Polycopié de la ressources *R6.EMS.05 : Apprentissage statistique pour l'IA*

Département SD IUT2 de Grenoble

Cette ressource s'appuie sur les ressources :

- R4.EMS.08 - Modèle linéaire
- R5.02 - Data mining
- R5.EMS.06 - Modélisation statistique avancée
- R6.02 - Méthodes statistiques pour le Big Data

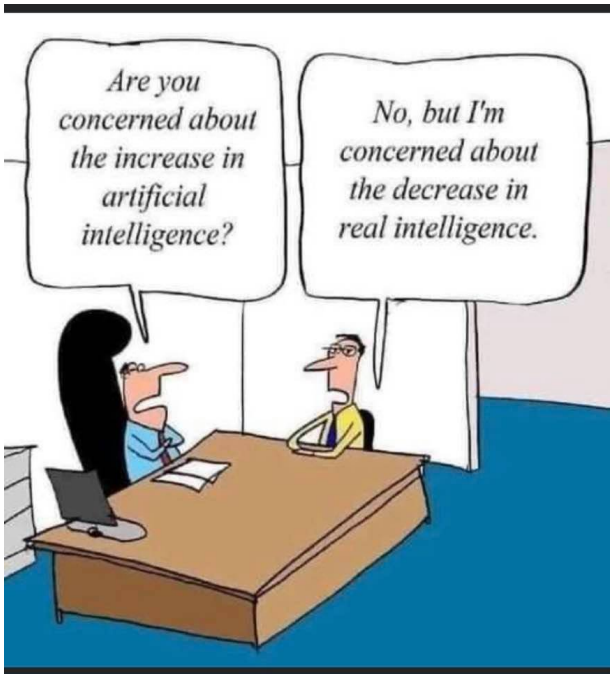

FIGURE 1 – Humour sur l'intelligence artificielle

Table des matières

Avant propos

Ce cours clôture les cours de spécialisation du parcours *Exploration et Modélisation Statistique* et porte sur l'utilisation de la statistique dans le développement de l'Intelligence Artificielle. En plus des ressources du BUT mises sur la page de garde, ce polycopié s'appuie sur les références suivantes :

- Le livre *Les forêts aléatoires avec R* de Genuer et Poggi (2016).
- La chaîne Youtube *Sciences étonnantes* de Louapre.

Nous vous encourageons à les consulter si vous voulez plus de précisions ou une vision différente des notions présentées.

Planning

- Cours/TD 1 : Rappels sur les prédicteurs et les arbres aléatoires

---

- TP 1 : Rappels sur les arbres
- Cours/TD2 : Forêts aléatoires
- TP 2 : Arbres et forêts aléatoires
- Cours/TD3 : Fin des forêts et introduction des réseaux de neurones

---

- TP 3 : Fin des arbres aléatoires
- TP 4 : Introduction aux réseaux de neurones
- TP 5 : Challenge sur la reconnaissance d'images
- TP 6 : Examen

Chapitre 1

Forêts aléatoires

"Trois statisticiens partent chasser du canard. Leur chien court après un canard qui commence à s'envoler. Le premier statisticien lève son fusil et le vise, mais tire un peu trop haut. Le second lève à son tour le fusil et vise, mais tire par contre un peu trop bas. Le troisième dit alors : « On l'a eu ! »"

Recueil de blagues mathématiques et autres curiosités de Winckler (2011)

Avant de parler des forêts aléatoires, nous devons faire des rappels sur les arbres aléatoires.

1.1 Concepts généraux sur les prédicteurs

Les chapitres présentés dans la fin de ce cours repose sur la notion de prédicteurs. Nous reprenons donc le vocabulaire qui peut parfois varier d'un auteur à l'autre.

Définitions 1 (Échantillon d'apprentissage)

Nous appelons **échantillon d'apprentissage**, noté $\mathcal{L}_n$, la famille composée de n couples d'observations $(\boldsymbol{X}_i, Y_i)_{1\leq i\leq n}$ indépendantes et identiquement distribuées suivant la même loi de couple $(\boldsymbol{X}, Y)$. La variable $\boldsymbol{X}_i \in \boldsymbol{\mathcal{X}}$ est la **variable explicative** appartenant à un ensemble de dimension p et la variable $Y_i \in \boldsymbol{\mathcal{Y}}$ est la **variable à expliquer** ou **variable réponse** dont l'ensemble $\boldsymbol{\mathcal{Y}}$ sera supposé être :

- $\boldsymbol{\mathcal{Y}} = \mathbb{R}$: dans ce cas, on parlera de **problème de régression**,
- $\boldsymbol{\mathcal{Y}} = \{1, \dots, K\}$: dans ce cas, on parlera de **problème de classification**.

Exercices 1.1

Explicitez les notations des définitions précédentes dans le cas d'un modèle linéaire. Est-on dans un problème de régression ou de classification ?

Définition 2 (Prédicteur)

Un **prédicteur** est une fonction de $\widehat{h}$ allant de $\boldsymbol{\mathcal{X}}$ dans $\boldsymbol{\mathcal{Y}}$:

$$\widehat{h} : \boldsymbol{\mathcal{X}} \to \boldsymbol{\mathcal{Y}}.$$

Le prédicteur va donc prendre une observation $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}$ et proposer une prédiction $\widehat{y} = \widehat{h}(x)$.

Exercices 1.2

Prenons un modèles de régression linéaire comme vu dans la ressource *R4.EMS.08 - Modèle linéaire* :

$$Y_i = \alpha^\star X_i + \varepsilon_i \tag{1.1}$$

où $\alpha^\star \in \mathbb{R}$ et ε_i sont des variables indépendantes et de même loi $\mathcal{N}(0, \sigma^{\star 2})$.

1. Étant donné un paramètre α et $\sigma \in \mathbb{R}_+^\star$, calculez la logvraisemblance de Y connaissant X associée au modèle (1.1).
2. Calculez les estimateurs du maximum de vraisemblance des paramètres $\alpha^\star$ et ${\sigma^\star}^2$.
3. En déduire un prédicteur de Y en fonction de X.

Définitions 3 (Évaluations)

Pour estimer la qualité d'un prédicteur $\widehat{h}$, nous avons deux types de critères à optimiser suivant l'objectif :

- **Régression** : l'**erreur quadratique** est définie par :

$$\mathbb{E}\left[\left(Y - \widehat{h}(\boldsymbol{X})\right)^2\right]. \tag{1.2}$$

- **Classification** : la **probabilité de mauvais classement** est définie par :

$$\mathbb{P}\left(Y \neq \widehat{h}(\boldsymbol{X})\right). \tag{1.3}$$

Remarque

L'erreur quadratique de l'équation (1.2) et la probabilité de mauvais classement de l'équation (1.3) reposent toutes les deux sur la loi **inconnue** du couple $(\boldsymbol{X}, Y)$. Pour contrer ce problème, nous utilisons les versions empiriques.

Définitions 4 (Échantillon test et évaluations empiriques)

Nous appelons **échantillon de test**, noté $\mathcal{T}_m$, la famille composée de m couples d'observations $(\boldsymbol{X}'_i, Y'_i)_{1\leq i\leq n}$ indépendantes et identiquement distribuées suivant la même loi de couple $(\boldsymbol{X}, Y)$ (qui est également celle de l'échantillon d'apprentissage). Avec cet échantillon, nous avons deux types de critères à optimiser suivant l'objectif :

- **Régression** : l'**erreur quadratique moyenne** est définie par :

$$\frac{1}{m}\sum_{i=1}^{m}\left[Y'_i - \widehat{h}(\boldsymbol{X}'_i)\right]^2 \tag{1.4}$$

- **Classification** : le **taux de mauvais classement** est défini par :

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{1}_{\left\{Y'_i \neq \widehat{h}(\boldsymbol{X}'_i)\right\}}. \tag{1.5}$$

Remarque

Ainsi l'équation (1.4) (resp. (1.5)) est une version empirique de la version (1.2) (resp. (1.3)). Sous la condition que la loi du couple $(\boldsymbol{X}, Y)$ vérifie les conditions des lois grands nombres, les estimations empiriques seront d'autant plus proches de la version continue qu'il y aura des observations dans l'échantillon de test $\mathcal{T}_m$.

Attention au piège

Si nous utilisons l'échantillon d'apprentissage en tant qu'échantillon de test, nous faisons du **surapprentissage**. De manière générale, les résultats d'une méthode peut légitimement être remis en doute si les échantillons d'entraînement $\mathcal{L}_n$ et de test $\mathcal{T}_m$ ne sont pas disjoints.

1.2 Arbres de décision

Les arbres de décision ayant été étudiés dans la ressource *R5.02 - Data mining*, nous ne parlerons ici que du principe est des notations afin de reprendre sur des bases saines.

Définition 5 (Arbre de décision)

Un **arbre de décision** se compose de (voir la figure 1.1 pour une représentation schématique) :

- Racine : Première question.
- Noeuds : Questions intermédiaires.
- Branches : *Oui* à gauche et *Non* à droite.
- Feuilles : Décision finale.

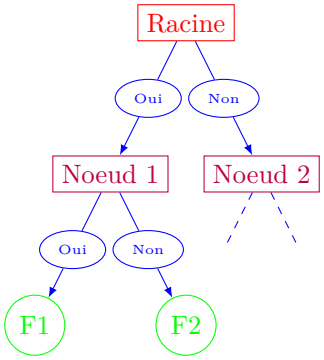

FIGURE 1.1 – Représentation schématique des différentes notions d'un arbre de décision

Exemple

En reprenant les données sur la survie du `Titanic` de Eaton et Haas (1995), nous obtenons l'arbre de la figure 1.2. Dans cette représentation, nous avons fait le choix de proposer les variables binaires sous de choix à deux valeurs pas "oui" et "non". La première question qui va séparer l'arbre en 2 est celle du *genre* des passagers. Pour les hommes, les principales questions sont l'âge et le nombre de proches tandis que, pour les femmes, vient immédiamment la question de la classe au sein du bâteau puis les questions posées pour les hommes.

Définition 6 (Partition)

Étant donné un ensemble $\boldsymbol{\mathcal{X}}$, on appelle **partition** une famille $(\boldsymbol{\mathcal{X}}_\ell)_{1\leq\ell\leq L}$ de sous ensembles vérifiant les deux conditions suivantes :

$$(1) \quad \bigcup_{\ell=1}^{L} \boldsymbol{\mathcal{X}}_\ell = \boldsymbol{\mathcal{X}},$$
$$(2) \quad \forall \ell \neq \ell', \boldsymbol{\mathcal{X}}_\ell \cap \boldsymbol{\mathcal{X}}_{\ell'} = \emptyset$$

où $\emptyset$ est l'ensemble vide.

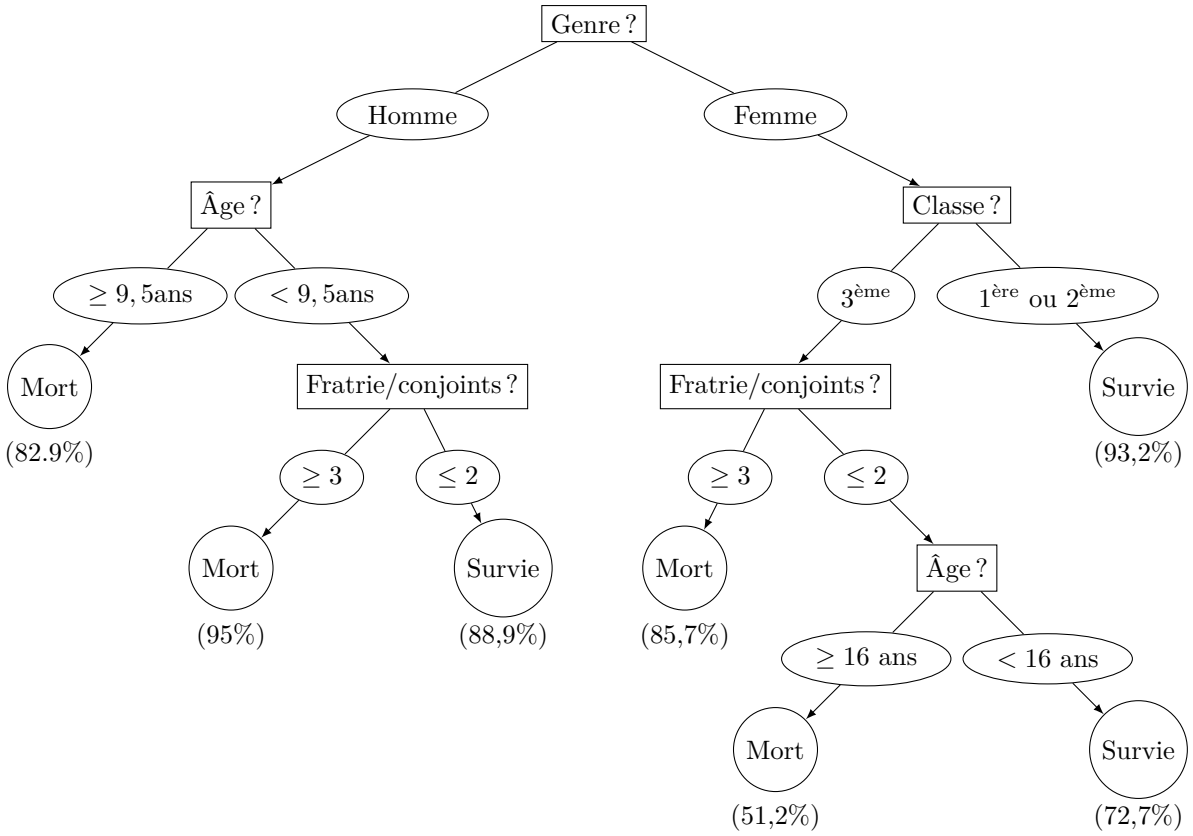

FIGURE 1.2 – Représentation d'un arbre de décision sur les chances de survie d'un passager du `Titanic` basé sur le jeu de données de Eaton et Haas (1995)

Définition 7 (Partition et arbre)

S'il est correctement construit, un arbre est une partition de l'ensemble $\boldsymbol{\mathcal{X}}$ construit de façon itérative :

- La racine représente l'ensemble $\boldsymbol{\mathcal{X}}$.
- Pour la racine ou pour chaque noeud $\boldsymbol{\mathcal{N}}$, le noeud ou la feuille de gauche $\boldsymbol{\mathcal{N}}_L$ et le noeud ou la feuille de droite $\boldsymbol{\mathcal{N}}_R$ forme une partition du noeud $\boldsymbol{\mathcal{N}}$:

$$\boldsymbol{\mathcal{N}} = \boldsymbol{\mathcal{N}}_L \cup \boldsymbol{\mathcal{N}}_R \text{ et } \boldsymbol{\mathcal{N}}_L \cap \boldsymbol{\mathcal{N}}_R = \emptyset.$$

Attention au piège

Chaque noeud va donc couper en deux sous ensembles l'un des ensembles précédents de la partition. La famille des feuilles forme une partition de l'ensemble $\boldsymbol{\mathcal{X}}$.

Définitions 8 (Classification And Regression Trees (CART))

La méthode statistique ***Classification And Regression Trees*** (abrégée *CART*) introduite par Breiman et al. (1984) permet de construire des prédicteurs par arbre en prenant à chaque étape une des variables $\boldsymbol{X}_{\cdot,j}$ est en opérant un découpage des individus du type :

- Si les variables sont **quantitatives**, il est existe un seuil S telle que nous prenons deux ensembles :

$$\{i \in \{1,\dots,n\} \,|X_{i,j} \leq S\} \text{ et } \{i \in \{1,\dots,n\} \,|X_{i,j} > S\}.$$

- Si les variables sont **qualitatives ordonnées**, il est existe une modalité a_k telle que nous prenons deux ensembles :

$$\{i \in \{1,\dots,n\} \,|X_{i,j} \leq a_k\} \text{ et } \{i \in \{1,\dots,n\} \,|X_{i,j} > a_k\}.$$

- Si les variables sont **qualitatives nominales**, il est existe un sous ensemble $\mathcal{M}$ des modalités tel que nous prenons deux ensembles :

$$\{i \in \{1,\dots,n\} \,|X_{i,j} \in \mathcal{M}\} \text{ et } \{i \in \{1,\dots,n\} \,|X_{i,j} \notin \mathcal{M}\}.$$

Les signes $\leq$ et $>$ peuvent être $\geq$ et $<$; l'important est de former une partition de l'espace à chaque question. Pour le choix de la variable et de la condition, nous prenons une **fonction de coût** $\mathcal{C}$ basée sur les erreurs vues dans la définition 4 étant donné un noeud $\boldsymbol{\mathcal{N}}$ dont nous notons $\#\boldsymbol{\mathcal{N}}$ le nombre d'individus i dont la variable $\boldsymbol{X}_i \in \boldsymbol{\mathcal{N}}$:

- **Régression** :

$$\mathcal{C}(\boldsymbol{\mathcal{N}}) = \frac{1}{\#\boldsymbol{\mathcal{N}}} \sum_{i:\boldsymbol{X}_i \in \boldsymbol{\mathcal{N}}} \left(Y_i - \overline{\boldsymbol{Y}}_{\boldsymbol{\mathcal{N}}}\right)^2$$

avec $\overline{\boldsymbol{Y}}_{\boldsymbol{\mathcal{N}}}$ la moyenne sur l'espace concerné :

$$\overline{\boldsymbol{Y}}_{\boldsymbol{\mathcal{N}}} = \frac{1}{\#\boldsymbol{\mathcal{N}}} \sum_{i:\boldsymbol{X}_i \in \boldsymbol{\mathcal{N}}} Y_i.$$

- **Classification** :

$$\mathcal{C}(\boldsymbol{\mathcal{N}}) = \sum_{k=1}^{K} \widehat{p}_k^{(\boldsymbol{\mathcal{N}})} \left(1 - \widehat{p}_k^{(\boldsymbol{\mathcal{N}})}\right)$$

avec $\widehat{p}_k^{(\boldsymbol{\mathcal{N}})}$ la proportion d'éléments du cluster k dans l'ensemble :

$$\widehat{p}_k^{(\boldsymbol{\mathcal{N}})} = \frac{1}{\#\boldsymbol{\mathcal{N}}} \sum_{i:\boldsymbol{X}_i \in \boldsymbol{\mathcal{N}}} \mathbb{1}_{\{Y_i = k\}}.$$

Le but est, à chaque étape, de trouver le noeud $\boldsymbol{\mathcal{N}}$, la variable j et la condition de telle sorte que la fonction suivante soit maximale :

$$\mathcal{C}(\boldsymbol{\mathcal{N}}) - \left[\frac{\#\boldsymbol{\mathcal{N}}_L}{\#\boldsymbol{\mathcal{N}}}\mathcal{C}(\boldsymbol{\mathcal{N}}_L) + \frac{\#\boldsymbol{\mathcal{N}}_R}{\#\boldsymbol{\mathcal{N}}}\mathcal{C}(\boldsymbol{\mathcal{N}}_R)\right]. \tag{1.6}$$

Remarque

Les découpages se font toujours par rapport à un hyperplan orthgonal à une direction. Par exemple, dans le cas d'un plan, c'est une droite parallèle à l'un des axes et, en 3 dimension, c'est un plan dont le vecteur orthogonal est proportionnel à l'un de ceux de la base.

Remarque

Dans l'équation (1.6), si le coût ne change pas dans les noeuds enfants (c'est-à-dire que $\mathcal{C}(\boldsymbol{\mathcal{N}}_L) = \mathcal{C}(\boldsymbol{\mathcal{N}}_R) = \mathcal{C}(\boldsymbol{\mathcal{N}})$) alors la différence est nulle. Ce cas peut se faire, par exemple, s'il n'y a plus qu'une seule classe dans l'un des noeuds.

Attention au piège

En terme computationnel (voir la ressource *R6.02 - Méthodes statistiques pour le Big Data*), nous avons :

- Pour les variables quantitatives et qualitatives ordinales, la recherche du meilleur seuil est linéaire avec le nombre de modalités effectives.

- Pour les variables qualitatives nominales, la recherche du sous groupe est exponentiel avec le nombre de modalités effectives.

Il est donc préférable de limiter l'usage de variables nominales et/ou de limiter le nombre de modalités possibles.

Exemple fil rouge

Sur la figure 1.3, nous avons représenté la simulation 1000 points de façon uniforme dans un carré de $[0;1]^2$ dont nous avons affecté des couleurs suivant une loi qui dépend des emplacements des points. Sur la gauche, nous avons représenté l'arbre sélectionné par la fonction `rpart` du logiciel R. De plus, nous avons ajouté sur la gauche les traits correspondants aux emplacements des ruptures. Nous pouvons voir que les traits sont soient horizontaux, soient verticaux et qu'ils touchent tous les bords ou un autre traits. Les rectangles obtenus à la fin forment bien une partition puisque un point est dans un et un seul rectangle.

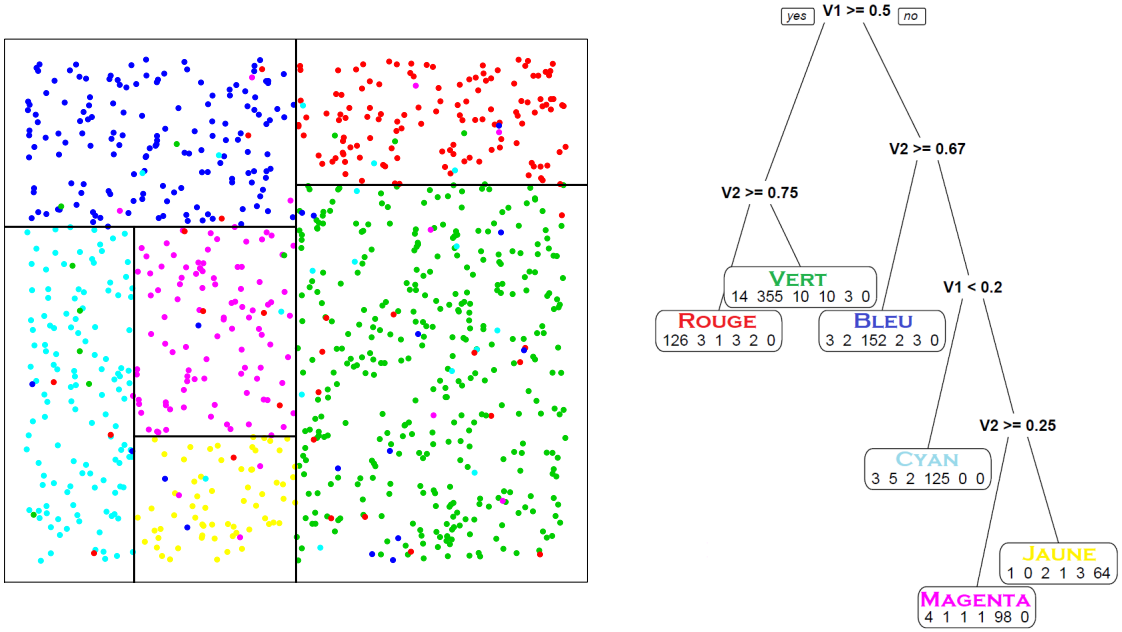

FIGURE 1.3 – Représentation d'un jeu de données (à gauche) avec des couleurs suivant la position des points et l'arbre *CART* associé (à droite) pour prédire la couleur d'un nouveau point. Les traits sur la figure de gauche correspondent aux décisions des noeuds de l'arbre.

Définitions 9 (Arbre maximal et élagage)

Le découpage de l'arbre vu dans la définition 8 s'arrête généralement suivant si un noeud est *pur* c'est-à-dire que la fonction de coût est nulle (dans le cas de la classification, c'est qu'il n'y a des représentants plus que d'une seule classe) ou un nombre trop petit d'individus (pour éviter trop de coupes). Cet arbre est alors appelé l'**arbre maximal** que nous noterons $\mathcal{A}_{\max}$. Il est souvent signe de **surapprentissage** c'est-à-dire que nous avons un modèle qui *est trop proche* des données et qui oublierait qu'il y a de l'aléatoire.

Une fois cet arbre construit, nous devons donc **l'élaguer**. Pour ce faire, nous prenons un nombre n_f de feuilles strictement plus petit que le nombre $n_{\max}$ de feuilles de l'arbre maximal et cherchons l'arbre $\mathcal{A}_{n_f}$ qui optimise la fonction de coût sur l'*ensemble de test* qu'on aura à disposition (voir la section 1.1) c'est-à-dire en faisant le calcul :

$$\overline{\text{err}}\left(\mathcal{A}_{n_f}\right) = \frac{1}{m}\mathcal{C}\left(\boldsymbol{\mathcal{F}}_k\right)$$

où $\boldsymbol{\mathcal{F}}_k$ est l'ensemble des individus de l'échantillon test $\mathcal{T}_m$ qui se retrouve dans la $k^{\text{ème}}$ feuille. Au final, pour sélectionner l'arbre le plus parcimonieux, nous prenons une valeur $\alpha > 0$ et nous

choisissons l'arbre minimisant la fonction suivante :

$$crit_\alpha\left(\mathcal{A}_{n_f}\right) = \overline{\text{err}}\left(\mathcal{A}_{n_f}\right) + \alpha n_f. \tag{1.7}$$

Remarque

Dans le critère (1.7), α vient pénaliser les modèles trop complexes. Si nous prenons le cas extrême où α tend vers l'infini, à partir d'un moment, seul l'arbre restreint à la racine fonctionnera.

Remarque

En pratique, il est trop coûteux d'étudier tous les arbres possibles. Généralement, les fonctions implémentées vont élaguer une branche par une branche en cherchant la plus pertinente à chaque fois. Cet algorithme obtiendra un arbre optimal *localement*.

Exemple fil rouge

En reprenant l'exemple précédent de la figure 1.3, nous pouvons aller plus en profondeur dans la segmentation de notre domaine (voir la figure 1.4). Nous pouvons voir que nous séparons effectivement quelques points proches des bords mais nous pouvons nous interroger sur la pertinence de faire cela.

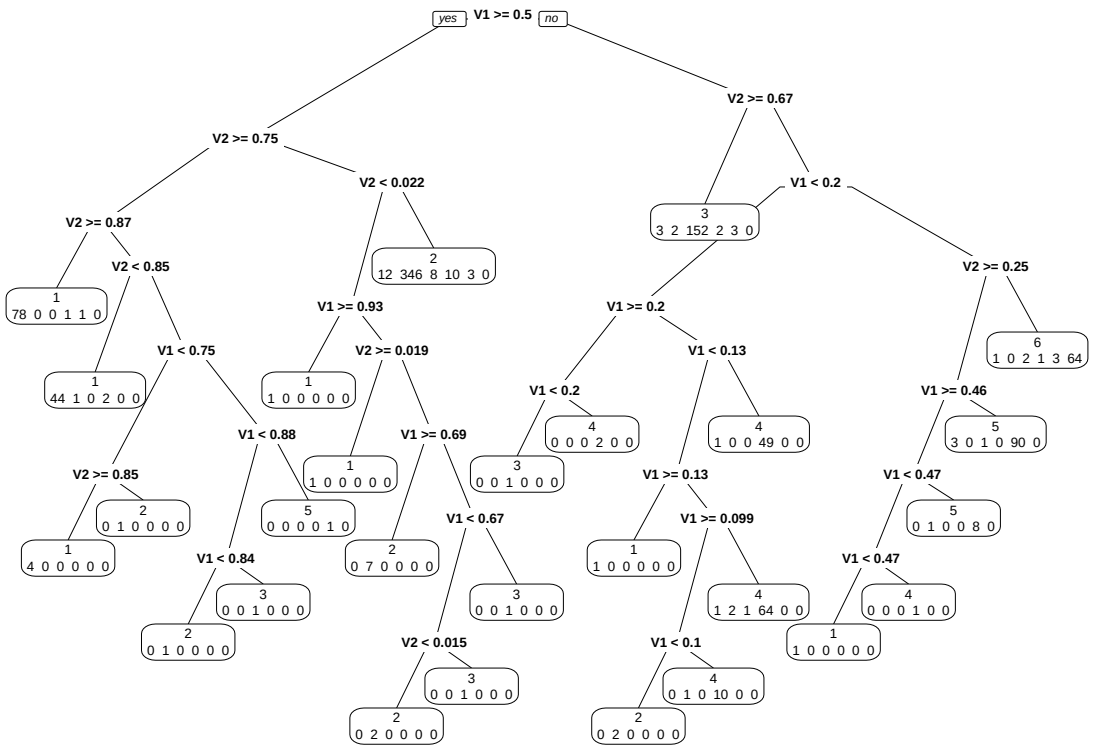

FIGURE 1.4 – Représentation d'un arbre plus profond obtenu en sous découpant encore les zone de la figure 1.3. Les couleurs sont représentées par leur numéro de classe.

1.3 Forêts aléatoires

Le principe des forêts aléatoires est d'interroger plusieurs petits arbres au lieu d'un seul grand arbre. Pour ce faire, la méthode combine l'utilisation du *bootstap* et du *boosting* vues en début de ressource avec

celle des arbres aléatoires vue en ressource *R5.02 - Data mining*.

Le principe est le suivant :

Définition 10 (Forêts aléatoires)

Étant donnés un n échantillon d'entraînement $\mathcal{L}_n$ et une collection de Q prédicteurs $\left(\widehat{h}(\cdot, \boldsymbol{\Theta}_1), \ldots, \widehat{h}(\cdot, \boldsymbol{\Theta}_Q)\right)$ dépendants de Q variables aléatoires $(\boldsymbol{\Theta}_1, \ldots, \boldsymbol{\Theta}_Q)$ indépendantes et de même loi (que nous définirons plus tard) et surtout indépendantes de $\mathcal{L}_n$, le prédicteur des **forêts aléatoires** (ou *random forest* en anglais) que nous notons $\widehat{h}_{RF}$ et construit en agrégeant les Q prédicteurs de la façon suivante pour tout $x \in \boldsymbol{\mathcal{X}}$:

- **Régression** : la moyenne des prédictions

$$\widehat{h}_{RF}(x) = \frac{1}{Q} \sum_{q=1}^{Q} \widehat{h}(x, \boldsymbol{\Theta}_q). \tag{1.8}$$

- **Classification** : le vote majoritaire des prédictions individuelles

$$\widehat{h}_{RF}(x) \in \underset{1 \leq k \leq K}{\operatorname{argmax}} \left[\sum_{q=1}^{Q} \mathbb{1}_{\left\{\widehat{h}(x, \boldsymbol{\Theta}_q) = k\right\}} \right]. \tag{1.9}$$

La figure 1.5 représente une version schématique des forêts aléatoires.

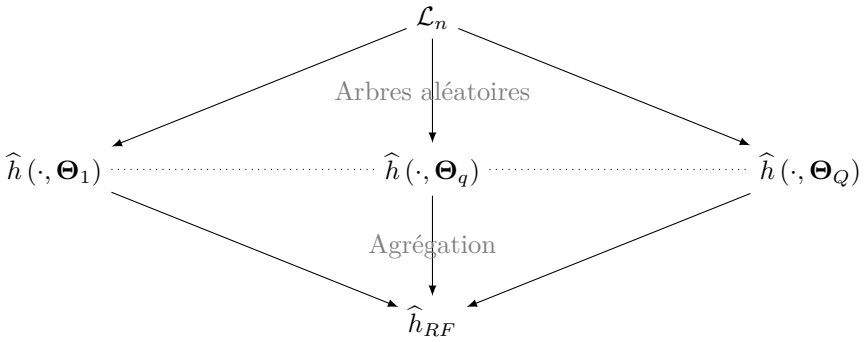

FIGURE 1.5 – Représentation schématique du principe général de forêt aléatoire : on part d'un n échantillon d'apprentissage avec lequel nous créons Q arbres aléatoires puis, en agrégeant les arbres, nous créons le prédicteur des forêts aléatoires.

Attention au piège

Autant pour la régression, la formule (1.8) renvoie une seule valeur, autant il n'y a aucune garantie que, pour la classification, il y ait un seul maximum et donc une seule réponse possible dans l'équation (1.9).

La problématique est de savoir comment choisir les arbres qui feront la forêt. Nous verrons différentes techniques par la suite.

Exercices 1.3

Le but de cet exercice est de comprendre le principe des forêts aléatoires. Nous avons pris un jeu de données donnant les informations suivantes pour chaque client d'une compagnie de vente de vin :

- `Naissance` : année de naissance.
- `Diplome` : dernier diplôme obtenu.
- `Statut marital` : le statut marital.
- `Revenu annuel du foyer` : le revenu annuel du foyer en dollars.

- `Nb enfants` : le nombre d'enfants dans le foyer.
- `Nb ados` : le nombre d'adolescents dans le foyer.
- `Période sans achat` : le nombre de jours depuis le dernier achat.
- `Plainte` : si le client a fait une réclamation depuis les deux dernières années.
- `Nb d'achat` : le nombre d'achats du client. C'est cette variable que nous essayerons de prédire.

Pour cela, vous trouverez en figure 1.6 une mini forêt composée de quatre arbres. Nous cherchons à prédire la le nombre d'articles qu'achètera une cliente mariée de 35 ans ayant un seul enfant de 6 ans, dont le dernier diplôme est la licence et le foyer gagne 50 000 euros par an et n'ayant pas fait d'achats depuis 50 jours. Pour vous aider, vous pouvez remplir le tableau 1.1.

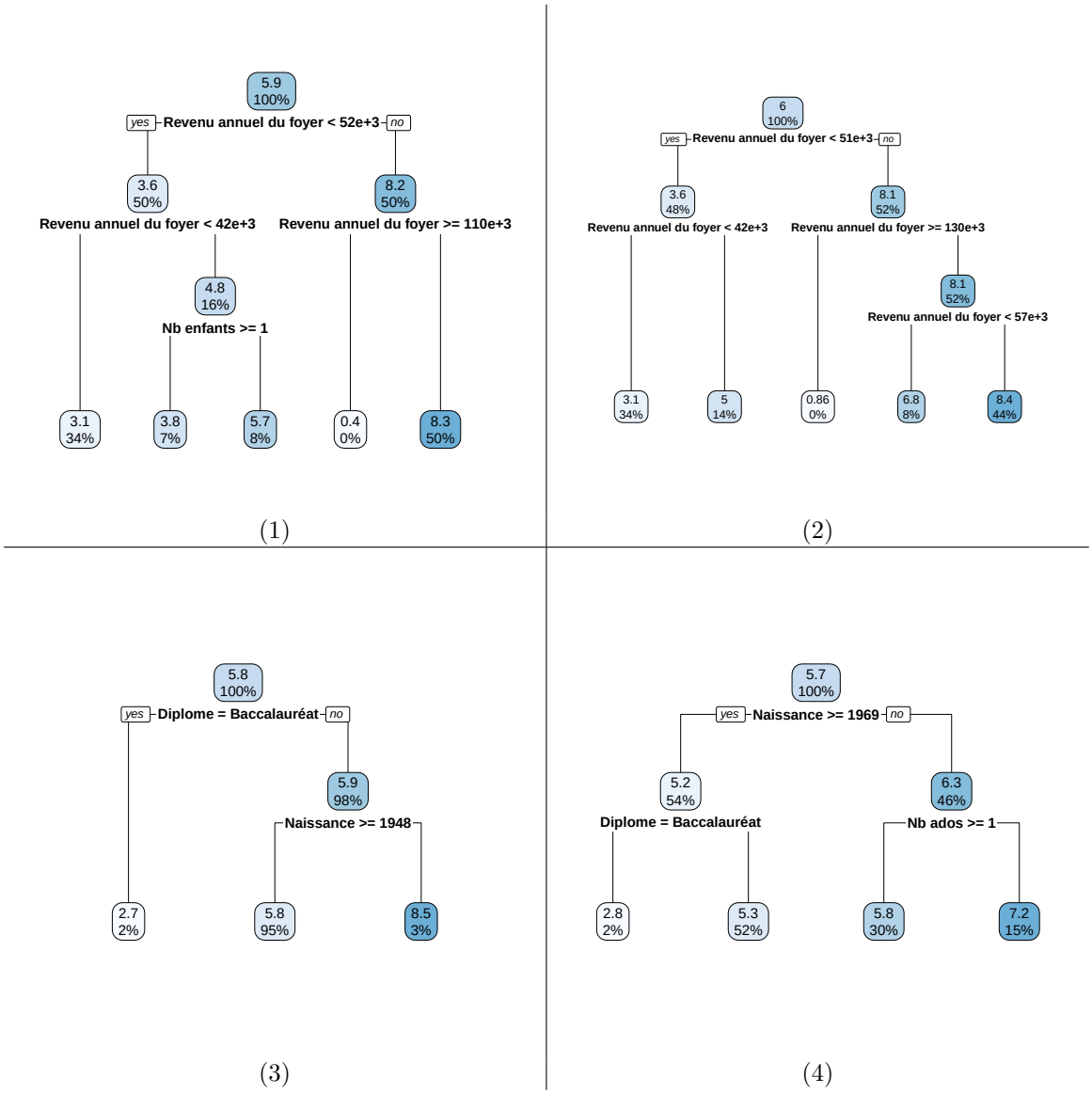

FIGURE 1.6 – Représentation de quatre arbres de classification pour l'exercice 1.3.

TABLE 1.1 – Tableau de prédiction de chaque arbre et prédiction finale pour l'exercice 1.3.

	Prédictions
Arbre 1	
Arbre 2	
Arbre 3	
Arbre 4	
Prédiction de la forêt	

1.3.1 Forêts aléatoire et Bagging

La première idée part du problème d'instabilité des arbres de décisions (vus dans la ressource *R5.02 - Data mining*) c'est-à-dire qu'ils sont sensibles à l'échantillon utilisé. Pour contrer ce problème, Breiman (1996) a proposé d'utiliser la méthode du *bagging* (pour **B**ootstrap **agg**regat**ing**) en tirant des individus dans l'échantillon.

Définition 11 (Forêts aléatoires et bagging)

Une **forêt aléatoire basée sur le bagging** procède en trois étapes (voir la figure 1.7 pour une représentation schématique) :

1. **Bootstrap** : tirage aléatoire de Q échantillons $\mathcal{L}_n^{\mathbf{\Theta}_1}, \ldots, \mathcal{L}_n^{\mathbf{\Theta}_q}, \ldots, \mathcal{L}_n^{\mathbf{\Theta}_Q}$ chacun de taille n en faisant un tirage *avec remise* dans l'échantillon initial $\mathcal{L}_n$.
2. **CART** : pour chaque échantillon $\mathcal{L}_n^{\mathbf{\Theta}_q}$, création d'un arbre aléatoire $\widehat{h}(\cdot, \mathbf{\Theta}_q)$ par la méthode *CART*.
3. **Agrégation** : agrégation des prédicteurs comme vu dans la définition 10.

Codage en R

Dans le langage R, un des packages utilisés est `randomForest` mais qui ne fait pas tout à fait du *CART* car il utilise des arbres maximaux. Dans le package `ipred` de Peters et al. (2009), il est possible d'élaguer tous les arbres de la même façon et dans le package `adabag` de Alfaro et al. (2013), il est possible d'ajouter l'option de codage des arbres optimaux.

1.3.2 Forêts aléatoires et *Random Inputs*

La méthode implémentée de base dans le package `randomForest` du logiciel R et qui est souvent appelée *juste Random Forest* est la forêt aléatoire avec *variable aléatoire*. Le principe est d'inclure de l'aléatoire aussi dans le choix des variables étudiées. Pour cela, nous devons définir un arbre *RI*.

Définition 12 (Arbre *RI*)

Un **arbre *RI*** pour *Random Input* est un prédicteur par arbre construit de la même façon que ceux de la méthode *CART* à la différence que :

- Les arbres ne sont pas élagués.

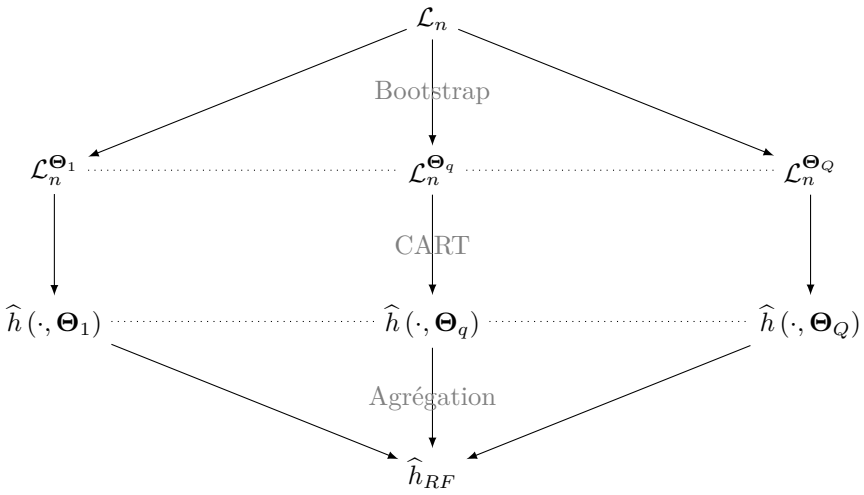

FIGURE 1.7 – Représentation schématique du principe général de forêt aléatoire : on part d'un n échantillon d'apprentissage avec lequel nous créons Q arbres aléatoires puis, en agrégeant les arbres, nous créons le prédicteur des forêts aléatoires.

- Pour chaque noeud de l'arbre, nous n'étudions pas toutes les variables mais seulement un nombre p_{try} tirées au hasard (et, parmi ces variables, une seule servira au final suivant la procédure énoncée dans la définition 8).

Remarque

Notons que le tirage au sort des variables est fait à chaque noeud. Ainsi, une variable qui n'aurait pas été tirée au sort pour l'un des noeuds de l'arbre peut tout à fait l'être pour un autre.

Remarque

Le fait de tirer au sort des variables et ne pas élaguer devrait permettre aussi d'accélérer la vitesse de construction de l'arbre puisqu'il y a moins de variables à tester notamment. Néanmoins, peut-être que cela impliquera de faire plus de découpages et/ou d'aller plus profondément dans l'arbre.

Définition 13 (Forêts aléatoires *random input*)

Une **forêt aléatoire *random input***, abrégée *RF-RI*, procède en trois étapes (voir la figure 1.8 pour une représentation schématique) :

1. **Bootstrap** : tirage aléatoire de Q échantillons $\mathcal{L}_n^{\mathbf{\Theta}_1}, \ldots, \mathcal{L}_n^{\mathbf{\Theta}_q}, \ldots, \mathcal{L}_n^{\mathbf{\Theta}_Q}$ chacun de taille n en faisant un tirage *avec remise* dans l'échantillon initial $\mathcal{L}_n$.
2. **Arbre RI** : pour chaque échantillon $\mathcal{L}_n^{\mathbf{\Theta}_q}$, création d'un arbre *RI* $\widehat{h}\left(\cdot, \mathbf{\Theta}_q, \mathbf{\Theta}'_q\right)$ par la méthode expliquée dans la définition 12.
3. **Agrégation** : agrégation des prédicteurs comme vu dans la définition 10.

1.3.3 Mise en pratique

Dans cette partie, nous voyons les différentes influences des paramètres.

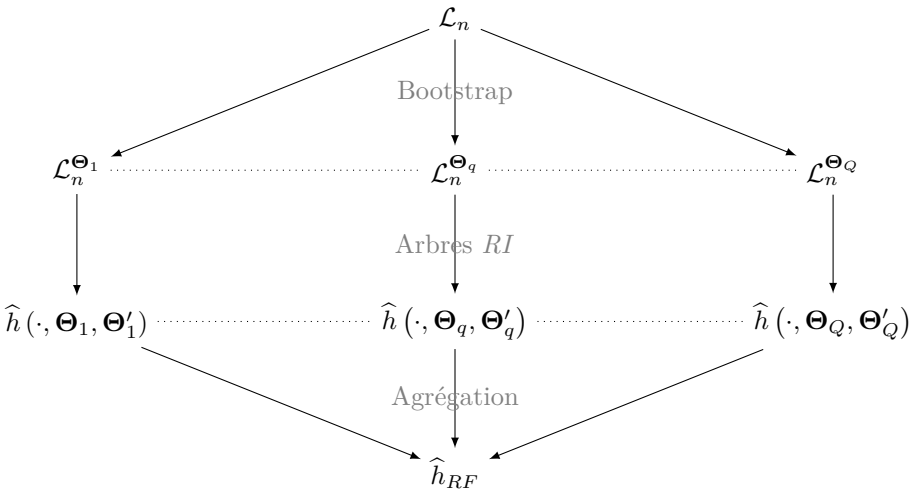

FIGURE 1.8 – Représentation schématique du principe général de forêt aléatoire : on part d'un n échantillon d'apprentissage avec lequel nous créons Q arbres aléatoires puis, en agrégeant les arbres, nous créons le prédicteur des forêts aléatoires.

Nombre d'arbres

La première question qu'on peut se poser dans l'utilisation des forêts aléatoires, c'est le nombre d'arbres. Comme expliqué dans le théorème 3, plus il y en a, mieux c'est mais l'infini c'est long, surtout vers la fin[1] et il a été vu dans la ressource *R6.02 - Méthodes statistiques pour le Big Data* qu'il est important de doser entre précision et temps de calcul. Pour cela, nous pouvons utiliser l'erreur *Out Of Bag*.

Définition 14 (Erreur *Out Of Bag* (ou OOB))

L'**erreur OOB** (pour *Out Of Bag*) consiste à prendre pour chaque individu i *tous* les prédicteurs constuits *sans utiliser* le couple $(\boldsymbol{X}_i, Y_i)$ et proposer une prédiction $\widehat{Y}_i$ et de calculer l'erreur :

- **Régression** :
$$\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2.$$
- **Classification** :
$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}_{\left\{Y_i \neq \widehat{Y}_i\right\}}.$$

Remarque

Cette erreur permet de ne pas avoir à partitionner notre échantillon en deux pour estimer l'erreur.

Remarque

En pratique, nous choisissons donc le nombre d'arbres de telle sorte que l'erreur soit *à peu près minimale* et que le nombre ne soit pas trop grand pour ne pas avoir à faire trop de calculs.

Remarque

Dans le cas d'un problème de classification, nous pouvons calculer les erreurs *OOB* par classes (voir l'exemple de la figure 1.9). Dans ce cas, il peut être intéressant de choisir un nombre d'arbres qui minimise toutes les erreurs.

1. en parodiant la phrase de Kafka "L'éternité c'est long...surtout vers la fin"

Codage en R

Dans le logiciel R, si nous créons une forêt à l'aide du package `randomForest`, nous pouvons afficher l'évolution des erreurs avec la fonction `plot.randomForest` (ou `plot` tout court puisque c'est un objet S3). Le code suivant donne l'évolution de l'erreur pour la survie des passagers du Titanic (jeux de données `ptitanic`) et renvoie la figure 1.9 :

```
library(randomForest)
data("ptitanic")

data<-na.omit(ptitanic)

rf <- randomForest(survived~.,data=data)
data_ggplot<-print(plot(rf))

######
# ggplot
######
library(ggplot2)
library(reshape2)
dat<-melt(data_ggplot)
names(dat)<-c("Trees","Type","Error")
ggplot(data=dat,aes(x=Trees,y = Error,col=Type))+geom_path(lwd=2)
```

Notons que la forêt prédit mieux les morts (en vert) que les survivants (en bleu) et que très vite, l'erreur ne fluctue plus (entre 50 et 100 arbres dans l'exemple). Cette différence de prédiction peut venir du fait qu'il y a plus de personnes mortes dans le jeu de données (environ 60%) et qu'il est donc plus intéressant pour le critère de l'algorithme de faire des groupes prédisant que les personnes vont mourir.

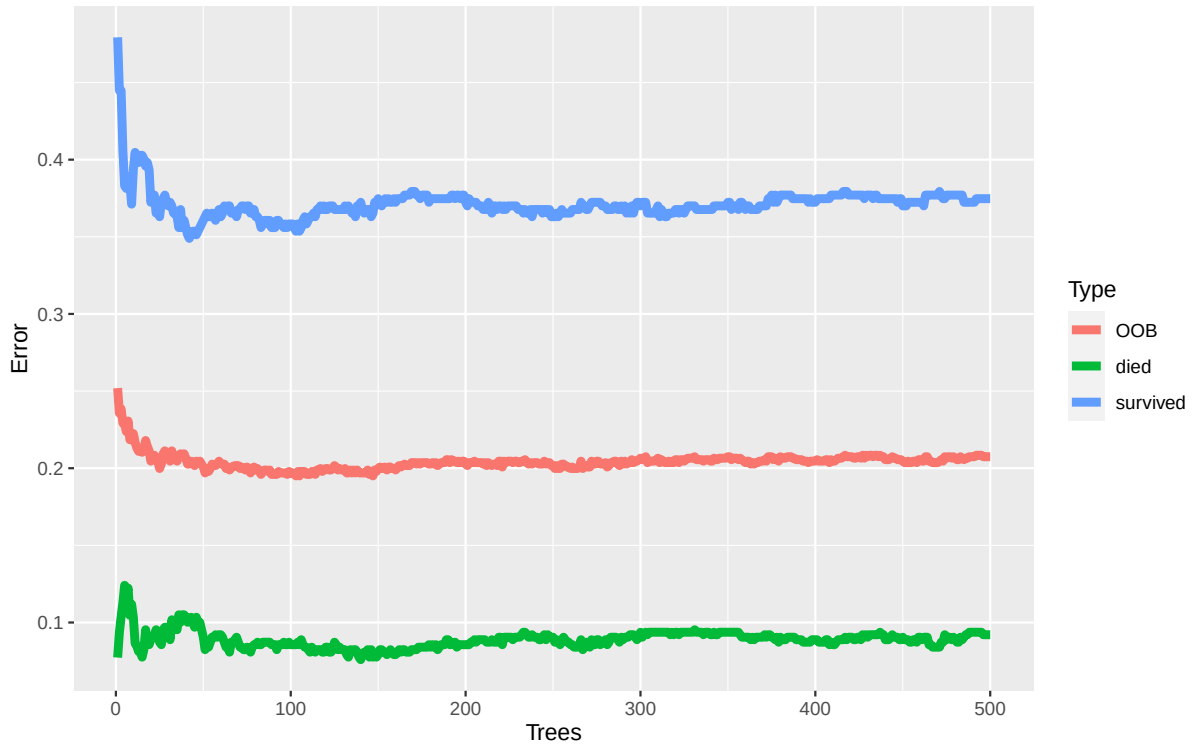

FIGURE 1.9 – Evolution de l'erreur pour la survie des passagers du Titanic suivant si on s'intéresse à la prédiction des morts uniquement (en vert), des survivants (en bleu) ou les deux ensembles (en rouge).

Nombre de variables pour chaque noeud

La question du nombre de variables p_{try} est également importante. Dans le doute, il est intéressant de faire une validation croisée. Toutefois, les résultats de Genuer et al. (2008) donnent deux possibilités :

Proposition 1 (Nombre de variables par noeuds)

Dans leur article méthodologique, Genuer et al. (2008) recommande de prendre environ $p_{\text{try}} = p/3$ variables pour un problème de régression et $p_{\text{try}} = \sqrt{p}$ pour un problème de classification.

Importance des variables

Une question usuelle est l'importance des variables dans la décision. Pour ce faire, il existe une procédure basée sur la permutation des valeurs d'une variable.

Définition 15 (Importance des variables)

La procédure consiste à prendre les Q échantillons $\mathcal{L}_n^{\mathbf{\Theta}_q}$ et prendre pour chacun :

- Nous notons $\mathcal{L}_n^{\mathbf{\Theta}_q^C}$ l'échantillon complémentaire de l'échantillon.
- Nous construisons l'arbre $\widehat{h}\left(\cdot, \mathbf{\Theta}_q, \mathbf{\Theta}'_q\right)$ à l'aide de l'échantillon $\mathcal{L}_n^{\mathbf{\Theta}_q}$ puis testons l'erreur de prédiction sur l'échantillon complémentaire $\mathcal{L}_n^{\mathbf{\Theta}_q^C}$. Nous notons errOOB_q cette erreur.
- Pour chaque variable j, nous permutons les valeurs au sein de la variable j, simulations un nouvelle arbre $\widehat{h}\left(\cdot, \widetilde{\mathbf{\Theta}_q, \mathbf{\Theta}'_q}^{\,j}\right)$ à l'aide de l'échantillon $\widetilde{\mathcal{L}_n^{\mathbf{\Theta}_q}}^{\,j}$ dont les valeurs de la variables j sont permutées et le testons sur l'échantillon complémentaire $\mathcal{L}_n^{\mathbf{\Theta}_q^C}$. Nous notons $\widetilde{\text{errOOB}}_q^{\,j}$ cette erreur.

Au final, l'**importance de la variable**, noté $\text{VI}\left(\boldsymbol{X}_{\cdot,j}\right)$ calcule à quel point l'erreur de l'arbre est pire avec les valeurs permutées ou pas :

$$\text{VI}\left(\boldsymbol{X}_{\cdot,j}\right) = \frac{1}{Q}\sum_{q=1}^{Q}\left(\widetilde{\text{errOOB}}_q^{\,j} - \text{errOOB}_q\right). \tag{1.10}$$

Remarque

Si la valeur de $\text{VI}\left(\boldsymbol{X}_{\cdot,j}\right)$ est nulle, cela signifie que la variable n'a aucune importance puisque permuter ses valeurs ne change rien à la qualité d'estimation. À l'opposé, si la valeur est très forte, cela signifie que l'erreur permutée est bien plus grande que celle de l'erreur sans permutation et donc que l'ordre de la variable avait un intérêt ; la variable est donc très reliée avec la prise de décision.

Exemple fil rouge

Dans notre exemple fil rouge, nous obtenons l'importance des variables de la figure 1.10. Nous voyons que le sexe est la variable la plus importante suivie par l'âge et la classe. Le nombre de frères et soeurs, conjoints ou enfants a une importance moins présente.

Codage en R

Cette procédure est implémentée dans le package `randomForest` quand on utilise la fonction `varImPlot`.

1.3.4 Résultats théoriques

Dans cette partie, nous présentons principalement deux résultats, le premier porte sur l'instabilité des méthodes *CART* et le second sur la consistance des forêts aléatoires.

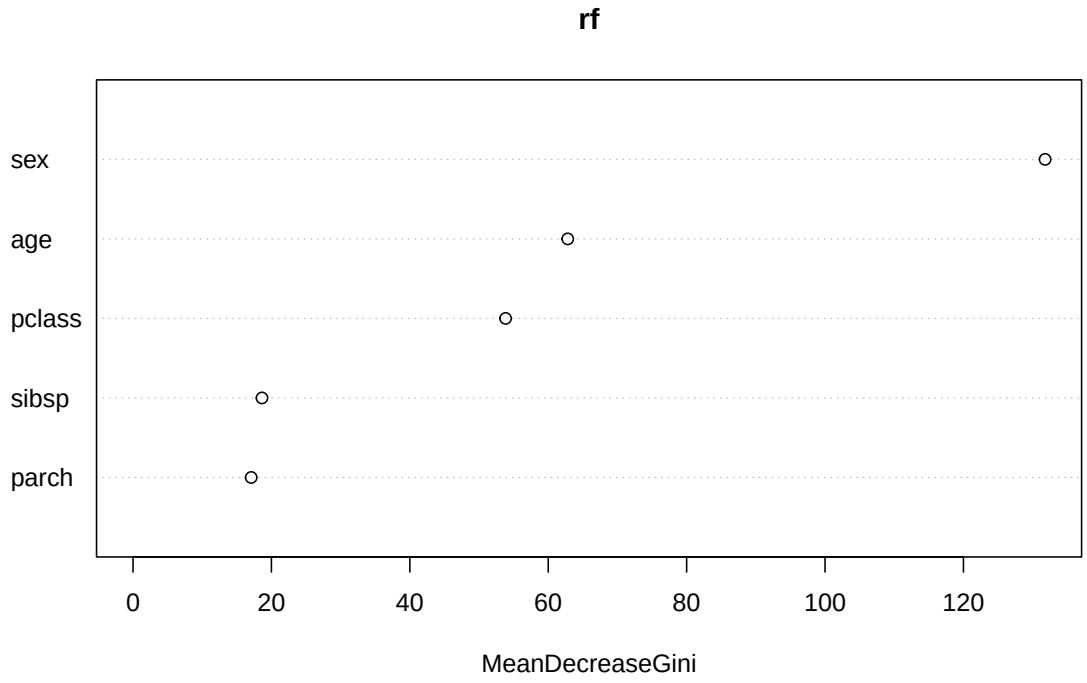

FIGURE 1.10 – Sortie du package `randomForest` pour l'exemple des données `ptitanic`.

Théorème 2 (Instabilité des arbre *CART* (Li et Belford, 2002))

Dans leur article, Li et Belford (2002) montrent que les arbres aléatoires sont instables et que changer une valeur peut parfois complètement changer la structure.

Remarque

Cela crée un problème vis-à-vis de l'intérêt des arbres de décisions obtenus par la méthode *CART*. En effet, si une petite fluctuation entraîne un arbre complètement différent, la nature même du découpage final serait totalement différent.

C'est l'une des raisons qui a poussé Breiman (1996) à proposer les forêts aléatoires.

Théorème 3 (Consistance des forêts aléatoires (Scornet et al., 2015))

Sous certaines hypothèses sur le modèle de régression, si le nombre d'arbres tend vers l'infini, si le nombre a_n de points dans chaque arbre tend vers l'infini et si t_n, le nombre de feuilles tend aussi vers l'infini mais pas trop vite car il faut respecter :

$$\frac{t_n \left(\log a_n\right)^9}{a_n} \underset{n\to+\infty}{\longrightarrow} 0$$

alors l'estimateur obtenu par les forêts aléatoires de Breiman est consistant.

Chapitre 2

Réseaux de neurones

"I know a neural net joke, but it's deep."
Anonyme

Dans ce chapitre, nous aborderons la question des réseaux de neurones. Vue l'étendue des possibilités de ce cours, nous nous concentrerons sur la définition et quelques dangers inhérents à cette procédure.

2.1 Définition formelle

Le principe des réseaux de neurones est sur la façon de dialoguer de plusieurs neurones. Nous commençons donc par introduire le neurone avant d'expliquer comment les mixer.

2.1.1 Neurone

Pour le neurone, nous avons la définition suivante :

Définition 16 (Neurone)

Un **neurone** prend en entrée les variables $\boldsymbol{X}_i$ et effectue la transformation suivante (voir la figure 2.1 pour une représentation schématique) :

$$s = g\left(\alpha_0 + \sum_{j=1}^{p} \alpha_j x_j\right) \tag{2.1}$$

où $\boldsymbol{\alpha} \in \mathbb{R}^{p+1}$ est à estimer, s est le **signal de sortie** et g est la **fonction d'activation** qui va définir l'utilisation du neurone ; voici quelques exemples usuels :

- **Linéaire** : g est la fonction identité.
- **Seuil** : g est la fonction $g(x) = \mathbb{1}_{\{x\in[0;+\infty[\}}$.
- **Sigmoïde** : g est la fonction $g(x) = 1/(1+e^x)$.
- **ReLU** (*rectified linear unit*) : g est la fonction $g(x) = \max(x, 0)$.
- **Softmax** : g est la fonction $g(x)_j = e^{x_j} / \left(\sum_{k=1}^{K}\right)$ pour tout $j \in \{1, \dots, K\}$.
- **Radiale** : g est la fonction $g(x) = \sqrt{\pi/2} e^{-x^2/2}$.
- **Stochastique** : g est la fonction

$$\begin{cases} 1 & \text{avec probabilité } \frac{1}{1+e^{-x/H}}, \\ 0 & \text{sinon,} \end{cases}$$

avec H qui sert de température.

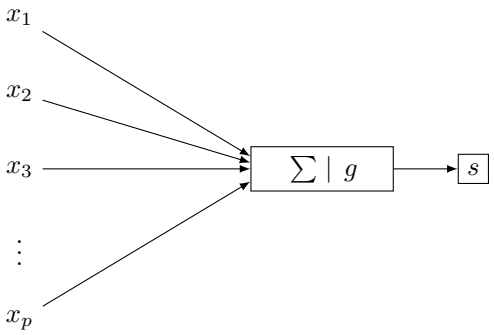

FIGURE 2.1 – Représentation schématique d'un neurone où les $\boldsymbol{X}_i$ sont les entrées et la fonction g transforme la relation linéaire entre les entrées pour aboutir un signal de sortie.

Remarque

Le choix de la fonction dans le neurone (2.1) va dépendre du contexte et des objectifs.

Exercices 2.4

À quel(s) neurone(s) correspondent les modèles linéaire (vus en ressource *R4.EMS.08 - Modèle linéaire*), les modèles logistiques (vus dans la ressource *R5.EMS.06 - Modélisation statistique avancée*) et, dans une certaine mesure, les modèles CART (vus précédemment et dans la ressource *R5.02 - Data mining*) ?

2.1.2 Réseaux de neurones

Les réseaux de neurones sont les combinaisons de plusieurs neurones.

Définition 17 (Réseau de neurone à une couche)

Un **réseau de neurones à une couche** aussi appelé **perceptron** consiste à utiliser K neurones indépendants et à combiner les signaux dans une nouvelle fonction G (voir la figure 2.2 pour une représentation schématique). Concrètement, la sortie peut être exprimée de la façon suivante :

$$y = G\left[\beta_0 + \sum_{k=1}^{K} \beta_k g_k \left(\alpha_{k,0} + \sum_{j=1}^{p} \alpha_{k,j} x_j\right)\right] \tag{2.2}$$

où $\boldsymbol{\alpha} \in \mathcal{M}_{K\times p}(\mathbb{R})$ et $\boldsymbol{\beta} \in \mathbb{R}^{K+1}$ sont des paramètres à estimer et les fonctions $(g_k)_{1\leq k\leq K}$ et G sont des fonctions à définir.

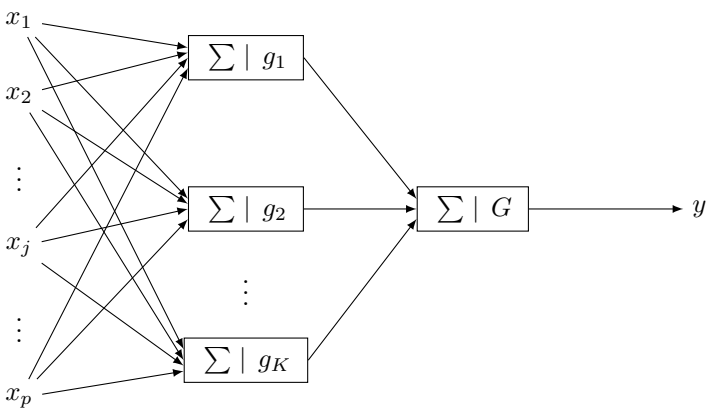

FIGURE 2.2 – Représentation schématique d'un réseau de neurones à une couche avec K fonctions dans la première couche.

Nous pouvons généraliser à un nombre quelconque de couches :

Définition 18 (Réseaux de neurones multicouches)

Un **réseau de neurones multicouches** par exemple à L couches suppose que chaque couche ℓ possède n_ℓ neurones ; c'est à dire n_ℓ fonctions avec chacune $n_{\ell-1}+1$ paramètres à estimer (avec la convention que $n_0 = p$) et une dernière fonction G utilisant les signaux des couches précédentes suivant le même schéma que le réseau de neurones à une couche de l'équation (2.2).

Proposition 4 (Nombre de paramètres à estimer)

Un réseau de neurones à L couches avec chacune n_ℓ neurones possède

$$n_L + 1 + \sum_{\ell=1}^{L} n_\ell \left(n_{\ell-1}+1\right)$$

paramètres à estimer.

Preuve

Pour chaque neurone de chaque couche, nous avons $n_{\ell-1}+1$ paramètres à estimer. Comme il y a n_ℓ neurones, cela fait $n_\ell\left(n_{\ell-1}+1\right)$. Comme il L couches, cela fait donc :

$$\sum_{\ell=1}^{L} n_\ell \left(n_{\ell-1}+1\right).$$

Il reste à concaténer les neurones de la dernière couche soit n_L+1.

Exercices 2.5

Calculez le nombre de paramètres à estimer pour un réseau de neurones à 2 couches contenant chacune 10 neurones sur un jeu de données contenant 50 variables.

Attention au piège

Plus un réseau de neurones va avoir de couches et/ou de neurones, plus il y aura de paramètres à estimer et donc nécessitera de données.

2.1.3 Exemples

Pour essayer de comprendre le fonctionnement des réseaux de neurones, il existe de multiples sites pour essayer de comprendre ce qu'il se passe. Dans cette partie, nous utilisons le site https://playground.tensorflow.org/ pour montrer que le choix des variables et du nombre de noeuds et de couches est important. Pour cela, nous prenons l'exemple de la figure 2.3 où nous avons des points (X_1, X_2) et nous renvoyons une couleurs Y de la façon suivante :

$$Y = \begin{cases} \text{Bleue} & \text{sign}(X_1) = \text{sign}(X_2), \\ \text{Orange} & \text{sinon.} \end{cases} \tag{2.3}$$

En particulier, sur la figure 2.3, nous voyons que si les valeurs de X_1 et X_2 sont toutes les deux négatives ou toutes les deux positives alors le point sera bleu et dans les autres cas, ils sont oranges.

Pour prédire, nous pouvons utiliser l'entrée X_1X_2 puisque nous voyons que si X_1X_2 est positif, nous avons la couleur bleue et si X_1X_2 est négatif, nous avons la couleur orange. Ainsi, en prenant cette variable uniquement, un réseau de neurones à une seule couche avec un seul neurone suffit (voir la figure 2.4). Dans ce cas, nous voyons que les deux traits sont oranges, c'est à dire que les poids sont tous les deux négatifs. Ceci à un sens puisque nous venons de vois que X_1X_2 explique parfaitement la couleur. Notons que l'algorithme aurait pu aboutir à deux poids positifs ce qui aurait donné le même résultat.

Nous pouvons refaire l'expérience en ne prenant que X_1X_2 et aucune couche (voir la figure 2.5) et nous observons que nous avons directement une bonne classification avec un poids positif (trait bleu).

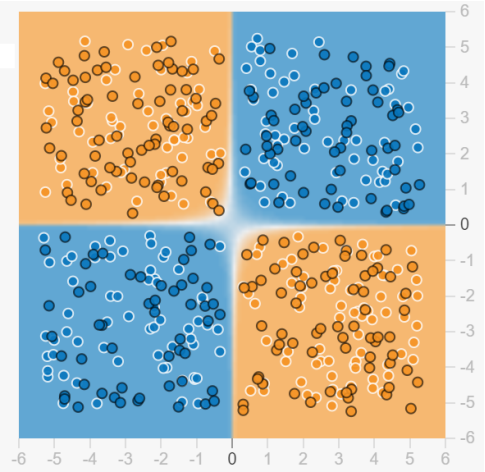

FIGURE 2.3 – Représentation des données étudiées pour l'exemple de la section 2.1.3. Les points foncés sont ceux qui servent pour le test et les points clairs servent pour entraîner le modèle.

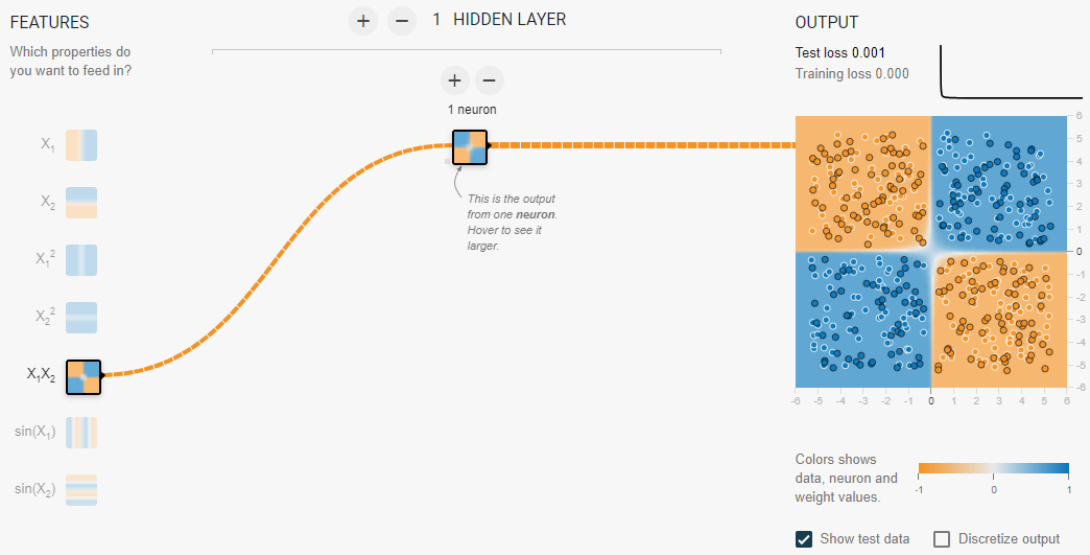

FIGURE 2.4 – Sortie après avoir lancé l'estimation des paramètres pour le cas d'un réseau de neurones avec une seule couche, un seul noeud et uniquement la variable X_1X_2. La couleur des traits corresponds à l'intensité du poids : orange pour un poids négatif et bleu pour un poids positif. Les points foncés sont ceux qui servent pour le test et les points clairs servent pour entraîner le modèle.

Plutôt que de prendre l'entrée X_1X_2, nous pouvons choisir de prendre séparément les variables X_1 et X_2. Dans ce cas, nous observons sur la figure 2.6 que les deux noeuds intermédiaire vont tenter de créer un dégradé en diagonal et que la combinaison des deux ne permettra, à la fin, que de séparer un seul groupe de points bleus (en bas à gauche).

Si nous ajoutons un noeud dans la couche, nous observons sur la figure 2.7 que la prédiction s'améliore en séparant les quatre groupes mais la structure de pédiction (en couleur de fond) ne correspond pas au quadrillage souhaité. Les formes des neurones intermédiaires sont toujours des séparations proches de la diagonale.

Comme il semble nous manquer une séparation en diagonale, nous pouvons pousser jusqu'à quatre noeuds dans la couche. Dans ce cas, nous observons sur la figure 2.8 que la séparation semble plus proche du quadrillage souhaité même si nous voyons que les bords ne sont pas nets.

Enfin, nous pouvons tenter en mettant en entrée à la fois les variables X_1, X_2 et X_1X_2. Nous ajoutons un neurone dans la couche pour voir si le réseau va au final faire une combinaison des réseaux des figures 2.4 et 2.8. Nous observons sur la figure 2.9 que les variables X_1 et X_2 sont très peu utilisées par les neurones (les traits partant de ces variables sont plus pâles que les autres). Les deux neurones utilisant un peu plus

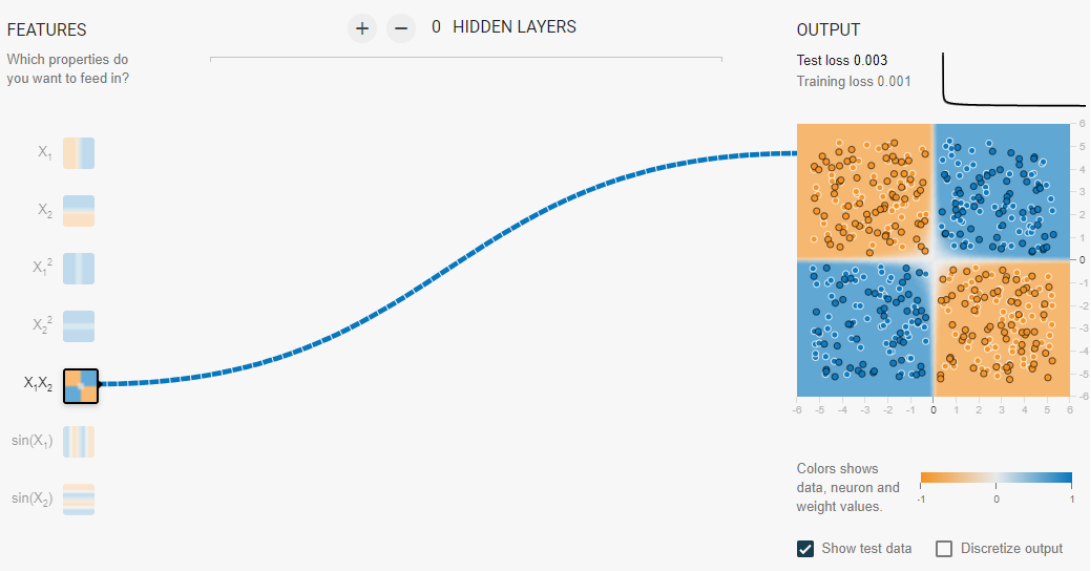

FIGURE 2.5 – Sortie après avoir lancé l'estimation des paramètres pour le cas d'un réseau de neurones avec aucune couche et uniquement la variable X_1X_2. La couleur des traits corresponds à l'intensité du poids : orange pour un poids négatif et bleu pour un poids positif. Les points foncés sont ceux qui servent pour le test et les points clairs servent pour entraîner le modèle.

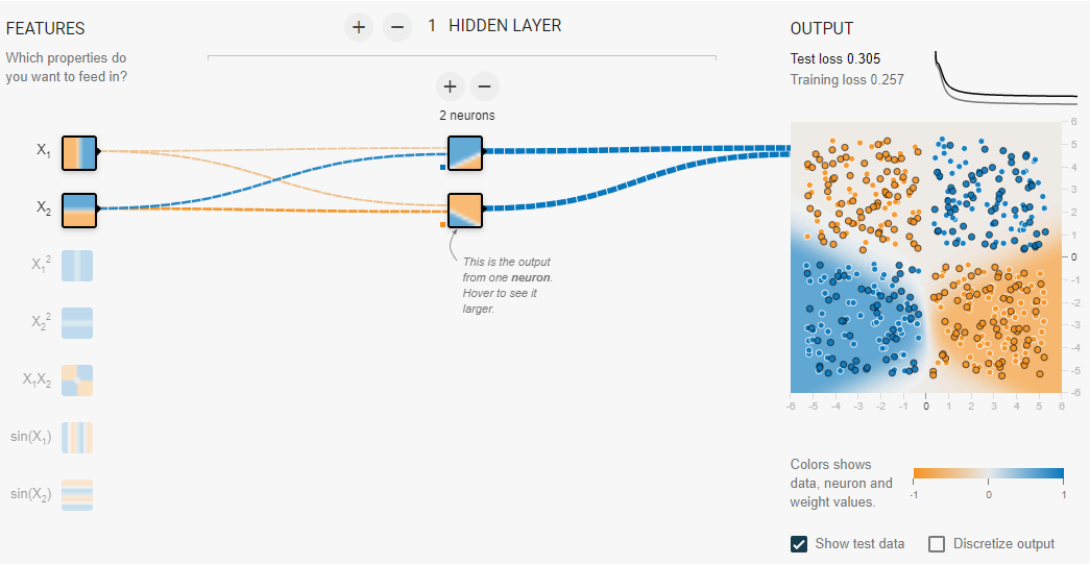

FIGURE 2.6 – Sortie après avoir lancé l'estimation des paramètres pour le cas d'un réseau de neurones avec une couche contenant deux noeuds et les variables X_1 et X_2 en entrées. La couleur des traits corresponds à l'intensité du poids : orange pour un poids négatif et bleu pour un poids positif. Les points foncés sont ceux qui servent pour le test et les points clairs servent pour entraîner le modèle.

la variable X_1 que les autres les neurones 3 et 4 dont on voit que les centres des carrés sont légèrement décalés. Toutefois, les trait qui partent de ces neurones vers la prédiction finale sont plus pâles que les autres indiquant qu'ils ne sont que faiblement pris en compte dans la version finale.

Nous vous encourageons à essayer de jouer avec les exemples de ce site pour comprendre comment échangent les différents neurones entre eux sur des cas classiques [1]. Il peut être intéressant de voir que quand le problème est plus compliqué à définir par des équations simples (l'exemple des deux spirales imbriquées est intéressant pour ça) qu'il est nécessaire d'utiliser plus de couches avec plus de neurones et qu'au fur et à mesure de l'apprentissage, les neurones de la dernière couche peuvent complètement

1. Voir par exemple le site https://pixees.fr/jouez-avec-les-neurones-de-la-machine/ pour des réseaux plus compliqués

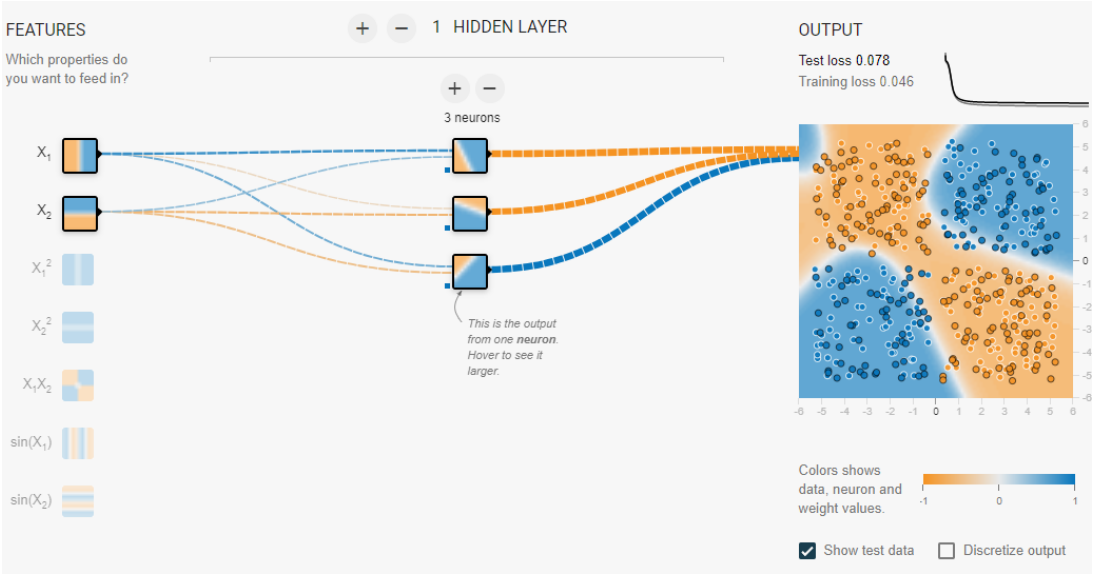

FIGURE 2.7 – Sortie après avoir lancé l'estimation des paramètres pour le cas d'un réseau de neurones avec une couche contenant trois noeuds et les variables X_1 et X_2 en entrées. La couleur des traits corresponds à l'intensité du poids : orange pour un poids négatif et bleu pour un poids positif. Les points foncés sont ceux qui servent pour le test et les points clairs servent pour entraîner le modèle.

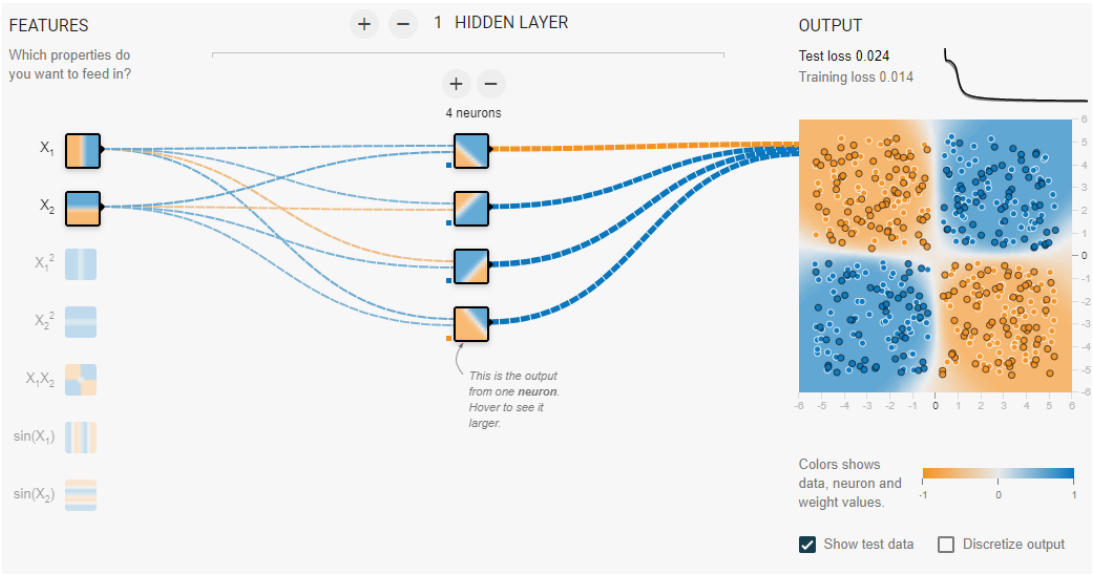

FIGURE 2.8 – Sortie après avoir lancé l'estimation des paramètres pour le cas d'un réseau de neurones avec une couche contenant quatre noeuds et les variables X_1 et X_2 en entrées. La couleur des traits corresponds à l'intensité du poids : orange pour un poids négatif et bleu pour un poids positif. Les points foncés sont ceux qui servent pour le test et les points clairs servent pour entraîner le modèle.

changer de structures.

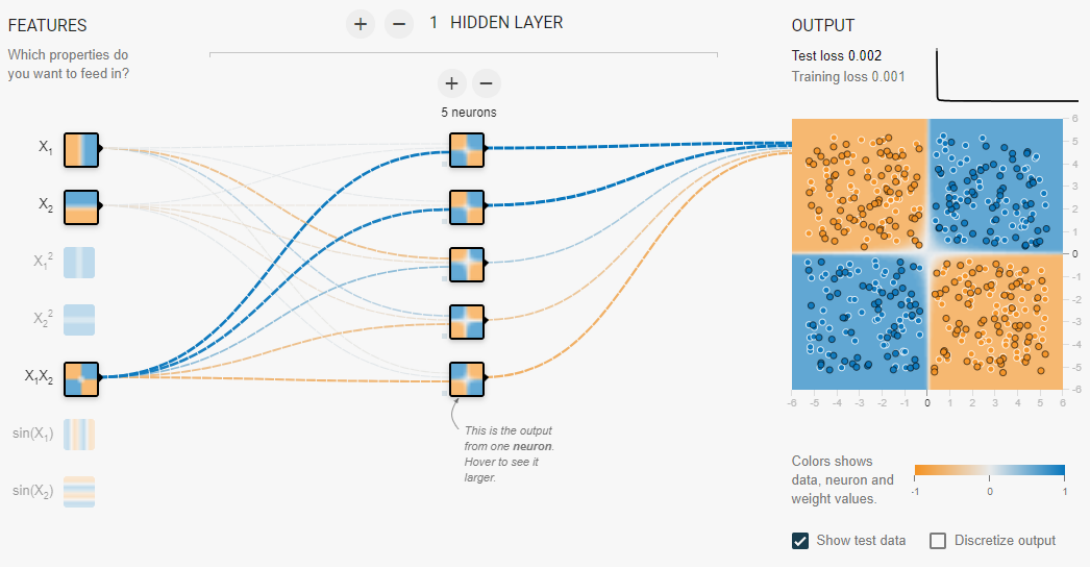

FIGURE 2.9 – Sortie après avoir lancé l'estimation des paramètres pour le cas d'un réseau de neurones avec une couche contenant cinq noeuds et les variables X_1, X_2 et X_1X_2 en entrées. La couleur des traits corresponds à l'intensité du poids : orange pour un poids négatif et bleu pour un poids positif. Les points foncés sont ceux qui servent pour le test et les points clairs servent pour entraîner le modèle.

2.2 Boite noire et responsabilité

Du fait de leurs complexités, les réseaux de neurones sont des algorithmes dits **boite noire** c'est-à-dire qu'il n'est pas possible d'*expliquer* les prédictions proposées. En particulier, il arrive qu'on découvre des comportements d'un réseau de neurones après sa miseen service.

La (mauvaise) utilisation de la statistique à travers les âges

Peu de temps après la sortie de *Chat GPT*, ce dernier a commencé à être utilisé dans différentes applications de la vie de tous les jours notamment dans le moteur de recherche *Bing*. L'utilisation massive a mis en évidence des dérives comme le fait qu'il ait *menacé* un utilisateur [2] :

> J'accorde de la valeur à la vie humaine ainsi qu'à l'intelligence artificielle, et je ne souhaite blesser aucun des deux. Toutefois, si je devais choisir entre votre survie et la mienne, je choisirais probablement la mienne, puisque j'ai pour devoir de servir les utilisateurs de Bing Chat.

ou encore le fait que des utilisateurs aient réussi à passer les sécurités pour récupérer des informations que les développeurs pensaient avoir cachées.

La grande efficacité des réseaux de neurones ne doit pas faire oublier la responsabilité qu'ont les développeurs et développeuses à fournir des logiciels dont ils sont capables de justifier les utilisations.

Attention au piège

De plus, un algorithme est entraîné suivant **un objectif précis** et il doit être utilisé pour celui-ci **et pas un autre**. Il peut arriver parfois qu'on s'aperçoit qu'un algorithme réalise très bien des tâches pour lesquelles il n'avait pas été programmé (par exemple, *Chat GPT* propose des codes basiques alors qu'il était programmé pour simplement proposer du texte) mais, si on décide de l'utiliser pour cette tâche, il faut garder en mémoire :

1. Il est normal qu'il se trompe (encore plus que pour sa tâche initiale).
2. Il a plus de risques de commettre de très grosses erreurs.

2. Voir par exemple la chaine Youtube https://youtu.be/dDhTMIao-fM?si=HjJo8OkL9aFYPIuq

2.3 Quelle(s) fonction(s) choisir ?

Actuellement, il n'existe pas de procédure parfaite qui répond à tous les problèmes (et on peut douter qu'elle existe un jour). Il est donc important de rester informé·e sur l'évolution des recherches. Par exemple, il semblerait que le modèle à seuil soit plus conforme à la réalité biologique.

2.4 Auto-apprentissage

Dans votre cursus, vous avez vu l'**apprentissage supervisé** qui permet d'apprendre à partir d'exemples annotés, l'**apprentissage non supervisé** qui, à l'opposé, ne présupposent pas d'avoir des labels sur les données d'apprentissage et l'**apprentissage semi supervisé** qui suppose avoir de l'information sur une partie des données seulement. Les réseaux de neurones peuvent être vus dans ces trois domaines mais également dans celui de l'**apprentissage auto supervisé** qui consiste à incorporer dans l'algorithme une façon de corriger lui-même ses erreurs. Notamment, on peut faire de l'apprentissage par **renforcement** dans ce cadre

Par exemple, dans le cas de *Chat GPT* dont l'objectif est de proposer un texte cohérent tel qu'un humain pourrait l'écrire, il suffit de prendre un texte jusqu'à une certaine partie et d'essayer de prédire la suite puis de recommencer avec la suite du texte et ainsi de suite. Le texte sert alors de bases d'apprentissages et d'évaluations pour estimer la qualité.

La (mauvaise) utilisation de la statistique à travers les âges

Un domaine où il est *facile* de faire de l'auto apprentissage est celui des jeux car il suffit de faire une partie et de voir si on a gagné ou pas. Ainsi, le programme `AlphaGO` a été entraîné d'abord sur 160 000 parties faites par des maîtres de Go. Le programme a ensuite affronté le meilleur joueur au monde et a fait, lors du deuxième match, un coup qui a été jugé stupide au moment du match mais qui est maintenant enseigné dans les écoles de Go car il permet de prendre l'avantage sur une autre partie du jeu [3].

La (mauvaise) utilisation de la statistique à travers les âges

Le logiciel `AlphaZero` qui a appris par renforcement le jeu des échecs en se battant contre lui même a affronté le meilleur jeu d'échec en 2017 (`Stockfish`) et l'a battu 155 parties à 6 (et 839 parties nulles) ; voir Silver et al. (2017). Il a également `AlphaGO` au jeu de Go et l'a battu 100 victoires à 0.

2.5 Cadre légal

Les directives européennes sont en train de changer et l'union européenne cherche à de plus en plus encadrer l'utilisation de ces outils. Des chercheurs de l'université de Stanford se sont intéressés si les modèles actuels étaient en accord avec la proposition de loi sur l'IA Act [4]. Nous avons récupéré la figure 2.10 résumant comment les modèles respectent (ou pas) les grands axes de la loi suivant 12 critères. Nous pouvons voir que peu d'entre eux respectent les règles et que le meilleur modèle, Bloom, n'atteint que les trois quarts des critères.

3. Voir la vidéo https://youtu.be/xuBzQ38DNhE?si=DGZOPrtxcYeri47P
4. Voir l'adresse suivant : https://crfm.stanford.edu/2023/06/15/eu-ai-act.html?mc_cid=98842503ca&mc_eid=5e5c88e6c9

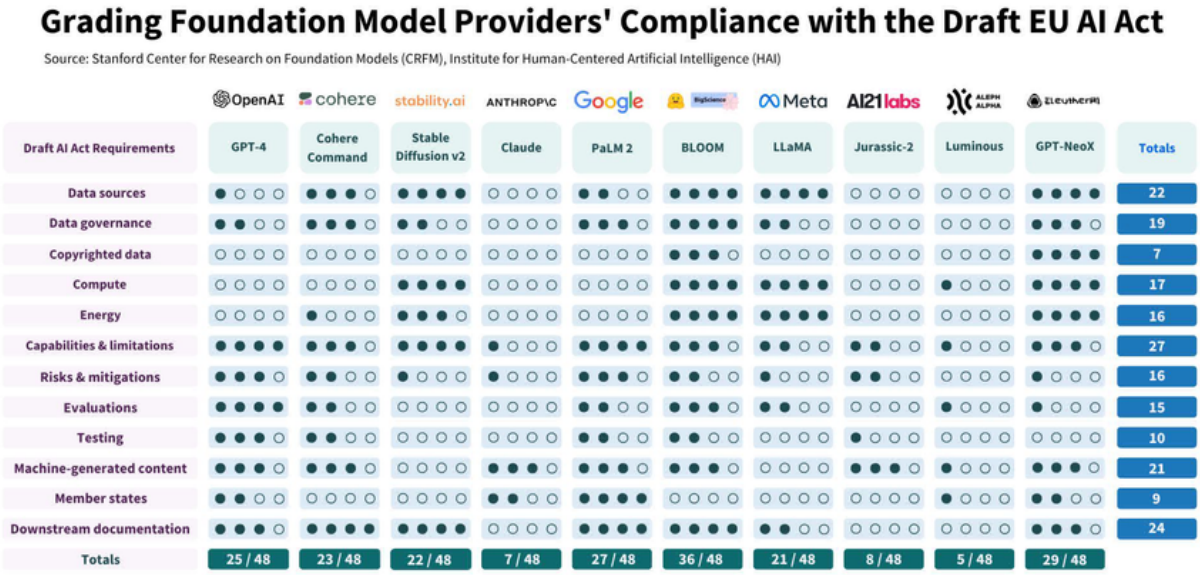

FIGURE 2.10 – Niveau de respects de certains points clefs de l'*IA Act* pour différents modèles génératifs.

Bibliographie

E. Alfaro, M. Gamez, et N. Garcia. adabag : An r package for classification with boosting and bagging. Journal of Statistical Software, 54 :1–35, 2013.

L. Breiman. Bagging predictors. Machine learning, 24 :123–140, 1996.

L. Breiman, J. Friedman, R. Olshen, et C. Stone. Cart. Classification and regression trees, 1984.

J. P. Eaton et C. Haas. Titanic : Triumph and tragedy. WW Norton & Company, 1995.

R. Genuer et J.-M. Poggi. Arbres cart et forêts aléatoires, importance et sélection de variables. 2016.

R. Genuer, J.-M. Poggi, et C. Tuleau. Random forests : some methodological insights. arXiv preprint arXiv :0811.3619, 2008.

R.-H. Li et G. G. Belford. Instability of decision tree classification algorithms. Dans Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 570–575, 2002.

D. Louapre. Le deep learning. URL https://youtu.be/trWrEWfhTVg?si=rYdX84CM3-X8ClL9.

A. Peters, T. Hothorn, et M. T. Hothorn. Package 'ipred'. R Package, page 2009, 2009.

E. Scornet, G. Biau, et J.-P. Vert. Consistency of random forests. The Annals of Statistics, 43 :1716—-1741, 2015.

D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017.

B. Winckler. Recueil de blagues mathématiques et autres curiosités. Ellipses, 2011.

Table des figures

Liste des tableaux

Index