École Centrale de Lille

Science des données et Intelligence artificielle

Pierre Chainais

Semestre d’automne

Décision et apprentissage

Une image contenant texte, Police, Graphique, capture d’écran

Description générée automatiquement

Table des matières

[1. Introduction 4](#_Toc149125533)

[1.1. Objectifs de l’apprentissage 4](#_Toc149125534)

[1.2. Position du problème 4](#_Toc149125535)

[1.3. Familles d’approches 4](#_Toc149125536)

[2. Modèles linéaires pour la régression 5](#_Toc149125537)

[2.1. Position du problème 5](#_Toc149125538)

[2.2. Modèle linéaire en et en 5](#_Toc149125539)

[2.3. Modèle linéaire en et non en 6](#_Toc149125540)

[2.4. Moindres carrés régularisés et Ridge Regression 7](#_Toc149125541)

[2.5. Formulation bayésienne du problème de régression linéaire 9](#_Toc149125542)

[2.6. Fonction de coût et estimateur théorique 11](#_Toc149125543)

[2.7. Compromis biais variance 11](#_Toc149125544)

[3. Classification et théorie de la décision 13](#_Toc149125545)

[3.1. Définitions 13](#_Toc149125546)

[3.2. Erreur de prédiction, fonction de coût 13](#_Toc149125547)

[3.3. Classification binaire 13](#_Toc149125548)

[3.4. Décision statistique théorique 14](#_Toc149125549)

[3.5. Inférence et décision : familles de modèles 14](#_Toc149125550)

[3.6. -NN : plus proches voisins 14](#_Toc149125551)

[3.7. Sélection de modèle 15](#_Toc149125552)

[4. Modèles linéaires pour la classification supervisée 16](#_Toc149125553)

[4.1. Frontières discriminantes linéaires 16](#_Toc149125554)

[4.2. Moindres carrés pour la classification 16](#_Toc149125555)

[4.3. Analyse discriminante linéaire / quadratique (LDA / QDA) 16](#_Toc149125556)

[4.4. Approche bayésienne naïve 20](#_Toc149125557)

[4.5. Régression logistique 22](#_Toc149125558)

[5. Arbres de décision 24](#_Toc149125559)

[5.1. Principes de construction 24](#_Toc149125560)

[5.2. Algorithme 24](#_Toc149125561)

[5.3. Mesures d’impureté d’un nœud 25](#_Toc149125562)

[5.4. Choix du sélecteur 26](#_Toc149125563)

[5.5. Élagage : régularisation des arbres 26](#_Toc149125564)

[5.6. Bagging : Bootstrap Aggregating 27](#_Toc149125565)

[5.6. Random Forest : Forêts aléatoires 27](#_Toc149125566)

[6. Boosting 29](#_Toc149125567)

# 1. Introduction

## 1.1. Objectifs de l’apprentissage

L’apprentissage comprend :

* La classification supervisée, semi-supervisée et non supervisée : la prédiction est discrète
* La régression : la prédiction est continue

## 1.2. Position du problème

Une image contenant texte, diagramme, Police, capture d’écran

Description générée automatiquement

Schéma d’un modèle

Les bonnes propriétés d’un modèle sont la capacité de généralisation et la possibilité de réutilisation du modèle dans un contexte différent.

## 1.3. Familles d’approches

Dans tous les cas l’objectif est la minimisation des erreurs de prédiction.

Cela peut être la **minimisation d’une fonction de coût** (risque) (par exemple, ). On parle **d’optimisation**.

Cela peut également être la **maximisation d’une probabilité** (par exemple, ). On parle dans ce cas de **raisonnement Bayésien**.

# 2. Modèles linéaires pour la régression

## 2.1. Position du problème

Les données sont de la forme et les cibles .

On peut faire l’hypothèse d’une dépendance linéaire en et en : .

On peut enrichir le modèle en supprimant la linéarité en : ( traduit le changement de représentation du problème).

## 2.2. Modèle linéaire en et en

Les données sont et le modèle de prédiction est :

Où :

On pose :

L’objectif est de minimiser l’erreur quadratique :

Soit :

L’erreur quadratique est convexe et différentiable donc il existe une solution unique. On cherche un optimum en calculant un point on le gradient s’annule.

La matrice est symétrique **définie** positive, i.e. ses valeurs propres sont **strictement** positives si est de rang plein. Dans ce cas, on a bien la convexité.

Sous cette hypothèse, est solution unique de :

On appelle pseudo-inverse de Moore-Penrose de la solution :

Si est inversible, alors , mais est rectangulaire donc cette hypothèse est rarement vérifiée.

Au lieu de résoudre , on cherche un minimum de . De plus, la résolution de n’est pas toujours évidente, donc on peut avoir recours à la descente de gradient. Cela a aussi l’avantage de ne pas avoir à inverser de matrice, et donc d’économiser du coût calculatoire.

Les prédictions sur l’ensemble d’entrainement sont : . On appelle la matrice « hat matrix ».

## 2.3. Modèle linéaire en et non en

On pose et les pour (pas forcément linéaires en ).

La fonction et permet d’obtenir la matrice de dimension .

Par exemple, si et , alors et .

Donc de manière similaire :

Par exemple, pour une régression polynomiale à l’ordre 3 d’un vecteur de dimension 2 :

Pour des sorties multiples, c’est-à-dire , avec des données sont et le modèle , on a :

On minimise le coût quadratique :

## 2.4. Moindres carrés régularisés et Ridge Regression

La dimension peut être grande, le nombre d’échantillons est fini donc limité, le modèle implique paramètres ( dans le cas où on reste dans la dimension initiale).

L’objectif est d’avoir un modèle précis donc assez riche (pas d’underfitting) mais qui ne surapprend pas (overfitting).

L’idée est de partir d’un modèle potentiellement trop riche et le régulariser, c’est-à-dire, sélectionner un sous-modèle.

Une image contenant diagramme, ligne, Tracé

Description générée automatiquement

Importance de l’ordre sur un modèle polynomial.

### 2.4.1. La méthode de Ridge Regression

On ne veut pas changer la moyenne mais étudier uniquement les variations autour d’elle. Donc on traite à part le terme d’ordre 0 qu’on fixe comme la moyenne des sorties : ; et on ne considère plus les vecteurs mais simplement).

On cherche qui minimise :

Le principe est de minimiser si et si de minimiser . On cherche donc un hyperparamètre intermédiaire appelé aussi **paramètre de régularisation**.

Le optimal correspond au modèle qui a la meilleure généralisation. Comme donne le même poids à tous les , cela nécessite de **normaliser** les données auparavant, c’est-à-dire de le rendre centrés-réduits :

On résout le problème pour sachant les données centrées-réduites :

On a :

Donc :

On cherche à résoudre par rapport à :

La solution est

Donc si et seulement si . Et est toujours bien définie car est toujours inversible. Numériquement, il arrive que soit mal conditionnée :

Plus généralement, on peut utiliser d’autres régularisations, par exemple en .

### 2.4.2. Au-delà de la Ridge Regression

On suppose qu’on observe , et qu’on a (base orthonormée) donc est le conjugué du transposé. On veut résoudre :

Où , .

Si , on parle de Ridge Regression :

Si , on parle de parcimonie :

On parle de seuillage dur : être supérieur ou inférieur à :

Si , on parle de LASSO :

On parle de seuillage doux, c’est-à-dire qu’on applique l’opérateur continu :

Si la solution est vraiment parcimonieuse, alors la solution LASSO est aussi parcimonieuse.

## 2.5. Formulation bayésienne du problème de régression linéaire

On adopte une vision probabiliste, c’est-à-dire que le monde est fait de variables aléatoires dont on observe les réalisations :

Où est la précision, la variance et le bruit est indépendant de . Donc chaque observation .

On cherche les paramètres qui expliquent le plus probablement de façon **vraisemblable**.

On suppose que les observations sont indépendantes et identiquement distribuées.

On appelle **vraisemblance**:

On appelle **Maximum Likelihood Estimate (MLE) de**  la grandeur :

On appelle **Maximum Likelihood Estimate (MLE) de**  (variance) la grandeur :

On cherche donc les arguments d’annulation de :

C’est-à-dire :

Avec régularisation (donc avec un a priori sur ), on ajoute par exemple . On peut prendre le chemin inverse en appliquant l’opérateur et on obtient , une loi gaussienne de précision et de moyenne .

On appelle **loi a posteriori** , la **loi a priori** et la **vraisemblance** et :

On appelle **Maximum A Posteriori (MAP) de**  la grandeur :

Pour régler et , on utilise et on règle comme un hyperparamètre. On utilise la cross-validation par exemple pour estimer , qui nous sert ensuite à minimiser .

Cette approche permet d’évaluer les incertitudes grâce aux formules de probabilité de , ce qui n’était pas possible avec les méthodes de régression.

## 2.6. Fonction de coût et estimateur théorique

On a la fonction de coût et on cherche à estimer la fonction idéalement en minimisant :

On l’écrit dans le cas  :

Donc on cherche :

Donc :

Donc :

On obtient :

On s’intéresse maintenant à la fonction de coût de où les sont définies par :

Lorsque , et on a vu que pour , .

## 2.7. Compromis biais variance

On cherche un modèle qui approche . On fait l’hypothèse sur la génération des données, avec le système réel et un bruit indépendant de , d’espérance nulle et de variance finie :

Le biais se trouve entre et et la variance entre cette espérance et .

Les trois sources d’écart sont donc la **variance du bruit**, la **variance liée aux données** et le **biais de la modélisation**.

Alors :

Donc :

Et est lié à l’erreur sur la **qualité des mesures** (incompressible), à l’**adéquation du modèle** au système réel et à la **sensibilité aux données**.

Une image contenant diagramme, Tracé, ligne

Description générée automatiquement Une image contenant texte, capture d’écran, ligne, diagramme

Description générée automatiquement

Compromis biais variance.

# 3. Classification et théorie de la décision

## 3.1. Définitions

Les données d’entrainements sont de la forme où et avec .

Typiquement, si et si .

Si , avec le en position de la classe de .

## 3.2. Erreur de prédiction, fonction de coût

La fonction de coût 0-1 :

Le taux d’erreur réelle est :

Et devient avec la fonction de coût 0-1 :

Le taux d’erreur empirique :

## 3.3. Classification binaire

Ici, on a et donc deux régions de décision :

Alors, l’espérance de la fonction de coût que l’on veut minimiser :

Une image contenant diagramme, Tracé, ligne

Description générée automatiquement

Fonctions de densité des régions 1 et 2.

Ici, est le seuil de décision et l’erreur est la somme des aires colorées. L’optimum est l’endroit où la zone rouge disparaît, i.e. . En plaçant ce seuil, on détermine les régions et de telle sorte que :

## 3.4. Décision statistique théorique

La **règle de Bayes** donne le **maximum a posteriori (MAP)** :

On appelle **Taux d’erreur Bayésien** la quantité .

La règle de bayes pour la classification est optimale, i.e. pour toute autre règle , .

## 3.5. Inférence et décision : familles de modèles

**L’inférence** est la détermination de et la **décision** est d’utiliser cette quantité pour affecter des classes.

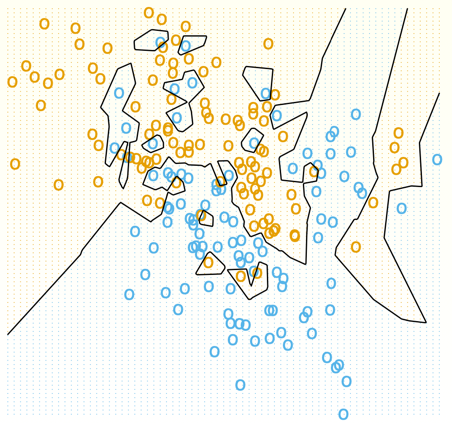
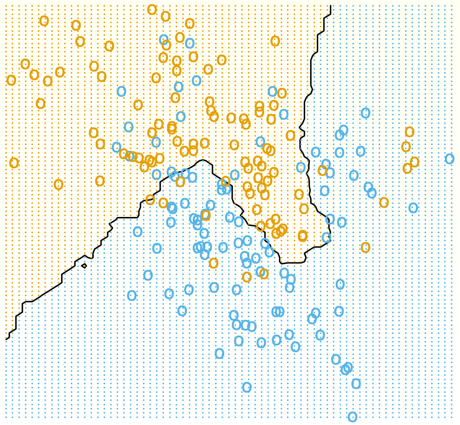
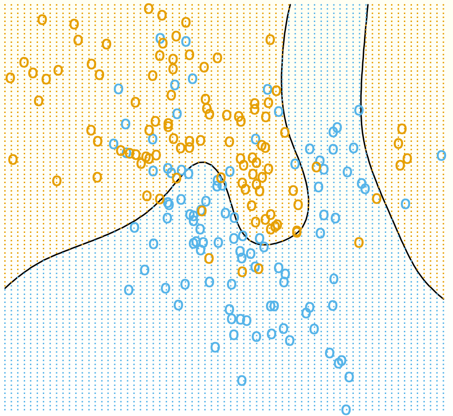
Un modèle **génératif** utilise et pour déduire. L’interprétation de ce modèle est simple mais ce dernier peut être mal adapté (discriminant linear analysis par exemple).

Un modèle **discriminant** estime directement (régression logistique par exemple).

Une **fonction discriminante** estime directement (-NN par exemple). Ces modèles peuvent être très efficaces mais sont souvent peu interprétables.

## 3.6. -NN : plus proches voisins

On classifie dans la classe dont la majorité de ses plus proches voisins fait partie.

Classification des données pour et et règle de bayes.

Lorsque , la fonction est trop riche car elle dépend de variables.

## 3.7. Sélection de modèle

Une image contenant texte, capture d’écran, ligne, diagramme

Description générée automatiquement

Erreur de prédiction en fonction de la complexité du modèle.

# 4. Modèles linéaires pour la classification supervisée

On cherche toujours à prédire une classe pour des entrées . L’idée est de définir régions qui sont séparées par des hyperplans en guise de frontières de décision.

## 4.1. Frontières discriminantes linéaires

Un hyperplan séparateur est de la forme :

Où avec la normale à l’hyperplan et le biais.

Une règle de décision peut séparer en deux régions et .

On travaille avec classifieurs pour éviter les zones d’indécision.

On définit fonctions et la fonction de classification :

## 4.2. Moindres carrés pour la classification

L’idée de faire une régression par moindres carrés pour apprendre n’est pas une bonne idée car elle estime une droite de classification et non un segment : on ne cherche pas à prédire un réel mais une probabilité.

De plus les évènements extrêmes exercent une influence excessive à l’entrainement.

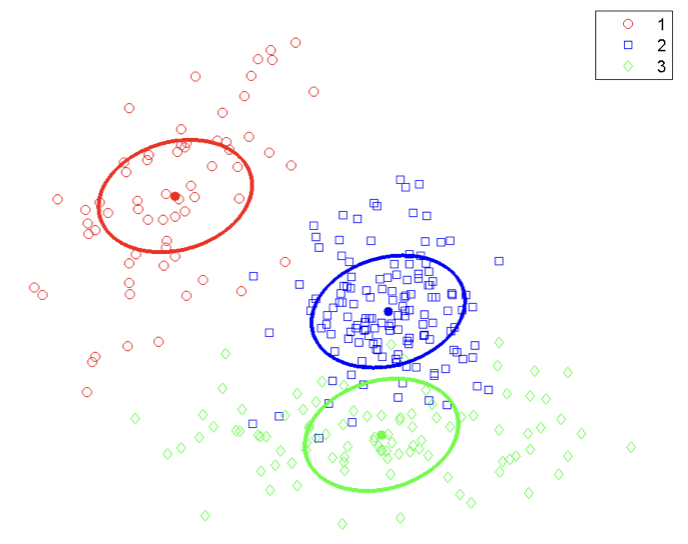
Enfin, il y a un phénomène de masquage lorsqu’une classe est intercalée entre d’autres.

Il ne faut donc **pas utiliser les moindres carrés pour la classification.**

## 4.3. Analyse discriminante linéaire / quadratique (LDA / QDA)

On modélise les probabilités par des gaussiennes (ellipses) où est le centre de l’ellipse et l’étalement selon les directions. Les **ellipses sont dirigées selon les vecteurs propres** et **l’étalement est selon les valeurs propres**.

L’analyse discriminante linéaire suppose et définit donc des droites de séparation. L’analyse discriminante quadratique fait intervenir des matrices de covariance différentes et définit donc des courbes.

 Une image contenant point, motif, Lilas, Point de croix

Description générée automatiquement

Classification par LDA.

Une image contenant cercle, texte, Dessin d’enfant, diagramme

Description générée automatiquementUne image contenant point, motif, Lilas, Point de croix

Description générée automatiquement

Classification par QDA.

Comme ces modèles sont génératifs, on calcule :

Et d’autre part, on pose :

Donc :

On utilise le maximum de vraisemblance pour estimer les paramètres :

La vraisemblance (de toutes les observations sachant tous le paramètres) est :

La neg-log-vraisemblance est :

### Expression des

Pour trouver l’expression de , on dérive :

On cherche l’optimum :

Or est définie positive en posant :

Donc :

### Expression des

Pour trouver l’expression de , on réécrit :

On dérive :

On cherche l’optimum :

Donc :

### Expression des

Pour trouver l’expression de , on réécrit :

On dérive :

On cherche l’optimum, en posant  :

Donc, on obtient la covariance empirique :

### Résumé des expressions

Donc en QDA :

Et en LDA :

### Frontières de décision

La frontière entre les classes :

Donc  :

Donc :

On peut réécrire avec la normale à l’hyperplan :

### Fonction de décision

La fonction de décision est :

Où en QDA :

Et en LDA :

## 4.4. Approche bayésienne naïve

On traite tout d’abord le cas des attributs binaires :

La probabilité conditionnelle peut prendre valeurs, nombre qui explose lorsque augmente.

On fait donc l’hypothèse simplificatrice d’**approche bayésienne naïve**: on suppose que les **attributs sont indépendants conditionnellement aux classes** :

L’avantage est que la probabilité ne prend plus que valeurs et :

Cette méthode a de bonnes performances en pratique.

Les données sont de la forme . Alors :

La vraisemblance s’écrit :

On cherche des estimateurs .

En posant

Donc en l’optimum :

Donc :

Pour , il faut faire attention au fait que de la même manière qu’en LDA et on obtient :

Donc la fonction de décision est :

Où :

Donc est linéaire en , donc les séparateurs sont des hyperplans :

## 4.5. Régression logistique

En régression logistique, on écrit une probabilité continue entre 0 et 1 :

Une image contenant diagramme, ligne, Tracé, texte

Description générée automatiquement

Fonction de probabilité en régression logistique.

On modélise les classes grâce aux avec une dépendance linéaire en et .

La classe sert de référence.

On montre que :

Dans le cas binaire, on a :

Donc la vraisemblance :

Où :

La fonction de coût :

On a la propriété :

Donc :

Et on obtient :

Où :

On doit résoudre le système non-linéaire avec une solution numérique (algorithme de Newton-Raphson) :

A chaque itération :

Où :

Avec :

# 5. Arbres de décision

## 5.1. Principes de construction

On regarde pour chaque classes les occurrences d’un attribut donné : la contingence.

Le maximum de vraisemblance :

Une image contenant diagramme, texte, Dessin technique, Plan

Description générée automatiquement

Exemple de construction d’un arbre de décision pour des attributs continus.

Un nœud pur est un nœud pour lequel toutes les données appartiennent à la même classe.

## 5.2. Algorithme

**Procédure** Construire-arbre (nœud m)  
**début**

**si**

Tous les points de m appartiennent à la même classe

**alors**

**Créer** une feuille portant le nom de cette classe

**sinon**

**Choisir** le meilleur attribut pour créer un nœud  
**Tester** attribut pour séparer m en 2 nœuds fils mL et mR Construire-arbre (mL)  
Construire-arbre (mR)

**fin** si

**fin**

L’algorithme termine car on partitionne les données restantes à chaque niveau de l’arbre.

## 5.3. Mesures d’impureté d’un nœud

On note l’ensemble des données d’apprentissage au nœud .

La fréquence de chaque classe au nœud est :

Le vote de majorité au nœud est donc :

Une fonction d’impureté définie sur avec et est telle que :

* a un maximum unique en
* est minimale en
* pour toute permutation

Les fonctions d’impureté standard sont :

* L’erreur de classification :
* L’indice de Gini :
* La cross-entropie (ou déviance)

Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquement

Exemples de fonctions d’impureté en dimension .

## 5.4. Choix du sélecteur

Si :

* et sont les nœuds fils du nœud
* et les mesures d’impureté des nœuds et
* la mesure d’impureté du père
* et les proportions de données partant des nœuds gauche et droit

Alors la qualité d’une partition au nœud est mesurée par :

On cherche à maximiser cette valeur pour augmenter la pureté.

## 5.5. Élagage : régularisation des arbres

Intuitivement, on veut stopper l’arbre au nœud si pour un seuil donné  :

Mais cette stratégie n’est pas efficace en pratique car plusieurs partitions successives peu décisive peuvent apporter un gain important.

La méthode est donc de construire un arbre trop profond en s’arrêtant lorsqu’une taille minimale est atteinte, puis on coupe les branches de cet arbre par rapport à un critère de complexité.

On définit le **critère coût-complexité**, où est le nombre de nœuds terminaux de et les nœuds terminaux :

Le paramètre règle la complexité de l’arbre :

* correspond à l’arbre
* large va favoriser des arbres peu profonds

Les avantages de cette méthode sont le passage à l’échelle (on travaille sur les attributs indépendamment) et le fait de pouvoir travailler avec des variables caractérielles ou quantitatives.

Les arbres de décision sont instables car un faible changement dans les données d’apprentissage peut donner un arbre très différent, du fait de la construction hiérarchique de l’arbre. Les solutions à cela sont le bagging et les forêts aléatoires.

## 5.6. Bagging : Bootstrap Aggregating

L’idée du bagging est de créer de façon artificielle plusieurs jeux de données pour .

On réalise d’abord un **bootstrap** : un tirage équiprobable avec remise de l’ensemble d’apprentissage de même taille que l’ensemble d’entrainement.

On réalise ensuite une **agrégation** : on construit un arbre de classification pour chaque jeu de données et on prédit avec   la prédiction donnée pour par l’arbre :

L’inconvénient de cette méthode est la perte de la structure hiérarchique, mais elle améliore souvent les performance (erreurs de classification) et gagne en stabilité.

## 5.6. Random Forest : Forêts aléatoires

Les forêts aléatoires sont une modification du bagging qui ajoute de l’aléa afin de décorréler les différents arbres.

Algorithme 15.1 Forêt aléatoire pour la régression ou la classification.

**Pour** b de 1 à B :

**Tirer** un échantillon bootstrap de taille à partir des données d'entraînement.

**Construire** un arbre de forêt aléatoire avec les données bootstrap, en répétant de manière récursive les étapes suivantes pour chaque nœud terminal de l'arbre, jusqu'à ce que la taille minimale du nœud soit atteinte.

**Sélectionner** variables au hasard parmi les variables.

**Choisir** la meilleure variable (point de division) parmi les .

**Diviser** le nœud en deux nœuds fils.

**Produire** l'ensemble d'arbres .

Pour effectuer une prédiction sur un nouveau point :

* **Régression**
* **Classification** : Soit   la prédiction de classe de l'arbre de forêt aléatoire . Alors   est le vote majoritaire de .

# 6. Boosting

## 6.1. Motivations

La nécessité de combiner des classifieurs découle de deux principaux avantages :

* Obtenir une décision plus fiable en agrégeant l'avis de plusieurs experts
* Décomposer la complexité d'un problème en construisant plusieurs sous-problèmes plus faciles à comprendre et à résoudre (diviser pour mieux régner)

## 6.2. Algorithme de boosting

**Début** (Données : Base d'apprentissage D : {(xn , yn ) n = 1, ...., N } )

**Estimer** un classifieur faible φ1 (x, D1) avec D1 sous-base d'apprentissage de D de taille N1 < N

**Estimer** un classifieur faible φ2 (x, D2) avec D2 sous-base d'apprentissage de D − D1 de taille N2 dont la moitié des échantillons sont mal classés par φ1

**Estimer** un classifieur faible φ3 (x, D3) avec D3 sous-base d'apprentissage de D − D1 − D2 de taille N3 où les échantillons mettent en désaccord φ1 et φ2

**Calculer** φBo (x) = vote (φ1 (x) , φ2 (x) , φ3 (x))

**Renvoyer** Classifieur φBo (x)

Fin

## 6.3. Algorithme AdaBoost

**Début**

D = {(x1,y1),...,(xN,yN)}, avec yn ∈ {−1,+1}, n = 1,...,N

**Pour** n allant de 1 à N **faire**

p0(xn) = 1/N

t = 0

**Pour** t ≤ T **faire**

**Tirer** un échantillon d’apprentissage Dt dans D selon proba pt

**Apprendre** une règle de classification ht sur Dt par algo A

Soit εt l’erreur apparente de ht sur Dt

αt ← 1/2 ln((1−εt)/εt)

**Pour** n allant de 1 à N **faire**

**Si** ht(xn) = yn (bien classée par ht)

P(t+1)(xn) = pt(xn)e^(-αt)

**Fin Si**

**Si** ht(xn) ≠ yn (mal classée par ht)

P(t+1)(xn) = pt(xn)e^(αt)

**Fin Si**

(Zt est une valeur de normalisation, P pt+1(xn) = 1)

**Fin Pour**

t←t+1

**Fin Pour**

**Renvoyer** H(x) = sign(Σt=0 à T αtht(x))

**Fin**

Une image contenant Rectangle, conception

Description générée automatiquementUne image contenant Rectangle, carré, conception

Description générée automatiquement

Exemple d’application d’Adaboost.

Une approche consiste à éliminer les outliers avant la phase d'apprentissage, mettant en œuvre des techniques de réduction de données pour assurer une qualité optimale. Une autre alternative est de gérer la présence des outliers pendant l'apprentissage.

Les avantages sont :

* Méta-algorithme d'apprentissage : L'algorithme Adaboost est un méta-algorithme d'apprentissage qui a la flexibilité d'utiliser n'importe quel algorithme d'apprentissage faible comme composant de base.
* Paramètre unique à régler : Un seul paramètre, le nombre d'itérations, nécessite un réglage, simplifiant ainsi le processus de configuration.
* Facilité de programmation : Sa conception intuitive le rend facile et accessible à programmer, facilitant son implémentation.
* Performances théoriques garanties : Adaboost offre des garanties théoriques en termes de performances, assurant une convergence vers des modèles performants.

Les inconvénients sont :

* Difficulté d'incorporation de connaissances a priori : L'incorporation de connaissances préalables dans le processus d'apprentissage avec Adaboost peut être complexe.
* Difficulté de régularisation : La régularisation du modèle peut s'avérer délicate, posant des défis lors de la gestion de la complexité du modèle.
* Choix délicat de l'apprenti faible : Sélectionner le meilleur algorithme d'apprentissage faible n'est pas toujours évident et peut nécessiter une exploration minutieuse.
* Frontières de décision non interprétables : Les frontières de décision générées par Adaboost peuvent souvent être complexes et difficiles à interpréter, limitant la compréhension du modèle dans certains cas.