Découverte de connaissances à partir de données[1]

Rémi Gilleron Marc Tommasi

7 novembre 2000

1. Ces notes de cours correspondent à une partie d'un cours donné en IUP MIAGE troisième année à Lille 1 et en maîtrise MASS à Lille 3

Table des matières

Une version de ce document est disponible sur

`http://www.grappa.univ-lille3.fr/polys`

avec bien d'autres polycopiés d'ailleurs. Des versions au format LaTeX et PostScript sont bien entendu disponibles à l'adresse

`http://www.grappa.univ-lille3.fr/polys/fouille/main.tgz`

sous forme d'archive compressée avec les programmes gzip et tar.

Marc Tommasi et Rémi Gilleron

Introduction

Ce cours présente des outils, des techniques liées à l'informatique décisionnelle. De l'entrepôt de données qui définit un support au système d'information décisionnel, aux outils de fouille de données permettant d'extraire de nouvelles connaissances, de nombreux moyens informatiques sont aujourd'hui mis en œuvre pour aider les organes de décision des entreprises.

L'informatique de gestion a gagné sa place dans l'entreprise depuis les années 60 par une succession de progrès technologiques, logiciels et méthodologiques qui ont tous contribué à une réduction des coûts d'exploitation. L'invention du compilateur et de la compatibilité des séries de machines dans les années 60 a permis aux grands comptes de s'équiper. Le microprocesseur et les bases de données dans les années 70 ont rendu l'informatisation accessible aux moyennes et grandes entreprises. Les bases de données relationnelles, les progiciels de gestion, ainsi que les premiers micro-ordinateurs des années 80 ont largement contribué à l'équipement des petites et moyennes entreprises, commerces, administrations. Jusque là, la plus grande partie des applications était dédiée au traitement des données directement liées à l'activité quotidienne des organisations : paie, comptabilité, commandes, facturation, … Ces applications que l'on regroupe sous le terme d'*informatique de production ou Informatique Opérationnelle*. L'architecture générale était l'architecture maître-esclave, avec le maître, un puissant ordinateur (mini, ou gros système) en site central et les esclaves, terminaux passifs en mode texte. L'organisation de l'entreprise était très hiérarchisée dans sa structure informatique et sa structure de pilotage. Si des techniques d'aide à la décision ont été mises en place (essentiellement basées sur des outils de simulation et d'optimisation, parfois aussi de systèmes experts), elles nécessitaient l'intervention d'équipes d'informaticiens pour le développement de produits spécifiques. Ces outils étaient mal intégrés dans le système d'information.

Avec l'apparition des ordinateurs personnels et des réseaux locaux, une autre activité a émergé, tout à fait distincte de l'informatique de production. Dans les secrétariats, les cabinets, on utilise des tableurs et des logiciels de traitements de texte, des petites bases de données sur des machines aux interfaces graphiques plus agréables. Jusqu'aux années 90, ces deux mondes (« bureautique vs informatique ») se sont ignorés, mais avec la montée en puissance des micro ordinateurs et l'avènement de l'architecture client-serveur, on observe aujourd'hui un décloisonnement remarquable. Le mot d'ordre principal est :

> *fournir à tout utilisateur reconnu et autorisé, les informations nécessaires à son travail.*

Ce slogan fait naître une nouvelle informatique, intégrante, orientée vers les utilisateurs et les centres de décision des organisations. C'est l'ère du client-serveur qui prend vraiment tout son essor à la fin des années 90 avec le développement des technologies Intranet.

Enfin, un environnement de concurrence plus pressant contribue à révéler *l'informatique décisionnelle*. Tout utilisateur de l'entreprise ayant à prendre des décisions doit pouvoir accéder en temps réel aux données de l'entreprise, doit pouvoir traiter ces données, extraire l'information pertinente de ces données pour prendre les « bonnes » décisions. Il se pose des questions du type : « quels sont les résultats des ventes par gamme de produit et par région pour l'année dernière ? » ; « Quelle est l'évolution des chiffres d'affaires par type de magasin et par période » ; ou encore « Comment qualifier les acheteurs de mon produit X ? » ... Le système opérationnel ne peut satisfaire ces besoins pour au moins deux raisons : les bases de données opérationnelles sont trop complexes pour pouvoir être appréhendées facilement par tout utilisateur et le système opérationnel ne peut être interrompu pour répondre à des questions nécessitant des calculs importants. Il s'avère donc nécessaire de développer des systèmes d'information orientés vers la décision. Il faut garder un historique et restructurer les données de production, éventuellement récupérer des informations démographiques, géographiques et sociologiques. Les *entrepôts de données ou datawarehouse* sont la réalisation de ces nouveaux systèmes d'information. De nouveau, cette apparition est rendue possible grâce aux progrès technologiques à coûts constants (grâce à l'augmentation importante des capacités de stockage et à l'introduction des techniques du parallélisme dans l'informatique de gestion, techniques qui permettent des accès rapides à de grandes bases de données).

L'informatique décisionnelle s'est développée dans les années 70. Elle est alors essentiellement constituée d'outils d'édition de rapports, de statistiques, de simulation et d'optimisation. Provenant des recherches en Intelligence Artificielle, les systèmes experts apparaissent. Ils sont conçus par « extraction » de la connaissance d'un ou plusieurs experts et sont des systèmes à base de règles. De bons résultats sont obtenus pour certains domaines d'application tels que la médecine, la géologie, la finance, ... Cependant, il apparaît vite que la formalisation sous forme de règles de la prise de décision est une tâche difficile voire impossible dans de nombreux domaines. Dans les années 90, deux phénomènes se produisent simultanément. Premièrement, comme nous l'avons montré dans les paragraphes précédents, il est possible de concevoir des environnements spécialisés pour l'aide à la décision. Deuxièmement, de nombreux algorithmes permettant d'extraire des informations à partir de données brutes sont arrivés à maturité. Ces algorithmes ont des origines diverses et souvent multiples. Certains sont issus des statistiques ; d'autres proviennent des recherches en Intelligence Artificielle, recherches qui se sont concentrées sur des projets moins ambitieux, plus ciblés ; certains s'inspirent de phénomèmes biologiques ou de la théorie de l'évolution. Tous ces algorithmes sont regroupés dans des logiciels de *fouille de données ou Data Mining* qui permettent la recherche d'informations nouvelles ou cachées à partir de données. Ainsi, dans le cas de systèmes à base de règles, plutôt que d'essayer d'extraire la connaissance d'experts et d'exprimer cette connaissance sous forme de règles, un logiciel va générer ces règles à partir de données. Par exemple, à partir d'un fichier historique des prêts contenant des renseignements sur les clients et le résultat du prêt (problèmes de recouvrement ou pas), le logiciel extrait un profil pour désigner un « bon » ou un « mauvais » client. Après validation, un tel système peut être implanté dans le système d'information de l'entreprise afin de « classer » ou de « noter » les nouveaux clients.

Plusieurs méthodes existent pour mettre en oeuvre la fouille de données. Le choix de l'une d'entre elles est une première difficulté pour l'utilisateur ou le concepteur. Aucune méthode n'est meilleure qu'une autre dans l'absolu. Néanmoins, l'environnement, les contraintes, les objectifs et bien sûr les propriétés des méthodes doivent guider l'utilisateur dans son choix.

Les entrepôts de données et la fouille de données sont les éléments d'un domaine de recherche et de développement très actifs actuellement : *l'extraction de connaissances à partir de données ou Knowedge Discovery in Databases (KDD for short).* Par ce terme, on désigne tout le cycle de découverte d'information. Il regroupe donc la conception et les accès à de grandes bases de données ; tous les traitements à effectuer pour extraire de l'information de ces données ; l'un de ces traitements est l'étape de *fouille de données ou Data Mining.* C'est l'objectif de ce cours que de vous en présenter les éléments essentiels.

Le plan de ce cours est le suivant :

- un premier chapitre présente les entrepôts de données (chapitre 1) en insistant sur les différences entre un tel système et les bases de données opérationnelles et transactionnelles ; en présentant des éléments méthodologiques pour la conception d'entrepôts de données, les modèles de données correspondants, les problèmes liés à l'alimentation de ces entrepôts, et quelques éléments d'information sur les technologies qui optimisent les accès à de tels systèmes.
- Un second chapitre (chapitre 2) présente le cycle complet de découverte d'informations à partir de données (Knowledge Discovery in Databases) : la préparation des données, le nettoyage, l'enrichissement, le codage et la normalisation, la fouille de données, la validation et l'intégration dans le système d'information.
- Le dernier chapitre porte une attention particulière sur la fouille de données (chapitre 3). Il est hors d'atteinte d'un tel cours de prétendre présenter toutes les techniques disponibles. Nous présentons la base des méthodes les plus classiques : l'algorithme des k-moyennes, les règles d'association, la méthode des plus proches voisins, les arbres de décision et les réseaux de neurones.

Les sources principales des auteurs se trouvent bien sûr en bibliographie page 63. Mais nous recommandons essentiellement la lecture des ouvrages suivants :

- le livre de P. Adriaans et D. Zantinge [AZ96], remarquable par sa clarté, qui présente la découverte de connaissances à partir de données ;
- le livre de R. Kimball [Kim97] sur les entrepôts de données ; le livre contient de nombreux exemples ;
- le livre de M. Berry et G. Linoff [BL97] qui présente, en contexte d'applications et clairement, les méthodes de fouille de données ;
- pour ceux qui seraient plus intéressés par les aspects algorithmiques et l'apprentissage automatique, l'ouvrage de T. Mitchell [Mit97] est, en tous points, exceptionnel. Pour

les aspects classification supervisée, il est possible également de consulter le poly de F. Denis et R. Gilleron [DG99] qui contient de nombreux exercices.

Chapitre 1

Entrepôts de données

Un métier du secteur informatique a certainement déjà disparu : monteur de bandes. Cette personne était chargée de manutentionner des bandes magnétiques depuis les lieux de stockage jusqu'au site informatique central ou trônait un gigantesque ordinateur avec ses lecteurs de bande comme seule unité de sauvegarde. Pour charger une nouvelle application, il fallait monter la bande correspondante dans cette armoire où l'on voyait deux disques tourner, s'arrêter, changer de sens etc. Ce précieux périphérique remplaçait avantageusement les lecteurs de cartes trop lents. Mais l'apparition du disque dur, d'une remarquable capacité de 5Mo, a envoyé la bande à la décharge au milieu des années 80. Aujourd'hui, le Mo est l'unité pour la mémoire vive et nous produisons des machines stockant plusieurs PetaOctets (des millions de milliards d'octets). Ce besoin de stockage est-il justifié et en quoi est-il nécessaire ?

Les sociétés de téléphone gardent au moins un an les positions géographiques et les consommations de leurs abonnés « mobiles ». Les grands magasins et les entreprises de vente par correspondance (VPC) conservent les achats de leurs clients (tickets de caisse en grande distribution, commandes en VPC), collectent des informations sur leurs clients grâce à des systèmes de cartes de fidélité ou de crédit, et achètent des bases de données géographiques et démographiques. Les sites web conservent des traces de connexions sur leurs sites marchands. En résumé, les entreprises en secteur très concurrentiel conservent les données de leur activité et achètent même des données.

Les motifs qui ont présidé à la conservation de ces données étaient : des obligations légales pour pouvoir justifier les facturations, des raisons de sécurité pour pouvoir détecter les fraudes, des motifs commerciaux pour suivre l'évolution des clients et des marchés. Quelle que soit la raison initiale, les entreprises se sont rendues compte que ces données pouvaient être une source d'informations à leur service. Ce constat, valable pour les sociétés du secteur marchand, peut être étendu à de nombreux domaines comme la médecine, la pharmacologie. Il faut donc définir des environnements permettant de mémoriser de grands jeux de données et d'en extraire de l'information.

Les structures qui accueillent ce flot important de données sont des entrepôts de données ou *data warehouse*. Ils sont construits sur une nouvelle architecture permettant d'extraire l'information, architecture bien différente de celle prévue pour l'informatique de production, basée elle sur des systèmes de gestion de bases de données relationnelles et des serveurs transactionnels. Un entrepôt de données est construit en l'alimentant *via* les serveurs transactionnels de façon bien choisie et réfléchie pour permettre aux procédures d'extraction de connaissances de bien fonctionner. L'organisation logique des données est particulièrement conçue pour autoriser des recherches complexes. Le matériel est évidemment adapté à cette utilisation.

Nous rappelons les différences essentielles entre informatique de production et de décision (section 1.1). La phase de construction est abordée en section 1.2. Nous la présentons en trois parties : l'étude préalable qui étudie la faisabilité, les besoins ; l'étude des données et de leur modélisation ; l'étude de l'alimentation l'entrepôt. Le chapitre se termine par une présentation rapide des outils d'exploitation et d'administration : navigation, optimisation, requêtes et visualisation (section 1.3).

1.1 Informatique décisionnelle vs Informatique de production

Une des principales caractéristiques des systèmes de production est une activité constante constituée de modifications et d'interrogations fréquentes des données par de nombreux utilisateurs : ajouter une commande, modifier une adresse de livraison, rechercher les coordonnées d'un client, ... Il faut conserver la cohérence des données (il faut interdire la modification simultanée d'une même donnée par deux utilisateurs différents). Il s'agit donc de privilégier un enregistrement rapide et sûr des données. À l'inverse, les utilisateurs des systèmes d'information de décision n'ont aucun besoin de modification ou d'enregistrement de nouvelles données. Ils vont interroger le système d'information et les questions posées seront de la forme : « quelles sont les ventes du produit X pendant le trimestre A de l'année B dans la région C ». Une telle interrogation peut nécessiter des temps de calcul importants. Or, l'activité d'un serveur transactionnel ne peut être interrompue. Il faut donc prévoir une nouvelle organisation qui permette de mémoriser de grands jeux de données et qui facilite la recherche d'informations.

Enfin, nous étudierons dans le chapitre suivant l'extraction de connaissances à partir de données. Les méthodes de fouille de données nécessitent une préparation et une organisation particulière des données que nous détaillerons au chapitre suivant. L'existence d'un entrepôt simplifiera cette tâche et permettra donc d'optimiser le temps de développement d'un projet d'extraction de connaissances. En résumé, on peut justifier la construction d'un entrepôt de données par l'affirmation suivante :

> il est beaucoup plus simple de trouver une information pertinente dans une structure organisée pour la recherche de connaissance.

La production

Le modèle Entité-Association (ou modèle EA) est l'un des formalismes les plus utilisés pour la représentation conceptuelle des systèmes d'information. Il permet de construire des modèles de données dans lesquels on cherche à tout prix à éviter des redondances. Toute donnée mémorisée plus d'une fois est source d'erreurs, d'incohérences, et va pénaliser les temps d'exécution et complexifier les procédures d'ajout, de suppression ou de modification.

D'un point de vue logique, ce sont les bases de données relationnelles qui ont su au mieux représenter ces modèles conceptuels. Les éditeurs de logiciels et les architectes d'ordinateurs ont dû fournir un effort important pour les rendre efficaces. Aujourd'hui les bases relationnelles ont acquis une maturité satisfaisante pour être largement utilisées et diffusées dans de nombreuses organisations, à toute échelle.

Conserver la cohérence de la base de données, c'est l'objectif et la difficulté principale pour l'informatique de production. Les systèmes transactionnels (temps réel) OLTP (*On-Line Transaction Processing*) garantissent l'intégrité des données. Les utilisateurs accèdent à des éléments de la base par de très courtes transactions indécomposables, isolées. Ils y accèdent très souvent pour des opérations d'ajout, suppression, modification mais aussi de lecture. L'isolation permet de garantir que la transaction ne sera pas perturbée ni interrompue. Les contrôles effectués sont élémentaires. La brièveté garantit que les temps de réponse seront acceptables (inférieurs à la seconde) dans un environnement avec de nombreux utilisateurs.

Bien sûr, dans le souci de la garantie des performances, une requête dont le calcul prendrait trop de temps est inacceptable. Par exemple, une jointure sur de grosses tables à l'aide de champs non indexés est interdite. Les travaux sur une telle base sont souvent simples et répétitifs. Dès lors qu'un travail plus important est nécessaire, l'intervention de programmeurs et d'administrateurs de la base est requise et une procédure *ad-hoc* est créée et optimisée.

Le modèle Entité-Association et sa réalisation dans un schéma relationnel sont pourtant des obstacles importants pour l'accès de l'utilisateur final aux données. Dans une situation réelle, le modèle des données est très large et contient plusieurs dizaines d'entités. Les bases sont alors constituées de nombreuses tables, reliées entre elles par divers liens dont le sens n'est pas toujours explicite. Souvent, les organisations ont choisi une norme pour définir des noms à chaque objet (table, champ,...) très « syntaxiques » sans sémantique claire. Le but était de faciliter le travail des développeurs et l'efficacité des procédures. L'utilisateur final n'est pas considéré ici car c'est à l'informaticien de lui proposer une abstraction de ces modèles à travers les outils dont il a besoin. La complexité des données, l'absence d'annuaire clair rend la base inutilisable aux non initiés sans l'intervention d'informaticiens et d'outils sur mesure.

L'informatique de production a donc été conçue pour privilégier les performances de tâches répétitives, prévues et planifiées tournées vers la production de documents standards (factures, commandes...). L'intervention de l'utilisateur est guidée à travers des outils spécifiques proposés par une équipe de développeurs.

La dernière caractéristique de ces bases de données est qu'elles conservent l'état instantané du système. Dans la plupart des cas, l'évolution n'est pas conservée. On conserve simplement des versions instantanées pour la reprise en cas de panne et pour des raisons légales.

Le décisionnel

Dans un système d'information décisionnel, l'utilisateur final formulera des questions du type :

- Comment se comporte le produit X par rapport au produit Y ?
- Et par rapport à l'année dernière ?
- Quel type de client peut bien acheter mon produit Z ?

Ces exemples permettent de mettre en évidence les faits suivants :

- les questions doivent pouvoir être formulées dans le langage de l'utilisateur en fonction de son « métier », c'est-à-dire de son secteur d'activité (service marketing, service économique, service gestion des ressources humaines, ...) ;
- la prévision des interrogations est difficile car elles sont du ressort de l'utilisateur. De plus, ses questions vont varier selon les réponses obtenues : : si le produit X s'est vendu moins bien que l'année précédente, il va être utile d'en comprendre les raisons et donc de détailler les ventes du produit X (par région, par type de magasin, ...)
- des questions ouvertes (profil client du produit Z) vont nécessiter la mise en place de méthodes d'extraction d'informations.

Ce qui caractérise d'abord les besoins, c'est donc la possibilité de poser une grande variété de questions au système, certaines prévisibles et planifiées comme des tableaux de bord et d'autres imprévisibles. Si des outils d'édition automatiques pré-programmés peuvent être envisagés, il est nécessaire de permettre à l'utilisateur d'effectuer les requêtes qu'il souhaite, par lui-même, sans intervention de programmeurs. Deux contraintes apparaissent alors immédiatement: la simplicité du modèle des données, la performance malgré les grands volumes.

Pour les entrepôts de données, on recherche plus de lisibilité, de simplicité que dans le cas des SGBD. La modélisation introduit les notions de *fait* et *dimension*. Les faits correspondent à l'activité de l'entreprise : ce sont les ventes pour une entreprise commerciale, les communications pour une entreprise de télécommunications, ... Les dimensions sont les critères sur lesquels on souhaite évaluer, quantifier, qualifier les faits : les dimensions usuelles sont le temps, le client, le magasin, la région, le produit...

Dans les exemples de requêtes citées au début de ce paragraphe, les faits et les dimensions apparaissent :

- les ventes en fait ;
- les produits, les clients, le temps, le lieu en dimensions.

Il sera souvent nécessaire de filtrer, d'agréger, de compter, sommer et de réaliser quelques statistiques élémentaires (moyenne, écart-type, ...). La structure logique doit être prévue pour rendre aussi efficace que possible toutes ces requêtes. Pour y parvenir, on est amené à introduire de la redondance dans les informations stockées en mémorisant des calculs intermédiaires (dans l'exemple, on peut être amené à stocker toutes les sommes de ventes par produit ou par année). On rompt donc avec le principe de non redondance des bases de production.

Si le critère de cohérence semble assuré avec les techniques du transactionnel, cette cohérence est toute relative. Elle se contrôle au niveau de la transaction élémentaire mais pas au niveau global et des activités de l'organisation. Pour les entrepôts, on requiert une cohérence interprétable par l'utilisateur. Par exemple, si les livraisons n'ont pas été toutes saisies dans le système, comment garantir la cohérence de l'état du stock ? Autre, exemple, pour établir un profil client ou étudier les performances d'un magasin, toutes les données utiles le concernant doivent être présentes dans le système, ce que n'assure pas le serveur transactionnel mais que doit assurer le serveur décisionnel.

Les entrepôts de données assureront donc plutôt une cohérence globale des données. Pour cette raison, leur alimentation sera un acte réfléchi et planifié dans le temps. Un grand nombre d'informations sera importé du système transactionnel lorsqu'on aura la garantie que toutes les données nécessaires auront été produites et mémorisées. Les transferts de données du système opérationnel vers le système décisionnel seront réguliers avec une périodicité bien choisie dépendante de l'activité de l'entreprise. Chaque transfert sera contrôlé avant d'être diffusé.

Une dernière caractéristique importante des entrepôts, qui est aussi une différence fondamentale avec les bases de production, est qu'aucune information n'y est jamais modifiée. En effet, on mémorise toutes les données sur une période donnée et terminée, il n'y aura donc jamais à remettre en cause ces données car toutes les vérifications utiles auront été faites lors de l'alimentation. L'utilisation se résume donc à un chargement périodique, puis à des interrogations non régulières, non prévisibles, parfois longues à exécuter.

1.2 Construction d'un entrepôt

L'entrepôt de données est donc bien différent des bases de données de production car les besoins pour lesquels on veut le construire sont différents. Il contient des informations historisées, globalement cohérentes, organisées selon les métiers de l'entreprise pour le processus de décision. L'entrepôt n'est pas un produit ou un logiciel mais un environnement. Il se bâtit et ne s'achète pas. Les données sont puisées dans les bases de production, nettoyées, normalisées, puis intégrées. Des métadonnées décrivent les informations dans cette nouvelle base pour lever toute ambiguïté quant à leur origine et leur signification.

Nous décrivons, dans ce chapitre, les problèmes liés à la construction d'un entrepôt, les modèles de données et son alimentation. Mais, il faut garder à l'esprit qu'un entrepôt se conçoit avec un ensemble d'applications qui vont réaliser ce pour quoi il a été construit : l'aide à la décision. Ce sont des outils d'accès aux données (requêteurs, visualisateurs, outils de fouille, ...) qui seront plus précisément décrits dans la suite de ce cours.

Nous avons relevé trois parties interdépendantes qui relèvent de la construction d'un entrepôt de données :

1. L'étude préalable qui va poser la question du retour sur investissement, définir les objectifs, préciser la démarche.
2. L'étude du modèle des données qui représente l'entrepôt conceptuellement et logiquement.
3. L'étude de l'alimentation qui reprend à un niveau plus précis l'examen des données, le choix des méthodes et des dates auxquelles les données entreront dans l'entrepôt.

Nous terminons par quelques remarques sur les machines dédiées aux entrepôts de données et les optimisations à mettre en œuvre pour assurer de bons temps de réponse.

1.2.1 Étude préalable

Cette partie de l'étude ressemble à toute étape préliminaire à l'implantation d'un nouveau système d'information automatisé. Les principes des méthodes connues pour le système de production restent valables ici.

Étude des besoins

L'étude des besoins doit déterminer le contenu de l'entrepôt et son organisation, d'après les résultats attendus par les utilisateurs, les requêtes qu'ils formuleront, les projets qui ont été définis. Le besoin d'informatisation peut provenir du système de pilotage ou d'un service particulier de l'entreprise. C'est souvent un projet au sein d'un schéma directeur qui va déclencher l'étude et la réalisation d'un cahier des charges. L'existant dans ce domaine est généralement un ensemble de petits produits ou développements sans grande intégration ni relations, disséminés dans divers services. L'information est dupliquée, les traitements répétés et aucune stratégie d'ensemble n'est définie. Le projet va donc s'orienter vers la recherche d'une solution intégrante résolument tournée vers l'utilisateur.

L'expression des besoins par les utilisateurs met souvent en évidence la volonté d'obtenir : des analyses sur ce qui s'est passé (par exemple comparer les performances actuelles d'un magasin avec celles de l'année dernière) ou des analyses prédictives (par exemple déterminer les achats potentiels pour un type de client, déterminer les clients qui risquent d'abandonner l'entreprise, ...).

Les interviews doivent permettre de préciser les faits à suivre et dans quelles dimensions. Il faut recenser les données nécessaires à un bon fonctionnement de l'entrepôt. Il faut alors recenser les données disponibles dans les bases de production, toutes les données de production ne sont pas utiles dans l'entrepôt. Il faut aussi identifier les données supplémentaires requises et s'assurer la possibilité de se les procurer (achat de bases géographiques, démographiques, ...).

L'examen des dimensions dans lesquelles les faits seront suivis doit donner lieu à une étude de l'unité de ces dimensions, de la granularité de ces faits. Par exemple, l'unité de temps doit-elle être le jour, la semaine ? Les produits sont-ils analysés par catégorie, par lot, par marque ?

La variété des besoins, leur modularité peut entraîner un découpage de l'entrepôt en plusieurs parties : *les datamart*. Les datamart sont alimentés par un entrepôt et sont dédiés à une activité particulière pour un ou plusieurs services (le suivi des clients, la prévision des stocks). On peut mener l'analyse soit de façon ascendante (en commençant par les datamart), soit de manière descendante.

Le lecteur intéressé est encouragé à consulter les neuf règles pour la conception d'entrepôts de données proposées par Ralf Kimball ([Kim97]).

Coûts de déploiement

Il faut une machine puissante, souvent une machine parallèle, spécialisée pour cette tâche. Les tentatives de mixer à la fois l'informatique de décision et l'informatique de production au sein d'une seule machine sont souvent des échecs. Comme nous l'avons vu précédemment les utilisations sont trop différentes. La capacité de stockage doit être très importante. Les prévisions de montée en charge du système devront évaluer cette quantité. Il faut noter que les prix de ces matériels ne cessent de décroître. En vertu de la loi de Moore, la puissance de stockage (resp. de calcul) est multipliée par deux tous les 2 ans (resp. 18 mois) et cette tendance s'accélère même, à prix constant.

Comme tout système informatique, il faut une équipe pour le maintenir, le faire évoluer (administrateur, développeurs, concepteurs, ...). Il faut prévoir la mise en place, la formation, etc.

Les coûts logiciels peuvent être importants. Ils concernent les logiciels d'administration de l'entrepôt, des logiciels d'interrogation et de visualisation ainsi que les coûts de l'environnement logiciel de data mining proprement dit. L'entrepôt devra être accessible par tout utilisateur et on se situe, généralement, dans un environnement client-serveur.

Bénéfices attendus

Nous verrons, au chapitre suivant, qu'il est possible de développer un projet d'extraction de connaissances à partir de données sans mise en place d'un entrepôt de données. Cependant, par définition de l'entrepôt, les données ont été préparées et le travail sera donc facilité.

Il est possible de calculer les espérances de gain en prévoyant les performances du système. Il faut, pour cela, effectuer une étude exhaustive des services de l'entreprise demandeurs d'un environnement d'aide à la décision à partir de données. Pour chacun des services, il faut recenser des projets sur lesquels des bénéfices peuvent être retirés d'une telle implantation. Pour chacun des projets, il faut alors estimer les bénéfices attendus.

Prenons l'exemple du service marketing direct qui veut augmenter le taux de réponse aux courriers qu'elle envoie. À l'aide d'un entrepôt de données, elle va mémoriser son activité et tenter d'établir des profils dans sa clientèle. Si un outil de fouille de données permet de prédire avec un certain taux d'erreur si un client répondra ou non au courrier, la société peut cibler son action. Avec des mailings maintenant ciblés, elle peut réduire le nombre d'envois tout en conservant le même nombre de retours. L'économie peut alors être quantifiée et comparée aux coûts d'investissement.

Il est important de noter que, au vu des investissements nécessaires, il est fréquent de commencer par le développement de serveurs départementaux pour l'entrepôt (Datamart) sur des projets clairement définis avant de passer à une généralisation à l'ensemble des services de l'entreprise.

L'étude préalable se conclue par la rédaction d'un cahier des charges comprenant la solution envisagée, le bilan du retour sur investissement et la décision d'implantation.

1.2.2 Modèles de données

Niveau conceptuel

Le modèle conceptuel doit être simplifié au maximum pour permettre au plus grand nombre d'appréhender l'organisation des données, comprendre ce que l'entrepôt mémorise. On parle de modèle multidimensionnel, souvent représenté sous forme de cube, parce que les données seront toujours des faits à analyser suivant plusieurs dimensions. Par exemple, dans le cas de ventes de produits à des clients dans le temps (3 dimensions, un vrai cube), les faits sont les ventes, les dimensions sont les clients, les produits et le temps. Les interrogations s'interprètent souvent comme l'extraction d'un plan, d'une droite de ce cube (*e.g.* lister les ventes du produit A ou lister les ventes du produit A sur période de temps D), ou l'agrégation de données le long d'un plan ou d'une droite (*e.g.* Obtenir le total des ventes du produit A revient à sommer les éléments du plan indiqué en figure 1.1).

Un avantage évident de ce modèle est sa simplicité, dès que les mots ventes, clients, produits et temps sont précisés, désambiguisés. Il reprend les termes et la vision de l'entreprise de tout utilisateur final concerné par les processus de décision. Même si dans un entrepôt important il peut exister plusieurs cubes, parce qu'il est nécessaire de suivre plusieurs faits dans des directions parfois identiques, parfois différentes, l'utilisateur pourra accéder à des versions simplifiées, car plus ciblées, dans des datamarts.

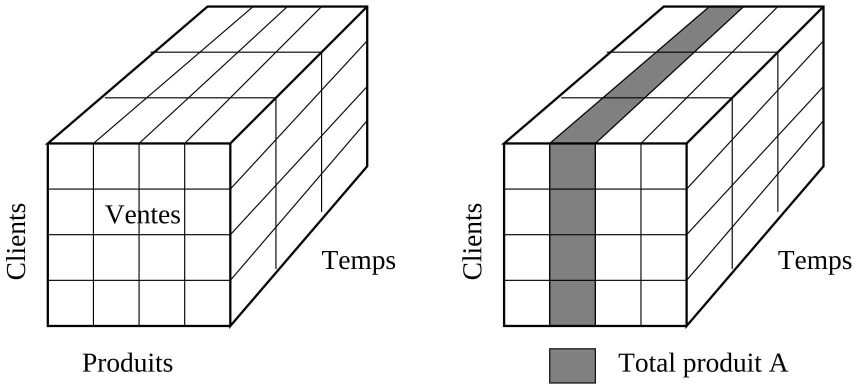

FIG. 1.1 – *Modèle en cube*

Niveau logique

Au niveau logique, l'unité de base est la table comme dans le modèle relationnel. L'implantation classique consiste à considérer *un modèle en étoile* avec au centre la table des faits et les dimensions comme autant de branches à l'étoile. Les branches de l'étoile sont des relations de 1 à plusieurs, la table des faits est énorme contrairement aux tables des dimensions. Le modèle est en cela très dissymétrique en comparaison avec les modèles relationnels des bases de production. Encore une fois, la simplicité du modèle obtenu rend la construction en étoile très attrayante.

Les faits sont qualifiés par des champs qui sont le plus souvent numériques et cumulatifs comme des prix, des quantités et des clés évidemment pour relier les faits à chaque dimension. Les tables des dimensions sont caractérisées par des champs le plus souvent textuels. Dans l'exemple présenté en figure 1.2, nous nous limitons au suivi d'un seul fait : le montant des ventes. Un enregistrement dans la table des faits `Ventes` correspond à un total des ventes à un client dans une tranche horaire d'un jour précis, pour un produit choisi.

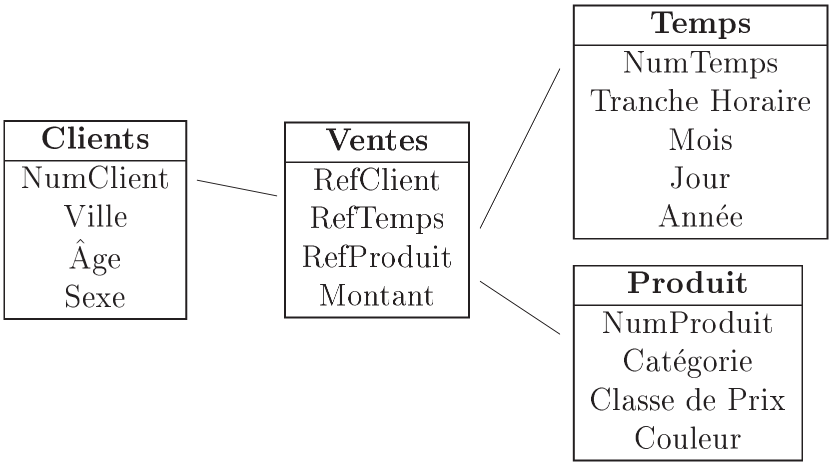

FIG. 1.2 – *Modèle en étoile*

Il existe un modèle concurrent : *le modèle en flocon*. L'avantage mis en avant par les tenants de ce modèle est l'économie de place de stockage. Mais, le plus souvent, un examen approfondi montre que cette structure n'apporte rien, et complique même le modèle. Quand une hiérarchie apparaît dans une dimension, il est préférable de tout enregistrer dans une seule et même table formant une grande dimension. Par exemple, pour une dimension produit avec des catégories, puis des sous-catégories (et ainsi de suite), toutes les échelles sont conservées dans la même table de dimension. Il sera ensuite possible de *naviguer* (ou *forer*) par des opérations de zoom dans cette échelle (en anglais *drill up* et *drill down*).

1.2.3 Alimentation

L'alimentation est la procédure qui permet de transférer des données du système opérationnel vers l'entrepôt de données en les adaptant. La conception de cette opération est une tâche assez complexe. Elle doit être faite en collaboration avec les administrateurs des bases de production qui connaissent les données disponibles et vérifient la faisabilité des demandes. Il est nécessaire de déterminer quelles données seront chargées, quelles transformations et vérifications seront nécessaires, la périodicité et le moment auxquels les transferts auront lieu.

Traditionnellement, ce sont en grande partie des faits qui seront ajoutés. Les dimensions évoluent selon leur nature soit peu soit assez rapidement. Dans notre exemple, nous pouvons imaginer que la gamme des produits n'est pas très souvent modifiée. Par contre, de nouveaux clients seront certainement ajoutés mais bien moins que des dates. Le chargement implique aussi des problèmes physiques comme le recalcul des clés d'index. L'alimentation est ponctuée d'une procédure d'assurance qualité qui veillera à ce que tout se soit déroulé correctement avant la publication du nouveau jeu de données.

Transformations, vérifications

L'étude des besoins a déterminé le contenu de l'entrepôt en partant des desiderata des utilisateurs. Néanmoins, la forme, le contenu des données de production ne convient pas toujours immédiatement au format choisi pour les données de l'entrepôt. Par conséquent, des transformations sont souvent nécessaires.

Format Le format physique des données provenant de la production peut ne pas être adéquat avec le système hôte de l'entrepôt. Des transformations de type sont parfois nécessaires (Système IBM vers système Unix...). Les données pouvant provenir de serveurs différents dans des services différents, il est nécessaire d'uniformiser les noms et les formats des données manipulées au niveau de l'entrepôt.

Consolidation Selon les choix des unités pour les dimensions, des opérations de consolidation devront accompagner le chargement des données (par exemple sommer les ventes pour obtenir et enregistrer un total par jour et non pas toutes les transactions).

Uniformisation d'échelle Pour éviter de trop grandes dispersions dans les valeurs numériques, une homogénéisation des échelles de valeurs est utile. Ne pas la réaliser peut pénaliser les outils d'analyse et de visualisation et peut-être simplement remplir inutilement les disques.

Autres Des transformations qui permettent de mieux analyser les données sont aussi réalisées pendant la phase de chargement. Par exemple, la transformation de la date de naissance en âge, assure une plus grande lisibilité des données et permet de pallier les problèmes apparus avec l'introduction de la dimension temps.

Malgré les efforts réalisés pour assurer l'intégrité des données de production, des erreurs peuvent survenir, en particulier, lorsque les données proviennent de sources différentes (par exemple, il est fréquent qu'un même client soit mémorisé plusieurs fois sur différents serveurs). Parmi les points à vérifier, on peut citer:

Erreurs de saisie Des doublons sont présents mais sont invisibles ; à cause des fautes de frappe: (Marcel dupont; 3,rue verte; Lille) et (Marcel dupond; 3,rue verte; Lille) sont certainement un seul et même client ; plusieurs membres d'un même foyer peuvent être présents ; ...

Intégrité de domaine Un contrôle sur les domaines des valeurs permet de retrouver des valeurs aberrantes. De façon plus générale, des valeurs douteuses peuvent se rencontrer, comme par exemple des dates au 11 novembre 1911 (11/11/11) ou 1 janvier 1901 (01/01/01).

Informations manquantes Des champs importants pour lesquels aucune valeur n'a été saisie peuvent pénaliser le processus de découverte d'information, ou bien encore avoir une signification particulière (ex: détection de fraudes). Il est parfois important d'insérer des valeurs par défaut significatives (comme NULL) plutôt que de laisser ces données vides.

Il convient de noter que les sources des données alimentant un entrepôt peuvent être hétérogènes. Les bases de production peuvent être nombreuses, différentes et délocalisées géographiquement. Des fichiers peuvent être achetées auprès d'entreprises qui se sont spécialisées dans la constitution et la revente de fichiers qui vont aussi entrer dans le processus d'alimentation de l'entrepôt. Les suites logicielles d'accompagnement d'entrepôts de données contiennent des outils susceptibles d'aider à développer des procédures d'alimentation qui prennent en compte ces problèmes de vérification et de normalisation.

Métadonnées

Des choix conceptuels, logiques et organisationnels ont été opérés tout au long de cette démarche d'informatisation du système d'information de décision. La perte de l'information sur ces choix peut induire de mauvaises interprétations. Donc, un annuaire spécialisé conserve toutes les informations (les métadonnées) au sujet du système d'information qui régit l'entrepôt. Cet annuaire contient entre autres choses :

- Les adresses et descriptions des objets de l'entrepôt : tables, champs, ... En particulier chaque donnée est définie par une description précise et claire, son origine, ... surtout lorsque les données proviennent de bases différentes et qu'elles peuvent avoir d'autres significations par ailleurs ;
- les utilisateurs, autorisations, . . . ;
- les règles de transformation, de vérification ;
- l'historique de l'entrepôt avec les dates de chargement, ...

Sans référentiel qui qualifie de façon précise ce que signifie chaque valeur dans la base, il n'est pas possible de conduire une analyse et interpréter les résultats. C'est ce rôle que joue l'annuaire des métadonnées.

Certification, publication

Comme nous l'avons montré précédemment, la cohérence de l'entrepôt est globale et ce n'est qu'à la fin de la procédure de chargement qu'il est possible de la vérifier. Une procédure de certification de la qualité des données chargées doit suivre le chargement. Cette procédure est particulière à l'entrepôt et ne peut être décrite précisément. Elle va par exemple vérifier la cohérence des données au niveau des activités. En cas de succès, la nouvelle version de l'entrepôt sera publiée. Dans le cas contraire, c'est souvent le chargement complet qui est annulé et repoussé à une date ultérieure. En fait, le chargement peut être vu comme une très grosse (et longue) transaction.

1.3 Utilisation, exploitation

L'alimentation des entrepôts s'accompagne, après validation, de l'édition automatique des tableaux de bord les plus courants. Ils sont prédéfinis, réalisés par le service informatique, et sont le reflet d'un besoin explicitement demandé au moment de la conception. Souvent, ils sont insuffisants lorsqu'une anomalie est détectée ou lorsqu'un nouveau besoin s'exprime. L'utilisateur final doit alors pouvoir interroger les données *en ligne* à l'aide d'outils simples et conviviaux. Ces outils commencent à se généraliser. Les éditeurs les nomment (ou les classent) : reporting tools, managed queries, Executive Information Systems (EIS), OLAP tools (Online analytical Processing), ... bien que les différences entre tous ces systèmes ne soient pas toujours très nettes.

On suppose avoir défini un entrepôt de données pour lequel les données sont organisées selon un modèle en étoile. Nous allons préciser les outils attendus dans un tel environnement : outils de requête (SQL) et d'analyse multi-dimensionnelle ; les opérations possibles pour l'utilisateur en analyse multi-dimensionnelle ; les outils de visualisation.

1.3.1 Requêtes

Nous présentons ici les outils destinés à l'utilisateur final qui permettent d'extraire des données de l'entrepôt.

Les outils de création de rapport (reporting tools) extraient les données et proposent une mise en forme destinée à la diffusion : par impression ou par des services internet ou intranet. Ils sont très utilisés pour générer des tableaux de bord conventionnels, qui sont souvent composés et diffusés automatiquement et périodiquement sans demande spécifique des utilisateurs. Lorsque leur intégration dans le système d'information est réussie, ils mettent en évidence la structure multidimensionnelle et présentent les agrégats, supportent la navigation. Ils sont accessibles aux utilisateurs finals pour créer de nouveaux tableaux de bord. Certains (managed queries tools comme par exemple Business Object) ne sont pas spécialisés pour les entrepôts de données mais proposent une métacouche entre l'utilisateur final et le système d'information destinée à simplifier le vocabulaire, éviter le SQL.

Les progiciels (ex : SAS) dans ce domaine ont aussi réalisé une percée importante. Ils sont souvent qualifiés de EIS tools et ajoutent des analyses classiques et paramétrables pour les ventes, les achats ou la finance par exemple.

Les outils les plus adaptés sont certainement les outils OLAP, néanmoins, on pourra relever quelques défauts qui expliquent peut-être la difficulté qu'ils ont à s'imposer. À l'origine, E.F. Codd précurseur dans les bases de données relationnelles a aussi écrit 12 règles pour définir ce que signifie OLAP. Ces règles portent sur l'évidence de multiples dimensions, la transparence du système, l'accessibilité aux utilisateurs, la performance, l'intégration dans le client-server, l'extensibilité des dimensions, la navigation, la flexibilité,... Les éditeurs ont tenté de bâtir des outils se réclamant de ces règles. Malheureusement, il est difficile de vérifier les principes OLAP seulement avec le langage SQL. De ce fait, la plupart des outils sont maintenant des systèmes propriétaires non ouverts et non standardisés. De plus, ils ne peuvent pas à l'heure actuelle traiter des bases de données volumineuses.

1.3.2 Agrégats et navigation

L'opération de *navigation* (ou forage) permet d'obtenir des détails sur la signification d'un résultat en affinant une dimension ou en ajoutant une dimension. Elle apparaît dans de nombreux outils et doit (parce qu'elle est souvent coûteuse) être intégrée dans le système. Pour illustrer le forage, supposons qu'un utilisateur final demande les chiffres d'affaires par produit, et s'étonne d'un résultat pour un produit donné. Il aura sûrement l'envie d'en analyser les raisons. Une solution consisterait à ajouter la dimension temps, dans l'unité de temps trimestrielle pour trouver une variation saisonnière, dans l'unité hebdomadaire pour envisager l'effet week-end, ou encore la dimension magasin pour mettre en évidence un effet géographique.

Pour des raisons de performance, il est utile de précalculer et préenregistrer dans l'entrepôt des agrégations de données. Dans notre exemple, si des requêtes fréquemment formulées impliquent le calcul de la somme des ventes par produit et par trimestre, il est alors simple d'optimiser le temps de réponse du système en mémorisant ce calcul. On introduit une redondance dans l'entrepôt mais elle se justifie de deux façons : d'une part on privilégie le temps de réponse ; d'autre part, puisque par principe les données d'un entrepôt ne sont pas modifiées, il n'est pas possible de rendre la base incohérente.

C'est une fois de plus l'étude des besoins qui doit mettre en évidence la création des agrégats. Pour insister sur l'importance de mémoriser les résultats de calculs on peut noter que des machines sont parfois dédiées à leur exécution et leur diffusion : ce sont des *serveurs d'agrégats*.

La stratégie de mémorisation de ces agrégats peut prendre deux formes. La première solution consiste à créer de nouvelles tables de faits pour les agrégats. Toutes les dimensions inutiles ou incohérentes avec l'agrégat seront supprimées. La deuxième solution consiste à enrichir la table initiale des faits par les agrégats avec une information supplémentaire indiquant le niveau de regroupement. Ces deux solutions ont quasiment la même incidence sur les volumes.

Pour expliquer un résultat, il est parfois nécessaire de le comparer avec d'autres faits. Par exemple, la baisse des ventes pour le mois de janvier peut s'expliquer par une baisse des achats ou une rupture de stock. Si l'entrepôt est conçu pour suivre les ventes et les achats ou le stock, et si les dimensions selon lesquelles ces trois faits sont suivis sont identiques, on doit pouvoir réaliser un rapport unique. On parle alors de *forage transversal* ou *drill across*. C'est une opération qu'il faut réaliser avec beaucoup de soins car mettre en œuvre une requête sur plusieurs tables de faits peut se révéler irréalisable. Engagée sans précautions, la requête va générer une table intermédiaire énorme qui sera le produit cartésien entre les deux tables de faits.

1.3.3 Visualisation

Les outils de visualisation sont très importants dans le processus de décision et peuvent intervenir à plusieurs niveaux. Ils sont utiles pour

1. découvrir de nouvelles informations, parce qu'une représentation permet de repérer plus simplement des singularités, des anomalies ;
2. présenter des résultats, dans l'optique d'une large diffusion, parce qu'un graphique est plus accessible qu'un tableau de chiffres ;
3. représenter un modèle issu d'une opération de fouille de données (représenter un arbre de décision, un ensemble des règles, un réseau de neurones...). Nous ne développons pas ce point.

Dans le premier cas, ils sont intégrés dans les outils d'analyse et doivent supporter des opérations comme comparer, modifier les échelles, retrouver les données correspondant à un point ou un objet tracé, zoomer sur des régions ou des sous-ensembles et enfin permettre la navigation (drill-up, drill down).

Lorsqu'il est nécessaire de présenter un objet reposant sur plusieurs dimensions sur un écran ou une feuille de papier, des problèmes bien connus apparaissent. Les représentations 3D ne sont pas toujours adaptées : les objets sont déformés, certains sont cachés... Des animations sont nécessaires pour faire pivoter la représentation. Les couleurs, utilisées en grand nombre, sont difficiles à choisir, facile à confondre. La visualisation scientifique reste un secteur de la recherche actif où vont se mêler des arguments informatiques, psychologiques, ...

Terminons en insistant sur l'importance du choix du type de graphique utilisé même en représentation 2D (surtout en situation de large diffusion). Un graphique en camembert représente une fonction de répartition (les parties en rapport avec le tout). Les courbes sont utiles lorsque les deux échelles sont numériques (et continues). Les histogrammes sont préférés quand une dimension est discrète (une liste de régions, une liste de produits). Souvent, l'axe des abscisses est réservé aux échelles respectant un ordre (des dates par exemple). Les superpositions de plusieurs séries sous forme d'aires ou de barres d'histogrammes sont réservées lorsque la somme des séries a un sens.

1.3.4 Supports physiques, optimisations

Nous terminons ce chapitre par quelques éléments à prendre en compte lors de l'implantation d'un entrepôt de données.

Calcul des volumes

C'est une évaluation nécessaire et d'ailleurs traditionnelle à réaliser au moment de la rédaction du cahier des charges lorsqu'on désire construire une base de données ou un entrepôt. Dès que les études logiques ont abouti, il est nécessaire d'estimer la place que les données occuperont sur disque.

Prenons par exemple le suivi des ventes d'une chaîne de 300 magasins suivant les dimensions temps, magasin, produit, promotion. Les faits suivis sont : le prix de vente, la quantité vendue, le prix d'achat, le nombre d'achats réalisé. Un enregistrement de la table des faits aura donc une structure logique de la forme :

Fait(RefTps,RefProduit,Refmagasin,Refpromo,
PrixVente,QtéVendue,PrixAchat,QtéAchat)

On estime conserver les enregistrements sur 2 ans jour par jour, soit 730 jours. Les 300 magasins vendent en moyenne 3000 produits par jour parmi 30000 possibles. Les produits ne bénéficient que d'une seule promotion dans un magasin à un jour donné. On aura donc $730 \times 300 \times 3000 \times 1 = 657$ millions d'enregistrements dans la table des faits. Suivant le système choisi, et la représentation physique des clés (4 clés) et des entiers (4 champs numériques à suivre), on peut estimer la taille de la table des faits à environ 21 Go. Il reste à estimer la taille des index, la taille des tables de dimension etc ...

Matrices creuses

Avec l'exemple ci dessus, on s'aperçoit que la table des faits ne contient pas un enregistrement pour chaque élément dans chaque dimension. Par exemple, 300 des 3000 produits sont vendus chaque jour dans un magasin et chaque produit ne bénéficie pas d'une promotion. En revenant à la définition conceptuelle des données en cube, on interprète cette constatation en disant que beaucoup de cubes unitaires ne contiennent aucune donnée, formant ainsi comme une matrice creuse (*i.e.* comportant beaucoup de zéros). Cette propriété peut-être exploitée pour améliorer la performance des algorithmes et optimiser le stockage.

Le problème des clés et des index

La gestion des clefs et des index est un problème important dans ces grandes tables. Il faut envisager leur évolution, la possibilité de les étendre sans les régénérer. L'organisation physique sur disque est déterminante pour les questions d'efficacité. Toutes ces questions sont très spécifiques et dépendent des machines utilisées (machines parallèles) et du type d'accès aux données.

La table des faits aura des index pour chaque clé étrangère et bien sûr pour les regroupements les plus fréquents de clés étrangères. Dans notre exemple, l'index sur la clé étrangère `RefProduit` est utile pour retrouver rapidement les achats d'un produit donné. Un index sur `RefProduit.RefTemps` (les 2 champs à la fois) est utile pour retrouver les achats d'un produit donné dans un mois donné. Les tables de dimension sont souvent indexées suivant tous leurs champs.

Mais les gains les plus remarquables s'obtiennent par deux techniques : utiliser des *machines parallèles* et *précalculer des agrégats*.

Machines parallèles

L'utilisation de machines parallèles est très adéquat pour les entrepôts de données et peut-être plus simple à mettre en œuvre que dans le cas du transactionnel. La caractéristique de n'accèder à l'entrepôt qu'en lecture permet de dupliquer des données, d'accepter plusieurs utilisateurs simultanés sans craindre des incohérences.

Les architectures parallèles considérées ici sont des machines à plusieurs processeurs qui partagent de la mémoire, du disque ou rien du tout. Un réseau de communication performant relie les processeurs. Ces machines améliorent les performances de plusieurs façons, suivant leur architecture, la répartition des données sur tous les disques et le grain de parallélisme envisagé. Le grain définit le niveau atomique ou indécomposable des tâches, soit encore le niveau auquel peut intervenir la mise en parallèle des tâches. On peut considérer que l'unité indécomposable est la requête, ou les opérations de base, comme la jointure, la sélection, le tri, le regroupement. Un pipe-line de tâches est aussi parfois possible. Par exemple, le résultat d'une sélection faite par un processeur est directement envoyé vers un second processeur qui effectue un tri et le tri peut commencer avant que la sélection soit terminée.

Si les données nécessaires à deux tâches exécutées en parallèle se trouvent sur la même unité de disque, l'exécution ne sera pas optimale, puisque les tâches devront partager une ressource, l'accès au disque. Inversement, un trop grand éclatement des données va pénaliser les tâches qui perdront du temps en communication. Dans cette approche, la répartition des données est donc très importante. Plusieurs solutions sont proposées mais aucune n'est universelle et c'est une étude au cas par cas qu'il convient de mener. Les répartitions usuelles tiennent compte d'une table de hachage, des clés, du schéma des données ou encore se réalisent à la demande.

Agrégats

Le précalcul de certains groupes les plus fréquemment demandés est l'optimisation la plus performante. Les gains vont d'un rapport de 10 à un rapport de 1000. Dans certains cas, des *serveurs d'agrégats*, des machines dédiées à distribuer des calculs intermédiaires sur les groupes, sont intégrées dans l'architecture de l'entrepôt.

Chapitre 2

Processus de découverte d'information

Nous présentons dans ce chapitre le processus complet d'extraction de connaissances. Nous nous limitons au traitement de données structurées. Il n'est pas question ici de recherche dans des textes ou des images, mais uniquement à partir de données. S'il est possible d'extraire de l'information à partir de toute source de données, l'existence d'un entrepôt diminue le temps de réalisation d'un projet.

Le processus que nous présentons maintenant est découpé en six parties : préparation, nettoyage, enrichissement, codage, fouille et validation. L'enchaînement des différentes étapes est présenté dans la figure 2.1.

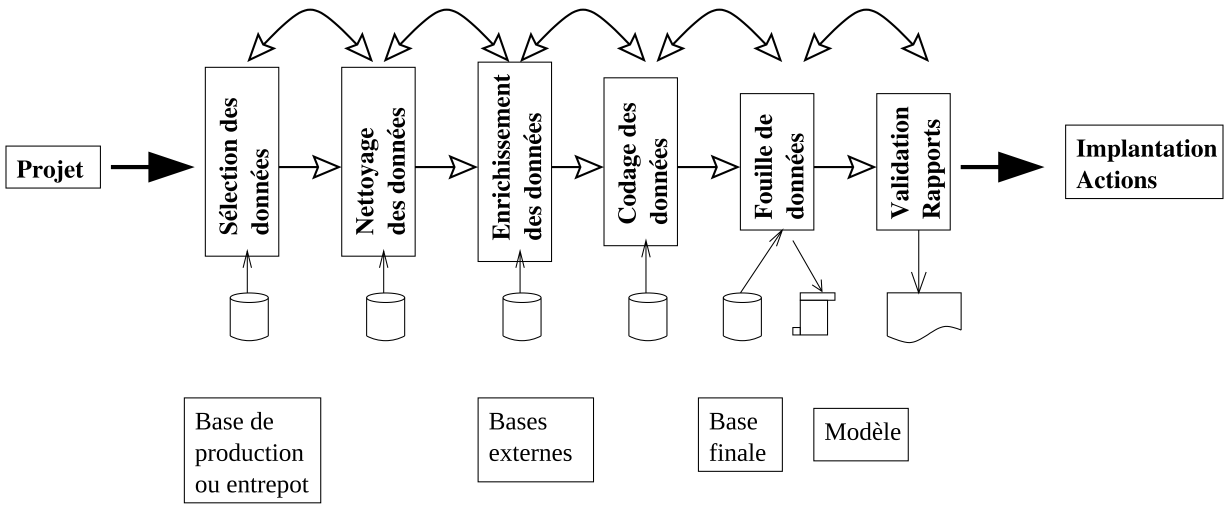

FIG. 2.1 – *Les étapes du processus d'extraction de connaissances à partir de données*

Le déroulement d'un projet n'est pas linéaire. On peut constater dans l'étape de validation que les performances obtenues ne sont pas suffisantes ou que les utilisateurs du domaine jugent l'information inexploitable, il s'agira alors de choisir une autre méthode de fouille, ou de remettre en cause les codages, ou de chercher à enrichir les données. On évalue dans un projet le temps passé à l'étape de fouille de données (qui est l'étape de découverte d'informations proprement dite) à moins de 20% du temps. Par conséquent, plus de 80% du temps est dédié aux opérations de sélection, nettoyage, enrichissement et codage.

Nous présentons maintenant l'exemple qui nous servira pour illustrer chaque étape du processus. Cet exemple est issu du livre de P. Adriaans et D. Zantige [AZ96].

> Un éditeur vend 5 sortes de magazines : sport, voiture, maison, musique et BD. Il souhaite mieux étudier ses clients pour découvrir de nouveaux marchés ou vendre plus de magazines à ses clients habituels. Les questions qu'il se pose sont :
>
> 1. « *Combien de personnes ont pris un abonnement à un magazine de sport cette année ?* »
> 2. « *A-t-on vendu plus d'abonnements de magazines de sport cette année que l'année dernière ?* »
> 3. « *Est-ce que les acheteurs de magazines de BD sont aussi amateurs de sport ?* »
> 4. « *Quelles sont les caractéristiques principales de mes lecteurs de magazines de voiture ?* »
> 5. « *Peut-on prévoir les pertes de clients et prévoir des mesures pour les diminuer ?* »

Les questions qui sont proposées ici sont de natures différentes et mettent en jeu des processus différents.

De simples requêtes SQL sont suffisantes pour répondre aux deux premières questions. Les données recherchées dans ce cas ne sont que le résultat d'un calcul simple sur un ou des groupes d'enregistrements. Ce qui distingue ces deux premières questions, c'est la notion de temps et la comparaison. Pour établir cette comparaison, les données doivent être présentes dans la base, ce qui n'est pas toujours le cas pour les bases opérationnelles. Nous pouvons donc supposer que la requête 1 est réalisable sans technique particulière à partir des données opérationnelles sous réserve d'indexations suffisantes des tables concernées. La seule difficulté est de ne pas pénaliser le serveur transactionnel par des requêtes trop longues.

La requête 2 nécessite de conserver toutes les dates de souscription même pour les abonnements résiliés. Nous pouvons imaginer aussi que l'utilisateur émettra une grande variété de requêtes de ce type. Nous supposons alors que ces questions seront résolues à l'aide d'outils de création de requêtes multidimensionnelles de type OLAP.

La question 3 est un exemple simplifié de problème où l'on demande si les données vérifient une règle. La réponse pourrait se formuler par une valeur estimant la probabilité que la règle soit vraie. Souvent, des outils statistiques sont utilisés. Cette question peut être généralisée. On pourrait chercher des associations fréquentes entre acheteurs de magazine pour effectuer des actions promotionnelles. On pourrait également introduire une composante temporelle pour chercher si le fait d'être lecteur d'un magazine implique d'être, plus tard, lecteur d'un autre magazine.

La question 4 est beaucoup plus ouverte. La problématique ici est de *trouver* une règle et non plus de la vérifier ou de l'utiliser. C'est pour ce type de question que l'on met en œuvre des outils de fouille de données et donc le processus décrit ci-dessous.

La question 5 est également une question ouverte. Il faut pour la résoudre disposer d'indicateurs qui pourraient sur notre exemple être : les durées d'abonnement, les délais de paiement, ... Ce type de question a une forte composante temporelle. Elle nécessite des données historiques. Elle a de nombreuses applications dans le domaine bancaire, par exemple.

Le processus de KDD utilise des outils de création de requêtes, des outils statistiques et des outils de fouille de données. Là encore, nous nous apercevons qu'une très grande partie des problèmes de décision se traite avec des outils simples. La fouille de données quand elle est nécessaire, suit souvent une analyse des données simple (OLAP) ou statistique.

2.1 Préparation des données

Dans notre exemple, nous avons identifié quelques objectifs précis, exprimés sous forme de questions.

La préparation des données consiste dans un premier temps à obtenir des données en accord avec les objectifs que l'on s'impose. Ces données proviennent le plus souvent de bases de production ou d'entrepôts. Les données sont structurées en champs typés (dans un domaine de définition).

Dans l'exemple, nous partons d'une liste des souscriptions d'abonnements avec cinq champs (voir figure 2.2).

N^o client	Nom	Adresse	Date d'abonnement	Magazine
23134	Bemol	Rue du moulin, Paris	7/10/96	Voiture
23134	Bemol	Rue du moulin, Paris	12/5/96	Musique
23134	Bemol	Rue du moulin, Paris	25/7/95	BD
31435	Bodinoz	Rue verte, Nancy	11/11/11	BD
43342	Airinaire	Rue de la source, Brest	30/5/95	Sport
25312	Talonion	Rue du marché, Paris	25/02/98	NULL
43241	Manvussa	NULL	14/04/96	Sport
23130	Bemolle	Rue du moulin, Paris	11/11/11	Maison

FIG. 2.2 – *Obtention des données*

Ces données sont tout d'abord copiées sur une machine adéquate, pour des questions de performance, mais surtout parce qu'elles seront modifiées.

L'obtention des données est souvent réalisée à l'aide d'outils de requêtage (OLAP, SQL,...).

Il faut, dès a présent, noter que l'on ne peut résoudre des problèmes que si l'on dispose des données nécessaires. Il semble, par exemple, impossible de s'attaquer à la question 5 de notre exemple avec les données dont nous disposons.

2.2 Nettoyage

Il y a évidemment de nombreux points communs entre nettoyage avant l'alimentation des entrepôts (1.2.3) et avant la fouille. Bien sûr, lorsque les données proviennent d'un entrepôt, le travail est simplifié mais reste néanmoins nécessaire : nous avons maintenant un projet bien défini, précis et les données doivent être les plus adaptées possibles.

Reprenons quelques remarques citées dans le chapitre précédent.

Doublons, erreurs de saisie Les doublons peuvent se révéler gênants parce qu'ils vont donner plus d'importance aux valeur répétées. Une erreur de saisie pourra à l'inverse occulter une répétition.

Dans notre exemple, les clients numéro 23134 et 23130 sont certainement un seul et même client, malgré la légère différence d'orthographe.

Intégrité de domaine Un contrôle sur les domaines des valeurs permet de retrouver des valeurs aberrantes.

Dans notre exemple, la date de naissance du client 23130 (11/11/11) semble plutôt correspondre à une erreur de saisie ou encore à une valeur par défaut en remplacement d'une valeur manquante.

Informations manquantes C'est le terme utilisé pour désigner le cas où des champs ne contiennent aucune donnée. Parfois, il est intéressant de conserver ces enregistrements car l'absence d'information peut être une information (ex: détection de fraudes). D'autre part, les valeurs contenues dans les autres champs risquent aussi d'être utiles.

Dans notre exemple, nous n'avons pas le type de magazine pour le client 25312. Il sera écarté de notre ensemble. L'enregistrement du client 43241 sera conservé bien que l'adresse ne soit pas connue.

N° client	Nom	Adresse	Date d'abonnement	Magazine
23134	Bemol	Rue du moulin, Paris	7/10/96	Voiture
23134	Bemol	Rue du moulin, Paris	12/5/96	Musique
23134	Bemol	Rue du moulin, Paris	25/7/95	BD
31435	Bodinoz	Rue verte, Nancy	NULL	BD
43342	Airinaire	Rue de la source, Brest	30/5/95	Sport
43241	Manvussa	NULL	14/04/96	Sport
23130	Bemol	Rue du moulin, Paris	NULL	Maison

FIG. 2.3 – *Après nettoyage*

Là encore, le langage SQL est utilisé pour la recherche de doublons et des informations manquantes.

2.3 Enrichissement

On peut avoir recours à d'autres bases, achetées ou produites en un autre lieu, pour enrichir nos données. L'opération va se traduire par l'ajout de nouveaux champs en conservant souvent le même nombre d'enregistrements.

Une première difficulté ici est de pouvoir relier des données qui parfois sont hétérogènes. Des problèmes de format de données apparaissent et des conversions sont souvent nécessaires. Une deuxième difficulté est l'introduction de nouvelles valeurs manquantes ou aberrantes et la phase de nettoyage sera certainement de nouveau utile.

Dans l'exemple, supposons que nous ayons accès à des informations sur les clients (voir figure 2.4).

client	Date de naissance	Revenus	Propriétaire	Voiture
Bemol	13/1/50	20 000 F	Oui	Oui
Bodinoz	21/5/70	12 000 F	Non	Oui
Airinaire	15/06/63	9 000 F	Non	Non
Manvussa	27/03/47	15 000 F	Non	Oui

FIG. 2.4 – *Enrichissment*

2.4 Codage, normalisation

À ce stade du processus, les choix sont particulièrement guidés par l'algorithme de fouille utilisé et des ajustements des choix de codage sont souvent nécessaires.

Regroupements Certains attributs prennent un très grand nombre de valeurs discrètes. C'est typiquement le cas des adresses. Lorsqu'il est important de considérer ces attributs pour la fouille de données il est obligatoire d'opérer des regroupements et ainsi obtenir un nombre de valeurs raisonnable.

Dans l'exemple, nous regroupons les adresses en 2 régions : Paris, province.

Attributs discrets Les attributs discrets prennent leurs valeurs (souvent textuelles) dans un ensemble fini donné. C'est le cas de la colonne *magazine* dans notre exemple qui peut prendre les valeurs *Sport, BD, Voiture, Maison, Musique.*

Deux représentations sont possibles pour ces données : une représentation verticale telle qu'elle est présentée en figure 2.3 ou une représentation horizontale ou éclatée (figure 2.5).

La représentation horizontale est plus adaptée à la fouille de données et certains calculs sont simplifiés. Par exemple, la somme de la colonne sport que divise le nombre

N°	Sport	BD	Voiture	Maison	Musique
23134	0	1	1	0	1
31435	0	1	0	0	0
43342	1	0	0	0	0
43241	1	0	0	0	0

FIG. 2.5 – *Codage des attributs discrets*

d'enregistrements calcule le pourcentage de clients ayant souscrit un abonnement à un magazine de sport. Notons que la date d'abonnement dépend du type de magazine. De façon générale, la modification présentée en figure 2.5 peut induire une perte d'information pour tous les champs en dépendance fonctionnelle avec le champ transformé. Nous choisissons de transformer le champ *date d'abonnement* en *date du plus vieil abonnement*. Il est à noter que cette transformation ne nous permet plus espérer générer des règles de la forme : un acheteur de BD s'abonne à un magazine de musique dans les deux ans qui suivent.

Dans notre exemple, le même codage en deux valeurs 0 et 1 sera réalisé avec les champs Oui/Non issus de l'enrichissement.

Changements de type Pour certaines manipulations, comme des calculs de distance, des calculs de moyenne, il est préférable de modifier les types de certains attributs. Dans notre exemple, la date de naissance et la date d'abonnement ne permettent pas d'effectuer simplement des comparaisons, des différences. Nous pouvons les convertir en âge ou en durée.

Uniformisation d'échelle Certains algorithmes, comme la méthode des plus proches voisins, sont basés sur des calculs de distance entre enregistrements. Des variations d'échelle selon les attributs sont autant de perturbations possibles pour ces algorithmes. Des échelles très « étirées » vont artificiellement rendre des dimensions trop « vides ».

C'est typiquement le cas pour le champ *Revenus* dans notre exemple. Les centaines de francs ne sont pas significatives ici. Nous convertissons donc les revenus en les divisant par mille. L'intervalle de valeurs est alors dans la même échelle que les dates de naissance et les durées d'abonnement.

N°	Sport	BD	Voiture	Maison	Musique	Date de naissance	Revenus	Propriétaire	Voiture	Paris?	Durée d'abonnement
23134	0	1	1	0	1	50	20	Oui	Oui	1	4
31435	0	1	0	0	0	30	12	Non	Oui	0	NULL
43342	1	0	0	0	0	37	9	Non	Non	0	5
43241	1	0	0	0	0	53	15	Non	Oui	NULL	4

FIG. 2.6 – *Codage des attributs discrets et normalisation.*

2.5 Fouille

La fouille de données est le cœur du processus car elle permet d'extraire de l'information des données. Néanmoins, c'est souvent une étape difficile à mettre en œuvre, coûteuse et dont les résultats doivent être interprétés et relativisés. Il faut aussi noter qu'en situation réelle, pour l'aide à la décision, une très grande majorité des résultats recherchés s'obtient uniquement par requêtes, par analyse multi-dimensionnelle ou grâce aux outils de visualisation.

Une approche traditionnelle pour découvrir ou expliquer un phénomène est de

1. regarder, explorer,
2. établir un modèle ou une hypothèse,
3. essayer de le contredire ou le vérifier comme en 1 ; recommencer le point 2 jusqu'à obtenir une réponse de qualité satisfaisante.

La partie 1 est traditionnellement réalisée avec des outils de reporting ou d'analyse multidimensionnelle. La partie 2 est une hypothèse émise par l'uilisateur. On peut voir les outils de fouille de données comme des procédures qui permettent de faciliter ou encore d'automatiser ce processus.

La qualité du modèle obtenu se mesure selon les critères suivants :

– Rapide à créer ;
– rapide à utiliser ;
– compréhensible pour l'utilisateur ;
– les performances sont bonnes ; Le modèle est fiable ;
– les performances ne se dégradent pas dans le temps ;
– Il évolue facilement.

Il va de soit qu'aucun modèle n'aura toutes ces qualités. Il n'existe pas de meilleure méthode de fouille. Il faudra faire des compromis selon les besoins dégagés et les caractéristiques connues des outils. Dans le chapitre 2.5 nous en présentons quelques unes avec leurs avantages et inconvénients, des indications pour mieux les choisir. Pour une utilisation optimale, une combinaison de méthodes est recommandée. On peut rapidement donner 3 catégories :

Classification, régression, prédiction Il s'agit de trouver une classe ou une valeur selon un ensemble de descriptions. Ce sont des outils très utilisés. Les algorithmes reposent sur des arbres de décision, des réseaux de neurones, la règle de Bayes, les k plus proches voisins.

Association, sequencing Il s'agit de trouver des similarités ou des associations. Le sequencing est le terme anglais utilisé pour préciser que l'association se fera dans le temps. Par exemple, *si j'achète un couffin aujourd'hui, j'ai trois fois plus de chance dans 3 mois d'acheter un lit bébé* (sequencing) ou encore *Si j'achète des pâtes et de la purée de tomates, j'ai deux fois plus de chance d'acheter aussi du parmesan* (association).

Segmentation La problématique est de trouver des groupes homogènes dans une population. On utilise souvent les algorithmes des k-moyennes ou de Kohonen. La difficulté essentielle dans ce type de construction est la validation qui nécessite l'intervention d'experts humains.

2.6 Validation

Les méthodes de validation vont déprendre de la nature de la tâche et du problème considéré. Nous distinguerons deux modes de validation : statistique et par expertise.

Pour certains domaines d'application (le diagnostic médical, par exemple), il est essentiel que le modèle produit soit compréhensible. Il y a donc une première validation du modèle produit par l'expert, celle-ci peut être complétée par une validation statistique sur des bases de cas existantes.

Pour les problèmes d'apprentissage non supervisé (segmentation, association), la validation est essentiellement du ressort de l'expert. Pour la segmentation, le programme construit des groupes homogènes, un expert peut juger de la pertinence des groupes constitués. La encore, on peut combiner avec une validation statistique sur un problème précis utilisant cette segmentation. Pour la recherche des règles d'association, c'est l'expert du domaine qui jugera de la pertinence des règles, en effet, l'algorithme, s'il fournit des règles porteuses d'information, produit également des règles triviales et sans intérêt.

Pour la validation statistique, la première tâche à réaliser consiste à utiliser les méthodes de base de statistique descriptive. L'objectif est d'obtenir des informations qui permettront de juger le résultat obtenu, ou d'estimer la qualité ou les biais des données d'apprentissage.

1. Calculer les moyennes et variances des attributs.
2. Si possible, calculer la corrélation entre certains champs.
3. Déterminer la classe majoritaire dans le cas de la classification.

Pour la classification supervisée, la deuxième tâche consiste à décomposer les données en plusieurs ensembles disjoints. L'objectif est de garder des données pour estimer les erreurs des modèles ou de les comparer. Il est souvent recommandé de constituer:

- Un ensemble d'apprentissage.
- Un ensemble de test.
- Un ensemble de validation.

Au moins deux ensembles sont nécessaires : l'ensemble d'apprentissage permet de générer le modèle, l'ensemble test permet d'évaluer l'erreur réelle du modèle sur un ensemble indépendant évitant ainsi un biais d'apprentissage. Lorsqu'il s'agit de tester plusieurs modèles et de les comparer, on peut sélectionner le meilleur modèle selon ses performances sur l'ensemble test et ensuite évaluer son erreur réelle sur l'ensemble de validation.

Lorsqu'on ne dispose pas de suffisamment d'exemples on peut se permettre d'apprendre et d'estimer les erreurs avec un même ensemble par la technique de *validation croisée*. . La validation croisée ne construit pas de modèle utilisable mais ne sert qu'à estimer l'erreur réelle d'un modèle selon l'algorithme suivant :

Validation croisée (S, x)
// S est un ensemble x est un entier
Découper S en x parties égales $\{S_1, \ldots, S_x\}$
Pour i de 1 à x
  Construire un modèle M avec l'ensemble $S - S_i$
  Evaluer l'erreur e_i de M avec S_i
Fin Pour
Retourner la moyenne des $e_i = \frac{\sum_{i=1}^{x} e_i}{x}$

Chapitre 3

Fouille de données

On se situe dans un environnement d'aide à la décision à partir de données. On suppose que de grandes quantités de données sont disponibles. En règle générale, ces données sont structurées et correspondent aux enregistrements d'une table ou de plusieurs tables d'une base dédiée à la décision (Infocentre ou entrepôt de données). Nous allons présenter, dans ce chapitre, les tâches, i.e. les problèmes que l'on cherche à résoudre et quelques-unes des méthodes disponibles pour résoudre ces tâches.

3.1 Généralités

Les tâches

On dispose de données structurées. Les objets sont représentés par des enregistrements (ou descriptions) qui sont constitués d'un ensemble de champs (ou attributs) prenant leurs valeurs dans un domaine. On peut mettre en évidence différentes problématiques. Les termes employés pouvant varier d'une discipline à l'autre (parfois même dans une même discipline selon le domaine d'application), nous définissons notre vocabulaire avec la description associée de la tâche.

La classification : consiste à examiner les caractéristiques d'un objet et lui attribuer une classe, la classe est un champ particulier à valeurs discrètes. Des exemples de tâche de classification sont :
- attribuer ou non un prêt à un client,
- établir un diagnostic,
- accepter ou refuser un retrait dans un distributeur,
- attribuer un sujet principal à un article de presse, ...

L'estimation : consiste à estimer la valeur d'un champ à partir des caractéristiques d'un objet. Le champ à estimer est un champ à valeurs continues. L'estimation peut être utilisée dans un but de classification. Il suffit d'attribuer une classe particulière pour un intervalle de valeurs du champ estimé. Des exemples de tâche d'estimation sont :
- noter un candidat à un prêt ; cette estimation peut être utilisée pour attribuer

un prêt (classification), par exemple, en fixant un seuil d'attribution,
- estimer les revenus d'un client.

La prédiction : consiste à estimer une valeur future. En général, les valeurs connues sont historisées. On cherche à prédire la valeur future d'un champ. Cette tâche est proche des précédentes. Les méthodes de classification et d'estimation peuvent être utilisées en prédiction. Des exemples de tâche de prédiction sont :
- prédire les valeurs futures d'actions,
- prédire au vu de leurs actions passées les départs de clients.

Les règles d'association (analyse du panier de la ménagère) : consiste à déterminer les valeurs qui sont associées. L'exemple type est la détermination des articles (le poisson et le vin blanc ; la baguette et le camembert et le vin rouge, ...) qui se retrouvent ensemble sur un même ticket de supermarché. Cette tâche peut être effectuée pour identifier des opportunités de vente croisée et concevoir des groupements attractifs de produit. C'est une des tâches qui nécessite de très grands jeux de données pour être effective. Cette tâche a engendré l'exemple (l'anecdote) suivant présent dans de nombreux articles sur le data mining : dans les supermarchés américains, il a été possible de mettre en évidence des corrélations entre achat de bières et achat de couches-culottes avant le week-end ! remarque justifiée par le comportement des jeunes pères américains qui préparent leur week-end en préparant leur provision de bière pour regarder la télévision et qui font les achats pour bébé au même moment. Gag ou réalité, je n'ai jamais lu de résultats sur les actions consistant à mettre, dans les supermarchés, les bières à côté des couches-culottes !!

La segmentation : consiste à former des groupes (clusters) homogènes à l'intérieur d'une population. Pour cette tâche, il n'y a pas de classe à expliquer ou de valeur à prédire définie a priori, il s'agit de créer des groupes homogènes dans la population (l'ensemble des enregistrements). Il appartient ensuite à un expert du domaine de déterminer l'intérêt et la signification des groupes ainsi constitués. Cette tâche est souvent effectuée avant les précédentes pour construire des groupes sur lesquels on applique des tâches de classification ou d'estimation.

Les données

Ce sont les valeurs des champs des enregistrements des tables de l'entrepôt. Ces données possèdent un type qu'il est important de préciser. En effet, la plupart des méthodes sont sensibles aux données manipulées. Par exemple, certaines méthodes sont mises en défaut par les données continues alors que d'autres peuvent être sensibles à la présence de données discrètes.

Les données discrètes

- les données binaires ou logiques : 0 ou 1 ; oui ou non ; vrai ou faux. ce sont des données telles que le sexe, être bon client, ...

  - les données énumératives : ce sont des données discrètes pour lesquelles il n'existe pas d'ordre défini a priori. Par exemple : la catégorie socioprofessionnelle, la couleur, ...
  - les données énumératives ordonnées : les réponses à une enquête d'opinion (1: très satisfait ; 2 : satisfait ; ...), les données issues de la discrétisation de données continues (1 : solde moyen < 2000 ; 2 : $2000 \leq$ solde moyen < 5000 ; ...)

Les données continues ce sont les données entières ou réelles : l'âge, le revenu moyen, ... mais aussi les données pouvant prendre un grand nombre de valeurs ordonnées.

Les dates sont souvent problématiques car mémorisées selon des formats différents selon les systèmes et les logiciels. Pour les applications en fouille de données, il est fréquent de les transformer en données continues ou en données énumératives ordonnées. On transforme une date de naissance en âge entier ou en une variable énumérative ordonnée correspondant à des tranches d'âge.

Les données textuelles ne sont pas considérées dans ce cours. Cependant, un texte peut, pour certaines applications, être résumé comme un n-uplet constitué du nombre d'occurrences dans le texte de mots clés d'un dictionnaire prédéfini.

Les méthodes

Nous ne présentons que certaines méthodes qui viennent compléter les outils classiques que sont : les requêtes SQL, les requêtes analyse croisée, les outils de visualisation, la statistique descriptive et l'analyse des données. Les méthodes choisies qui seront détaillées dans les sections suivantes sont :

- un algorithme pour la segmentation,
- les règles d'association,
- les plus proches voisins (raisonnement à partir de cas),
- les arbres de décision,
- les réseaux de neurones,
- les algorithmes génétiques.

La présentation ne prétend pas à l'exhaustivité. Les choix effectués sont subjectifs, quoique guidés par la disponibilité des méthodes dans les environnements de data mining standard. Des méthodes importantes ne sont pas étudiées dans ce cours. Citons les réseaux bayésiens, la programmation logique inductive et les machines à vecteurs de support (SVM for Support Vector Machine). De plus, pour les méthodes proposées dans ce cours, il existe de nombreuses variantes, non présentées, qui peuvent s'appliquer à des tâches différentes. Par exemple, nous présentons, dans la section réseaux de neurones, le perceptron et le perceptron multi couches et leurs applications en classification et estimation. Cependant, il existe d'autres modèles de réseaux de neurones bien adaptés pour la segmentation (réseaux de Kohonen par exemple) ou spécialisés pour la prédiction (réseaux récurrents). En tout état de cause, un fait important communément admis est :

Il n'existe pas de méthode supérieure à toutes les autres

Par conséquent, à tout jeu de données et tout problème correspond une ou plusieurs méthodes. Le choix se fera en fonction

- de la tâche à résoudre,
- de la nature et de la disponibilité des données,
- des connaissances et des compétences disponibles,
- de la finalité du modèle construit. Pour cela, les critères suivants sont importants : complexité de la construction du modèle, complexité de son utilisation, ses performances, sa pérennité, et, plus généralement,
- de l'environnement de l'entreprise.

Nous nous attachons dans la présentation des différentes méthodes à préciser les particularités de chacune des méthodes selon ces axes.

3.2 Un algorithme pour la segmentation

Pour résoudre certains problèmes complexes, il peut s'avérer utile de commencer par segmenter la population (la diviser en groupes) en espérant que le problème soit alors plus simple à résoudre sur les groupes ainsi constitués. La segmentation est une tâche d'apprentissage « non supervisée » car on ne dispose d'aucune autre information préalable que la description des exemples. Après application de l'algorithme et donc lorsque les groupes ont été construits, d'autres techniques ou une expertise doivent dégager leur signification et leur éventuel intérêt.

Nous présentons ici la méthode des k-moyennes car elle est très simple à mettre en oeuvre et très utilisée. Elle comporte de nombreuses variantes et est souvent utilisée en combinaison avec d'autres algorithmes.

Introduction

La méthode est basée sur une notion de similarité entre enregistrements. Nous allons pour introduire l'algorithme considérer un espace géométrique muni d'une distance, deux points sont similaires si ils sont proches pour la distance considérée. Pour pouvoir visualiser le fonctionnement de l'algorithme, nous allons limiter le nombre de champs des enregistrements. Nous nous plaçons donc dans l'espace euclidien de dimension 2 et considérons la distance euclidienne classique. L'algorithme suppose choisi a priori un nombre k de groupes à constituer. On choisit alors k enregistrements, soit k points de l'espace appelés les centres. On constitue alors les k groupes initiaux en affectant chacun des enregistrements dans le groupe correspondant au centre le plus proche. Pour chaque groupe ainsi constitué, on calcule son nouveau centre en effectuant la moyenne des points du groupe et on réitère le procédé. Le critère d'arrêt est : d'une itération à la suivante, aucun point n'a changé de groupe, i.e. les groupes sont stables.

Considérons l'exemple présenté en Figure 3.1, les données d'entrée sont constituées de 8 points A, ..., H de l'espace euclidien de dimension 2. On choisit $k = 2$, c'est-à-dire on cherche à constituer deux groupes dans cet ensemble de 8 points. On tire aléatoirement les 2 centres initiaux, B et D sont choisis. On répartit les points dans les deux groupes (nommés B et D) en fonction de leur proximité aux centres B et D. Géométriquement, il suffit de tracer la médiatrice du segment [BD]. Seul D est affecté au groupe D. On calcule les nouveaux centres pour l'étape 2, on obtient D et I(26/7;17/7) où les coordonnées de I sont les moyennes des coordonnées des 7 points autres que D. On recrée les groupes en fonction de leur proximité aux centres D et I, on remarque que A et C changent de groupe. Le procédé est réitéré, la stabilité est obtenue à l'étape 4 car on constate que les groupes ne sont plus modifiés. Les deux groupes obtenus sont {A, B, C, D} et {E, F, G, H}.

	étape 1 centre D(2;4) centre B(2;2)	étape 2 centre D(2;4) centre I(27/7;17/7)	étape 3 centre J(5/3;10/3) centre K(24/5;11/5)
points	groupe	groupe	groupe
A(1;3)	B	D	J
B(2;2)	B	I	J
C(2;3)	B	D	J
D(2;4)	D	D	J
E(4;2)	B	I	K
F(5;2)	B	I	K
G(6;2)	B	I	K
H(7;3)	B	I	K

FIG. 3.1 – *exemple pour l'algorithme des k-moyennes*

Description de l'algorithme

On travaille avec des enregistrements qui sont des n-uplets de valeurs. On suppose définie une notion de similarité qui permet de comparer les distances aux centres, un calcul de moyenne pour le calcul des nouveaux centres. L'algorithme est paramétré par le nombre k de groupes que l'on souhaite constituer.

Algorithme des k-moyennes
paramètre : le nombre k de groupes
entrée : un échantillon de m enregistrements $\vec{x_1}, \ldots \vec{x_m}$

1. choisir k centres initiaux $\vec{c_1}, \ldots \vec{c_k}$
2. pour chacun des m enregistrements, l'affecter au groupe i dont le centre $\vec{c_i}$ est le plus proche
3. si aucun élément ne change de groupe alors arrêt et sortir les groupes
4. calculer les nouveaux centres : pour tout i, $\vec{c_i}$ est la moyenne des éléments du groupe i
5. aller en 2

Nous avons pu constater sur l'exemple simple présenté dans la Figure 3.1 que lorsque les points sont placés dans un espace euclidien, l'algorithme est très simple et très naturel. Où est le problème ? Les problèmes sont en fait multiples.

Un premier problème est que les bases de données commerciales sont constituées d'enregistrements qui ne sont pas des points d'un espace géométrique. En effet, les données sont des clients, des produits, des ventes qui ont peu de rapport avec des points d'un diagramme. L'algorithme est basé sur la notion de proximité. Il nous faut donc préciser comment définir cette notion pour que des points proches correspondent à des enregistrements similaires dans la base de données.

Pour les données continues, une mesure de similarité est de prendre la distance euclidienne entre deux points. Il faut être attentif à la normalisation des échelles pour les différents champs. En effet, si un champ A prend ses valeurs dans l'intervalle réel [0,1] et un champ B ses valeurs dans [0,100 000], le second est privilégié, en effet deux enregistrement seront considérés comme proches si les valeurs pour le champ B sont proches.

Lorsque les enregistrements contiennent essentiellement des champs à valeurs discrètes, on abandonne les mesures géométriques pour considérer des mesures basées sur le niveau de chevauchement. Une mesure usuelle est : étant donné deux enregistrements $\vec{x}$ et $\vec{y}$, la similarité est égale au rapport entre le nombre de champs égaux divisé par le nombre total de champs. Cette mesure possède de nombreuses variantes. Il est possible de définir des mesures qui mixent les calculs sur champs à valeurs réelles et champs à valeurs discrètes. Nous revenons plus en détail sur les problèmes de définitions de distances dans la présentation de la méthode des plus proches voisins en section 3.4.

Un deuxième problème est le choix initial du nombre de groupes. Ce nombre est choisi a priori et fourni à l'algorithme. Il peut avoir été fixé par un expert mais, en règle générale, il est inconnu. Dans ce cas, on fait fonctionner l'algorithme avec différentes valeurs de k (ce peut être une boucle externe ajoutée à l'algorithme), on choisit ensuite une valeur de k telle que, pour les groupes obtenus, les distances à l'intérieur du groupe soient petites et les distances entre centres des groupes soient grandes.

L'algorithme possède de nombreuses variantes selon : la méthode utilisée pour choisir les k premiers centres, la mesure de similarité choisie, le choix de la valeur de k et le calcul des distances entre groupes, la possibilité de pondérer les champs en fonction d'une connaissance initiale du problème.

Autres méthodes pour la segmentation

Une première méthode est d'utiliser un algorithme d'agglomération (méthode ascendante). À partir d'un échantillon de m enregistrements, on commence avec m groupes, chacun des groupes est constitué d'un enregistrement. On calcule la matrice des distances deux à deux entre enregistrements. On choisit la plus petite valeur dans cette matrice des similarités. Elle identifie la paire de groupes les plus similaires. On regroupe ces deux groupes en un seul, on remplace les deux lignes par une ligne pour le groupe ainsi créé. On réitère le procédé jusqu'à obtenir un groupe constitué de tous les enregistrements. On a ainsi construit une suite de partitions de l'échantillon d'entrée où le nombre k de groupes varie de m à 1. On choisit ensuite la valeur de k (ce choix est discuté plus loin dans cette section).

Une seconde méthode est basée sur les arbres de décision (voir plus loin dans ce chapitre, méthode descendante). On démarre avec un groupe constitué de l'ensemble des enregistrements et on divise récursivement les groupes en utilisant une fonction de diversité qui minimise la distance moyenne dans un groupe et maximise la distance entre groupes. On choisit ensuite la valeur de k.

Pour choisir la valeur de k dans ces différentes méthodes, à chaque étape, on calcule la distance intra groupe et la distance inter groupes. Les méthodes diffèrent par les distances utilisées et la façon de définir ces distances. La distance intra groupe peut être une moyenne des distances entre éléments du groupe. Pour calculer la distance inter groupes, il existe trois possibilités :

- le rattachement simple où la distance entre deux groupes est la plus petite distance entre les éléments les plus proches ;
- le rattachement complet où la distance entre deux groupes est donnée par la distance entre les membres les plus éloignés ;
- la distance entre les centres.

On choisit alors la valeur de k qui minimise la distance intra groupe et maximise la distance entre groupes.

Une dernière méthode est d'utiliser des réseaux de neurones bien adaptés pour la segmentation que sont les réseaux de neurones auto-organisés, en particulier les cartes de Kohonen.

Critiques de la méthode

Apprentissage non supervisé : la méthode des k-moyennes et ses variantes résolvent une tâche dite non supervisée, c'est-à-dire qu'elle ne nécessite aucune information sur les données. La segmentation peut être utile pour découvrir une structure cachée qui permettra d'améliorer les résultats de méthodes d'apprentissage supervisé (classification, estimation, prédiction).

Tout type de données : en choisissant une bonne notion de distance, la méthode peut s'appliquer à tout type de données (mêmes textuelles).

Facile à implanter : la méthode ne nécessite que peu de transformations sur les données (excepté les normalisations de valeurs numériques), il n'y pas de champ particulier à identifier, les algorithmes sont faciles à implanter et sont, en règle générale, disponibles dans les environnements de data mining.

Problème du choix de la distance : les performances de la méthode (la qualité des groupes constitués) sont dépendantes du choix d'une bonne mesure de similarité ce qui est une tâche délicate surtout lorsque les données sont de types différents.

Le choix des bons paramètres : la méthode est sensible au choix des bons paramètres, en particulier, le choix du nombre k de groupes à constituer. Un mauvais choix de

k produit de mauvais résultats. Ce choix peut être fait en combinant différentes méthodes, mais la complexité de l'algorithme augmente.

L'interprétation des résultats : il est difficile d'interpréter les résultats produits, i.e. d'attribuer une signification aux groupes constitués. Ceci est général pour les méthodes de segmentation.

3.3 Les règles d'association

Les règles d'association sont traditionnellement liées au secteur de la distribution car leur principale application est « l'analyse du panier de la ménagère » qui consiste en la recherche d'associations entre produits sur les tickets de caisse. Le but de la méthode est l'étude de ce que les clients achètent pour obtenir des informations sur qui sont les clients et pourquoi ils font certains achats. La méthode recherche *quels produits tendent à être achetés ensemble*. La méthode peut être appliquée à tout secteur d'activité pour lequel il est intéressant de rechercher des groupements potentiels de produits ou de services : services bancaires, services de télécommunications, par exemple. Elle peut être également utilisée dans le secteur médical pour la recherche de complications dues à des associations de médicaments ou à la recherche de fraudes en recherchant des associations inhabituelles.

Un attrait principal de la méthode est la clarté des résultats produits. En effet, le résultat de la méthode est un ensemble de *règles d'association*. Des exemples de règles d'association sont :

- si un client achète des plantes alors il achète du terreau,
- si un client achète du poisson et du citron alors il achète du vin blanc.
- si un client achète une télévision, il achètera un magnétoscope dans un an.

Ces règles sont intuitivement faciles à interpréter car elles montrent comment des produits ou des services se situent les uns par rapport aux autres. Ces règles sont particulièrement utiles en marketing. Les règles d'association produites par la méthode peuvent être facilement utilisées dans le système d'information de l'entreprise. Cependant, il faut noter que la méthode, si elle peut produire des règles intéressantes, peut aussi produire des règles triviales (déjà bien connues des intervenants du domaine) ou inutiles (provenant de particularités de l'ensemble d'apprentissage). La recherche de règles d'association est une méthode non supervisée car on ne dispose en entrée que de la description des achats.

Introduction de la méthode

On suppose avoir prédéfini une classification des articles (nous reviendrons sur ce point par la suite). Les données d'entrée sont constituées d'une liste d'achats. Un achat est lui-même constitué d'une liste d'articles. On peut remarquer que, contrairement aux enregistrements d'une table, les achats peuvent être de longueur variable. Pour introduire la méthode, nous considérons l'exemple suivant :

	produit A	produit B	produit C	produit D	produit E
achat 1	X			X	
achat 2	X	X	X		
achat 3	X				X
achat 4	X			X	X
achat 5		X		X	

FIG. 3.2 – *liste des achats*

À partir de ces données, si on recherche des associations entre deux produits, on construit le tableau de co-occurrence qui montre combien de fois deux produits ont été achetés ensemble :

	produit A	produit B	produit C	produit D	produit E
produit A	4	1	1	2	1
produit B	1	2	1	1	0
produit C	1	1	1	0	0
produit D	2	1	0	3	1
produit E	1	0	0	1	2

FIG. 3.3 – *tableau de co-occurrence*

Un tel tableau permet de déterminer avec quelle fréquence deux produits se rencontrent dans un même achat. Par exemple, le produit A apparaît dans 80% des achats, le produit C n'apparaît jamais en même temps que le produit E, les produits A et D apparaissent simultanément dans 40% des achats. Ces observations peuvent suggérer une règle de la forme : « si un client achète le produit A alors il achète le produit D ».

Il nous faut préciser comment extraire les règles et, pour cela, il nous faut quantifier leur pertinence. Considérons les règles

1. si A alors B,
2. si A alors D,
3. si D alors A.

A et B apparaissent dans 20% des achats, A et D apparaissent dans 40% des achats, la règle 1 a un support de 20% et les règles 2 et 3 ont un support de 40%. Considérons les règles 2 et 3, c'est-à-dire les produits A et D. D apparaît dans trois achats et, dans ces trois achats, A apparaît deux fois. Par contre, A apparaît quatre fois et, lorsqu'il apparaît, D n'apparaît que deux fois. On définit alors la confiance d'une règle, la règle 3 a une confiance de 67% et la règle 2 a une confiance de 50%. On préfère donc la règle 3 : si D alors A.

Ces idées se généralisent à toutes les combinaisons d'un nombre quelconque d'articles. Par exemple, pour trois articles, on cherche à générer des règles de la forme si X et Y alors Z. Nous allons maintenant formaliser ces remarques introductives.

Description de la méthode

On suppose avoir défini une liste d'articles. On dispose en entrée d'une liste d'achats.

Définitions

Une *règle d'association* est une règle de la forme : Si **condition** alors **résultat**. Dans la pratique, on se limite, en général, à des règles où la *condition est une conjonction d'apparition d'articles* et le *résultat est constitué d'un seul article*. Par exemple, une règle à trois articles sera de la forme : Si X et Y alors Z ; règle dont la sémantique peu être énoncée : Si les articles X et Y apparaissent simultanément dans un achat alors l'article Z apparaît.

Pour choisir une règle d'association, il nous faut définir les quantités numériques qui vont servir à valider l'intérêt d'une telle règle. Le *support* d'une règle est la fréquence d'apparition simultanée des articles qui apparaissent dans la condition et dans le résultat dans la liste d'achats donnée en entrée, soit

$$\text{support} = freq(\text{condition et résultat}) = \frac{d}{m} \tag{3.1}$$

où d est le nombre d'achats où les articles des parties condition et résultat apparaissent et m le nombre total d'achats. La *confiance* est le rapport entre le nombre d'achats où tous les articles figurant dans la règle apparaissent et le nombre d'achats où les articles de la partie condition apparaissent, soit

$$\text{confiance} = \frac{freq(\text{condition et résultat})}{freq(\text{condition})} = \frac{d}{c} \tag{3.2}$$

où c est le nombre d'achats où les articles de la partie condition apparaissent.

La confiance ne dépend que des articles qui apparaissent dans la règle. Des règles, dont le support est suffisant, ayant été choisies, la règle de confiance maximale sera alors privilégiée. Cependant, nous allons montrer sur l'exemple suivant que le support et la confiance ne sont pas toujours suffisants. Considérons trois articles A, B et C et leurs fréquences d'apparition :

article(s)	A	B	C	A et B	A et C	B et C	A et B et C
fréquence	45%	42,5%	40%	25%	20%	15%	5%

Si on considère les règles à trois articles, elles ont le même niveau de support 5%. Le niveau de confiance des trois règles est :

règle	confiance
si A et B alors C	0.20
si A et C alors B	0.25
si B et C alors A	0.33

La règle si B et C alors A possède la plus grande confiance. Une confiance de 0.33 signifie que si B et C apparaissent simultanément dans un achat alors A y apparaît aussi avec une probabilité estimée de 33%. Mais, si on regarde le tableau des fréquences d'apparition des articles, on constate que A apparaît dans 45% des achats. Il vaut donc mieux prédire A sans autre information que de prédire A lorsque B et C apparaissent. C'est pourquoi il est intéressant d'introduire l'*amélioration* qui permet de comparer le résultat de la prédiction en utilisant la règle avec la prédiction sans la règle. Elle est définie par :

$$\text{amélioration} = \frac{\text{confiance}}{freq(\text{résultat})} \tag{3.3}$$

Une règle est intéressante lorsque l'amélioration est supérieure à 1. Pour les règles choisies, on trouve :

règle	confiance	f(résultat)	amélioration
si A et B alors C	0.20	40%	0.50
si A et C alors B	0.25	42.5%	0.59
si B et C alors A	0.33	45%	0.74

Par contre, la règle si A alors B possède un support de 25%, une confiance de 0.55 et une amélioration de 1.31, cette règle est donc la meilleure. En règle générale, la meilleure règle est celle qui contient le moins d'articles.

Recherche des règles

Une liste de n articles étant définie, on considère une liste de m achats. On procède comme suit :

1. on calcule le nombre d'occurrences de chaque article,
2. on calcule le tableau des co-occurrences pour les paires d'articles,
3. on détermine les règles de niveau 2 en utilisant les valeurs de support, confiance et amélioration,
4. on calcule le tableau des co-occurrences pour les triplets d'articles,
5. on détermine les règles de niveau 3 en utilisant les valeurs de support, confiance et amélioration,
6. ...

La valeur du nombre m d'achats est, en règle générale, très grande. Il est fréquent d'avoir à traiter des problèmes où m est de l'ordre du million. Des parcours de cette liste sont nécessaires pour construire les tableaux de co-occurrences d'où la nécessité d'architectures qui permettent des accès rapides à de grands jeux de données. Pour le nombre n d'articles, cette valeur peut également être grande. Étudions les tailles des tableaux en fonction de n et du nombre d'articles apparaissant dans les règles:

	2	3	4
n	$\frac{n(n-1)}{2}$	$\frac{n(n-1)(n-2)}{6}$	$\frac{n(n-1)(n-2)(n-3)}{24}$
100	4950	161 700	3 921 225
10 000	$\approx 5 \times 10^7$	$\approx 1.7 \times 10^{11}$	$\approx 4.2 \times 10^{14}$

On constate que les calculs nécessaires à la construction des règles (tableaux, support, confiance, amélioration) dépassent très vite les capacités de calcul des machines (malgré l'évolution continuelle de ces capacités).

Une technique de réduction du nombre des articles et de leurs combinaisons pris en considération à chaque étape est appelée *élagage par support minimum*. On ne considère, à une étape donnée (par exemple recherche des règles à trois articles), que les règles faisant apparaître des articles pour lequel le support est supérieur à un taux fixé a priori. Par exemple, pour une liste d'achats de $m = 1\ 000\ 000$ d'articles, si le support minimum est fixé à 2%, à l'étape 2 on ne considère que les règles de la forme si X alors Y où X et Y apparaissent simultanément dans 20 000 achats. L'élagage par support minimum permet d'éliminer les articles qui sont trop peu fréquents pour générer des règles intéressantes. En faisant varier le support minimum selon l'étape, on peut trouver des combinaisons rares d'articles fréquents (si il diminue) ou des combinaisons fréquentes d'articles rares (si il augmente).

Pour certaines applications, il est possible de limiter le nombre de combinaisons en limitant la forme des règles recherchées. Par exemple, la conclusion de la règle est restreinte à un sous-ensemble de l'ensemble des articles comme les articles nouvellement vendus.

Une dernière façon de limiter les calculs est la création de groupes d'articles. Ceci nécessite de déterminer le bon niveau pour les articles, ce qui ne peut être fait qu'avec des experts du domaine demandeur de l'étude. Considérons des produits de supermarché. On peut aller d'une description d'un article qui va de « conserve » au code barre de l'article en passant par « conserve de légumes », « conserve de légumes de telle marque », « conserve de légumes de telle taille », « conserve de légumes de telle marque et telle taille ». Le choix du bon niveau de description est difficile et dépend du problème à résoudre.

Critiques de la méthode

Résultats clairs : les règles d'association sont faciles à interpréter. Elles sont faciles à utiliser pour des utilisations concrètes.

Apprentissage non supervisé : la méthode ne nécessite pas d'autre information qu'une classification en articles et la donnée d'une liste d'articles pour extraire les règles.

Achats de taille variable : la méthode est l'une des rares méthodes qui prend en entrée des achats qui sont des listes d'articles de taille variable. En effet, la plupart des autres méthodes travaillent sur la base d'enregistrements d'une table, ces enregistrements ayant une longueur fixe.

Introduction du temps : il est possible d'adapter la méthode pour traiter des séries temporelles pour générer des règles de la forme : « un client ayant acheté le produit A est susceptible d'acheter le produit B dans deux ans » ; il est possible d'introduire des « articles virtuels » tels que le jour, la période ou la saison et limiter la forme des règles si on souhaite rechercher des comportements d'achat qui dépendent du temps.

Simplicité de la méthode : la méthode et les calculs sont élémentaires (on ne calcule que des fréquences d'apparition). La méthode peut être programmée aisément sur tableur (pour des tailles de problème raisonnables) et est disponible dans la plupart des environnements de fouille de données.

Coût de la méthode : par contre, la méthode est coûteuse en temps de calcul. Le regroupement d'articles et la méthode du support minimum permettent de diminuer les calculs mais on peut alors éliminer malencontreusement des règles importantes.

Le choix des articles : il est difficile de déterminer le bon niveau d'articles. Les traitements préalables sur les achats peuvent être complexes : par exemple, il peut être difficile de retrouver l'article à partir de son code-barre enregistré sur le ticket de caisse.

Les articles rares : la méthode est plus efficace pour les articles fréquents. Pour les articles rares, on peut restreindre la forme des règles choisies ou faire varier le support minimum.

La qualité des règles : la méthode peut produire des règles triviales ou inutiles. Les règles triviales sont des règles évidentes (si camembert alors vin rouge) qui, par conséquent, n'apportent pas d'information. Les règles inutiles sont des règles difficiles à interpréter qui peuvent provenir de particularités propres à la liste des achats ayant servie à l'apprentissage.

3.4 Les plus proches voisins

La méthode des plus proches voisins (PPV en bref, nearest neighbor en anglais ou kNN for short) est une méthode dédiée à la classification qui peut être étendue à des tâches d'estimation. La méthode PPV est une méthode de raisonnement à partir de cas. Elle part de l'idée de prendre des décisions en recherchant un ou des cas similaires déjà résolus en mémoire. Contrairement aux autres méthodes de classification qui seront étudiées dans les sections suivantes (arbres de décision, réseaux de neurones, algorithmes génétiques), il n'y a pas d'étape d'apprentissage consistant en la construction d'un modèle à partir d'un échantillon d'apprentissage. C'est l'échantillon d'apprentissage, associé à une fonction de distance et d'une fonction de choix de la classe en fonction des classes des voisins les plus proches, qui constitue le modèle. L'algorithme générique de classification d'un nouvel exemple par la méthode PPV est :

Algorithme de classification par k-PPV
paramètre : le nombre k de voisins
donnée : un échantillon de m enregistrements classés $(\vec{x}, c(\vec{x}))$
entrée : un enregistrement $\vec{y}$
1. déterminer les k plus proches enregistrements de $\vec{y}$
2. combiner les classes de ces k exemples en une classe c

sortie : la classe de $\vec{y}$ est $c(\vec{y}) = c$

Nous allons, par conséquent, présenter les différents choix possibles pour la définition de la distance et pour le mode de sélection de la classe du cas présenté.

Définition de la distance

Le choix de la distance est primordial au bon fonctionnement de la méthode. Quoique les distances les plus simples permettent d'obtenir des résultats satisfaisants (lorsque c'est possible), il faut être attentif à différents pièges. Tout d'abord, rappelons qu'une distance doit satisfaire les propriétés suivantes :

$$d(A,A) = 0$$
$$d(A,B) = d(B,A)$$
$$d(A,B) \leq d(A,C) + d(B,C)$$

pour tous points A, B et C de l'espace. Dans notre cadre, les points sont des enregistrements d'une base de données. D'après la première de ces propriétés, le plus proche voisin d'un enregistrement est lui-même. Par la suite, nous désignons par plus proche voisin un point le plus proche de l'enregistrement autre que lui-même. Les deux propriétés suivantes rendent la notion de proximité stable. On peut cependant noter qu'un point A peut avoir un plus proche voisin B tandis que B possède de nombreux voisins plus proches que A (voir figure 3.4).

FIG. 3.4 – *A a un plus proche voisin B, B a de nombreux voisins proches autres que A*

Pour définir la fonction de distance, on définit d'abord une distance sur chacun des champs, puis on combine ces distances pour définir la distance globale entre enregistrements. Commençons par étudier les choix possibles selon le type du champ.

champs numériques : la distance entre deux valeurs numériques x et y peut être choisie égale à :

- $d(x,y) = |x-y|$, ou
- $d(x,y) = \frac{|x-y|}{d_{max}}$ où d_{max} est la distance maximale entre deux réels du domaine considéré.

Lorsque nous allons combiner les différentes distances sur les champs, il faut éviter que les valeurs de distance aient des ordres de grandeurs trop différents. La seconde des deux distances normalise les distances dans l'intervalle réel [0,1].

champs discrets : la définition dépend alors des valeurs possibles.

- les données binaires : 0 ou 1. On choisit $d(0,0) = d(1,1) = 0$ et $d(0,1) = d(1,0) = 1$.
- les données énumératives : la distance vaut 0 si les valeurs sont égales et 1 sinon.
- les données énumératives ordonnées : elles peuvent être considérées comme des valeurs énumératives mais on peut également définir une distance utilisant la

relation d'ordre. Par exemple, si un champ prend les valeurs 1, 2, 3, 4 et 5, on peut définir la distance en considérant 5 points de l'intervalle [0,1] avec une distance de 0,2 entre deux points successifs, on a alors $d(1,2) = 0,2$; $d(1,3) = 0,4$; ... ; $d(4,5) = 0,2$.

autres champs une distance peut être définie sur de nombreux types de champs comme des textes, des images, des informations géographiques.

Il reste à combiner les distances entre champs pour définir une distance entre enregistrements. Soit $\vec{x} = (x_1, \ldots, x_n)$ et $\vec{y} = (y_1, \ldots, y_n)$ deux enregistrements, d_1, ..., d_n sont les distances définies sur les différents champs, la distance entre deux enregistrements peut être définie par :

– la distance euclidienne : $d_e(\vec{x}, \vec{y}) = \sqrt{d_1(x_1, y_1)^2 + \ldots + d_n(x_n, y_n)^2}$,
– la sommation : $d_s(\vec{x}, \vec{y}) = d_1(x_1, y_1) + \ldots + d_n(x_n, y_n)$.

Considérons trois enregistrements $\vec{x} = (30, 1, 1000)$, $\vec{y} = (40, 0, 2200)$ et $\vec{z} = (45, 1, 4000)$ où le premier champ est l'âge, le second indique si le client est propriétaire de sa résidence principale, le troisième est un montant des mensualités des crédits en cours. Les champs n'ayant pas été normalisés, on choisit sur chacun des champs la distance normalisée. On combine les distances par les deux méthodes, on obtient alors :

– $d_e(\vec{x}, \vec{y}) = \sqrt{\frac{10}{15}^2 + 1^2 + \frac{1200}{3000}^2} \approx 1.27$,
– $d_e(\vec{x}, \vec{z}) = \sqrt{\frac{15}{15}^2 + 0^2 + \frac{3000}{3000}^2} \approx 1.41$,
– $d_e(\vec{y}, \vec{z}) = \sqrt{\frac{5}{15}^2 + 1^2 + \frac{1800}{3000}^2} \approx 1.21$,
– $d_s(\vec{x}, \vec{y}) = \frac{10}{15} + 1 + \frac{1200}{3000} \approx 2.07$,
– $d_s(\vec{x}, \vec{z}) = \frac{15}{15} + 0 + \frac{3000}{3000} = 2$,
– $d_s(\vec{y}, \vec{z}) = \frac{5}{15} + 1 + \frac{1800}{3000} \approx 1.83$.

On peut constater que, selon la distance choisie, le plus proche voisin de $\vec{x}$ est $\vec{y}$ ou $\vec{z}$. La distance euclidienne favorise les voisins dont tous les champs sont assez voisins, la distance par sommation permet de tolérer une distance importante sur l'un des champs.

Nous avons présenté différents choix possibles, selon le type des données, pour définir une distance sur un champ. Des variantes de ces définitions existent et, parfois, une distance spécifique peut être définie en fonction d'une expertise. De même, il existe de nombreuses variantes pour combiner les distances entre champs. Il est également possible de pondérer l'importance des différents champs.

Sélection de la classe

L'idée de la méthode est la recherche de cas similaires au cas à résoudre et d'utiliser les décisions des cas proches déjà résolus pour choisir une décision. La méthode la plus simple est de rechercher le cas le plus proche et de prendre la même décision. C'est la méthode 1-PPV (1-NN) du plus proche voisin. Si cette méthode peut fournir de bons résultats sur des problèmes simples pour lesquels les points (les enregistrements) sont bien répartis en groupes denses d'enregistrements de même classe, en règle générale, il faut considérer un nombre de voisins plus important pour obtenir de bons résultats.

Nous supposons avoir déterminé les k voisins $(\vec{x_1}, c(\vec{x_1}))$, ..., $(\vec{x_k}, c(\vec{x_k}))$ d'un enregistrement $\vec{y}$ auquel on souhaite attribuer une classe $c(\vec{y})$.

Une première façon de combiner les k classes des k voisins les plus proches est le *vote majoritaire*. Elle consiste simplement à prendre la classe majoritaire. Dans le cas de deux classes, on choisit une valeur de k impaire.

Une seconde façon est le *vote majoritaire pondéré*. Chaque vote, c'est-à-dire chaque classe d'un des k voisins sélectionnés, est pondéré. Soit $\vec{x_i}$ le voisin considéré, Le poids de $c(\vec{x_i})$ est inversement proportionnel à la distance entre l'enregistrement $\vec{y}$ à classer et $\vec{x_i}$.

Dans les deux cas précédents, il est possible de définir une confiance dans la classe attribuée égale au rapport entre les votes gagnants et le total des votes.

Lorsque la technique est appliquée à une tâche d'estimation, donc à prédire la valeur d'un attribut continu, la notion de vote perd tout son sens. Une première solution pour combiner les réponses est l'interpolation, c'est-à-dire de calculer une moyenne pondérée des réponses. Un défaut de cette solution est de «lisser» les données. Une deuxième solution est de considérer les k enregistrements avec la valeur prédite correspondante et d'utiliser les techniques de régression linéaire pour estimer la valeur en $\vec{y}$.

Mise en place de la méthode

La méthode ne nécessite pas de phase d'apprentissage. Le modèle sera constitué de l'échantillon d'apprentissage, de la distance et de la méthode de combinaison des voisins.

Il faut choisir l'échantillon, c'est-à-dire les attributs pertinents pour la tâche de classification considérée et l'ensemble des enregistrements. Il faut veiller à disposer d'un nombre assez grand d'enregistrements par rapport au nombre d'attributs et à ce que chacune des classes soit bien représentée dans l'échantillon choisi.

Le choix de la distance par champ et du mode de combinaison des distances se fait en fonction du type des champs et des connaissances préalables du problème. Il est possible de choisir la distance en faisant varier cette distance et, pour chacun des choix, estimer l'erreur réelle. On choisit alors la distance donnant la meilleure erreur réelle estimée. L'estimation de l'erreur réelle se fait classiquement avec un ensemble test ou la validation croisée (voir section 2.6).

Le choix du nombre k de voisins peut, lui aussi, être déterminé par utilisation d'un ensemble test ou par validation croisée. Une heuristique fréquemment utilisée est de prendre k égal au nombre d'attributs plus 1. La méthode de vote ou d'estimation peut aussi être choisie par test ou validation croisée.

Critiques de la méthode

pas d'apprentissage : c'est l'échantillon qui constitue le modèle. L'introduction de nouvelles données permet d'améliorer la qualité de la méthode sans nécessiter la reconstruction d'un modèle. C'est une différence majeure avec des méthodes telles que les arbres de décision et les réseaux de neurones.

clarté des résultats : bien que la méthode ne produit pas de règle explicite, la classe attribuée à un exemple peut être expliquée en exhibant les plus proches voisins qui ont amené à ce choix.

tout type de données : la méthode peut s'appliquer dès qu'il est possible de définir une distance sur les champs. Or, il est possible de définir des distances sur des champs complexes tels que des informations géographiques, des textes, des images, du son. C'est parfois un critère de choix de la méthode PPV car les autres méthodes traitent difficilement les données complexes. On peut noter, également, que la méthode est robuste au bruit.

nombre d'attributs : la méthode permet de traiter des problèmes avec un grand nombre d'attributs. Mais, plus le nombre d'attributs est important, plus le nombre d'exemples doit être grand. En effet, pour que la notion de proximité soit pertinente, il faut que les exemples couvrent bien l'espace et soient suffisamment proches les uns des autres. Si le nombre d'attributs pertinents est faible relativement au nombre total d'attributs, la méthode donnera de mauvais résultats car la proximité sur les attributs pertinents sera noyée par les distances sur les attributs non pertinents. Il est donc parfois utile de d'abord sélectionner les attributs pertinents.

temps de classification : si la méthode ne nécessite pas d'apprentissage, tous les calculs doivent être effectués lors de la classification. Ceci est la contrepartie à payer par rapport aux méthodes qui nécessite un apprentissage (éventuellement long) mais qui sont rapides en classification (le modèle est créé, il suffit de l'appliquer à l'exemple à classifier). Certaines méthodes permettent de diminuer la taille de l'échantillon en ne conservant que les exemples pertinents pour la méthode PPV, mais il faut, de toute façon, un nombre d'exemples suffisamment grand relativement au nombre d'attributs.

stocker le modèle : le modèle est l'échantillon, il faut donc un espace mémoire important pour le stocker ainsi que des méthodes d'accès rapides pour accélérer les calculs.

distance et nombre de voisins : les performances de la méthode dépendent du choix de la distance, du nombre de voisins et du mode de combinaison des réponses des voisins. En règle générale, les distances simples fonctionnent bien. Si les distances simples ne fonctionnent pour aucune valeur de k, il faut envisager le changement de distance, ou le changement de méthode !

3.5 Les arbres de décision

Nous étudions les algorithmes de génération d'arbres de décision à partir de données. Les deux algorithmes les plus connus et les plus utilisés (l'un ou l'autre ou les deux sont présents dans les environnements de fouille de données) sont CART (Classification And Regression Trees [BFOS84]) et C5 (version la plus récente après ID3 et C4.5 [Qui93]).

Ces algorithmes sont très utilisés car performants et car ils génèrent des procédures de classification exprimables sous forme de règles.

Qu'est-ce qu'un arbre de décision ?

Un arbre de décision est une représentation graphique d'une procédure de classification. Les nœuds internes de l'arbre sont des tests sur les champs ou attributs, les feuilles sont les classes. Lorsque les tests sont binaires, le fils gauche correspond à une réponse positive au test et le fils droit à une réponse négative. Un exemple d'arbre de décision est:

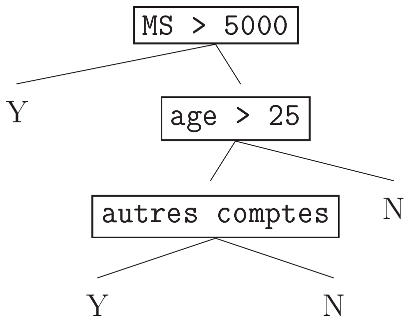

FIG. 3.5 – *exemple d'arbre de décision ; `MS` est la moyenne des soldes du compte courant, `autres comptes` est un champ binaire qui vaut oui si le client dispose d'autres comptes, la classe Y indique un a priori favorable pour l'attribution d'un prêt.*

Pour classer un enregistrement, il suffit de descendre dans l'arbre selon les réponses aux différents tests pour l'enregistrement considéré. Soit l'enregistrement $\vec{x}$ défini par : `nom=Digra`, `prénom=Omar`, `age=32`, `csp=2`, `MS=2550`, `prets en cours = oui`, `autres comptes = oui`. Cet enregistrement sera classé Y car `MS<=2550` et `age > 25` et `autres comptes = oui`. On peut déjà remarquer quelques propriétés importantes des arbres de décision :

- la procédure de classification associée est compréhensible par tout utilisateur,
- la classe associée à un enregistrement particulier peut être justifiée,
- les attributs apparaissant dans l'arbre sont les attributs pertinents pour le problème de classification considéré.

On peut également remarquer qu'un arbre de décision est un système de règles. Il est immédiat de transformer l'arbre de la Figure 3.5 en :

Si `MS > 5000`	alors	Y
Si `MS <= 5000` et `age > 25` et `autres comptes=oui`	alors	Y
Si `MS <= 5000` et `age > 25` et `autres comptes= non`	alors	N
Si `MS <= 5000` et `age <= 25`	alors	N

Les systèmes de règles construits sont particuliers. En effet, pour tout enregistrement une et une seule règle s'applique, c'est-à-dire que les règles sont exhaustives et mutuellement exclusives.

Apprentissage des arbres de décision

Avec en entrée, un échantillon de m enregistrements classés $(\vec{x}, c(\vec{x}))$, un algorithme d'apprentissage doit fournir en sortie un arbre de décision. La plupart des algorithmes procèdent de façon descendante, c'est-à-dire qu'ils choisissent la racine de l'arbre (en général un test) puis, récursivement, choisissent l'étiquette des fils. Pour simplifier la présentation, nous nous limitons, dans cette section à des problèmes où les attributs sont discrets et le nombre de classes égal à 2. L'algorithme générique peut s'écrire :

Algorithme d'apprentissage par arbres de décision
donnée : un échantillon S de m enregistrements classés $(\vec{x}, c(\vec{x}))$
initialisation : arbre vide ; nœud courant : racine ; échantillon courant : S
répéter
  décider si le nœud courant est terminal
  si le nœud courant est terminal **alors**
    étiqueter le nœud courant par une feuille
  sinon
    sélectionner un test et créer le sous-arbre
  finsi
  nœud courant : un nœud non encore étudié
  échantillon courant : échantillon atteignant le nœud courant
jusque production d'un arbre de décision
élaguer l'arbre de décision obtenu
sortie : arbre de décision élagué

Nous allons préciser les différents points de cet algorithme en mettant en évidence les particularités des différents algorithmes.

décider si le nœud courant est terminal : le nœud courant est terminal si

- il n'y a plus d'attributs disponibles, c'est-à-dire que sur le chemin menant de la racine au nœud courant tous les tests disponibles ont été utilisés ;
- tous les exemples de l'échantillon courant sont dans une même classe.

Les critères précédents sont indiscutables, les autres critères sont spécifiques aux différents algorithmes et sont souvent paramétrables. Des exemples de critères sont :

- la proportion d'exemples d'une classe est supérieure à un seuil prédéfini. Par exemple, on décide de l'arrêt si une des classes contient plus de 95% des exemples.
- C5 utilise le critère suivant : si il n'existe pas de test ayant au moins k éléments sur deux branches alors le nœud est terminal. L'objectif de ce critère est d'éviter une croissance trop grande de l'arbre par l'exploration de branches comprenant trop peu d'exemples. La valeur de k est, par défaut, égale à 2, elle peut être modifiée par l'utilisateur.

étiqueter le nœud courant par une feuille : On étiquette le nœud courant par la classe majoritaire. Par exemple, si le nœud courant est terminal et si il y a 5 exemples de classe 0 et 20 exemples de classe 1, on étiquette par 1. Cependant, pour certains problèmes, il se peut que les erreurs de classification d'une classe vers l'autre aient des conséquences différentes. C'est le cas, par exemple, pour un diagnostic médical pour lequel classer un individu malade comme sain ou classer un individu sain comme malade n'ont pas les mêmes conséquences. Dans ce cas, il est possible de définir des coûts de mauvaise classification et la classe choisie le sera en fonction des coûts attribués.

sélectionner un test : on suppose que le nœud courant n'est pas terminal. Soit S l'échantillon associé au nœud courant. Pour introduire les possibles critères de sélection du test, considérons l'exemple suivant : S contient 100 exemples, 60 de classe 0 et 40 de classe 1. Le nœud courant sera étiqueté par le couple (60,40). Supposons que deux tests soient disponibles, et que ces deux tests déterminent les répartitions suivantes :

$$(60,40) \quad \rightarrow \quad \overset{A}{(30,10)\ (30,5)\ (0,25)}$$

$$(60,40) \quad \rightarrow \quad \overset{B}{(40,20)\ (20,20)}$$

Pour choisir le test, on utilise des fonctions qui mesurent le « degré de mélange » des différentes classes. Pour les problèmes à deux classes, on peut utiliser une des fonctions suivantes :

- la fonction de Gini : $Gini(x) = 4x(1-x)$
- la fonction entropie : $Entropie(x) = -x \log x - (1-x)\log(1-x)$

où x désigne la proportion d'éléments dans l'une des deux classes. Ces deux fonctions sont à valeurs dans l'intervalle réel $[0, 1]$, prennent leur minimum pour $x = 0$ ou $x = 1$ (tous les exemples sont dans une même classe) et leur maximum lorsque $x = 1/2$ (les exemples sont également répartis entre les deux classes). Choisissons, par exemple, la fonction de Gini. Pour le nœud courant, $x = 60/100$ et $Gini(x) = 4 \times 60/100 \times 40/100 = 0.96$. Si on choisit le test A, pour le premier fils (le plus à gauche), $x = 3/4$ et $Gini(x) = 0.75$, pour le second fils $x = 6/7$ et $Gini(x) = 0.49$, pour le troisième fils, $Gini(x) = 0$. Pour comparer les trois tests, on estime le « degré de mélange espéré » en pondérant les degrés de mélange des fils par la proportion des exemples allant sur ce fils, on obtient:

- pour A : $\frac{40}{100} \times 0.75 + \frac{35}{100} \times 0.49 + \frac{25}{100} \times 0 = 0.47$
- pour B : $\frac{60}{100} \times 0.89 + \frac{40}{100} \times 1 = 0.93$

On choisit alors le test qui fournit de degré de mélange espéré minimum. Souvent, on introduit le Gain qui est égal au degré de mélange du nœud courant diminué du degré de mélange espéré par l'introduction du test, on choisit alors le test qui apporte le gain maximal.

élaguer l'arbre de décision obtenu : il est possible de poursuivre la croissance de l'arbre jusqu'à obtention d'un arbre d'erreur nulle (si c'est possible : si il n'existe pas d'exemples ayant la même description mais des classes différentes) ou d'un arbre d'erreur mesurée sur l'ensemble d'apprentissage la plus petite possible. Cependant, l'objectif d'une procédure de classification est de bien classer des exemples non encore rencontrés, on parle de pouvoir de généralisation. Si l'algorithme fournit en sortie un arbre très grand qui classe bien l'échantillon d'apprentissage, on se trouve confronté au problème de sur-spécialisation : on a appris « par coeur » l'ensemble d'apprentissage, mais on n'est pas capable de généraliser. L'objectif de la phase d'élagage est d'obtenir un arbre plus petit (on élague des branches, c'est-à-dire que l'on détruit des sous-arbres) dans le but d'obtenir un arbre ayant un meilleur pouvoir de généralisation (même si on fait augmenter l'erreur sur l'ensemble d'apprentissage).

Les algorithmes d'apprentissage

CART

CART a été défini dans les années 80 ([BFOS84]). Depuis, il a été intégré à de nombreux environnements de fouille de données sous de nombreuses variantes. Nous donnons ici ses principales particularités. A l'origine, l'algorithme ne considérait que des tests binaires. La fonction qui mesure le degré de mélange (et donc le gain) est la fonction de Gini (les versions diffusées proposent d'autres choix). Pour l'élagage, on effectue un parcours ascendant de l'arbre construit. Pour décider si un sous arbre peut être élagué, on compare l'erreur réelle estimée de l'arbre courant avec l'arbre élagué. L'estimation de l'erreur réelle est mesurée sur un ensemble test ou par validation croisée.

C5

C5 est la version la plus récente d'un algorithme ID3 développé par R. Quinlan [Qui93]. L'algorithme peut prendre en compte des attributs d'arité quelconque. La fonction qui mesure le degré de mélange (et donc le gain) est la fonction entropie. Cette fonction a tendance à privilégier les attributs possédant un grand nombre de valeurs. Pour éviter ce biais, une fonction gain d'information est également disponible. L'élagage est effectué avec l'ensemble d'apprentissage par une évaluation pessimiste de l'erreur. Bien que cette technique puisse sembler inadaptée, elle donne de bons résultats en pratique.

Extensions

Bien que les algorithmes précédents semblent spécialisés pour les attributs discrets, ils ont été adaptés pour considérer également les attributs continus. Par exemple, dans C5, soit A un attribut continu, pour sélectionner un test, l'algorithme fait participer à la compétition tous les tests de la forme $A > a$ où a est une valeur prise par l'attribut A dans l'ensemble d'apprentissage.

Les algorithmes peuvent traiter les valeurs manquantes (descriptions contenant des champs non renseignés) pour l'apprentissage, mais aussi pour la classification.

Pour les attributs discrets possédant plusieurs valeurs, il est possible de demander à l'algorithme de grouper les valeurs pour le choix des tests. Par exemple, si un attribut A prend ses valeurs dans l'ensemble $\{1, 2, 3, 4, 5\}$, l'arbre engendré pourra contenir des tests de la forme $A \in \{1, 3, 5\}$.

C5 propose également de générer un système de règles à partir de l'arbre de décision. Le système obtenu n'est pas une simple réécriture de l'arbre car des transformations et simplifications sont effectuées.

Critiques de la méthode

lisibilité du résultat : un arbre de décision est facile à interpréter et est la représentation graphique d'un ensemble de règles. Si la taille de l'arbre est importante, il est difficile d'appréhender l'arbre dans sa globalité. Cependant, les outils actuels permettent une navigation aisée dans l'arbre (parcourir une branche, développer un nœud, élaguer une branche) et, le plus important, est certainement de pouvoir expliquer comment est classé un exemple par l'arbre, ce qui peut être fait en montrant le chemin de la racine à la feuille pour l'exemple courant.

tout type de données : l'algorithme peut prendre en compte tous les types d'attributs et les valeurs manquantes. Il est robuste au bruit.

sélection des variables : l'arbre contient les attributs utiles pour la classification. L'algorithme peut donc être utilisé comme pré-traitement qui permet de sélectionner l'ensemble des variables pertinentes pour ensuite appliquer une autre méthode.

classification efficace : l'attribution d'une classe à un exemple à l'aide d'un arbre de décision est un processus très efficace (parcours d'un chemin dans un arbre).

outil disponible : les algorithmes de génération d'arbres de décision sont disponibles dans tous les environnements de fouille de données.

extensions et modifications : la méthode peut être adaptée pour résoudre des tâches d'estimation et de prédiction. Des améliorations des performances des algorithmes de base sont possibles grâce aux techniques de bagging et de boosting : on génère un ensemble d'arbres qui votent pour attribuer la classe.

sensible au nombre de classes : les performances tendent à se dégrader lorsque le nombre de classes devient trop important.

évolutivité dans le temps : l'algorithme n'est pas incrémental, c'est-à-dire, que si les données évoluent avec le temps, il est nécessaire de relancer une phase d'apprentissage sur l'échantillon complet (anciens exemples et nouveaux exemples).

3.6 Les réseaux de neurones

Les réseaux de neurones sont des outils très utilisés pour la classification, l'estimation, la prédiction et la segmentation. Ils sont issus de modèles biologiques, sont constitués d'unités élémentaires (les neurones) organisées selon une architecture. Nous nous limitons dans ce paragraphe aux réseaux de neurones dédiés aux tâches d'estimation et classification que sont les *Perceptrons multicouches (PMC)*. Ceux-ci obtiennent de bonnes performances, en particulier, pour la reconnaissance de formes et sont donc bien adaptés pour des problèmes comprenant des variables continues éventuellement bruitées. Le principal désavantage est qu'un réseau est défini par une architecture et un grand ensemble de paramètres réels (les coefficients synaptiques), le pouvoir explicatif est faible : on parle parfois de « boîte noire ».

Qu'est-ce qu'un réseau de neurones ?

Un neurone

Un neurone est une unité de calcul élémentaire dont le modèle est issu de certains principes de fonctionnement du neurone biologique. L'unité de calcul combine des entrées réelles $x_1, \ldots, x_n$ en une sortie réelle o. Les entrées n'ont pas toutes la même importance et à chaque entrée x_i est associée un poids (ou coefficient synaptique) w_i. L'unité calcule d'abord l'activité d'entrée. En règle générale, pour le neurone formel, l'activité en entrée est mesurée par la somme pondérée des entrées $\Sigma_i w_i x_i$.

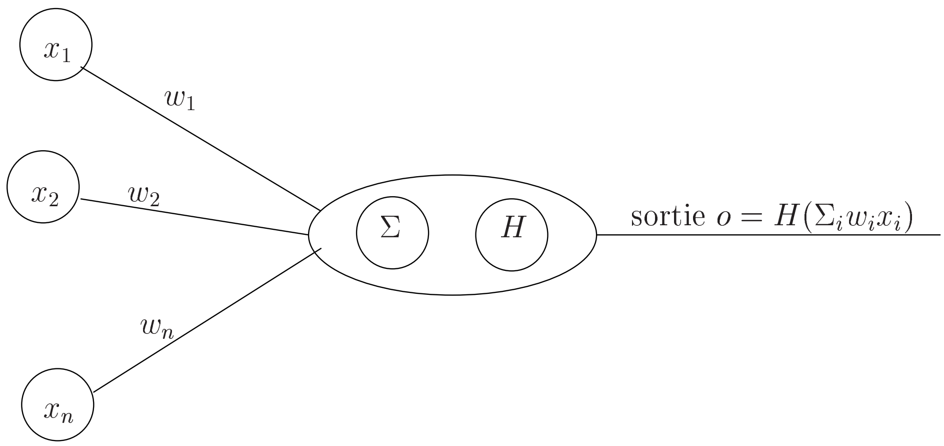

FIG. 3.6 – *Un neurone ou perceptron*

Pour le neurone biologique, lorsque l'activité en entrée dépasse un certain seuil, sa sortie est activée. Pour le neurone formel, une fonction de transfert H est appliquée à l'activité d'entrée. Cette fonction doit être telle que sa valeur de sortie soit 1 lorsque l'activité est suffisante et 0 sinon (parfois -1). Un premier choix possible est la fonction de Heaviside définie par $H(x) = 1$ si $x > 0$ et $H(x) = 0$ sinon. Cette fonction n'est pas dérivable et les algorithmes utilisant des méthodes de gradient, on utilise plus souvent des approximations dérivables de cette fonction. Parmi elles, la fonction la plus utilisée est la fonction sigmoïde définie par :

$$\sigma(x) = \frac{e^x}{e^x + 1} = \frac{1}{1 + e^{-x}}$$

Cette fonction (représentée dans la figure 3.7) prend ses valeurs dans l'intervalle $[0, 1]$, passe de 0 à 1 lorsque l'entrée est suffisante, tout en étant continue et dérivable. Il existe d'autres modèles de neurones selon la façon de cumuler les entrées, la fonction de transfert choisie. Nous nous limitons au modèle présenté : la sortie est à valeurs dans [0,1], la fonction de combinaison des entrées est la somme pondérée, la fonction de transfert est la fonction sigmoïde. Ces choix ayant été fixés, on voit qu'un neurone est entièrement déterminé par la donnée du vecteur $\vec{w} = (w_1, \dots, w_n)$ des n poids.

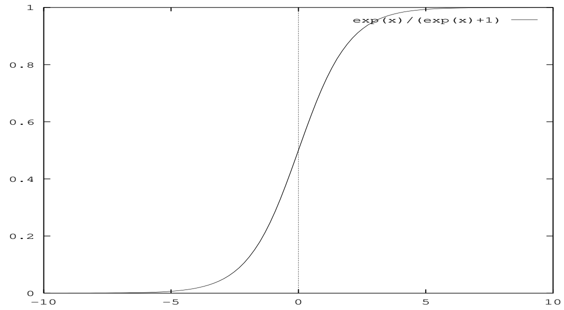

FIG. 3.7 – *La fonction sigmoïde*

Un Perceptron multicouches (PMC)

Le perceptron multicouches est un modèle de réseaux de neurones. Les neurones sont répartis en couches successives, les calculs sont effectués des entrées vers la ou les sorties, le neurone élémentaire est, en règle générale, celui considéré dans le paragraphe précédent.

La première couche est la *couche d'entrée* : aucun calcul n'est effectué, une cellule d'entrée ne fait que copier son entrée vers sa sortie. Les entrées correspondent aux attributs (ou leurs codages) du problème considéré. Ensuite, viennent une ou plusieurs *couches cachées* constituées de neurones élémentaires. Tous les neurones d'une couche sont les entrées de chaque neurone de la couche suivante et de elle seulement : autrement dit, il n'y a pas de retour arrière et on passe d'une couche à la suivante. Enfin, la dernière couche est la *couche de sortie* qui contient un ou plusieurs neurones. Les sorties de ces neurones correspondent aux valeurs à estimer (ou leurs codages) pour le problème considéré.

Algorithme d'apprentissage

Dans la très jeune histoire de l'informatique, il faut savoir que les réseaux de neurones sont anciens (première définition du neurone en 43 par McCulloch et Pitts), que les recherches ont été nombreuses et prometteuses. Cependant, il a été vite constaté que, pour

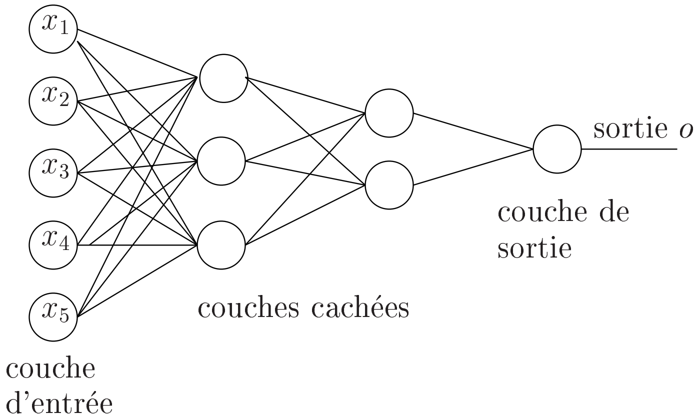

FIG. 3.8 – *Un perceptron multicouches : une couche d'entrée de 5 cellules, 2 couches cachées possédant respectivement 3 et 2 neurones, 1 couche de sortie à 1 neurone*

résoudre des problèmes complexes, le neurone élémentaire (éventuellement légèrement complexifié : perceptron de Rosenblatt 58) ne suffisait pas. Il fallait donc considérer des réseaux de neurones tels que le PMC défini auparavant. Le problème, qui a freiné les études sur les réseaux de neurones, est que l'on ne disposait pas d'algorithme d'apprentissage pour ces architectures. Il a fallu attendre le début des années 80 pour qu'un tel algorithme apparaisse : *l'algorithme de rétropropagation du gradient* défini par divers auteurs (Rumelhart, McClelland, Parker, Hinton, Le Cun).

L'étude de cet algorithme en tant que tel n'est pas l'objectif de ce cours. Nous nous contenterons donc d'en énoncer les principes de base et d'expliquer les paramètres à régler lors de l'utilisation d'un générateur de réseaux de neurones dans un environnement de fouille de données.

Nous considérons un problème d'estimation ou de classification. On dispose donc, en entrée, d'un échantillon S de m enregistrements classés $(\vec{x}, c(\vec{x}))$ où $\vec{x}$ est constitué de n entrées réelles et $c(\vec{x})$ est un réel de [0,1] ou un élément de $\{0, 1\}$. On suppose avoir fixé une architecture à n entrées et 1 sortie en choisissant le nombre de couches cachées et le nombre de neurones sur chacune des couches cachées (nous étudions le problème du choix de l'architecture dans la suite). On note $o(\vec{x})$ la sortie calculée par le réseau sur l'entrée $\vec{x}$. L'objectif de l'algorithme est de minimiser l'erreur apparente mesurée sur l'ensemble d'apprentissage. Cette erreur E est l'erreur quadratique mesurée sur S et peut s'écrire :

$$E = 1/2 \sum_{(\vec{x}, c(\vec{x})) \in S} (c(\vec{x}) - o(\vec{x}))^2 \tag{3.4}$$

Le principe de l'algorithme est le suivant :

– on initialise aléatoirement les poids de tous les neurones du réseau ;

- de façon itérative, on présente un élément $\vec{x}$ de S en entrée du réseau, le réseau calcule une sortie $o(\vec{x})$, la sortie attendue est $c(\vec{x})$, on peut mesurer l'écart entre sortie calculée et sortie attendue, on procède alors à une rétropropagation du gradient, c'est-à-dire qu'à l'aide d'une méthode mathématique (méthode du gradient), on modifie les poids (les coefficients synaptiques) de la couche de sortie vers la couche d'entrée pour minimiser l'erreur sur cet exemple ;
- un critère d'arrêt doit être défini.

Nous supposons toujours que l'architecture du réseau est fixée. Nous allons présenter les différents paramètres à régler pour l'apprentissage d'un PMC (ils sont résumés dans la figure 3.1). Tout d'abord, on choisit une *fonction d'activation*. Un fichier d'apprentissage étant défini, on initialise aléatoirement les poids. Il faut noter que des initialisations différentes peuvent produire des résultats différents.

Le paramètre *nombre d'exemples* définit le nombre d'exemples qui sont passés dans le réseau avant qu'un calcul d'erreur soit effectué et que les poids soient mis à jour. Les valeurs extrêmes sont 1 et m : le cardinal de l'ensemble d'apprentissage. Pour la valeur 1, il y a mise à jour des poids après chaque exemple. Dans ce cas, la convergence est rapide, mais on peut tomber dans un minimum local (on trouve un minimum de l'erreur mais ce n'est pas l'erreur minimale que l'on peut espérer). pour la valeur m, on passe l'échantillon complet, on calcule l'erreur $E(PMC)$ définie dans l'équation 3.4, puis on met à jour les poids. La convergence est plus lente.

Le paramètre *taux d'apprentissage* souvent nommé η ou ϵ est une valeur réelle de l'intervalle [0,1] qui influe sur la modification des poids. Une valeur grande signifie que l'on modifie fortement les poids à chaque mise à jour, une valeur petite entraine des modifications minimes de ces poids. Il est fréquent de choisir une valeur initiale qui diminue avec le nombre d'itérations. La valeur initiale est choisie « assez grande » (0.2 ou 0.1 par défaut) pour une convergence rapide, puis diminue pour éviter les oscillations et converger vers un minimum.

Le paramètre *inertie* souvent nommé α est une valeur réelle de l'intervalle [0,1] qui lisse les modifications de poids. Plus précisément, si on modifie les poids après chaque exemple, il se peut que certains poids soient alternativement augmentés ou diminués. Pour atténuer ce phénomène, on modifie les poids en tenant compte des modifications faites aux étapes précédentes.

La *tolérance* définit l'erreur cible, c'est-à-dire l'erreur que l'on souhaite atteindre avant de s'arrêter. Elle est choisie en fonction du mode de calcul de l'erreur, l'erreur $E(PMC)$ étant parfois normalisée. Des critères d'arrêt sur le nombre d'exemples de l'échantillon d'apprentissage S qui sont bien classés (ou bien estimés : erreur inférieure à un certain seuil) sont parfois ajoutés au précédent.

Mise en oeuvre

codage du problème: les PMC prennent des entrées réelles. Cependant, pour un bon fonctionnement des algorithmes, il est souvent nécessaire de normaliser les entrées

paramètre	parameter	objet du paramètre
fonction d'activation	activation function	fonction d'activation pour le neurone élémentaire, souvent la sigmoïde
nombre d'exemples	epoch length	nombre d'exemples avant mise à jour des poids, par défaut 1
taux d'apprentissage	learn rate	coefficient qui contrôle la vitesse de modification des poids
inertie	momentum	coefficient qui permet de lisser la variation des poids
tolérance	error tolerance ; target error	seuil d'erreur pour arrêt de l'apprentissage

TAB. 3.1 – *paramètres d'apprentissage pour la rétropropagation*

dans l'intervalle [0,1]. Cette normalisation peut être prise en charge par le générateur de réseaux de neurones ou est à la charge du programmeur. Les entrées binaires ne nécessitent pas de codage. Pour les données énumératives, plusieurs codages sont possibles. Prenons l'exemple d'un attribut A prenant ses valeurs dans l'ensemble $\{1, 2, 3, 4, 5\}$. Plusieurs codages sont envisageables :

- 5 entrées à valeurs binaires ; dans ce cas, la donnée $A = 3$ est codée par la troisième entrée vaut 1, les autres valent 0.
- on peut coder 5 valeurs sur 3 bits. L'attribut A peut alors être codé sur 3 entrées ; par exemple, la donnée $A = 3$ est codée par première entrée à 0, deuxième entrée à 1, troisième entrée à 1 si on choisit le codage binaire. D'autres codages sur 3 bits sont possibles si une notion d'ordre ou de proximité existe sur les valeurs de l'attribut A.
- 1 entrée réelle ; dans ce cas, on peut par exemple coder les valeurs 1, 2, 3, 4 et 5 par 0, 0.25, 0.5, 0.75 et 1.

Les PMC peuvent avoir une ou plusieurs sorties réelles. Pour un problème d'estimation d'une ou plusieurs variables, un neurone de sortie sert à estimer une des variables (normalisée). Pour un problème de classification à plusieurs classes, on peut créer une sortie pour chacune des classes ou coder en binaire (pour quatre classes, deux sorties suffisent). Remarquer que le codage « une sortie par classe » est intéressant si un même exemple peut appartenir à plusieurs classes.

choix de l'architecture : le codage étant choisi, on connaît le nombre d'entrées et le nombre de sorties du réseau. Il reste à choisir l'architecture. Un exemple d'architectures possibles est présenté dans la Figure 3.9. Plus la structure du réseau est riche, plus le pouvoir d'expression du modèle est grand. Sur l'exemple présenté, l'architecture 4-1 est moins puissante que l'architecture 4-2-1, elle-même moins puissante que l'architecture 4-4-1.

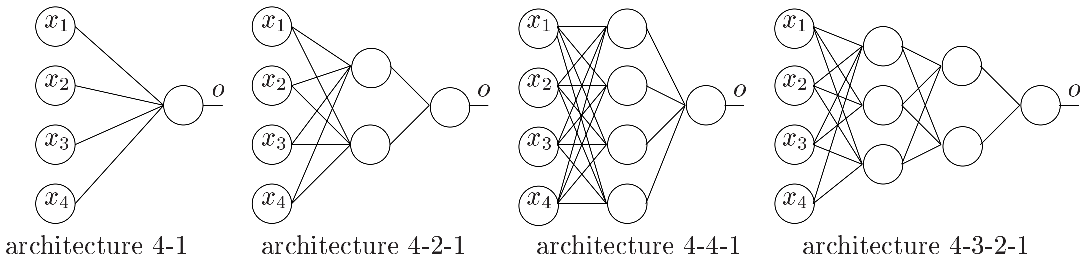

FIG. 3.9 – *différentes architectures possibles pour un problème à quatre entrées et une sortie*

Si on souhaite avoir un réseau de neurones ayant un bon pouvoir de généralisation, il faut choisir une architecture assez riche, mais pas trop ! En effet, on rencontre ici la même difficulté que pour les arbres de décision, il faut éviter la sur-spécialisation et si l'architecture est trop riche, on prend le risque d'un apprentissage « par coeur », sans pouvoir de généralisation. Le choix de l'architecture est fait, soit par l'expérience, soit par des essais successifs.

réglage des paramètres et choix de l'architecture : Nous venons de signaler l'importance du choix de l'architecture sur la qualité du réseau de neurones produit. Il faut également, une architecture étant choisie, régler les différents paramètres d'apprentissage pour générer un « bon » réseau de neurones. Pour cela, si les données disponibles sont en nombre suffisant, on découpe l'échantillon en trois ensembles : un ensemble d'apprentissage A, un ensemble de test T et un ensemble de validation V. Pour un choix d'architecture et un réglage des paramètres, on apprend avec A, on estime la qualité du réseau produit avec T. Ensuite, on choisit l'architecture et le réglage ayant obtenu les meilleures performances sur T. On relance l'apprentissage, l'estimation de l'erreur réelle est obtenue en calculant l'erreur sur V.

Critiques de la méthode

lisibilité du résultat : Le résultat de l'apprentissage est un réseau constitué de cellules organisées selon une architecture, définies par une fonction d'activation et un très grand nombre de poids à valeurs réelles. Ces poids sont difficilement interprétables. Pour un vecteur d'entrée, il est difficile d'expliquer le pourquoi de la sortie calculée.

les données réelles : les réseaux traitent facilement les données réelles – préalablement normalisées – et les algorithmes sont robustes au bruit. Ce sont, par conséquent, des outils bien adaptés pour le traitement de données complexes éventuellement bruitées comme la reconnaissance de formes (son, images sur une rétine, ...).

classification efficace : le réseau étant construit, le calcul d'une sortie à partir d'un vecteur d'entrée est un calcul très rapide.

outil disponible : les algorithmes de génération de réseaux de neurones sont disponibles dans tous les environnements de fouille de données.

paramètres d'apprentissage : il n'est pas facile, sans expérience approfondie, de choisir l'architecture et de régler les paramètres d'apprentissage. Il est possible de procéder par différents essais mais le point suivant nous montre que ce n'est pas toujours possible.

temps d'apprentissage : l'échantillon nécessaire à l'apprentissage doit être suffisamment grand et représentatif des sorties attendues. Il faut passer un grand nombre de fois tous les exemples de l'échantillon d'apprentissage avant de converger et donc le temps d'apprentissage peut être long.

évolutivité dans le temps : comme pour les arbres de décision, l'apprentissage n'est pas incrémental et, par conséquent, si les données évoluent avec le temps, il est nécessaire de relancer une phase d'apprentissage pour s'adapter à cette évolution.

en combinaison avec d'autres méthodes : pour des problèmes contenant un grand nombre d'attributs pour les entrées, il peut être très difficile de construire un réseau de neurones. On peut, dans ce cas, utiliser les arbres de décision pour sélectionner les variables pertinentes, puis générer un réseau de neurones en se restreignant à ces entrées.

extensions : les extensions sont nombreuses pour les tâches de classification et d'estimation : autres fonctions d'activation, algorithmes d'apprentissage, apprentissage « on line », ... Rappelons également que des modèles de réseaux de neurones existent pour les tâches de prédiction (réseaux récurrents : la sortie d'un neurone peut influer sur les neurones des couches précédentes) et pour les tâches de segmentation (réseaux associatifs, cartes de Kohonen).

Bibliographie

[AZ96] P. Adriaans and D. Zantinge. *Data Mining.* Addison-Wesley, 1996.

[BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and regression trees. Technical report, Wadsworth International, Monterey, CA, 1984.

[BL97] M. Berry and G. Linoff. *Data Mining : techniques appliquées au marketing, à la vente et aux services clients.* InterEditions, 1997.

[DG99] F. Denis and R. Gilleron. Apprentissage à partir d'exemples. Technical report, Grappa - Université de Lille 3, 1999.

[Kim97] R. Kimball. *Entrepôts de données.* 1997.

[Mit97] T. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[Qui93] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

Glossaire

Machine Parallèle .... Machine souvent utilisée pour de lourdes tâches (gros calculs scientifiques, serveur de nombreux utilisateurs,...). Elle est composée de plusieurs processeurs et intègre des procédures spécialisées permettant de les mettre à profit.

SGBD .... Système de Gestion de Bases de Données. C'est l'ensemble des programmes qui permettent de manipuler les données contenues dans une base de données.

SQL .... Structured Query Language. Langage de manipulation, définition de données. Ce langage a été plusieurs fois normalisé. Il est très répandu et largement utilisé dans la plupart des SGBD, surtout pour la communication entre clients et serveurs.

OLAP .... Online Analytical Processing.

Table de hachage .... Table qui permet un accès rapide aux éléments d'une structure de données (tableau, fichier). C'est une technique très utilisée pour l'implantation d'index dans les bases de données.

Tableau de bord .... Document de référence utilisé par les utilisateurs du système d'information de décision. Il indique des valeurs d'indices qui décrivent l'évolution de faits utiles au processus de décision. Par exemple, le bilan des ventes du mois, la progression du nombre de client...

Validation croisée .... Procédure d'estimation de l'erreur réelle d'un modèle qui permet de se passer d'un ensemble test. La validation croisée est souvent utilisée lorsqu'on ne dispose pas de suffisamment d'exemples pour valider un modèle.

Index