Ne rien écrire hors du cadre.

Regon 15 : Exemples d'algorithmes d'apprentissage supervisés et non supervisés

Niveau : MP2I / MPI

I - Introduction

I.1 - Définition

Def 1 : On dit que l'algorithme A apprend via l'entrainement E pour un ensemble de tâches T et une mesure de performance P si sa performance sur T s'améliore après E.

Def 2 Un algorithme d'apprentissage se déroule en deux phases : l'apprentissage, puis la prédiction sur de nouvelles données.

Remarque 3 : Ces deux phases sont similaires à un algorithme détectant si des elements sont dans une liste : le prétraitement serait ici le tri de la liste, la seconde phase celle de recherche dichotomique.

Def 4 : On considère deux familles de l'apprentissage :

1) l'apprentissage supervisé : on dispose d'espace X d'entrée et Y de sortie, et d'un ensemble E d'exemples $(x_i, y_i) \in X \times Y$, représentant une fonction $f : X \to Y$. On cherche à construire $\hat{h} : X \to Y$ approximant f.

2) l'apprentissage non supervisé : on dispose seulement de X dont on veut découvrir les structures sous-jacentes. Le but est alors de partitionner X (clustering).

Exemple 5 : Sur un ensemble d'images d'animaux, on peut :

- Savoir lesquels sont des chats, des chiens, etc... (apprentissage supervisé)
- Savoir quels animaux sont de la même espèce (apprentissage non supervisé)

Remarque 6 : Il existe d'autres familles, comme l'apprentissage par renforcement où on maximise un critère d'utilité par expériences successives.

I.2 - Evaluation d'un algorithme d'apprentissage

On cherche à évaluer la performance d'un algorithme d'apprentissage.

Def 7 : On découpe nos données d'entrée E en deux :

- les données d'apprentissage, servant à entraîner l'algorithme
- les données de validation, servant à imiter la prédiction, mais en comparant le résultat obtenu à celui attendu.

Remarque 8 : Si on a trop peu de données, on peut également faire une validation croisée, consistant à utiliser alternativement des données comme apprentissage et comme validation.

Remarque 9 : Cette phase est utile au calibrage des paramètres de l'algorithme.

II - Apprentissage supervisé

On reprend ici les notations de la définition 4.

Def 10 : Si Y est un ensemble fini de classes, on parle de problème de classification

Si $Y = \mathbb{R}$, on parle de problème de régression.

Exemple 11 : Sur un ensemble d'animaux, on peut avoir :

- $Y = \{\text{chat}, \text{chien}, \text{loup}\}$
- $Y = \mathbb{R}$ représentant le poids d'un animal

Concentrons nous sur le problème de classification. On cherche à inférer de E, la classe de $x \in X$. Notons C la partition de X représentant les différentes classes.

II.1 - k plus proches voisins

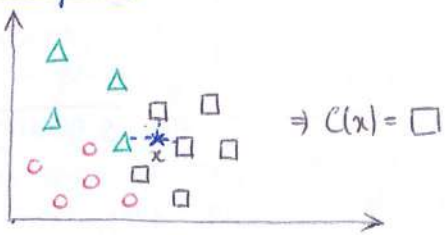
On s'intéresse ici au cas où $X = \mathbb{R}^d$

Algorithme 12 : k plus proches voisins (x) :

1) Déterminer les k plus proches voisins de x dans E
2) Choisir la classe majoritaire parmi ces voisins.

Developpement 1 : Présentation de l'algorithme avec discussion sur le choix du paramètre.

Exemple 13 : $k=3$



Remarque 14 : Ici, pour choisir k, on utilise la méthode de performance du I.2) en essayant plusieurs paramètres.

TD 15 : Appliquer l'algorithme sur le jeu de donnée MINST.

Remarque 16 : Pour un problème de régression, on ne choisirait pas la classe majoritaire mais on agrégerait les données (exemple : moyenne)

Implémentation 17 :

Solution initiale : Stocker nos k valeurs en cours dans une file de priorité implémentée par un tas max et parcourir les n points en mettant à jour la file de priorité.
Complexité : $O(n \log k)$

Solution diviser pour régner : Pré-traitement où on stocke nos valeurs dans un arbre d-dimensionnel.
Complexité : Prétraitement : $O(n \log n)$
Recherche des k voisins : $O(k \log n)$ en moyenne
$O(nk)$ dans le pire des cas

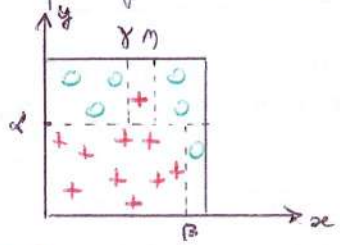
Developpement 2 : Présentation de la structure d'arbre d-dimensionnel.

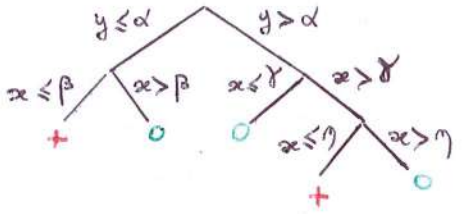
Remarque 18 : On parle en général d'arbre K-dimensionnel, mais pour éviter la confusion avec k plus proches voisins, je dis d-dimensionnel.

II.2 - Arbre de décision

On s'intéresse ici au cas $X = \mathbb{R}^d$ (ou $\{0,1\}^d$)

Idée 19 : Partitionner récursivement X grâce à un arbre de décision où chaque feuille a une classe.





Def 20 : Définissons l'entropie d'une partie S de E :

$$H(S) = \sum_{c \in C} \frac{n_c}{|S|} \log_2\left(\frac{n_c}{|S|}\right)$$ où n_c : nombre d'éléments de S de classe c

Remarque 21 : Si tout le monde est classifié pareil, $H(S) = 0$

Def 22 : Le gain d'une partition S_1, S_2 de S est :

$$H(S) - \left(\frac{|S_1|}{|S|} H(S_1) + \frac{|S_2|}{|S|} H(S_2)\right)$$

Algorithme 23 : ID3

```
//On construit récurrivement notre arbre de décision sur notre ensemble
//S de données restantes
Si S est vide :
    choisir la classe la plus représentée du noeud parent
Si toutes les données de S ont la même classe :
    en faire une feuille, avec cette classe
Sinon, on choisit la coordonnée i et la valeur m tq la
partition S1 = {(x1, ..., xn) ∈ S | xi ≤ m} et S2 = {x ∈ S | xi > m}
maximise le gain.
```

Remarque 24 : On atteint vite du surapprentissage. Pour éviter cela, on peut élager le bas de l'arbre (algorithme de Cart).

TD25 : Application à détecter la langue d'une page wikipedia à partir de la matrice de fréquence de facteurs de 2 lettres.

Remarque 26 : Dans le cas d'une regression, on peut prendre comme mesure d'impureté la variance.

II.3 - Représentation de la qualité des classes

Quand on mesure la performance de notre algorithme, on peut chercher à représenter la qualité de nos prédictions.

Def 27 : (matrice de confusion) : On associe Y à $\{1,..,n\}$. La matrice de confusion est alors la matrice carrée M de taille n tq $M_{i,j}$: nombre d'élements de la classe i qui ont été classés à j.

Propriété 28 : Plus notre algorithme est correcte, plus la diagonale est dominante.

III - Apprentissage non supervisé

Remarque 29 : Il existe plusieurs types d'apprentissage non supervisés. On peut par exemple penser à la réduction de dimensions : On a $X \subseteq \mathbb{R}^d$ et on cherche $f : X \to \mathbb{R}^{d'}$ avec $d' < d$ et tq $f(x)$ et $f(y)$ sont proches si x et y le sont.

On se concentre ici sur le clustering (regroupement) qui consiste à trouver $f : X \to \{1..,k\}$ essayant de regrouper les éléments les plus proches.

Exemple 30 : On a un manuscrit dans un alphabet inconnu, et on cherche à savoir quelles lettres sont les mêmes.

Exercice 31 : Quel X prendre ?

III.1 - Classification hiérarchique ascendante

Idée 32 : Chacun est seul dans sa classe au début, et tant qu'on a k classes, on fusionne les classes les plus proches.

Remarque 33 : C'est un algorithme glouton.

Remarque 34 : Pour définir la distance entre deux classes, on peut choisir :

$$d(S_1,S_2) = \min_{x\in S_1,\, y\in S_2} (d(x,y)) \quad \text{ou} \quad d(S_1,S_2) = \frac{1}{|S_1|\times|S_2|} \sum_{\substack{x\in S_1\\ y\in S_2}} d(x,y)$$

$$d(S_1,S_2) = \max_{x\in S_1,\, y\in S_2} (d(x,y))$$

Activité 35 : représenter l'execution de l'algorithme sur un dendogramme

III.2 - Algorithme des k-moyennes

On cherche à trouver $S_1,..,S_k$ une partition de X et $z_1,..,z_k \in \mathbb{R}^d$ minimisant $\sum_{i=1}^{k} \sum_{x\in S_i} d(x,z_i)$

Algorithme 36 : K-mean

Initialisation : Associer à chaque valeur une classe aléatoire

répéter jusqu'à stabilisation :

pour x dans X :

assigner à x la classe $\operatorname{argmin}_{i\in\{1..,k\}} d(x,z_i)$

pour i dans $1..,k$:

$z_i \leftarrow \frac{1}{|S_i|} \sum_{x\in S_i} x$

Propriété : Cet algorithme termine. Variant : $\sum_{i=1}^{k} \sum_{x\in S_i} d(x,z_i)$

Remarque 37 : On ne trouve pas toujours le minimum, on peut tomber dans un minimum local. On relance donc plusieurs fois l'algo, d'où l'initialisation aléatoire.

TP38 : Réduction de la palette d'une image.

Remarque 39 : Comment choisir k ? On affiche la valeur de la cible en fonction de k, et on cherche le changement de pente.