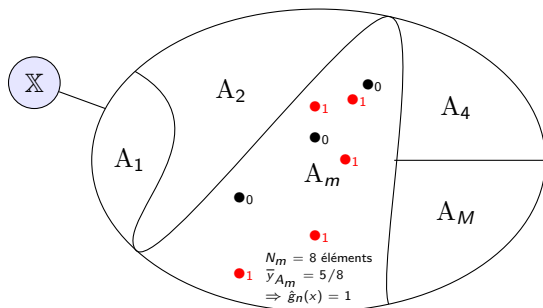
Classifieur binaire optimal

$$\forall m = 1..M, \forall x \in A_m, \hat{g}_n(x, \mathcal{A}) = \begin{cases} 1 & \text{si } \bar{Y}_{A_m} > 1/2 \\ a_m & \text{si } \bar{Y}_{A_m} = 1/2 \\ 0 & \text{si } \bar{Y}_{A_m} < 1/2 \end{cases} \quad (9)$$

Minimiseur du risque empirique pour la régression MC

$$\hat{g}_n(x, \mathcal{A}) = \sum_{m=1}^M \bar{Y}_{A_m} \mathbb{1}_{A_m}(x) = \sum_{m=1}^M \left(\frac{1}{N_m} \sum_{i=1}^n Y_i \mathbb{1}_{A_m}(X_i) \right) \mathbb{1}_{A_m}(x) \quad (10)$$

Démonstration en exercice !



Dans une cellule de la partition de l'espace des « features », l'estimateur prendra comme valeur (constante) la valeur majoritaire des points de l'échantillon d'entraînement présents dans la cellule.

Méthodes à base de partition -1-

Soit $\mathcal{A} = (A_1, \dots, A_M)$ une partition de \mathbb{X} et $\mathcal{G} = \mathcal{G}(\mathcal{A})$ classe des fonctions constantes sur chaque élément de la partition :

$$\mathcal{G} = \{g : \mathbb{X} \rightarrow \mathbb{Y} : \forall m = 1, \dots, M, g \text{ constante sur } A_m\} \quad (11)$$

Le **minimiseur du risque empirique** associé à cette partition est la fonction

$$\hat{g}_n = \hat{g}_n(\cdot, \mathcal{A}) = \operatorname{argmin}_{g \in \mathcal{G}(\mathcal{A})} \frac{1}{n} \sum_{i=1}^n l(Y_i, g(X_i)) \quad (12)$$

Notons N_m le nombre d'observations X_i de l'échantillon qui se trouvent dans l'élément A_m de la partition :

$$N_m = \sum_{i=1}^n \mathbb{1}_{A_m}(X_i) \quad (13)$$

Méthodes à base de partition -2-

et notons \bar{Y}_m la moyenne des étiquettes observées dans l'élément A_m :

$$\bar{Y}_m = \sum_{i=1}^n Y_i \mathbb{1}_{A_m}(X_i) \quad (14)$$

Alors le minimiseur du risque empirique pour la classification binaire est, si $x \in A_m$,

$$\hat{g}_n(x) = \mathbb{1}_{[\bar{Y}_m > 1/2]} \quad (15)$$

$\forall m = 1, \dots, M$. Autrement dit :

$$\hat{g}_n(x) = \sum_{m=1}^M \mathbb{1}_{[\bar{Y}_m > 1/2]} \mathbb{1}_{A_m}(x) \quad (16)$$

On généralise facilement de classifieur binaire à une **situation multiclassées** avec $\mathbb{Y} = \{1, \dots, K\}$. $\forall x \in \mathbb{X}$,

$$\hat{g}_n(x) = \arg \max_{k=1, \dots, K} \sum_{m=1}^M \left(\frac{1}{N_m} \sum_{i=1}^n \mathbb{1}_{[Y_i=k]} \mathbb{1}_{A_m}(X_i) \right) \mathbb{1}_{A_m}(x) \quad (17)$$

$$= \arg \max_{k=1, \dots, K} \sum_{m=1}^M w_{m,k} \mathbb{1}_{A_m}(x) \quad (18)$$

avec $w_{m,k} = N_m(k)/N_m$ proportion des observations dans A_m pour lesquelles $Y = k$.

2. Algorithme des k plus proches voisins

Méthode des k -ppv (k -NN)

Soit $k \leq n$. $\forall x \in \mathbb{R}^d$, $\forall i = 1, \dots, n$, $d_i(x) = \|X_i - x\|$

Soient $r_i(x)$ indice du i ème ppv de x parmi X_1, \dots, X_n :

$$r_1(x) = j \iff \begin{cases} d_j(x) = \min_{i=1, \dots, n} d_i(x) \\ d_j(x) < \min_{1 \leq i < j} d_i(x) \end{cases} \quad (19)$$

$$(20)$$

et par récurrence sur $k \geq 1$,

$$r_k(x) = j \iff \begin{cases} d_j(x) = \min_{i=1..n; i \neq r_1, \dots, r_{k-1}} d_i(x) \\ d_j(x) < \min_{1 \leq i < j; i \neq r_1, \dots, r_{k-1}} d_i(x) \end{cases} \quad (21)$$

$$(22)$$

Partition définie par les rangs

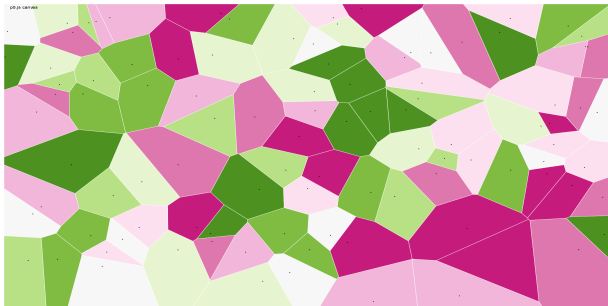
$\forall k \in \llbracket 1, \dots, n \rrbracket$, les $d_j(x)$ définissent une partition $\mathcal{A}_k = (A_{m,k})_m$ telle que, $\forall m = 1, \dots, M$:

$$A_m = \{x \in \mathbb{X} = \mathbb{R}^d : C_m = (r_1(x), \dots, r_k(x))\} \quad (23)$$

C_m combinaison de k éléments parmi n et $M = \binom{n}{k}$. Les A_m sont les parties de \mathbb{X} pour lesquelles $x \mapsto (r_1(x), \dots, r_k(x))$ est constante : la partition forme des zones caractérisées par un choix de k observations parmi n et les points de cette zone sont les plus près des k observations caractérisant la zone. Pour $x, x' \in A_m$, les k -ppv de x et x' parmi X_1, \dots, X_n sont les mêmes. Si $x \in A_m$ et $x' \in A_{m'}$ avec $m \neq m'$, les k -ppv de x sont \neq des k -ppv de x' .

Diagrammes de Voronoi -1-

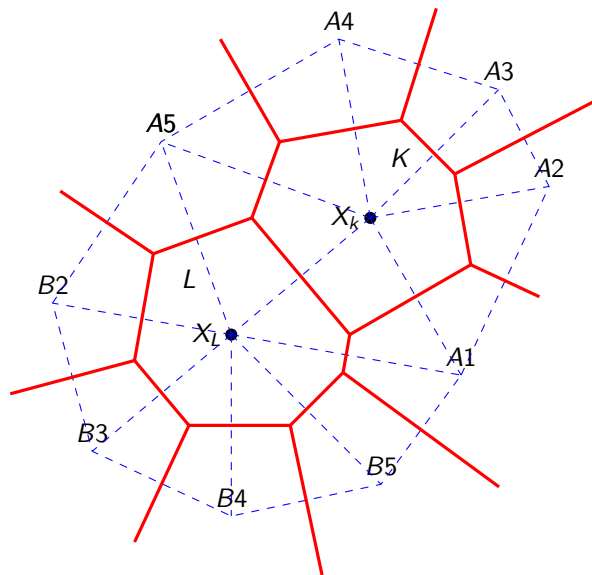
Ces partitions forment des diagrammes de Voronoi. Chaque zone est une **cellule de Voronoi**. Les figures illustrent des partitions ppv pour $k = 1$.



Cellules de Voronoi pour $k = 1$.

<https://strongriley.github.io/d3/ex/voronoi.html>

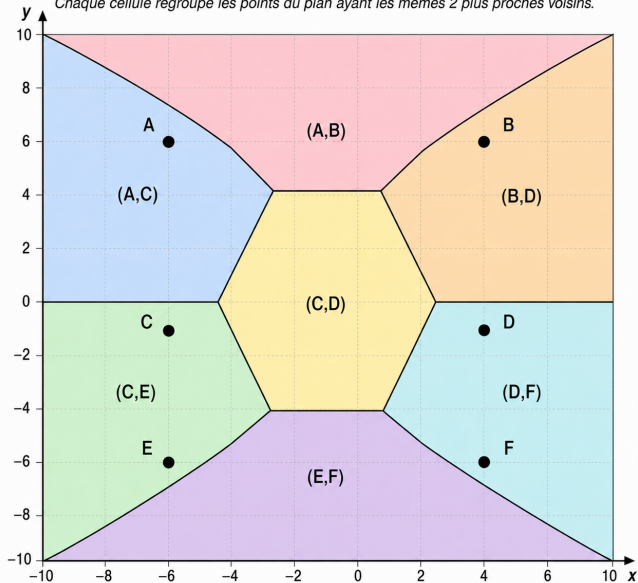
Diagrammes de Voronoi -2-



Diagrammes de Voronoi -3-

Diagramme de Voronoi d'ordre 2 ($k = 2$)

Chaque cellule regroupe les points du plan ayant les mêmes 2 plus proches voisins.



Prédicteur k -NN pour la régression aux moindres carrés

$$\hat{\eta}_{n,k}(x) = \sum_{m=1}^M \bar{Y}_{A_m} \mathbb{1}_{A_m}(x) \quad (24)$$

où $(A_m)_m$ partition définie à partir des rangs.

Classifieur binaire k -NN

$$\hat{g}_{n,k}(x) = \mathbb{1}_{[\hat{\eta}_{n,k}(x) > 1/2]} = \sum_{m=1}^M \mathbb{1}_{]1/2,1]}(\bar{Y}_{A_m}) \mathbb{1}_{A_m}(x) \quad (25)$$

où $(A_m)_m$ partition définie à partir des rangs.

- k -NN = k -ppv !

Exemple des iris de Fisher -1-

- Exemple incontournable : classification des iris de Fisher.
- Utilisé par Ronald Fisher en 1936 (données d'Edgar Anderson).
- 50 échantillons des 3 espèces d'iris (setosa, virginica et versicolor). 4 variables statistiques : longueur et largeur des sépales et des pétales.



Figure 1 – Les 3 espèces d'Iris (de gauche à droite versicolor, setosa, virginica),
Crédit photo : Frank Mayfield et Kosaciec Szczecinkowaty, Wikipédia.

Exemple des iris de Fisher -2-

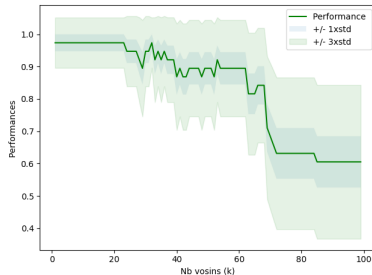
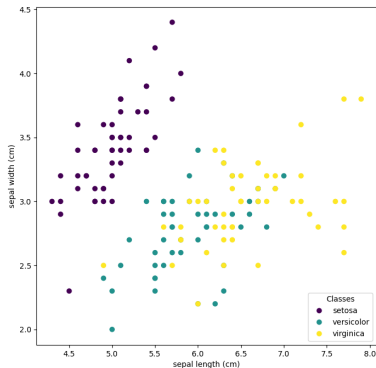


Figure 2 – À gauche, nuage de points représentant les iris de Fisher (une espèce par couleur) en fonction de la longueur et de la largeur de leur sépale. À droite, une courbe indiquant les performances de k -ppv sur la tâche de classification des iris, en fonction du nombre k .

Exemple des iris de Fisher -3-

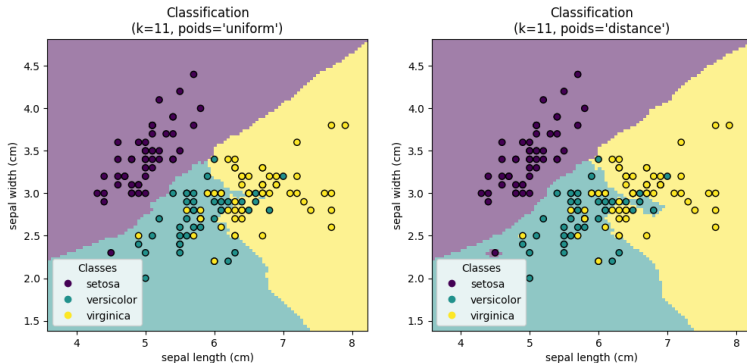
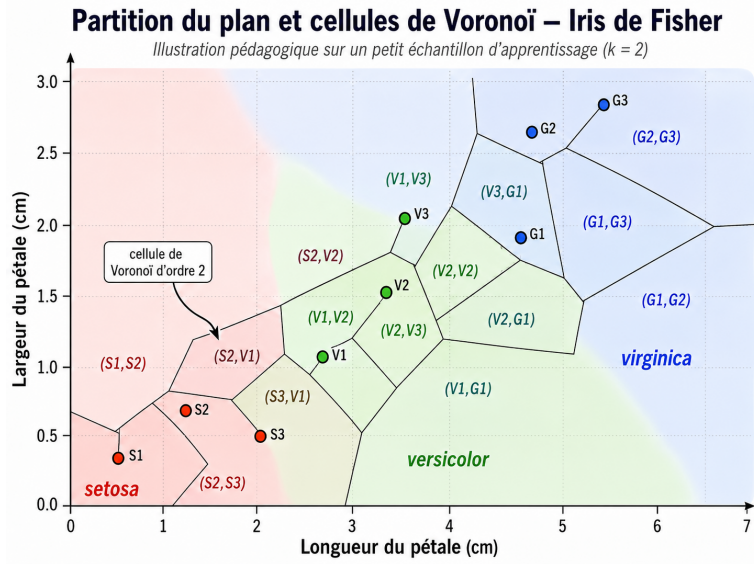


Figure 3 – Illustration d'un classifieur k -ppv pour $k = 11$, avec deux types de poids différents. Les zones de couleur représentent l'espèce qui sera affectée à une nouvelle observation selon la longueur et la largeur de son sépale. Simulations effectuées sous Python.

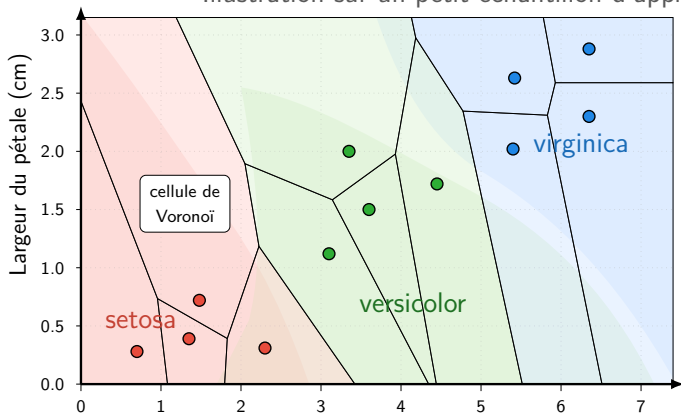
Exemple des iris de Fisher -4-



Exemple des iris de Fisher -5-

Partition du plan — Iris de Fisher

Illustration sur un petit échantillon d'apprentissage



Points d'apprentissage
(quelques points par espèce)

- *setosa*
- *versicolor*
- *virginica*

Cellules de Voronoï $k = 1$

- cellule générée par un point setosa
- cellule générée par un point versicolor
- cellule générée par un point virginica

Régions de décision
approximatives
classifieur 1-ppv

- région setosa
- région versicolor
- région virginica



Chaque point d'apprentissage possède sa **cellule de Voronoï** : l'ensemble des positions du plan pour lesquelles ce point est le plus proche.

Les **régions de décision** du classifieur 1-ppv regroupent toutes les cellules des points d'une même espèce.

k -ppv en régression

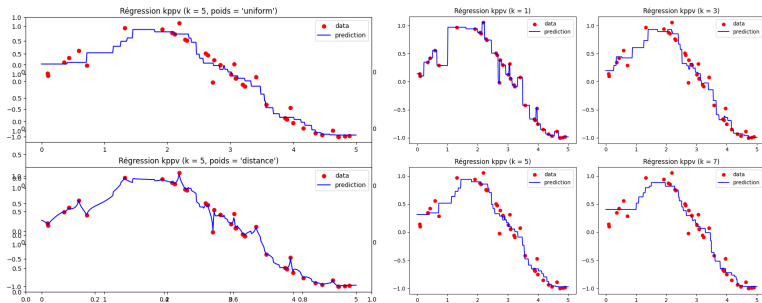


Figure 4 – Régression à l'aide de la méthode k -ppv. Un nuage de points (rouges) bruité est généré en perturbant aléatoirement les points d'une sinusoïde. La courbe bleue, issue de la méthode des k -ppv, doit reconstituer au mieux cette sinusoïde. Les deux courbes à gauche correspondent à $k = 5$ pour deux types de poids différents. Les 4 courbes de droite représentent la régression pour des valeurs différentes de $k = 1, 3, 5, 7$. Simulations effectuées sous Python.

k -ppv : une méthode non paramétrique

- La méthode des k -ppv est dite **non paramétrique** parce qu'elle ne suppose pas une forme fonctionnelle fixe pour la relation entre X et Y .
- Pour les k -ppv, on ne cherche pas à estimer un vecteur fini de paramètres θ . **On garde tout l'échantillon d'apprentissage** : la fonction apprise dépend directement des données stockées. Quand n augmente, le modèle devient plus riche, car il dispose de plus de points de référence.

Le terme **non paramétrique ne veut pas dire « sans aucun paramètre »**. Il y a des choix à faire et des paramètres à calibrer, mais qui ne définissent pas un modèle de dimension fixe :

- le nombre de voisins k ,
 - la distance utilisée (euclidienne, Manhattan, etc.)
 - une pondération des voisins.
 - la normalisation des variables.
- Non paramétrique : la complexité du modèle croît avec n .

- k est l'hyperparamètre le plus important du modèle.
- Choix par validation croisée : compromis entre variabilité et lissage.
- Méthode du coude («elbow») comme pour la méthode k -means.
- k petit : modèle flexible, mais plus instable (overfitting).
- k grand : modèle plus stable, mais biaisé (underfitting) et plus complexe.
- k -ppv fonctionne souvent très bien avec de petites valeurs de k !
- Parfois, on conseille $k \approx \sqrt{n}$...

- Existence de **nombreuses distances** : eculidienne, Manhattan, Mahalanobis, Hamming, Jaccard, **Levenshtein** (pour un texte).
- **Pas de choix universel** : dépend du problème.
- On peut pondérer les données ou pas (choix par défaut).
- Si aucun *a priori*, on peut choisir distance et pondération par validation croisée.

k -NN : complexité

- **Presque pas d'apprentissage** («lazy training»), mais la prédiction peut devenir coûteuse.
- Si n taille de l'échantillon, d dimension des données, k nombre de voisins...
- Coût de l'apprentissage : $\mathcal{O}(nd)$.
- Calcul des distances : $\mathcal{O}(nd)$.
- Trouver les k -ppv : $\mathcal{O}(n \ln n)$.
- Pour chaque nouveau point prédire : $\mathcal{O}(nd)$.
- **Principale limite de la méthode** : en grande dimension d , tous les points deviennent équidistants (malédiction de la dimension)....

k-NN Tableau de synthèse

Méthode	Inter-prétation	Perf.	Calibra-tion	Overfit.	Coût calc.
Régression linéaire	▲	▼	✗	▲	▲
Régression logistique	▲	▼	✗	▲	▲
Naive Bayes	▲	▼	▲	▲	▲
<i>k</i> -ppv	▼	▼	▲	▼	▲
SVM (kernel RBF)	✗	▲	▼	▼	▼
Réseau de neurones	✗	▲	✗	✗	✗
Arbres de décision	▲	✗	▲	✗	▲
Boosting	✗	▲	▲	▼	▼
Bagging (forêt)	✗	▲	▼	▲	▼

▲ : bon ▼ : moyen ✗ : faible ou problématique

• Ce tableau est une sorte de moyenne de ce que l'on trouve sur internet, il vaut ce qu'il vaut et ne donne qu'une idée générique. Le contenu des cases est finalement assez aléatoire. En fait, ne vous fiez pas du tout à ce type de tableau et quand vous traitez un problème de Machine Learning, **essayez toutes les méthodes !**

- The Elements of Statistical Learning, de Hastie, Tibshirani, Friedmann, éditions Springer, 2008 :
<https://www.sas.upenn.edu/~fdiebold/NoHesitations/BookAdvanced.pdf>.
- An introduction to Statistical Learning with applications in Python, de James, Witten, Hastie, Tibshirani, Taylor, chez Springer :
<https://www.statlearning.com/>.
- Introduction à l'apprentissage statistique, de Frédéric Sur :
https://members.loria.fr/FSur/enseignement/apprauto/poly_apprauto_FSur.pdf.
- Excellent cours de Ricco Rakotomalala :
<https://cours-machine-learning.blogspot.com/>

3. Arbres de décision et algorithme CART

Arbre de décision -1-

Un graphe est un ensemble de nœuds reliés par des arêtes. Un arbre est un graphe sans cycle.

Dans un **arbre de décision**, chaque nœud correspond à un sous-ensemble $A \subset \mathbb{X}$ et un test statistique T (**critère de segmentation, d'impureté**) est appliqué sur les variables explicatives $x \in \mathbb{X}$. Si le test peut donner K résultats $1, \dots, K$, alors le nœud (A, T) donne naissance à k nœuds fils, tel que l'ensemble A_k associé au k^e fils est $A_k = \{x \in A : T(x) = k\}$.

L'algorithme est initialisé à la racine correspondant à $A = \mathbb{X}$, puis on itère le procédé. Des branches sont élaguées si elles ne dégradent pas trop le taux d'erreur de l'arbre.

Différentes implémentations possibles selon :

- Les **critères de segmentation** choisis.
- Le **critère d'arrêt**.
- Le **critère d'élagage**.

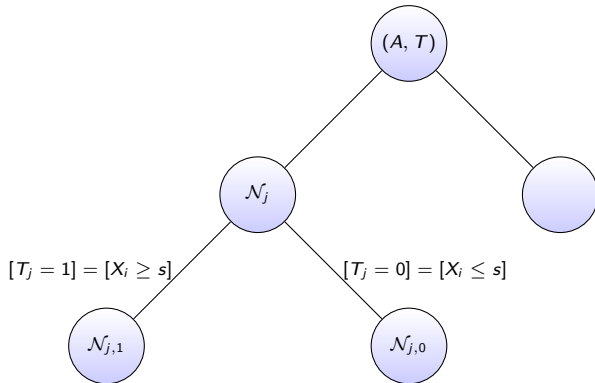


Figure 5 – Exemple d'arbre avec 5 nœuds et 4 arêtes. La racine A correspond à l'ensemble \mathbb{X} . À un nœud donné \mathcal{N}_j de profondeur j est affecté un sous-ensemble de \mathbb{X} . Le test courant T_j est effectué sur les observations contenues dans \mathcal{N}_j , qui sont réparties dans les deux nœuds fils $\mathcal{N}_{j,0}$ et $\mathcal{N}_{j,1}$ en fonction du résultat de T_j .

Algorithme CART de Breiman, 1984

Algorithm 1: CART algorithm

Input: Arbre = nœud racine

// Expansion

for chaque nœud n de Arbre **do**

if $n \neq$ condition d'arrêt **then**

 Choisir critère de segmentation T

 Créer les nœuds fils

 Maj : Arbre = Arbre \cup nœuds fils

end

end

// Élagage

for chaque nœud n de Arbre **do**

if $n =$ condition d'élagage **then**

 Maj : Arbre = Arbre $-$ nœuds fils et descendants

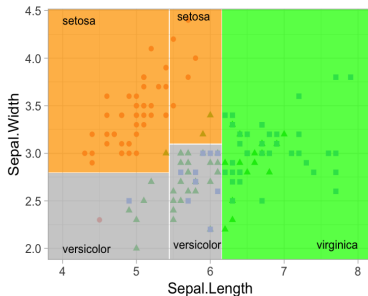
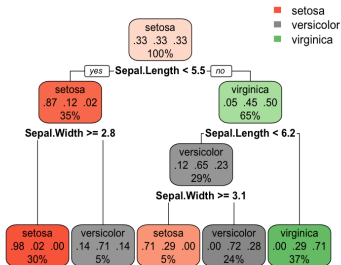
end

end

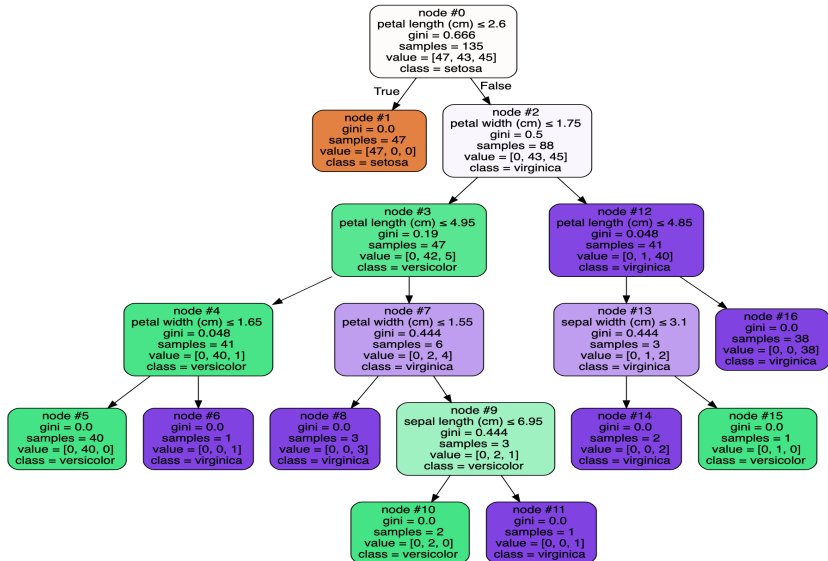
CART exemple 1 : iris de Fisher (2 variables)

L'algorithme CART est appliqué à la base de données des iris de Fisher.

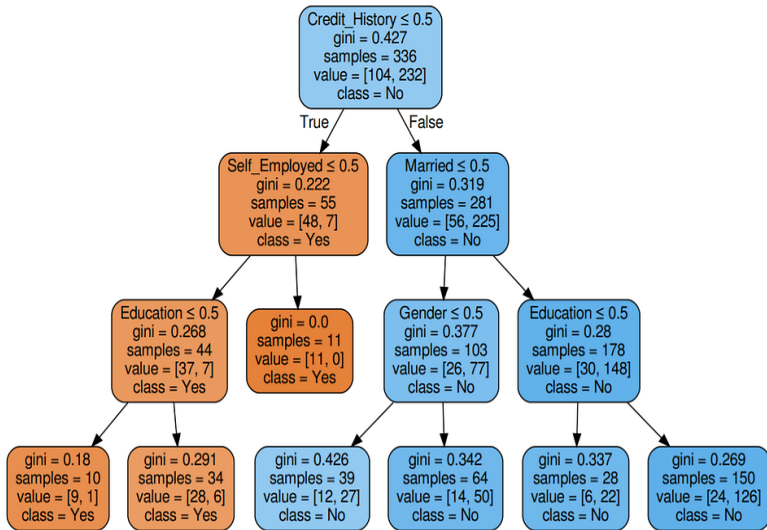
- À gauche, l'arbre de décision construit sur les variables longueur et largeur de la sépale.
- À droite, la partition de l'ensemble $\mathbb{X} = \mathbb{R}^2$ en fonction des critères.



CART exemple 2 - Iris de Fisher (4 variables)



CART exemple 3 : base de données des défauts de paiement



Une fois l'arbre construit, la règle de classification consiste simplement à **parcourir l'arbre depuis la racine** :

- Pour un $x \in \mathbb{X}$, on détermine la feuille (nœud terminal) qui le contient en parcourant l'arbre de haut en bas.
- En classification, on affecte à x l'étiquette y correspondant à la classe majoritairement représentée par les exemples x_i de cette feuille.
- En régression, on affecte à x la moyenne des étiquettes y_i correspondant aux exemples x_i de la feuille.

CART : segmentation, critère d'arrêt et élagage

Le **critère d'arrêt** vérifie l'une des conditions suivantes sur un seuil donné :

- Chaque feuille satisfait un seuil d'homogénéité.
- La profondeur de l'arbre dépasse le seuil.
- Le nombre de feuilles dépasse le seuil.
- L'effectif du nœud est inférieur au seuil.

La **segmentation** peut produire de bonnes performances sur l'ensemble des données d'entraînement, mais est susceptible de provoquer du **sur-apprentissage** si l'arbre est trop complexe. Un arbre avec moins de branches aura une variance plus faible en contrepartie d'un biais un peu plus fort.

Une stratégie pour simplifier l'arbre est d'élaguer des branches qui ne contribuent pas trop à augmenter le risque estimé, c'est le processus d'**élagage** (« **pruning** »).

Critères de segmentation 1 : l'indice de Gini

- **Indice de Gini** ou **entropie** pour classification, **variance** pour régression.

Définition de l'indice de Gini

$$\forall A \subset \mathbb{X}, G(A) = 1 - \bar{Y}_A^2 - (1 - \bar{Y}_A)^2 \quad (26)$$

- $G(A) = 0 \iff \bar{Y}_A = 0$ ou 1 .
- Un nœud sera de bonne qualité (**homogène, pur**) si une très grande majorité des étiquettes des exemples associés à ce nœud sont identiques. Il est alors très discriminant et $G(A)$ est presque nul.
- Pour évaluer la qualité d'un critère de segmentation, on calcule le gain d'**homogénéité** lorsque A est segmenté en A_1 et A_2 :

$$I_G(A_1, A_2) = G(A) - qG(A_1) - (1 - q)G(A_2) \quad (27)$$

avec $q = N_{A_1}/N_A$ proportion des $x_i \in A$ qui se dirigent vers A_1 . CART choisit la partition qui maximise I_G à chaque étape.

Critères de segmentation 2 : l'entropie

La **quantité d'information de Shannon** apportée par la réalisation d'un évènement A est $\mathbb{I}(A) = -\mathbb{P}(A) \log_2 \mathbb{P}(A)$. Elle mesure la vraisemblance de cet évènement.

L'**entropie** d'une va mesure l'**information moyenne** sur l'ensemble des réalisations possibles. C'est le nombre moyen de questions binaires que l'on doit poser pour déterminer la valeur exacte de la variable aléatoire.

Entropie d'une va ou d'une mesure de probabilité

$$H(X) = -\mathbb{E}[\log_2 \mathbb{P}(X)] = -\sum_{i=1}^n p_i \log_2 p_i \quad (28)$$

On l'utilise comme critère de segmentation en calculant l'entropie de la mesure de probabilité empirique uniforme créée par la partition des données (p_i proportion de 1 (ou 0) dans chaque classe).

Critères de segmentation 3 : la variance en régression

- On rappelle que dans une tâche de régression, la valeur affectée à un nœud est la moyenne des observations appartenant à ce nœud.
- **L'hétérogénéité est mesurée par la variance du nœud.** Si l'on note \mathcal{N} le nœud et \bar{y} la moyenne des valeurs y_i des observations se trouvant dans le nœud correspondant,

$$\mathbb{V}(\mathcal{N}) = \frac{1}{|\mathcal{N}|} \sum_{i: x_i \in \mathcal{N}} (y_i - \bar{y})^2. \quad (29)$$

- La segmentation privilégie le découpage en deux nœuds homogènes dont la variance sera la plus faible possible.

- Élaguer pour éviter le sur-apprentissage et garder un modèle simple.
- Complexité d'un arbre donnée par sa dimension de Vapnik (hors programme) ou bien son nombre de coupures ou sa profondeur.
- Tester tous les sous-arbres ? Trop coûteux. Méthode de Breiman : se limiter à une suite de sous-arbres emboîtés de taille raisonnable.
- Choisir un arbre de la suite par minimisation d'un risque d'ajustement.

Si T arbre dont les noeuds terminaux sont les \mathcal{N}_m . Deux risques d'ajustement classiques (respectivement régression et classification)

$$R_r(m) = \frac{1}{N_m} \sum_{i: x_i \in \mathcal{N}_m} (y_i - \bar{y}_m)^2 \text{ et } R_c(m) = \frac{1}{N_m} \sum_{i: x_i \in \mathcal{N}_m} \mathbb{1}_{[y_i \neq \bar{y}_m]} \quad (30)$$

- On définit un critère **côût/complexité** par

$$C_\alpha(T) = \sum_{m=1}^M N_m R_m(T) + M\alpha \quad (31)$$

- $M = |T|$ nombre de nœuds de T , $\alpha \geq 0$ paramètre d'ajustement.
- La somme représente l'erreur totale (le risque) sur l'arbre, le paramètre αM pénalise le nombre total de feuilles (la complexité).
- On cherche l'arbre T qui minimise $C_\alpha(T)$ pour α bien choisi ($\alpha = 0$ arbre entier, $\alpha = \infty$ racine uniquement).
- **Approche CART de Breiman** : régularise pour améliorer les performances en prédiction, évite toute l'exploration de l'espace des solutions, intègre une préférence pour la simplicité (règle de l'écart-type).

Théorème de Breiman, 1984

Il existe une suite finie $0 = \alpha_0 < \dots < \alpha_M$ avec $M \leq |T|$ et une suite imbriquées de sous-arbres $(T_{\alpha_m})_m$ avec

$$T = T_{\alpha_0} \subseteq T_{\alpha_1} \subseteq \dots \subseteq T_{\alpha_M} = \text{racine} \quad (32)$$

telle que $\forall \alpha \in [\alpha_m, \alpha_{m+1}[$,

$$T_m \in \arg \min_{T_i \subset T} C_\alpha(T_i) \quad (33)$$

- Choisir un arbre revient à choisir une valeur de α : sur le chemin de coût/complexité, en augmentant α , on trouve une succession d'arbres emboîtés de taille décroissante.

- Excellente vidéo de Ricco Rakotomalala :

<https://www.youtube.com/watch?v=if6QEtJP77E>

- La visualisation de l'arbre peut donner une idée sur l'**importance des variables**.
- Mais certaines variables possèdent une grande importance sans apparaître explicitement dans l'arbre.
- Difficile de quantifier l'importance juste en regardant l'arbre !
- C'est le même problème que la **significativité** d'un régresseur dans une régression linéaire ou logistique.
- La mesure d'importance d'un arbre est basée sur le **gain d'impureté des noeuds internes**.

Plus de détails seront donnée en TP.

CART : exemple en classification

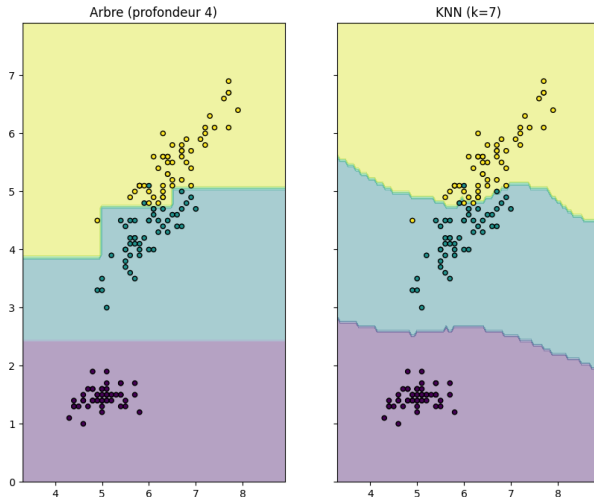


Figure 6 – Classification des iris de Fisher par arbre CART profondeur 4 (à gauche) et k -ppv pour $k = 7$ (à droite).

CART : exemple en régression

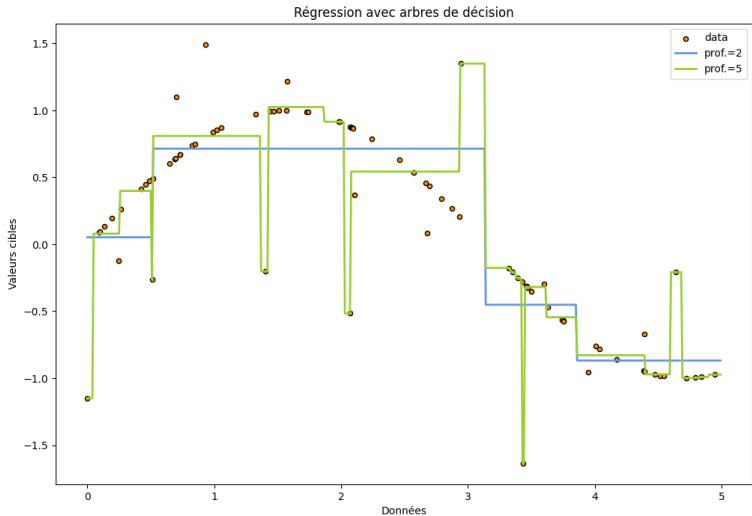
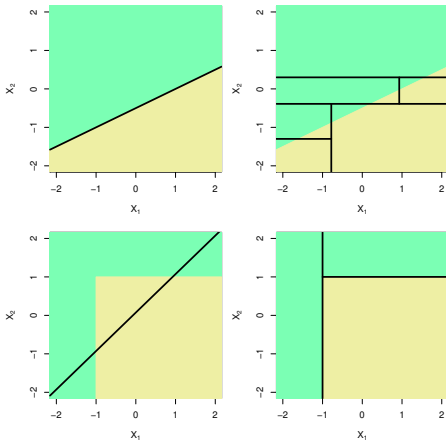


Figure 7 — Régression CART. Nuage de points (oranges) généré avec sinusoïde + bruit aléatoire. Objectif : reconstruire la courbe avec arbre de profondeur 2 (en bleu) ou 5 (en vert).

Arbres : une méthode non linéaire

En fonction des données, les arbres peuvent obtenir de meilleures performances (ou pas) que les méthodes linéaires.



Remarques finales

- Les arbres sont des **modèles hiérarchiques, non linéaires**, itératifs.
- Ils sont adaptés aux problèmes continus, discrets, multiclassés, multimodaux.
- Ils prédisent par l'intermédiaire d'une **stratification (partition)** de l'espace des données.
- Les arbres proposent une **méthode graphique intuitive** et facile à comprendre.
- Ils ne sont **pas performants** !
- Ils ne sont **pas robustes** au bruit dans les données.
- Variantes diverses : CID3, C4.5, C5, CHAID, MARS, QUEST, etc.

Des méthodes plus robustes et ayant de meilleures performances peuvent se déduire des arbres de décision : les forêts aléatoires et le bagging, par exemple, dont nous parlerons dans un autre chapitre.