MASTER MIAGE 2ème année
Université Paris Nanterre

Mémoire de fin d'études

Optimisation de parcours d'arbre de décisions dans le cadre du MNIST

Auteur :
Alexandre PETIT-PAS

Tuteur :
P. Marie-Pierre
GERVAIS

Résumé

Résumé

Remerciements

Remerciements

Sommaire

1 Concepts formels **7**

1.1 Introduction 7

1.2 Rappels de probabilité 7

1.3 Arbre de décision 8

1.4 Arbre de décision pour le classement 9

1.5 Arbre de décision pour l'optimisation 13

2 Lazy Decision Tree **17**

2.1 Lazy Decision Tree pour le classement 17

2.2 Lazy Decision Tree pour l'optimisation 17

3 Classement supervisé **19**

3.1 Introduction 19

3.2 K Nearest Neighbors 19

3.3 CART 21

3.4 ID3 21

3.5 C4.5 21

3.6 A * 21

4 Apprentissage du MNIST **23**

4.1 Définition du MNIST 23

4.2 Méthode de recherche 23

4.3 Algorithme de classement 23

4.4 Algorithme Lazy 23

4.5 Comparaison des résultats 23

Bibliographie **25**

Introduction

Motivations

J'ai choisi ce sujet suite à trois événements. Durant l'un de mes stages, j'ai pu travaillé avec un thésard en informatique qui faisait des recherches sur de l'apprentissage. C'est aussi durant ce stage que j'ai approché pour la première fois l'informatique décisionnelle. Ce sujet m'a rapidement intéressé étant donné la grandeur et la popularité de celui-ci. Cependant, je ne voulais pas aborder sous le thème de la recommandation (qui est populaire mais a été traité de nombreuses fois), c'est pourquoi je me suis intéressé en profondeur à ce qui se faisait en terme d'arbre de décision.

C'est en découvrant l'existence des *Lazy Decision Trees* que le sujet était choisi. Les algorithmes paresseux, ou *lazy*, sont beaucoup utilisés dans le domaine du Web pour leur économie en ressource. Les ORM utilisent beaucoup cette technique qui leur permet de minimiser les requêtes vers la base de données.

Par ailleurs, durant mon stage de troisième année de licence, j'ai eu l'occasion de surveiller des partiels de première année (L1). Leurs épreuves étaient sous forme de QCM avec certaines réponses libres. Toutes les questions fermées étaient corrigées automatiquement cependant les réponses libres ne l'étaient pas. Un autre problème qui m'a marqué était l'écriture du numéro étudiant, composé de chiffre. Celui-ci n'était pas récupéré automatiquement car l'outil ne pouvait pas reconnaître les chiffres écrits à la main.

Ce sont ces trois idées qui m'ont permis d'écrire ce mémoire, afin de comprendre le domaine de l'apprentissage supervisé en répondant à un besoin d'enseignants universitaires.

Problématique

La reconnaissance d'image est un domaine populaire qui est devenu mature avec le temps. Des problématiques de performance continuent d'exister de différentes manières : l'image reconnu est-elle la bonne et l'algorithme est-il suffisamment performant pour supporter la charge ? Dans certains contexte, l'erreur peut coûter très cher et dans d'autre, il n'est pas possible de reconnaître efficacement l'image. Nous allons nous intéresser à la reconnaissance de chiffres provenant du MNIST. Cependant, il est possible par extension de l'appliquer sur d'autres jeux de données. La problématique est de savoir si les solutions recherchés sont suffisamment efficaces pour être appliquée dans un cas concret de reconnaissance de numéros étudiants à des fins d'évaluation.

Objectifs

L'objectif de ce mémoire est de comprendre comment les techniques de classement fonctionnent et de les appliquer sur un ensemble de données connu. Les résultats permettront alors de connaître si un algorithme est plus performant qu'un autre et d'apporter une réflexion autour de l'amélioration de ceux-ci.

Chapitre 1

Concepts formels

1.1 Introduction

L'objectif de ce chapitre est dans un premier temps de rappeler des notions de probabilités [1, 2, 3] qui serviront à la compréhension de ce document. Il s'agit ensuite d'expliquer dans sa généralité les concepts d'arbres décisionnels afin d'introduire des principes de classement et d'optimisation.

1.2 Rappels de probabilité

1.2.1 Expérience aléatoire

Une expérience est dire aléatoire si les résultats possibles sont connus à l'avance sans vraiment savoir celui obtenu au préalable. On appelle univers, noté Ω, l'ensemble de toutes les issues possibles d'une expérience et ω une réalisation de l'expérience.

$$\Omega = \left\{\omega_1, \omega_2, \cdots, \omega_n\right\}, n \in \mathbb{N} \tag{1.1}$$

1.2.2 Événements

Un événement correspond à un ensemble de résultats possibles pour une expérience. Si cet ensemble est constitué d'un seul élément, on parle alors d'événement élémentaire. Si l'ensemble de résultats est égal à l'univers Ω, alors l'événement est dit certain. En revanche, si aucun résultat n'est présent (ensemble $\emptyset$), alors c'est un événement impossible.

1.2.3 Probabilités

Une probabilité est une fonction qui à un événement A, associe un poids.

$$\begin{cases} P(\Omega) = 1 \\ P(\emptyset) = 0 \\ 0 \leq P(A) \leq 1 \\ P(\bigcup_{i=1}^{n} A_i) = \sum_{i=1}^{n} P(A_i) \end{cases} \tag{1.2}$$

Plus la probabilité est proche de 1, plus il est possible que l'événement se réalise. Soit A et B deux événements quelconques, $P(A)$ est dite conditionnelle si son résultat est influencé par l'événement B :

$$\begin{cases} B \neq \emptyset \\ P(B) \neq 0 \\ A = (A \cap B) \cup (A \cap \overline{B}) \end{cases} \tag{1.3}$$

On peut alors déduire deux formules : le théorème de Bayes (1.4) permettant de calculer $P_B(A)$ et le théorème es probabilités totales (1.5) qui permet de connaitre la valeur de $P(A)$ à partir de deux événements A et B.

$$P_B(A) = \frac{P(A \cap B)}{P(B)} \tag{1.4}$$

$$P(A) = P_B(A)P(B) + P_{\overline{B}}(A)P(\overline{B}) \tag{1.5}$$

1.3 Arbre de décision

1.3.1 Définition

Un arbre de décision est une représentation structurelle qui permet d'aboutir à un choix. C'est un graphe acyclique orienté composé :

- d'un sommet sans parents appelé racine,
- de sommets appelés nœuds, correspondant à des tests,
- de sommets terminaux nommés feuilles,
- des arrêtes, ou branches, désignant chacune les résultats d'un test

Pour construire un arbre de décision, une approche *Top-Down* est utilisée, appelée *Top Down Induction of Decision Tree* (*TDIDT*). Elle peut se décomposer en plusieurs parties :

1. Partir du jeu complet de données et construire la racine.
2. Réaliser un test afin de séparer les données.
3. Séparer le nœud actuel en fonction des résultats possibles.
4. Appliquer récursivement jusqu'à atteindre les feuilles.

Lorsque chaque nœuds sont composés exactement de deux descendants (hors feuilles), on parle alors d'arbre de décision binaire. C'est un cas très largement utilisé des algorithmes de construction d'arbres tels que CART [4], qui sera expliqué ultérieurement.

1.3.2 Exemple

Prenons comme exemple les données météorologiques présente dans la table 1. Cette table est constitué de différentes colonnes concernant différentes information ainsi qu'une colonne indiquant si la décision de sortir a été prise ou non. Dans un peu premier temps, il est possible de déduire les tests à réaliser comme par exemple "La

température est-elle élevée ?" ou encore "Quel est le temps ?". A ces questions découlent des réponses possibles, {oui, non} pour la première et {soleil, nuageux, pluie} pour la deuxième. Une fois l'approche *TDIDT* utilisée, un arbre de décision est alors obtenu (figure 1.1).

Jour	Temps	Température	Humidité	Vent	Sortie
1	soleil	élevée	haute	faible	N
2	soleil	élevée	haute	fort	N
3	nuageux	élevée	haute	faible	Y
4	pluie	moyenne	haute	faible	Y
5	pluie	basse	normale	faible	Y
6	pluie	basse	normale	fort	N
7	nuageux	basse	normale	fort	Y
8	soleil	moyenne	haute	faible	N
9	soleil	basse	normale	faible	Y
10	pluie	moyenne	normale	faible	Y
11	soleil	moyenne	normale	fort	Y
12	nuageux	moyenne	haute	fort	Y
13	nuageux	élevée	normale	faible	Y
14	pluie	moyenne	haute	fort	N

Table 1.1 – Table Météo

1.4 Arbre de décision pour le classement

1.4.1 Attributs et classes

Soit un ensemble d'observation S de taille n :

$$S = \{s_1, s_2, ..., s_n\}. \tag{1.6}$$

Chaque observation s_i est composée d'un ensemble de m attributs :

$$A_s = \{a_1, a_2, ..., a_m\}. \tag{1.7}$$

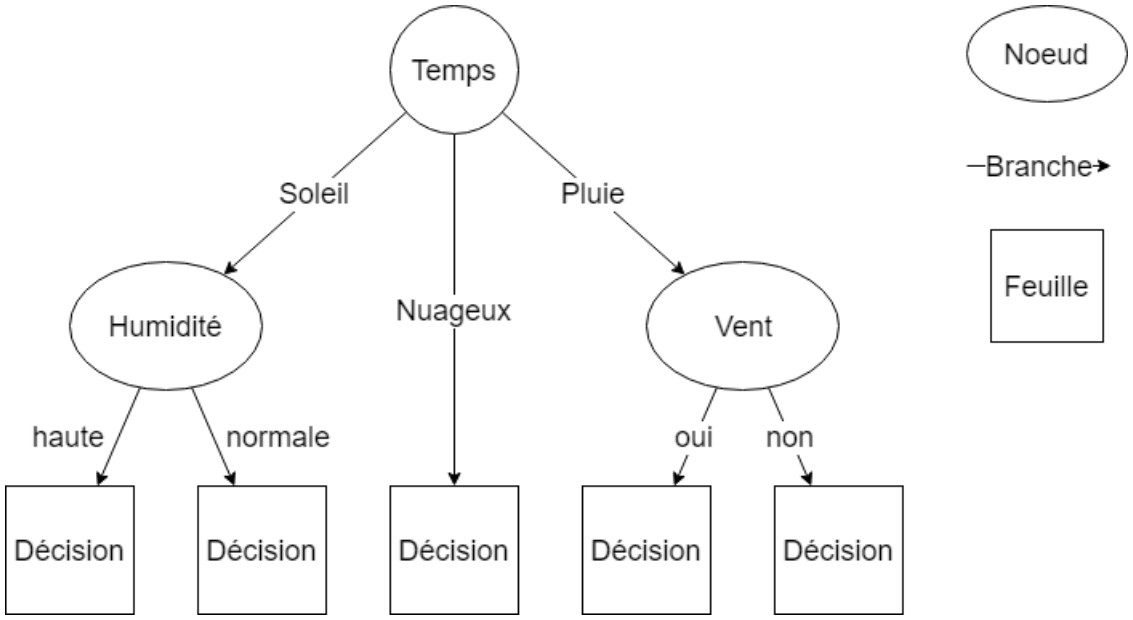

Figure 1.1 – Arbre de décision de la table Météo

Soit V_j l'ensemble de taille l des valeurs possibles de l'attribut a_j d'une observation s_i tels que :

$$V_j = \{v_1, v_2, ..., v_l\}. \tag{1.8}$$

La valeur de l'attribut a_j sera représentée par v_k. Un attribut est dit qualitatif si l'ensemble des valeurs possibles est symbolique (non numérique), par exemple si V_j représente les couleurs d'écriture d'un mot. On obtiendrait alors $V_j = \{$ bleu, rouge, noir, vert $\}$.

Une donnée est quantitative si l'ensemble des valeurs possibles est un ensemble numérique fini ou infini :

- Si un attribut peut prendre une infinité de valeurs dans son ensemble, alors celui-ci est qualifié de continu, par exemple le temps d'exécution d'un processus.
- Dans le cas contraire, une variable dite discrète possède une valeur finie. Elle est généralement liée à une énumération, comme par exemple le nombre de trait dans un caractère.

a_j est nominal si la notion d'ordre n'est pas présente dans l'ensemble des valeurs possibles, par exemple si $V_j = \{$ un, deux, trois, quatre, cinq, six, sept, huit, neuf $\}$ représente le nom d'un chiffre en toutes lettres.

Un attribut est ordinal si la notion les valeurs possibles contiennent la notion d'ordre. Cela peut être par exemple l'appréciation d'un client : $V_j = \{$ mauvais, bon, très bon $\}$

Une variable est qualifiée de binaire si l'ensemble V_j des valeurs possibles est de taille $l = 2$

La classe d'une observation correspond a une "catégorie" et permet de se rapprocher ou de se différencier des autres observations. Elle correspond à une feuille dans un arbre décisionnel.

Reprenons la table 1.1 (Météo), l'attribut jour correspond à l'identifiant dans l'observation et donc n'est pas pris en compte. *Temps*, *Température*, *Humidité* et *Vent* sont des attributs qualitatifs dont deux d'entre eux binaires : $V_{Humidite} = \{haute, normale\}$ et $V_{Vent} = \{fort, faible\}$. Chaque observation possède une classe *Sortie* pouvant prendre Y ou N comme valeur.

1.4.2 Apprentissage supervisé

Le domaine de l'apprentissage peut se séparer en deux types : le supervisé et le non-supervisé.

Dans le cas de l'apprentissage non-supervisé, le but recherché est d'assembler les similitudes entre les observations. Une des méthodes les plus communes est le *clustering* [5].

Le second type d'apprentissage est dit supervisé. Dans ce cas, le but est d'apprendre du modèle pour arriver à déterminer la classe de l'observation. C'est dans ce domaine que se situe les arbres de décision. Les jeux de données sont alors séparés en deux : une partie jeu d'entrainement (*training set*) et une partie jeu de test (*test set*). Le but est d'apprendre du jeu d'entrainement afin de classer les observations du jeu de test. Soit C un ensemble de classe de taille n :

$$C = \{c_1, \cdots, c_n\} \tag{1.9}$$

Classer une observation s_i revient à trouver le c_j qui lui correspond le mieux par le biais d'une fonction de classement F, calculé grâce au jeu de données d'entrainement.

1.4.3 Quantification de l'information

La composante principale des arbres de décision et la quantification de l'information. *Dans la théorie de l'information, les notions de quantité d'information et d'incertitude sont équivalentes* [2, 6]. Partant de ces concepts, la quantité d'information $h(x)$ d'une probabilité $P(x)$ est une fonction croissante : si $P(x)$ augmente, $h(x)$ augmente. Par ailleurs, les événements certains et impossibles n'apportent aucune information étant donné que le résultat est connu d'avance.

Afin de mesurer le gain d'information, il est nécessaire d'introduire le concept d'entropie correspondant à l'impureté d'une observation en théorie de l'information [7].

Soit S un ensemble d'observations de taille n où $s_i = \{\text{Y}, \text{N}\}$. L'entropie E de S, où p_Y correspond à la probabilité d'avoir Y et p_N d'avoir N [8], est définie par

$$E(S) = -p_Y log_2 p_Y - p_N log_2 p_N \tag{1.10}$$

La figure 1.2 représente la fonction d'entropie de S en fonction de p_Y. On observe que l'entropie est comprise entre 0 et 1 et vaut 0 lorsque p_Y vaut 0 ou 1. Cela confirme le fait que les événements impossibles et certains n'apportent aucun gain d'information.

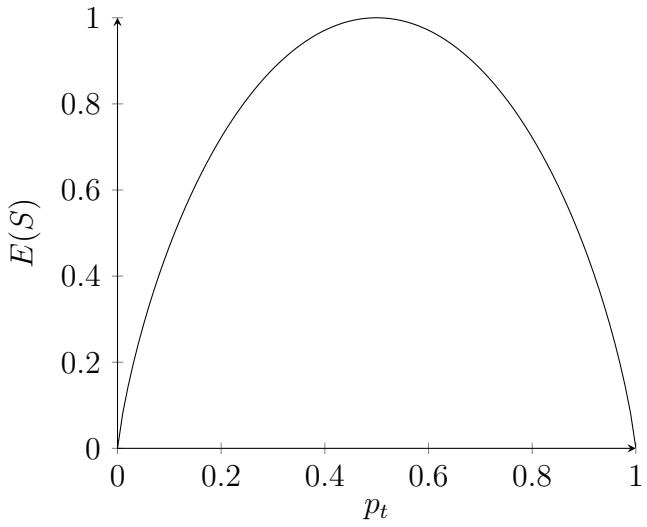

Figure 1.2 – Entropie de S en fonction de p_t

Nous venons de montrer l'entropie de S pour un attribut binaire. Par extension, si une observation s_i peut prendre l valeurs, alors

$$E(S) = \sum_{i=1}^{l} -p_i log_2 p_i \tag{1.11}$$

Par exemple, d'après la table 1.1, l'entropie de l'attribut sortie est égale à

$$\begin{aligned} E(Sortie) &= -\frac{9}{14} log_2(\frac{9}{14}) - \frac{5}{14} log_2(\frac{5}{14}) \\ &= 0.4098 + 0.5305 \\ &= 0.940 \end{aligned} \tag{1.12}$$

A partir de cette entropie, il est possible d'obtenir une mesure appelée gain d'information. Celle-ci correspond à la réduction de l'entropie et permet de quantifier l'apport en information lors du choix d'un test. Elle est définie [8] par

$$G(S, A) = E(S) - \sum_{v \in V_A} \frac{|S_v|}{|S|} E(S_v) \tag{1.13}$$

Par exemple, afin de créer l'arbre de décision de la table Météo, il faut déterminer l'attribut qui sera la racine de l'arbre. Pour cela, il faut déterminer quel attribut maximise le gain d'information :

$$
\begin{aligned}
G(S, Temps) &= E(S) - \frac{5}{14}E(S_{T,Soleil}) - \frac{4}{14}E(S_{T,Nuageux}) - \frac{5}{14}E(S_{T,Pluie}) = 0.246 \\
G(S, Temper) &= E(S) - \frac{4}{14}E(S_{Tp,Elev}) - \frac{6}{14}E(S_{Tp,Moy}) - \frac{4}{14}E(S_{Tp,Basse}) = 0.030 \\
G(S, Humidite) &= E(S) - \frac{7}{14}E(S_{H,Haute}) - \frac{7}{14}E(S_{H,Normale}) = 0.152 \\
G(S, Vent) &= E(S) - \frac{6}{14}E(S_{V,Fort}) - \frac{8}{14}E(S_{V,Faible}) = 0.048
\end{aligned}
\tag{1.14}
$$

D'après les résultats obtenus :
$G(S, Temps) > G(S, Humidite) > G(S, Vent) > G(S, Temperature)$. L'attribut *Temps* sera donc choisi comme racine de l'arbre de décision. Ce processus sera ensuite répété pour chaque nouveau nœud crée jusqu'à ce que toutes les feuilles soit atteintes (figure 1.1).

1.5 Arbre de décision pour l'optimisation

1.5.1 Problème d'*overfitting*

Un des problèmes principal des arbres de décision est le sur-ajustement (*overfitting*) : la taille d'un arbre augmente linéairement avec la taille du jeu de données d'apprentissage. Ce principe d'*overfitting* peut être définit de la manière suivante : *soit H un espace d'hypothèses et $h \in H$, h sur-ajuste le jeu d'apprentissage s'il existe une alternative $h' \in H$, tel que h a une marge d'erreur plus petite que h' sur les jeux d'apprentissage, mais une marge d'erreur plus grande sur la totalité du jeux de données.* [8] On peut voir sur la figure 1.3 que plus la précision du jeu d'apprentissage augmente, plus la précision du jeu de test diminue. C'est pourquoi des méthodes d'élagage et d'amélioration d'arbre sont mises en place afin de minimiser ce problème.

1.5.2 Élagage

Pré-élagage

Le pré-élagage consiste à arrêter prématurément la construction de l'arbre même si chaque feuille ne correspondent pas à une et seulement une classe. Certains nœuds ne seront alors pas plus développés, selon différents critères d'arrêt choisi au préalable.

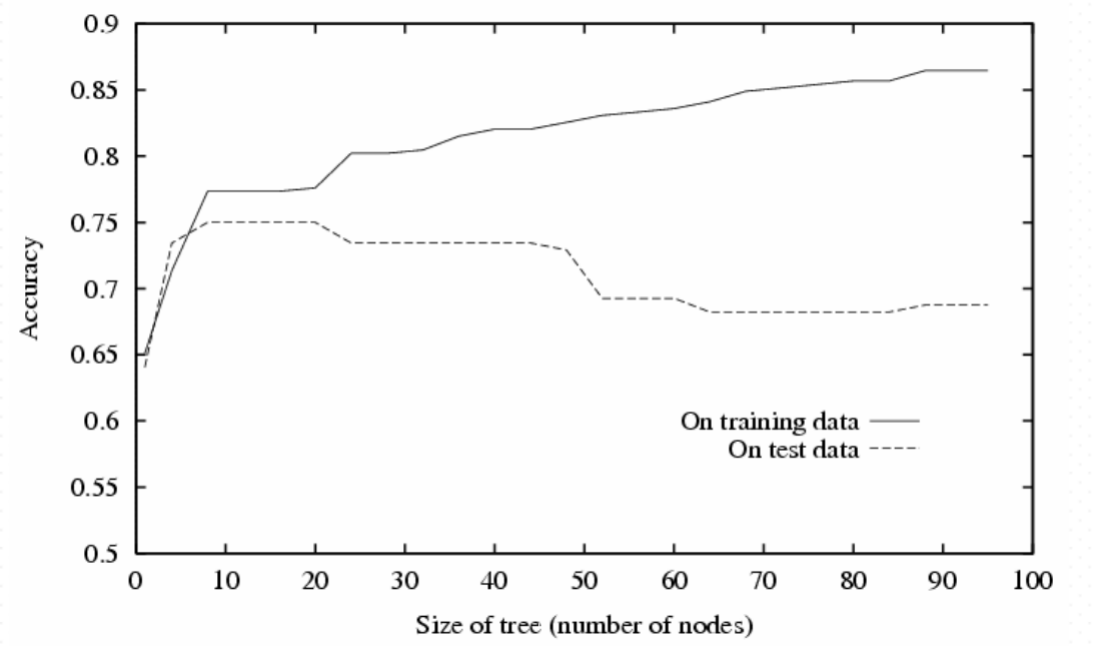

Figure 1.3 – Overfitting d'un arbre de décision sur une base de données médicales afin de déterminer si les patients sont atteints de diabète. [8]

Une première méthode consiste à définir un nombre d'objets minimum par nœud. Si lors de la création de nouveaux nœuds, l'un deux ne dépasse pas le seuil, alors ceux-ci ne sont pas créés et le nœud parent devient une feuille de l'arbre. Une deuxième méthode utilise le test du χ^2 afin de déterminer s'il est statistiquement pertinent de sélectionner un attribut pour construire la suite de l'arbre. Il existe par ailleurs d'autre méthodes qui consiste à définir des seuils pour arrêter la construction [2, 9].

- Le seuil maximum de feuilles a été atteint.
- Le seuil maximum d'attributs a été franchi.
- Le gain d'information est inférieur à celui fixé.

Post-élagage

Contrairement au pré-élagage, le post-élagage se place une fois la construction de l'arbre faite. L'objectif est de supprimer des sous-ensemble de l'arbre afin de les remplacer par des feuilles en minimisant l'impact sur le taux d'erreur, voire réduire le taux d'erreur. Il existe différentes manières d'élaguer un arbre de décision, quatre d'entre-elles seront étudiées [10].

Une première méthode de post-élagage est le *reduced error pruning* (Quinlan [11]). Elle compare le nombre d'erreur de classement entre le sous-arbre T_t intact et le sous-arbre T'_t lorsque que t est transformé en feuille. Si T'_t est plus performant, alors T_t est élagué et t devient une feuille de l'arbre.

Le *pessimistic error pruning* est une méthode d'élagage proposée par Quinlan [11]. Elle remplace le nombre J d'observations mal classées sur une feuille par $J + \frac{1}{2}$. D'après lui, le rapport $\frac{J}{K}$ d'observations mal classées, où K est le nombre d'observations sur une feuille, est une estimation optimiste pas assez fiable, c'est pourquoi il a choisi de la remplacer par une distribution binomiale [8, 11].

La troisième méthode de post-élagage est le *cost complexity pruning*. Elle consiste à construire une séquence d'arbre $T_0, T_1, \ldots, T_L$ en minimisant la valeur de α correspondant à la moyenne du nombre d'observations mal classées par feuille, T_0 étant l'arbre initial et T_L l'arbre final. T_{i+1} est obtenu en élaguant tous les nœuds de T_i qui ont la plus petite valeur de α [10].

Enfin, le *minimum error pruning* permet de minimiser le taux d'erreur attendu d'un sous-arbre élagué. Si celui-ci est inférieur au taux d'erreur du sous-arbre sans élagage, alors le nœud sera remplacé par une feuille. Ce taux d'erreur est définit par

$$E_k = \frac{(n - n_j + k - 1)}{n + k} \tag{1.15}$$

où n correspond au nombre d'observations dans une feuille, n_j observations appartenant à la classe j et k le nombre de valeurs de la classe [10].

1.5.3 Méthodes d'amélioration

TODO : bagging, boosting et Genetic

Chapitre 2

Lazy Decision Tree

2.1 Lazy Decision Tree pour le classement

2.2 Lazy Decision Tree pour l'optimisation

Chapitre 3

Classement supervisé

3.1 Introduction

Il existe de nombreuses méthodes de classement supervisé. L'objectif est d'analyse les algorithmes les plus importants et utilisés afin de mettre en avant la partie mise en œuvre de l'apprentissage supervisé et de comprendre le fonctionnement de chacun. La première méthode est le *K-Nearest Neighbors*, suivie de CART, puis de ID3 et C4.5 pour finir par A*.

3.2 K Nearest Neighbors

3.2.1 Concept

La méthode des k plus proches voisins (*K-Nearest Neighbors ou K-NN* est une méthode d'apprentissage supervisée. Elle peut s'utiliser de différente manière, l'une étant pour le classement. *Le problème des K-NN est de créer une structure de données pour un ensemble d'objets qui, étant donné un objet q, son plus proche voisin dans l'ensemble peut être trouver rapidement*. [12] L'objectif est de placer les observations du jeu d'apprentissage sur un espace métrique. Naturellement, ces observations seront majoritairement regroupés vers un même espace. Lors de l'ajout d'une nouvelle instance, l'algorithme va analyser quels sont ses k voisins les plus proches afin de déterminer la classe à attribuer.

La figure 3.1 montre un ensemble de données [1] contenant deux attributs $x \in [0, 30]$ et $y \in [0, 30]$ et deux classes, l'une rouge et l'autre jaune. Il est clairement visible que les points rouges se trouvent sur la partie supérieure gauche du graphique tandis que les points jaunes, sur la partie inférieure droite (figure 3.2). Si une nouvelle instance apparaissait, $\forall k \in \mathbb{N}$, celle-ci serait relativement simple à classer.

En revanche, avec *K-NN*, le choix du k est arbitraire et celui-ci peut influencer le résultat obtenu.

1. Disponible à l'adresse : `https://www.kaggle.com/alqamahjsr/topic-3-decision-trees-and-knn`

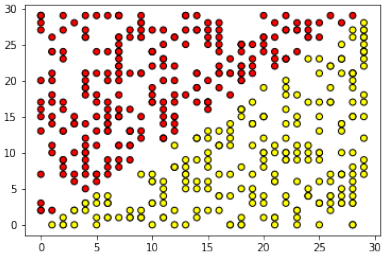

Figure 3.1 – Représentation graphique de points ayant pour classe "rouge" ou "jaune"

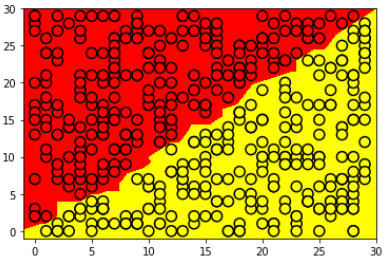

Figure 3.2 – Séparation des données par leur classe

3.2.2 Choix de k

k est une valeur arbitraire choisie avant le lancement de l'algorithme. Cette valeur permet de choisir le nombre de voisins à regarder afin de déterminer la classe d'une observation. Cela signifie que si k est trop petit, l'algorithme retournera la valeur de l'observation la plus proche. Un petit k générera alors un modèle trop précis, qui engendrera un problème d'*overfitting*. Dans le cas contraire, un trop grand k pourrait réduire la précision du modèle et rendre le comportement du modèle trop abstrait [**data_minig**].

Par exemple, la figure 3.3 [2] met en avant le problème du choix de k. Cette figure possède une observation de test (cercle vert) à classer soit en carré bleu, soit en triangle rouge. Dans le premier cas, $k = 3$, l'algorithme va donc regarder les trois voisins les plus proche du cercle : deux triangles rouges et un carré bleu et déduire que l'observation est un triangle rouge. Dans le deuxième cas, $k = 5$, les cinq plus proches voisins sont deux triangles rouges et trois triangles bleus : l'observation sera alors un carré bleu. Dans cet exemple, un k trop grand a changer la classe en carré bleu alors que celle-ci semble être plus proche des triangles rouge.

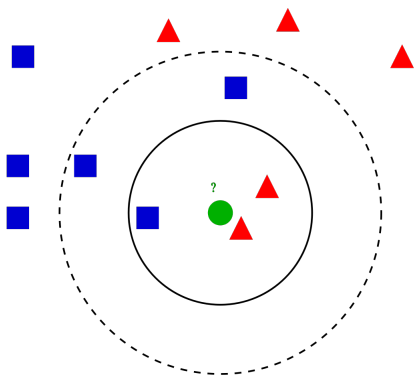

Figure 3.3 – Exemple de K-NN avec deux valeurs différentes de k

2. Antti Ajanki AnAj - Own work, CC BY-SA 3.0, `https://commons.wikimedia.org/w/index.php?curid=2170282`

algo

Afin d'améliorer *K-NN*, il est possible de donner un poids aux voisins et aux attributs. Ainsi, un voisin proche aura plus d'importance qu'un voisin éloigné qui risque de ne pas être de la même classe. Par ailleurs, un attribut n'étant pas pertinent aura beaucoup moins d'importance qu'un attribut aidant au classement [5].

3.3 CART

Classification And Regression Trees (CART) est une méthode développée par Breiman et al. [13]. Cette approche permet de créer des arbres binaires (chaque nœud a deux descendants directs). Chaque test, ou *split*, correspond à la question binaire qui maximise la réduction de l'entropie. Pour cela, l'algorithme va parcourir tous les attributs de chaque nœud et garder le meilleur pour en dégager le *split* maximisant la réduction de l'entropie. Il va ensuite comparer les *splits* pour garder le meilleur.

3.4 ID3

3.5 C4.5

3.6 A *

Chapitre 4

Apprentissage du MNIST

4.1 Définition du MNIST

4.2 Méthode de recherche

4.3 Algorithme de classement

4.4 Algorithme Lazy

4.5 Comparaison des résultats

Bibliographie

[1] Hélène Guérin. *Qu'est ce qu'une probabilité ?* 2008.

[2] Lamis Hawarah. "Une approche probabiliste pour le classement d'objets incomplètemet connus dans un arbre de décision". Thèse de doct. Université Joseph Fourier - Grenoble I, 2008.

[3] R. Gilleron F. Denis. *Apprentissage à partir d'exemples*. 2000.

[4] Jan Kozak. *Decision tree and ensemble learning based on ant colony optimization*. Springer, 2019.

[5] Daniel T. Larose. *Discovering knowledge in data : an introduction to data mining*. Wiley-Interscience, 2005.

[6] Steven Roman. *Introduction to coding and information theory*. Undergraduate texts in mathematics. 1997.

[7] C. E. Shannon. "A Mathematical Theory of Communication". In : *The Bell System Technical Journal, Vol. 27*. 1948, p. 379-423, 623-656.

[8] Tom M. Mitchell. *Machine Learning, International Edition*. McGraw-Hill Series in Computer Science. 1997.

[9] Nikita Patel et Saurabh Upadhyay. "Study of Various Decision Tree Pruning Methods with their Empirical Comparison in WEKA". In : *International Journal of Computer Applications, Volume 60* (2012).

[10] Floriana Esposito, Donato Malerba et Giovanni Semeraro. "A Comparative Analysis of Methods for Pruning Decision Trees". In : *IEEE Trans. Pattern Anal. Mach. Intell.* 19.5 (1997), p. 476-491.

[11] J. Ross Quinlan. "Simplifying Decision Trees". In : *International Journal of Man-Machine Studies* 27.3 (1987), p. 221-234.

[12] Jorge Moraleda. "Gregory Shakhnarovich, Trevor Darrell and Piotr Indyk : Nearest-Neighbors Methods in Learning and Vision. Theory and Practice". In : *Pattern Anal. Appl.* 11.2 (2008), p. 221-222.

[13] Leo Breiman et al. *Classification and Regression Trees*. Wadsworth, 1984. isbn : 0-534-98053-8.

Table des matières

1 Concepts formels **7**

1.1 Introduction 7

1.2 Rappels de probabilité 7

1.2.1 Expérience aléatoire 7

1.2.2 Événements 7

1.2.3 Probabilités 7

1.3 Arbre de décision 8

1.3.1 Définition 8

1.3.2 Exemple 8

1.4 Arbre de décision pour le classement 9

1.4.1 Attributs et classes 9

1.4.2 Apprentissage supervisé 11

1.4.3 Quantification de l'information 11

1.5 Arbre de décision pour l'optimisation 13

1.5.1 Problème d'*overfitting* 13

1.5.2 Élagage 13

1.5.3 Méthodes d'amélioration 15

2 Lazy Decision Tree **17**

2.1 Lazy Decision Tree pour le classement 17

2.2 Lazy Decision Tree pour l'optimisation 17

3 Classement supervisé **19**

3.1 Introduction 19

3.2 K Nearest Neighbors 19

3.2.1 Concept 19

3.2.2 Choix de k 20

3.3 CART 21

3.4 ID3 21

3.5 C4.5 21

3.6 A * 21

4 Apprentissage du MNIST **23**

4.1 Définition du MNIST 23

4.2 Méthode de recherche 23

4.3 Algorithme de classement 23

4.4 Algorithme Lazy 23

4.5 Comparaison des résultats 23

Bibliographie **25**

Table des figures

1.1 Arbre de décision de la table Météo 10
1.2 Entropie de S en fonction de p_t 12
1.3 Overfitting d'un arbre de décision sur une base de données médicales afin de déterminer si les patients sont atteints de diabète. [8] 14

3.1 Représentation graphique de points ayant pour classe "rouge" ou "jaune" 20
3.2 Séparation des données par leur classe 20
3.3 Exemple de K-NN avec deux valeurs différentes de k 20

Liste des tableaux

1.1 Table Météo 9