CRISP-DM Cross Industry Standard Process - Data Mining

REST Representational State Transfer

SVM Support Vector Machines

UAT Univesity Administrative Tools

CHAID Chi-square Automatic Interaction Detector

Murhabazi Buzina Espoir Tech 2 GI ULPGL 2017

Sujet : DataMining appliqué à la prédiction de l'orientation des élèves finalistes du secondaire à l'université

INTRODUCTION

0.1 Problématique

Arrivés à la fin de leurs études secondaires la plupart des élèves finalistes du secondaire et, futurs universitaires sont confrontés au problème de choix de filières pour poursuivre leur études universitaires. Plusieurs options s'offrent à eux et ainsi ils se trouvent dans un embarras de choix, la plupart d'entre eux choisissent mal leurs orientations. C'est pour cela que nous remarquons qu'en première année d'université,le pourcentage d'échec ou d'abandon est très élevé suite à une mauvaise orientation des étudiants.[1]

En effet,améliorer le processus d'orientation des nouveaux étudiants à leur entré à l'université pourrait diminuer le taux d'échec et d'abandon en première année .

Il s'avère donc important de doter les universités des outils d'aide à la décision qui pourront leur permettre de bien orienter les étudiants avant d'entreprendre leurs études universitaires et ainsi leur permettre d'y tirer pleine satisfaction.

D'où nous nous sommes posés les questions suivantes :
- *Comment utiliser les techniques du DataMining pour doter les universités des outils d'aide à la décision les permettant de bien orienter les étudiants dès leur entrée à l'université ?*
- *Comment peut-on aider les élèves finalistes du secondaires à pouvoir faire le choix de leur filières à l'université ?*
- *Les techniques du DataMining peuvent - elles apporter leur contribution dans ce domaine ?*

0.2 Intérêts et Motivations du Sujet

Nous avons choisi de parler du sujet sur le DataMining à cause de notre passion pour les mathématiques et les sciences prédictives mais aussi car au 21

ème siècles l'immensité des données générées par les systèmes d'information ne cesse de croitre d'où il s'avère important de les analyser et apprendre de ces données !
Mais aussi l'intérêt scientifique de ce travail sera de fournir un outil de base à tous chercheur qui aimera aussi travail dans ce domaine dans les jours à venir.
Nous avons choisi de parler de l'orientation des étudiants car durant notre parcours universitaire nous avons constaté un fort taux d'abandon et d'échec due à une mauvaise orientation et ainsi nous voulions aider tant soit peut à résoudre ce problème.

0.3 Hypothèses de Travail

Une hypothèse est une supposition qui est faite en réponse à une question de recherche. [2]
Pour notre travail nous supposerons que les universités ainsi disposent d'une énorme quantité des données et qu'on peut les analyser enfin d'y découvrir des patterns cachés qui peuvent nous permettre de prédire l'orientation d'un nouvel étudiant sur base de ses résultats à l'école secondaire.

0.4 Objectifs du travail

L'objectif d'une recherche se divise en deux parties : l'objectif général concerne la contribution que les chercheurs espèrent apporter en étudiant un problème donné ; les objectifs opérationnels concernent les activités que les chercheurs comptent mener en vue d'atteindre l'objectif général. [2]
Pour notre travail l'objectif général sera de fournir un outil d'aide à la décision en nous basant sur les techniques du DataMining plus précisément les arbres de décisions et la technique du random forrest.
Cet outil se présentera sous forme de service web de type Representational State Transfer (REST) que les universités peuvent intégrer facilement dans leurs systèmes d'informations.

0.5 Méthodes et Techniques de Recherche

Pour atteindre notre objectif,notre travail utilisera la méthodologie de Cross Industry Standard Process - Data Mining (CRISP-DM) qui est le processus standard d'un projet DataMining, elle définit les étapes pour la

conduite d'un projet en le rendant plus efficace plus rapide et moins couteux. [3]
Cette méthodologie se base sur la technique documentaire qui consister à consulter les archives des universités et ainsi que le système d'information Univesity Administrative Tools (UAT)en vue d'y collecter les données.
Elle utilise également les enquêtes ainsi que les interviews auprès des conseillers d'orientations d'universités pour savoir comment se déroule l'activité de l'orientation des étudiants.

0.6 Subdivision du travail

Hormis l'introduction et la conclusion ce travail sera subdivisé en cinq chapitres.

1. Le premier chapitre sera intitulé : Les généralités sur le DataMining.
2. Le Second sera intitulé : Analyse du domaine de l'orientation.
3. Le troisième sera intitulé : Présentation et exploitation des données obtenus.
4. Le quatrième sera intitulé : Conception ,développement et Présentation de notre Solution.
5. le Cinquième sera intitulé : Planification du Projet .

Chapitre 1

Généralités sur le DataMining

Nous sommes submergées des données et leur quantité augmente du jour au lendemain . Les ordinateurs, les smart-phones et de plus en plus des équipements connectées nous submergent et sont omniprésents dans notre vie et cela nous permet de générer des très grandes quantités des données , toutes nos décisions , nos choix dans les supermarchés,nos habitudes financières sont sauvegardées dans d'énormes bases des données.

L'internet est aussi submergé des informations c'est ce qui fait que chaque choix, chaque clic que nous faisons soit sauvegardé , ceci n'étant que des choix personnels mais qui ont d'innombrables contreparties dans le monde du commerce et de l'industrie . Mais paradoxalement on a pu constaté que plus les données augmentent et sont générées de moins en moins les personnes les comprennent et ainsi un immense fossé s'est créé entre le volume des données générées et la capacité de compréhension de celles ci. D'où l'importance de mettre en place des méthodes et techniques qui faciliterons l'analyse et l'obtention des informations considérables de ces données.

1.1 Définitions

Dans cette partie nous allons définir 3 termes qui portent souvent à confusion : L'intelligence Artificielle,Le Machine Learning ou apprentissage automatique, La fouille des données ou Le DataMining, nous tenterons de dégager à la fin les différences et les ressemblances entre ces termes.

1.1.1 L'intelligence Artificielle

L'intelligence artificielle(IA) est un domaine de l'informatique dédié à la création de matériel et de logiciels capables d'imiter la pensée humaine. Le

but principal de l'intelligence artificielle est de rendre les ordinateurs plus intelligents en produisant des logiciels permettant à un ordinateur d'émuler des fonctions du cerveau humain dans des applications définies. L'idée n'est pas de remplacer l'être humain mais de lui donner un outil plus puissant afin de l'aider à accomplir ses tâches.[4]
L'Intelligence artificielle est un domaine de l'Informatique qui a pour but de développer des machines (= ordinateurs) "intelligentes", c'est-à-dire capables de résoudre des problèmes pour lesquels les méthodes conventionnelles sont inefficaces et inapplicables. [4]

1.1.2 Le Machine Learning

Le Machine Learning est définie comme une branche de l'intelligence artificielle qui se penche sur la création des algorithmes qui peuvent apprendre et faire des prédictions à partir des données[5]

D'autres auteurs le définissent comme :
- Un type de l'intelligence artificielle qui donne aux machines la possibilité d'apprendre sans être explicitement programmer[6]
- La science qui consiste à la création des logicielles qui apprennent d'eux même en fonction des données.[6]

1.1.3 Le DataMining

Le DataMining ou la fouille de données est une technique consistant à rechercher et extraire de l'information (utile et inconnue) de gros volumes de données stockées dans des bases ou des entrepôts de données en utilisant les techniques du Machine Learning .[5]

Certains auteurs comme M. BATER[7] le définissent comme une discipline scientifique qui a pour but l'analyse exploratoires des grandes quantités des données , la découvertes des modèles utiles ,valides , inattendus ainsi que la connaissances compréhensibles dans celles ci. Outre la découvertes de l'information il englobe la collecte , le nettoyage et le traitement de ces données.

Le datamining peut aussi être défini comme un processus inductif, itératif et interactif de découverte dans les bases de données larges de modèles des données valides, nouveaux, utiles et compréhensibles.[8] - Itératif : nécessite plusieurs passes
- Interactif : l'utilisateur est dans la boucle du processus
- Valides : valables dans le futur
- Nouveaux : non prévisibles

- Utiles : permettent à l'utilisateur de prendre des décisions
- Compréhensibles : présentation simple

1.1.4 Différence entre le DataMining , le Machine Learning, ainsi que les statistiques . [6] [9] [10]

A première vue on pourrait constater que ces 3 disciplines n'ont aucune différence car tous traitent de la même question : comment apprendre des données ?, couvrent les mêmes matières et utilisent les mêmes techniques qui sont entre autres : la régression linéaire, la régression logistique , le réseau des neurones , le Support Vector Machine ,....

Mais en analysant de prêt on remarque qu'ils ont des différences surtout concernant les sujets et matières sur lesquels ils insistent, ou en d'autres termes quand et comment les méthodes qu'ils ont en commun sont utilisées.

- Les statistiques insistent sur les méthodes de la statistique référentielle,descriptives , multivariés (intervalle de confiance, test des hypothèses , estimation optimale) dans un problème à faible dimension (une petite quantité des données) et elles font des prédiction sur base de celle ci.[9]

- Le Machine Learning est beaucoup plus concentré vers le Génie logiciel il est beaucoup plus focalisé sur la construction des logicielles qui font des prédiction à partir des données apprennent de l'expérience. On dit qu'un programme apprend de l'expérience si ses performances sur les taches qu'il exécute augmentent avec l'expérience . ces performances sont évaluées à l'aide de certaines métriques que nous verront dans la suite.
Il nécessite l'étude des algorithmes qui peuvent extraire l'information utile dans des grosses quantités des données automatiquement , dans certaines mesures il s'inspirent des statistiques. [10]

-Le DataMining quand à lui s'inspire des techniques du Machine learning , des statistiques souvent avec un objectif bien fixé en vue de découvrir des patterns cachées dans des grosses volumes des données. - L'intelligence artificielle se focalise sur la création des agents intelligents , en pratique il consiste à programme un ordinateur pour qu'il simule le fonctionnement du cerveau humain . Qu'il exécute certaines taches comme un homme parmi ces taches figure la faculté d'apprendre sur base de l'expérience.

1.2 Différentes Techniques du Machine Learning [11]

Dans cette partie nous allons passer en revue quelques méthodes et algorithmes utilisées dans le machine learning. Notons que ces méthodes se divisent en 2 groupes : l'apprentissage supervisée et l'apprentissage non supervisé.Concernant l'apprentissage supervisé nous parlerons de la régression linéaire , la régression, le réseau des neurones, le SVM(Support Vector Machine) , les arbres de décision et nous finirons par le Random Forrest qui n'est qu'une amélioration des arbres de décisions, et pour l'apprentissage non supervisé nous allons parler de K-means.

1.2.1 Régression Linéaire

La régression est une technique consistant à prédire la sortie d'une observation en partant d'un certain nombre des variables en entrée.
On part d'un certain nombre des données d'apprentissage ou d'un training set avec :
m : nombres d'éléments de l'ensemble d'apprentissage
x_1 ,x_2,x_3 .... x_n : les variables d'entrées(qui peuvent varier de 1 (pour la régression linéaire avec une seule variable) à n pour plusieurs variables).
y : la variable de sortie on notera le (x_1,x_2,x_3 .... x_n, y) : un exemple d'apprentissage et
(x_1^i,x_2^i,x_3^i.....x_n^i,y^i) un exemple choisie le ème exemple d'apprentissage avec i comme index sur nos données.

Le but de la régression c'est de trouver la fonction qui permet de prédire la sortie en fonction de l'entrée cette fonction doit minimiser l'erreur entre les valeurs de la fonction aux points d'apprentissage et la valeur de sortie dans les données d'apprentissage .
Cette fonction on l'appelle hypothèse qui sera une droite ou un polynôme selon qu'on utilise la régression linéaire ou polynomiale.
L'hypothèse se présente de la manière suivante :

1 ère cas régression avec un seule variable :

Par exemple on tente de prédire le prix d'une maison en fonction de sa superficie :

$h_\theta(x) = \theta_0 + \theta_1 x$.

Cela signifie que y est une fonction linéaire de x avec θ_i des paramètres qu'on cherchera à déterminer . on peut le remarque sur la figure suivante ou notre hypothèse est une droite :

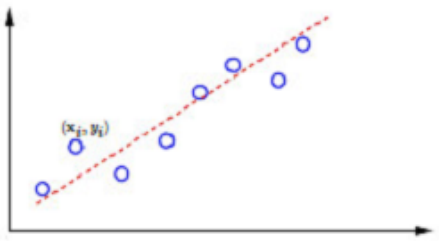


FIGURE 1.1 – Régression Linéaire avec une variable

2 ème cas régression avec plusieurs variables :

Pr exemple prédire le prix d'une maison cette fois en fonction de la superficie, du nombre des chambres ,et de l'année de construction

$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + ...\theta_n x_n$ Dans ce cas notre hypothèse sera en fonction des variables en entré un plan ou un hyperplan.

3 eme cas la regression polynomiale :

Dans ce cas au lieu d'utiliser une droite on utilise un polynôme à n degree qui es donnée par :

$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3 + ...\theta_n x_n^n$ le travail restant sera de trouver

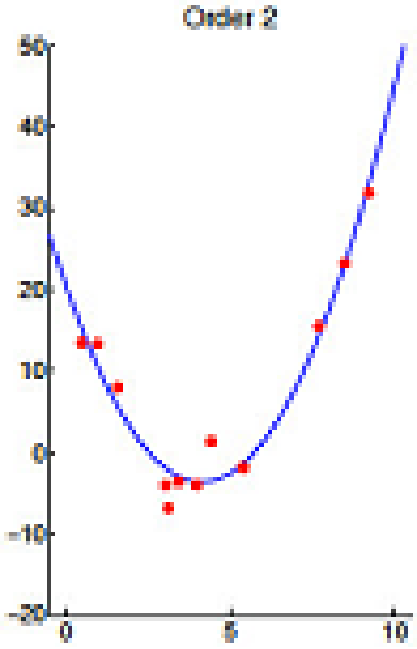


FIGURE 1.2 – Régression Polynomiale avec un polynôme de degré 2

les paramètres θ_i pour notre fonction $h_\theta(x)$ mais comment les trouver ?

Fonction Cout et calcul de L'erreur

On appelle l'erreur de prédiction , la valeur définie par :

$J(\theta_1, \theta_2, ...\theta_n) = \frac{1}{2m} \sum_{=1}^{m} \left[h_\theta\left(x^{(i)}\right) - y^{(i)}\right]^2$ avec m : le nombre des nos données d'apprentissage. Cette fonction représente l'erreur commise lors de la prédiction avec notre hypothèse par rapport à la valeur exacte. Les θ_i sont les valeurs qui minimisent cette erreur sur toutes les données de notre apprentissage.

Il existe différents techniques de minimisation de cette fonction entre autre :

- L'annulation de la dérivée première (trouver les valeurs θ_i qui annulent notre dérivée première) .Cette méthode à pour inconvénient le fait qu'il

convient pas pour les données d'apprentissage avec plusieurs attribues et plusieurs données.

- une autre méthode c'est la descente du gradient : celle ci consiste à effectuer plusieurs itérations sur les valeurs de θ_i jusqu'à trouver celle qui minimise l'erreur (jusqu'à ce qu'il converge vers zéro)

voici l'algorithme utilisée : 1

Algorithm 1 Algorithme de la descente du gradient

$\theta_i \leftarrow 0$

while J ne converge pas **do** ▷ faire pour chaque tuple

$\theta_i \leftarrow \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta_1, \theta_2, ...\theta_n)$

avec α : learning rate qui est compris entre [0,001 ;0,1] il sert à réguler notre algorithme

s'il est trop grand θ ne converge pas s'il est trop petit θ converge après plusieurs itérations. Dans la plupart des cas J converge après un nombre d'itérations élevé comme on peut le remarqué sur la figure suivante :

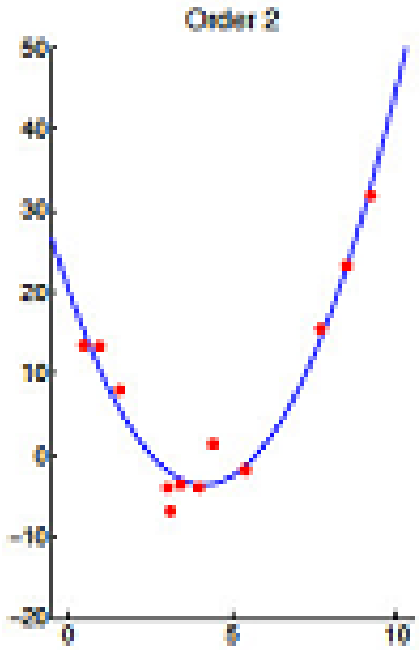


FIGURE 1.3 – Variation de J en fonction du nombre des itérations

NB : La Normalisation

Dans la pratique on peut avoir plusieurs données qui ne sont pas dans la même échelle

par ex : on cherche à prédire le prix d'une maison en fonction de la surface x_1 : compris entre 30 – 400 m^2, le nombres des chambres x_2 : 1-10,
Nous remarquons que nos 2 variables ne sont pas dans la même intervalle et cela peut présenter un problème de convergence et ainsi empêcher les θ_i de converger rapidement . pour limiter ce problème on effectue un normalisation elle consister à remplacer les x_i pour chaque tuple par :

$x_j^i = \frac{x_j^i - \mu(x_j)}{\sigma(x_j)}$ avec :

$\mu(x_j)$: la moyenne des termes x_j et $\sigma(x_j)$: la variance des termes x_j

1.2.2 Régression Logistique

La régression logistique est une technique de classification , elle consiste à appliquer une fonction h sur des éléments x_1 ,x_2,x_3 .... x_n en entrée et trouver une sortie *discrète* y.
Cette sortie nous permet de déterminer la classe du tuple , généralement y prend 2 valeurs 0 ou 1 et quelque fois -1 ou 1.
On utilise ce type de classification dans différents cas :

- Classification des emails : spams ou non spam
- Transaction en Ligne : Frauduleux ou pas
- Crédit bancaire : risquant ou non
- Tumeur : Bénigne ou Maligne

Représentation de l'hypothèse

Une première approche serait d'utiliser la régression linéaire avec la fonction $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + ... \theta_n x_n$ pour effectuer la classification mais cela peut un problème car $h_\theta(x)$ peut prendre des valeur qui sont à l'extérieur de l'intervalle [0,1]. La meilleur solution serait de trouver une fonction qui ne prend que des valeurs comprises entre 0 et 1 .

Pour les problèmes de classification on utilise la fonction sigmoïde ,elle est définie par :

$$g(z) = \frac{1}{1+e^{-z}}$$

Et ainsi notre hypothèse sera :

$$h_\theta(x) = g(\theta^T x)$$

Avec $\theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + ... \theta_n x_n$. Voici comment se présente cette fonction :

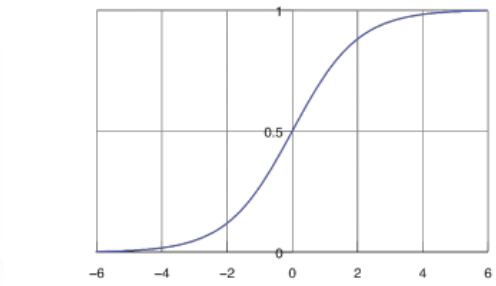


FIGURE 1.4 – Fonction Sigmoïde

On remarque que cette fonction ne prend que des valeurs comprise entre 0 et 1.

En d'autres termes $h_\theta(x)$ représente la probabilité que y=1 sur l'entrée x_n .

$$h_\theta(x) = P(y = 1|x; \theta)$$

La probabilité que y soit égale à 1 sachant x_n paramétré par θ.

On l'utilise pour effectuer une classification en supposant que :

- si $h_\theta(x) > 0.5$ la classe prédite est 1.
- si $h_\theta(x) < 0.5$ la classe prédite est 0.

On démontre que :

$$P(y = 1|x; \theta) + P(y = 0|x; \theta) = 1 \text{ et}$$
$$P(y = 0|x; \theta) = 1 - P(y = 1|x; \theta)$$

Frontière de Décision

Analysons notre fonction $g(z)$, on remarque qu'il est égale à 0.5 pour z = 0. Donc si z est positif $g(z)$ sera supérieur à 0.5, dans le cas contraire $g(z)$ sera inférieur à 0.5. Et ainsi avec notre hypothèse $h(\theta^T x) = 1$ si $\theta^T x > 0$ et $h(\theta^T x) = 0$ si $\theta^T x < 0$ On comprend qu'a partir du plan $\theta^T x$ qui divise l'espace en 2 partie on peut prédire la classe de chaque tuple sans problème .

Cette droite ou hyperplan s'appelle ***la frontière de décision*** . Voici comment elle se présente si on n'a que 2 attributs x_1 et x_2 :

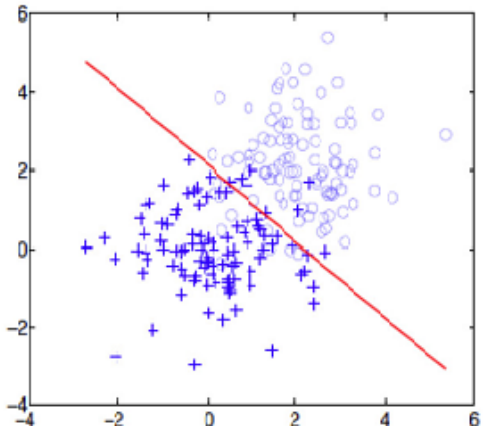


FIGURE 1.5 – Frontière de décision pour la Régression Logistique à 2 variables

La Fonction Coût : Calcul De L'erreur

On pourrait être tenter d'utiliser la même fonction coût que pour la régression linéaire

$$J(\theta_1, \theta_2, ...\theta_n) = \frac{1}{2m} \sum_{=1}^{m} \left[h_\theta\left(x^{(i)}\right) - y^{(i)}\right]^2$$

Mais cette fonction présente des problèmes de convergence à cause de la non linéarité de la fonction $h_\theta(x)$. Pour ce faire,on définit une nouvelle fonction de l'erreur :

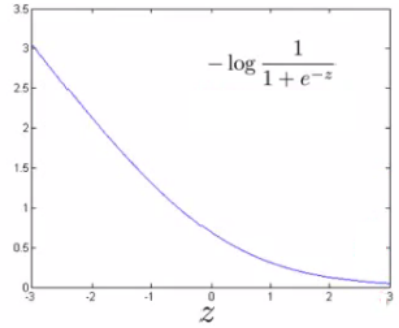
$$Err(h_\theta(x), y) = \begin{cases} -\log[1 - h_\theta(x)], & \text{si } y{=}0 \\ -\log[h_\theta(x)], & \text{si } y{=}1 \end{cases}$$

Analyse de l'erreur :

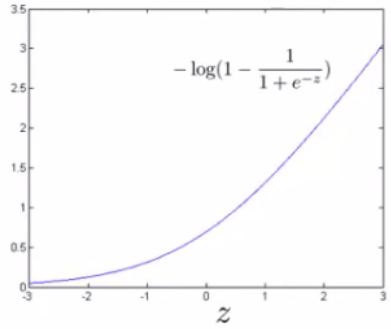
- si y= 1 et notre fonction $h_\theta(x) = 1$, alors l'erreur vaut 0 et augmente si $h_\theta(x)$ décroit.

- si y= 0 et notre fonction $h_\theta(x) = 0$, alors l'erreur vaut 0 et augmente si $h_\theta(x)$ croit .

On peut facilement le remarquer sur les figures suivantes :
Notre fonction cout peut prendre la forme compacte suivante :



(a) cas ou Y =1



(b) Y = 0

FIGURE 1.6 – Erreur pour la régression logistique

$$Err(h_\theta(x), y) = -y \log[h_\theta(x)] - (1 - y) \log[1 - h_\theta(x)]$$

Et Donc notre Fonction cout pour les θ sera :

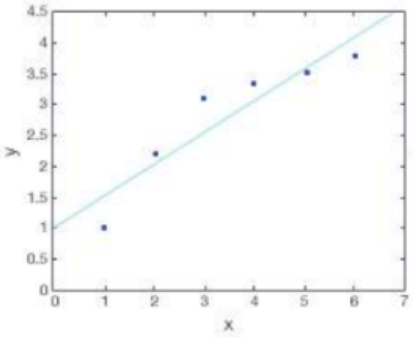
$$J(\theta) = \frac{1}{2m}[\sum_{=1}^{m} -y^{(i)} \log[h_\theta\left(x^{(i)}\right)] - (1 - y^{(i)}) \log[1 - h_\theta\left(x^{(i)}\right)]]$$

Avec $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

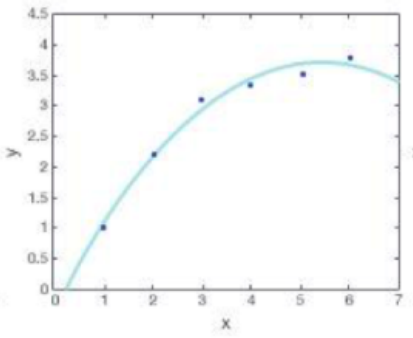
Pour le minimiser,on recours aux mêmes techniques que pour la régression linéaire et c'est le même algorithme de la descente du gradient qu'on utilise dans ce cas aussi.

Régulation : Phénomène de sur-apprentissage

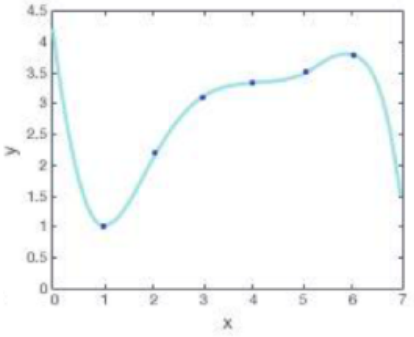
Nous venons de passer en revue quelques modèles pour effectuer la régression linéaire ainsi que la classification avec la régression logistique. Jetons un coup d'œil aux 3 images de la 'FIGURE 7
A la première figure (a) : avons la régression linéaire comme hypothèse mais



(a) $h_\theta(x) = \theta_0 + \theta_1 x$



(b) $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$



(c) $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

FIGURE 1.7 – a : Sous-apprentissage et c : Sur-apprentissage avec la régression linéaire

on remarque que cela n'est pas un bon modèle de prédiction car dans ce cas l'erreur est trop grande.on parle de *sous-apprentissage ou Underfitting*. A la deuxième figure notre hypothèse c'est un polynôme du second dégrée le modèle convient bien à notre ensemble d'apprentissage et dans ce cas l'erreur est moins élevée. Par contre sur la troisième cas l'hypothèse c'est un polynôme du quatrième dégrée qui convient parfaitement à notre ensemble d'apprentissage et l'erreur est nulle, mais c'est modèle n'est pas bon car il convient parfaitement sur notre ensemble d'apprentissage et n'est pas adaptable aux

nouvelles données, elle n'est pas une solution générale.Dans ce cas on parle de *Sur-Apprentissage ou Overfitting* .

On remarque ce même phénomène avec la régression logistique .

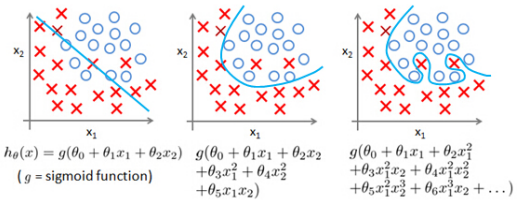


FIGURE 1.8 – Overfitting et Underfitting avec la régression logistique

Solutions aux problèmes de sur apprentissage

Signalons qu'une des cause de ce phénomène c'est le fait d'avoir peut des données avec plusieurs attributs .

Dans ce cas la première de solution consisterait à réduire le nombre des attribues , choisir celle qu'on doit retenir et laisser les autres,mais cette approche possède un inconvénient car on risquerait de perdre l'information apportée par ces attributs et cela pénaliserait notre modèle .

La seconde solution est celle qu'on appelle la *La régularisation* elle consiste à garde tous les attribues mais réduire l'ampleur des paramètres θ dans le calcul de l'erreur . On remarque que cette technique fonctionne très bien lorsqu'on dispose de plusieurs attribues qui contribue à la prédiction de la valeur y. Ainsi notre fonction de cout pour le calcul de l'erreur de la régression linéaire sera :

$$J(\theta_1, \theta_2, ...\theta_n) = \frac{1}{2m}[\sum_{=1}^{m} \left[h_\theta\left(x^{(i)}\right) - y^{(i)}\right]^2 + \lambda \sum_{=1}^{n} \theta_j^2]$$

Et Pour la régression logistique elle sera :

$$J(\theta) = \frac{1}{2m}[\sum_{=1}^{m} -y^{(i)} \log[h_\theta\left(x^{(i)}\right)] - (1 - y^{(i)}) \log[1 - h_\theta\left(x^{(i)}\right)] + \lambda \sum_{=1}^{n} \theta_j^2]$$

Avec λ le facteur de régularisation. C'est cette nouvelle fonction cout qu'on utilise pour l'algorithme de la descente du gradient régularisé.

1.2.3 Le Réseaux des Neurones

Pourquoi avons nous besoins du réseau des neurones ? Supposons que nous avons un problème complexe de classification, la première approche serait d'utiliser la classification logistique avec un polynôme de plusieurs termes :

- cela conviendrait pour problème avec 1, ou 2 attribut.

- supposons maintenant que nous avons des tuples avec plus de 1000 attributs ,

Ex : On veut prédire la chance d'une maison d'être vendu dans 6 mois : - Plusieurs facteurs entrent en jeu ,à peut près plus de 100 , on aura un polynôme avec les termes suivants $(x_1^2, x_1x_2, x_1x_3, x_1x_4, ...x_1x_{100})$, on aura a peut près 5000 termes et ce nombre augmentera énormément si on choisi de considérer les 3 ème degré. La régression logistique n'est pas vraiment approprié pour des problèmes de classification des données avec plusieurs attributs .

Par exemple dans le cas de la vision artificielle ou n peut aller de 2500 à 50000000 attributs. On remarque bien qu'une simple régression logistique n'est pas approprié pour ce cas.

Les Neurones et le cerveau[12]

Une des motivations qui ont poussé les scientifique à créer le réseaux des neurones c'est celui de répliquer le fonctionnement du cerveau humain . En effet le cerveaux humain dispose d'une capacité énorme d'apprentissage il s'apprend lui même comment apprendre et peut apprendre de n'importe quelle source des données,il contient environ 100 milliards de neurones. Voici à quoi ressemble à quoi ressemble un neurone dans le cerveau :

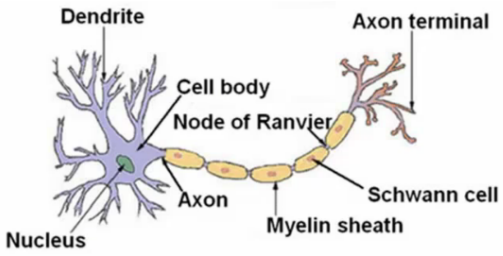


FIGURE 1.9 – Un Neurone Naturel

Les neurones reçoivent les signaux (impulsions électriques) par des extensions très ramifiées de leur corps cellulaire (les dendrites) et envoient l'information par de longs prolongements (les axones). Les impulsions électriques sont régénérées pendant le parcours le long de l'axone. La durée de chaque

impulsion est de l'ordre d'1 ms et son amplitude d'environ 100 mvolts. Les contacts entre deux neurones, de l'axone à une dendrite, se font par l'intermédiaire des synapses. Lorsqu'un potentiel d'action atteint la terminaison d'un axone, des neuromédiateurs sont libérés et se lient à des récepteurs post-synaptiques présents sur les dendrites. L'effet peut être excitateur ou inhibiteur. Chaque neurone intègre en permanence jusqu'à un millier de signaux synaptiques. Ces signaux n'opèrent pas de manière linéaire (effet de seuil).

Représentation d'un Neurone Artificiel

On pourrait définir un neurone ou perceptrons comme étant une unité de traitement qui reçoit des données en entrée sous forme vectorielle et produit une sortie réelle , cette sortie est fonction des entrées et des poids de connexions.

Il se caractérise par :

- les signaux en entrée $x_1, x_2, ..., x_n$,

il est toujours préférable d'ajouter le biais x_0 qui est égale à 1.,

- les coefficients synaptiques ou poids des connexions $\theta_{i0}, \theta_{i1}, ...\theta_{in}$,
- Une Fonction d'activation $h_\theta(x)$,
- l'état interne d'activation $a = h_\theta(x)$

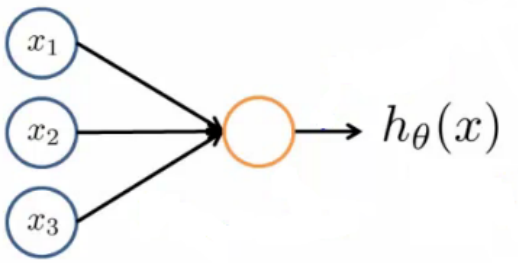


FIGURE 1.10 – un Perceptron ou Unité de traitement

D'une façon plus générale un réseau des neurones est un ensemble de plusieurs neurones liés entre eux pour effectuer un calcul se compose :

- D'une couche d'entré ou activation layer
- D'une couche de sortie ou output layer
- D'une ou plusieurs couches intermédiaires ou hidden layers .

Chaque couche dispose des perceptrons ou unités d'activation.

On note :

- a_i^j : l'unité d'activation i de la couche j

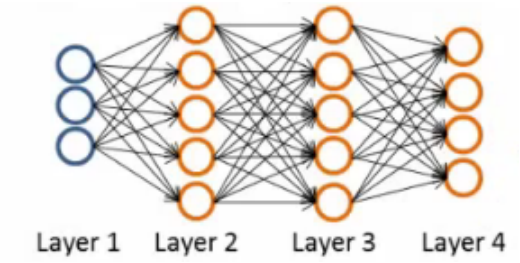


FIGURE 1.11 – un réseau des neurones complets avec 2 hidden layers

- θ^l : matrice des poids de connexions contrôlant le passage de la couche l à couche l+1.
La matrice θ^l est constituée des termes θ_{ij} avec :
- i l'indice de l'unité de la couche de destination
- j l'indice de l'unité de la couche d'origine .
Donc θ_{12} représente le poids de passage de l'unité 2 de la couche d'origine a l'unité 1 de la couche de destination.
Regardons encore une fois l'image suivante :

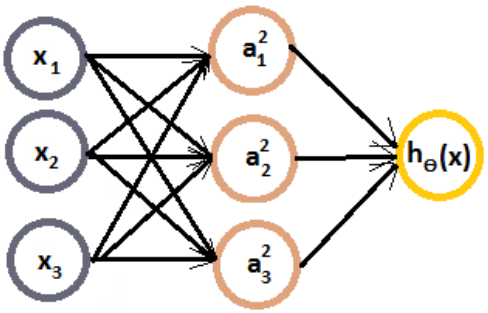


FIGURE 1.12 – Réseaux des neurones avec des unités d'activation

Nous avons :
$a_1^{(2)} = g(\theta_{10}^{(1)}x_0 + \theta_{11}^{(1)}x_1 + \theta_{12}^{(1)}x_2 + \theta_{13}^{(1)}x_3)$
$a_2^{(2)} = g(\theta_{20}^{(1)}x_0 + \theta_{21}^{(1)}x_1 + \theta_{22}^{(1)}x_2 + \theta_{23}^{(1)}x_3)$
$a_3^{(2)} = g(\theta_{30}^{(1)}x_0 + \theta_{31}^{(1)}x_1 + \theta_{32}^{(1)}x_2 + \theta_{33}^{(1)}x_3)$
$h_\theta(x) = a_1^{(3)} = g(\theta_{10}^{(2)}a_0^{(2)} + \theta_{11}^{(2)}a_1^{(2)} + \theta_{12}^{(2)}a_2^{(2)} + \theta_{13}^{(2)}a_3^{(2)})$

Chaque unité d'activation de la couche l+1 est obtenue en appliquant la fonction g (une sigmoïde) aux à la combinaison linéaire des unités de la couche précédente avec les éléments de la matrice θ^l.

Apprentissage par Réseau des neurones

Le réseau des neurones est un des plus puissant algorithme d'apprentissage , il cherche à trouver les paramètres du modèle d'apprentissage uniquement à partir de l'ensemble d'apprentissage .
Voici notre configuration :

- Notre ensemble d'apprentissage est : $(x^1, y^1), (x^2, y^2), (x^3, y^3), ...(x^n, y^m)$
- L nombre des couche de notre réseau . Pour notre cas L=4
- s_l nombre des unités dans la couche l

Donc en se référant à la FIGURE 11 On remarque que :

- L= 4
- $s_1 = 3$
- $s_2 = 5$
- $s_3 = 5$
- $s_4 = 4$

Avec le réseaux des neurones on peut faire aussi bien des classification binaires que des classifications avec plusieurs classe.
Pour la classification binaire la couche de sortie ne comprend qu'une seule unité k = 1 avec k le nombre des unités de la couche de sortie.
Pour une classification multi-classe k prend des valeurs supérieurs à 3,
la sortie $y \in R^K$ Pour 3 classes $y^1 = [1, 0, 0]$, $y^2 = [0, 1, 0]$, $y^3 = [0, 0, 1]$;

Fonction Cout

Rappelons que pour la régression logistique la fonction cout régularisée est la suivante :

$$J(\theta) = \frac{1}{2m}[\sum_{i=1}^{m} -y^{(i)} \log[h_\theta\left(x^{(i)}\right)] - (1 - y^{(i)}) \log[1 - h_\theta\left(x^{(i)}\right)] + \lambda \sum_{j=1}^{n} \theta_j^2]$$

Pour le réseau des neurones la fonction cout n'est que la généralisation de celui de la régression logistique mais au lieu d'une sortie nous avons K sorties .

$h_\theta(x) \in R^k$ et $(h_\theta(x))_k = k^{eme}$ sortie

$J(\theta) = \frac{1}{2m}[\sum_{i=1}^{m} \sum_{k=1}^{K} -y_k^{(i)} \log[(h_\theta\left(x^{(i)}\right))_k] - (1 - y_k^{(i)}) \log[1 - (h_\theta\left(x^{(i)}\right))_k] + \lambda \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\theta_{ji}^{(l)})^2]$

Une fois la fonction cout connu le reste du travail consiste à entrainer notre réseau , donc de trouver :

- la valeur des unités d'activation a^l pour chaque couche
- Ainsi que celles de $\theta_{ji}^{(l)}$ qui minimisent la fonction cout .

Calcul des Valeurs des unités d'activation *Forward propagation*

Cette technique est celle qui permet de calculer de calculer les valeur des unités d'activation du réseau des neurones en partant de la couche d'entrée et propagé ces valeurs jusqu'à la couche de sortie d'où le nom de *Forward propagation.*
Nous avons déjà établie que :
- $a_1^{(2)} = g(\theta_{10}^{(1)}x_0 + \theta_{11}^{(1)}x_1 + \theta_{12}^{(1)}x_2 + \theta_{13}^{(1)}x_3)$
- $a_2^{(2)} = g(\theta_{20}^{(1)}x_0 + \theta_{21}^{(1)}x_1 + \theta_{22}^{(1)}x_2 + \theta_{23}^{(1)}x_3)$
- $a_3^{(2)} = g(\theta_{30}^{(1)}x_0 + \theta_{31}^{(1)}x_1 + \theta_{32}^{(1)}x_2 + \theta_{33}^{(1)}x_3)$

on définie :
$z_1^2 = \theta_{10}^{(1)}x_0 + \theta_{11}^{(1)}x_1 + \theta_{12}^{(1)}x_2 + \theta_{13}^{(1)}x_3$ et cela veut dire que :
$a_1^{(2)} = g(z_1^2)$
On peut ainsi définir les termes z_2^2 ,et z_3^2 qui sont les unités d'activation de la couche 2.
En utilisant les matrices nous pouvons écrire :

$$z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix} \text{ pour la 2ème couche et l'entrée peut être noté } x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Ainsi $z^{(2)} = \theta^{(1)}x$ et $a^{(2)} = g(z^{(2)})$
Pour la troisième couche écrit : $z^{(3)} = \theta^{(2)}a^2$ et $a^{(3)} = g(z^{(3)})$ et ainsi de suite jusqu'à la couche de sortie .
D'une façon générale on écrit : $z^{(l)} = \theta^{(l-1)}a^{l-1}$ et $a^{(l)} = g(z^{(l)})$ pour toutes les couches du réseau .

Minimisation de la fonction de cout : back propagation

Pour minimiser la fonction cout on doit calculer ses dérivées partielle par rapport aux éléments $\theta_{ji}^{(l)}$ ce qui n'est pas une tache facile . Le but est de trouver les $\theta_{ji}^{(l)}$ qui minimisent la fonction cout. On veut calculer $\frac{\partial}{\partial\theta_{ji}^{(l)}}J(\theta)$
Pour calculer les dérivées partielles on utilise la technique qu'on appelle *back propagation* ou *rétro-propagation du gradient* ,c'est une méthode qui permet de calculer le gradient de l'erreur pour chaque neurone d'un réseau de neurones, de la dernière couche vers la première.
Notons δ_j^l : l'erreur à l'unité j de la couche l.
- Pour la 4eme couche cette erreur n'est que la différence entre la valeur exacte et la valeur calculé par notre réseau des neurones.

D'ou : $\delta_j^4 = a_j^4 - y_j$
Pour calculer l'erreur de la couche l on utilise la couche suivante l+1 d'ou le

non de *rétro-propagation du gradient* Ainsi l'algorithme d'entrainement de notre algorithme sera le suivant : **??**

Algorithm 2 Algorithme de la rétro-propagation du gradient

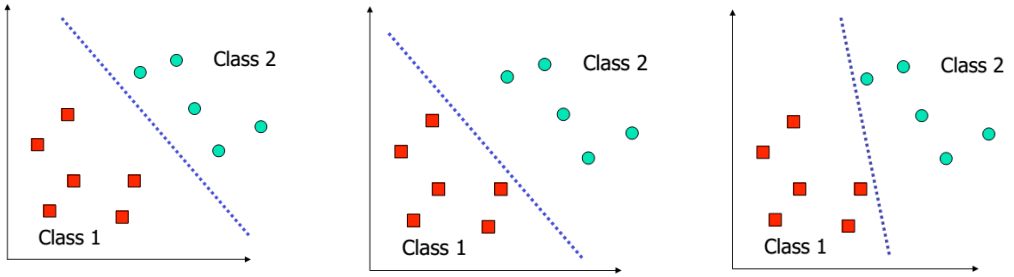
Require: $(x^1, y^1), (x^2, y^2), ...(x^n, y^m)$
  for $i = 1, 2...., s_{l+1}, j = 1, 2, ...s_l; l = 1, 2, ...L$ **do**
    $\theta_{ij}^{(l)} \leftarrow 0$ ▷ initialsation des valeurs des θ pour chaque couche à Zero
  for all tuple dans l'ensemble d'apprentissage **do**
    $a^1 \leftarrow x^i$ ▷ initialisation de la premiere couche avec les valeurs de l'entréé
    repeat
      for L =2,3.....L **do** ▷ Forward Propagation
        $z^{(l)} \leftarrow \theta^{(l-1)} a^{l-1}$
        $a^{(l)} \leftarrow g(z^{(l)})$
      $\delta^L \leftarrow a^{(L)} - y^{(i)}$ ▷ pour la dernière couche
      for L =L-1,....,2,1 **do** ▷ Backward Propagation
        $\delta^l \leftarrow (\theta^{(l)})^T \delta^{l+1} .* (a^{(l)} .* (1 - a^{(l)}))$
        $\frac{\partial J(\theta)}{\partial \theta^{(l)}} \leftarrow \delta^{l+1} . (a^l)^T$
        $\theta^{(l)} \leftarrow \theta^{(l)} + \frac{\partial J(\theta)}{\partial \theta^{(l)}}$ ▷ Descente d gradient
    until $J(\theta)$ converge vers 0
    if $j \neq 0$ **then**
      $D^l \leftarrow \frac{1}{2m} \theta^{(l)} + \lambda \theta^{(l)}$ ▷ Regularisation
    else
      $D^l \leftarrow \frac{1}{2m} \theta^{(l)}$ ▷ Au cas ou il n'ya pas de regularisation

1.2.4 Support Vector Machine - SVM [13]

Support Vector Machine est un classeur discriminant qui est défini par un hyperplan , étant donné un ensemble d'apprentissage l'algorithme de SVM permet de trouver un hyperplan qui va catégoriser les éléments de l'ensemble et d'autres nouveaux éléments en maximisant la marge entre les élements de l'ensemble d'apprentissage à la manière de la frontière de décision pour la régression logistique. On peut dire que SVN est une amélioration de la régression logistique .

Données Séparables linéairement

Si nous considérons le même problème de classification binaire que celui de la régression logistique sauf que les label de nos classe ici sont 1 et -1 au lieu de 0 et 1 donc ici y $\in$ {-1,1} , on constate que nous avons un large variétés des choix pour notre frontière de décision comme on peut le voir à la figure ci dessous : Nous remarquons que toutes ces frontières des décisions



(a) (b) (c)

FIGURE 1.13 – Différentes frontières de décision possibles

conviennent pour notre ensemble d'apprentissage.
Mais comment alors choisir la bonne ? ? celle qui est optimale ?
Remarquons que pour mieux généraliser les données la meilleur frontière de décision est celle qui doit être le plus éloigné que possible des nos données . Le but de SVM revient à trouver cette hyperplan qui constituerai notre frontière de décision.

Représentation de l'hypothèse Rappelons que pour la régression logistique notre hypothèse s'écrivait de la manière suivante :

$$h_\theta(x) = g(\theta^T x)$$

Avec $g(z)$ étant définit comme la fonction sigmoïde.
On a aussi fait remarquer qu'on utilise l'équation de la frontière de décision qui est $(\theta^T x)$ pour effectuer la classification. et de la on déduit que les données de la classe positif vérifierons l'équation $\theta^T x > 1$ et les autres $\theta^T x < -1$.
Ainsi la limite de la classe positif est l'hyperplan d'équation $\theta^T x = 1$ et celle des négatifs est $\theta^T x = -1$ si on tient compte de b comme distance avec l'origine ces droites s'écriront $\theta^T x + b = 1$ et $\theta^T x + b = -1$.

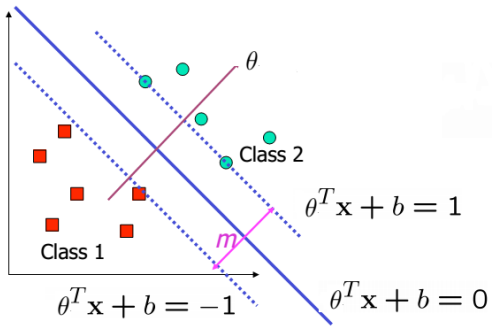


FIGURE 1.14 – Problème de SVM en 2 dimesions

Ainsi le but de SVM serait de trouver l'hyperplan qui maximise la marge ou la distance entre ces 2 hyperplans, et d'après la géométrie analytique cette distance se calcule de la manière suivante $d = \frac{2}{||\theta||}$ Notre problème se formulerait de la manière suivante :

$$\begin{cases} max \frac{2}{||\theta||}, \\ \text{Avec } \theta^T x + b > 1, & \text{si } y{=}1 \\ \text{et } \theta^T x + b < -1, & \text{si } y{=}\text{-}1 \end{cases}$$

Or maximiser $\frac{2}{||\theta||}$ revient simplement à minimiser $\frac{1}{2}\theta$ et nos 2 conditions peuvent se combiner en une seule qui est $y_i(\theta^T x_i + b) \geq 1 \ \forall i$
Ainsi le problème à résoudre devient le suivant :

$$\begin{cases} min \frac{1}{2}||\theta||, \\ y_i(\theta^T x_i + b) \geq 1, \quad \forall i \end{cases}$$

Ceci n'est rien d'autre qu'un problème d'optimisation sous contrainte, pour le résoudre on utilise la méthode de Lagrange .
Elle consiste à cherche le lagrangien de notre problème et égaliser son gradient à zéro.
le lagrangien est donné par :

$$L = \frac{1}{2}\theta^T\theta + \sum_{i=1}^{m} \alpha_i(1 - y_i(\theta^T x_i + b)) \text{ , car } ||\theta||^2 = \theta^T\theta$$

et $\nabla L = 0$
Ce qui nous donnent :
$\frac{\partial L}{\partial \theta} = \theta - \sum_{i=1}^{m} \alpha_i y_i x_i = 0$ ce qui donne $\theta = \sum_{i=1}^{m} \alpha_i y_i x_i$
$\frac{\partial L}{\partial b} = 0$ nous donne $\sum_{i=1}^{m} \alpha_i y_i = 0$
En mettant ces 2 expressions des L on obtient :
$L = \frac{-1}{2} \sum_{i=1}^{m} \sum_{i=j}^{m} \alpha_i \alpha_j y_i y_j X_i^T . X_j + \sum_{i=1}^{m} \alpha i$
,qui n'est qu'une fonction de α_i
Il est connu sous le nom d'un problème de dualité car si on connait α_i on connait θ L doit être maximiser maintenant , ainsi notre problème de dualité s'écrira :

$$\begin{cases} max \sum_{i=1}^{m} \alpha i - \frac{1}{2} \sum_{i=1}^{m} \sum_{i=j}^{m} \alpha_i \alpha_j y_i y_j X_i^T . X_j, \\ \text{Avec } \sum_{i=1}^{m} \alpha_i y_i = 0, \\ \text{et } \alpha_i \geq 0 \end{cases}$$

Avant de s'attaquer à la résolution de ce problème remarquons quelque éléments intéressants qu'il présente :
- la grande partie des valeurs de α_i sont nulles et aux valeurs non nulles correspondent des x_i qu'on appelle support Vector.
- Pour prédire la classe d'un nouveaux élément z nous avons qu'a verifer s'il se place au dessus ou en dessous de notre frontière de décision qui est donnée par $\theta^T z + b = \sum_{i=1}^{m} \alpha_i y_i (X_i^T . Z)$.
- Mais aussi il présente l'avantage que l'algorithme comprend le produit scalaire des tuples d'entrés , cet avantage sera utilisé lorsqu'on va traiter des frontières de décisions non linéairement séparable.

Données non Séparables linéairement : Notion de Kernel [14]

Dans la plupart des cas surtout dans la pratique les données ne peuvent pas toujours être classer avec une droite ou un hyperplan comme on peut le voir à la FIGURE15 :
Une propriété stipule que les données qui ne sont pas linéairement séparable dans un espace de dimension n le seront dans un espace de dimension m avec $m \geq n$ [**8**]. il suffit juste de faire un mapping des tuples x_i dans R^n vers R^m en utilisant une fonction $\phi(x)$. Voyons un exemple à la FIGURE16

Définition : Kernel Un Kernel est une fonction qui permet de calculer le produit scalaire du mapping d'un tuple dans un espace de dimension supérieur que celui dans lequel il est définie . on a :

$$K(x, z) = \phi^T(x) . \phi(z)$$

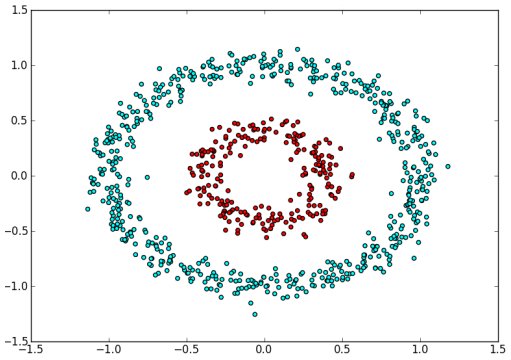


FIGURE 1.15 – Données non séparable linéairement

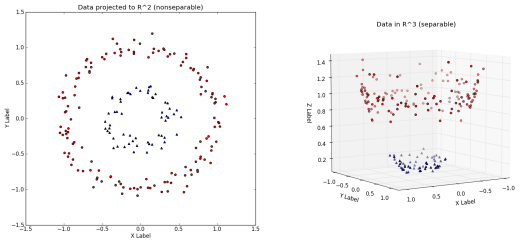


FIGURE 1.16 – Données non séparable linéairement dans R^2 mais qui le devient dans R^3 avec $\phi(x_1, x_2) = (x_1, x_2, x_1^2 + x_2^2)$

Mais on n'est pas obligé de connaitre $\phi(x)$ pour calculer $K(x, z)$ il existe des Kernel bien définie qui permettent de calculer le produit scalaire sans pour autant connaitre ϕ et ainsi faciliter les calcul .
En voici quelque uns les plus utilisé :
- Le Kernel Polynomiale de degré d : $K(x, z) = (x^T z + 1)^d$
- Le Kernel Gaussien : $K(x, z) = \exp(\frac{-||x-z||^2}{2\sigma^2})$
- Le Kernel sigmoïde :$K(x, z) = \tanh(kx^T y + \theta)$

Ainsi dans notre problème d'optimisation on peut juste remplacer le produit scalaire de x_i et x_j et ensuite palier aux problème des frontières de décisions non séparable linéairement.
On choisie un Kernel en fonction des caractéristiques de l'ensemble d'apprentissage .
Le problème sera :

$$\begin{cases} max \sum_{i=1}^{m} \alpha i - \frac{1}{2} \sum_{i=1}^{m} \sum_{i=j}^{m} \alpha_i \alpha_j y_i y_j \phi^T(x).\phi(y), \\ \text{Avec } \sum_{i=1}^{m} \alpha_i y_i = 0, \\ \text{et } \alpha_i \geq 0 \end{cases}$$

Et en introduisant le Kernel on a :

$$\begin{cases} max \sum_{i=1}^{m} \alpha i - \frac{1}{2} \sum_{i=1}^{m} \sum_{i=j}^{m} \alpha_i \alpha_j y_i y_j K(x, y), \\ \text{Avec } \sum_{i=1}^{m} \alpha_i y_i = 0, \\ \text{et } \alpha_i \geq 0 \end{cases}$$

Et ainsi notre frontière de décision sera : $\theta^T z + b = \sum_{i=1}^{m} \alpha_i y_i K(x, y) + b$.
Pour résoudre ce problème on utilise la méthode de SMO ou Sequential Minimal Optimization qui est la technique la plus utilisée pour la résolution de ces genres des problèmes , il est implémenté dans la plupart des package d'apprentissage automatique .

1.2.5 Arbres des Decisons [15] [16]

Définitions

Une arbre de décision est un ensemble des règles de classification et de régression basant leurs décisions sur des tests associé aux attributs ,organisées de manière arborescente .
C'est une représentation d'une procédure d'apprentissage.
Voici quelques termes ou notions liées aux arbres de décisions :
Noeud Principale ou Root Node : c'est l'attribut qui se situe au premier niveau et sur base de celui- ci s'effectue notre prédiction.
Spliting ou segmentation : c'est un processus consistant à diviser un nœud en 2 ou plusieurs sous nœuds sur base d'un test sur un attribue.
Nœud Interne ou Nœud de décision : c'est un nœud étiqueté par un test qui peut être appliqué à toute description d'un individu de la population.
Une feuille : c'est le nœud où on ne peut plus diviser ou segmenter l'arbre , il est étiqueté par une classe.

Construction de l'arbre et Algorithme

Mesure de la pureté des feuilles Pour Construire les nœuds de l'arbre,les choix des « questions les plus discriminantes » peuvent se faire selon plusieurs critères : l'algorithme CART utilise l'indice de Gini, l'algorithme C4.5 utilise l'entropie, l'algorithme Chi-square Automatic Interaction Detector (CHAID) utilise le test de khi carré, et plusieurs autres algorithmes.Dans cette partie nous ne parlerons que de deux premiers. Ces deux outils mathématiques visent à évaluer la « pureté » de chaque feuille : lorsque l'on se situe à un nœud donné de l'arbre, le but est de créer deux feuilles qui soient plus homogènes que le nœud qui les précède. Il faut donc disposer d'un moyen de mesurer cette homogénéité, ou « pureté ». Grâce à cela, à chaque nœud, le split est construit de manière à maximiser le gain d'information apporté par une question donnée sur la connaissance de la variable réponse.

Entropie

L'entropie est une fonction mathématique créée par Claude Shannon en 1948, pour des questions initialement liées à la théorie du signal.
En Machine Learning c'est la mesure du désordre ou de l'inégalité de répartition pour le choix d'un test à une position de l'arbre.
Notons par E notre ensemble d'apprentissage divisé en classes $\omega_1, \omega_2, ..., \omega_k$
- L'entropie de la distribution des classes = quantité moyenne d'information

nécessaire pour identifier la classe d'un exemple de E.

$$H(E) = -\sum_{j=1}^{k} P(\omega_k) \log_2(\omega_k)$$

où $P(\omega_k)$ est la probabilité a priori de la classe ω_k
- Soit un test T(portant sur une variable X) ayant m alternatives possibles qui divisent En sous-ensembles E_j , caractérisé par une entropie $H(E_j)$.
- L'entropie de la partition résultante, c'est-à-dire l'entropie conditionnelle de E étant donné T, est définie comme l'entropie moyenne des sous-ensembles :

$$H(E|T) = \sum_{j=1}^{j} P(E_j)H(E_j)$$

- Le gain d'information apporté par le test Test donc :

$$Gain(E, T) = H(E) - H(E|T)$$

L'algorithme de l'arbre de decisison se base sur ces propriétés.

L'indice de Gini

L'indice (ou coefficient) de Gini est une mesure, comprise entre 0 et 1, de la dispersion d'une distribution. Il est très souvent utilisé en économie ou en sociologie afin de mesurer les inégalités sociales au sein d'un pays. Dans ce contexte, plus le coefficient est proche de 1 et plus la société est inégalitaire. il se calcule de la manière suivante :

$$Gini = \sum_{i \neq j} p(\omega_i)p(\omega_j)$$

Avec $p(\omega_i)$ la probabilité de la classe ω_i .
De la même façon que l'entropie on calcul le gain d'information avec l'indice de gini.

Algorithme

En général, on décide qu'un nœud est terminal lorsque tous les exemples associés à ce nœud, ou du moins la plupart d'entre eux sont dans la même classe, ou encore, s'il n'y a plus d'attributs non utilisés dans la branche correspondante.

En général, on attribue au nœud la classe majoritaire (éventuellement calculée à l'aide d'une fonction de coût lorsque les erreurs de prédiction ne sont pas équivalentes). Lorsque plusieurs classes sont en concurrence, on peut choisir la classe la plus représentée dans l'ensemble de l'échantillon, ou en choisir une au hasard.

Algorithm 3 Pseudo code de l'arbre de décision

Require: Training set E

Initialiser l'arbre courant 'a l'arbre vide ; la racine est le nœud courant
while On n'as pas encore obtenu l'arbre **do**
Décider si le nœud courant est terminale
if le nœud est terminal **then**
lui affecter une classe
else
Sélectionner un test et créer autant de nouveaux nœuds fils qu'il y a de réponses possibles au test
Passer au nœud suivant non exploré s'il en existe

1.2.6 Élagage de l'arbre ou *Prunning*

L'objectif de cette étape est de supprimer les parties de l'arbre qui ne semblent pas performantes pour prédire la classe de nouveaux cas et les remplacées par un nœud terminal (associé à la classe majoritaire).
Le processus est remplacées par un nœud terminal (associé à la classe majoritaire).
il existe différentes façons d'estimer le taux d'erreur entre autre :
– sur base de nouveaux exemples disponibles ;
– via une validation croisée ;
– sur base d'une estimation statistique, ex : borne supérieure d'un intervalle de confiance construit sur un modèle binomial .

1.2.7 Avantages et inconvénients des arbres de décisions

Avantages

- Interprétabilité : chaque élément du modèle est facile à comprendre et à analyser pour un humain, et peut donner de l'information sur les données. Ceci est surtout vrai pour les petits arbres. À cause de l'instabilité des arbres (par exemple si on varie le choix des variables ou des données), il faut utiliser des techniques spécialisées pour determiner quelles variables sont réellement importantes. On peut faire correspondre un arbre de décision à un ensemble de règles SI-ALORS (en introduisant des nouvelles classes correspondant aux noeuds cachés de l'arbre).
- Flexibilité : peut être utilisé sur des données de n'importe quel type (dont évidemment les variables continues et discrètes) pour lesquelles un ensemble fini (pas trop grand) de questions possibles pour la partition peut être défini

(en principe cela peut être appliqué à n'importe quel type de structures de données).

Inconvénients

Un des inconvénients principaux des méthodes d'apprentissage par arbres de décision est leur instabilité. Sur des données réelles, il s'en faut souvent de peu qu'un attribut soit choisi plutôt qu'un autre et le choix d'un attribut-test, surtout s'il est près de la racine, influence grandement le reste de la construction. La conséquence de cette instabilité est que les algorithmes d'apprentissage par arbres de décision ont une variance importante, qui nuit à la qualité de l'apprentissage. Des méthodes comme le Bagging(pour Bootstrap Aggregating) ou les Random Forests(qui consiste à utiliser plusieurs arbres et utiliser le vote classe faite par chaque arbre pour classer une instance) [17] permettent dans une certaine mesure de remédier à ce problème.

Random Forrest

1.2.8 Segmentation Par K-means

La Segmentation est une technique d'apprentissage non supervisé, Les classes des données ne sont pas connus en avance . La segmentation consiste à grouper les données dans des clusters ou segments selon leurs ressemblance , et leurs similarités. L'objectif de la segmentation est la maximisation des distances inter-cluster et la minimisation de la distance intra-cluster . L'unité de mesure de la ressemblance entre les données c'est la distance .
Il existes plusieurs métriques pour définir la distance entre 2 points x_i et y_i :
- La distance euclidienne $d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
- La distance de Manhattan $d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$
- La distance de Minkowski $d(x, y) = \sqrt[q]{\sum_{i=1}^{n} |x_i - y_i|^p}$

Supposons que nous disposons de m points x_i que nous voulons segmenter en k clusters , l'objectif est d'assigner un cluster à chaque point .
La methode K-means consiste à trouver les positions μ_k k = 1,2...K qui minimisent la distance distance du point aux cluster . L'algorithme K-means consiste à résoudre le roblème suivant :

$$\min \sum_{i=1}^{k} \sum_{\underline{X} \in c_i} d(\underline{x}, \mu_k)$$

ou c_i est l'ensemble des points appartenant au cluster i

Algorithme K-means

1.3 Méthodologie de mise en place du processus de data mining[3]

. Au début des années 90, l'intérêt croissant pour le data mining a mis en lumière l'absence d'une méthodologie pour la mise en place d'un processus de découverte de connaissances, applicable quelle que soit l'industrie visée ou l'outil utilisé. De ce besoin est née l'initiative CRISP-DM. A partir du

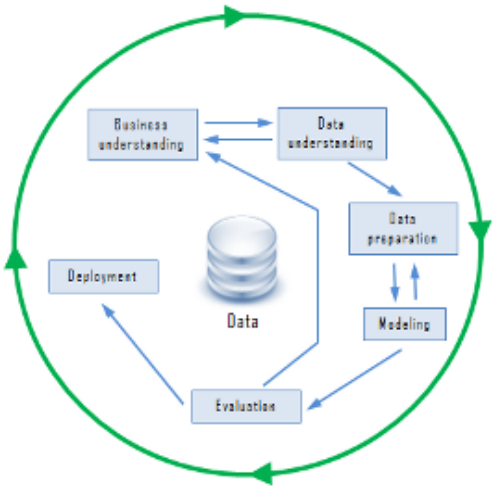


FIGURE 1.17 – Data mining process model

processus de découverte de connaissances utilisé dans les premiers projets de data mining ,CRISP-DM a défini et validé une méthodologie potentiellement applicable dans tous les secteurs de l'industrie. Elle permet de rendre les projets de data mining à grande échelle plus rapides, moins coûteux, plus fiables et surtout améliorer leur gestion. Cette méthodologie ne vise pas que les grands projets car même les petits projets de découverte de connaissances peuvent tirer profit de son utilisation. Essentiellement, cette méthodologie fournit un aperçu du cycle de vie d'un projet de data mining. Elle identifie clairement les principales phases de ce processus au travers de tâches et des relations entre ces tâches. Même si le modèle ne le spécifie pas explicitement, il y a des relations possibles entre toutes les tâches en fonction des objectifs d'analyse et des données qui sont analysées. Les six phases importantes du processus sont :

1. *La compréhension du problème métier* : concerne la définition du problème d'analyse sur la base des objectifs métiers qui en sont à l'origine.
2. *La compréhension des données* :cette phase vise à déterminer précisément les données à analyser et à identifier la qualité des données

.

3. *La préparation des données* : couvre les activités liées à la construction de l'ensemble précis des données à analyser à partir des données brutes. Ceci inclue le nettoyage des données, la sélection d'attributs, le choix des observations, etc.
4. *La modélisation* : est la phase consistant dans le paramétrage et le test de différentes techniques de data mining sur les données choisies, dans l'objectif d'optimisation du modèle ou des connaissances obtenues par ces techniques.
5. *L'évaluation* :vise à vérifier le modèle ou les connaissances obtenues afin de s'assurer qu'ils répondent aux objectifs formulés au début du processus. Elle contribue aussi à la décision de déploiement du modèle ou, au contraire, de la nécessité que ce dernier soit reconstruit ou amélioré.
6. *Le déploiement* : est l'étape finale du processus de découverte de connaissances. Son objectif est de mettre la connaissance obtenue par la modélisation dans une forme adaptée et l'intégrer au processus de prise de décision. Le déploiement peut aller, selon les objectifs, de la simple génération d'un rapport décrivant les connaissances obtenue jusqu'à la mise en place d'une application spécifique permettant l'utilisation du modèle obtenu pour la prédiction des valeurs inconnues d'un paramètre d'intérêt.

Bibliographie

[1] E. B. OMAR S. IMANE. "Prediction approach for improving students orientation in university : case of sidi mohammed ben abdelah university". In : *International Journal of Computer Science and Applications* (2015), p. 108.

[2] I. Perrier R. R. TREMBLAY. *Savoir plus : outils et méthodes de travail intellectuel*. Les Éditions de la Chenelière inc, 2006.

[3] RITHME CONSULTING. *Méthodologie de mise en place du processus de data mining*. 2017. URL : `http://www.rithme.eu/?m=resources&p=dmmethod&lang=fr` (visité le 29/04/2017).

[4] K. MULENDA. "cours d'intelligence artificielle et systèmes experts". inédit FSTA ULPGL. juin 2016.

[5] Mark A. Hall IAN H. WITTEN Eibe Frank. *Data Mining Practical Machine Learning Tools and Techniques*. USA : Morgan Kaufmann Publishers, 2011, p. 1.

[6] DHAKSHAYANI N. *What is the difference between data mining, artificial intelligence and machine learning ?* Sous la dir. de QUORA. Answer To quora Question. URL : `\url{https://www.quora.com/What-is-the-difference-between-data-mining-artificial-intelligence-and-machine-learning}`.

[7] M. BATER. *Learning Data mining with R*. Mumbai : Pack publish, 2015, p. 10.

[8] Djazia CHAMI. "Une plate forme orientée agent pour le data mining". Thèse de doct. Université de Batna 2, 2010.

[9] SHARP SIGHT LABS. *What is the difference between machine learning,statistics,and datamining ?* Sous la dir. de SHARP SIGHT LABS. `http://www.sharpsightlabs.com/differenceÂŋmachineÂŋlearningÂŋstatisticsÂŋda` `?utm_content=bufferf9474\&utm_medium=social`. Accessed : 2016-05-16.

[10] Gongsun L. *What is the difference between data mining, statistics, machine learning and AI ?* Sous la dir. de StackExchange. Answer To StackExchange Question. URL : \url{https://stats.stackexchange.com/questions/5026/what-is-the-difference-between-data-mining-statistics-machine-learning-and-ai}.

[11] Andrew Ng. *Stanford CS229 -Machine Learning -Ng*. https://www.coursera.org/learn/machine-learning. Accessed : 2016-11-23 au 2016-12-05. 2011.

[12] Bishop C. *Neural networks for pattern recognition*. Oxford : Clarendon Press, 1995.

[13] Alexandre KOWALCZYK. *SVM Tutorial*. http://www.svm-tutorial.com/. Accessed : 2016-12-6. 2014.

[14] Eric Kim. *Everything You Wanted to Know about the Kernel Trick*. http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html. Accessed : 2016-12-09. 2011.

[15] M.Saerens C. Decaestecker. *Les arbres de décision (decision trees)*. Belgique : inédit ULB, 2006, p. 1.

[16] ANALYTICS VIDHYA CONTENT TEAM. *A Complete Tutorial on Tree Based Modeling from Scratch (in R & Python) MACHINE LEARNING PYTHON R*. 2016. URL : https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/ (visité le 27/03/2017).

[17] Leo Breiman. "Random forests". In : t. 45. 1. Springer, 2001, p. 5–32.