Big Data et Machine Learning

Manuel du data scientist

Pirmin Lemberger
Consultant senior et responsable de la veille technologique dans le groupe SQLI

Marc Batty
Co-fondateur de Dataiku

Médéric Morel
Fondateur et directeur général de Contexeo

Jean-Luc Raffaëlli
Directeur de projets stratégiques au sein de la DSI du Groupe La Poste

Préface de **Michel Delattre**
DSI du Groupe La Poste

Copyright © 2015 Dunod.

DUNOD

Toutes les marques citées dans cet ouvrage sont des marques déposées par leurs propriétaires respectifs.

Data Science Studio est une marque déposée de Dataiku.

Illustration de couverture : Toronto skyline, Canada
© Beboy, Fotolia.com

Le pictogramme qui figure ci-contre mérite une explication. Son objet est d'alerter le lecteur sur la menace que représente pour l'avenir de l'écrit, particulièrement dans le domaine de l'édition technique et universitaire, le développement massif du photocopillage.
Le Code de la propriété intellectuelle du 1er juillet 1992 interdit en effet expressément la photocopie à usage collectif sans autorisation des ayants droit. Or, cette pratique s'est généralisée dans les établissements d'enseignement supérieur, provoquant une baisse brutale des achats de livres et de revues, au point que la possibilité même pour les auteurs de créer des œuvres nouvelles et de les faire éditer correctement est aujourd'hui menacée. Nous rappelons donc que toute reproduction, partielle ou totale, de la présente publication est interdite sans autorisation de l'auteur, de son éditeur ou du Centre français d'exploitation du droit de copie (CFC, 20, rue des Grands-Augustins, 75006 Paris).



© Dunod, 2015
5 rue Laromiguière, 75005 Paris
www.dunod.com

ISBN 978-2-10-072392-8

Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5, 2° et 3° a), d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite » (art. L. 122-4).
Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

Copyright © 2015 Dunod.

Préface

Les raisons de l'émergence du Big Data sont bien connues. Elles sont d'abord économiques et technologiques. La chute exponentielle des coûts de l'espace de stockage et de la CPU intervenue au cours de ces vingt dernières années en est la cause première. Plus récemment, au cours des dix dernières années, les géants du Web tels que *Google*, *LinkedIn*, *Facebook* et *Yahoo!* ont été amenés à développer pour leurs propres besoins de nouvelles technologies : systèmes de stockage distribués à l'échelle du pétaoctet, traitements massivement parallèles et valorisation des données non structurées. Simultanément les méthodes mathématiques et statistiques et les algorithmes sophistiqués qui sont au cœur de l'analyse prédictive ont pris un essor considérable. Big Data et *machine learning* sont les deux faces de la révolution de la donnée.

Ces innovations ont par la suite essaimé dans le monde open source, très souvent avec le concours des mêmes acteurs du Web, les rendant ainsi disponibles à toutes les organisations. Cette démocratisation des technologies Big Data a par ailleurs largement bénéficié des services cloud, l'entreprise cliente n'ayant plus à prendre à sa charge la mise en place d'une infrastructure d'exécution et de stockage lourde et coûteuse.

Dans le sillage de cette effervescence technologique ont fleuri toute une série d'annonces vantant les mérites de telle ou telle solution supposée tout stocker puis tout analyser. Profondément enfouis dans les logs des serveurs web des sites e-commerce ou dans les commentaires laissés par des myriades d'internautes sur les réseaux sociaux, se tapiraient de colossaux gisements de données non structurées qui ne demanderaient qu'à être exploités.

Les grandes entreprises ont progressivement pris conscience du potentiel de création de valeur que leur apportaient ces nouvelles technologies du Big Data, appliquées en particulier à la masse croissante de données, structurées et non structurées, qu'elles peuvent mobiliser sur leurs clients. Mieux connaître ces clients, comprendre leurs comportements et leurs attentes, anticiper leurs réactions permet de les fidéliser et de leur faire les offres personnalisées les mieux adaptées. C'est un enjeu majeur pour toutes les entreprises.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Dans la révolution numérique en cours, la valorisation de l'information est donc au cœur de la stratégie des grands groupes.

Cette valorisation est d'autant plus complexe que l'entreprise est désormais étendue. Elle est immergée dans un écosystème informationnel qui dépasse de loin ses frontières traditionnelles et qui inclut ses clients mais aussi ses fournisseurs, ses partenaires et les territoires sur lesquels elle intervient. L'information traverse l'entreprise au rythme de la transformation numérique. Ces échanges se vivent de manière de plus en plus naturelle, sans pour autant que les flux de données soient pleinement valorisés. Extraire de manière pertinente la valeur de l'information repose en effet sur la capacité à capter le sens des données en les contextualisant de manière très ciblée.

Pourtant passer du « tout stocker » au « tout analyser » n'est pas toujours possible, d'autant plus que les écueils sur la route ne sont pas uniquement technologiques. Des mutations sont à prévoir impliquant des redistributions du pouvoir au sein des organisations et l'apparition de nouveaux métiers au premier rang desquels celui de data scientist. Dans cette nouvelle configuration, l'audace et la volonté d'innover en brisant les silos et en faisant travailler ensemble les différentes équipes de l'entreprise sont au cœur de la valorisation de la donnée. C'est le pari réussi des grands acteurs du Web et ce sera aussi celui des entreprises performantes de demain qui seront capables de se transformer pour innover.

Le présent ouvrage fournira au lecteur des éléments d'appréciation factuels qui lui permettront de se forger sa propre opinion. Quelles sont les données qui relèvent effectivement du Big Data ? Quelles sont les compétences à développer pour pleinement tirer parti de ces outils ? Quels seront les impacts du Big Data au niveau de l'organisation de l'entreprise ? Voilà autant de questions auxquelles les auteurs se proposent d'apporter des réponses argumentées.

La difficulté pour présenter un sujet aussi protéiforme que le Big Data, où il est question aussi bien de nouveaux usages, de nouvelles technologies que de nouveaux métiers est d'éviter le syndrome de l'inventaire à la Prévert. Dresser un catalogue des technologies actuelles est certes possible, au risque cependant d'être sanctionné par une obsolescence à brève échéance .... et au prix aussi d'un considérable ennui pour le lecteur.

Le parti pris des auteurs du présent ouvrage est de présenter les outils et les méthodes du Big Data sous un angle spécifique, celui de l'analyse prédictive. Comment peut-on exploiter des données massives pour construire des modèles prédictifs de comportements humains, de prix ou de phénomènes sociaux ? Ainsi, ce qui est perdu en exhaustivité est gagné en clarté et en structuration du propos.

Un accent particulier est mis sur la définition du métier de data scientist, dont les contours sont encore flous. Les chapitres de présentation concrète des outils de base du Big Data, comme les composantes de l'écosystème Hadoop, alternent avec des chapitres plus conceptuels sur des questions fondamentales. Pourquoi les systèmes relationnels ne suffisent-ils plus ? Selon quels principes faut-il représenter les données multidimensionnelles du Big Data ? Quel est le prix à payer pour automatiser la mise en parallèle de traitements au moyen de MapReduce ?

Copyright © 2015 Dunod.

Ce livre sera particulièrement utile à toute personne, ingénieur IT, architecte, DSI qui souhaite avoir une vue globale du Big Data dans le contexte de l'analyse prédictive. Aux profils plus techniques il fournira une bonne introduction, intuitive et sans formalisme, aux principaux concepts du *machine learning*, leur permettant d'aborder ensuite les ouvrages plus spécialisés.

Michel DELATTRE
DSI du Groupe La Poste

Copyright © 2015 Dunod.

Copyright © 2015 Dunod.

Table des matières

Préface V

Avant-propos XVII

Première partie – Les fondements du Big Data

Chapitre 1 – Les origines du Big Data 3

1.1 La perception de la donnée dans le grand public 3

1.1.1 La révolution de l'usage 3

1.1.2 L'envolée des données 4

1.1.3 Un autre rapport à l'informatique 4

1.1.4 L'extraction de données ou d'information ? 5

1.2 Des causes économiques et technologiques 5

1.2.1 Une baisse des prix exponentielle 5

1.2.2 Des progrès initiés par les géants du web 6

1.2.3 Où se trouve la frontière du Big Data ? 7

1.3 La donnée et l'information 8

1.3.1 La recherche pertinente 8

1.3.2 Un avantage concurrentiel 8

1.3.3 Des clients plus exigeants 8

1.4 La valeur 9

1.5 Les ressources nécessaires 10

© Dunod – Toute reproduction non autorisée est un délit.

Copyright © 2015 Dunod.

1.6 De grandes opportunités 11

Chapitre 2 – Le Big Data dans les organisations 13

2.1 La recherche de l'Eldorado 13

2.1.1 L'entreprise dans un écosystème 13

2.1.2 Une volonté de maîtrise 14

2.1.3 Des besoins forts 14

2.2 L'avancée par le cloud 14

2.3 La création de la valeur 15

2.4 Les « 3V » du Big Data 15

2.4.1 Le volume 16

2.4.2 La vélocité 16

2.4.3 La variété 16

2.5 Un champs immense d'applications 17

2.6 Exemples de compétences à acquérir 18

2.6.1 Appréhender de nouveaux modèles de traitement des données 19

2.6.2 Maîtriser le déploiement de Hadoop ou utiliser une solution cloud 20

2.6.3 Se familiariser avec de nouvelles méthodes de modélisation 20

2.6.4 Découvrir de nouveaux outils d'analyse de données 21

2.7 Des impacts à tous les niveaux 21

2.7.1 Impacts sur la conception des systèmes 21

2.7.2 Conséquences sur l'organisation de l'entreprise 22

2.7.3 Impacts sur les relations entre clients et fournisseurs 22

2.7.4 Implications juridiques 23

2.8 « B » comme Big Data ou Big Brother ? 23

2.8.1 Connaissance client et préservation de la vie privée 23

2.8.2 La lassitude est à notre porte 23

2.8.3 Vers une démarche active 24

Chapitre 3 – Le mouvement NoSQL 27

3.1 Bases relationnelles, les raisons d'une domination 27

3.2 Le dogme remis en question 32

3.2.1 Les contraintes des applications web à très grande échelle 32

Copyright © 2015 Dunod.

3.2.2 *Le « théorème » CAP* 33
3.2.3 *Sacrifier la flexibilité pour la vélocité* 35
3.2.4 *Peut-on définir ce qu'est une base de données NoSQL ?* 36
3.3 Les différentes catégories de solutions 37
3.3.1 *Les entrepôts clé-valeur* 38
3.3.2 *Les bases orientées documents* 40
3.3.3 *Les bases orientées colonnes* 42
3.3.4 *Les bases de données orientées graphes* 46
3.4 Le NoSQL est-il l'avenir des bases de données ? 48

Chapitre 4 – L'algorithme MapReduce et le framework Hadoop 51
4.1 Automatiser le calcul parallèle 51
4.2 Le pattern MapReduce 52
4.3 Des exemples d'usage de MapReduce 56
4.3.1 *Analyse statistique d'un texte* 56
4.3.2 *Calcul d'une jointure entre deux grandes tables* 57
4.3.3 *Calcul du produit de deux matrices creuses* 60
4.4 Le framework Hadoop 61
4.4.1 *Planning des exécutions* 62
4.4.2 *Tolérance aux pannes* 63
4.4.3 *Découpage des données en lots* 64
4.4.4 *Fusion et tri des listes intermédiaires* 64
4.4.5 *Monitoring des processus* 66
4.5 Au-delà de MapReduce 66

Deuxième partie – Le métier de data scientist

Chapitre 5 – Le quotidien du data scientist 71
5.1 Data scientist : licorne ou réalite ? 71
5.1.1 *L'origine du terme data scientist et définitions courantes* 71
5.1.2 *Les compétences clés du data scientist* 73
5.1.3 *Comment recruter ou se former* 77
5.2 Le data scientist dans l'organisation 78
5.2.1 *Le data lab – une clé pour l'innovation par la donnée* 78

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

5.2.2 *La data lab – quelle place dans l'organisation ?* 80
5.3 Le workflow du data scientist 80
5.3.1 *Imaginer un produit ou un service* 81
5.3.2 *Collecte des données* 82
5.3.3 *Préparation* 84
5.3.4 *Modélisation* 84
5.3.5 *Visualisation* 85
5.3.6 *Optimisation* 86
5.3.7 *Déploiement* 87

Chapitre 6 – Exploration et préparation de données 89
6.1 Le déluge des données 89
6.1.1 *Diversité des sources* 90
6.1.2 *Diversité des formats* 92
6.1.3 *Diversité de la qualité* 93
6.2 L'exploration de données 94
6.2.1 *Visualiser pour comprendre* 94
6.2.2 *Enquêter sur le passé des données* 95
6.2.3 *Utiliser les statistiques descriptives* 96
6.2.4 *Les tableaux croisés dynamiques* 97
6.3 La préparation de données 99
6.3.1 *Pourquoi préparer ?* 99
6.3.2 *Nettoyer les données* 99
6.3.3 *Transformer les données* 100
6.3.4 *Enrichir les données* 102
6.3.5 *Un exemple de préparation de données* 103
6.4 Les outils de preparation de données 104
6.4.1 *La programmation* 104
6.4.2 *Les ETL* 105
6.4.3 *Les tableurs* 105
6.4.4 *Les outils de préparation visuels* 105

Copyright © 2015 Dunod.

Chapitre 7 – Le machine learning 107

7.1 Qu'est-ce que le machine learning ? 107

7.1.1 Comprendre ou prédire ? 107

7.1.2 Qu'est-ce qu'un bon algorithme de machine learning ? 112

7.1.3 Performance d'un modèle et surapprentissage 112

7.1.4 Machine learning et Big Data – sur quoi faut-il être vigilant ? 115

7.2 Les différents types de machine learning 117

7.2.1 Apprentissage supervisé ou non supervisé ? 117

7.2.2 Régression ou classification ? 118

7.2.3 Algorithmes linéaires ou non linéaires ? 118

7.2.4 Modèle paramétrique ou non paramétrique ? 118

7.2.5 Apprentissage hors ligne ou incrémental ? 119

7.2.6 Modèle géométrique ou probabiliste ? 119

7.3 Les principaux algorithmes 120

7.3.1 La régression linéaire 120

7.3.2 Les k plus proches voisins 121

7.3.3 La classification naïve bayésienne 123

7.3.4 La régression logistique 124

7.3.5 L'algorithme des k-moyennes 125

7.3.6 Les arbres de décision 127

7.3.7 Les forêts aléatoires 130

7.3.8 Les machines à vecteurs de support 131

7.3.9 Techniques de réduction dimensionnelle 133

7.4 Illustrations numériques 134

7.4.1 Nettoyage et enrichissement des données 135

7.4.2 Profondeur d'un arbre et phénomène de surapprentissage 136

7.4.3 Apport du « feature engineering » 139

7.4.4 Sensibilité de l'algorithme KNN au bruit 143

7.4.5 Interprétabilité de deux modèles 144

7.4.6 Bénéfices de l'approche ensembliste 145

Chapitre 8 – La visualisation des données 147

8.1 Pourquoi visualiser l'information ? 147

8.1.1 Ce que les statistiques ne disent pas 147

Copyright © 2015 Dunod.

© Dunod – Toute reproduction non autorisée est un délit.

8.1.2 *Les objectifs de la visualisation* 150
8.2 Quels graphes pour quels usages ? 151
8.3 Représentation de données complexes 158
8.3.1 *Principes d'encodage visuel* 158
8.3.2 *Principes de visualisation interactive* 160

Troisième partie – Les outils du Big Data

Chapitre 9 – L'écosystème Hadoop 167
9.1 La jungle de l'éléphant 168
9.1.1 *Distribution ou package* 168
9.1.2 *Un monde de compromis* 169
9.1.3 *Les services autour de Hadoop* 170
9.2 Les composants de Apache Hadoop 170
9.2.1 *Hadoop Distributed File System* 171
9.2.2 *MapReduce et YARN* 172
9.2.3 *HBase* 173
9.2.4 *ZooKeeper* 173
9.2.5 *Pig* 175
9.2.6 *Hive* 176
9.2.7 *Oozie* 176
9.2.8 *Flume* 176
9.2.9 *Sqoop* 177
9.3 Les principales distributions Hadoop 177
9.3.1 *Cloudera* 177
9.3.2 *Hortonworks* 177
9.3.3 *MapR* 178
9.3.4 *Amazon Elastic MapReduce* 179
9.4 Les briques analytiques à venir 180
9.4.1 *Impala* 180
9.4.2 *Drill* 180
9.5 Les librairies de calcul 182
9.5.1 *Mahout* 182
9.5.2 *MLlib de Spark* 183

Copyright © 2015 Dunod.

9.5.3 *RHadoop* 184

Chapitre 10 – Analyse de logs avec Pig et Hive 187

10.1 Pourquoi analyser des logs ? 187

10.2 Pourquoi choisir Pig ou Hive ? 188

10.3 La préparation des données 189

10.3.1 Le format des lignes de logs 190

10.3.2 L'enrichissement des logs 190

10.3.3 La reconstruction des sessions 192

10.3.4 Agrégations et calculs 192

10.4 L'analyse des parcours clients 194

Chapitre 11 – Les architectures λ 197

11.1 Les enjeux du temps réel 197

11.1.1 Qu'est-ce que le temps réel ? 197

11.1.2 Quelques exemples de cas réels 198

11.2 Rappels sur MapReduce et Hadoop 199

11.3 Les architectures **λ** 199

11.3.1 La couche batch 200

11.3.2 La couche de service 201

11.3.3 La couche de vitesse 202

11.3.4 La fusion 203

11.3.5 Les architectures λ en synthèse 204

Chapitre 12 – Apache Storm 207

12.1 Qu'est-ce que Storm ? 207

12.2 Positionnement et intérêt dans les architectures λ 208

12.3 Principes de fonctionnement 209

12.3.1 La notion de tuple 209

12.3.2 La notion de stream 209

12.3.3 La notion de spout 210

12.3.4 La notion de bolt 210

12.3.5 La notion de topologie 211

12.4 Un exemple très simple 212

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Index 215

Copyright © 2015 Dunod.

Avant-propos

Pourquoi un ouvrage sur le Big Data ?

Le Big Data est un phénomène aux multiples facettes qui fait beaucoup parler de lui mais dont il est difficile de bien comprendre les tenants et aboutissants. Il est notamment difficile de prévoir quel sera son impact sur les acteurs et sur les métiers de la DSI.

Cet ouvrage se veut un guide pour comprendre les enjeux des projets d'analyse de données, pour appréhender les concepts sous-jacents, en particulier le *machine learning* et acquérir les compétences nécessaires à la mise en place d'un data lab. Il combine la présentation des concepts théoriques de base (traitement statistique des données, calcul distribué), la description des outils (Hadoop, Storm) et des retours d'expérience sur des projets en entreprise.

Sa finalité est d'accompagner les lecteurs dans leurs premiers projets Big Data en leur transmettant la connaissance et l'expérience des auteurs.

À qui s'adresse ce livre ?

Ce livre s'adresse particulièrement à celles et ceux qui curieux du potentiel du Big Data dans leurs secteurs d'activités souhaitent franchir le pas et se lancer dans l'analyse de données. Plus spécifiquement, il s'adresse :

- aux décideurs informatiques qui souhaitent aller au-delà des discours marketing et mieux comprendre les mécanismes de fonctionnement et les outils du Big Data ;
- aux professionnels de l'informatique décisionnelle et aux statisticiens qui souhaitent approfondir leurs connaissances et s'initier aux nouveaux outils de l'analyse de données ;
- aux développeurs et architectes qui souhaitent acquérir les bases pour se lancer dans la *data science* ;
- aux responsables métier qui veulent comprendre comment ils pourraient mieux exploiter les gisements de données dont ils disposent.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Des rudiments de programmation et des connaissances de base en statistiques sont cependant nécessaires pour bien tirer parti du contenu de cet ouvrage.

Comment lire ce livre ?

Ce livre est organisé en trois parties autonomes qui peuvent théoriquement être lues séparément. Nous recommandons néanmoins au lecteur d'accorder une importance particulière au chapitre 3 (le mouvement NoSQL) et au chapitre 4 (l'algorithme Map Reduce).

La première partie commence par traiter des origines du Big Data et de son impact sur les organisations. Elle se prolonge par la présentation du mouvement NoSQL et de l'algorithme Map Reduce.

La deuxième partie est consacrée au métier de *data scientist* et aborde la question de la préparation des jeux de données, les bases du *machine learning* ainsi que la visualisation des données.

La troisième partie traite du passage à l'échelle du Big Data avec la plateforme Hadoop et les outils tels que Hive et Pig. On présente ensuite un nouveau concept appelé architecture λ qui permet d'appliquer les principes du Big Data aux traitements en temps réel.

Travaux pratiques

À plusieurs reprises dans cet ouvrage, le logiciel Data Science Studio est utilisé afin d'illustrer et de rendre plus concret le contenu. Cet outil, développé par la startup française Dataiku, fournit un environnement complet et intégré pour la préparation des données et le développement de modèles de *machine learning*.

Le chapitre 7 est ainsi illustré avec des exemples traités avec Data Science Studio.

Vous pouvez retrouver les jeux de données ainsi qu'une version de démonstration du logiciel à l'adresse suivante : www.dataiku.com/livre-big-data.

Remerciements

Les auteurs tiennent tout d'abord à remercier leurs proches pour leur patience et leur soutien pendant les périodes de rédaction de cet ouvrage. Leur reconnaissance va aussi à Didier Fauque et Nicolas Larousse respectivement directeur général de SQLI et directeur de l'agence SQLI de Paris ainsi qu'à Florian Douetteau, co-fondateur et directeur général de Dataiku.

Ils remercient leurs collègues, amis et clients qui ont bien voulu relire l'ouvrage et aider à le compléter par leurs retours d'expérience, en particulier Manuel Alves et Étienne Mercier.

Enfin, les auteurs remercient tout particulièrement Pierre Pfennig pour sa contribution sur le *machine learning*, Jérémy Grèze pour son aide sur la préparation des données et Pierre Gutierrez pour sa participation sur Pig, Hive et les parcours clients.

Copyright © 2015 Dunod.

PREMIÈRE PARTIE

Les fondements du Big Data

Cette première partie décrit les origines du Big Data sous les angles économiques, sociétaux et technologiques. Comment et pourquoi une nouvelle classe d'outils a-t-elle émergé ces dix dernières années ?

- Le premier chapitre explique comment le rattachement de la valeur à l'information en général plutôt qu'aux seules données structurées, et la baisse des coûts des ressources IT de plusieurs ordres de grandeur ont fait progressivement émerger le **Big Data**.
- Le chapitre 2 traite de l'impact du Big Data dans les organisations et présente la **caractérisation dite des 3V**. Il montre en quoi le Big Data n'est pas, loin s'en faut, un défi uniquement technique.
- Le chapitre 3 décrit l'émergence d'une nouvelle classe de systèmes de stockage : les **bases de données NoSQL**. Après avoir analysé les limites du modèle relationnelle classique face aux nouvelles exigences de performance et de disponibilité des applications web à très grande échelle, une classification de ces nouveaux systèmes sera proposée.
- Le chapitre 4 propose un zoom sur **MapReduce**, un schéma de parallélisation massive des traitements, introduit il y a une dizaine d'années par Google, qui est au cœur de beaucoup de projets Big Data. Des exemples d'usage de MapReduce seront décrits. Les limitations de ce modèle seront discutées avant d'esquisser les évolutions futures qui essaient de les surmonter.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Copyright © 2015 Dunod.

1

Les origines du Big Data

Objectif

Au commencement de l'informatique était la donnée. Le Big Data refocalise l'attention sur l'*information* en général et non plus sur la seule donnée structurée ce qui ouvre la voie à des usages inédits. Ce chapitre dresse un premier panorama des origines et des éléments fondamentaux de l'approche Big Data qui permettent d'accéder à la notion de valeur de l'information.

1.1 LA PERCEPTION DE LA DONNÉE DANS LE GRAND PUBLIC

1.1.1 La révolution de l'usage

Depuis le début de l'informatique personnelle dans les années 1980, jusqu'à l'omniprésence du Web actuelle dans la vie de tous les jours, les données ont été produites en quantités toujours croissantes. Photos, vidéos, sons, textes, logs en tout genre... Depuis la démocratisation de l'Internet, ce sont des volumes impressionnants de données qui sont créés quotidiennement par les particuliers, les entreprises et maintenant aussi les objets et machines connectés.

Désormais, le terme « Big Data », littéralement traduit par « grosses données » ou « données massives » désigne cette explosion de données. On parle également de « datamasse » en analogie avec la biomasse, écosystème complexe et de large échelle.

À titre d'exemple, le site *Planetoscope* (*http://www.planetoscope.com*) estime à 3 millions le nombre d'e-mails envoyés dans le monde chaque seconde, soit plus

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

de 200 milliards par jour en comptant les spams qui représentent presque 90 % des flux, et ce pour une population d'internautes qui avoisine les 2,5 milliards d'individus. En 2010, le zettaoctet de données stockées dans le monde a été dépassé et on prévoit en 2020 10 Zo (zettaoctets), soit 10 400 milliards de gigaoctets de données déversés tous les mois sur Internet.

1.1.2 L'envolée des données

Dans les domaines des systèmes d'informations et du marketing, la presse et les campagnes de mails regorgent de propositions de séminaires ou de nouvelles offres de solutions Big Data qui traduisent un réel engouement pour le sujet.

Comme mentionné précédemment, ce déluge d'informations n'est pas seulement imputable à l'activité humaine. De plus en plus connectées, les machines contribuent fortement à cette augmentation du volume de données. Les stations de production énergétiques, les compteurs en tout genre et les véhicules sont de plus en plus nombreux à être équipés de capteurs ou d'émetteurs à cartes SIM pour transférer des informations sur leur milieu environnant, sur les conditions atmosphériques, ou encore sur les risques de défaillance.

Même l'équipement familial est concerné par l'intermédiaire d'un électroménager intelligent et capable d'accompagner la vie de tous les jours en proposant des services de plus en plus performants (pilotage du stock, suggestion d'entretien, suivi de régime, etc.).

À cette production et cet échange massif de données s'ajoutent les données libérées par les organisations et les entreprises, désignées sous le nom d'open data : horaires de transports en commun, statistiques sur les régions et le gouvernement, réseau des entreprises, données sur les magasins...

1.1.3 Un autre rapport à l'informatique

Le nombre de données produites et stockées à ce jour est certes important mais l'accélération du phénomène est sans précédent ces cinq dernières années. Cette accélération est principalement due à un changement dans nos habitudes : ce que nous attendons des ordinateurs a changé et la démocratisation des smartphones et des tablettes ainsi que la multiplication des réseaux sociaux encouragent les échanges et la création de nouveaux contenus. La croissance du volume de données produites suit donc des lois exponentielles.

Ce volume impressionnant est à mettre en relation avec la consumérisation de l'informatique et avec les campagnes des grands du Web (Apple, Google, Amazon...) visant à encourager un usage toujours croissant de leurs services. Cette hausse de la consommation de leurs services se traduit mécaniquement par une demande croissante de puissance de traitement et de stockage de données qui engendre une obsolescence rapide des architectures IT habituellement utilisées : bases de données relationnelles et serveurs d'applications doivent laisser la place à des solutions nouvelles.

Copyright © 2015 Dunod.

1.1.4 L'extraction de données ou d'information ?

La présence de volumes de données importants est devenue, presque inconsciemment, une valeur rassurante pour les utilisateurs. Ainsi nombre d'entreprises sont restées confiantes sur la valeur de leurs bases de données, considérant que leur seule taille constituait en soi un bon indicateur de leur valeur pour le marketing.

Pour autant ce qui est attendu de ces données, c'est la connaissance et le savoir (c'est-à-dire le comportement d'achat des clients). Il est assez paradoxal de constater que les acteurs qui communiquent le plus sur l'utilisation des données produites sur le Web se gardent généralement d'aborder le sujet non moins important du ratio pertinence / volume.

Il y a donc une fréquente confusion entre la donnée et l'information qu'elle contient, le contexte de la création de la donnée et celui de son utilisation qui viennent enrichir à leur tour cette information. C'est là tout le champ d'investigation du Big Data.

1.2 DES CAUSES ÉCONOMIQUES ET TECHNOLOGIQUES

Les deux principales causes de l'avènement du Big Data ces dernières années sont à la fois économiques et technologiques.

1.2.1 Une baisse des prix exponentielle

Durant ces vingt dernières années le prix des ressources IT a chuté de manière exponentielle en accord avec la célèbre loi de Moore[1]. Qu'il s'agisse de la capacité de stockage, du nombreux de nœuds que l'on peut mettre en parallèle dans un data center, de la fréquence des CPU ou encore de la bande passante disponible (qui a favorisé l'émergence des services cloud). La figure 1.1 représente schématiquement ces évolutions.

La mise en place par des géants du web comme *Google*, *Amazon*, *LinkedIn*, *Yahoo!* ou *Facebook* de data center de plusieurs dizaines de milliers de machines bon marché a constitué un facteur déterminant dans l'avènement des technologies Big Data. Ce qui nous amène naturellement au second facteur.

1. La version la plus courante de cette « loi », qui est en réalité plutôt une conjecture, stipule que des caractéristiques comme la puissance, la capacité de stockage ou la fréquence d'horloge double tous les 18 mois environ.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

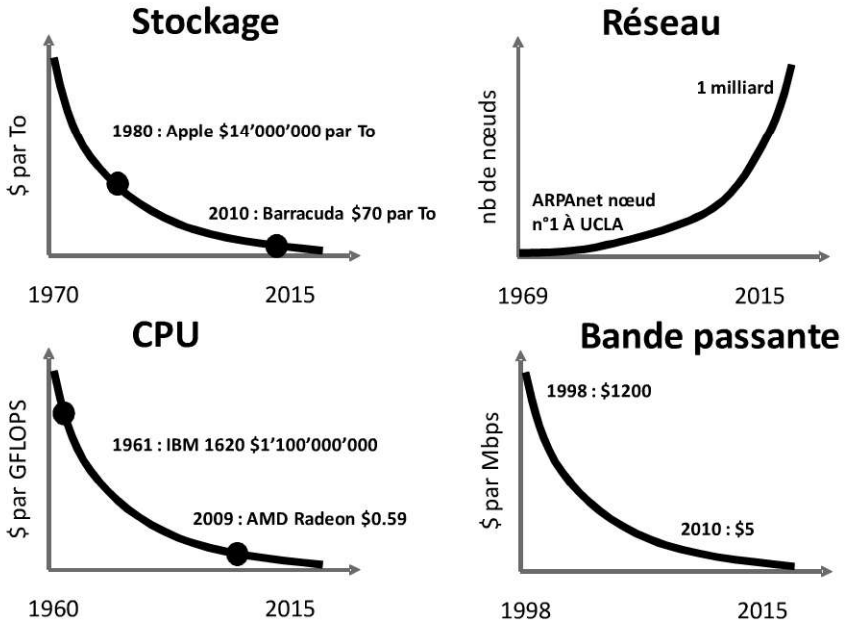


Figure 1.1 — Évolution des prix des ressources IT au cours des dernières décennies.

1.2.2 Des progrès initiés par les géants du web

Pour bénéficier de ces ressources de stockage colossales, les géants du web ont du développer pour leurs propres besoins de nouvelles technologies notamment en matière de parallélisation des traitements opérant sur des volumes de données se chiffrant en plusieurs centaines de téraoctets.

L'un des principaux progrès en la matière est venu de Google qui, pour traiter les données récupérées par les *crawlers* de son moteur de recherche et indexer la totalité du web, a mis au point un modèle de conception qui permet d'automatiser la parallélisation d'une grande classe de traitements. C'est le célèbre modèle **MapReduce** que nous décrirons en détail au chapitre 4.

De nombreuses avancées en génie logiciel développées à cette occasion ont par la suite essaimées dans la communauté open source qui en a proposé des systèmes équivalents gratuits. Le système de traitement parallèle **Hadoop Apache** est le principal exemple de ce transfert de technologie vers le monde open source. L'écosystème qui l'accompagne, notamment les systèmes de base de données non relationnelle comme *HBase*, le système de fichiers distribués *HDFS*, les langages de transformation et de requêtage *Pig* et *Hive* seront décrits au chapitre 9.

Face aux nouvelles exigences de montée en charge et de disponibilité, une nouvelle classe de système de gestion de base de données non relationnelle a émergé. On désigne habituellement ces systèmes au moyen du sigle NoSQL. Nous les décrirons en détail au chapitre 3 après avoir expliqué pourquoi et dans quelles circonstances ils se sont progressivement imposés face aux bases relationnelles.

Copyright © 2015 Dunod.

1.2.3 Où se trouve la frontière du Big Data ?

Donner une définition claire d'un terme aux contours aussi flous que le Big Data est délicat. On peut toutefois s'en faire une bonne idée en considérant les différents ordres de grandeurs d'espaces de stockage représentés sur la figure 1.2.

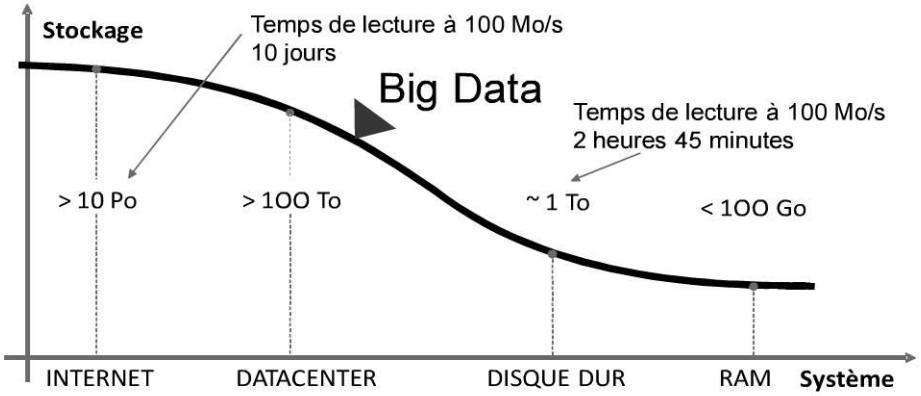


Figure 1.2 — Quelques ordres de grandeurs d'espaces de stockage ainsi que la frontière approximative du Big Data.

On s'accorde généralement pour situer la frontière des volumes qui relèvent du Big Data à partir du moment où ces données ne peuvent plus être traitées en un temps « raisonnable » ou « utile » par des systèmes constitués d'un seul nœud. À titre d'exemple si l'on doit traiter ou analyser un téraoctet de données, qui correspond à la taille d'un disque dur standard, en quelques minutes il faudra impérativement recourir à une mise en parallèle des traitements et du stockage sur plusieurs nœuds. Selon cette définition le Big Data n'est pas uniquement lié à la taille mais à la vitesse des traitements. Nous y reviendrons au chapitre 2 lorsque nous parlerons des trois « V » : volume, vitesse et variété.

Venons-en à quelques exemples et contre-exemples.

Quelques exemples qui ne relèvent pas du Big Data

- Un volume de données que l'on peut traiter au moyen d'une fiche Excel.
- Des données que l'on peut héberger dans un seul nœud d'une base de données relationnelle.
- Les données qui sont « chères » à produire telles que celles qui sont collectées par sondage ou par recensement ou produite par un organisme tel que l'INSEE. L'idée ici est que les données qui relèvent du Big Data sont typiquement créées pour un coût quasi nul.
- Les données issues de capteurs physiques comme ceux de l'internet des objets à venir.
- Les données publiques librement disponibles au téléchargement. Là encore on se place dans une perspective où ce qui relève du Big Data ne pourra être téléchargé au moyen d'une seule connexion Internet même à très haut débit.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Quelques exemples qui relèvent du Big Data

- Les volumes de données qu'il n'est pas possible de stocker ou de traiter avec les technologies traditionnelles que les SGBDR ou pour lesquelles le coût d'un tel traitement serait prohibitif.
- Les données de logs transactionnelles d'un site web d'une grande enseigne de distribution. Nous examinerons en détail ce cas de figure au chapitre 10.
- Le trafic d'un gros site web.
- Les données de localisation GSM d'un opérateur téléphonique sur une journée.
- Les données boursières échangées quotidiennement sur une grande place financière.

1.3 LA DONNÉE ET L'INFORMATION

1.3.1 La recherche pertinente

L'exemple le plus fréquent donné en illustration de l'usage des données est celui des moteurs de recherches qui constituent aujourd'hui un bon point de départ pour un parcours ciblé sur le Internet.

Il y a trente ans, plus l'utilisateur fournissait de mots clefs pour sa recherche et plus le nombre de réponses augmentait à l'écran. Dans la logique de l'utilisateur, cela était incohérent la plupart du temps, puisque l'ajout de mots clefs devait permettre au contraire de retrouver l'article recherché avec plus de précision. Il manquait un élément essentiel à ces moteurs, c'était la capacité de saisir ce que l'utilisateur avait « en tête » lorsqu'il cherchait une information. On parle alors de recherche contextuelle.

1.3.2 Un avantage concurrentiel

La principale avancée de Google (et dans une moindre mesure de Yahoo!), qui explique en partie son succès, est d'avoir très tôt travaillé sur la compréhension de l'univers utilisateur, certes à des fins publicitaires mais également pour apporter de la performance dans la recherche et se distinguer de la concurrence.

Cette réussite brillante de Google a métamorphosé la relation entre l'utilisateur et les entreprises. Dans de nombreux domaines, une prise de conscience de l'énorme potentiel offert par le traitement de l'information est venue encourager le déploiement de nouvelles méthodes d'analyse comportementale et de processus qui étaient auparavant l'apanage quasi exclusif des grands acteurs du Web.

1.3.3 Des clients plus exigeants

L'information pertinente est donc celle conditionnée (et attendue) par l'utilisateur dans de grands nombres de solutions. On dit souvent que 80 % de la réponse attendue

Copyright © 2015 Dunod.

est déjà présente dans le cerveau de l'utilisateur. Cela fait partie désormais de la culture de base de la relation client / systèmes d'informations.

C'est d'autant plus frappant que le rapport avec le progiciel a grandement changé : la plupart des entreprises exigent maintenant des éditeurs de solutions logicielles que l'utilisateur final puisse disposer de plus en plus de liberté dans la constitution de ses tableaux de bords.

De manière désormais innée, les clients de solutions informatiques savent qu'un logiciel qui s'adapte à l'environnement de travail maximise les chances de fournir une information pertinente et à forte valeur. À l'inverse, si le contexte d'utilisation ne peut pas être modifié, c'est à la solution informatique d'être capable de comprendre plus finement ce qui est recherché.

Autre progrès visible dans l'analyse de la donnée, le « stocker tout » cohabite avec le « stocker mieux ». L'idée sous-jacente est de *vectoriser* la donnée, c'est-à-dire de remplacer les données complexes par une description simplifiée et de se concentrer sur le sens. Cette rationalisation de l'information n'est cependant pas toujours aisée à mettre en œuvre.

Dans le cadre particulier de la connaissance du client, l'analyse comportementale et la recherche d'opportunités ou de nouveaux produits, nécessitent désormais la mise en relation de différents univers (potentiel financier du client, contexte économique, perception de l'image de l'entreprise, etc.). La complexité exposée plus tôt devient donc multidimensionnelle et à la limite de ce qu'un cerveau humain (ou même plusieurs) peut résoudre.

L'utilisation et l'analyse de la donnée sont donc à relier avec une finalité. Des résultats pertinents ayant du sens avec le métier sont conditionnés à la mise en place d'un ensemble d'informations prêt à être exploité de manière ciblée. Une information ciblée et satisfaisante devient enfin porteuse d'une valeur attendue. L'information extraite des données constitue un socle intéressant permettant des analyses ultérieures.

1.4 LA VALEUR

Parallèlement à la croissance régulière des données internes des entreprises, la montée en puissance des grands du Web (Google, Amazon, Linkedin, Twitter...) a révélé le potentiel immense des données externes à l'entreprise. Cette « masse » de données constitue la base des opportunités de demain. Ces acteurs réussissent à dégager des *business models* à forte rentabilité en nettoyant et en exploitant ces données pour se recentrer sur la pertinence de l'information isolée au service de leur stratégie.

Très progressivement, les entreprises et les éditeurs réalisent qu'il faut tirer profit de ce « trésor » et mettre en application les enseignements des récents progrès dans l'analyse des données. Dans cette perspective, les acteurs traditionnels de l'IT (IBM, Oracle, HP...) se sont lancés à leur tour bénéficiant au passage des travaux de R&D des grands du Web dont la plupart des produits sont sous licences open source.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Le marché du Big Data adresse une vaste étendue de besoins :

- La mise en relation de données d'origine très diverses.
- La prise en charge de volume jusqu'alors problématiques pour les systèmes d'information.
- La performance des traitements en termes de vitesse et de pertinence fonctionnelle.

Quant aux champs d'investigation possibles, les solutions sont capables de passer de la recherche sous différentes échelles : de l'information « macro », illustrant des tendances par exemple, à la recherche de l'information à l'échelle « micro », facteur déclenchant d'une décision très précise.

Les valeurs sont donc multiples : de la valeur stratégique à la valeur tactique et opérationnelle, en passant par les champs d'expérimentation pour les opportunités de demain encore non explorées.

1.5 LES RESSOURCES NÉCESSAIRES

Dans l'histoire récente des systèmes d'informations, les bases de données ont été créées pour rationaliser, structurer et catégoriser les données. Cet ensemble de solutions a connu un vif succès, et les années 1980 ont vu naître des bases de données de plus en plus complexes sans que l'on prenne le temps de simplifier leurs structures, avec des environnements de plus en plus gourmands. L'échelle de temps des utilisateurs ne permettait pas de se repencher sur ce qui avait été mis en place : les machines de la génération suivante répondaient seulement à chaque fois au supplément de ressources requises.

De nombreux constructeurs ont été ravis de vendre de la puissance de calcul et des infrastructures toujours plus puissantes. Toutefois la plupart des spécialistes sont restés prudents : certains d'entre eux ont rappelé tous les bienfaits de l'optimisation des traitements et de la rationalisation des systèmes d'informations, conscients qu'une limite des ressources allait prochainement imposer de revoir les besoins et les solutions architecturales proposées.

L'apparition de phénomènes non linéaires dans les bases de données, pourtant optimisées, a constitué une première alerte. Des architectes ont constaté des phénomènes troublants, comme un doublement de volume de la base qui entraînait un temps de traitement multiplié par 5, 10 voire 100. Les *capacity planners* ont commencé à observer des courbes exponentielles entre le nombre de demandes et les temps de réponses. Les engorgements de systèmes complexes et inadaptés ont confirmé que, pour certains traitements, il fallait radicalement revoir la manière avec laquelle les données étaient traitées.

Ainsi les structures traditionnelles ont montré leurs limites face à cet engouement de « consommation de la donnée » et il a bien fallu reconnaître que la progression de la performance passait inévitablement par la simplification et la délégation des processus des traitements à des machines optimisées.

Copyright © 2015 Dunod.

Ces évolutions ont été là aussi très diverses :

- De nombreux efforts ont été apportés aux structures matérielles en permettant des exécutions encore plus rapides, avec un recours massif à la mémoire, plutôt qu'à l'accès direct au stockage.
- La parallélisation est venue donner une réponse à ce besoin en s'inscrivant dans la simplification des traitements imbriqués et la délégation à des nœuds de traitements isolables[1].
- Les nouveaux moteurs de requête (*Not only SQL* ou NoSQL) se sont affranchis de certaines imperfections du SQL et ouvrant la porte à d'autres modes d'accès[2].
- Le recours à des algorithmes d'apprentissage statistiques a relancé l'exploration de données dans les limites du périmètre initial, comme cela est souvent le cas dans le domaine du décisionnel[3].

1.6 DE GRANDES OPPORTUNITÉS

Les possibilités offertes par le Big Data sont immenses :

- Analyser une grande variété de données, assembler des volumes extrêmes.
- Accéder à un éventail de données jamais atteint par le passé.
- Capter la donnée en mouvement (i.e. très changeante), même sous forme de flux continu.
- Compiler de grands ensembles et les mettre en correspondance en les recoupant.
- Découvrir, expérimenter et analyser des données cohérentes, tenter des rapprochements.
- Renforcer des associations entre données, intégrer et construire des ensembles performants et industriels.

En résumé

Le Big Data correspond bien à une réalité de l'usage de la donnée et de ce qui est désormais attendu d'un système d'informations.

Le Big Data, par son changement d'échelle et les réponses architecturales comme fonctionnelles qui y sont associés, est effectivement une rupture d'approche dans l'analyse de l'information.

1. Un cas particulier important de mécanisme de parallélisation, l'algorithme MapReduce sera décrit en détail au chapitre 4.
2. Le sujet sera abordé en détail au chapitre 3.
3. C'est le sujet des quatre chapitres de la partie 2.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

La valeur est devenue l'élément central de l'approche Big Data dans des dimensions qui n'ont pas plus rien à voir avec les « anciennes » extractions de l'informatique décisionnelle.

Copyright © 2015 Dunod.

2

Le Big Data dans les organisations

L'ambition clairement formulée du Big Data a rendu les entreprises impatientes d'atteindre le Graal de la valeur tant vanté par les acteurs du Web. Ce chapitre décrit les principaux changements inhérents à la révolution numérique dans les entreprises et en quoi, celles-ci doivent se réorganiser pour optimiser l'accès à l'information, aussi bien en interne qu'en externe.

2.1 LA RECHERCHE DE L'ELDORADO

2.1.1 L'entreprise dans un écosystème

Le paysage autour de l'entreprise a radicalement changé. À la fois dans les informations qu'elle requiert et les informations qu'elle doit fournir pour exister dans un écosystème économique et social où elle se doit d'être présente et réactive pour se développer. La notion de proximité numérique avec les clients est inscrite dans de nombreux plans stratégiques. C'est une vraie rupture culturelle qui a bousculé les repères qu'ils soient dans l'entreprise comme chez ses clients.

L'entreprise à l'ère du numérique devient de plus en plus ouverte aux flux d'informations, de façon directe ou indirecte (ne serait-ce que par ses collaborateurs). Il est attendu de plus en plus de la part de l'entreprise une contribution numérique au devenir citoyen, de la même manière que ses actions pouvaient, il y a trente ans, contribuer au développement de l'économie nationale. Progressivement le réel enjeu des données est compris non seulement comme un moyen de simplifier le contact avec

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

ses partenaires, ses prospects ou ses clients mais également comme le moyen d'acquérir des données nécessaires à la compréhension des grandes tendances de société.

2.1.2 Une volonté de maîtrise

Ces données, de plus en plus d'entreprises cherchent à les exploiter dans de nombreux domaines : personnalisation de leur relation client, marketing ciblé, traçabilité des parcours et retours clients, pilotage et valorisation de leur image sur les réseaux sociaux, optimisation des processus logistiques... ces données constituent un gisement d'informations au potentiel quasi illimité, à condition de pouvoir les maîtriser. Or les outils traditionnels ne suffisent plus.

Formatées sur le traitement de données dites structurées, les applications, qui jusqu'alors donnaient satisfaction, sont devenues impuissantes devant la diversité des données à traiter : origines diverses sous formes de texte, mais aussi sons, images, vidéos, sites web et blogs aux formats très variés.

2.1.3 Des besoins forts

Les éditeurs qui ont compris ce bouleversement et ces nouveaux besoins, ont commencé à regarder du côté du calcul scientifique pour traiter toutes ces données avec le maximum d'efficacité. Pour le marketing, de nouvelles méthodes ont permis de stocker indifféremment des formats, presque de manière brute, principalement pour offrir des possibilités de recouvrements avec des données issues de processus complètement différents (réseau des boutiques, parcours sur le web, fréquentation des sites, échanges téléphoniques, dossiers clients, etc.). Cela a permis d'établir des regroupements d'informations totalement nouveaux et constituer des profils opportunistes sur les clients, sur leur potentiel, leurs projets et leurs envies.

Les exemples ne manquent pas dans la presse décrivant des entreprises qui ont assemblé des données de plusieurs sources et peuvent désormais proposer à leurs clients un ciblage des offres (réduction, abonnement) en fonction de leur localisation, à proximité d'une galerie commerciale par exemple et cela, grâce au Big Data.

2.2 L'AVANCÉE PAR LE CLOUD

Un des facteurs important dans l'émergence du Big Data est le *cloud computing* qui a grandement facilité l'accès aux infrastructures. Basé sur des ressources ajustables, par durée identifiée et à un coût plus adapté, le cloud computing a ouvert de nombreuses portes aux projets innovants en abaissant considérablement le coût du ticket d'entrée sur ces solutions.

Place est faite aux expérimentations, aux POC (*Proof of Concept*) sur plateformes performantes qui permettent d'essayer certains traitements, certains recoupements et transformations, sans (trop) ponctionner le budget de la DSI.

Copyright © 2015 Dunod.

Lieu de convergence des données d'origines très diverses (météo, localisation, économie, tendances multiples), le cloud a fait entrer l'entreprise dans des domaines de croisement et d'études rarement tentés par le passé, en combinant l'analyse de la mesure plus ou moins poussée, pour aborder celui fascinant de la prédiction.

2.3 LA CRÉATION DE LA VALEUR

Deux manières de créer de la valeur grâce au Big Data sont :

- **La conception de nouveaux produits**. Les grands acteurs du Web aujourd'hui sont des exemples suivis par des centaines de startups en Europe et aux USA. Les domaines touchés sont immenses et même dans des métiers connus, ces produits peuvent apporter une étendue de vision et de justesse très appréciée, comme dans le domaine bancaire ou celui de l'assurance en fournissant le produit réellement attendu par le public.
- **L'angle analytique en élargissant le champ d'investigation à des données qui n'étaient pas accessibles sur d'autres outils**. Par exemple, la mise en relation de données disponibles sur des réseaux plus personnels (réseaux sociaux) pour déterminer l'écosystème autour du client, son réseau, ses influences et celui des ventes potentielles en le reliant avec le contexte géographique et la proximité du réseau des magasins. C'est-à-dire construire les conditions favorables pour assurer le meilleur accueil du produit auprès du client.

Le déploiement de tels outils ne peut se faire sans effort. L'émergence du Big Data en entreprise doit impérativement bénéficier de changements culturels profonds afin qu'il soit un succès (la section 2.7 reviendra sur le sujet).

La promotion des valeurs telles que celles de l'agilité et la flexibilité est indispensable, alors que souvent elles sont opposées aux modes d'organisations pyramidales actuels. L'innovation portée par ces approches Big Data doit reposer sur l'ambition et la confiance des structures managériales.

Les cycles courts de test doivent stimuler la créativité dans les équipes et de nombreux efforts doivent être maintenus pour sortir de la culture de l'échec trop souvent présente dans les entreprises françaises. C'est à ce prix que de nouvelles opportunités s'offriront aux équipes s'attelant aux défis du Big Data.

2.4 LES « 3V » DU BIG DATA

Afin de mieux cerner les caractéristiques du Big Data des spécialistes d'IBM ont proposé trois propriétés qui les caractérisent à des degrés divers, il s'agit du *volume*, de la *vélocité* et de la *variété*. On les appelle communément les **3V**. D'autres dimensions sont fréquemment rajoutées, mais on ciblera ici la présentation sur ces trois propriétés.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

2.4.1 Le volume

La question du volume a déjà été évoquée dans la section 1.2.3 nous n'y reviendrons donc pas ici.

L'une des technologies aujourd'hui couramment utilisées pour y faire face est le framework Hadoop. Le sujet sera abordé en détail aux chapitres 4 et 9. Disons simplement ici que le stockage est assuré par un système de fichiers distribué appelé HDFS et que la parallélisation des traitements exploite un pattern appelé MapReduce. Tous les problèmes de parallélisation ne peuvent exploiter l'algorithme MapReduce, mais lorsque c'est le cas, celui-ci offre une scalabilité quasi infinie. Hadoop est aujourd'hui un projet de la fondation Apache. La version 2.0 de Hadoop réutilise la gestion distribuée des ressources développées pour exécuter MapReduce pour permettre l'exécution d'autres schémas de calcul, plus généraux.

Hadoop est aujourd'hui disponible dans le cloud, la solution Elastic Map Reduce de Amazon est un exemple.

2.4.2 La vélocité

Au cœur du Time To Market

La vélocité ou vitesse de circulation des données a connu une évolution similaire à celle du volume au sein des entreprises. Contexte économique et évolution de la technologie, les entreprises se trouvent de plus en plus au milieu d'un flux continuel de données, qu'il soit interne ou externe.

Rien de bien nouveau pour certains métiers comme celui de la finance, où le *trading*, par exemple, tire de la vélocité un avantage concurrentiel fondamental qui permet chaque jour de prendre une longueur d'avance dans la décision, source de gains financiers. Le temps réel est déjà là, depuis plusieurs années dans ces familles de métiers, bien installé. Le sujet sera abordeì en deìtail aux chapitres 11 et 12.

Au service des clients

Le paramètre vélocité est déterminant dans les situations où il s'agit de tenir compte en temps réel des souhaits exprimés par un client et de l'état de disponibilité du stock pour fournir le meilleur service possible. Dans d'autres situations on pourra utiliser des données de géolocalisation pour guider un client vers un magasin moins exposé aux problèmes de circulation ou pour lui offrir des services de proximités

2.4.3 La variété

L'une des grandes ambitions du Big Data est de proposer le recoupement d'informations ou la mise en correspondance de données d'origines très diverses. Ce serait même l'une des principales sources de valeur. Pourtant, il ne faut pas se leurrer, parmi les 3V c'est le seul pour lequel il n'existe à l'heure actuelle aucune méthode universelle. C'est sans nul doute le problème le plus épineux car le traitement et le recoupement de données de sources et de formats variés demandent une étude adaptée à chaque situation.

Copyright © 2015 Dunod.

Un exemple typique d'utilisation de données hétéroclites est celui d'un croisement entre des données contenues dans un CRM (gestionnaire de la relation client), des données de géolocalisation, des données extraites d'un réseau social qui, collectivement, permettront d'enrichir un profil utilisateur avec des informations à caractère affectif très souvent corrélées au déclenchement d'un acte d'achat.

Pour faire face à cette diversité de structures, certains systèmes NoSQL (voir le chapitre 3) utilisent des schémas de données qui s'écartent du modèle relationnel classique. Les bases orientées graphes ou les bases orientées colonnes sont des exemples.

La prise en compte de la diversité des formats de données fera typiquement partie de la phase de préparation des données, le sujet du chapitre 6.

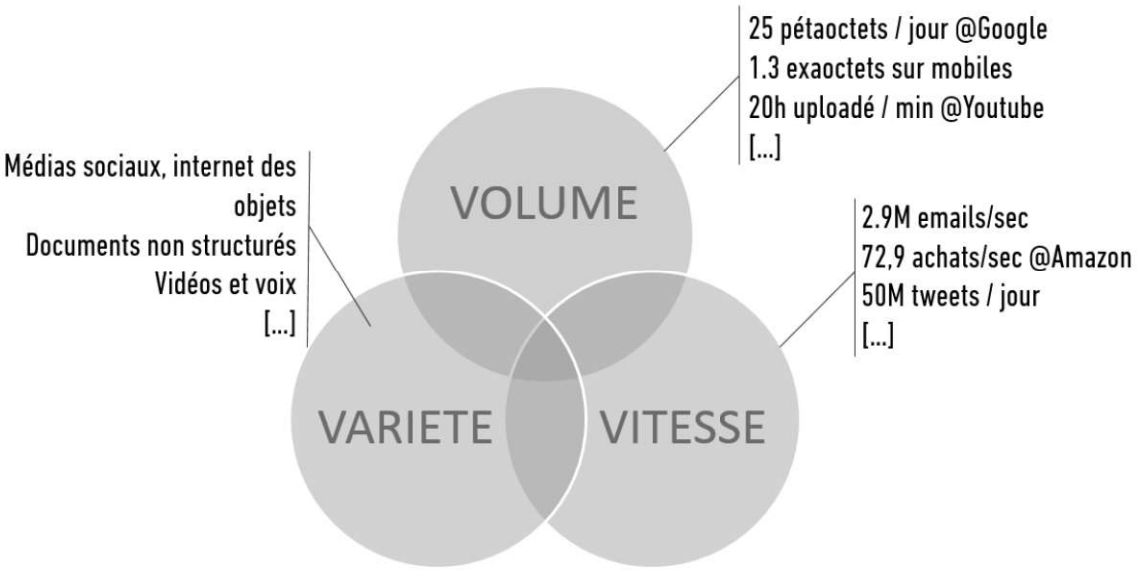


Figure 2.1 — Le Big Data à l'intersection des trois « V ».

2.5 UN CHAMPS IMMENSE D'APPLICATIONS

Voici quelques exemples d'applications du Big Data en entreprise. La liste est loin d'être exhaustive :

- **Analyser les données en mouvement**
  - La surveillance d'un grand nombre de nœuds de données (cloud, hébergement, traitements répartis).
  - La sécurité électronique et la détection de fraude dans un contexte banque assurance.
  - Le suivi en temps réel et à forte réactivité de clients (commerce de détail, téléassistance).
  - Le pilotage avancé de systèmes complexes (recherche en astronomie et en physique des particules, domaine spatial, forces armées).

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

  - L'analyse des logs (voir chapitre 10) et surveillance des processus.
  - Le tracking des identifiants, connexions et pistage des transports (avec le RFID par exemple).
  - La logistique avancée avec ou sans contact physique.

- **Analyser une grande variété de données**
  - Le décodage et l'analyse des humeurs sur les réseaux sociaux ou dans la blogosphère.
  - L'accostage avec la stratégie d'entreprise.
  - Le suivi de phénomènes propagés (santé : épidémies, intrusions, ingénierie sociale, terrorisme).
  - L'analyse multimédia à partir de formats vidéo, audio, texte, photos (sécurité, traductions, médiamétrie).

- **Traiter un volume conséquent de données**
  - Rapprochement d'objets métiers : produits, clients, déploiement, stocks, ventes, fournisseurs, lieux promotionnels.
  - Détection de fraudes, repérage de clients indélicats ou manipulateurs.
  - Toxicité, défense des intérêts d'un groupe, *feedback* de crise.
  - Management du risque, prise de décision sensible appuyée de modèles prévisionnels.
  - Analyse de contexte, d'environnement.

- **Découvrir et expérimenter**
  - Approcher la notion de désir ou de sentiment déclencheur d'action.
  - Cerner l'entourage social d'un client, les conditions favorables.
  - Expérimenter l'impact d'un nouveau produit, son ressenti.
  - Mesurer l'efficacité d'une stratégie de conquête, mesurer des écarts ou des erreurs de positionnement d'image.
  - Profilage de nouveaux comportements.
  - Identification de nouveaux indicateurs d'influence.

2.6 EXEMPLES DE COMPÉTENCES À ACQUÉRIR

Les paragraphes qui suivent illustrent quelques exemples de compétences qu'une société pourra être amenée à acquérir pour faire du Big Data. Il serait vain de vouloir en donner une liste exhaustive mais les exemples qui suivent aideront à s'en faire une idée.

Copyright © 2015 Dunod.

2.6.1 Appréhender de nouveaux modèles de traitement des données

Nous avons déjà évoqué le pattern MapReduce, un des algorithmes historiques du Big Data puisqu'il est au cœur des premières versions de Hadoop. Même si comme nous le verrons au chapitre 9 ce modèle est supplanté par d'autres, plus flexibles il continue à jouer un rôle structurant dans de nombreux traitements asynchrones.

Si son principe est simple comme nous le verrons au chapitre 4, les détails en revanche sont plus ardus et exigeront pour être pleinement maîtrisés un effort d'apprentissage significatif de la part des développeurs, l'expérience des pionniers que sont Facebook et Yahoo! l'a amplement démontré. Il est vraisemblable cependant qu'une majorité des développeurs métiers pourra se dispenser d'une connaissance approfondie de ces aspects algorithmiques. En effet, pour bénéficier du parallélisme massif de MapReduce, ils n'auront que rarement à implémenter explicitement l'algorithme MapReduce[1] mais utiliseront plutôt des abstractions de haut niveau, essentiellement des langages de scripts (comme Pig) ou des langages pseudo SQL (comme Hive QL) plus familiers.

Dès lors la question se pose : est-il possible de faire l'impasse complète sur les concepts de MapReduce ? Hélas non, et c'est là que résident la subtilité et les risques. Comme souvent dans l'IT, on ne peut pas oublier complètement une complexité algorithmique qui a été masquée. Une connaissance élémentaire des mécanismes sous-jacents reste utile pour faire face aux situations où les automatismes dysfonctionnent ou dès lors que les performances exigent des optimisations.

Dans le même ordre d'idées, il faudra veiller à ce que les plans d'exécution MapReduce générés par Hive ne soient pas trop complexes pour être exécutables et testables en un temps raisonnables. Liées aux ordres de grandeurs inhabituels des quantités de données traitées, c'est tout un jeu d'intuitions nouvelles à acquérir par les concepteurs.

Quel est alors l'effort à consentir pour maîtriser la programmation MapReduce explicite (sans scripts) sous Hadoop ? Une durée comprise entre six mois à un an ne semble pas surestimée s'il s'agit d'acquérir une expérience significative. Ces compétences sont aujourd'hui détenues par quelques petites sociétés spécialisées et sont rarement développées en interne bien qu'il existe des cursus de formation dédiés.

Pour ce qui concerne les langages de plus haut niveau comme Pig et Hive QL, on peut estimer à deux ou trois semaines le temps de parvenir à un niveau de compétences suffisant, vu la proximité avec les langages existants comme SQL. En donnant ces estimations, nous présupposons qu'un coach soit à disposition pour assurer la formation initiale des développeurs ou alors qu'une cellule spécialisée et dédiée, fonctionnant en mode R&D, soit mise en place localement dont la mission sera de monter en compétence sur ces sujets et, à terme, former le reste des troupes.

1. L'implémentation des fonctions Map et Reduce comme on le verra au chapitre 4.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

2.6.2 Maîtriser le déploiement de Hadoop ou utiliser une solution cloud

À l'appréhension des concepts et des langages de scripts spécifiques à MapReduce, il convient d'ajouter la maîtrise du déploiement d'une plateforme qui implémente ce modèle. Le rôle d'un tel framework consiste à masquer l'énorme complexité technique qu'implique un traitement massivement parallèle par un cluster de machines non fiables : planification des tâches en fonction des ressources du cluster, réplication des données, réexécution des tâches en cas d'échec, garantie de très haute disponibilité, etc. Un outil sans véritable concurrence à ce jour domine la scène, il s'agit de Hadoop, un framework libre développé ses dernières années sous l'égide de la fondation Apache.

Bien qu'il existe d'excellents ouvrages sur le sujet[1], le déploiement « *on premise* » d'un cluster Hadoop et son optimisation reste un travail d'experts hautement qualifiés. Dès lors, l'alternative pour une DSI qui souhaiterait mettre en œuvre Hadoop, consistera à choisir de faire appel à un prestataire spécialisé ou alors à utiliser un déploiement de Hadoop dans le Cloud chez un fournisseur PaaS. La solution Elastic Map Reduce d'Amazon Web Services constitue ici une référence. Dans ce dernier cas, pour les très gros volumes de données, interviendront des considérations de coûts de transferts qu'il ne faudra pas prendre à la légère. La maturité des plateformes PaaS et la disponibilité de tutoriels pédagogiques devraient rendre accessibles le paramétrage et le déploiement d'applications à la plupart des DSI. Quelques semaines d'expérimentation devraient suffire pour permettre à une équipe IT chevronnée de maîtriser l'exploitation de Hadoop en mode PaaS.

2.6.3 Se familiariser avec de nouvelles méthodes de modélisation

Traiter d'énormes quantités de données non structurées exige non seulement de nouveaux algorithmes et des nouvelles infrastructures de calcul, mais impose également de repenser l'organisation des données elles-mêmes. Dans ce nouveau contexte, les enjeux de performances priment le plus souvent sur ceux d'une stricte intégrité référentielle, contrairement aux principes qui s'appliquent aux bases relationnelles. Par ailleurs, le caractère non structuré des données rend caduque la notion de schéma prédéfini. De nouvelles catégories de base de données répondent à ces exigences, les bases NoSQL que nous décrirons au chapitre 3. Ce type de bases pourra jouer aussi bien le rôle de source que de destination pour les jobs MapReduce. L'expérience montre qu'elles s'avèrent particulièrement bien adaptées aux données hiérarchiques ou structurées en graphe qui sont monnaie courante dans un contexte Big Data. L'archétype ici est celui d'une *webtable* qui répertorie les attributs d'une collection de milliards de pages web indexées par leur URL. À la panoplie des savoir-faire déjà évoqués, il convient donc d'ajouter de nouvelles techniques de modélisation. À cette occasion, certains réflexes anciens devront être désappris, les processus de dé-normalisation et de duplications, autrefois prohibés, étant désormais à l'origine même des gains en performances et des capacités à monter en charge linéairement.

1. White, T., *Hadoop: The Definitive Guide*, 3rd Edition. O'Reilly, 2012.

Copyright © 2015 Dunod.

Une compréhension fine des algorithmes ou des applications qui exploitent les données sera nécessaire pour pouvoir les modéliser. Contrairement aux systèmes relationnels la structure optimisée de ces nouveaux modèles dépend fortement des applications qui les utilisent (voir chapitre 3).

2.6.4 Découvrir de nouveaux outils d'analyse de données

À la différence de la *business intelligence* classique qui exploite des données structurées d'une entreprise pour fournir une aide à la décision, le Big Data cherche à exploiter des quantités massives de données non structurées, situées en partie hors de l'entreprise, afin d'en dégager des tendances de consommation, des schémas de comportement, voir pour favoriser l'innovation.

Ce genre de tâches s'appuie sur de nouveaux outils d'analyse et de visualisation. De nombreux acteurs commerciaux, de même que la communauté du logiciel libre proposent des outils pour faciliter l'interaction avec un cluster Hadoop et permettre l'exploration et la visualisation interactive des données dans les outils BI et OLAP classiques. Dans la deuxième partie de cet ouvrage, nous décrirons les outils et les méthodes de travail que l'on utilise pour faire de l'analyse prédictive (le chapitre 8 est dédié à la visualisation des données).

Contrairement au framework Hadoop ou aux subtilités de la programmation MapReduce, les outils d'analyse Big Data sont des extensions des outils classiques de Data Mining. Leur maîtrise par les experts métiers concernés ne devrait pas par conséquent poser de problème majeur, ni constituer un investissement significatif. Toutefois, comprendre le rôle respectif des différents acteurs dans la galaxie Big Data pourra s'avérer un peu déroutant car les relations entre la myriade de distributeurs de solutions Hadoop, de fournisseurs de solutions Map Reduce dans le cloud, d'éditeurs d'outils d'analyse en temps réel et d'intégrateurs sont souvent complexes et parfois redondantes.

Un point important à noter est que l'analyse de données Big Data nécessitera le plus souvent le développement d'applications spécifiques. Les développeurs continueront en l'occurrence à utiliser leurs environnements de développement favori et les langages de programmation qu'ils maîtrisent déjà mais les applications qu'ils développent s'adresseront à une catégorie d'utilisateurs finaux spécialisés, les analystes de données, dont ils devront assimiler le vocabulaire et avec qui ils devront apprendre à collaborer.

2.7 DES IMPACTS À TOUS LES NIVEAUX

2.7.1 Impacts sur la conception des systèmes

Comme nous le verrons au chapitre 3 consacré aux bases de données NoSQL, la conception de systèmes qui exploitent des données Big Data exige parfois de remettre en question certaines priorités au premier rang desquelles l'intégrité des données. Dans de nombreuses situations il est désormais plus important pour un site d'e-commerce ou

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

pour une plateforme sociale, d'assurer une très haute disponibilité du système quitte à y **sacrifier la stricte intégrité des données**.

De **nouveaux modèles de données**, plus complexes (bases de données orientées colonnes) ou plus simples (entrepôts clé-valeur) que le modèle relationnel voient le jour avec lesquels les développeurs devront se familiariser (voir chapitre 3). Dans beaucoup de situation les bonnes pratiques font pour l'instant défaut et les équipes travailleront par conséquent dans en mode plus expérimental qu'avec les modèles éprouvés depuis des décennies.

2.7.2 Conséquences sur l'organisation de l'entreprise

Ce mode de travail expérimental et agile des équipes de développement, qui n'est qu'une conséquence de l'absence de recul sur des technologies encore récentes, impliquera de repenser le rôle des managers opérationnels au sein des DSI. Ceux-ci ne pourront plus se contenter d'être de simples gestionnaires mais devront se muer en agents d'innovation. **Valoriser la curiosité, instaurer une culture du prototypage et faire la place belle à l'apprentissage par l'erreur**, telles seront les rôles que devra assumer le manager agile.

De nouveaux métiers apparaissent, le plus connu étant sans doute celui de **data scientist**. Son rôle sera de piloter la réalisation d'applications prédictives et d'apporter leur expertise statistique aux équipes métiers. Les compétences se situent à la confluence de plusieurs disciplines : programmation, statistiques, intelligence artificielle et expertise métier. Les chapitres 5 à 8 de la deuxième partie de cet ouvrage détailleront les différentes facettes de ce nouveau métier.

Il est vraisemblable que l'apparition de ces nouveaux rôles conduira à une redistribution du pouvoir dans certaines entreprises. Des décisions prises jusque-là par quelques individus qui pensent, à tort ou à raison, pouvoir se prévaloir d'une longue expérience seront désormais prises de manière plus factuelle, sur la base de données objectives et d'analyses menées par des data scientists. Trouver le bon équilibre entre la créativité humaine, qui est une projection dans l'avenir, et l'analyse factuelle de données, qui est un retour vers le passé, fera inévitablement l'objet de tâtonnements dans beaucoup d'organisations.

2.7.3 Impacts sur les relations entre clients et fournisseurs

Les relations entre clients et fournisseurs sont d'ores et déjà chamboulées par les réseaux sociaux et par les nombreux sites d'évaluation de produits et de services que permettent les technologies Big Data. De plus en plus, les clients font davantage confiance aux évaluations postées par d'autres acheteurs qu'aux déclamations un peu naïves et auto-satisfaites de la publicité traditionnelle. Pour gérer ces nouveaux modes de communication, plus ouverts et plus libres, un nouveau métier, toujours en évolution, est apparu : celui de **community manager**. Verra-t-on là aussi un transfert de pouvoir entre les experts marketing au sens traditionnel du terme et ces nouveaux gestionnaires de communauté ?

Copyright © 2015 Dunod.

2.7.4 Implications juridiques

Hormis la question de l'anonymat des données qui est l'objet de la section suivante les deux principaux enjeux juridiques sont les suivants :

- **Usage de données hors cadre** – Un des principes de droit qui prévaut actuellement dans les législations qui réglementent l'usage des données personnelles est que chaque donnée doit en principe posséder un usage bien déterminé. Ce principe entre naturellement en conflit direct avec de nombreux usages du Big Data qui consistent précisément à faire l'inverse, à savoir utiliser des données hors des contextes initialement prévus. Ces questions sont complexes et, pour l'heure, de nombreuses zones grises existent encore.
- **Localisation des données** – Si elle ne pose pas de problèmes pour les usages grand public, la localisation de données sensibles peut en revanche poser des problèmes aux entreprises ou aux collectivités publiques en particulier dans le domaine de la santé. Le problème est d'autant plus épineux que les législations ne sont pas harmonisées d'un pays à un autre. Alors qu'en droit européen les données confiées à un tiers reste la propriété du client, dans les pays anglo-saxons c'est le prestataire qui en devient le propriétaire, sans que celui-ci ne soit soumis à aucune obligation de protection. Depuis la promulgation du Patriot Act aux États-Unis, le gouvernement américain a même l'autorisation légale de fouiller des données entreposées sur son sol. Là aussi la situation évolue avec l'apparition de cloud nationaux ou européens comme les services *Cloudwatt*, *Numergy* ou OVH par exemple.

2.8 « B » COMME BIG DATA OU BIG BROTHER ?

2.8.1 Connaissance client et préservation de la vie privée

La préservation de la vie privée a connu récemment un rebond avec les révélations de l'affaire Snowden. La manipulation à grande échelle de ces données met en lumière l'importance de l'enjeu de la préservation de la vie privée.

Les entreprises européennes y sont très sensibles. Et si ce n'est pas le cas, pour certaines d'entre elles, les organismes régulateurs et les associations sont là pour y veiller.

De plus en plus d'entreprises utilisent indirectement des données personnelles par le biais de l'anonymisation qui, dans bien des cas, est assez simple à mettre en place. Une entreprise aurait tort de ne pas saisir l'occasion de valoriser des données qu'elle possède, alors qu'il existe des outils qui peuvent le faire conformément à la législation de protection de la vie privée.

2.8.2 La lassitude est à notre porte

Le point de friction est, à notre avis, ailleurs : celui de l'exaspération qu'un sur-usage des données peut engendrer auprès d'un prospect ou d'un client.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Encore récemment, on a appris par la presse qu'un grand réseau social avait mené une expérience « sociale » à l'insu de ses clients. Cette expérience, rendue possible par les conditions d'utilisation (que personne ne lit), avait pour but de mesurer l'impact sur l'humeur des utilisateurs suscitée par des articles ciblés. Cet impact avait été analysé via le contenu de leur mail.

Il est assurément légitime de considérer une telle initiative comme préoccupante face au respect de la vie privée. L'émotion provoquée par cette « affaire » a cependant été limitée dans le grand public. Ce manque de réaction est sans doute à corréler avec la redéfinition implicite de l'intimité par les réseaux sociaux. En effet, depuis quelques années, de nombreuses personnes exposent leur vie privée *captivante* aux yeux de tous. De nombreux experts en ont conclu qu'un cap était passé et que la vie privée n'avait plus de sens. Les mêmes se sont emballés en affirmant que ceux qui voulaient se protéger de cette exposition « *avaient probablement des choses à se reprocher* ». Le PDG de Google, Eric Schmidt est même allé jusqu'à déclaré « *Si vous faites quelque chose et que vous ne voulez que personne ne le sache, peut-être devriez-vous déjà commencer par ne pas le faire* ».

D'autres spécialistes, en analysant réellement ce type de données, ont plus décrit ce phénomène comme une mise en scène de la vie, cette fois-ci virtuelle, finalement contrôlée et souvent assez différente de la vraie vie.

2.8.3 Vers une démarche active

Il y a à l'évidence un engouement pour les sites qui ciblent pertinemment les envies des clients (lorsqu'ils ne les suscitent pas). Ces « détecteurs de désir » sont de plus en plus efficaces et les exemples ne manquent pas pour illustrer leur redoutable efficacité. Mais comment réagira le client lorsque son portable n'arrêtera pas de lui signaler une énième offre « immanquable » au cours de ses promenades ?

Il est probable que cette sur-sollicitation deviendra invivable malgré toute l'intelligence déployée par les solutions Big Data. Pourquoi ? Parce que le client n'aura plus l'impression d'être à l'origine de cette sollicitation.

Il semble donc que, parallèlement aux discours centrés sur les données adaptées aux entreprises, l'usage des solutions Big Data nécessite un recadrage dans le respect de cette sphère, immatérielle certes, mais fondamentale à la relation client qui est celle du bien-être de l'utilisateur.

L'avenir est donc majoritairement aux solutions attentives aux expressions du client issues d'une démarche volontaire, résumées sous le terme de VRM (*Vendor Relationship Management*) ou gestion de la relation vendeur, c'est tout simplement la réciproque du CRM (*Customer Relationship Management*). Le VRM est une phase importante où le client détermine qui a le droit de le contacter mais dans un cadre très précis d'une famille de produits pour lequel le client attend des suggestions de la part des vendeurs.

Copyright © 2015 Dunod.

En résumé

Loin d'être un défi purement technologique consistant à faire face aux 3V, le Big Data promet de chambouler le monde de l'entreprise, son organisation, ses métiers et ses modes de décisions. Nombre de projets Big Data auront un caractère expérimental dont il faut être conscient. De simples gestionnaires les managers devront se muer en agents d'innovation au service des équipes techniques qu'ils encadrent. Nous avons examiné les impacts que l'on pouvait attendre au niveau de la conception des systèmes, de l'organisation de l'entreprise, des relations entre clients et fournisseurs et des incertitudes juridiques. Enfin, nous avons énuméré quelques-unes des nouvelles compétences et savoir-faire dont devront se doter les entreprises qui souhaitent bénéficier du Big Data.

Copyright © 2015 Dunod.

Copyright © 2015 Dunod.

3

Le mouvement NoSQL

Objectif

Apparus dans le sillage du mouvement Big Data, les systèmes NoSQL sont des systèmes stockage d'un genre nouveau. Ce chapitre décrit le contexte historique dans lequel ils sont apparus, comment et dans quelle mesure ils parviennent à résoudre les problèmes spécifiques au Big Data. Leurs caractéristiques seront comparées à celles des SGBDR classiques. Nous décrirons ensuite quatre catégories de systèmes NoSQL : les entrepôts clé-valeur, les bases orientées documents, les bases de données orientées colonnes et enfin les bases de données orientées graphes. L'accent est mis sur la compréhension des différents modèles de données et sur les compromis qu'ils impliquent en termes de disponibilité, de performance et de tolérance aux pannes. Des exemples d'usages seront proposés à chaque fois.

3.1 BASES RELATIONNELLES, LES RAISONS D'UNE DOMINATION

Le cœur du mouvement Big Data réside dans la mise en œuvre de traitements massivement parallèles qui bénéficient, d'une part, des percées technologiques et algorithmiques réalisées ces dix dernières années par les géants du web et, d'autre part, des prix en chute libre de l'espace de stockage. Avant d'aborder dans les chapitres qui suivent le traitement, l'analyse et la visualisation des données, commençons par décrire une classe importante de systèmes de stockage utilisés par le Big Data que l'on dénomme usuellement par le sigle NoSQL. Comme bien souvent dans l'industrie IT, foisonnement des solutions rime rarement avec clarté conceptuelle. Pour cette raison, avant de nous risquer à caractériser ces systèmes, nous effectuerons dans un premier

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

temps un flashback dans l'histoire de l'informatique. Notre objectif sera de comprendre les raisons qui expliquent la domination, quasiment sans partage depuis quarante ans, de leurs prédécesseurs : les bases de données relationnelles (SGBDR)). Dans un second temps nous examinerons les limitations de ces SGBDR dans le contexte d'applications web à grande échelle. Le contexte ainsi posé, nous examinerons alors comment et à quel prix les systèmes NoSQL répondent aux exigences nouvelles.

De manière assez surprenante les fondements conceptuels des bases de données relationnelles furent essentiellement l'œuvre d'un seul individu : E. F. Codd (1923-2003) qui, en 1970, publia un article intitulé « *A Relational Model of Data for Large Shared Data Banks* ». Ignorés dans un premier temps par les équipes d'IBM, les travaux de Codd inspirèrent en revanche Larry Ellison qui fonda Oracle. La suite de l'histoire est connue. Le succès originel du modèle relationnel ne doit rien au hasard. En réalité, son origine est mathématique. Paraphrasant L. Kronecker, un mathématicien du XIX^{e} siècle qui forgea l'aphorisme célèbre « *God made the integers, all else is the work of man* », R. Ramakrishnan, l'auteur d'un manuel sur les bases de données relationnelles, proposa le jeu de mot (intraduisible) « *Codd made relations, all else is the work of man* ». Ce clin d'œil a pour but d'attirer l'attention sur ce qui fait la spécificité du modèle relationnel inventé par Codd. Son fondement n'est pas une technologie, inévitablement vouée à l'obsolescence, mais une série de théorèmes qui décrivent une structure mathématique intemporelle appelée **algèbre des relations**. Les principaux éléments de cette algèbre ne sont autres que les opérations bien connues des informaticiens, la *sélection*, la *projection*, *l'agrégation*, l'*union*, la *jointure*, etc. qui opèrent sur données structurées en tables pour en créer de nouvelles.

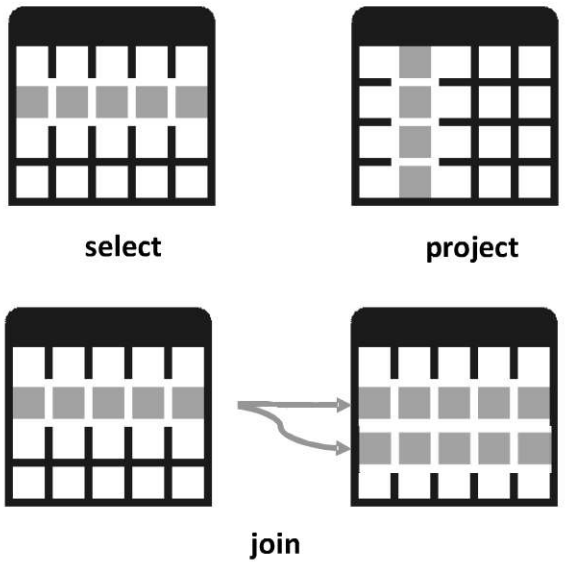


Figure 3.1 — Les principales opérations de l'algèbre des relations

La notion de jointure est particulièrement importante puisque c'est elle qui permet de relier entre eux des enregistrements situés dans des tables différentes et de le faire implicitement, au moyen de clés étrangères, plutôt qu'à l'aide de pointeurs explicites. Elle est au cœur des principes de normalisation des données et de la flexibilité du

Copyright © 2015 Dunod.

modèle relationnel. Utilisées en cascade, les jointures peuvent cependant rapidement s'avérer très gourmandes en ressources. Pour cette raison, leur usage sera remis en question par les systèmes NoSQL.

Qu'est-ce qu'un modèle de données ?

Un modèle de données est défini schématiquement par trois éléments :

Une structure : plus précisément la structure logique selon laquelle les données sont enregistrées sur disque. Dans le cas des SGBDR par exemple, il s'agit de tables constituées de lignes et de colonnes. Pour une base de données de type document il s'agit d'une collection de documents aux formats JSON ou XML indexés par une clé. Pour une base de données objet la structure est celle d'une collection de grappes et des hiérarchies d'objets, etc.

Des contraintes : elles définissent les données que l'on considère valides. À titre d'exemple, une contrainte pour un SGBDR est que tous les enregistrements possèdent impérativement les mêmes colonnes. Le schéma d'un SGBDR contient la description de toutes les contraintes, qu'il s'agisse de limites sur des valeurs numériques, de la syntaxe admise pour des chaînes de caractères ou d'intégrité des références.

Des opérations : elles définissent comment on lit les données, comment on les met à jour et comment les supprime. Dans le cas d'un SGBRD et à un haut niveau d'abstraction, elles sont définies par le langage de requêtes SQL. Au niveau de l'implémentation, il s'agira plutôt des opérations de l'algèbre relationnelle. Pour une base de données orientée graphe, une opération pourra consister à récupérer, par exemple, tous les parents d'un certain nœud.

Les règles de l'algèbre usuelle, celles qui permettent par exemple de transformer une formule faisant appel à quatre opérations élémentaires, sont utiles car elles permettent de simplifier une expression a priori complexe comme $x - (y/y) - x + 1$ en une autre beaucoup, plus simple, 0 en l'occurrence. De manière similaire, l'étude de l'algèbre des relations effectuées par Codd a démontré qu'il est possible de transformer une succession d'opérations en une autre, équivalente, non pour la simplifier, mais pour optimiser les ressources utilisées comme le temps de calcul et la quantité de mémoire utilisée. Les SGBDR prennent en charge cette tâche d'optimisation du plan d'exécution. Le problème est complexe et des années de R&D ont été investies par les éditeurs pour parvenir à des solutions à la fois robustes et performantes.

La valeur ajoutée d'un SGBDR ne se limite pas cependant à ce travail d'optimisation puisqu'un développeur n'a nullement à se soucier d'écrire des relations algébriques. Plus simplement, il utilise un **langage déclaratif** qui lui permet de spécifier la réponse qu'il cherche plutôt que la manière d'effectuer cette recherche. C'est le rôle du langage SQL devenu depuis lors un quasi standard dans l'industrie informatique.

```
SELECT DISTINCT c.nom FROM Clients c
WHERE Pays='France'
AND City='Paris'
```

Les SGBDR, enfin, gèrent les **transactions**. Une transaction correspond par exemple à un transfert d'argent entre deux comptes bancaires ou à la réservation d'un

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

billet d'avion et d'une chambre d'hôtel. Réalisée comme une succession d'opérations sur une ou plusieurs bases de données, son rôle est de garantir que, soit toutes les opérations qu'elle englobe sont effectuées avec succès, soit qu'aucune opération n'est effectuée. Les transactions, tout comme les contraintes d'intégrité imposées par le schéma du SGBDR, contribuent ainsi à garantir la cohérence des données hébergées.

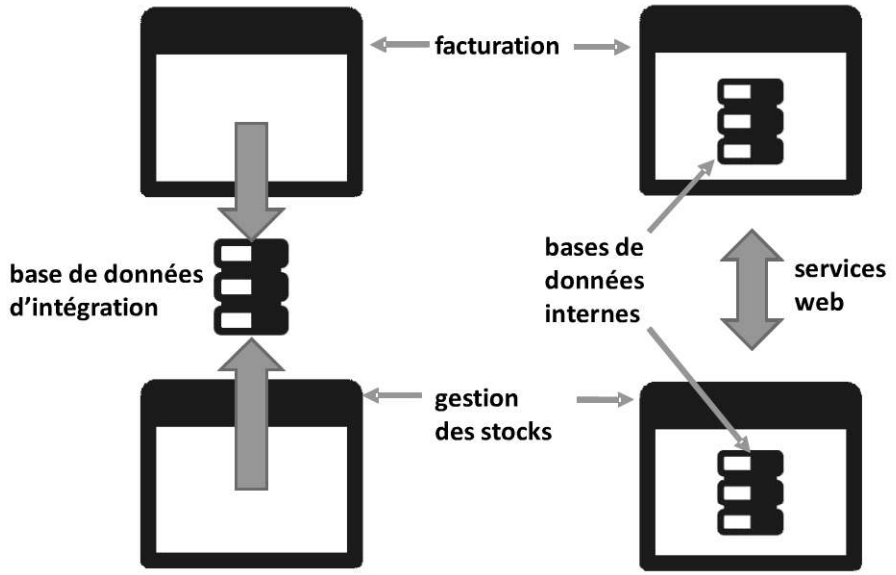


Figure 3.2 — L'expérience a montré que la démarche d'intégration traditionnelle par un SGBDR est souvent trop coûteuse. On considère aujourd'hui qu'il est généralement préférable d'intégrer des applications ou des services, qui contiennent un SGBDR, au moyen d'un ESB par exemple.

Les exigences qui caractérisent une transaction SGBDR sont définies par le célèbre acronyme **ACID**. Rappelons en brièvement la signification :

- **A**tomicité : désigne le caractère indivisible d'une transaction évoqué plus haut.
- **C**ohérence : désigne l'objectif même des transactions, à savoir laisser les données dans un état cohérent.
- **I**solation : deux transactions simultanées A et B n'interfèrent jamais. La transaction B ne peut ni voir ni modifier les données sur lesquelles A opère tant que A n'a pas été validé.
- **D**urabilité : une fois une transaction validée, les données doivent être permanentes.

En résumé – Les mécanismes de schémas et de transactions assurent pour une large part la cohérence des données au sein d'un SGBDR. Par ailleurs l'existence d'un langage déclaratif standard comme SQL et l'optimisation automatique des plans d'exécution permettent un découplage logique maximal entre les données et les applications qui les utilisent. Cette flexibilité permet d'utiliser les données hébergées par un SGBDR de différentes façons et permet même d'envisager des usages ultérieurs qui n'avaient pas été anticipés à l'origine.

Copyright © 2015 Dunod.

Ces facteurs à eux seuls ne suffisent pas cependant à expliquer le demi-siècle d'hégémonie des SGBDR. Des facteurs de politique interne aux DSI entrent aussi en considération. Le fait que les SGBDR soient progressivement devenus un mécanisme d'intégration pour une majorité de DSI est l'un de ces facteurs. Remplacer un système comme un SGBDR auquel de lourds investissements ont été consentis est dès lors devenu de plus en plus problématique. De ce point de vue, il n'est pas exagéré d'affirmer que la domination des SGBDR a probablement entravé l'avènement d'autres solutions parfois plus rationnelles et plus performantes.

Donnons un exemple. Les SGBDR sont notoirement mal adaptés lorsqu'il s'agit de sauvegarder et de récupérer des grappes d'objets tels que celles qui sont manipulées par des langages objet comme Java ou C#. C'est le célèbre problème du « mismatch d'impédance », illustré par la figure 3.3. Alors que les bases de données objets apparues au milieu des années 1990 promettaient d'éradiquer ce genre de misères, l'industrie IT a préféré développer des frameworks d'un maniement souvent complexe, comme *Hibernate*, pour palier ces inconvénients. Sans même parler du coût de l'inélégance de ces solutions bâtardes, leur coût en complexité et en abstraction a finalement été jugé moins élevé que la migration complète d'un SGBDR.

Les SGBDR, assurément, ont la peau dure ! Ni les bases de données objets, plus performantes, ni les sommes astronomiques englouties dans des projets d'intégration sans fin n'ont eu raison d'elles. Le coup de boutoir viendra d'ailleurs. Au début des années 2000, les applications web à très grande échelle et l'émergence des réseaux sociaux ont fini par changer la donne.

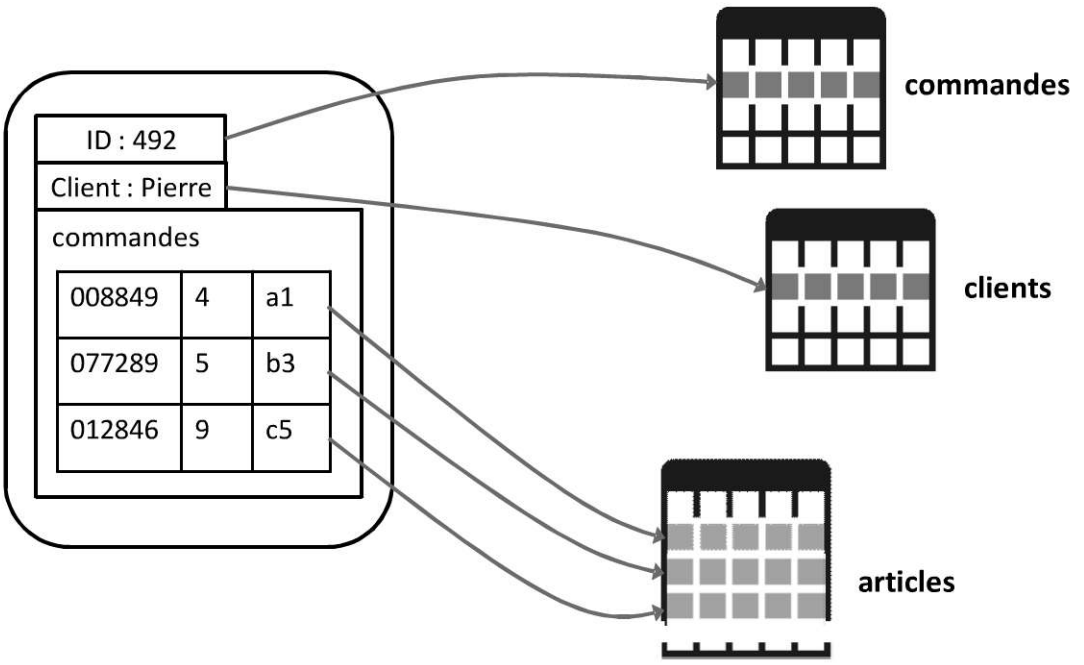


Figure 3.3 — Transférer des données entre une grappe d'objets et des tables d'un SGBDR est une opération complexe car les données agrégées dans un objet ou une grappe d'objets sont généralement disséminées dans plusieurs tables et certaines notions du monde objet, comme l'héritage, n'ont pas d'équivalent naturel dans le monde des SGBDR.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

3.2 LE DOGME REMIS EN QUESTION

3.2.1 Les contraintes des applications web à très grande échelle

Avec ces nouveaux systèmes, sont apparus de nouveaux besoins métiers et de nouvelles exigences techniques. Via l'isolation des transactions et l'application de contraintes d'intégrité les SGBDR garantissent des données avec un haut niveau de cohérence. Dans les applications e-business à grande échelle comme Amazon ou dans les réseaux sociaux comme Twitter, les priorités sont cependant très différentes. Pour une entreprise comme Amazon, par exemple, une plateforme indisponible, ne serait-ce que quelques minutes, est inenvisageable car elle causerait la perte de dizaine de milliers de commandes. Priorité absolue est donc donnée aux performances et à la disponibilité. Celles-ci sont assurées aujourd'hui par la mise en œuvre d'architectures distribuées sur des centaines voir des milliers de machines. Dans un tel contexte, parvenir à parfaite intégrité des données et une stricte isolation des transactions est extrêmement difficile et coûteux. Le volume des échanges d'information nécessaires pour parvenir à une synchronisation parfaite des nœuds d'un SGBDR distribué limite en effet le nombre de serveurs à quelques dizaines, tout au plus. De manière encore plus significative encore, l'ancienne exigence de cohérence des données ne correspond souvent plus aux priorités du business. Qui se soucie vraiment que la liste des commentaires d'un post sur un réseau social soit vue de la même manière au même instant par tous les utilisateurs ?

Naturellement l'inconsistance des données pourra avoir des conséquences qu'il faudra traiter le moment venu. Une même chambre a-t-elle été réservée par deux clients sur Booking.com ? Un même article a-t-il été commandé sur Amazon par plusieurs clients à qui l'on a indiqué qu'il était disponible alors qu'il ne restait plus qu'un exemplaire ? Dans chacune de ces situations la solution est d'ordinaire humaine et pragmatique. Dans le cas d'Amazon par exemple, les clients lésés se verront contactés par le service client qui, pour atténuer leur courroux, leur présentera un mot d'excuse assorti d'un chèque à hauteur du préjudice subi. Quant au surbooking pour des billets d'avion ou des chambres d'hôtel, on peut imaginer des partenariats avec d'autres compagnies aériennes ou un quota de chambres réservé pour des situations d'urgence. Quoi qu'il en soit, la solution dans ce genre de situations n'est pas technique. C'est aux métiers de prendre une décision éclairée basée sur une analyse comparative entre les coûts des inconsistances qu'il faudra inévitablement gérer avec les coûts d'une indisponibilité chronique d'un système pénalisé par l'exigence d'une cohérence de tous les instants.

On retrouve ici la distinction usuelle entre l'**approche pessimiste** de la gestion des conflits, qui cherche à les éviter au moyen de verrous, et l'**approche optimiste**, qui cherche à détecter les conflits et à les traiter au moment où ils surviennent, le cas échéant manuellement.

Les remarques qui précèdent n'impliquent pas pour autant que toute notion de cohérence soit désormais caduque. Dans les faits, l'exigence de cohérence est le plus souvent remplacée par une contrainte plus faible dite de **cohérence à terme** (ou *eventual consistency* en anglais).

Copyright © 2015 Dunod.

Par **cohérence à terme** (*eventual consistency*) on entend le fait qu'un système distribué converge à longue échéance vers un état où tout utilisateur verra chaque donnée dans l'état où l'a laissée la dernière mise à jour. Une propriété qui contraste avec la consistance permanente prônée par les systèmes qui vérifient les propriétés ACID de manière stricte.

Un nouvel acronyme, sous forme de jeu de mot un peu alambiqué, a été forgé pour caractériser ces systèmes : **BASE** pour *Basically Available Soft State Eventual Consistency*.

Aux considérations précédentes sur la disponibilité des données, il convient d'ajouter encore leur caractère souvent non structuré dans un contexte Big Data, souvenons-nous du V = « variété ». Lui aussi contribue à une mise en cause du modèle de données structurées des SGBDR. Des informations précieuses ne demandant qu'à être analysées sont aujourd'hui tapies dans d'obscurs fichiers logs, des feuilles Excel, des mails ou même des fichiers multimédias.

3.2.2 Le « théorème » CAP

Même si les termes de disponibilité et de cohérence n'ont pas été définis avec une grande rigueur, la discussion qui précède suggère qu'il est possible de choisir, selon les circonstances, entre ces deux caractéristiques. Lorsqu'un système est distribué il convient de prendre en compte une troisième caractéristique : la tolérance aux partitions accidentelles.

Cohérence – Disponibilité – Résistance au morcellement

Sur un plan intuitif, une base de données distribuée devrait idéalement posséder les trois caractéristiques suivantes :

– **Cohérence** (*consistency*) : chaque donnée est présente dans la même version sur tous les nœuds du réseau. Il n'y a pas de retard dans la propagation des mises à jour.

– **Disponibilité** (*availability*) : l'application est accessible à tout instant et chaque requête reçoit une réponse qui confirme si elle a été traitée avec succès ou non.

– **Résistance au morcellement** (*partition tolerance*) : le système doit pouvoir continuer à fonctionner lorsque différents nœuds sont isolés consécutivement à une rupture du réseau.

En 2000, lors d'un symposium consacré au calcul distribué à l'université de Berkeley, l'un des participants, Eric Brewer, proposa une conjecture qui stipulait que parmi les trois caractéristiques précédentes, seules deux pouvaient être réalisées simultanément. Bien qu'aucun des trois termes invoqués dans cette conjecture n'ait été défini avec la précision qu'exigerait une formulation mathématique, l'expression « **théorème CAP** » est restée. Des tentatives de formulations plus précises et des démonstrations partielles ont bien été proposées mais aucune ne répond aux critères d'un authentique théorème. Le « théorème » CAP est donc à considérer comme une intuition utile, en partie folklorique, concernant les systèmes distribués.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

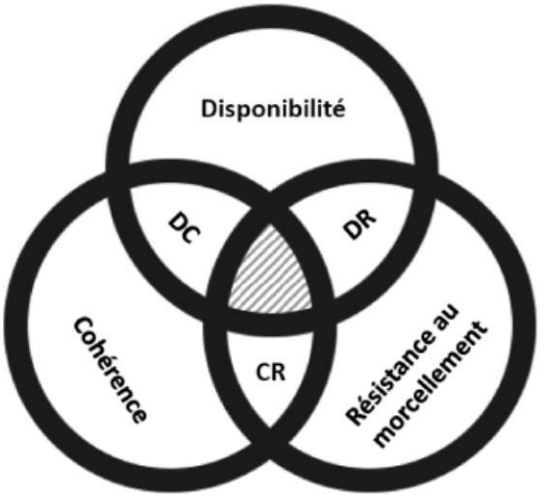


Figure 3.4 — Le « théorème » CAP dans sa formulation originale suggère que dans un système distribué seules deux des trois caractéristiques souhaitables mentionnées peuvent être réalisées simultanément.

L'intuition qui se cache derrière cette affirmation est en réalité assez simple à saisir. Imaginons pour cela un processus de réservation d'un billet d'avion qui implique simultanément deux clients. Imaginons par ailleurs que le système de réservation soit distribué sur plusieurs nœuds et que les nœuds qui traitent les deux requêtes soient distants. En cas de rupture de connexion entre les deux serveurs l'alternative est alors la suivante :

- soit l'on exige une stricte cohérence des données auquel cas il faudra attendre le rétablissement de la connexion ;
- soit l'on exige une disponibilité du système à tout instant et dans ce cas il faudra réconcilier les données incohérentes ultérieurement et assumer les risques de cet ajustement différé.

Dans la réalité le compromis est plus subtil que ne le laisse croire l'affirmation strictement binaire du « théorème CAP ». La cohérence et la disponibilité, ou plus précisément le temps de latence, sont toutes deux des variables continues que l'on peut ajuster selon les besoins. En fonction des priorités, ce choix pourra d'ailleurs être adapté au type d'opération effectuée. Cependant, quoi qu'il en soit, disposer d'un haut niveau de cohérence parmi les données répliquées implique forcément un échange d'information important entre ces nœuds, ce qui occasionnera un temps de latence accru.

Dans une approche plus fine de la notion de cohérence des données il faudrait aussi spécifier le niveau de granularité des données sur lesquelles opèrent les transactions. Là encore il ne s'agit pas d'une variable binaire. En effet, pour prémunir un système des modifications intempestives de deux transactions concurrentes qui risqueraient de le laisser dans un état incohérent, la première transaction peut poser un des verrous sur les données qu'elle manipule. Selon le degré d'isolation souhaité, ces verrous

Copyright © 2015 Dunod.

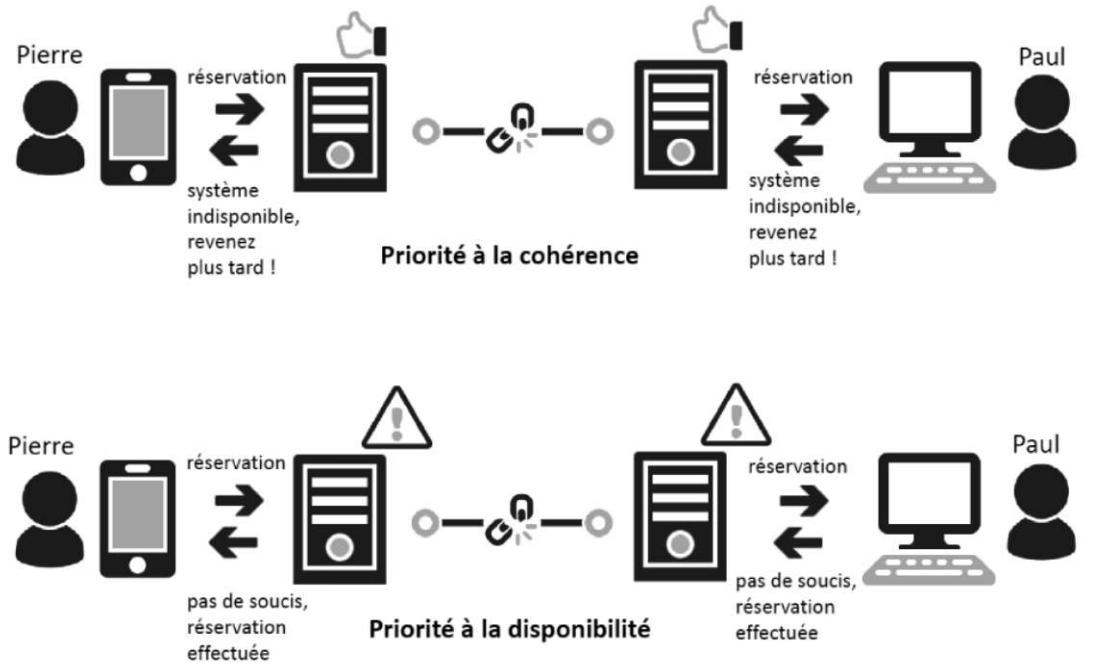


Figure 3.5 — Lors d'une réservation effectuée sur un système distribué il faut choisir entre un système disponible, au risque de créer des incohérences de données, et un système dont les données sont cohérentes au risque cette fois de créer de l'attente et de perdre des clients.

pourront affecter soit toutes les tables concernées par une transaction, soit tous les enregistrements d'une table en particulier ou seulement certains agrégats dont on souhaite à tout prix préserver la cohérence. Dans la section suivante, lorsque nous décrirons les différents types de systèmes NoSQL, nous indiquerons à chaque fois en quoi consiste, lorsqu'il existe, l'agrégat atomique. Au-delà, la cohérence devra être assurée par l'application cliente ou, comme on l'a vu, par une intervention humaine.

3.2.3 Sacrifier la flexibilité pour la vélocité

Un haut niveau de cohérence des données pénalise les performances ou la disponibilité d'une base de données, telle était l'une des conclusions du paragraphe précédent. De manière analogue, la nouvelle donne du Big Data suggère de redéfinir les compromis entre flexibilité dans l'usage des données et performances.

L'un des grands atouts des SGBDR est le découplage logique qu'ils autorisent entre les applications clientes et la structure des données que celles-ci utilisent. Tous les usages sont en principe possibles, puisque chaque requête SQL est automatiquement convertie par le SGBDR en plan d'exécution. Cette flexibilité a toutefois un coût, car un schéma de données n'est évidemment optimal que pour certaines requêtes.

Dans les contextes Big Data où la vélocité prime souvent sur toute autre considération, ce postulat de flexibilité des SGBDR, devra donc lui aussi remis en question. Un changement de priorité qui n'est évidemment pas sans conséquences, puisque c'est désormais au développeur d'une application d'anticiper les usages qu'il fera des

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

données. Il devra concevoir un code optimisé pour la structure de données spécifique à l'application qu'il développe. Tout changement ultérieur du modèle de données nécessitera une révision du code.

Dans le même ordre d'idée, examinons l'utilité des jointures entre tables. Étroitement liées au modèle de données relationnel dont elles assurent la flexibilité et la non-redondance, celles-ci ne sont plus perçues désormais comme des opérations incontournables. Pour le comprendre, considérons l'exemple type d'un client auquel on souhaite associer ses commandes. Rien n'empêche en effet de **co-localiser les données** du client et celles de ses commandes au sein d'une même table. Il suffit pour cela d'insérer dans l'enregistrement du client des groupes de colonnes contenant ses commandes. Le prix à payer est double :

- Le **chemin d'accès aux données devient alors essentiellement unique**. Il sera possible d'accéder aisément aux commandes à partir des données clients. Le chemin inverse, à savoir trouver les clients ayant commandé un certain produit, ne sera cependant plus disponible. Plus précisément, il sera excessivement inefficace car il faudra examiner tous les clients, un par un.
- Il faut parfois **accepter une part de redondance** dans les données enregistrées. Des commandes identiques émises par des clients différents seront inévitablement dupliquées dans un tel schéma là où le mécanisme de clés étrangères des SGBDR permettait précisément d'éviter cette redondance.

ID_client	nom	prénom	commande_1		commande_2		commande_3	
			produit	quantité	produit	quantité	produit	quantité

Figure 3.6 — La colocation des données impose de choisir un chemin d'accès aux données. Récupérer la liste des produits commandés par un client est aisé mais récupérer la liste des clients qui ont commandé un produit exige de parcourir toute la table.

3.2.4 Peut-on définir ce qu'est une base de données NoSQL ?

Jusque-là nous n'avons proposé aucune définition des bases de données NoSQL, et pour cause, car celle-ci s'avère particulièrement délicate. Aucun concept clair en effet ne les caractérise vraiment et le sigle NoSQL suggère tout au plus, ce qu'elles ne sont pas. Apparues ces dernières années dans le contexte d'application e-commerce à grande échelle, les priorités auxquelles elles répondent sont les suivantes :

- **Distribuer les traitements et le stockage** sur des centaines voire des milliers de nœuds constitués de serveurs banalisés.
- **Donner la priorité aux performances et à la disponibilité** sur l'intégrité des données.

Copyright © 2015 Dunod.

- **Traiter efficacement les données non structurées** ou seulement partiellement structurées.

Bien qu'une définition honnête soit impossible à notre avis il est possible néanmoins d'identifier un certain nombre de points communs à ces systèmes :

1. Ces systèmes sont le plus souvent **clusterisables** et permettent une montée en charge approximativement linéaire. En d'autres termes, un doublement du nombre de serveurs permet, grosso modo, de traiter deux fois plus de requêtes dans un même laps de temps.
2. Ils sont en règle générale **dépourvus de schémas**, inadaptés aux données non structurées. Cette caractéristique leur confère d'ailleurs un atout significatif vis-à-vis des SGBDR : ils permettent une grande rapidité de développement précisément à cause de cette simplicité. De ce fait ils sont particulièrement adaptés aux méthodes de développement agiles.
3. Ils sont souvent **dépourvus de transactions** au sens habituel du terme, ou alors proposent des transactions qui garantissent seulement l'intégrité de certains agrégats de données naturels. Nous y reviendrons dans la prochaine section consacrée à une classification de ces systèmes.
4. Ils sont **non relationnels** dans le sens où ils n'offrent pas de jointures.
5. Beaucoup de ces systèmes sont aujourd'hui proposés en **open source**.

Que dire alors du sigle « NoSQL » ? Le moins que l'on puisse dire est que cette dénomination est malheureuse. D'une part, aux dires de certains, le « *No* » ne voudrait pas forcément dire « *no* », il pourrait signifier aussi « *not only* ». La partie « SQL » est plus trompeuse encore puisque ce n'est pas tant l'absence du langage SQL qui est significative pour beaucoup de ces systèmes, mais plutôt l'absence de transactions au sens usuel du terme. En réalité ce sigle n'a pas été choisi suite à de profondes réflexions conceptuelles. Il s'agit d'un simple hashtag utilisé en 2009 pour notifier sur Twitter l'organisation d'un débat à San Fransisco sur ces bases de données d'un nouveau genre.

3.3 LES DIFFÉRENTES CATÉGORIES DE SOLUTIONS

Mettre un peu d'ordre dans le foisonnement de systèmes de bases de données auxquels on peut accoler l'étiquette NoSQL n'est pas une chose aisée car proposer une catégorisation revient, de fait, à définir des cloisons étanches qui ne correspondent ni à la réalité technologique, ni à celle des usages. Au plus haut niveau de granularité on peut cependant distinguer deux catégories vraiment disjointes.

La première est constituée de ce que l'on pourrait appeler les **bases de données orientées agrégats** (BDOA), un terme dû à Martin Fowler[1]. L'un des facteurs qui a déterminé l'émergence des systèmes NoSQL est la nécessité, on l'a vu, de déployer

1. *NoSQL Distilled - A Brief Guide to the Emerging World of Polyglot Persistence*, Pramod J. Sadalage et Martin Fowler, Addison Wesley, 2012.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

les systèmes sur des clusters. L'une des stratégies pour y parvenir est de renoncer à éclater les données dans une multitude de tables et de les regrouper plutôt au sein d'agrégats qui contiennent les données auxquelles l'on accède le plus souvent. Le « mismatch d'impédance » disparaît dans ce cas et les agrégats fonctionnent alors à la fois comme unités pour les opérations de mise à jour et comme frontières naturelles pour les transactions. Enfin les différentes stratégies de distributions, de réplication et de partitionnement horizontal (*sharding*) utilisent elles aussi ces agrégats comme unités de base.

Parmi ces BDOA nous distinguerons trois sous-catégories, selon la structure des agrégats qu'elles manipulent :

- les entrepôts clé-valeur,
- les bases de données orientées documents,
- et enfin les bases de données orientées colonnes.

La seconde grande catégorie est celle des **bases de données orientées graphes** (BDOG) conçues pour naviguer efficacement dans un graphe de données, une tâche pour lesquelles, les SGBDR sont paradoxalement inadaptés.

Dans les quatre sections suivantes nous décrirons chacun de ces quatre modèles. Dans la mesure du possible nous établirons des parallèles avec les concepts plus familiers des SGBDR et nous examinerons en particulier dans quelle mesure ils gèrent les transactions. Enfin nous donnerons à chaque fois les principaux cas d'usage.

3.3.1 Les entrepôts clé-valeur

Concepts

Un entrepôt clé-valeur (ECV) peut être envisagé comme une collection de tables de hachage persistantes c'est-à-dire comme une collection de couples clé-valeur persistées sur disque. La valeur en question peut être un morceau d'information (un *blob*) sans aucune structure a priori. Il peut s'agit d'un nombre, d'un fichier XML, d'une vidéo, d'une grappe d'objets métiers sérialisés ou encore d'un fichier texte sans structure particulière.

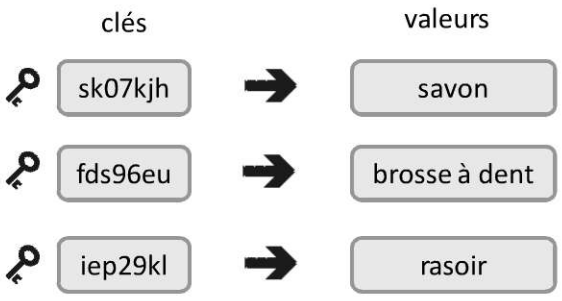


Figure 3.7 — Parmi toute la panoplie des systèmes NoSQL, l'entrepôt clé-valeur est le système le plus rudimentaire. Il s'agit d'une simple collection de couples clé-valeur enregistrée sur le disque.

Copyright © 2015 Dunod.

Les opérations que l'on peut effectuer sur un tel entrepôt sont : la récupération de la valeur pour une clé donnée, la mise à jour, la création ou la suppression d'une valeur pour une certaine clé. La base ne « connaît » pas la structure des données, charge étant à l'application cliente d'interpréter le contenu des valeurs. La contrepartie d'une telle rusticité qui n'autorise l'accès aux données que par le biais d'une clé primaire est la performance de ces systèmes.

Dans un tel contexte, stocker toutes les données dans un seul espace est généralement considéré comme une mauvaise pratique car, chaque agrégat ayant potentiellement une structure différente, les risques de conflits entre clés augmentent. Un usage typique d'un ECV est celui du stockage des sessions de multiples utilisateurs.

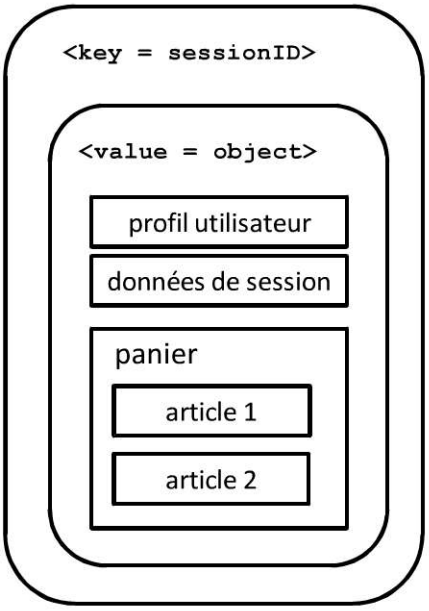


Figure 3.8 — Un identifiant de session fonctionne comme une clé primaire pour désigner la grappe d'objets associée à une session utilisateur, en l'occurrence un profil utilisateur, de données de session ainsi qu'une liste d'articles.

Certains entrepôts clé-valeur proposent pour cette raison de partitionner l'espace des clés en différentes zones dédiées chacune à un type de données spécifique, facilitant ainsi la sérialisation et la dé-sérialisation par le client de l'ECV.

Le modèle de cohérence à terme (*eventual consistance*) est celui qui est retenu par une majorité des implémentations d'ECV. Le paramétrage fin de la consistance peut alors se faire au moyen de trois paramètres :

- Le **facteur de réplication** qui définit le nombre *N* de nœuds sur lesquels doivent être répliquées les données.
- Le **quorum d'écriture** *W* qui spécifie le nombre de nœuds dont on exige qu'ils notifient que les données ont été écrites avec succès pour considérer une opération d'écriture comme valide.
- Le **quorum de lecture** *R* enfin, qui spécifie le nombre de nœuds minimal qu'il convient d'interroger pour être sûr d'obtenir la dernière version d'une donnée.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

On parle de *consistance forte* lorsque $R + W > N$.

La montée en charge des ECV est généralement assurée par un mécanisme dit de *sharding* (ou de partition horizontale). Alors que la réplication consiste à enregistrer des copies d'une donnée sur plusieurs nœuds, le sharding consiste à répartir des données différentes sur des nœuds différents. Dans le contexte d'un ECV, c'est la valeur de la clé qui déterminera sur quel serveur est hébergée la valeur associée.

Les implémentations les plus répandues des entrepôts clé-valeur sont *Riak*, *Redis*, *Memcached DB* et *DynamoDB* chez Amazon.

Usages

Le stockage de données de session utilisateur est le cas typique d'utilisation d'un entrepôt clé-valeur. De manière similaire, le stockage des préférences ou des profils des utilisateurs ou encore les associations entre clients et paniers d'achat des sites d'e-commerce conviennent bien aux ECV. La structure des données en table de hachage les rend cependant inadaptés pour retrouver des données à partir d'une valeur plutôt que d'une clé.

3.3.2 Les bases orientées documents

Concepts

Sur un plan conceptuel, les bases de données orientées document (BDOD) diffèrent peu des ECV. Une BDOD peut schématiquement se décrire comme un ECV dont les valeurs sont des documents semi-structurés. Par ce terme on entend des documents auto-descriptifs généralement écrits avec un format de type XML, JSON ou similaire. Par contraste avec les SGBDR qui exigent que chaque enregistrement d'une table ait les mêmes colonnes, spécifiées par un schéma, rien n'exige que les documents stockés dans une BDOD aient tous le même format.

De ce point de vue, une BDOD est donc beaucoup plus flexible qu'un SGBDR dans la mesure où il est possible de modifier au coup par coup la structure d'un document sans avoir à respecter un schéma préétabli. Alors qu'une valeur non définie d'une colonne d'un enregistrement d'un SGBDR est représentée par la valeur NULL, un document d'une BDOD ne fera tout simplement pas figurer l'attribut en question.

Pour clarifier la relation entre SGBDR (*Oracle*) et BDOD (*MongoDB*), le tableau 3.1 présente une liste d'analogies utiles.

Les BDOD offrent par ailleurs la possibilité d'examiner le contenu d'un document sans qu'il soit nécessaire pour autant de le récupérer en intégralité, en quoi elles se distinguent des ECV. Rappelons que c'est à l'application cliente d'un ECV d'interpréter le contenu d'une valeur.

Dans une BDOD chaque document constitue un élément atomique qui délimite une frontière transactionnelle naturelle. En règle générale il n'est pas possible d'impliquer plusieurs documents comme le ferait un SGBDR sur différents enregistrements.

Copyright © 2015 Dunod.



Figure 3.9 — Une base orientée document.
Les documents peuvent chacun avoir leur propre structure.

Tableau 3.1 — Équivalences entre SGBDR et BDOD

BDOD	SGBDR
base de données	schéma
collection (de documents)	table
document	enregistrement
Id de document	Id d'enregistrement
DBRef (référence entre documents)	jointure (entre tables)

La disponibilité des BDOD est généralement assurée par un mécanisme de réplication qui utilise un schéma de type maître-esclave. Les requêtes sont adressées par défaut au maître qui les redirige vers l'un des esclaves pour les opérations de lecture. La montée en charge se fait par simple ajout de nœuds esclaves. Si le maître tombe en panne, les nœuds esclaves procèdent à un vote pour élire un nouveau maître selon des critères de distance entre les nœuds ou de quantité de RAM disponible. L'application cliente n'a pas, quant à elle, à se soucier de se connecter à un nœud particulier ni à gérer les pannes de réseau. La technique de sharding est souvent utilisée par les BDOD, le système assurant une répartition équilibrée des données entre les nœuds en tenant compte notamment de la distance qui les sépare.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Les implémentations les plus répandues de BDOD à ce jour sont : *MongoDB*, *CouchDB*, *RavenDB* et *Lotus Notes*.

Usages

Les applications d'e-commerce dont les produits varient trop souvent pour être décrits au moyen d'un schéma stable, bénéficieront de l'usage d'une BDOD.

Les applications qui manipulent naturellement des documents comme les systèmes de gestion de contenu ou les plateformes de blogs pourront elles aussi utiliser avec profit une BDOD.

Mentionnons encore les systèmes de logs qui, par définition, ont à stocker des associations entre des évènements et des contenus sans structure prédéfinie.

3.3.3 Les bases orientées colonnes

Concepts

Parmi les bases de données orientées agrégats (BDOA), les bases orientées colonnes (BDOC) sont sans aucun doute celles dont le modèle de données est le plus riche. De fait, la principale difficulté pour aborder ce sujet tient moins à sa complexité intrinsèque qu'à la barrière d'une nomenclature hélas souvent confuse et qui, de surcroît, varie d'un produit à un autre. Pour cette raison, nous décrirons ici le modèle de données de la BDOC *Cassandra*, son modèle étant particulièrement complet et clair. D'autres BDOC, comme *HBase*, pourront alors être envisagées, tout au moins sur un plan conceptuel, comme des cas particuliers de celui que nous présentons.

Pour tenter d'y voir clair, établissons d'emblée le parallèle avec deux catégories de systèmes qui nous sont d'ores et déjà familiers : les ECV et les SGBDR classiques.

La première manière de définir, sur le plan conceptuel, une BDOC comme *Cassandra* est qu'il s'agit d'un ECV dont les valeurs possèdent une structure bien particulière. Cette structure en l'occurrence est celle de colonnes dont les noms peuvent être soit *statiques*, ils sont alors partagés par tous les enregistrements d'une collection de colonnes, soit *dynamiques*, c'est-à-dire qu'ils peuvent varier d'un enregistrement à l'autre au sein d'une telle collection.

La possibilité d'avoir des colonnes dynamiques permet d'avoir des enregistrements avec colonnes différentes en type et en nombre, chose impossible dans un SGBDR. Un autre élément des structures, commun à beaucoup de BDOC, est la possibilité de regrouper entre elles des colonnes contenant des données sémantiquement liées. L'enregistrement d'un client contiendra toutes ses commandes par exemple. Ces regroupements s'appellent des **super-colonnes**. Elles aussi peuvent à leur tour être statiques ou dynamiques. Selon que les colonnes ou les super-colonnes sont statiques ou dynamiques on peut alors construire quatre types de structures, que l'on appellera des **familles de colonnes**, appropriées à des situations différentes. Les quatre figures qui suivent illustrent chacune de ces structures.

Copyright © 2015 Dunod.

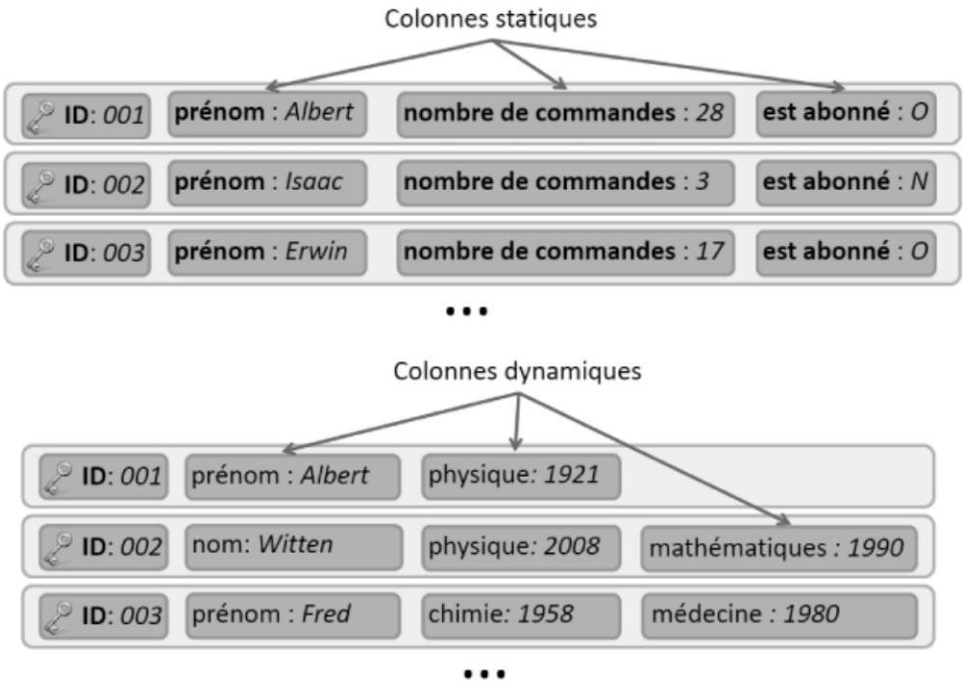


Figure 3.10 — Les colonnes d'une BDOC peuvent être statiques ou dynamiques. Dans le cas statique, on retrouve la structure habituelle d'une table d'un SGBDR, avec des enregistrements possédant un nombre fixe de colonnes prédéfinies. Les données d'un enregistrement qui utilisent des colonnes dynamiques peuvent aussi se concevoir comme une hashmap qui associe des noms (de colonne) à des valeurs.

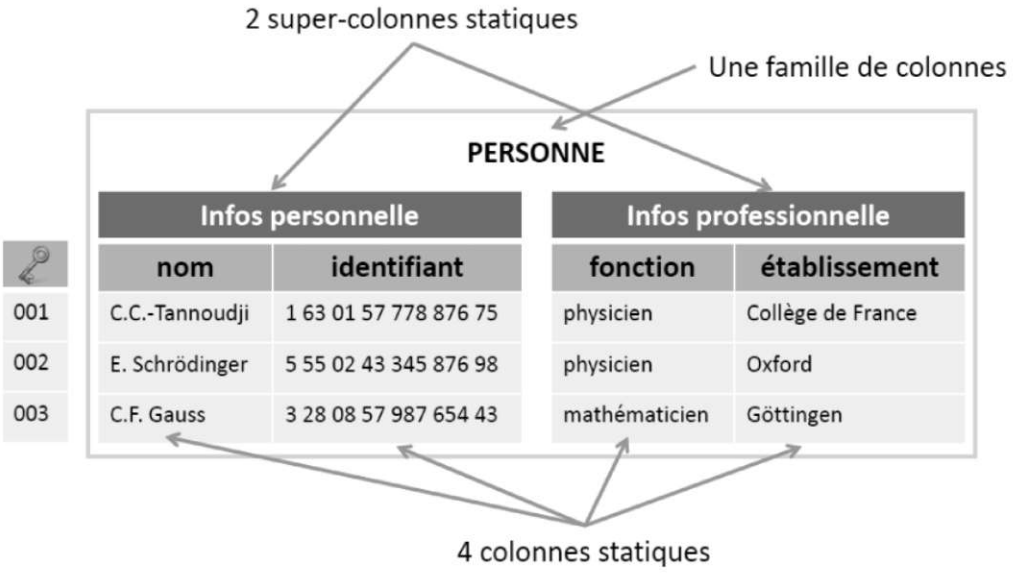


Figure 3.11 — Lorsque les super-colonnes et les colonnes sont statiques la structure de la famille de colonnes est celle d'une table de SGBDR classique.

Copyright © 2015 Dunod.

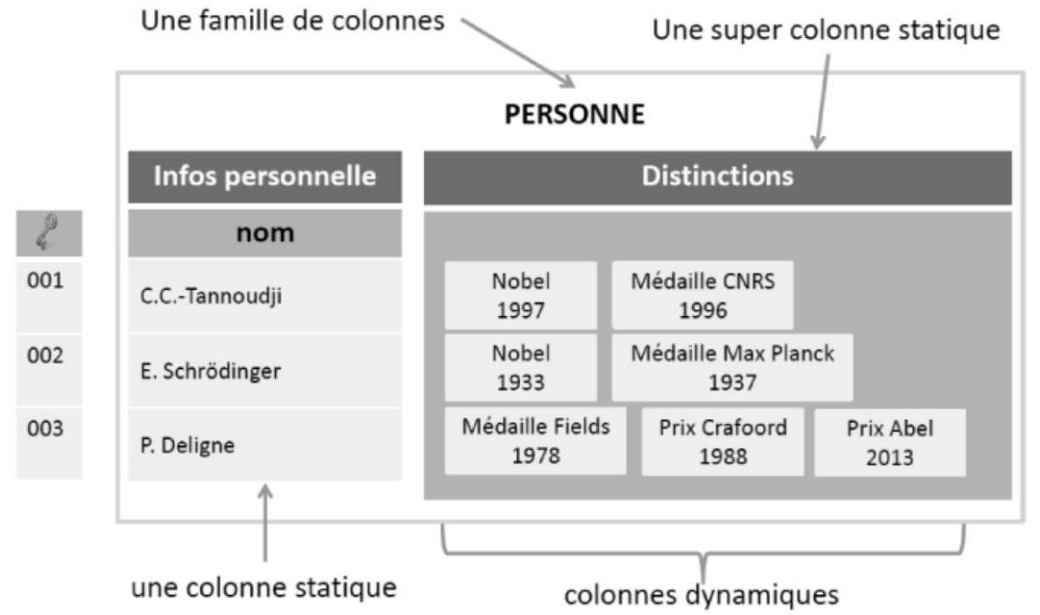


Figure 3.12 — Lorsque les super-colonnes sont statiques et les colonnes dynamiques, ces dernières permettent de représenter une hashmap associée à chaque clé.

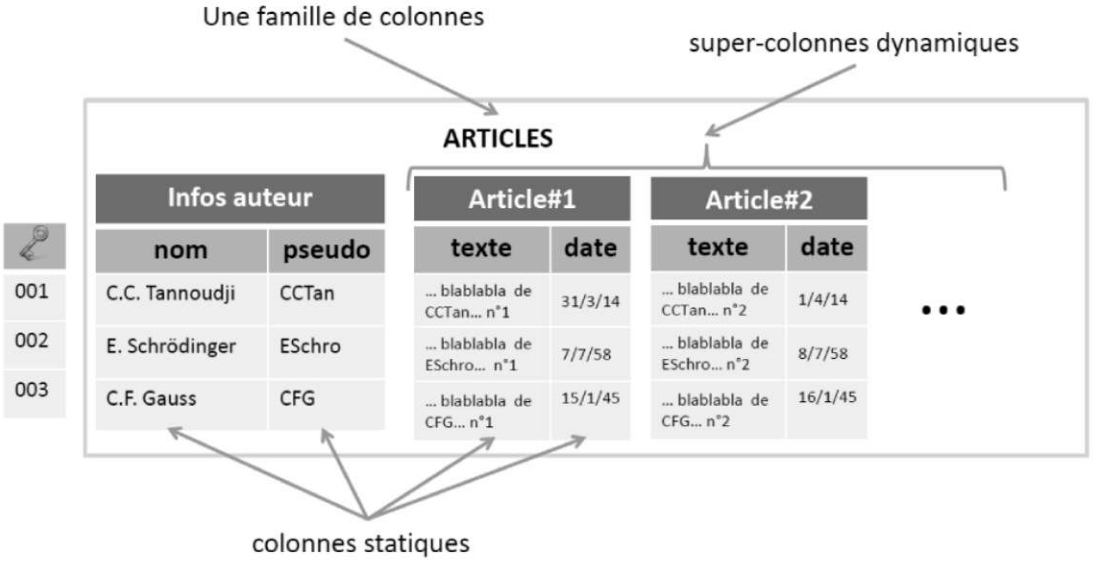


Figure 3.13 — Des super-colonnes dynamiques à l'intérieur desquelles figurent des colonnes statiques permettent de stocker, par exemple, des commentaires d'un article de blog.

Comme on le constate, les familles de colonnes autorisent une grande flexibilité et sont par conséquent adaptées à une grande variété d'usages. Pour clarifier définitivement le sujet, les analogies les plus pertinentes avec un SGBDR sont résumées dans le tableau 3.2.

Copyright © 2015 Dunod.

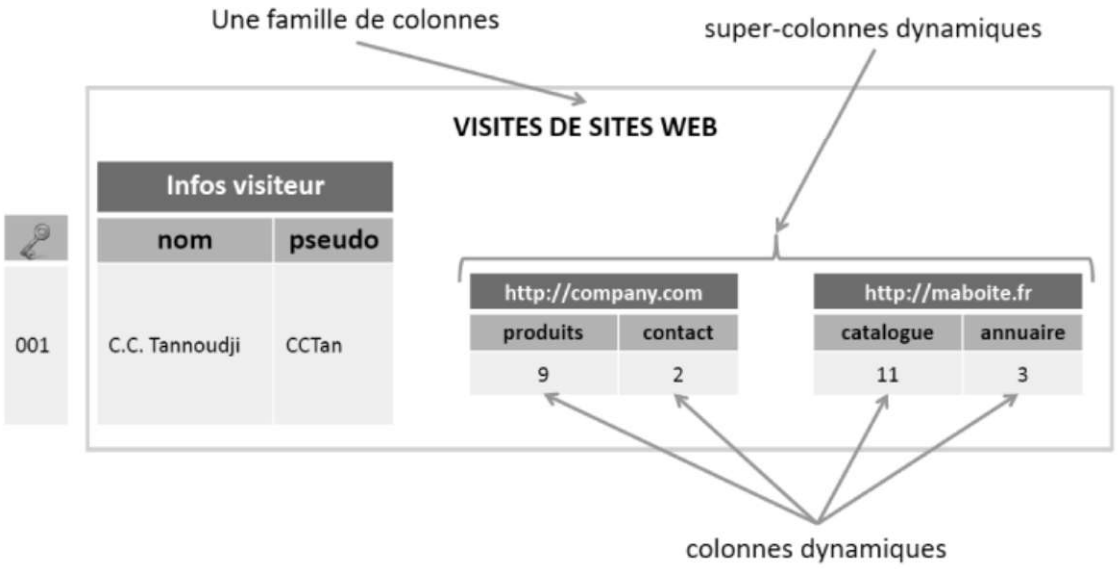


Figure 3.14 — La forme la plus flexible correspond à la situation où les super-colonnes et les colonnes sont toutes deux dynamiques. Un cas d'utilisation possible est le stockage des nombres de visites d'un site web, classés par page dans le site. Une autre manière de voir cette structure est que pour chaque enregistrement on a deux hashmap imbriquées. Dans l'exemple, la première associe une URL à une collection de pages du site visité. La seconde associe une page à un nombre de visites.

Tableau 3.2 — Les analogies les plus pertinentes entre BDOC et SGBDR

BDOC (Cassandra)	SGBDR
cluster	instance de base de données
keyspace	base de données
famille de colonnes	table
enregistrement	enregistrement
colonnes qui peuvent varier pour différents enregistrements si elles sont dynamiques	colonnes qui sont les mêmes dans tous les enregistrements d'une table

Comme pour les BDOD, il est généralement possible de paramétrer le niveau de cohérence souhaité, en lecture ou en écriture. En écriture par exemple, Cassandra propose les niveaux suivant :

- **ONE** : permet d'optimiser la performance. Il se peut que certaines opérations d'écriture soient perdues si le nœud auquel on envoie la requête tombe.
- **QUORUM** : on exige qu'une opération d'écriture se propage à une majorité de nœuds du cluster avant d'être validée.
- **ALL** : on exige que tous les nœuds aient répondu positivement pour qu'une opération d'écriture (ou de lecture) soit validée.

Le nombre de répliques peut être choisi lors de la création du keyspace.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

S'agissant des transactions, c'est l'enregistrement qui fait figure de frontière naturelle. Autrement dit une opération d'insertion ou de mise à jour est atomique au niveau d'un enregistrement mais pas au-delà.

La disponibilité est assurée par un mécanisme de réplication dans un mode pair-à-pair sans nœud maître.

Les BDOC les plus populaires à l'heure actuelle sont : *Cassandra* (dont nous avons repris les concepts et le vocabulaire), *HBase*, *Hypertable* (une implémentation open source inspirée du système propriétaire *BigTable* de *Google*), *Amazon SimpleDB*.

Usages

La flexibilité des BDOC en fait un choix idéal, par exemple, pour stocker des rapports d'erreur issus de plusieurs applications dont chacune pourra utiliser le groupe de colonnes qui lui convient. Les blogs avec leurs publications, leurs liens, les rétro-liens ainsi que les commentaires associés sont particulièrement bien adaptés à l'utilisation d'une BDOC. Les systèmes de comptage de nombre de visites, comme nous l'avons vu dans le dernier exemple, peuvent tirer profit de la classification à deux niveaux, par site et par page, que permet le modèle des colonnes dynamiques.

3.3.4 Les bases de données orientées graphes

Concepts

Dans de nombreuses situations comme les réseaux sociaux, il s'agit de relier des entités par différentes relations, elles-mêmes dotées d'attributs et d'un sens de navigation. Telle personne « aime », « connaît » ou « fait partie du cercle professionnel » de telle autre. Malgré leur nom, les SGBDR (où « R » = relationnel) sont mal adaptés pour parcourir rapidement un graphe. Un SGBDR n'est bien adapté que pour décrire des relations simples, comme celles qui lient un client à ses commandes, par exemple. Cependant, dès que le nombre de liens à parcourir excède un ou deux, la nécessité de définir à chaque fois des clés étrangères et d'effectuer de nombreuses jointures pénalise les performances au point de rendre cette approche impraticable.

À ces limitations des SGBDR, il faut ajouter encore l'impossibilité pure et simple d'effectuer certaines opérations naturelles sur un graphe. Parcourir l'intégralité d'un graphe ou trouver un chemin entre deux nœuds vérifiant certaines contraintes sont deux exemples élémentaires. Les bases de données orientées graphes ont été conçues pour répondre spécifiquement à cette catégorie de besoins. Elles permettent dans un premier temps de créer des nœuds puis, indépendamment, de créer des associations entre eux et de les modifier par la suite.

Beaucoup de BDOG ne prennent pas en charge la distribution des nœuds de données sur un cluster et l'ACIDité n'est assurée qu'au sein d'un seul serveur. On pourrait peut-être arguer qu'elles n'ont pas vraiment leur place dans un ouvrage consacré au Big Data. Nous les présentons néanmoins rapidement par souci de cohérence dans ce chapitre consacré aux systèmes NoSQL.

Copyright © 2015 Dunod.

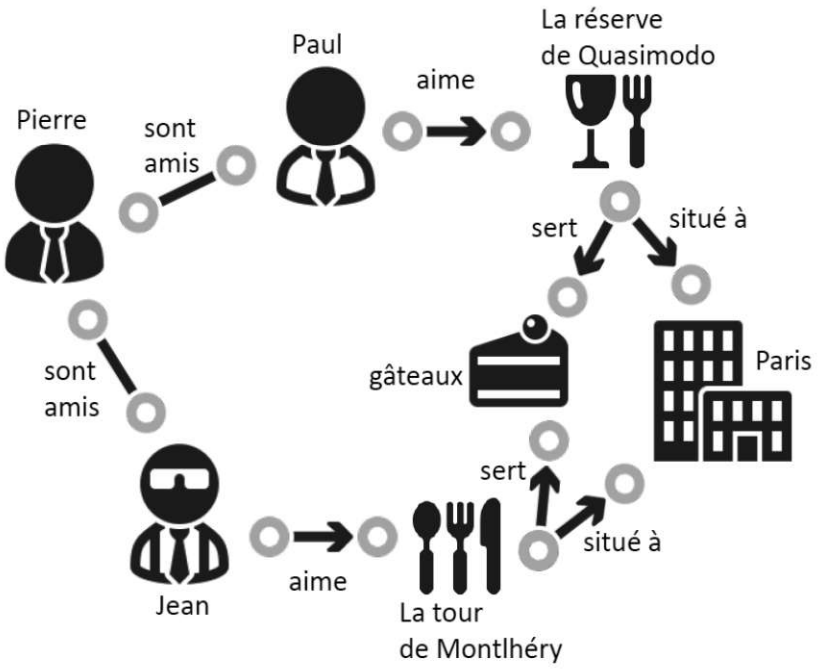


Figure 3.15 — Un exemple de graphe qui relie des individus, des services et des lieux géographiques.

D'une certaine manière la stratégie mise en œuvre dans une BDOG est à l'opposé de celle utilisée par BDOA. Alors que ces dernières regroupent des données au sein d'agrégats, les BDOG les éclatent autant que possible dans un graphe. Pour cette raison les BDOG sont généralement ACID, comme les SGBDR, pour la simple raison que l'éclatement des données exige que celles-ci soient protégées encore davantage que dans un SGBDR. Alors que les agrégats constituent des limites transactionnelles naturelles pour les BDOA, celles-ci n'existent pas dans le cas des BDOG et doivent donc être définies explicitement.

Parmi les BDOG les plus utilisées aujourd'hui citons : *Neo4J*, *Infinite Graph* et *OrientDB*.

Usages

A priori tous les domaines métiers qui s'expriment naturellement à l'aide de graphes sont éligibles à l'utilisation d'une BDOG. Ils sont particulièrement utiles dans les situations où différentes catégories de liens expriment des appartenances à certains groupes sociaux ou géographiques. Dans ce dernier cas, l'un des attributs des liens sera la distance qui sépare les nœuds qu'ils connectent. Tous les systèmes de recommandation ou de détection de fraude qui ont à tenir compte d'une certaine proximité, géographique, sociale ou autre pourront tirer profit d'un BDOG.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

3.4 LE NOSQL EST-IL L'AVENIR DES BASES DE DONNÉES ?

Comme le montrent les quatre sections précédentes, il n'existe pas de principe universel ou de percée algorithmique majeure sur laquelle s'appuieraient les systèmes NoSQL. De manière plus pragmatique, des systèmes avec des modèles de données différents répondent à des besoins différents. Le contraste avec les SGBDR est ici frappant, ceux-ci ayant longtemps été envisagés comme des systèmes quasi universels. L'abandon de cette universalité est en réalité une partie du prix à payer pour obtenir des performances élevées qui répondent aux exigences des applications web à grande échelle.

Une autre fraction de ce prix réside dans le transfert de la complexité technique des bases de données vers les applications. La gestion de la cohérence des données n'est désormais plus entièrement du ressort des bases de données mais doit être prise en charge par les applications métiers. Dans certains cas la gestion des incohérences aura même avantage, comme on l'a vu, à être assurée manuellement.

L'abandon d'un modèle de données universel et le transfert de complexité vers les applications impliquent une rigidité accrue du modèle de données dont il faut prendre conscience. Les applications devront être conçues pour un usage spécifique des données. L'utilisation de BDOA présupposera une direction de navigation privilégiée dans les données, du client vers ses commandes par exemple. Toute modification ultérieure sera cependant très coûteuse car elle exigera une réécriture d'une part significative du code de l'application.

Ne nous leurrons pas, les solutions NoSQL ne bénéficient pas encore d'un grand nombre de retours d'expériences. Il existe cependant deux motivations principales à leur mise en œuvre. La première est l'accroissement de la productivité des développeurs. Ceux-ci bénéficieront de la simplicité d'outils où aucun schéma ne vient contraindre les données. La seconde motivation est l'augmentation drastique des performances et de la scalabilité qu'ils permettent, notamment par une mise en *cluster*. Les projets pour lesquels la réduction du *time to market* est déterminante et ceux pour lesquels un avantage compétitif justifie une prise de risque technologique pourront utiliser avec profit ces nouvelles solutions. Les SGBDR ne sont pas morts pour autant, ils coexisteront encore longtemps avec ces solutions non orthodoxes.

Copyright © 2015 Dunod.

En résumé

La force du modèle relationnel classique réside avant tout dans la flexibilité de l'usage de données qu'il permet, et dans les garanties offertes par le caractère ACID des transactions.

Dans le contexte des applications web à grande échelle, ce modèle universel est désormais remis en question. Les exigences de performance et de disponibilité demandent de l'abandonner au profit des systèmes dont les modèles de données sont spécifiquement adaptés aux applications qui les exploitent tout en étant simultanément moins flexible.

Les applications métiers devront par ailleurs prendre en charge tout ou partie de la gestion de la cohérence des données. Les bases de données NoSQL coexisteront encore pendant de nombreuses années avec les SGBDR. Le recours à l'option NoSQL se fera avant tout sur la base d'un examen lucide et chiffré des compromis acceptables entre consistance des données et disponibilité des systèmes. Il s'agit donc au moins autant d'un choix commercial que d'un choix technologique.

Copyright © 2015 Dunod.

Copyright © 2015 Dunod.

4

L'algorithme MapReduce et le framework Hadoop

Objectif

Ce chapitre présente MapReduce, un modèle de programmation récent, qui a permis d'automatiser le déploiement de traitements massivement parallèles sur des clusters de centaines ou de milliers de serveurs peu coûteux. Nous examinerons les atouts et les contraintes qu'impose ce nouveau paradigme de programmation ainsi que ses limitations. Pour nous forger une intuition sur ses possibilités, nous illustrerons son fonctionnement au moyen de trois exemples simples.

Dans un contexte Big Data, l'algorithme MapReduce à lui seul ne serait pas d'une grande utilité sans une infrastructure logicielle dédiée, raison pour laquelle nous présenterons les grandes lignes du fonctionnement de Hadoop, qui est à ce jour la principale implémentation open source d'une telle infrastructure.

4.1 AUTOMATISER LE CALCUL PARALLÈLE

Au chapitre précédent nous avons décrit comment les contraintes propres au Big Data, notamment celles liées au volume de données et aux performances, ont contribué à faire émerger une nouvelle classe de systèmes de stockage, les bases NoSQL. Ce chapitre examine maintenant l'impact de ces contraintes sur les traitements de ces grands volumes de données. Une des voies praticables pour passer à l'échelle consiste à les paralléliser sur un cluster de plusieurs centaines ou milliers de machines. Hélas, cette mise en parallèle est en règle générale une tâche excessivement ardue qui

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

nécessite un savoir faire spécialisé et une longue phase de conception. De plus, dans les situations où un code non parallélisé existe déjà, un important travail de réécriture est le plus souvent nécessaire.

Un des progrès les plus significatifs dans l'automatisation de la parallélisation est venu de travaux de recherches effectués par Google pour les besoins de son moteur de recherche. Au début des années 2000, des centaines de processus parallèles avaient été conçus par les ingénieurs de Mountain View pour traiter les quantités pharamineuses d'information récoltées par les *web crawlers* du moteur de recherche : création d'annuaires inversés (liste de documents contenant certains mots clés), analyse des graphes de liens entre documents, identification des requêtes les plus fréquentes, etc. Bien que tous ces processus aient été conçus à l'origine par des équipes différentes, les ingénieurs se sont aperçus a posteriori que nombre de ces programmes mettaient en œuvre une stratégie similaire pour paralléliser les traitements.

C'est ce pattern de répartition des tâches que nous allons décrire dans ce chapitre et que l'on appelle **MapReduce**. Après l'avoir défini formellement, nous décrirons trois exemples d'utilisation élémentaires qui donneront une idée du mode de pensée qu'implique la conception d'un tel algorithme. La seule logique des traitements, telle qu'elle est prescrite par MapReduce, ne serait cependant pas d'une grande utilité, du moins dans un contexte Big Data, sans une infrastructure logicielle qui prend en charge la complexité liée à la distribution des traitements, à la réplication des données et à la tolérance aux pannes. Nous présenterons pour cette raison les grandes lignes du fonctionnement de Hadoop, à ce jour l'implémentation open source de référence d'un tel framework d'exécution distribuée du pattern MapReduce. Le chapitre 9 sera consacré à une description des principaux éléments de l'écosystème Hadoop.

Ajoutons encore pour être précis, que la définition des abstractions MapReduce a été inspirée non seulement par une synthèse d'expériences chez Google mais aussi par des travaux antérieurs dans le domaine de la programmation fonctionnelle[1].

4.2 LE PATTERN MAPREDUCE

Le pattern MapReduce a été décrit la première fois en 2004 dans un article fondateur[2] rédigé par deux ingénieurs de Google. Sur un plan intuitif, MapReduce part de l'observation que, dans de nombreuses situations, la mise en parallèle d'un traitement peut s'effectuer sur un plan logique au moyen de seulement deux types d'opérations, possédant une structure bien précise que nous allons décrire.

La première contrainte imposée par l'algorithme MapReduce est que les données en entrée doivent être structurées comme une liste de couples clé-valeur, analogues

1. En programmation fonctionnelle l'accent est mis sur l'évaluation des fonctions plutôt que sur les changements d'état des données associés à la programmation impérative usuelle.

2. MapReduce: Simplified Data Processing on Large Clusters par Jeffrey Dean et Sanjay Ghemawat, (2004)Article disponible en ligne http://static.googleusercontent.com/media/research.google.com/fr//archive/mapreduce-osdi04.pdf

Copyright © 2015 Dunod.

à celles qui sont stockées dans les ECV ou les BDOD décrits au chapitre précédent. En pratique, cette liste peut s'avérer très volumineuse, sa taille pouvant se chiffrer en téraoctets ou même en petaoctets. Cette liste en entrée est tout d'abord scindée en plusieurs lots (par le framework), chaque lot étant attribué à une des machines appelées *mapper*, voir la figure 4.1.

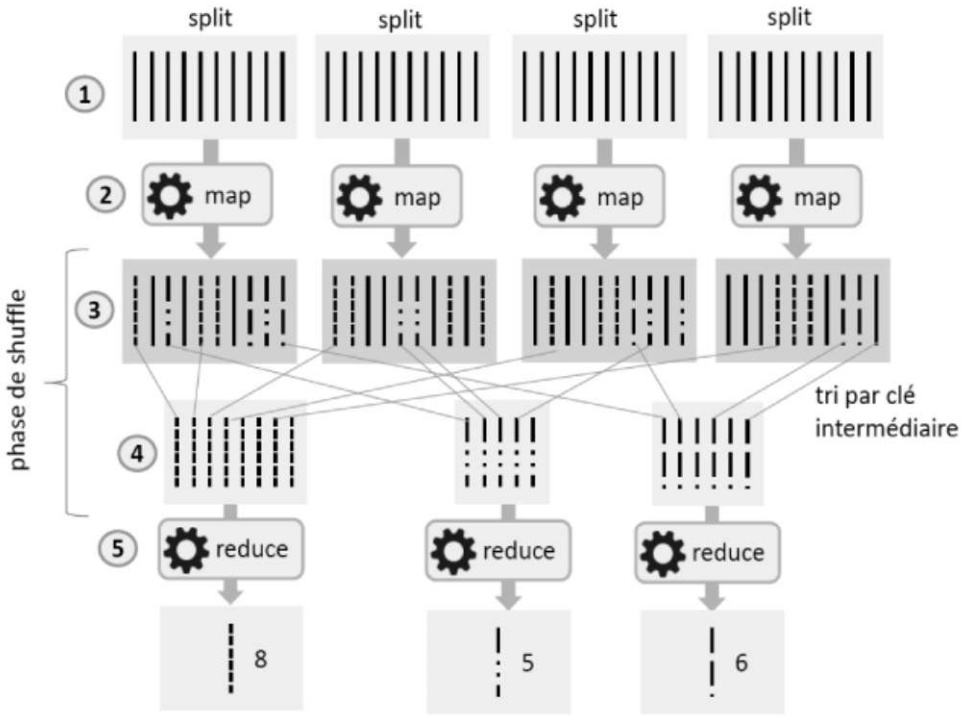


Figure 4.1 — Les données initiales sont scindées en lots. Chaque lot, aussi appelé *splits*, est confié à un *mapper* qui évalue la fonction *map* sur chaque enregistrement. Les listes obtenues en sortie des *mappers* sont concaténées, classées par valeur de la clé intermédiaire et scindées en lots par le framework. C'est la phase dite du *shuffle*. Les *reducers* évaluent la fonction *reduce* sur des listes de valeurs associées à une même valeur de clé intermédiaire.

- Dans une première phase, ces *mappers* exécutent une opération dite « *map* » sur chacun des enregistrements du lot qui leur a été confié. Chaque invocation de cette opération *map* retourne alors une liste de clés-valeurs intermédiaires que l'on peut envisager comme un résultat intermédiaire du calcul à effectuer sur la liste complète.

Remarque : l'indépendance des traitements map est absolument cruciale puisque c'est elle qui autorise la mise en parallèle de leur exécution, soit sur des machines différentes, soit au sein de processus distincts.

- Dans une seconde phase, une fois que tous les *mappers* ont tous achevé leur tâche, les fonctions « *reduce* », exécutées par des *reducers*, agrègent systématiquement toutes les valeurs intermédiaires associées à une même clé intermédiaire. Pour cela le framework fusionne au préalable les listes intermédiaires en sortie des *mappers*, les trie selon la valeur des clés, les répartit en lots puis enfin les achemine vers les *reducers*. Chaque lot sera assigné à un seul *reducer* et les

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

lots sont constitués de telle manière que les toutes les valeurs associées à une même clé figurent dans le même lot.

L'opération « *reduce* » peut donc être envisagée comme une opération de recombinaison ou d'agrégation des résultats partiels obtenus par les *mappers* lors de la première phase.

Pour rendre les choses plus concrètes, envisageons le calcul de données agrégées à partir d'un historique de vente sur plusieurs années, voir la figure 4.2. Chaque enregistrement de la liste correspond à une vente de plusieurs articles. Les seules données qui nous intéressent en l'occurrence sont les noms des articles, leur prix ainsi que la quantité vendue. Notre objectif est d'utiliser MapReduce pour calculer le prix de vente total et le nombre de ventes par produit. Conceptuellement, le calcul se réduit à une longue liste d'additions. La difficulté pratique, n'est donc pas de nature conceptuelle mais tient au fait qu'en pratique il faut généralement réunir des données réparties sur un grand nombre de serveurs. La tâche de la fonction *map* consiste alors à extraire pour chaque vente, la liste des produits associés à leur prix et au nombre de ventes. Après un regroupement par nom de produit, pris en charge par le *framework*, les fonctions *reduce* calculeront les deux sommes qui nous intéressent.

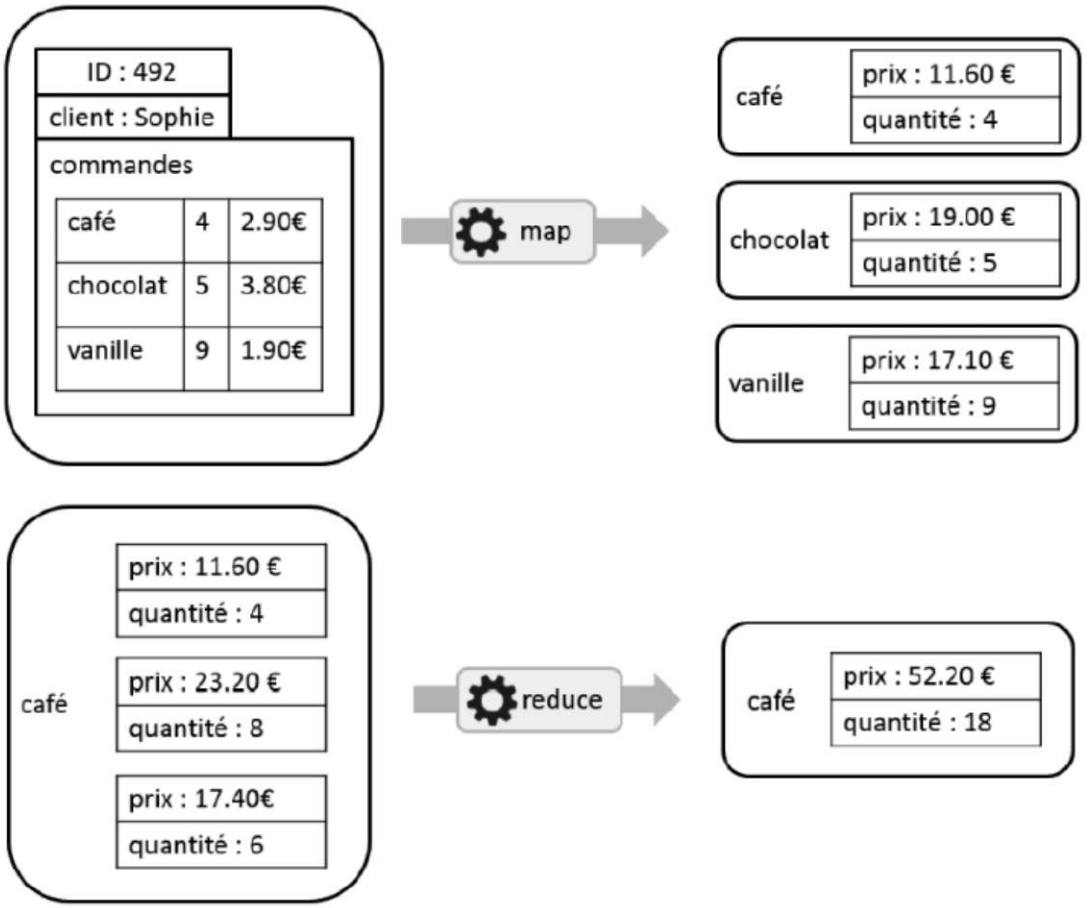


Figure 4.2 — Un exemple de fonctions *map* et *reduce* adaptées au calcul d'un prix total et d'une quantité totale par ventes à partir d'un historique de ventes.

Copyright © 2015 Dunod.

Voici, le pseudo-code des deux opérations `map` et `reduce` :

```
// La fonction map lit l'agrégat order et en extrait les données
// que l'on souhaite agréger à savoir la liste des quantités
// et des prix des produits vendus.
map(int orderID, Order order)
    List intermediateKeyValueList;
    for each item in order
        intermediateKey := item.getItemName();
        value := item.getPriceAndQuantity();
        intermediateKeyValueList.append(intermediateKey, value);
return {intermediateKeyValueList};
// La fonction reduce consomme une liste de valeurs intermédiaires
// associées à une même clé, c.-à-d. à un même article, et calcul
// la somme des prix et des quantités de cette liste intermédiaire
reduce(int intermediateKey, List intermediateListOfValues)
       totalPrice := 0;
       totalQuantity := 0;
       for each value in intermediateListOfValues
              totalPrice := totalPrice + value.getPrice();
              totalNumber := totalNumber + value.getQuantity();
return {totalNumber, totalQuantity};
```

Une version plus abstraite et plus succincte de l'interface MapReduce peut s'écrire :

```
map    : (k1, v1)       → list(k2, v2)
reduce : (k2, list(v2)) → aggregate(v2)
```

L'exemple qui vient d'être donné est particulièrement bien adapté au pattern MapReduce. Cependant, rien ne garantit qu'en général un traitement puisse se formuler de cette manière. Dans certaines situations la chose sera même impossible. Dans d'autres, elle nécessitera un travail conséquent de conception et de traduction d'un code non-MapReduce. C'est là le prix à payer pour automatiser la parallélisation : on se restreint à une classe particulière de traitements.

> **Remarque** : l'écriture d'algorithmes MapReduce est une tâche délicate qui ne peut être confiée qu'à des développeurs spécialisés qui se seront forgé au préalable des intuitions propres à ce nouveau modèle de programmation. Le plus souvent les tâches complexes à paralléliser sont converties non pas en un seul job MapReduce mais en une succession de petits jobs plus faciles à concevoir.

Nous retrouvons au niveau des traitements MapReduce une situation que nous avions déjà décrite à propos des bases NoSQL. Dans un cas comme l'autre le prix à payer pour la performance se traduit par un surcroît de complexité du code (celui des fonctions Map et Reduce en l'occurrence) qui devra être spécifiquement adapté au jeu de données utilisé. La situation est donc en contraste marqué avec celle des SGBDR où la connaissance du langage SQL suffisait dans la plupart des cas de figure.

Considérant, d'une part, la disponibilité des compétences SQL et, d'autre part, les subtilités de la programmation MapReduce, on comprend vite l'intérêt de disposer d'un mécanisme qui convertirait des scripts écrits dans un langage de haut niveau,

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

dans l'idéal proche de SQL, en jobs MapReduce. De tels mécanismes de compilation existent. Les plus connus se nomment Pig et Hive et seront décrits au chapitre 9 et illustré par des exemples au chapitre 10.

4.3 DES EXEMPLES D'USAGE DE MAPREDUCE

Pour nous familiariser avec l'algorithme MapReduce, examinons trois exemples élémentaires.

4.3.1 Analyse statistique d'un texte

L'une des premières applications de l'algorithme MapReduce pour un moteur de recherche comme Google a été l'analyse de textes en vue de construire des annuaires inversés (quelles pages contiennent quels mots clés) et des statistiques de toutes sortes. Considérons un exemple dans cette deuxième catégorie et imaginons que nous souhaitions analyser plusieurs millions de documents et construire l'histogramme de la longueur des mots qu'ils contiennent. Admettons que nous nous intéressions à trois catégories de mots, classés selon leur longueur.

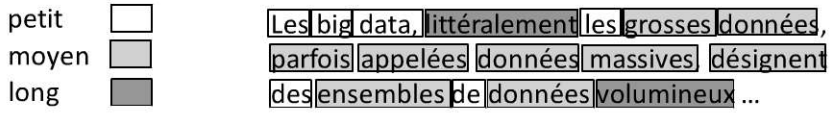


Figure 4.3 — Un texte dont les mots ont été triés en trois catégories. Les mots longs, de plus de 10 lettres, les mots de longueur moyenne de 5 à 9 lettres, les petits mots de moins de cinq lettres.

Avant de transmettre la liste de fichiers aux *mappers*, le *framework* la scindera en lots plus petits, chacun étant confié à un des *mappers*. Dans le cas de figure où l'on dispose d'un millier de *mappers*, chaque *mapper* aura donc à traiter quelques milliers de fichiers. On supposera pour simplifier que chaque fichier individuel est suffisamment court pour être traité par une opération *map*. Le rôle des fonctions *map* et *reduce* est dès lors facile à concevoir.

La fonction *map*, rappelons-le, prend en entrée un couple clé-valeur. La clé en l'occurrence n'est autre que le nom ou l'identifiant d'un document et la valeur son contenu. La fonction *map* effectuera le travail de décompte des trois catégories de mots au niveau d'un document. Chaque invocation de *map* fournira donc une liste de mots associés à leur fréquence dans le document, c'est la liste de clé-valeur intermédiaire.

Le *framework* fusionnera ensuite ces listes intermédiaires, les triera et les scindera par mot pour constituer de nouveaux lots, chacun étant assigné à un *reducer*.

La fonction *reduce*, rappelons-le, prend en entrée une liste de valeurs intermédiaires associées à une même clé et calcule l'agrégat souhaité. Chaque invocation de *reduce* calculera en l'occurrence la somme des décomptes partiels pour un des mots.

Copyright © 2015 Dunod.

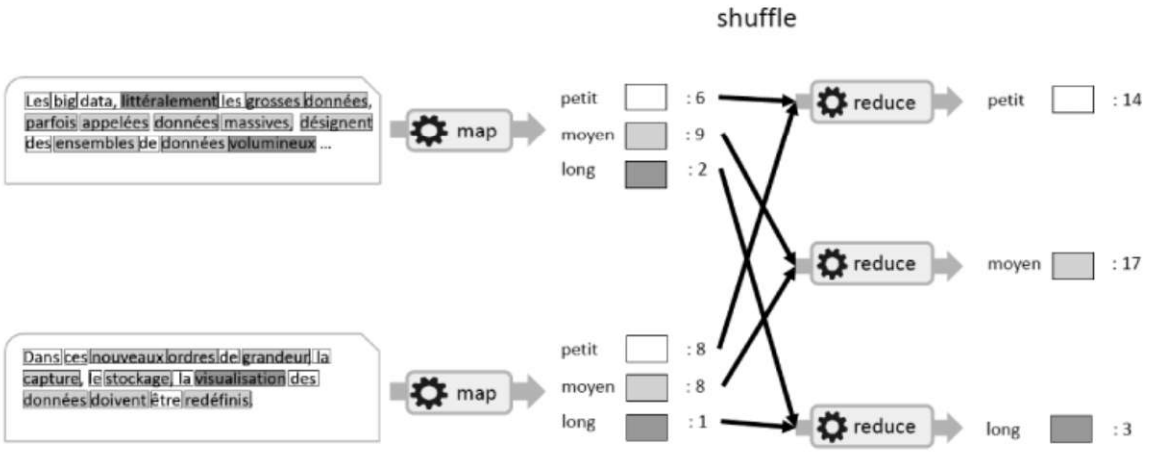


Figure 4.4 — La succession des tâches *map* et *reduce* utilisées pour le calcul de l'histogramme de fréquence de mots d'une certaine longueur.

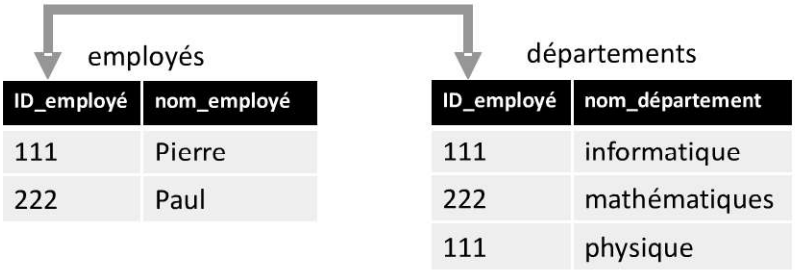
4.3.2 Calcul d'une jointure entre deux grandes tables

Comme nous l'avons indiqué au chapitre précédent, le calcul d'une jointure entre deux tables d'un modèle relationnel est une opération particulièrement gourmande en ressources. À tel point que certains modèles de données en agrégats utilisés par certains systèmes NoSQL ont été conçus pour les éviter complètement dans un contexte Big Data. On préfère dans ces situations colocaliser les données qui sont reliées dans un seul et même enregistrement. C'est ce que font les BDOC et les BDOD au prix, il est vrai, d'une moindre flexibilité du modèle (qui doit être adapté à un cas d'usage spécifique) et d'une redondance accrue. Dans certains cas cependant, le calcul de grandes jointures reste incontournable et dans ces situations l'algorithme MapReduce vient alors à la rescousse à point nommé.

Rappelons qu'une jointure entre deux tables A et B, envisagées comme des listes d'enregistrements, est définie par deux clés, une par table. Le résultat sera une nouvelle relation dont les enregistrements sont obtenus en concaténant tous les couples formés d'un enregistrement de A avec un enregistrement de B dont les deux clés coïncident.

L'idée de base pour réaliser cette jointure avec MapReduce consiste à réaliser une opération en apparence triviale avant même la phase *map*, à savoir regrouper les enregistrements des deux tables dans une seule longue liste d'enregistrements. De plus, pour chaque enregistrement, on retiendra non seulement son contenu, mais également le nom de la table dont il est issu, voir figure 4.6. Bien que logiquement séparée du reste du traitement, cette tâche pourra en fait être prise en charge par les fonctions *map* elles-mêmes.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.



employés

ID_employé	nom_employé
111	Pierre
222	Paul

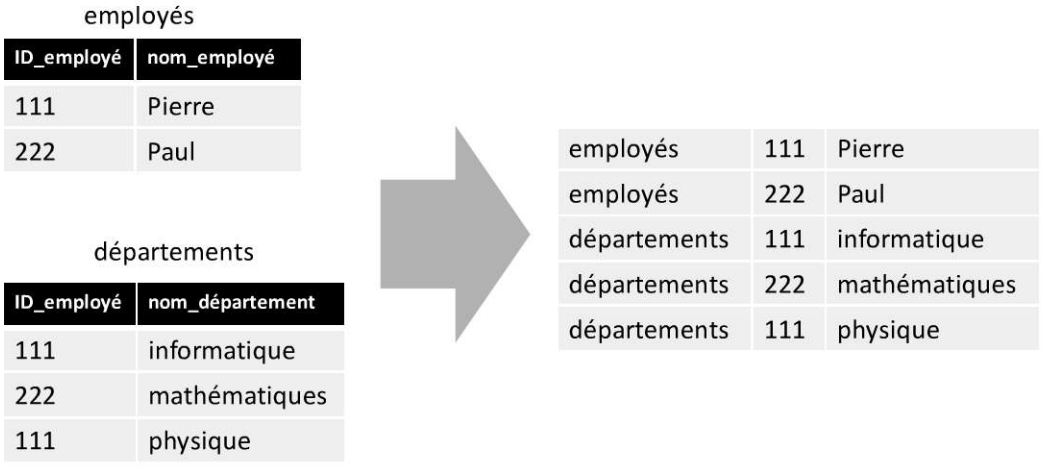
départements

ID_employé	nom_département
111	informatique
222	mathématiques
111	physique

jointure employés-département

ID_employé	nom_employé	nom_département
111	Pierre	informatique
111	Pierre	physique
222	Paul	mathématiques

Figure 4.5 — Une jointure entre deux tables, définie par un couple de clés, consiste à concaténer des enregistrements pour lesquelles les clés coïncident.



employés

ID_employé	nom_employé
111	Pierre
222	Paul

départements

ID_employé	nom_département
111	informatique
222	mathématiques
111	physique

employés	111	Pierre
employés	222	Paul
départements	111	informatique
départements	222	mathématiques
départements	111	physique

Figure 4.6 — Avant la phase *map* proprement dite, on collectionne l'ensemble des enregistrements dans une liste qui contient à la fois les données et le nom de la table dont elles sont issues.

La tâche d'une opération *map* est dès lors d'une simplicité biblique : pour chaque enregistrement en entrée, elle renverra un couple clé-valeur dont la clé est la clé de jointure et la valeur est le contenu de l'enregistrement auquel on a adjoint le nom de la table originale, voir la figure 4.7.

Le *framework* regroupera ensuite en listes d'enregistrements intermédiaires les enregistrements qui partagent une même clé de jointure. Chacune de ces listes est affectée ensuite à un *reducer*. Une invocation d'opération *reduce* a pour tâche de produire la jointure proprement dite, c'est-à-dire d'effectuer toutes les concaténations

Copyright © 2015 Dunod.

possibles d'un enregistrement de la table A avec un enregistrement de la table B associées à une même valeur de clé de jointure.

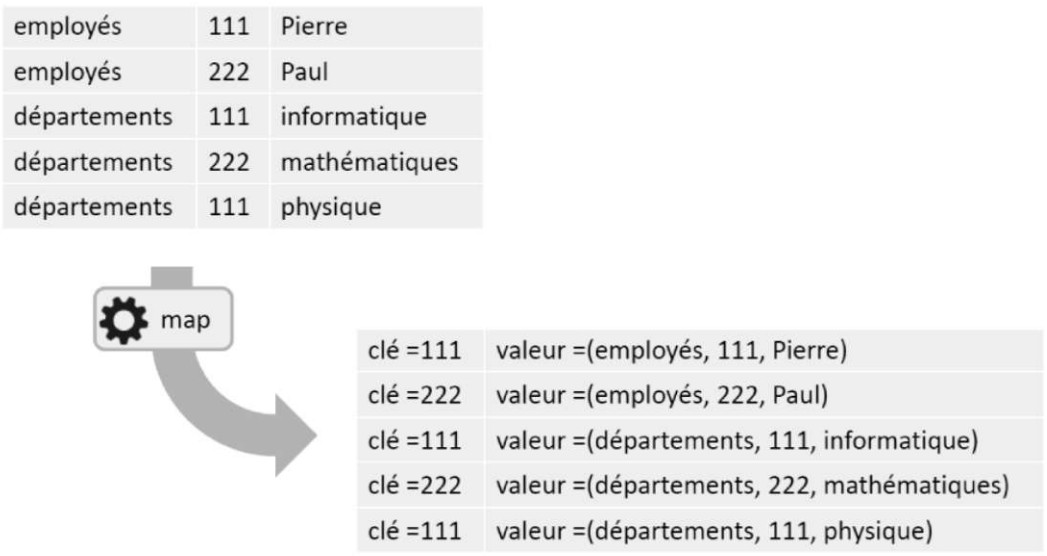


Figure 4.7 — La fonction *map* retourne le contenu d'un enregistrement et le nom de la table originale dans la valeur de retour. La clé est la clé de jointure.

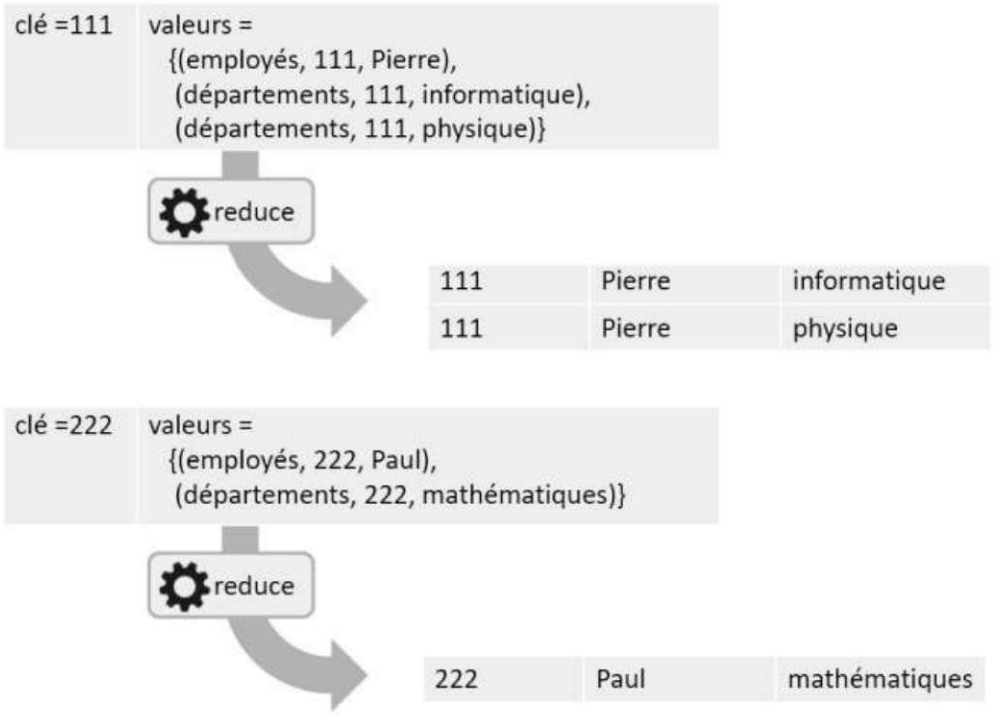


Figure 4.8 — La fonction *reduce* produit toutes les concaténations voulues à partir d'une liste d'enregistrement intermédiaires résultats du shuffle. Pour chaque valeur de clé de jointure, elle produit la liste des concaténations des couples d'enregistrements associés à cette clé.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

4.3.3 Calcul du produit de deux matrices creuses

Dans des contextes d'analyse de données à grande échelle il n'est pas rare que les tableaux de données comportent des milliers ou des dizaines de milliers d'attributs dont très peu sont renseignés. Dans ces situations, les traitements statistiques impliquent la manipulation de matrices dont une majorité d'éléments sont des zéros à l'exception de quelques valeurs. On les appelle des **matrices creuses**. Dans ce dernier exemple, nous illustrerons comment on peut utiliser MapReduce pour calculer le produit de deux matrices de ce type. La représentation usuelle d'une matrice creuse A se fait sous la forme d'une liste de couples de type $\{i, j, a_{ij}\}$ où a_{ij} désigne la valeur de l'élément figurant dans la ligne i et la colonne j. Sont omis de cette liste tous les éléments pour lesquels a_{ij} est nul. La figure 4.9 rappelle comment se calcule un produit de deux matrices : en effectuant un produit scalaire entre les lignes de A avec les colonnes de B.

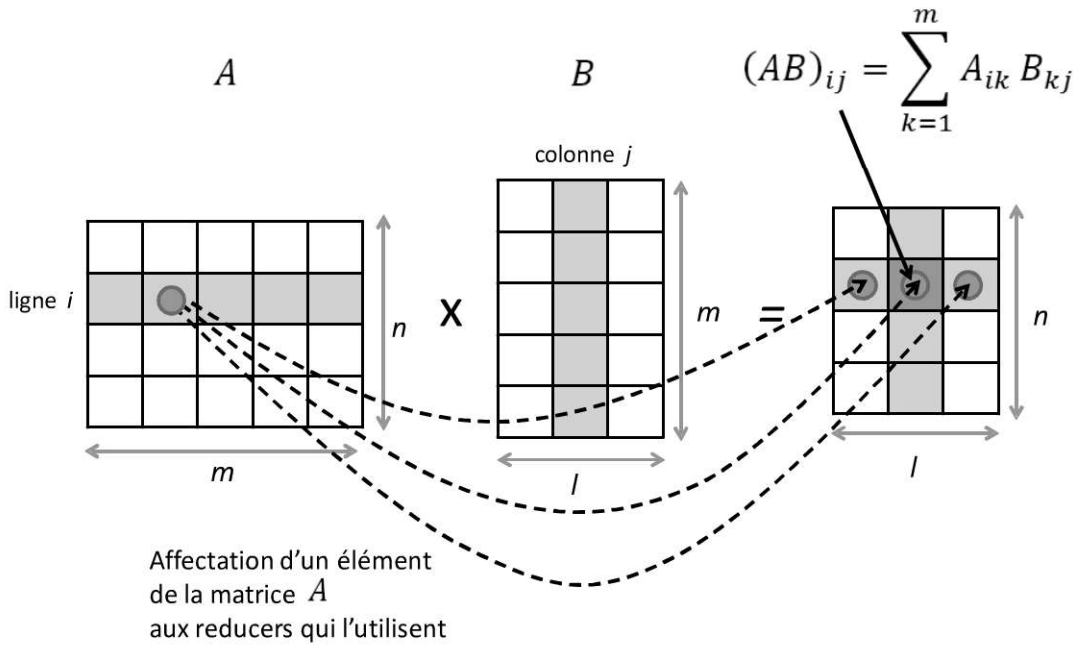


Figure 4.9 — Schéma du calcul du produit de deux matrices creuses à l'aide de l'algorithme MapReduce.

Deux idées permettent d'appliquer MapReduce à la parallélisation de ce calcul. La première consiste à utiliser un *reducer* pour chacun des $n \times l$ éléments de la matrice AB que l'on veut calculer. Afin que chaque *reducer* puisse calculer la somme des produits requise, il faut que celui-ci ait accès aux seuls éléments des matrices A et B dont il a besoin. C'est précisément la tâche des *mappers* que de les lui fournir. Plus explicitement, en commençant, une fois n'est pas coutume, par l'opération *reduce* :

a) **Reduce** : considérons le *reducer* en charge du calcul de l'élément (i,j) du produit AB. D'après la définition, on voit qu'il lui faut disposer pour cela de deux listes de valeurs. D'une part la liste des éléments a_{ik} pour chaque k de 1 à m et d'autre part la liste des éléments b_{kj} , également pour k de 1 à m. Ces deux listes

Copyright © 2015 Dunod.

associées à (i,j), considéré comme une clé intermédiaire, sont les deux listes de valeurs intermédiaires que le *framework* devra fournir au *reducer*.

b) **Map** : les fonctions *map* devront par conséquent fournir ces listes aux *reducers* en utilisant le couple (i,j) comme clé intermédiaire. Chaque *mapper* parcourt le lot des éléments de matrices qui lui a été attribué. L'action de la fonction *map* est différente, quoique similaire, selon que l'élément en entrée appartient à la matrice A ou à la matrice B :

- Un élément aik, qui est un élément de la ligne n°i de A, devra être fourni à tous les *reducers* qui l'utilisent, à savoir aux l *reducers* associés à la ligne n° i de AB. En d'autres termes, *map* devra retourner la liste de couples clé-valeur : $\{(i,j), a_{ik}\}$ pour $j=1,..., l$.
- Un élément b_{kj}, qui est un élément de la colonne n°j de B, devra être fourni à tous les *reducers* qui l'utilisent, à savoir aux m *reducers* associés à la colonne n° j de AB. En d'autres termes, *map* devra retourner la liste de couples clé-valeur : $\{(i,j), bkj\}$ pour $j=1,..., m$.

L'éventuel début de migraine qu'aura pu susciter la lecture des deux derniers paragraphes ne doit pas inquiéter outre mesure. Il n'est que le symptôme d'une intuition MapReduce qui se fortifie.

4.4 LE FRAMEWORK HADOOP

Notre présentation de l'algorithme MapReduce s'est limitée jusque-là à ses seuls aspects logiques, en particulier aux éléments principaux que sont les deux abstractions « *map* » et « *reduce* ». Si la maîtrise de ces API et l'aptitude à les utiliser pour reformuler un problème de parallélisation sont des compétences indispensables pour quiconque souhaite utiliser MapReduce, elles ne suffisent pas cependant pour construire un système performant.

En effet, aussi élégantes soient-elles, les abstractions « *map* » et « *reduce* » ne masquent pas, hélas, toute la complexité de la parallélisation d'un calcul. Une connaissance élémentaire de ce qui se trame dans les coulisses du *framework* reste indispensable. En d'autres termes, si un *framework* comme Hadoop prend effectivement en charge un grand nombre de tâches de bas niveau, encore faut-il savoir lesquelles, dans quel ordre celles-ci sont invoquées et quelles ressources elles consomment. Nous retrouvons, une fois encore, le phénomène de non-découplage entre les abstractions et leur implémentation lorsqu'il s'agit d'optimiser les performances.

L'implémentation de l'algorithme MapReduce dépend de l'environnement dans lequel on souhaite l'exécuter. Dans un contexte Big Data, c'est une implémentation sur un cluster de serveurs qui nous intéresse. Les deux paragraphes qui suivent présentent

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

les grandes lignes du fonctionnement de Hadoop, le *framework* MapReduce le plus populaire à l'heure actuelle[1].

Hadoop est un *framework* open source écrit en Java et développé sous l'égide de la fondation Apache depuis 2005. La version 1.0 a vu le jour fin 2011.

4.4.1 Planning des exécutions

L'unité de travail qu'une application cliente peut exécuter dans Hadoop s'appelle un **job** MapReduce. Chaque job est découpé en **tâches** qui peuvent appartenir à deux catégories : les **tâches map** ou les **tâches reduce**.

L'ensemble des jobs en cours d'exécution par le framework est coordonné et supervisé par un processus maître appelé le **jobtracker**. Son rôle est de planifier l'exécution des tâches qu'il attribue à des processus esclaves, appelés **tasktrackers**. Pour isoler les exécutions des tâches *map* et *reduce* et éviter que celles-ci ne fassent tomber le serveur qui les exécute en cas de plantage, le *tasktracker* démarre chaque tâche *map* ou *reduce* dans une JVM séparée. Voilà qui nous amène naturellement au sujet de la tolérance aux pannes.

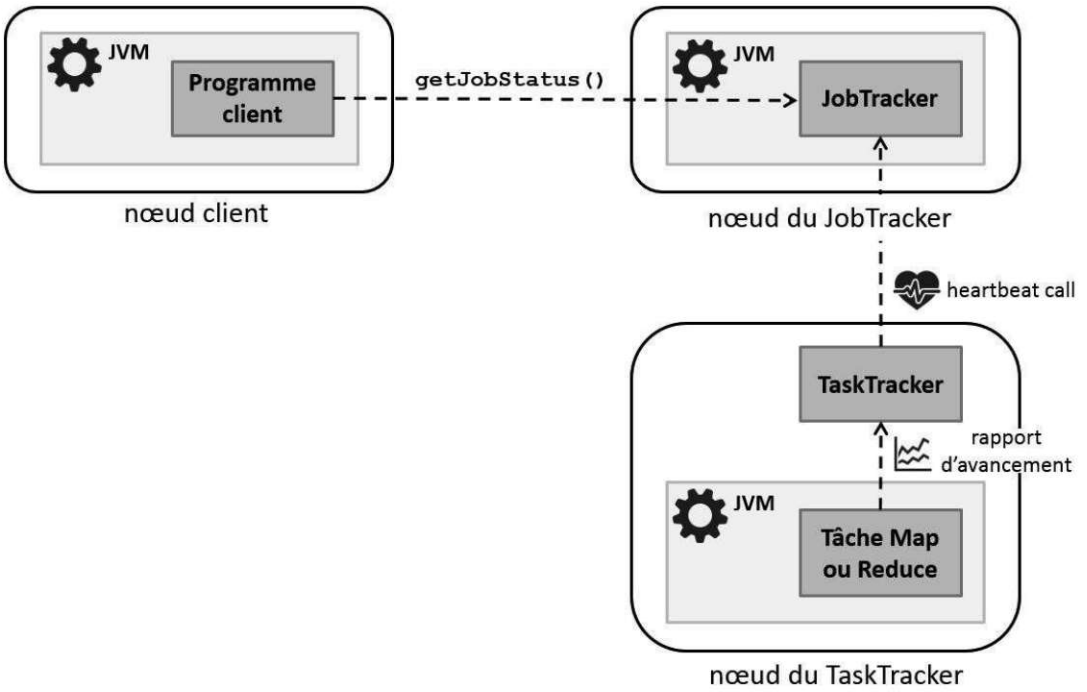


Figure 4.10 — Les relations entre un client d'un job MapReduce, le *jobtracker* et un *tasktracker*. Un processus de notification régulier informe le *jobtracker* (maître) de l'état d'avancement des tâches exécutées par chaque *tasktracker* (esclave). Par sécurité, chaque tâche *map* ou *reduce* s'exécute au sein d'une JVM séparée du processus du *tasktracker*.

1. Le lecteur souhaitant une introduction technique à Hadoop pourra consulter par exemple *Hadoop: The Definitive Guide* par Tom White chez O'Reilly (2012).

Copyright © 2015 Dunod.

4.4.2 Tolérance aux pannes

Jobtrackers et tasktrackers

Le système Hadoop, comme le système propriétaire de Google dont il s'inspire, est conçu pour exécuter MapReduce sur un cluster homogène de plusieurs milliers de serveurs bon marché. De ce fait, la tolérance aux pannes est au cœur même de sa conception. Imaginons à titre d'exemple une fréquence d'une panne par machine et par an. La fréquence des pannes pour un cluster de 10 000 machines sera dès lors d'environ une par heure. La responsabilité du *framework* est donc de faire en sorte qu'en dépit de ces pannes inévitables le résultat de l'exécution de MapReduce soit le même que celui que produirait une machine parfaite.

Le cœur du mécanisme qui assure la tolérance aux pannes est un mécanisme de notification. Chaque *tasktracker* est censé envoyer à intervalles réguliers au *jobtracker* un rapport d'avancement des tâches qui lui ont été confiées. Ce signal s'appelle le **heartbeat call**. En cas de besoin, le *jobtracker* ordonnera la réexécution de toutes les tâches dont on sait explicitement qu'elles ont échoué ou des tâches exécutées par des *tasktrackers* qui n'ont pas donné de nouvelles après un certain délai.

Les causes possibles d'une panne sont évidemment nombreuses : un bug dans l'implémentation d'une des tâches *map* ou *reduce*, un plantage du *tasktracker* lui-même ou la panne d'une machine physique, etc. Les causes d'erreur les plus communes sont toutefois des erreurs d'implémentation dans le code des fonctions *map* ou *reduce*. En cas d'erreur, le code utilisateur lève une exception d'exécution que la JVM répercute au *tasktracker* avant que celle-ci ne s'arrête. L'erreur est historisée dans un log utilisateur et la tâche qui a échoué est notée comme telle par le *tasktracker* qui libère alors les ressources utilisées par la tâche fautive pour les allouer à une autre tâche. Le *jobtracker* sera notifié de l'échec au prochain *heartbeat call* et c'est à lui que revient la responsabilité de planifier une nouvelle tentative d'exécution en évitant autant que possible de réallouer la tâche à un *tasktracker* où celle-ci aurait déjà échoué précédemment. Le nombre de tentatives de réexécution d'une tâche est paramétrable.

HDFS : un système de fichiers distribué

Dans un contexte Big Data le volume de donnée à traiter excède le plus souvent la capacité d'une seule machine. Un système de fichiers distribués sur tous les nœuds du réseau devient dès lors indispensable. L'implémentation que propose Hadoop se nomme **HDFS** pour *Hadoop Distributed File System*. Il prend en charge toute la complexité liée à la distribution et à la réplication des données sur les nœuds du cluster pour garantir qu'aucune donnée ne soit perdue en cas de panne d'un ou plusieurs disques durs. Nous décrirons ce système plus en détail au chapitre 9.

Exécutions spéculatives

Mis à part les plantages, il peut également arriver aussi qu'une tâche s'exécute plus lentement que prévu lorsque l'on compare son temps d'exécution à la moyenne des autres tâches. Statistiquement, ce phénomène est en réalité tout aussi inévitable que les plantages. Remarquons qu'un tel ralentissement est particulièrement préjudiciable

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

à la performance globale d'un job lorsqu'il se produit dans une tâche *map*. En effet, rappelons que la totalité des processus *map* doit être achevée avant de pouvoir passer à la phase *reduce*. C'est donc la tâche *map* la plus lente qui est déterminante pour la performance globale d'un job.

Plutôt que de diagnostiquer les raisons d'un éventuel ralentissement, Hadoop lancera alors une ou plusieurs nouvelles exécutions de la même tâche, les mettant ainsi en concurrence. Sitôt qu'un des processus en compétition s'achève avec succès, les autres tâches devenues redondantes sont stoppées. Ce mécanisme de mise en compétition est appelé l'**exécution spéculative**.

4.4.3 Découpage des données en lots

La possibilité même d'un calcul parallèle repose sur le découpage des données en entrée en plusieurs lots. Comme cela a été mentionné dans la section 4.2, chaque *mapper* se voit affecter un des lots que l'on nomme aussi **split** dans le vocabulaire Hadoop. Le nombre de *mappers* coïncide donc avec celui des *splits*. Il sera fonction du volume des données à traiter, du temps d'exécution individuel des fonctions *map* et du temps d'exécution moyen attendu pour chaque job.

La bonne taille des *splits* est le résultat d'un compromis. Un grand nombre de petits lots favorise une bonne répartition de la charge entre les différents processus *map* et donc les performances du système. Mais, passé un certain seuil, le découpage pénalisera les performances car les ressources nécessaires à la gestion des *splits* et à la création des processus *map* l'emporteront sur le bénéfice d'une bonne répartition de la charge. En règle générale un bon choix pour la taille des *splits* correspond à la taille des blocs du système de fichiers distribué, environ 64 Mo.

4.4.4 Fusion et tri des listes intermédiaires

Cette tâche a déjà été largement décrite dans la section 4.2. C'est la phase de *shuffle* durant laquelle les listes de clé-valeur intermédiaires en sortie des opérations *map* sont d'abord fusionnées avant d'être triées par valeur de clé puis enfin transmises aux *reducers*.

> **Remarque** : la phase de *shuffle* constitue le véritable cœur de l'algorithme MapReduce. C'est le tri opéré durant cette phase qui relie les *mappers* aux *reducers* et permet en définitive la parallélisation des traitements. Une compréhension de ce qui s'y passe est indispensable dès lors que l'on souhaite procéder à certaines optimisations. Cette partie centrale du framework Hadoop bénéficie d'améliorations continuelles.

Le processus de *shuffle* est réparti entre les *mappers* et les *reducers*.

Copyright © 2015 Dunod.

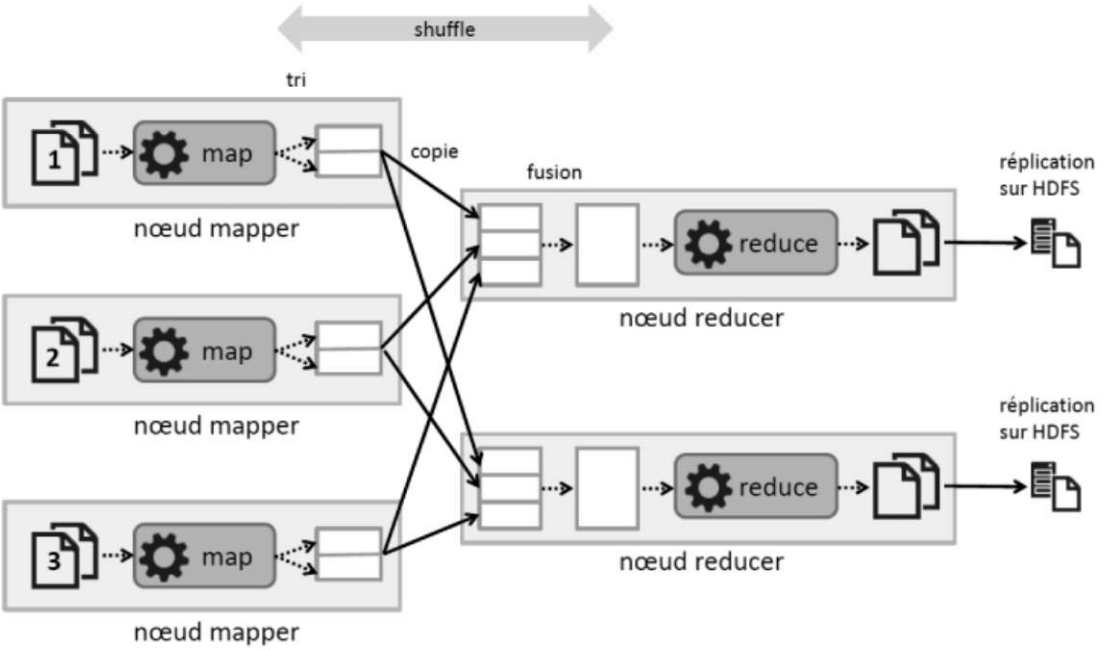


Figure 4.11 — Articulation des phases *map, shuffle* et *reduce*.

Le tri côté mapper

Les listes de clé-valeurs intermédiaires produites par les *mappers*, sont des objets temporaires, raison pour laquelle elles ne sont pas stockées sur le système de fichiers distribués HDFS. L'écriture de ces listes se fait d'abord dans un *buffer* en mémoire alors qu'un processus en arrière-plan effectue un tri en mémoire selon les valeurs des clés intermédiaires et construit des partitions associées aux *reducers* auxquelles les listes seront transmises. Toutes les valeurs associées à une même clé intermédiaire sont écrites dans une même partition. Régulièrement, le contenu du *buffer* est copié sur le disque local du *tasktracker* qui héberge le processus *map*.

Le contenu du disque local n'est effacé qu'une fois les données consommées par les *reducers* et lorsque ceux-ci ont notifié le succès de l'opération au *jobtracker*.

Pour économiser la bande passante consommée par la transmission des listes triées aux *mappers*, un développeur peut intervenir sur l'opération de tri pour rendre la sortie des *mappers* plus compacte. Il devra pour cela en implémenter une définie par l'API MapReduce de Hadoop appelée *combiner*[1].

Le tri côté reducer

Contrairement aux données des *mappers*, les listes intermédiaires en entrée d'un *reducer* ne sont pas localisées sur le disque local du *tasktracker* qui l'exécute. Elles proviennent d'une multitude de *mappers* répartis dans le *cluster*. La copie des listes des *mappers* vers les *reducers*, qui s'opère en mémoire ou sur le disque local selon le volume

1. On pourra consulter sur ce point l'ouvrage *Hadoop: The Definitive Guide* déjà mentionné.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

des données, débute sitôt que la totalité des *mappers* ont terminé leur travail. Une fois l'intégralité des listes intermédiaires copiées localement, les listes sont fusionnées. Chaque invocation de fonction *reduce* traite la liste de valeurs associées à une clé. À l'issue du traitement le résultat est écrit sur HDFS qui répliquera les résultats sur d'autres nœuds par sécurité.

4.4.5 Monitoring des processus

Étant donné que les disfonctionnements, matériels ou logiciels, sont inévitables dans un *cluster*, il est crucial de pouvoir superviser en continu son état de santé pour être en mesure d'intervenir à temps, qu'il s'agisse d'augmenter le nombre de machines dans le *cluster* ou de remplacer celles qui sont défectueuses. En plus des traditionnels mécanismes de logs, Hadoop offre toute une panoplie de métriques et de compteurs.

Les métriques sont collectées par des processus en arrière-plan et fournissent aux administrateurs systèmes des informations techniques comme le nombre de requêtes utilisateurs ou le nombre de blocs répliqués. Certaines de ces métriques sont disponibles via l'API Java standard JMX.

Les compteurs s'adressent quant à eux aux utilisateurs MapReduce que sont les concepteurs d'applications métiers. Ils contiennent des informations globales à l'échelle d'un job comme le nombre d'enregistrements invalides par exemple. Il s'agit d'outils de contrôle de qualité ou d'aide à la création de statistiques.

4.5 AU-DELÀ DE MAPREDUCE

Depuis quelques années le modèle de programmation MapReduce et le *framework* d'exécution Hadoop offrent une solution pragmatique à la parallélisation massive des traitements pour des volumes de données qui, désormais, peuvent se chiffrer en pétaoctets. Pour autant ce modèle comporte aussi des faiblesses dont il faut être conscient. Rappelons à ce titre quelques-unes des contraintes qu'impose le paradigme MapReduce :

a) Seuls les traitements que l'on peut reformuler à l'aide d'un couple de fonctions *map* et *reduce* sont éligibles à ce type de parallélisme.

b) Même lorsque la chose est possible, le temps nécessaire pour écrire un job MapReduce et le tester se chiffre souvent en jours.

c) Le *framework* Hadoop reste un outil complexe dont le paramétrage et l'optimisation requièrent un niveau de compétence élevé.

d) Bien que l'architecture en *cluster* garantisse une scalabilité linéaire, les temps de latence sont caractéristiques d'un traitement batch. Ils s'échelonnent entre quelques dizaines de secondes, au mieux, à plusieurs heures.

Les points c) et d) imposent dès lors un constat implacable :

Copyright © 2015 Dunod.

Avertissement
Hadoop et MapReduce ne sont pas adaptés aux traitements interactifs de grands jeux de données. Pour cela d'autres outils, indépendants ou complémentaires à Hadoop, sont nécessaires.

Dit autrement, des trois V du Big Data, MapReduce n'en traite en vérité qu'un seul : le volume des données. S'agissant de la prise en charge de la variété des données, assurément le problème le plus ardu des trois, aucune solution miracle ne se profile à l'horizon (voir toutefois la description de Drill au chapitre 9). Pour ce qui est de la vitesse en revanche, des solutions alternatives, plus performantes et plus flexibles que Hadoop, font depuis peu leur apparition sur le marché. Qu'il s'agisse de tirer parti d'un contexte client fugace ou de prendre en compte de données de géolocalisation, les besoins d'analyse en quasi-temps réel, en effet, ne manquent pas. Par temps réel on entend généralement un temps de latence inférieur à la seconde, délai que Hadoop est incapable de fournir.

En réponse à ces nouvelles exigences de vélocité et d'interactivité, certaines librairies qui utilisaient à l'origine Hadoop, ont récemment fait le choix d'abandonner ce modèle pour se tourner vers d'autres technologies plus récentes et plus performantes. C'est le cas par exemple d'Apaches Mahout, un *framework* dédié aux *machines learning* distribuées, qui utilise désormais un nouveau modèle de calcul en mémoire appelé Apache Spark qui sera décrit dans la section 9.5, modèle qui promet dans certaines circonstances des performances jusqu'à cent fois supérieures à celles de Hadoop. La partie 3 de ce livre présentera une sélection de ces nouveaux systèmes de traitement en temps réel.

Comme bien souvent, face à la frénésie de nouveautés technologiques qui agite le secteur de l'IT se pose la question : « *Hadoop n'était-il en définitive qu'une mode passagère, comme bien d'autres, destinées à passer bientôt aux oubliettes ?* ». La réponse mérite d'être nuancée car il faut distinguer ici le modèle de calcul MapReduce et les systèmes qui appartiennent à l'écosystème Hadoop, au premier rang desquels le système de fichiers distribués HDFS.

S'agissant du modèle de programmation MapReduce on peut imaginer qu'il sera progressivement remplacé par d'autres, plus flexibles, dans de nombreuses situations, certains d'entre eux seront décrits au chapitre 9. Parallèlement, les systèmes qui émergent comme alternatives, comme le *framework* d'analyse prédictive Apache Spark ou la base de données en mémoire SAP HANA proposent tous des mécanismes d'intégration à un cluster Hadoop et notamment à HDFS. À cela deux raisons principales.

La première tient au fait qu'il existe aujourd'hui un grand nombre de *clusters* Hadoop installés dans les entreprises et que ceux-ci donnent entière satisfaction à leurs utilisateurs. Dès lors, pour asseoir leur propre influence, les nouveaux arrivants ont tout intérêt à tenir compte de cette base technologique existante. Toutes proportions gardées, on peut anticiper pour Hadoop un phénomène d'inertie similaire à celui qui a assuré l'hégémonie des SGBDR pendant un demi-siècle.

La seconde raison est plus objective et tient au fait que dans des contextes d'analyse et de *machine learning* à très grande échelle Hadoop fait des merveilles dans les tâches

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

de préparation de quantité énorme de données : nettoyage, conversion de formats, transformations et extractions de données agrégées diverses. Dans ces situations Hadoop fait figure d'ETL boosté aux hormones. Les systèmes de *data mining* ou les processus de *machine learning* dont il sera question au chapitre 7, interviennent a posteriori sur ces données soigneusement préparées et sélectionnées par Hadoop. Dans de tels contextes, Hadoop et les systèmes de traitements en mémoire plus récents ne sont plus des concurrents mais fonctionnent en symbiose.

En résumé

MapReduce est un *pattern* logiciel qui permet de paralléliser le traitement de très grands volumes de données à condition que le calcul puisse se formuler à l'aide de deux fonctions, *map* et *reduce*, qui opèrent sur des listes de clé-valeur.

Hadoop est aujourd'hui le standard de fait en tant que plateforme d'exécution de MapReduce. Conçu pour fonctionner sur des *clusters* homogènes de machines peu coûteuses, il prend en charge l'essentiel des problématiques de bas niveau lié à la planification des tâches, à la répartition de charge et à la réplication des données. Hadoop est particulièrement bien adapté pour le traitement *batch* de très grands volumes de données comme le nettoyage ou la préparation d'un jeu de donnée en amont d'une analyse de type *machine learning*.

MapReduce en tant que tel n'est cependant pas adapté aux traitements interactifs en temps réel.

Copyright © 2015 Dunod.

DEUXIÈME PARTIE

Le métier de data scientist

L'analyse prédictive est un ensemble de techniques et d'outils qui permettent de formuler des prédictions statistiques à propos de phénomène sociaux ou économiques à partir d'un historique de données représentatives. Elle permet de construire dynamiquement des modèles prédictifs directement à partir de jeux de données plutôt que l'élaboration manuelle de règles métiers figées.

L'approche de l'analyse prédictive se démarque en particulier de l'approche de l'informatique décisionnelle traditionnelle en ce qu'elle cherche à exploiter toutes sortes de données et pas seulement des indicateurs spécifiquement collectés à des fins d'analyse. Les compétences nécessaires à sa mise en œuvre sont nombreuses et situées à la confluence de plusieurs disciplines : IT, intelligence artificielle, analyse statistique marketing. Un nouveau métier apparu ces dernières années recouvre l'essentiel de ces compétences : celui de data scientist. Cette deuxième partie a pour vocation de présenter les différentes facettes de ce nouveau métier.

- Le chapitre 5 propose **un survol global du métier de data scientist** en essayant d'aller au-delà du simple buzz marketing. On y décrira en particulier la place d'un data scientist dans une organisation, de même que les moyens qui existent pour recruter ou construire ces nouvelles compétences. Enfin, un séquencement typique des différentes tâches d'un data scientist sera proposé. Trois de ces étapes sont particulièrement importantes et seront décrites dans les chapitres qui suivent.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

- Le chapitre 6 décrit en détail la **phase de préparation des données** qui vise à rendre statistiquement exploitable l'ensemble des données que l'on peut recueillir dans un SI ou à l'extérieur à des fins de prédictions. Les sources de données les plus courantes et les outils disponibles pour mener à bien ces tâches préparatoires seront présentés.
- Le chapitre 7 entre dans le vif du sujet et présente les principaux **concepts et les méthodes du machine learning**. Il précisera ce que l'on entend par prédictions lorsque celles-ci concernent des phénomènes sociaux ou économiques. Les principes sur lesquels reposent les huit principaux algorithmes utilisés dans ces contextes seront décrits en termes intuitifs mais avec un souci de clarté conceptuelle.
- Le chapitre 8 aborde la problématique de la **visualisation des données** qui est transverse à toutes les étapes du travail d'un data scientist. Le problème de fond ici est celui de savoir quelles sont les bonnes pratiques pour représenter de manière intelligible sur un écran les données du Big Data qui sont en général multidimensionnelles.

Copyright © 2015 Dunod.

5

Le quotidien du data scientist

Objectif

Data scientist est un métier récent, apparu il y a quelques années dans le sillage du « Big Data », et dont les contours sont encore assez flous. De très nombreux débats ont lieu sur l'éventail des compétences qui le caractérisent, sur la différence avec les profils de statisticiens « classiques », sa place dans l'organisation, son quotidien. Beaucoup le voient comme un « mouton à cinq pattes » ou un être imaginaire issu du *buzz marketing*. Il y a pourtant des tendances qui se dégagent et un vrai métier qui émerge.

5.1 DATA SCIENTIST : LICORNE OU RÉALITE ?

Statisticiens, informaticiens, scientifiques, ingénieurs, analystes, ils sont nombreux à exploiter des données informatiques depuis des décennies. Une nouvelle catégorie d'utilisateur de données a fait son apparition depuis la fin des années 2000 : le **data scientist**.

5.1.1 L'origine du terme data scientist et définitions courantes

Le terme de *data scientist* s'est fait connaître à partir de 2008 quand deux employés de Facebook et Linkedin, DJ Patil and Jeff Hammerbacher, se sont attribués le titre de « data scientist ».

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

L'usage du terme s'est généralisé à partir d'octobre 2012 lors de la publication d'un désormais célèbre article « *Data Scientist: The Sexiest Job of the 21st Century* »[1]. Le rôle du data scientist gagne en importance dans l'entreprise, devient indispensable dans le contexte du Big Data et de l'explosion du volume de données non structurées. Pourtant, les profils ayant les compétences nécessaires sont rares et les formations encore peu nombreuses. Tout le monde s'accorde à dire que lors des dix prochaines années le profil de data scientist sera très recherché, un peu à l'image des analystes financiers dans les années 1980 et 1990 à Wall Street.

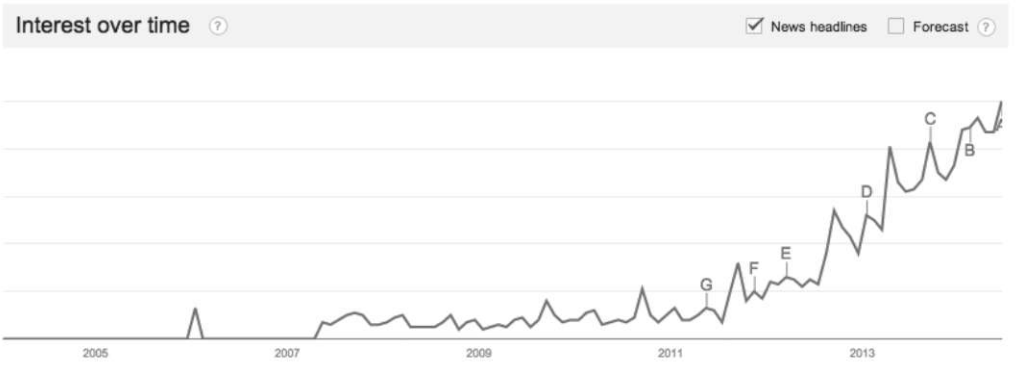


Figure 5.1 — Google Trends – Décollage de l'usage du terme « data scientist » depuis 2012.

L'histoire du data scientist chez LinkedIn

Jonathan Goldman est arrivé chez LinkedIn, le réseau social professionnel, en juin 2006. Le site était en pleine croissance et comptait à l'époque environ 8 millions de membres. Pourtant, l'expérience sociale était assez pauvre sur le site, peu de membres entrant en contact avec leurs connaissances professionnelles ou leurs collègues. Suite à son arrivée, J. Goldman a commencé à analyser les connexions existant entre les membres. L'idée était de fouiller dans l'ensemble des bases de données du réseau des clés pour expliquer l'appartenance à des groupes sociaux. Petit à petit, il a perfectionné ses conjectures et a pu créer un modèle prédisant pour un membre donné à quels réseaux professionnels il appartenait. À l'époque, l'équipe technique du site s'intéressait peu à ses travaux et préférait se concentrer sur d'autres aspects du site. Travaillant seul, il a eu l'autorisation de tester ses théories sur des petits modules du site prenant la place des publicités. Goldman a alors mis en place la première version du fameux encart « *people you may know* » qui suggère à chaque membre d'entrer en relation avec d'autres membres qu'il devrait connaître (anciens collègues, plusieurs connaissances communes avec une même personne, etc.). Il n'a pas fallu longtemps avant que l'équipe technique reconnaisse l'avancée de ce module et décide de l'intégrer parmi les fonctionnalités les plus mises en avant sur le site. Le « *people you may know* » est souvent considéré comme une des principales raisons du succès de LinkedIn.

1. *Data Scientist: The Sexiest Job of the 21st Century* par Thomas H. Davenport et D.J. Patil publié par Harvard Business Review, octobre 2012.

Copyright © 2015 Dunod.

5.1.2 Les compétences clés du data scientist

Quoi de neuf ?

Ce métier est apparu dans un contexte d'évolution profonde des technologies, d'une disponibilité accrue et à bas coût de la puissance de calcul, et d'explosion des sources de données disponibles.

Les premiers data scientists « autoproclamés », à l'instar de Goldman, sont ceux à qui ont su tirer parti des technologies et des données disponibles afin d'améliorer drastiquement un pan critique de leur activité (l'engagement des membres dans le cas de LinkedIn). Ces personnes ont fait preuve d'une combinaison assez rare de compétences en maîtrisant à la fois des concepts statistiques avancés, une appropriation technique de logiciels complexes et une compréhension des enjeux métiers et stratégiques de leur entreprise.

Ils ont ainsi contribué, probablement malgré eux, à dresser un portrait-robot du data scientist, presque inaccessible et qui a découragé bon nombre d'organisations à passer à l'action au cours des trois ou quatre dernières années. Il est pourtant tout à faire possible de recruter un data scientist de nos jours et de constituer des équipes pluridisciplinaires capables d'innovation par la donnée. Nous y reviendrons un peu plus tard, mais intéressons nous d'abord à la palette type de compétences d'un bon data scientist.

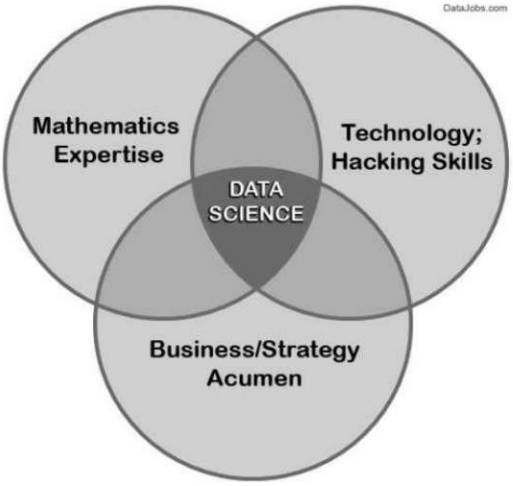


Figure 5.2 — La palette de compétences d'un data scientist se situe à l'intersection de plusieurs disciplines

Comme l'illustre le diagramme précédent ces compétences s'organisent autour de trois disciplines.

Une dimension mathématiques / statistiques

C'est évidemment à cette dimension que l'on pense en premier lieu. Des connaissances en statistiques et en algorithmes d'auto-apprentissages (« *machine learning* ») sont indispensables. Le data scientist doit être en mesure de comprendre un concept comme

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

le **niveau de signification** d'un test, de corriger des **biais**, de calculer des **probabilités**, etc.

La maîtrise des algorithmes prédictifs et de *clustering* est aussi très importante dans le cadre de son travail. L'enjeu consiste alors à choisir les bons paramètres, les bonnes variables afin d'obtenir le meilleur modèle possible. À ce stade la créativité et la curiosité sont essentielles pour incorporer les bons facteurs dans un modèle et pour dénicher de nouvelles sources de données : Open Data, API tierces, données payantes, logs, capteurs, etc.

Une des différences entre un statisticien et un data scientist dans son acception moderne est que ce dernier accorde moins d'importance à la pureté statistique d'un indicateur ou d'un algorithme qu'à son utilité « business ». Il y a d'ailleurs de grands débats dans les communautés de statisticiens et de data scientists sur les notions de niveau de signification[1] et d'interprétabilité des algorithmes.

Nous reviendrons sur les algorithmes et leur optimisation dans le chapitre 7 de ce livre.

Une dimension technologique / informatique

C'est ici que le terme data scientist prend tout son sens.

Dans son travail, le data scientist se contente rarement des logiciels traditionnels : tableurs, suites de BI traditionnelle (reporting/exploration) ou même de logiciels d'analyse de données et de statistique. L'explosion des volumes de données et la disponibilité de nombreux frameworks open source destinés à opérer des transformations et des enrichissements complexes à grande échelle sur les données (*Hadoop*, *Spark*, *Python*), l'amène à devoir utiliser un éventail de technologies et de langages de programmation bien plus vaste que par le passé.

Ainsi, une de ses compétences clés est sa capacité à **programmer** afin de s'affranchir des limites logicielles. Il maîtrise généralement plusieurs langages de programmation pour réaliser ses tâches courantes sur les données : récupération, agrégation, nettoyage, transformation, prototypage et modélisation. Les grands volumes de données n'effraient pas le data scientist, il est familier des problématiques de **passage à l'échelle** (*scaling* en anglais). Il connaît les limitations techniques d'une machine, sait paralléliser un traitement sur un ensemble de machines (« cluster ») quand il le faut, appréhende la performance des algorithmes sous un angle informatique avec un souci d'optimiser les temps de calculs.

Pourtant, il s'éloigne des développeurs informatiques sur un point : il ne connaîtra pas un langage dans tous ses rouages, il est plutôt un « hacker » c'est-à-dire qu'il aura tendance à bricoler, à prototyper, à se débrouiller pour obtenir ce qu'il veut, coûte que coûte. En effet, l'écosystème du Big Data et de la Data Science est encore en construction. Il existe une multitude de technologies qui ne sont pas toujours compatibles entre elles. Les données sont par ailleurs souvent stockées ou formatées de manière hétéroclite si bien que leur assemblage nécessite des jonglages permanents.

1. « Effect versus *p*-value » pour les initiés

Copyright © 2015 Dunod.

Tout cela motive ce profil de « bidouilleur » et de « détective » du data scientist qui doit arriver à ses fins parfois au détriment de la manière ou des conventions de programmation classiques.

Une dimension métier

Il serait probablement exagéré d'ériger cette dimension « métier » en caractéristique nouvelle du métier de data scientist. Les statisticiens en effet ont toujours eu pour objectif d'optimiser certaines facettes du métier de leur entreprise. Pourtant cette dimension ne fait que croître dans un contexte Big Data et on attend énormément d'un data scientist sur ce point. Comprendre le métier, analyser les enjeux commerciaux de son secteur et de son entreprise font partie de ses attributions. Confronté à de nombreux chiffres et statistiques, il doit en outre être en mesure de comprendre les échelles utilisées, de saisir les subtilités pour en extraire les informations pertinentes pour son entreprise.

Un data scientist a également vocation à construire des applications ou des « *data products* » utilisables en production, de prototyper rapidement et d'expérimenter des solutions en mode agile pour obtenir un feedback rapide des utilisateurs. Il est résolument tourné vers l'action et ne se contentera pas d'effectuer une simple étude sur des données historiques. Un système de recommandation de produits n'a pas de raison d'être s'il demeure à l'état expérimentation interne, il faut le confronter aux clients, le lancer en production et l'améliorer en temps réel.

Le data scientist doit également être un bon communicant. Les structures de données qu'il manipule peuvent être complexes à appréhender. Il doit par conséquent être capable de les synthétiser pour convaincre ses collaborateurs et, plus encore, son management, de la pertinence de ses analyses. Il ne pourra pas se contenter d'envoyer un simple fichier Excel abscon à ses collaborateurs mais devra créer des graphiques, des présentations dynamiques et même des animations, on parle alors de « *story telling* ».

Il travaille également en collaboration avec les décideurs et managers et les assiste dans leurs prises de décision.

Alors mouton à cinq pattes ?

Le data scientist se démarque des profils tels que celui des développeurs, des analystes traditionnels, des statisticiens ou des data miners, en particulier par la polyvalence que l'on attend de lui dans des domaines comme la programmation et l'analyse statistique.

Ce qui justifie plus encore cette appellation nouvelle de data scientist, c'est qu'il est amené à réaliser un travail qui était impossible il y a peu. La puissance de calcul et l'espace de stockage ayant été multipliés par plusieurs ordres de grandeur, c'est tout un monde de possibilités inédites qui s'est ouvert et qui reste largement à découvrir.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Thomas Cabrol, Chief Data Scientist chez Dataiku, décrit son rôle dans l'édition 2014 du référentiel des métiers de l'APEC.

Titulaire d'un DESS en analyse décisionnelle de la relation client à l'université de Montpellier en 2003, Thomas Cabrol occupe un poste de consultant data mining chez *TMIS Consulting*. Durant trois ans, il analyse et modélise le comportement d'achat des abonnés d'*Orange France*. En 2006, il rejoint *Catalina Marketing* en tant que sénior data miner, une société spécialisée dans le couponning. Il travaille au prototypage et au développement des solutions permettant aux distributeurs et industriels d'améliorer leur connaissance des clients et leur marketing relationnel. En 2008, recruté par *Apple Europe* comme data mining manager, il est chargé de développer l'intelligence spatiale et le géomarketing. En 2010, il encadre l'équipe data, chez *IsCool Entertainement*, une entreprise spécialisée dans le social gaming sur Facebook. Avec son équipe, il analyse le comportement de centaines de milliers de joueurs, il élabore l'infrastructure analytique avec des outils big data et met en place les outils d'analyse, de profiling et de reporting. En 2012, il participe avec trois autres spécialistes du big data à la création de Dataiku, un éditeur de logiciel spécialisé dans la Data Science, où il occupe la fonction de Chief data scientist.

En tant que data scientist, Thomas Cabrol intervient sur le développement de modèles visant à exploiter de manière opérationnelle des données volumineuses, non structurées tels que des avis de consommateurs, des parcours de navigation, des informations issues de réseaux sociaux ou des données de géolocalisation.

Son travail se distingue en particulier de celui d'un **data miner** dans le sens où il utilise directement Hadoop pour faire des calculs et pour prototyper la recommandation d'un produit. Il maîtrise des langages comme *Python* et *Ruby* pour la transformation des données.

Son travail se distingue également de celui d'un **spécialiste BI** qui a pour objectif de produire des rapports. Il se focalise davantage sur l'innovation au niveau des produits, des fonctionnalités et des services à partir des données brutes.

Selon Thomas Cabrol, le concept d'équipe pluridisciplinaire de trois ou quatre spécialistes permet d'avoir des compétences optimisées : « Mes collègues sont experts de l'architecture de plateforme, alors que je suis davantage orienté vers l'analyse et l'extraction des données et des connaissances à partir des données. Dans le big data, il faut être à la fois spécialiste de l'architecture des données, du développement informatique, de l'algorithmique, de la modélisation statistique, du machine learning mais également du marketing pour anticiper sur les besoins des clients. »

Face aux (trop) nombreuses compétences du data scientist, mais surtout face à la nouveauté du profil, une sorte de mythe s'est créé autour de ses compétences et même de son existence. James Kobielus, Big Data Evangelist chez IBM, a dressé en 2012 une liste des mythes existants autour du data scientist afin de mieux les briser. Ainsi il affirme : « *Non ce ne sont pas des licornes, ce ne sont ni des intellectuels, ni des thésards qui ont échoué mais tout simplement des personnes avec des compétences en statistiques acquises au cours de leurs parcours scolaire. Ils utilisent Hadoop, des modèles prédictifs et des graphes et c'est généralement ce qui les distingue des analystes BI. Enfin, ils ont pour objectif de créer des applications métiers. Les modèles prédictifs et les technologies qu'ils utilisent sont en*

Copyright © 2015 Dunod.

permanente évolution, ce qui les amène à travailler sur un mode proche de celui de chercheurs en sciences expérimentales, par élaborations successives de prototypes, donnant parfois la fausse impression de se faire plaisir avec les dernières technologies en vogue. »

5.1.3 Comment recruter ou se former

Ce profil est aujourd'hui très recherché par les entreprises, le buzz autour du Big Data et des data sciences ayant précipité la chasse aux profils quand bien même les formations et les diplômes étaient encore sous-dimensionnés voire inadaptés. S'agissant d'un métier nouveau, les profils expérimentés sont par nature rares. Qui donc peut aujourd'hui sérieusement se prévaloir de dix années d'expérience sur Hadoop ou les outils NoSQL ? Personne ! Dans les data sciences plus encore que dans les autres branches de l'IT, le célèbre adage d'Einstein « *L'imagination est plus importante que le savoir.* » est parfaitement de mise. Recruter un candidat data scientist sur la seule base d'une longue liste d'API, de sigles et de solutions est donc une aberration, si tant est que cette démarche n'ait jamais eu un sens.

Les filières académiques

Les choses cependant évoluent et de plus en plus de data scientists juniors sortent de filières académiques dédiées. Voici quelques formations en France :

- **Telecom ParisTech** : master Spécialisé Big Data
- **Université Pierre et Marie Curie** : filière Big Data du Master de Mathématiques et Applications
- **ENSAE** : spécialisation Data Science
- **ENSAI** : master Big Data
- **ENS** Cachan : M2 MVA Mathématiques / Vision / Apprentissage

À noter qu'il existe d'autres formations dans des filières mieux établies telles que l'intelligence artificielle, qui permettent elles aussi d'acquérir des compétences en *machine learning* par exemple.

À n'en pas douter, le nombre de filières va exploser dans les années à venir et il sera de plus en plus facile de trouver des data scientists juniors. Contrairement à ce que l'on pourrait penser, ces profils peuvent se révéler très productifs en peu de temps, car ils maîtrisent souvent mieux les technologies et langages de scripting comme *Python* ou *R* que beaucoup de leurs aînés. Certains d'entre eux participent activement, parfois victorieusement, à des challenges internationaux de data science comme ceux proposés sur le site kaggle.com ou datascience.net par exemple.

Les formations professionnelles

De plus en plus d'entreprises proposent des formations sur ces sujets. Il est possible de se former à une technologie en particulier, comme Hadoop par exemple, à des méthodes statistiques, à des langages (par ex. *R* ou *Python*) afin de compléter sa formation initiale. Certains acteurs spécialisés ont un catalogue particulièrement étoffé. Cependant on

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

ne saurait trop insister sur l'importance de mettre activement cet apprentissage en pratique. Pour ceux qui ont quitté les bancs de l'école depuis belle lurette, l'un des meilleurs moyens pour franchir le pas consiste à participer à des concours de data science.

L'autoformation fait partie intégrante du métier de data scientist. Qu'il ait reçu ou non une formation spécifique, il devra constamment mettre à jour ses connaissances sur les nouvelles technologies, les nouveaux algorithmes ou de nouvelles méthodes.

De nombreuses ressources permettent d'apprendre en ligne. Par exemple, plusieurs MOOC (cours en ligne ouverts à tous) sur les plateformes Coursera et edX sont disponibles. Un cours en ligne fait d'ailleurs figure de référence : « *L'apprentissage automatique* » de Andrew Ng de l'université de Stanford[1].

La participation à des concours

Les concours en ligne se sont multipliés ces dernières années. L'objectif est de trouver les meilleurs modèles prédictifs pour résoudre des cas business réels, le plus souvent avec des récompenses à la clé. Un exemple célèbre est le concours lancé par Netflix en 2007. L'entreprise américaine qui loue des films en ligne proposait un prix de un million de dollars pour la personne ou l'équipe qui proposerait le meilleur système de recommandation de films. Le prix a été remporté en 2009 par une équipe de sept ingénieurs. Les enjeux pour les candidats, outre l'appât du gain, sont de se former sur des cas réels et de se faire remarquer par les employeurs.

Les reconversions

Les data scientists expérimentés sont naturellement plus difficiles à trouver. Ils sont issus pour la plupart de filières statistiques ou *data mining* après quoi ils ont choisi de se spécialiser dans les disciplines propres au Big Data. Parmi ces profils on trouve également des administrateurs de bases de données (DBA), des développeurs spécialisés dans la donnée, des profils *Business Intelligence* ou *Web Analytics* reconvertis.

Enfin, on peut élargir le cercle à des profils scientifiques tels que les mathématiciens, physiciens ou biologistes qui, dans le cadre de leur travail ou de leurs recherches, ont été amenés à développer des compétences proches de celle nécessaires au Big Data. Le séquençage de l'ADN est un exemple.

5.2 LE DATA SCIENTIST DANS L'ORGANISATION

5.2.1 Le data lab – une clé pour l'innovation par la donnée

Les attentes en termes de compétences vis-à-vis des data scientists placent souvent la barre assez haut comme on vient de le voir. Trop haut même, à tel point que beaucoup d'organisations préfèrent aujourd'hui constituer de petites équipes pluridisciplinaires,

1. Disponible sur le site de « coursera » : https://www.coursera.org/course/ml

Copyright © 2015 Dunod.

hautement qualifiées et réactives, plutôt que d'essayer de recruter à prix d'or un data scientist « super-star ». C'est l'idée même de constituer un **data lab**.

Selon la taille de l'entreprise, le data lab, peut être constitué d'une ou plusieurs équipes travaillant en **mode agile** sur des projets stratégiques, innovants et créateurs de valeur pour l'entreprise à court et moyen terme. Il est légitime de parler de data lab dès lors que deux ou trois profils complémentaires s'associent. Dans les plus grandes organisations les plus matures, la taille d'un data lab pourra se chiffrer en dizaines d'individus. On compte dans ses rangs tous les data scientists au sens large du terme. Chacun d'eux pourra avoir des sensibilités et des compétences différentes :

- Des **architectes logiciels** qui conçoivent les systèmes prédictifs.
- Des **analystes métiers** qui contribuent à l'identification des nouveaux cas d'utilisation.
- Des **data scientists** au sens strict du terme qui prennent en charge l'optimisation des processus d'apprentissage automatique et la conception des modèles prédictifs.
- Des **développeurs** *back* et/ou *front* chargés de réaliser les systèmes conçus par les architectes et les data scientists.
- Des **designers web** qui élaborent des représentations graphiques dynamiques des résultats d'analyse.

Un point important pour constituer ces équipes pluridisciplinaires est de choisir des personnalités curieuses, ouvertes et même enthousiastes à l'idée de quitter les sentiers battus de leur discipline de prédilection.

Pour bien fonctionner, un data lab, doit en outre parvenir à s'affranchir de certaines contraintes propres aux organisations qui misent encore beaucoup sur les processus et la bureaucratie.

- Chaque membre d'un data lab doit disposer d'une large autonomie technique. À titre d'exemple, il doit pouvoir administrer les composants, qu'il développe et être à même d'installer un nouveau module *R* ou *Python* sans avoir à recourir à un lourd processus de validation, ou à passer par la lecture fastidieuse d'un long document d'architecture.
- Dans le même ordre d'idée, il convient de fluidifier au maximum les échanges entre différentes équipes et de favoriser la réutilisation de code, de scripts, de jeux de données externes utiles ou instructifs.
- Il faut mettre en place des conditions de travail qui favorisent à la fois la créativité individuelle et l'échange des idées. Ainsi on réservera des espaces calmes propices à la concentration et à la production de livrables alors que d'autres espaces seront consacrés aux échanges et au brainstorming.
- Il convient également de remettre en cause les méthodes traditionnelles de pilotage des projets, encore trop souvent inspirées du BTP, plutôt que de la réalité sociale des projets IT innovants. Par une sorte de déni de la réalité, la volonté d'élaborer des planifications détaillées l'emporte trop souvent sur la nécessité d'instaurer une culture de l'expérimentation et du prototypage. De

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

toute évidence, la complexité, l'instabilité et la multiplicité des technologies utilisées dans les contextes Big Data rendent aujourd'hui une telle planification illusoire. L'échec doit être envisagé comme une étape normale et inévitable de l'acquisition de connaissances par l'expérimentation.

5.2.2 La data lab – quelle place dans l'organisation ?

Une fois que la mise en place d'un data lab est acquise, se pose encore la question de son rattachement. Sera-t-il rattaché à l'un des départements métier ? À l'IT ? À l'innovation ?

Dans l'idéal, il devrait, par sa nature transdisciplinaire, être transverse à l'organisation. Il faut être conscient cependant que ce statut peut aussi le rendre relativement inopérant et sans pouvoir réel sur l'orientation stratégique des départements métiers. L'une de ses principales missions, rappelons-le, est la réalisation de prototypes qui, souvent, devront être testé dans des conditions réelles. Il est dès lors crucial d'obtenir l'adhésion des métiers qui bénéficieront d'autant mieux des innovations du data lab qu'ils y auront apporté leurs contributions.

Même s'il n'a pas un statut transverse, un data lab devrait être sponsorisé au niveau d'un comité de direction ou d'un comité exécutif. Le rôle de « **Chief data officer** » apparu ces dernières années est précisément de faciliter le déploiement des innovations à toute l'entreprise.

5.3 LE WORKFLOW DU DATA SCIENTIST

Le travail d'un data scientist consiste à concevoir et à réaliser de bout en bout le prototype d'un nouveau produit qui met en œuvre les techniques d'analyse prédictive. De la conception d'un nouveau service au développement du prototype, en passant par la collecte et le nettoyage des données, il devra se montrer autonome techniquement, faire preuve d'imagination et être à l'écoute des besoins métiers qu'il devra formaliser pour les implémenter. Les six paragraphes qui suivent décrivent chacune les activités d'un data scientist ou, si l'on préfère, celles d'un data lab. La succession des paragraphes qui suivent correspond d'avantage à un enchaînement logique des activités qu'à une stricte succession chronologique, étant donné le caractère itératif de l'élaboration d'un service d'analyse prédictive.

Certains termes propres au *machine learning* seront utilisés ici de manière informelle mais, que le lecteur se rassure, les concepts associés seront présentés en détail au chapitre 7 assortis de définitions en bonne et due forme.

Plusieurs sujets esquissés dans les sections qui suivent seront traités en détail dans des chapitres dédiés au vu de leur importance : la préparation des données au chapitre 6, la modélisation au chapitre 7 et la visualisation au chapitre 8.

Copyright © 2015 Dunod.

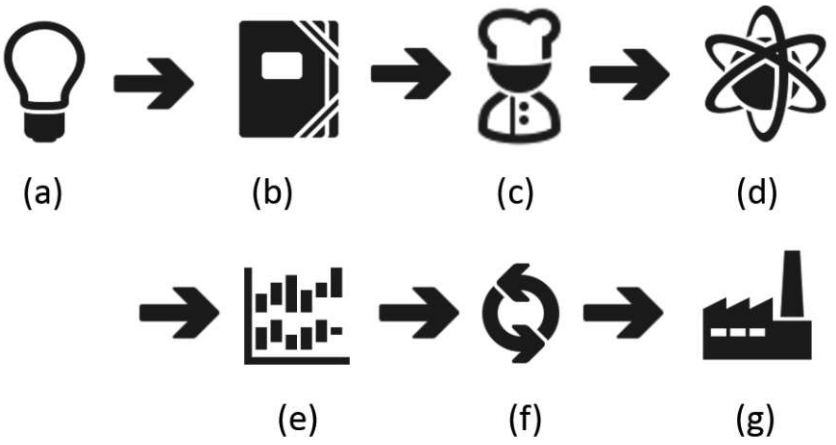


Figure 5.3 — Le workflow typique d'un data scientist : (a) imaginer un produit, (b) collecter les données, (c) préparer les données, (d) concevoir un modèle prédictif, (e) visualiser les résultats, (f) optimiser le modèle, (g) déployer et industrialiser.

5.3.1 Imaginer un produit ou un service

Durant cette première phase il s'agit de passer d'une description informelle d'un besoin ou d'une opportunité métier à une formulation plus rigoureuse susceptible d'être implémentée dans un modèle prédictif. Le data scientist devra en l'occurrence « penser produit » en se posant tout d'abord la question : que veut-on impacter au niveau des métiers ? De la réponse apportée à cette question dépendra en particulier la définition d'une (ou plusieurs) **variable**(s) **cible**(s) que l'on souhaitera prédire au moyen de différentes **variables prédictives** avec lesquelles elle est corrélée (voir chapitre 7). La définition d'une variable cible est parfois moins évidente qu'il n'y parait, comme l'illustre l'exemple suivant.

Évaluation du désintérêt (ou « churn ») d'un abonné à un service

Prenons le cas d'un service en ligne réservé à des abonnés dont le nombre est déterminant pour la pérennité du service. Détecter au plus tôt les abonnés tentés d'abandonner le service est dès lors crucial. Comme nous le verrons en détail au chapitre 7, la plupart des modèles prédictifs (ceux que l'on dit « **supervisés** ») sont alimentés au moyen d'un historique de données à partir duquel on va essayer d'extraire un modèle prédictif. Dans la situation qui nous intéresse, cet historique devrait comporter deux types d'information.

D'une part la description des comportements des abonnés (le nombre d'achats qu'ils effectuent, le nombre de simulations d'achats n'ayant pas abouti à une commande, la fréquence des visites du site etc.) et, d'autre part, l'information indiquant s'ils ont abandonné le service ou non.

C'est là que réside la difficulté : comment en effet déterminer de manière objective, à un instant donné, si abonné se désintéresse ou non du service ? La résiliation pure et simple, en effet, sanctionne souvent une décision intervenue beaucoup plus tôt. Dès lors, pour associer une variable cible au désintérêt d'un abonné, le data scientist devra concevoir une approximation de la variable cible « *l'abonné x se désintéresse du*

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

service » qui restera objectivement indéfinissable. C'est là qu'intervient la connaissance détaillée du métier. Selon les situations, on pourra décider, a posteriori, qu'un individu qui a résilié son abonnement à une date donnée a en réalité cessé de s'intéresser au service trois, six ou neuf mois plutôt. Alternativement, on pourrait considérer qu'un délai entre deux visites successives qui s'allonge au-delà d'un certain seuil caractérise un individu en passe d'abandonner le service.

Une fois que la variable cible (ou le proxy de cette variable) a été déterminée, reste à identifier des variables prédictives, celles dont l'observation sur les nouveaux abonnés va permettre de prédire leur comportement ultérieur. Là encore, une connaissance détaillée du métier ou de la psychologie des clients sera déterminante. À cette occasion, le data scientist sera amené à travailler en étroite collaboration avec les équipes métiers et avec le marketing.

Cette première étape comporte donc une part importante d'innovation. Il s'agit d'imaginer de nouveaux produits, de nouvelles utilisations, de combiner entre elles des sources de données encore inexploitées pour leur conférer une valeur ajoutée.

Il faudra également imaginer comment on va mesurer le succès d'une prédiction. Quels tests statistiques seront utilisés ? À partir de quel seuil de signification[1] décidera-t-on que le modèle prédictif est vraiment efficace ?

« Prepare to fail ![2] »

Durant cette phase il pourra s'avérer nécessaire, si ce n'est déjà fait, de sensibiliser le management au fait que l'élaboration d'un modèle prédictif implique de travailler sur un mode expérimental et qu'une part significative du code développé à cette occasion sera jetable. La réticence du management à abandonner les démarches traditionnelles, rigoureusement et illusoirement planifiées, au profit d'une approche plus agile et plus expérimentale de l'IT, qui n'est que la contrepartie de l'innovation authentique, reste à l'heure actuelle l'un des principaux freins aux projets de data science.

5.3.2 Collecte des données

La collecte des données est une phase qui varie considérablement d'un projet à un autre, aussi est-il hasardeux de proposer une démarche standard qui aurait vocation à s'appliquer à toutes les situations. Tout au plus peut-on lister les questions qu'il faudra traiter à cette occasion.

Copyright © 2015 Dunod.

1. Le seuil ou niveau de signification désigne en statistique les probabilités que l'effet que l'on observe soit dû au hasard. Ainsi un niveau de signification de 5% caractérise un phénomène qui n'a qu'une chance sur vingt d'être le fruit du hasard.
2. Préparez-vous à échouer !

Disponibilité des données

Après avoir imaginé les variables prédictives et la variable cible sur lesquelles se basera le modèle prédictif, encore faut-il s'assurer de la disponibilité effective de ces données. Sont-elles accessibles en volume suffisant ? Si oui, à quel coût ? Peut-on les acheter auprès d'organismes tiers, d'opérateurs, de fournisseurs de fichiers marketing, de recensements ou de listes de sinistres ? Existe-t-il des sources « open data » ?

L'historique des données s'étend-il sur une durée suffisante pour déterminer les proxys de variable cible telle que nous l'avons envisagée au paragraphe précédent ?

Qualité des données

Étroitement liée à la première question, la qualité des données revêt plusieurs facettes. La précision des données est-elle suffisante ? A-t-on identifié les sources d'erreurs pour pouvoir les corriger le cas échéant ? Les données sont-elles affectées par un biais qui risque de fausser les prédictions ?

Exemple : évaluation de la disponibilité de places de stationnement

Imaginons en guise d'illustration de la notion de biais qu'un système prédictif ait pour objectif de prédire la disponibilité des places de stationnement dans une grande agglomération. Parmi les variables prédictives facilement accessibles on pourrait utiliser par exemple le nombre de tickets délivrés par les automates. Hélas, à bien y réfléchir, ces relevés risquent de ne pas correspondre au taux d'occupation réel des places de stationnement. Ceci, pour toute une série de raisons : beaucoup d'automobilistes partent avant ou après l'expiration effective de leur ticket, certains prennent un ticket à un endroit mais se garent ailleurs, d'autres ont des tickets résidents, d'autres encore fraudent.

Pour minimiser les problèmes de biais, il est indispensable de disposer de données suffisamment riches et représentatives du phénomène que l'on souhaite prédire. Ce problème du biais prend d'autant plus d'importance que les évènements que l'on souhaite prédire sont rares.

Enfin, parmi les données que l'on pourra récupérer il faut évaluer la proportion de données manquantes. Pourra-t-on y remédier en croisant les informations disponibles avec d'autres sources de données ?

Questions techniques

La récupération de données peut poser des questions liées aux formats de données, aux technologies qui permettent de les extraire et aux compétences des personnes chargées de leur collecte.

Enjeux juridiques

Les données que l'on envisage d'utiliser sont-elles libres de droit ? Risque-t-on d'enfreindre la législation, notamment au niveau du respect de la vie privée, par la

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

désanonymisation que rendent possible les recoupements entre jeux de données multiples ? Existe-t-il des problèmes liés à la localisation géographique où seront stockées les données convoitées ?

Enjeux politiques

La possession de données est bien souvent liée à des enjeux de pouvoir au sein des entreprises. Une utilisation non concertée risque par conséquent de générer des conflits. Ainsi un département marketing ne voudra-t-il pas divulguer ses données les considérant comme un trésor de guerre. Les équipes IT, soucieuses de ne pas déstabiliser les applications dont elles ont la responsabilité, seront quant à elles réticentes à ouvrir les accès sans connaître par avance comment et par qui ils seront utilisés.

5.3.3 Préparation

Une fois les données récupérées auprès des différentes sources il faudra encore les rendre utilisables par les algorithmes d'apprentissage tel que ceux que nous décrirons au chapitre 7.

Ces opérations de préparation seront décrites en détail au chapitre 6. Mentionnons ici qu'il s'agit d'**homogénéiser les formats** des différentes sources, de **nettoyer les données** pour supprimer les enregistrements qui comportent des **données manquantes** ou alors pour les combler au moyen de calculs appropriés ou de sources tierces.

Des opérations de **mise à l'échelle** destinées à harmoniser les unités utilisées pour les différents paramètres pourront faire partie de ce travail de préparation.

Les opérations de **croisements des données** initiales avec d'autres sources interviennent à cette étape. Des données comportementales tirées de logs de sites web pourront ainsi être croisées avec des données d'achats tirées d'une base transactionnelle.

5.3.4 Modélisation

La modélisation procède le plus souvent de manière itérative avec beaucoup de tâtonnements. On peut cependant distinguer deux tâches principales.

Il s'agit premièrement de choisir un nombre restreint de variables qui seront utilisées pour alimenter les algorithmes d'apprentissage, on les appelle les **variables prédictives**. L'activité qui consiste à utiliser l'expertise métier pour imaginer de nouvelles variables qui amélioreront la qualité de prédiction d'un modèle s'appelle le « **feature engineering** ». C'est principalement à ce niveau qu'interviennent la créativité et la connaissance métier des data scientists qui considèrent pour cette raison cette activité comme la « partie noble » et gratifiante de leur métier, par contraste avec les préoccupations très terre à terre de la préparation des données. Lorsque le nombre de ces variables est trop élevé (plus de dix), une phase dite de **réduction dimensionnelle** peut s'avérer nécessaire. Différentes possibilités s'offrent au data scientist pour réduire le nombre de paramètres utilisé dans ses modèles prédictifs, elles seront décrites succinctement dans la section 7.3.9.

Copyright © 2015 Dunod.

Il s'agit ensuite de choisir un **algorithme d'apprentissage** adapté à la situation. Le chapitre 7 décrira les huit algorithmes les plus couramment utilisés par la communauté de l'analyse prédictive. L'intuition du data scientist face à une situation particulière et l'expérience cumulée de la communauté des statisticiens jouent ici un rôle prépondérant dans ce choix. Même dans les situations où il est difficile de justifier rationnellement le choix d'un algorithme au détriment d'un autre, il existe des usages et des pratiques qui ont fait leurs preuves dans des situations similaires à celle que l'on examine.

Les questions que l'on pose à cette étape sont les suivantes : s'agit-il de prédire un nombre décimal, un prix par exemple, ou s'agit-il plutôt de prédire une catégorie comme l'appartenance d'un individu à une classe à risque ? Souhaite-t-on mettre en place un **apprentissage au fil de l'eau** (aussi dit « on line ») dont les performances augmenteront au fur et à mesure que s'accumuleront les données (comme dans les systèmes de recommandation ou les filtres antispam) ? Ou alors un mode d'**apprentissage statique** (aussi dit hors-ligne), dans lequel on entraîne un modèle avant de l'utiliser, pourra-il faire l'affaire ? A-t-on par avance une idée du type de relation qui lie les variables prédictives à la variable cible ? Un exemple élémentaire est fourni par la régression linéaire où l'on anticipe que cette relation est linéaire. On parle alors de **modèle paramétrique**. Dans le cas inverse, lorsque l'on n'anticipe aucune forme particulière, on parle d'un **modèle non paramétrique**.

L'**interprétabilité** d'un algorithme pourra faire partie des critères à retenir dans le choix d'un algorithme. Dans les situations où il est important, non seulement de prédire un phénomène, mais de comprendre les mécanismes à l'origine des corrélations entre variables prédictives et variables cibles on pourra privilégier l'interprétabilité au détriment de la précision.

Dicton : « *Better data outweighs clever maths*[1], ce dicton résume bien la sagesse acquise par les data scientists au sujet de la modélisation prédictive. En termes plus explicites : un bon jeu de données l'emportera souvent sur des algorithmes très sophistiqués.

5.3.5 Visualisation

Dans la panoplie de compétences que l'on attend d'un data scientist ou, de manière plus réaliste, d'un data lab, la communication n'est pas la moins importante. Le langage des statistiques et des algorithmes, au cœur de son propre métier, n'est ni celui des experts métiers et encore moins celui des clients auxquels sont destinés les services qu'il conçoit. Il devra par conséquent faire œuvre de pédagogie et être à même d'adapter sa communication en fonction des interlocuteurs auxquels il s'adresse. La visualisation des données sera son meilleur atout pour rendre palpables les intuitions et les conclusions qu'il tire de ses analyses statistiques : « une image vaut mille mots » dit le dicton.

1. De meilleures données pèsent plus lourd que des mathématiques sophistiquées.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

Pour fixer les idées, on peut identifier quatre catégories d'interlocuteurs ou quatre niveaux d'abstractions qu'un data scientist devra pratiquer pour visualiser ses analyses :

1. Dans ses échanges avec les **développeurs web**, chargés de réaliser des maquettes interactives des futurs services d'analyse prédictive, il sera amené à utiliser un vocabulaire technique de type HTML5, JavaScript ou à comprendre ce qu'apporte une librairie comme D3.Js. Sans être un spécialiste de ces sujets techniques, des rudiments de connaissance lui seront utiles pour envisager de manière réaliste les meilleures manières de réaliser le rendu final des services prédictifs qu'il imagine.
2. Dans ses échanges avec les **utilisateurs métiers** il utilisera de préférence pour présenter ses analyses des représentations graphiques, aussi bien statiques (histogrammes, boîtes à moustaches, diagrammes de dispersion) que dynamiques (utilisation d'outils comme tableausoftware.com). Il pourra également avoir à interagir avec ces mêmes experts pour concevoir des services *back office* comme ceux qui leur permettront par exemple d'évaluer l'appétence des clients pour un nouveau produit.
3. Même s'il n'interagit pas directement avec les **clients**, le data scientist devra « penser produit » pour imaginer à quoi ressemblera une application *front* sur le smartphone du client. On peut imaginer par exemple une application qui aidera un client souscripteur d'un service payant à garer sa voiture en fonction de la disponibilité des places de parking.
4. Enfin avec ses pairs enfin, un data scientist utilisera également des représentations graphiques. Pour représenter la performance prédictive d'un algorithme ou pour évaluer le degré de corrélation entre variables prédictives, et par là leur utilité, il existe des représentations graphiques spécifiques. Nous présenterons les plus courantes au chapitre 8.

> La visualisation est une activité omniprésente dans l'élaboration d'applications prédictives. Elle intervient à toutes les étapes : de la préparation des données à l'optimisation du système prédictif en passant par la modélisation statistique.

5.3.6 Optimisation

L'optimisation d'un système prédictif procède naturellement de manière itérative. Les **démarches agiles** sont ici toutes indiquées. L'objectif qui doit guider le data scientist, ou le data lab, est la réalisation rapide, de bout en bout, d'une solution simple et fonctionnelle du service envisagé. La complexification intervient par la suite de manière incrémentale après validation par toutes les parties prenantes du projet.

Parmi les tâches d'optimisation figurent les ajustements des paramètres d'algorithmes d'apprentissage :

Copyright © 2015 Dunod.

- Test de différentes valeurs des paramètres des modèles[1] pour optimiser la performance de généralisation d'un algorithme, voir chapitre 7.
- Optimisation de la **taille des échantillons** pour raccourcir le temps d'apprentissage tout en préservant la précision des prédictions.
- Ajustement de la **complexité** (ou de la flexibilité) **d'un modèle** pour éviter le surapprentissage, voir chapitre 7.
- Exclusion de certaines **valeurs aberrantes** qui biaisent les prédictions d'un modèle.
- Omission de variables peu prédictives.
- Création de nouvelles variables plus prédictives (**feature engineering**).
- Pénalisation des observations trop bruitées.

Comme pour toute tâche créative, le data scientist ne pourra jamais considérer une analyse comme définitive mais devra, au contraire, la challenger en permanence pour réexaminer un même phénomène sous des angles différents.

Une part de l'activité d'optimisation recouvre celle déploiement d'une application prédictive en production (voir ci-dessous). Les premiers tests sur une population restreinte révèlent parfois des biais insoupçonnés auxquels il faudra remédier. Un exemple classique est celui de l'utilisation, pour l'entraînement d'un modèle prédictif, de variables dont on découvre a posteriori qu'elles ne sont pas disponibles au moment où l'on souhaite faire les prédictions.

5.3.7 Déploiement

Les deux principaux problèmes à résoudre lors d'un déploiement en production d'une application prédictive sont le **passage à l'échelle** et l'industrialisation.

Dans des contextes big data le passage à l'échelle impliquera parfois de traiter des volumes de données de plusieurs ordres de grandeurs supérieurs à ceux utilisés durant la conception et les premiers tests. Il arrive à cette occasion qu'une partie du code doive être réécrite pour des raisons de fiabilité ou de performance. Sont concernés aussi bien les traitements utilisés lors la préparation des données que les mécanismes d'apprentissage et de prédiction proprement dits. Ce travail de réécriture (passage de *Python* à *Java*, de SQL à *Hive*, de *Scikit-Learn* à *Mahout*, voir chapitre 9) sera en règle générale réalisé par les équipes IT plutôt que par les data scientists.

Remarquons au passage que le choix d'un algorithme d'apprentissage doit en principe anticiper ces besoins de passage à l'échelle, tous les algorithmes n'étant pas susceptibles de parallélisation.

Pour être en mesure de faire des prédictions rapides face à des situations changeantes, il faut industrialiser autant que possible la chaîne de traitement qui vient d'être décrite : collecte de données, nettoyage, transformation, enrichissement et représentations graphiques (sur le *front office*). Toute la chaîne des traitements, y

1. Lorsqu'il s'agit d'explorer des paramètres multi-dimensionnels on utilise le terme de *grid-search*.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

compris les différentes variantes testées, doit par conséquent être conservée pour pouvoir être rejouée sans erreur le moment venu. Des outils de productivité qui facilitent cette démarche d'industrialisation commencent à apparaître sur le marché. *Data Science Studio* de la société *Dataiku* est exemple de cette nouvelle classe d'outils.

En résumé

Dans ce chapitre nous avons présenté les multiples facettes de l'activité d'un data scientist : programmation informatique, analyse statistique, analyse métier et marketing. Ces compétences sont toutefois si nombreuses qu'elles sont rarement réunies chez un seul individu. Pour cette raison, la plupart des entreprises choisiront aujourd'hui de mettre en place un data lab que l'on peut envisager comme un data scientist polycéphale. L'activité d'un data scientist, ou d'un data lab, se déploie essentiellement dans deux directions. Sur un versant prospectif, on trouve des tâches comme : la conception de services prédictifs innovants, la veille technologique sur les outils IT, les avancées algorithmiques ou les nouvelles méthodes d'analyse statistique. Sur un versant plus opérationnel, on trouve la conception de prototypes de services prédictifs dont nous avons détaillé le workflow : collecte de données, préparation des données, modélisation, visualisation et enfin optimisation des modèles. Toujours sur le versant opérationnel, un data scientist aura également une activité de conseil auprès des équipes métiers. Il les aidera à comprendre et à anticiper des phénomènes sociaux ou économiques en apparence aléatoires.

Copyright © 2015 Dunod.

6

Exploration et préparation de données

Objectif

Ce chapitre a pour but de présenter la première étape des projets de Data Science, souvent négligée, voire mal-aimée : l'exploration et la préparation des données. L'objectif de cette étape cruciale est de transformer des données brutes, éparses, hétérogènes, de qualité souvent variable en un jeu de données de qualité homogène et mesurée qui sera exploitable pour l'analyse statistique, pour le *reporting business* ou pour l'entraînement de modèles d'apprentissage automatique qui feront l'objet du chapitre suivant.

6.1 LE DÉLUGE DES DONNÉES

Il est amusant de commencer par souligner la schizophrénie du secteur IT qui désigne un même phénomène, la quantité et la variété croissante des données disponibles pour les organisations, à la fois comme une manne encore largement inexploitée mais aussi comme un fléau quand il parle de « déluge » des données. Cet état de fait est sans doute à relier à la tension née du désir d'exploiter une nouvelle ressource et de la conscience de la difficulté de donner un sens à ce qui s'apparente souvent à un véritable fatras de fichiers Excel, d'e-mails et de fichiers CSV.

C'est le « V= variété » du Big Data auquel il s'agit de faire face. Il ne faut compter ici sur aucun miracle de la technologie mais bien davantage sur le sens du discernement et l'intuition des data scientists qui seront en charge durant une première phase

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

cruciale commune à tous les projets d'analyse prédictive de nettoyer et de préparer les données avant qu'elles ne puissent être exploitées à des fins de reporting ou d'apprentissage automatique dont il sera question au chapitre 7.

6.1.1 Diversité des sources

Voici une liste non exhaustive des sources de données auxquelles il est aujourd'hui facile d'accéder.

Les SI transactionnels et la BI d'une entreprise

Les bases de **données de production**, les entrepôts de données (*datawarehouses*) avec les **données sur les clients** (CRM[1]), les **transactions** d'achats et de ventes de produits, les **processus de gestion** (ERP[2]) sont les premières sources que va utiliser une entreprise. Toutes ces sources ont l'avantage d'être aisément accessibles et régulièrement mises à jour. De plus elles sont, du moins en principe, structurées selon des schémas rigoureux même s'il faut souvent tenir compte des incompatibilités et des redondances qui s'accumulent aux cours des années.

En contraste avec l'idée répandue qui envisage le Big Data comme l'analyse du web et des réseaux sociaux, nombre d'entreprises débutent en pratique leurs projets Big Data en exploitant leurs données internes. Souvent historisées depuis des années, celles-ci ne demandent qu'à être exploitées avec les nouveaux outils de stockage et d'analyse prédictive. Les applications sont multiples : amélioration de la stratégie de l'entreprise, économies sur l'usage de ressources, anticipation de l'avenir au moyen de modèles prédictifs. Après avoir acquis une première expérience sur l'exploitation Big Data de ces données internes une entreprise pourra ensuite se tourner vers des sources externes, soit par l'achat de bases de données commerciales soit en exploitant des données disponibles en « open data » (voir ci-dessous).

Les nouveaux entrepôts de données comportementales

On peut ranger ces sources dans trois sous-catégories :

- Les **logs** (« journal » en français) des sites internet contiennent, entre autres informations, l'historique de navigation de tous les visiteurs du site. À partir de l'ensemble des actions des utilisateurs, il est parfois possible de reconstituer l'ensemble de leur parcours de navigation. Ce sont les données essentielles pour effectuer des mesures fines d'audience (*web analytics*). Vu l'importance du sujet, le chapitre 10 sera consacré à un exemple d'analyse de logs.
- Les données issues des capteurs et autres **objets connectés** sont promises à une croissance exponentielle ces prochaines années avec l'émergence progressive de l'internet des objets. Elles aussi sont enregistrées puis historisées dans des logs.

1. CRM : *Customer Relation Management*.
2. ERP : *Enterprise Resource Planning*.

Copyright © 2015 Dunod.

- Lorsqu'elles sont publiquement accessibles, les données issues des **réseaux sociaux** comme *Facebook*, *Twitter* ou *LinkedIn* permettent d'enrichir les profils des utilisateurs. Leur valeur ajoutée tient beaucoup au fait que les utilisateurs se chargent eux-mêmes de la mise à jour de ces données.

Les données géographiques

Toutes les données avec des informations spatiales (comme le couple latitude-longitude) sont très intéressantes à exploiter. Ainsi, une entreprise pourra associer des **informations de géolocalisation** aux livraisons en cours, permettant aux clients de suivre en temps réel l'acheminement de leurs commandes.

Les **points d'intérêts** sont de la même nature. Il s'agit de lieux identifiés et géolocalisés souvent rangés par catégorie comme les restaurants, les distributeurs de billets, les pharmacies, les parkings, les écoles, les entreprises d'un secteur, etc.

Les **données socio-économiques** sont particulièrement utiles lorsqu'elles sont recoupées avec des informations géographiques : revenu par habitant par ville, densité de population, taux de criminalité par commune, etc.

Les **données météorologiques** peuvent être mises à profit pour comprendre certains comportements sociaux ou économiques : achats de vêtements, réservations d'hébergements, prévision de trafic, etc.

L'open data

De plus en plus d'entreprises ou de collectivités publient des données librement exploitables sur des domaines variés ; c'est ce qu'on appelle l'**open data**. Le site *data.gouv.fr* est une plateforme de diffusion de données publiques placée sous l'autorité du gouvernement. On y trouve les résultats des derniers recensements en France, des résultats économiques, des informations sur l'éducation, le logement ou la santé.

Les bases de données commerciales

Il est également possible d'acheter des jeux de données auprès d'entreprises spécialisées telles que des instituts de sondages, de mesure d'audience, etc. On notera à ce titre le développement récent de ***data marketplaces*** qui vendent des jeux de données à l'unité. *Microsoft Azure Marketplace*, *Datamarket*, *Data Publica* sont parmi les plus utilisés.

Les données obtenues par crawling

Une dernière source d'information est la collecte automatique de contenu de sites internet par des robots (*crawling*). À noter que les données obtenues de cette manière peuvent s'avérer très déstructurées et que cela pose par ailleurs des questions de droits, la législation actuelle laissant pour l'instant cette pratique dans une zone grise.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

6.1.2 Diversité des formats

À la diversité des sources, il faut ajouter la diversité des **formats d'échange** et des **structures de données** stockées. La question des formats d'échange et celle des structures de données devrait toujours être posée en amont d'un projet de type data science. Une situation courante est celle dans laquelle des échanges de fichiers Excel sont souhaités par des équipes métiers (car ce format leur est familier depuis des années) alors que ce format est totalement inapproprié à des projets d'automatisation à grande échelle.

Voici une liste non exhaustive de formats d'échange et de structure de données.

Les fichiers classiques

Ces fichiers sont stockés, soit localement dans le système de fichier d'un ordinateur soit dans un système de fichiers distribué de type HDFS sur Hadoop, (voir chapitres 4 et 9). Ils sont faciles à déplacer et à éditer. On distingue les principaux formats suivants :

- Les fichiers **textuels standards** (CSV, TXT...). En dépit de leur apparente simplicité, des lignes terminées par des séparateurs, les fichiers CSV posent souvent des problèmes liés au fait qu'aucun standard ne s'est dégagé pour les séparateurs, les lignes multiples, etc. Cela reste toutefois l'un des formats de stockage et d'échange les plus utilisés.
- Les **nouveaux formats** de stockage comme *Apache Parquet* ou le format de sérialisation *Apache Avro*. Ils ont pour objectif de dépasser certaines limitations des fichiers standards et de faciliter l'accès en lecture par colonne pour accélérer certains calculs analytiques.
- Les **fichiers au format objet** comme JSON ou XML. Ils se distinguent des fichiers standards par une structure et séparation des données complexe et bien définie. Ils structurent mieux les données et sont ainsi souvent utilisés pour l'échange via des API.
- Les « **shapefile** » sont des fichiers utilisés dans les systèmes d'information géographique (SIG). Ils contiennent des informations propres à la géométrie des objets cartographiques : des points, des lignes et des polygones.
- Les **fichiers multimédias** au format binaire (image, vidéo, voix) sont plus difficiles à exploiter.

Les bases de données relationnelles

L'historique et la raison de la domination des bases de données relationnelles (SGBDR) ont été décrits au chapitre 3. Rappelons ici ce qui les caractérise :

- Les données sont stockées dans des **tables** reliées entre elles par des clés étrangères.
- Les SGBDR assurent la cohérence des **transactions** constituées d'une succession d'opérations qui doivent toutes réussir sous peine de révocation.
- Ils implémentent des **contrôles d'intégrité** basés sur des schémas qui décrivent la structure des données considérées comme valide.

Copyright © 2015 Dunod.

- Enfin, les SGBDR, peuvent être interrogés au moyen d'un langage déclaratif standard, le **SQL**, qui spécifie ce que l'on cherche, le moteur de la base de données se chargeant de l'exécution d'un plan d'exécution optimisé.

Les solutions les plus courantes sont *Microsoft SQL Server*, *Oracle*, *MySQL*, *PostgreSQL*, *IBM DB2*.

Le standard SQL est aujourd'hui si bien implanté que la plupart des systèmes de stockage distribués récents essaient de proposer des langages d'interrogation qui s'en rapprochent autant que possible. C'est le cas par exemple de *HiveQL* par exemple dans l'écosystème *Hadoop* dont il sera question aux chapitres 9 et 10.

Les bases NoSQL

Les systèmes NoSQL ont fait l'objet d'une description détaillée au chapitre 3. Rappelons simplement ici que ces systèmes abandonnent le modèle relationnel pour d'autres structures de données plus flexibles comme les entrepôts clé-valeurs, les bases de données orientées documents, les bases de données orientées colonnes et enfin les bases de données orientées graphes. Tous ces systèmes sont nés des contraintes techniques et de nouvelles priorités propres aux sites e-commerce à très grande échelle (disponibilité au détriment de la cohérence stricte des données) pour lesquels les SGBDR se révèlent soit inadaptés, soit excessivement coûteux.

6.1.3 Diversité de la qualité

La qualité d'une prédiction, est naturellement tributaire de la qualité des données utilisées pour alimenter le modèle prédictif. S'il est vrai qu'à qualité égale, un gros échantillon de valeurs expérimentales est toujours préférable à un échantillon modeste, les fluctuations statistiques diminuent en effet avec le nombre d'observations, il serait toutefois erroné d'en conclure que la quantité puisse être un substitut à la qualité. Bref, le Big Data ne saurait se dispenser de tout souci de qualité des données et il faut être conscient que celle-ci à un prix, non seulement pour l'acquisition mais aussi pour l'exploitation. Des calculs précis sont par exemple plus coûteux que des calculs approximatifs.

Examinons d'un peu plus près ce que l'on entend par qualité des données qui comporte en réalité plusieurs facettes.

L'exhaustivité

Un jeu de données complet sera naturellement de meilleure qualité qu'un jeu incomplet. L'importance de la complétude varie toutefois d'une situation à l'autre. Ainsi un journal de transactions avec des clients serait rapidement incomplet s'il manquait quelques jours ou même une seule transaction dans l'hypothèse d'un usage comptable.

Le critère d'exhaustivité possède deux dimensions. D'une part il peut être question de la complétude au niveau de la liste des enregistrements. D'autre part on peut envisager la complétude du renseignement des champs de chacun des enregistrements

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

disponibles. Dans certaines situations il se peut que la seule présence d'un enregistrement, même incomplet, fasse l'affaire.

La granularité

La granularité correspond au degré de finesse des données, qu'elles soient spatiales, temporelles ou sociales. Pour des enregistrements effectués par des capteurs, possède-t-on des mesures quotidiennes, horaires ou par minute ? Des données de pression atmosphérique sont-elles disponibles pour tous les départements, toutes les communes, pour chaque kilomètre carré du territoire ? Possède-t-on des estimations de revenus pour toutes les catégories socioprofessionnelles ? Est-il possible de se contenter de données agrégées ou de résumés statistiques comme des moyennes ou des médianes ?

L'exactitude

Les informations ou valeurs enregistrées sont-elles fiables ? Les données issues des CRM sont souvent évaluées selon ce critère. Prenons l'exemple de données postales. Celles-ci peuvent être saisies par l'utilisateur lui-même. En l'absence de mécanisme de vérification automatique dans un annuaire, celles-ci seront inévitablement entachées d'erreurs. De même pour des coordonnées dictées par téléphone. Dans le cas de données alimentées par des capteurs, un certain niveau de redondance pourrait pallier les éventuelles imprécisions de l'un d'entre eux.

La fraîcheur

La fréquence de rafraîchissement ainsi que la dernière date de mise à jour sont des critères importants de qualité dans certaines situations comme dans le cas des données boursières ou de la position géographique d'un produit en cours de livraison.

Un point important à saisir est qu'il n'existe pas de mesure absolue de la qualité d'un jeu de données comme il existe une qualité absolue pour la mesure d'une grandeur physique. La qualité des données doit au contraire être adaptée à chaque situation et à chaque problème de prédiction en tenant compte des coûts d'acquisition et d'exploitation.

6.2 L'EXPLORATION DE DONNÉES

6.2.1 Visualiser pour comprendre

La phase d'analyse proprement dite démarre avec l'exploration des données. Se contenter de quelques indicateurs statistiques comme la moyenne, la médiane ou la variance des données n'est pas suffisant pour révéler la structure des données. La visualisation des données qui exploite toute l'acuité du système visuel humain est bien plus efficace dans cette phase.

L'analyste et le data scientist ont tout intérêt à multiplier les différentes représentations possibles afin d'explorer différentes pistes, de tester des combinaisons de variables, de regroupement. Au vu de l'importance de cette phase de visualisation dans le travail

Copyright © 2015 Dunod.

du data scientist, nous y consacrerons tout le chapitre 8 et montrerons en particulier au moyen d'un exemple, le *quartet d'Anscombe*, à quel point la seule exploitation des statistiques peut s'avérer trompeuse. Nous verrons également comment représenter au mieux des **données multidimensionnelles** c'est-à-dire des données comportant un grand nombre de paramètres, une situation courante dans un contexte Big Data.

6.2.2 Enquêter sur le passé des données

L'exploration des données s'apparente à une enquête. Ont-elles été saisies à la main, si oui par qui, ou ont-elles été enregistrées automatiquement ? Dans quel but ? L'enregistrement a-t-il été effectué en plusieurs étapes ? Si oui, les conditions ont-elles changé d'une étape à l'autre ? Quels sont les référentiels utilisés ? Quelles sont les unités de mesures ? On le voit, les questions à poser sont nombreuses et le meilleur atout reste la curiosité.

Appréhender la structure globale des données n'est cependant pas la seule raison d'être de cette phase d'enquête, celle-ci révélera également les incohérences et les biais qui, détectés de manière précoce, permettront d'éviter les incompréhensions et les remises en questions ultérieures.

Exemple : les fuseaux horaires dans des logs webs

Un problème qui survient couramment dans l'exploration de données de logs de sites web est la prise en compte des fuseaux horaires et des heures d'hiver et d'été des dates enregistrées.

Les logs d'un site comportent typiquement une ligne par page visitée. Les données enregistrées varient en fonction des trackers mais il est courant d'avoir au moins deux dates et heures différentes par ligne :

– la date et heure de la visite de la page enregistrée par le serveur

– la date et heure de la visite de la page sur l'appareil du visiteur

Ces deux enregistrements peuvent être différents dans plusieurs cas : outre les différences de fuseaux horaires (ne pas oublier les fuseaux différents à la demi-heure comme l'Inde à UTC+5:30), le serveur ou l'ordinateur peuvent ne pas être à l'heure. Dans ce cas l'enregistrement effectué par le serveur est toutefois considéré comme le plus fiable.

Il est très important de comprendre dans quels fuseaux horaires sont enregistrées ces dates et heures. Un décalage horaire d'une ou deux heures n'étant pas très visible, l'erreur peut passer inaperçue pendant longtemps.

Dans l'idéal, pour faciliter ce travail d'enquête, les données devraient faire l'objet d'une documentation complète qui spécifie leur provenance, les formats, etc. Dans la pratique ce genre de métadonnées est cependant rarement disponible, surtout lorsqu'on exploite des données anciennes. La seule solution dès lors est d'interroger les différents acteurs qui ont contribué à l'enregistrement des données.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

6.2.3 Utiliser les statistiques descriptives

Si elles ne suffisent pas à révéler la structure globale d'un jeu de données, les statistiques descriptives restent toutefois utiles en complément de la visualisation. Les statistiques descriptives permettent de résumer certains aspects d'un ensemble de données comme son centre et la dispersion des valeurs autour de ce centre. La **moyenne arithmétique** est vraisemblablement la statistique la plus connue. Dans certaines situations on lui préfère cependant la **médiane** qui, rappelons-le, est définie comme la valeur qui sépare un ensemble de valeurs numériques en deux groupes de taille égale, en dessus et au-dessous de cette valeur. Elle possède l'avantage sur la moyenne d'être moins sujette aux valeurs extrêmes. Les **quantiles** généralisent la notion de médiane, ainsi le **quartile** est défini comme le plus petit nombre en dessous duquel on trouve un quart des valeurs de l'échantillon. Des mesures de dispersion existent aussi comme la **variance**, qui est la moyenne des carrées des écarts à la moyenne. L'**écart-type** est la racine de la variance. Elle représente l'ordre de grandeur des fluctuations des observations autour de la moyenne.

Les **tableaux croisés dynamiques**, que proposent les tableurs comme Excel, constituent un outil commode pour synthétiser des ensembles de données rangées dans une table. Les différentes fonctions d'agrégat disponibles correspondent précisément aux statistiques élémentaires que nous venons d'évoquer : moyenne, variance, somme, nombre d'éléments, etc. Définis par des formules, les tableaux sont dynamiques au sens où il est en tout temps possible d'ajuster la structure.

On peut également représenter graphiquement les principales statistiques d'un échantillon. Voici les deux graphes les plus couramment utilisés.

Les box plot

Le **box plot** (ou « la boîte à moustache ») permet de résumer une série de valeurs numériques en représentant à l'échelle le minimum, le maximum, la médiane et les quartiles. On représente parfois aussi les valeurs aberrantes sous forme de points isolés, en marge de la boîte à moustache.

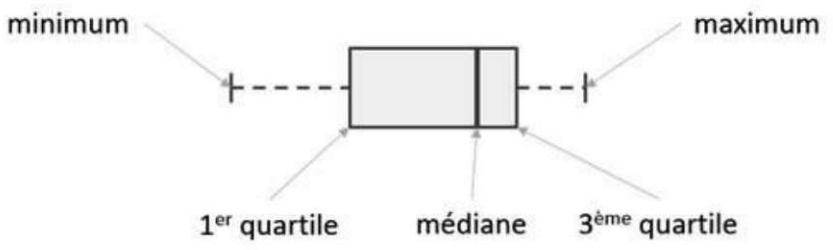


Figure 6.1 — Exemple de box plot – Sur cette représentation, le box plot indique que 50 % des valeurs se trouvent dans la « boîte ».

Copyright © 2015 Dunod.

Les histogrammes

Pour représenter la répartition d'un ensemble de valeurs numériques discrètes, on utilise un **histogramme** qui représente le nombre d'observations comprises dans une succession d'intervalles de valeurs. Pour une distribution de valeurs continues, l'analogue d'un histogramme est une **courbe de densité**. Le nombre d'observations dans un intervalle est alors proportionnel à la surface sous la courbe dans l'intervalle en question.

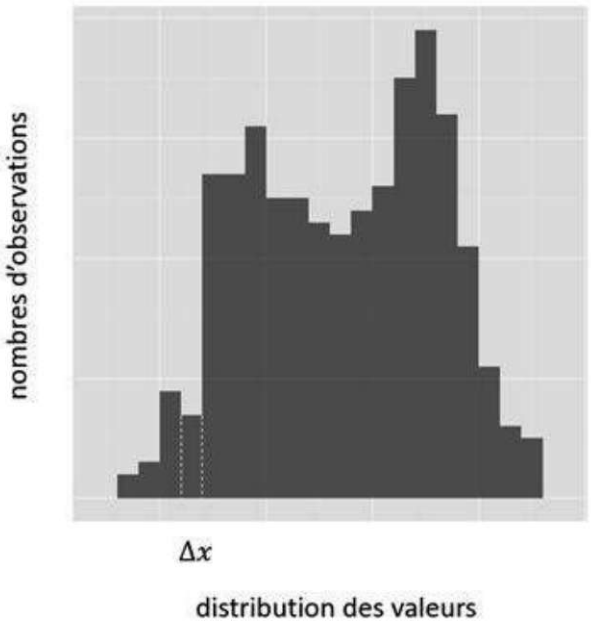


Figure 6.2 — Exemple d'un histogramme. Les valeurs sont découpées en classes de largeur Δx dont l'échelle est en abscisse. En ordonnée, le nombre d'observations dans chaque classe. Sur cet exemple, on peut voir deux modalités ou « pics » qu'il serait difficile d'identifier avec les seuls indicateurs classiques comme la moyenne, la médiane ou même avec un box plot.

La meilleure approche consiste généralement à multiplier les représentations pour avoir une vue des données selon différents angles.

6.2.4 Les tableaux croisés dynamiques

Une autre possibilité utile pour explorer de grands volumes de données est l'utilisation des « tableaux croisés dynamiques » aussi appelés « pivots tables » en anglais.

Proposés par des tableurs comme Microsoft Excel, ils permettent de générer différentes synthèses d'un tableau de données. Les opérations les plus couramment utilisées sont : le regroupement de données par lignes ou par colonnes et les opérations d'agrégation comme le calcul de somme, de moyenne, d'extremum ou le dénombrement. Il est ainsi aisé d'explorer la répartition des valeurs de différentes variables sous différentes conditions. Multiplier les vues sur un même jeu de données est le principal atout des tableaux croisés.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Row Labels	Values Average of temperature	Max of temperature	Min of temperature
Paris	17,6	34	6
Clear	20,5	29	13
Cloudy	16,0	24	8
Drizzle	15,2	19	12
Fair	18,2	34	8
Fog	14,8	21	10
Heavy Drizzle	16,3	19	14
Heavy Rain	14,0	14	14
Light Drizzle	15,1	17	11
Light Rain	16,0	28	11
Light Rain with Thunder	18,0	20	16
Mist	16,1	19	14
Mostly Cloudy	17,7	31	8
Overcast	15,8	17	14
Partly Cloudy	18,7	29	8
Rain	16,8	26	13
Scattered Clouds	21,1	31	15
Shallow Fog	16,5	20	13
Thunder	19,8	25	15
Thunder in the Vicinity	21,3	28	17
Thunderstorm	19,8	21	19
Unknown	23,9	28	15
(blank)	14,8	24	6
Toulouse	20,2	34	6
Clear	21,2	34	11
Cloudy	18,8	27	11
Drizzle	16,7	17	16
Fair	20,8	34	8
Fair/Windy	23,5	26	22

Figure 6.3 — Exemple d'un tableau croisé.
À partir de relevés météorologiques sur plusieurs villes, il est possible de regrouper les lignes en deux catégories : par ville et, à un second niveau, par condition météorologique. Pour chacune de ces lignes, on agrège les valeurs de température pour calculer les températures moyennes, maximales et minimales.

En conclusion, l'exploration de données doit permettre d'appréhender rapidement la structure globale d'un jeu de données. Débusquer les éventuelles valeurs aberrantes est particulièrement important. Négliger cette phase peut être lourd de conséquence pour la suite de l'analyse, par exemple lorsque l'on découvre tardivement qu'une modélisation statistique a été biaisée consécutivement à la non-exclusion de certaines valeurs aberrantes.

Copyright © 2015 Dunod.

6.3 LA PRÉPARATION DE DONNÉES

6.3.1 Pourquoi préparer ?

L'objectif de la préparation des données est de constituer un jeu de données de qualité homogène, dont la structure et les formats sont cohérents et bien définis. Ainsi préparées, les données pourront être exploitées pour identifier des corrélations utiles entre certaines variables. L'apprentissage automatique consistera à exploiter ces corrélations pour créer des modèles prédictifs utiles au business, ce sera le sujet du chapitre 7.

Traiter les valeurs manquantes, supprimer les valeurs aberrantes, redéfinir les échelles des paramètres pour homogénéiser leur variabilité, enrichir les données avec des référentiels externes sont les principales actions menées durant cette phase.

Difficile à automatiser entièrement, exigeant patience et minutie, cette phase est souvent envisagée comme la partie ingrate du travail d'un data scientist. Elle n'en demeure pas moins indispensable et peut représenter jusqu'à 80 % de son travail.

6.3.2 Nettoyer les données

Le nettoyage des données consiste à éliminer toutes les informations que l'on ne souhaite pas conserver telles quelles. Ces informations peuvent être erronées, inexactes, induire des erreurs ou simplement être sans intérêt pour la suite de l'analyse ou la modélisation.

Imaginons que l'on souhaite constituer une liste de clients pour l'envoi d'une newsletter personnalisée. D'une table de clients on pourrait par exemple décider d'éliminer dans un premier temps tous les enregistrements dont les noms et les prénoms ne sont pas renseignés, puis tous ceux dont l'adresse e-mail ne se conforme pas à la syntaxe `nom@societe.fr`.

Le nettoyage des données va consister à identifier toutes les valeurs que l'on souhaite retirer ou corriger :

- Pour chaque champ on détecte les types ou la syntaxe attendus (numéro postal, téléphone, e-mail, URL, IP...) et on identifie les valeurs qui ne s'y conforment pas.
- De même, on recherche les valeurs numériques aberrantes. Celles-ci peuvent être définies en excluant par exemple 1 % des valeurs les plus extrêmes d'un échantillon. Si l'on a des raisons de croire que les données obéissent à une certaine loi de probabilité (loi de Poisson, loi exponentielle, distribution normale, etc.), on peut utiliser cette loi supposée pour éliminer des valeurs peu vraisemblables au vu de leur trop faible probabilité.

Une fois ces valeurs détectées, différentes actions sont envisageables :

- Supprimer la valeur en question et la laisser sans valeur (=null).
- Supprimer la ligne entière incriminée.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

- Créer une autre variable binaire supplémentaire (un « flag ») qui indique si la valeur est conforme à ce que l'on attendait.
- Remplacer la valeur par la moyenne, la médiane ou la valeur la plus courante.
- Remplacer la valeur par la valeur précédente ou pour autant que les données soient ordonnées.

6.3.3 Transformer les données

La phase de préparation des données peut également consister à manipuler, modifier voire créer de nouvelles informations à partir des informations disponibles.

En possession d'une liste d'articles de presse, on pourrait par exemple extraire de chacun d'eux les dix mots les plus fréquents à l'exclusion de certains pronoms grammaticaux dépourvus de tout contenu sémantique.

À partir d'une date au format `JJ/MM/AAAA` on pourrait extraire le jour de la semaine et déterminer si cette semaine correspond ou non à une période de vacances. Ces nouvelles variables auxiliaires pourront s'avérer utiles par la suite pour comprendre certains comportements humains et ainsi permettre d'élaborer un meilleur modèle prédictif.

D'autres manipulations porteront plutôt sur la syntaxe des données considérées comme des chaînes de caractères. Voici quelques exemples classés par catégorie.

Séparation et extraction

- La troncature de variable consiste à ne retenir que certaines parties d'une chaîne de caractères.
- La séparation en éléments consiste procéder à une analyse syntaxique pour transformer une chaîne de caractères une liste de chaînes plus courtes dont chacun représente la valeur d'une variable.
- Extraire un élément particulier d'une chaîne comme un nombre, une adresse e-mail

Agrégation

Une agrégation consiste à regrouper ou assembler plusieurs valeurs en une seule.

- Juxtaposition, avec ou sans séparateur, chaînage avec compteur pour compter le nombre d'apparitions de certains éléments.
- Calculs de moyenne, de somme, de médiane, de maximum, de minimum, de quantile à partir d'une série numérique.
- Identification de l'élément le plus fréquent ou le moins fréquent d'une liste.

Transformation

- Toutes les transformations sur les dates : changement de fuseau horaire, extraction d'une heure, du jour de semaine du mois ou de la différence entre deux dates.

Copyright © 2015 Dunod.

- Toutes les opérations qui consistent à réorganiser les données ou à les agréger par catégories pertinentes.
- Copie de valeurs sur plusieurs lignes ou colonnes, ainsi pour les valeurs manquantes, on pourra recopier par exemple la dernière valeur disponible.

Un exemple de transformation avec les cohortes

En statistique, on appelle **cohortes** des groupes d'individus qui partagent un même indicateur temporel. On regroupe souvent les individus par la date d'une action (comme l'inscription sur un site, la première commande), par leur date de naissance ou autre. Ainsi, l'analyse par cohortes croisée avec un autre indicateur permet de comprendre l'évolution temporelle d'un phénomène en relation avec un événement compréhensible et non pas en fonction du calendrier traditionnel.

Prenons l'exemple des clients d'un site de e-commerce. À partir d'une liste d'achats associée à des clients dont on connaît la date d'arrivée sur le site, il est alors possible d'effectuer les opérations suivantes :

1. Regrouper les clients par leur date d'inscription sur le site.
2. Regrouper les achats en fonction du moment de vie de chaque client.
3. Agréger les dépenses et calculer la dépense moyenne par client (ou le « panier moyen ») pour chacun des groupes ainsi constitués.

La cohorte ainsi créée permet d'observer l'évolution du comportement des clients au cours du temps. Ainsi pourra-t-on répondre à des questions du genre : les nouveaux clients ont-ils un panier moyen similaire aux anciens clients au même moment de « leur vie » sur le site ?

	Panier moyen 1er mois	Panier moyen 2me mois	Panier moyen 3me mois	Panier moyen 4me mois	Panier moyen 5me mois
Inscrits en Janvier	11 €	10 €	5 €	3 €	1 €
Inscrits en Février	12 €	10 €	6 €	2 €	
Inscrits en Mars	10 €	9 €	7 €		
Inscrits en Avril	13 €	10 €			
Inscrits en Mai	14 €				

Figure 6.4 — Exemple d'une cohorte présentant le panier moyen des clients qui sont regroupés selon deux critères (la date d'inscription et leur moment de vie sur le site).

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

6.3.4 Enrichir les données

On parle d'enrichissement de données lorsqu'on croise les données existantes avec de nouvelles informations, parfois extérieures à l'entreprise. Ces nouvelles variables pourront alors être utilisées pour identifier de nouvelles corrélations significatives et, à terme, pour élaborer des modèles de prédictions plus précis.

Ainsi, une position géographique (un couple de coordonnées latitude/longitude) pourrait-elle être enrichie avec le nom de la ville la plus proche beaucoup plus facile à interpréter par un cerveau humain. Une base de données géographique est nécessaire afin de réaliser cette transformation.

Dans le cas d'un SGBDR la notion de croisement n'est autre que celle, bien familière, de jointure. Ainsi, à partir d'un historique de transactions ou de ventes on pourrait souhaiter récupérer des informations contenues dans un CRM.

Voici quelques exemples d'enrichissements :

- **Géographique :**
  - Le **géocodage** consiste à affecter des coordonnées géographiques (un couple latitude/longitude) à une adresse (rue, ville, code postal). La qualité des données ainsi enrichies dépendra à la fois de la précision (veut-on localiser à la ville ou au numéro d'une rue ?) et du taux de réussite (a-t-on bien identifié l'adresse ?).
  - Le **géocodage inversé,** c'est l'opération inverse, c'est-à-dire l'attribution d'une adresse à un couple de coordonnées géographiques.
  - La **résolution d'adresse IP** consiste à convertir l'adresse IP d'un internaute en coordonnées géographiques, même si cela reste très approximatif.
- **Temporels :**
  - Déterminer si une date se trouve dans une période de vacances scolaire pour un pays donné. Il faut ici avoir recours à une base de données externe.
  - Déterminer si une date correspond à un jour férié dans un pays donné.
- **Socio-économique :**
  - Associer des indicateurs sur la population (nombre, niveau de richesse, etc.) à une zone géographique.
  - Associer des informations juridiques et financières au nom d'une entreprise.

Toutes sortes d'enrichissement sont envisageables dès lors que l'on a accès à des informations complémentaires et que l'on peut les relier entre elles. Concrètement, cela se fait par jointure entre les données existantes et les nouvelles données. Voici quelques exemples de champ sur lesquels effectuer la jointure :

- **Jointure sur chaînes de caractères**
  - Dans le cas d'une **jointure exacte**, deux chaînes de caractères sont comparées et les enregistrements correspondants sont joints lorsque la coïncidence est exacte. Ainsi dans un CRM par exemple de nombreuses jointures se font sur les adresses e-mails.

Copyright © 2015 Dunod.

  - Dans le cas d'une **jointure approximative** ou « *fuzzy join* » on accepte des erreurs ou des approximations pour effectuer une jointure. C'est utile pour corriger les fautes de frappes.

- **Géographiques** – Il s'agit généralement de rapprocher deux éléments géographiques proches. À un couple longitude-latitude on associera par exemple l'hôtel le plus proche.
- **Temporelles** – De même, on pourra associer deux éléments proches dans le temps dans le sens où ils ont eu lieu le même jour ou la même heure.
- **Multiples** – Il est également possible d'effectuer une jointure sur plusieurs champs simultanément. Ainsi une donnée météorologique pourra être associée à une date et un lieu géographique.

6.3.5 Un exemple de préparation de données

Ecometering, une filiale du Groupe GDF Suez qui travaille dans le secteur de l'énergie, a proposé pendant l'été 2014 un challenge public sur le site `datascience.net`. Dans ce cadre, les candidats pouvaient télécharger des données réelles afin de les analyser puis de proposer leurs prédictions de consommation d'électricité pour différents sites industriels et tertiaires. Les candidats ayant obtenu les meilleurs modèles ont ensuite été récompensés.

La préparation des données, aussi bien de la réorganisation que de l'enrichissement, a constitué une partie substantielle du travail des équipes en lice. Le fichier de données principal, fourni dans le cadre du challenge comportait des relevés de consommation électriques pour différents sites industriels et tertiaires, voir la figure 6.5. En moyenne un enregistrement était effectué toutes les dix minutes. L'une des difficultés à surmonter était que les périodes de relevés variaient d'un site à l'autre.

DATE_LOCAL	ID01	ID02	ID03	ID04	ID05	ID06	ID07	ID08	ID09
01/01/2011 00:00	4314	3020				176	35	205	
01/01/2011 00:10	4295	2941				163	26	212	
01/01/2011 00:20	4346	2877				137	46	195	
01/01/2011 00:30	4379	2836				131	32	188	
01/01/2011 00:40	4350	2877				133	46	190	

Figure 6.5 — Le fichier de données principal qui contient les relevés de consommation électrique avant transformation. Les colonnes ID01, ID02... contiennent les consommations des sites correspondants.

Afin de faciliter l'apprentissage automatique sur les données, il est plus pratique de n'utiliser qu'une seule colonne pour désigner un site, raison pour laquelle on a procédé à cette transformation comme le montre la figure 6.6 (a). Ainsi le nombre de colonnes diminue tandis que le nombre de lignes augmente.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

DATE_LOCAL	Site	consomation
1/1/11 0:00	ID01	4314
1/1/11 0:00	ID02	3020
1/1/11 0:00	ID06	176
1/1/11 0:00	ID07	35
1/1/11 0:00	ID08	205
1/1/11 0:00	ID10	207
1/1/11 0:00	ID16	75

(a)

DATE_LOCAL	Site	consomation	temperature
01/01/2011 00:00	ID01	4314	-2,5
01/01/2011 00:00	ID02	3020	-2,5
01/01/2011 00:00	ID06	176	-0,5
01/01/2011 00:00	ID07	35	0,1
01/01/2011 00:00	ID08	205	-0,7
01/01/2011 00:00	ID10	207	2,3
01/01/2011 00:00	ID16	75	0,1

(b)

Figure 6.6 — (a) Aperçu des mêmes données après transformation. On a désormais une ligne par site et par horaire. (b) Aperçu après « enrichissement » avec les températures.

Dans le cadre du challenge, d'autres fichiers étaient fournis pour enrichir ces données de base avec des informations telles que :

- Les températures sur les sites obtenues par une jointure sur l'heure et la localisation du site.
- Des informations plus générales sur le site comme le type d'entreprise (tertiaire, industrielle, etc.) obtenues par jointure sur l'identifiant du site.

L'intérêt d'enrichir ainsi le modèle initial est d'augmenter la précision de prédiction du modèle, voir chapitre 7. Ainsi peut-on imaginer par exemple que la température relevée sur un site soit corrélée de manière exploitable avec la consommation qui est la grandeur que l'on souhaite prédire.

6.4 LES OUTILS DE PREPARATION DE DONNÉES

La préparation des données est une phase coûteuse en temps, essentiellement car elle n'est pas entièrement automatisable. Les principaux outils disponibles pour cette tâche peuvent être répartis en quatre catégories.

6.4.1 La programmation

Les langages de programmation généralistes comme Python ou Java, des langages de transformation ou d'interrogation de données comme Pig ou Hive ou encore des langages dédiés exclusivement à l'analyse statistique comme R offrent un maximum de flexibilité mais exigent en contrepartie un haut niveau de compétence et leur courbe d'apprentissage est raide. S'adressant aux data scientists férus de programmation qui souhaitent explorer en profondeur les données, ces outils offrent généralement une productivité limitée.

Copyright © 2015 Dunod.

6.4.2 Les ETL

Ce sont des outils qui permettent de créer des *workflow* de transformation de données par combinaison de briques de transformation de bas niveau (split, jointure, opération sur des strings, etc.). Rappelons cependant que ces outils ont un cadre d'utilisation beaucoup plus large que la simple préparation de données. Dans le cadre de la préparation des données ils apportent un élément d'industrialisation non négligeable. Les éditeurs les plus connus en la matière sont *Talend* et *Informatica*.

6.4.3 Les tableurs

Situés à l'extrémité opposée des outils de programmation, les tableurs de type Microsoft Excel sont des outils généralement bien maîtrisés par des utilisateurs métiers qui les utilisent directement sur leur poste de travail. S'ils permettent de réorganiser et de traiter de faibles volumes de données, les tableurs n'offrent en revanche aucune traçabilité et modifient les données sur lesquelles ils opèrent.

6.4.4 Les outils de préparation visuels

Il s'agit d'une catégorie d'outils récents qui ont l'ambition de concilier les atouts de traçabilité et d'industrialisation des ETL avec les aspects visuels et intuitifs des tableurs. *Data Science Studio* de la société *Dataiku* est un exemple que l'on peut ranger dans cette catégorie.

En résumé

Parfois considérée comme la partie ingrate du travail du data scientist, en raison de la part importante de travail manuel très terre à terre qu'elle exige, la préparation des données est une phase incontournable de tout projet d'exploitation des données. Elle comprend d'une part des tâches des bas niveaux, sans valeur métier, comme la conversion des données, le traitement des séparateurs ou l'analyse syntaxique. D'autre part elle englobe également des tâches plus proches du métier comme le traitement des valeurs manquantes, l'enrichissement par des sources de données externes et diverses opérations de nettoyage. Si les outils de productivité s'avèrent indispensables pour mener à bien ces tâches phase, la préparation des données reste cependant un domaine où les esprits agiles et débrouillards seront les plus à l'aise.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Copyright © 2015 Dunod.

7

Le machine learning

Objectif

Ce chapitre a pour objectif de définir le *machine learning* (ML) ou l'apprentissage automatique et comment il peut remplacer avec profit un jeu de règles métiers statiques. Nous examinerons en particulier comment juger de la qualité et de la performance d'un algorithme et quelles sont les précautions à prendre dans un contexte Big Data. Les principaux algorithmes seront présentés de manière intuitive sans formalisme. Ils seront par ailleurs illustrés par un cas concret dans lequel on souhaite prédire les revenus générés par de nouveaux clients à partir de leur comportement observé sur un site web et d'autres informations externes croisées.

7.1 QU'EST-CE QUE LE MACHINE LEARNING ?

7.1.1 Comprendre ou prédire ?

Prédire l'avenir est l'un des rêves séculaires de l'humanité. Dès l'Antiquité des ressources infinies de fantaisie et de créativité ont été investies dans ce projet aux motivations très variées : pronostiquer le comportement d'un adversaire avant un engagement militaire, anticiper l'arrivée de la pluie pour régler les semailles et les récoltes, prédire les éclipses du Soleil... Les arts divinatoires et l'oracle de Delphes ont aujourd'hui cédé leur place aux statistiques et aux algorithmes informatiques mais les objectifs sont restés similaires à ce qu'ils étaient il y a deux millénaires. Le sujet qui nous intéresse dans cet ouvrage, l'analyse prédictive au service des entreprises, est sans doute à rapprocher des préoccupations militaires de l'Antiquité puisqu'il s'agit

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

d'anticiper certains comportements humains et d'obtenir ainsi un avantage compétitif sur un adversaire qu'on espère moins bien équipé.

Voici quelques exemples d'utilisation du *machine learning* parmi les plus communs :

- Détecter des comportements frauduleux lors de transactions financières en ligne.
- Estimer un taux de transformation sur un site marchand en fonction du nombre de clics sur certaines pages.
- Prédire les risques de non-solvabilité d'un client en fonction de ses ressources et de son profil socioprofessionnel.
- Anticiper les intentions de résiliation d'un service en fonction des activités d'un souscripteur.
- Découvrir les préférences d'un client que l'on souhaite retenir pour lui suggérer des produits et des services adaptés à ses goûts et ses besoins.

Ces préoccupations ne sont évidemment pas nouvelles, elles sont même le pain quotidien des départements de marketing, des RSSI ou des actuaires.

Remarque : l'essor récent de l'analyse prédictive tient moins à des percées décisives dans le domaine des algorithmes prédictifs, dont beaucoup sont connus depuis des décennies, qu'aux nouvelles opportunités qu'offre la disponibilité des données en quantités massives, aujourd'hui *via* les réseaux sociaux et demain *via* la myriade de capteurs de l'internet des objets, conjuguée avec la puissance virtuellement illimitée d'architectures massivement parallèles. L'analyse prédictive combinée à ces technologies est un domaine où la technologie précède les usages qui, pour beaucoup, restent encore à inventer. Ce n'est d'ailleurs pas là le moindre des intérêts du métier de data scientist.

Commençons par clarifier la notion de **prédiction** en rappelant ce qu'elle recouvre dans le cadre d'une discipline scientifique classique comme l'astronomie.

Ayant patiemment observé le mouvement des planètes, les astronomes de l'Antiquité en on déduit, avec beaucoup d'ingéniosité, il faut le reconnaître, les lois des récurrences des éclipses. La science ne s'arrête pas là cependant car elle a non seulement l'ambition de **prédire** mais encore de **comprendre** le phénomène observé, au moyen d'un modèle explicatif. Ainsi les lois de la gravitation découvertes par Newton des siècles plus tard ont fourni un modèle explicatif aux prédictions des premiers astronomes. L'équivalent des lois déterministes de la gravitation dans un contexte commercial pourrait être un système de règles métiers destinées au calcul du prix de vente d'un produit en fonction de paramètres comme la marge souhaitée, le temps de confection et le prix des composants.

Dans un contexte social comme celui du monde des affaires, cette démarche des sciences déterministes est cependant impraticable si l'on souhaite faire des prédictions. À cela, deux raisons principales :

- L'inextricable complexité des comportements humains implique de renoncer à toute prédiction déterministe. Faire des prédictions en termes de probabilités

Copyright © 2015 Dunod.

de type « *Quelle est la probabilité que tel individu ne rembourse pas son crédit ?* » est le seul objectif atteignable.

- Dans beaucoup de situations, il faudra renoncer également à trouver un modèle explicatif et se contenter d'établir de simples corrélations entre les observations passées. En admettant que ces corrélations demeurent vraies dans le futur, on pourra alors les utiliser pour faire des prédictions.

C'est dans un tel cadre que s'inscrit le **machine learning**, les termes seront précisés dans les paragraphes qui suivent. Chaque observation passée d'un phénomène, comme par exemple « M. *Dupont a souscrit un crédit immobilier dans telles et telles conditions ce qui a rapporté tel bénéfice à la banque* » est décrite au moyen de deux types de variables :

- Les premières sont appelées les **variables prédictives** (ou attributs ou paramètres), en l'occurrence l'âge du client, son historique bancaire et ses revenus. Ce sont les variables à partir desquelles on espère pouvoir faire des prédictions. Les p variables prédictives associées à une observation seront notées comme un vecteur $\mathbf{x}=(x_1,...,x_p)$ à p composantes. Un ensemble de N observations sera constitué de N tels vecteurs $\mathbf{x}^{(1)},..., \mathbf{x}^{(N)}$.
- Une **variable cible** dont on souhaite prédire la valeur pour des évènements non encore observés. Dans notre exemple, il s'agirait du bénéfice ou de la perte estimée pour la compagnie d'assurance suite à l'octroi du prêt à un individu. On notera y cette variable cible avec les mêmes significations pour les indices que pour $\mathbf{x}$.

Pour fixer les idées, on considère schématiquement que la valeur observée y de la variable cible résulte de la superposition de deux contributions :

1. Une **fonction** $F(\mathbf{x})$ des variables prédictives. C'est donc une contribution entièrement déterminée par les variables prédictives $\mathbf{x}$ de l'observation. C'est le signal que l'on souhaite mettre en évidence.
2. Un **bruit** $\varepsilon(\mathbf{x})$ aléatoire. C'est un fourre-tout qui englobe les effets conjugués d'un grand nombre de paramètres dont il est impossible de tenir compte.

Aussi bien F que ε resteront à jamais inconnus mais l'objectif d'un modèle de *machine learning* est d'obtenir une « *bonne approximation* » du signal F à partir d'un ensemble d'observations. Cette approximation sera notée f, on l'appelle la **fonction de prédiction**.

Notons au passage qu'une tâche fondamentale de la statistique consiste à évaluer l'importance relative d'un signal par rapport au bruit. Ainsi le **niveau de signification** d'un test statistique indique, très schématiquement, la probabilité qu'un effet soit dû au hasard. Un niveau de 5 % indique donc que l'effet observé possède une chance sur 20 d'être le fruit du seul hasard.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

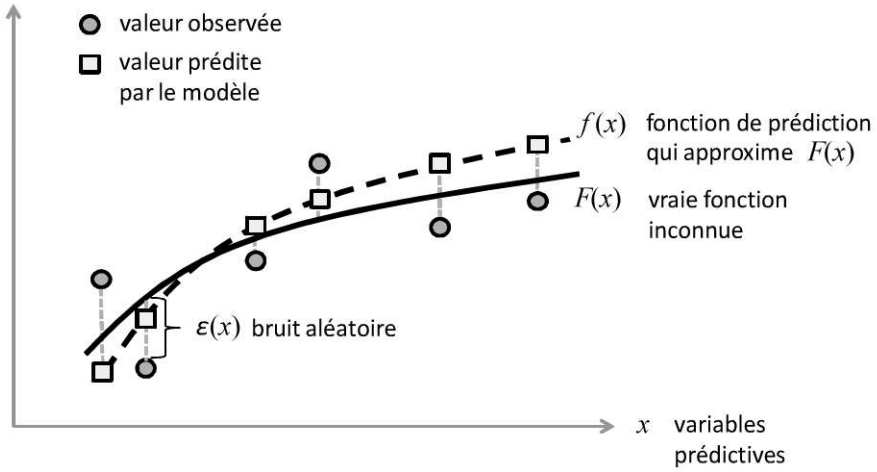


Figure 7.1 — La valeur d'une variable cible est la somme d'une fonction déterministe *F* et d'un bruit aléatoire ε. L'objectif du ML est de trouver une bonne approximation de *F* à partir des observations. Cette approximation *f* est la fonction de prédiction qui permet d'obtenir des estimations *f*(*x*) de *F*(*x*).

Formalisons la discussion précédente avec quelques définitions :

Le **machine learning** (ML) est un ensemble d'outils statistiques ou géométriques et d'algorithmes informatiques qui permettent d'automatiser la construction d'une fonction de prédiction *f* à partir d'un ensemble d'observations que l'on appelle l'**ensemble d'apprentissage**.

Le ML est donc une discipline hybride qui se situe à cheval sur plusieurs sciences et techniques que sont l'analyse statistique, l'intelligence artificielle, la BI et bien sûr l'IT.

Un **modèle de machine learning** est un procédé algorithmique spécifique qui permet de construire une fonction de prédiction *f* à partir d'un jeu de données d'apprentissage. La construction de *f* constitue l'**apprentissage** ou l'**entraînement** du modèle. Une **prédiction** correspond à l'évaluation $f(\mathbf{x})$ de la fonction de prédiction *f* sur les variables prédictives d'une observation **x**.

Comme on le verra au paragraphe 7.3, chaque modèle s'appuie sur des outils statistiques ou géométriques différents ainsi que sur certaines hypothèses concernant le signal *F* que l'on souhaite mettre en évidence[1], par exemple le signal *F* est-il linéaire ou non ? Pour choisir un bon modèle un data scientist devra avoir une double connaissance du métier qu'il doit modéliser et des hypothèses que présupposent chaque algorithme. Son choix sera souvent l'aboutissement d'un processus itératif

1. Techniquement, la distribution de probabilité du bruit ε est aussi une hypothèse de certains modèles. Beaucoup de modèles supposent une distribution du bruit de nature gaussienne.

Copyright © 2015 Dunod.

au cours duquel différents modèles sont successivement entraînés puis évalués, voir section 5.3.4.

Un data scientist peut également être amené à guider le processus d'apprentissage en choisissant certaines données plus significatives que d'autres, phase durant laquelle la visualisation des données multidimensionnelles joue un rôle important. Nous y consacrons le chapitre 8.

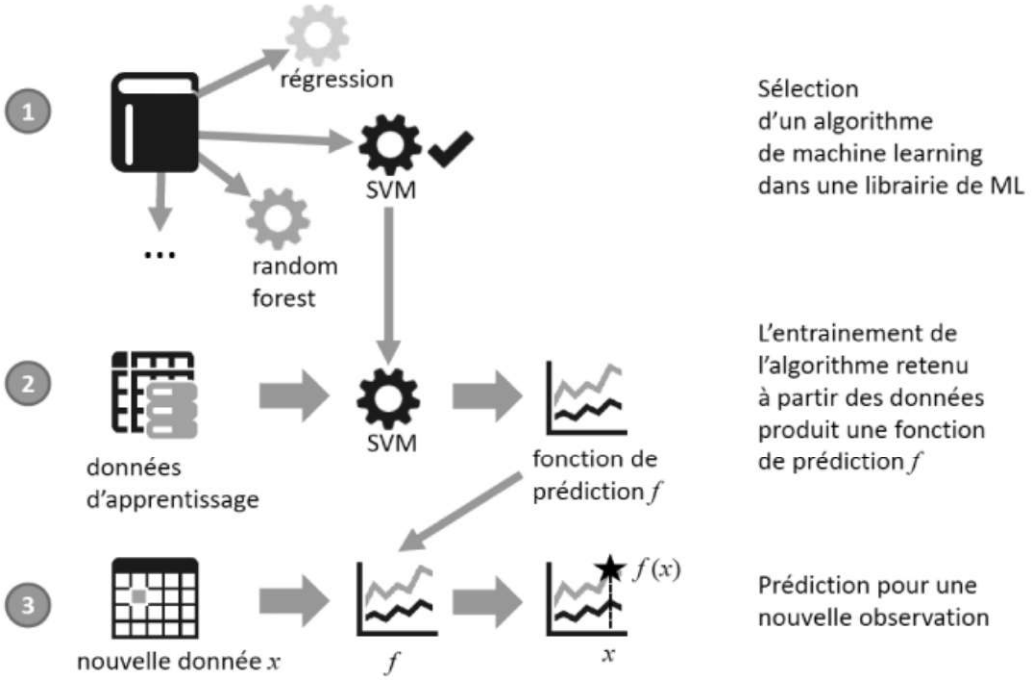


Figure 7.2 — (1) Choix d'un modèle de ML, (2) apprentissage (ou entraînement) du modèle et enfin (3) prédiction pour une nouvelle observation.

Le *machine learning* est donc une **démarche fondamentalement pilotée** par les données. Il sera utile pour construire des systèmes d'aide à la décision qui s'adaptent aux données et pour lesquels aucun algorithme de traitement n'est répertorié.

On trouve parfois les images suivantes pour décrire le ML :

- Le ML permet de créer des systèmes qui apprennent leurs propres règles.
- Le ML permet d'extraire un programme (notre fonction prédictive f) à partir des données.
- Le modèle de ML apprend par lui-même des associations et des similarités à partir d'un jeu données.

Un mot encore, pour conclure, sur la causalité. Distinguer dans un phénomène les causes des effets implique d'examiner la chronologie des évènements. C'est l'une des tâches primordiales de toute démarche scientifique puisqu'elle participe de l'explication recherchée. Il faut bien voir cependant que l'ambition du *machine learning* n'est pas de trouver des **causes**, mais d'identifier des **corrélations utiles** entre des variables prédictives et des variables cibles. Rien n'exige en effet qu'une

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

variable prédictive soit la cause d'un phénomène décrit par une variable cible. Ainsi pourra-t-on observer chez un fumeur une corrélation « utile » entre la couleur jaune de ses dents, envisagée comme une variable prédictive, et le taux de nicotine détecté dans ses poumons, considéré comme variable cible. Faut-il préciser que la couleur des dents n'a aucune influence sur l'état des poumons du pauvre homme ?

7.1.2 Qu'est-ce qu'un bon algorithme de machine learning ?

Le choix d'un algorithme par un data scientist se fait sur plusieurs critères. Lors du congrès Big Data Paris 2014, Ted Dunning, architecte en chef chez *MapR* et figure de proue du *machine learning*, énumérait quelles étaient selon lui les cinq principales qualités que devrait posséder un algorithme de *machine learning* dans un contexte business :

- La « **déployabilté** » : les algorithmes astucieux et très élaborés ne sont d'aucune utilité s'il est impossible de passer à l'échelle sur un *framework* de distribution du calcul comme Hadoop par exemple.
- La **robustesse** : la vraie vie est pleine de données « sales », à la fois incohérentes et incomplètes, avec lesquelles il faudra compter. Les algorithmes délicats n'y ont donc pas leur place, il s'agit de privilégier une forme de rusticité.
- La **transparence** : les applications qui intègrent le *machine learning* devraient en principe voir leur performance s'améliorer au fur et à mesure que progresse le processus d'apprentissage. Pourtant il arrive aussi que ces performances se dégradent, aussi est-il crucial que ces situations soient détectées au plus vite par l'algorithme lui-même.
- L'**adéquation aux compétences** disponibles : pour être utilisable en pratique, un algorithme ne devrait pas exiger pour son implémentation ou son optimisation d'expertise trop poussée.
- La **proportionnalité** : la quantité d'énergie ou de temps investi dans l'amélioration ou l'optimisation d'un algorithme doivent être au moins proportionnelle au gain apporté. Inutile d'investir un an de R&D de cent chercheurs pour une amélioration de 1 %.

Émanant d'un expert éminent du domaine, cette liste est particulièrement instructive et tout aspirant data scientist se doit de la méditer. Cette liste omet cependant, tant elle est évidente, la première qualité d'un algorithme : sa **performance**. Encore faut-il nous accorder sur ce que l'on entend par ce terme dans le contexte du ML. C'est le sujet du paragraphe suivant.

7.1.3 Performance d'un modèle et surapprentissage

La première idée qui vient à l'esprit consiste à définir la performance d'un algorithme de ML comme la proportion de prédictions correctes (ou acceptables dans un sens à définir) faites sur le jeu de données utilisé pour l'entraîner.

Copyright © 2015 Dunod.

Pourtant, à y regarder de plus près, cette définition n'est guère satisfaisante. En effet, l'objectif du ML n'est pas de reproduire avec une précision optimale les valeurs des variables cibles connues mais bien de prédire les valeurs de celles qui n'ont pas encore été observées ! En d'autres termes, il nous faut juger de la qualité d'un algorithme de ML à l'aune de sa capacité à généraliser les associations apprises durant la phase d'entraînement à des nouvelles observations. Il s'agit d'éviter les situations dans lesquelles un modèle de ML trop flexible finisse, à force d'entraînement, par reproduire à la perfection les données d'apprentissage sans pour autant être à même de faire des prédictions précises. C'est le problème du **surapprentissage** du ML.

Le surapprentissage caractérise une situation dans laquelle un modèle de ML reproduit avec une grande précision les données d'apprentissage tout en étant incapable de faire des extrapolations précises.

Une telle situation survient lorsque la complexité d'un modèle est trop grande par rapport à la complexité du signal F que l'on souhaite appréhender. Le modèle apprend alors le bruit ε spécifique aux données d'apprentissage, voir figure 7.1.

La parade à ce problème fondamental du ML consiste à répartir les données dont on dispose en deux groupes séparés. Le premier, que l'on appelle les **données d'entraînement**, sera utilisé durant la phase d'apprentissage. Le second, que l'on appelle les **données de test**, est utilisé pour évaluer la qualité de prédiction du modèle. Une répartition usuelle consiste à en conserver 80 % pour l'entraînement et à en réserver 20 % pour le test.

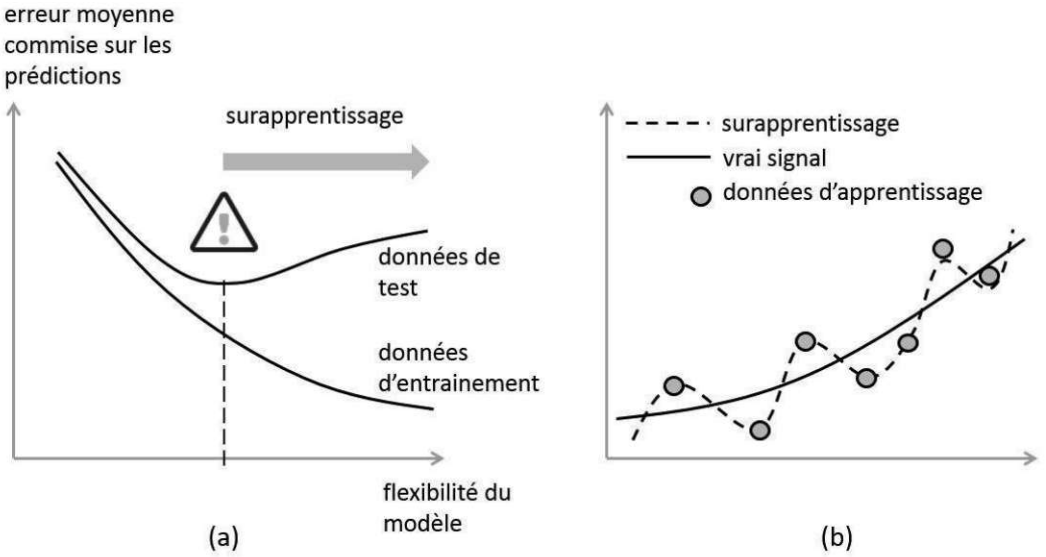


Figure 7.3 — Le surapprentissage : (a) montre comment évoluent les erreurs commises sur les données de tests et celles sur les données d'apprentissage à mesure que la flexibilité du modèle croît. Dans un premier temps, les deux erreurs diminuent de concert. Puis, lorsque s'amorce le surapprentissage, l'erreur sur les données d'apprentissage continue à diminuer alors que celle sur les données de test augmente. (b) illustre un exemple ou une courbe passe par tous les points d'apprentissage mais ne réussit pas à capter la tendance générale.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Pour être complet, il faudrait aussi préciser quelle métrique sera utilisée pour mesurer les erreurs commises. Le sujet est trop technique pour être abordé dans le cadre de cet ouvrage. Disons simplement que les métriques diffèrent selon que la variable cible est continue ou discrète, selon la nature des fluctuations ε et selon la robustesse que l'on souhaite par rapport aux valeurs aberrantes. Nous y reviendrons brièvement à la section 7.3.1.

Dans la pratique on améliore le principe de séparation des données en données d'apprentissage et en données de test au moyen de la **validation croisée**. L'idée consiste à découper de manière aléatoire les données initiales, non pas en deux parties, mais en k parties disjointes de taille égales. Pour chacune des k parties, on entraîne alors un modèle sur l'ensemble des données à l'exception précisément de celles appartenant à cette partie qui sont utilisées pour évaluer la précision des prédictions du modèle. À l'issue des k entraînements, les prédictions pour chaque partie sont comparées aux vraies valeurs et une erreur moyenne sur l'ensemble des observations est estimée.

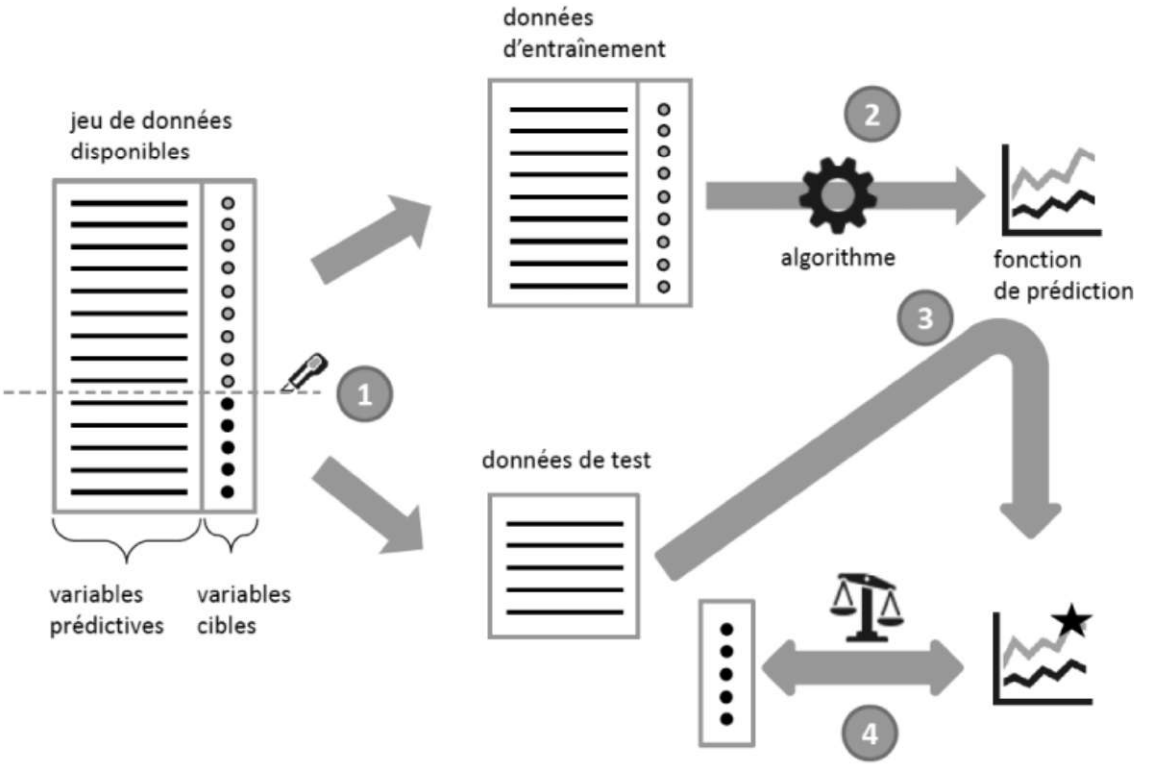


Figure 7.4 — Le principe de la validation croisée : (1) séparation du jeu de données initial en deux jeux de données : d'entraînement et de tests, (2) entraînement de l'algorithme avec les données d'entraînement qui produit une fonction de prédiction, (3) évaluation de la fonction de prédiction sur les données de tests pour obtenir des prédictions (4) comparaison des prédictions aux valeurs cibles des données de test.

La question de savoir quelle quantité de données il convient de récolter pour obtenir une certaine précision est délicate. Dans des situations très simples, la théorie des probabilités montre que pour diviser par N la variabilité des prédictions il faut multiplier la taille des échantillons par N^2. Mais en règle générale on procède empiriquement. Quoi qu'il en soit le volume des données d'apprentissage croît avec

Copyright © 2015 Dunod.

(1) la précision souhaitée, (2) le nombre de variables prédictives utilisées et (3) la complexité du modèle.

7.1.4 Machine learning et Big Data – sur quoi faut-il être vigilant ?

Forgée comme beaucoup d'autres par des départements de marketing plus soucieux d'impact émotionnel que de rigueur sémantique, l'expression « Big Data » laisse ouverte la question cruciale de savoir ce qui est gros au juste dans les « Big Data ».

Première interprétation, suggérée par le « V » = volume du sigle VVV : ce qui est gros, c'est le volume des données. Dès lors, ce qui était une source de difficulté pour l'IT, avec la nécessité comme on l'a vu aux chapitres 3 et 4 d'introduire des systèmes de stockage et des traitements parallèles adaptés, est a priori une bonne nouvelle pour le statisticien. Disposer d'une masse suffisante de données est en effet une condition nécessaire pour limiter les fluctuations statistiques des prédictions. Bien évidemment elle n'est pas suffisante puisqu'un important volume de données ne saurait en aucun cas remplacer des données de qualité.

Seconde interprétation, suggérée par le « V » = variété : ce qui est grand, c'est le nombre de paramètres p qui caractérisent chaque observation. Dans certains contextes Big Data ce nombre de paramètres pourrait se chiffrer en milliers. Cette fois, le statisticien a lui aussi des soucis à se faire et ceci pour plusieurs raisons.

- **Le « fléau de la dimension »** : supposons que chaque observation d'un échantillon utilisé en phase d'apprentissage comporte p paramètres. Pour qu'un échantillon soit statistiquement représentatif de la population, il faut disposer d'un nombre suffisant de valeurs pour chaque paramètre. Admettons, pour fixer les idées, qu'il en faille 10. Il faudra dès lors disposer d'environ 10^p observations pour que l'échantillon soit représentatif. Cette croissance exponentielle est bien évidemment irréalisable en pratique, rappelons en effet que p peu facilement excéder 1 000, si bien que les ensembles de données seront systématiquement clairsemés même si leur volume se chiffre dans l'absolu en téra ou en pétaoctets.
- Pour faire face à ce problème on utilise des techniques dites de **réduction dimensionnelle** dont l'objectif est de compresser l'information en éliminant autant que possible les paramètres inutiles ou redondants ou en identifiant les combinaisons de paramètres les plus prédictives. Nous évoquerons brièvement ce sujet à la section 7.3.9.
- **Les corrélations fictives** : en termes imagés, dans les ensembles de données très riches du Big Data, on finit toujours par trouver ce que l'on cherche et même ce qui n'existe pas ! Dit autrement, une utilisation trop naïve des statistiques qui ne prendrait pas certaines précautions élémentaires[1], interprétera à tort certaines corrélations comme des phénomènes réels alors qu'elles ne sont que

1. Il s'agit en l'occurrence des règles propres aux tests d'hypothèses multiples qui commandent d'ajuster le niveau de signification d'un test au nombre d'hypothèses testées simultanément. La règle la plus utilisée à cet effet s'appelle la correction de Bonferroni.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

le fruit du hasard et d'un niveau de signification fixé à un seuil trop faible. La figure 7.5 fournit un exemple classique de « fausses trouvailles ».

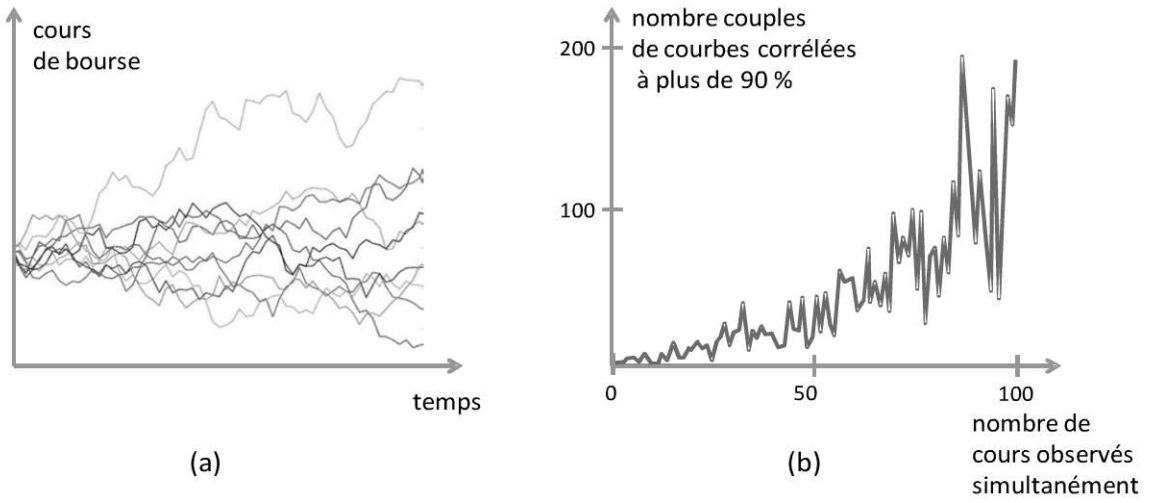


Figure 7.5 — La partie (a) représente des simulations de cours boursiers indépendants et parfaitement aléatoires. La partie (b) représente le nombre de couples de courbes corrélées à plus de 90 % en fonction du nombre total de cours observés simultanément.

- **La parallélisation des algorithmes** : nombre d'algorithmes et de logiciels ont été conçus à une époque ou les données comportaient tout au plus quelques milliers d'observations. Il faudra donc veiller à ce que les algorithmes soient parallélisables pour qu'ils soient utiles dans un contexte Big Data où le nombre d'observations peut se chiffrer en milliards et le nombre de paramètres en millions. Des outils récents comme le *framework* Apache Mahout ou Jubatus sont consacrés à l'exécution d'algorithmes de ML sur des architectures massivement parallèles.

Concluons ce paragraphe pas deux remarques :

- Dans de nombreuses utilisations les traitements qui impliquent un parallélisme massif de type Hadoop/MapReduce ne concernent que la phase de préparation des données : nettoyage, tri, conversion, normalisation, détection des inconsistances etc. L'application des algorithmes de ML proprement dit se fait alors, a posteriori, sur des jeux de données beaucoup plus modestes que l'on peut traiter en mémoire sur une machine au moyen d'implémentations classiques.
- L'un des crédos qui sous-tend à l'heure actuelle une bonne part de l'industrie Big Data affirme que les meilleures performances sont obtenues au moyen d'algorithmes relativement peu sophistiqués qui opèrent sur de grandes quantités de données, plutôt que l'inverse ce qu'indiquait déjà le dicton Better data outweighs clever maths mentionné section 5.3.4. Le succès des algorithmes de suggestion ou de traduction automatique de Google qui mettent en œuvre ce principe atteste de la validité de ce parti pris.

Copyright © 2015 Dunod.

7.2 LES DIFFÉRENTS TYPES DE MACHINE LEARNING

« *Tous les modèles sont faux, mais certains sont utiles !* »
George Box, 1919-2013

Par cette boutade, George Box un statisticien britannique renommé, souhaite nous rendre attentifs sur le fait qu'il n'existe pas de choix objectif pour un modèle statistique. En dernière analyse, c'est le modèle le plus prédictif qui sera considéré le meilleur et non celui dont la justification théorique est la plus élaborée. Le principe de parcimonie d'Occam, qui considère que l'explication la plus simple d'un phénomène est en général la plus plausible, pourra en revanche être un bon guide.

Une autre expression du folklore des statisticiens éclaire bien cet état de fait. Il s'agit du « *No Free Lunch Theorem* »[1]. Intuitivement, ce résultat signifie que lorsqu'on considère de grands ensembles de problèmes, le coût en ressources de calcul pour obtenir une solution (une prédiction utile) moyennée sur tous les problèmes de cet ensemble, ne dépend *pas* de l'algorithme utilisé. En d'autres termes : si l'on cherche un algorithme capable de résoudre simultanément une très grande classe de problèmes, on peut choisir n'importe lequel ! La seule issue consiste par conséquent restreindre la classe de problèmes que l'on souhaite résoudre et à hoisir un algorithme adapté à cette classe particulière.

Avant d'en venir à une description des algorithmes les plus utilisés, nous en proposons une classification selon plusieurs critères qui peuvent être combinés. Le critère le plus important est celui décrit au paragraphe suivant, il distingue l'apprentissage supervisé de l'apprentissage non supervisé.

7.2.1 Apprentissage supervisé ou non supervisé ?

L'**apprentissage supervisé** est la forme la plus courante de *machine learning*. C'est celle à laquelle nous avons implicitement fait référence jusque-là. Elle présuppose que l'on dispose d'un ensemble d'exemples, caractérisés par des variables prédictives $\mathbf{x}^{(1)},..., \mathbf{x}^{(N)}$ pour lesquels on connaît les valeurs de la variable cible $y^{(1)},..., y^{(N)}$. On parle parfois à ce titre d'exemples étiquetés. L'objectif durant la phase d'apprentissage est de généraliser l'association observée entre variable explicatives et variable cible pour construire une fonction de prédiction *f*. La régression linéaire et les machines à vecteurs de support sont des exemples.

L'**apprentissage non supervisé** ou **clustering** ne présuppose aucun étiquetage préalable des données d'apprentissage. L'objectif est que le système parvienne, par lui-même, à regrouper en catégories les exemples fournis en exemple. Cela présuppose qu'il existe une notion de distance ou de similarité entre observations qui peut être utilisée à cet effet. L'interprétation des catégories identifiées reste à la charge d'un expert humain. L'algorithme des *k*-moyennes appartient à cette catégorie.

1. Wolpert, David (1996), « *The Lack of A Priori Distinctions between Learning Algorithms* », *Neural Computation*, pp. 1 341-1 390.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

7.2.2 Régression ou classification ?

On distingue usuellement les catégories de variables (prédictives ou cibles) suivantes :

- Les **variables quantitatives** sont des variables numériques qui correspondent à des quantités. Un prix est une variable quantitative de même qu'un nombre de commandes.
- Les **variables qualitatives** définissent des catégories ou des classes. On distingue deux sous-catégories :
  - Les *variables nominales* – Ce sont des variables qualitatives définies par des noms, comme des noms pays par exemple.
  - Les *variables ordinales* – Ce sont des variables qualitatives que l'on peut comparer entre elles comme des niveaux de risques par exemple.

Ce qui nous amène aux deux définitions suivantes :

- Un **modèle de régression** est un modèle de ML dont la variable cible est quantitative. La régression linaire est un exemple.
- Un **modèle de classification** est un modèle de ML dont la variable cible est qualitative. La régression logistique est un exemple où la variable cible est binaire.

7.2.3 Algorithmes linéaires ou non linéaires ?

Un algorithme **linéaire** est un algorithme dont la fonction de prédiction f est une fonction de la combinaison linéaire $a_1x_1 + \ldots + a_nx_n \equiv \mathbf{a} \cdot \mathbf{x}$ des variables prédictives. Remarquons que $f(x) = g(\mathbf{a} \cdot \mathbf{x})$ pour une certaine fonction g n'a cependant aucune raison d'être linéaire. La combinaison $\mathbf{a} \cdot \mathbf{x}$ est souvent appelée **score** et est alors interprétable, comme un niveau de risque par exemple.

La régression logistique est un exemple d'un tel algorithme linéaire. Les systèmes de neurones et les machines à vecteurs de support sont en revanche des algorithmes non linéaires.

7.2.4 Modèle paramétrique ou non paramétrique ?

Dans certains modèles de ML on postule pour la fonction de prédiction f une forme particulière. Un exemple type pourrait être celui d'une régression non linéaire où l'on postule une relation de type $y = a\,x^2 + b\,x + c$ entre une variable prédictive x et une variable cible y, les paramètres a, b, et c étant estimés à partir des données d'apprentissage. Un tel modèle qui présuppose pour f une forme particulière, avec un nombre spécifié par avance de paramètres ajustables, est un **modèle paramétrique**.

Lorsqu'aucune forme particulière n'est postulée par avance pour la fonction de prédiction, on parle de modèle non paramétrique. Les arbres de décisions ou les k plus proches voisins sont des exemples de **modèles non paramétriques**.

Copyright © 2015 Dunod.

7.2.5 Apprentissage hors ligne ou incrémental ?

Dans les situations où l'on connait l'intégralité des données d'apprentissage avant même de procéder à l'apprentissage, on parle d'une méthode d'apprentissage **hors ligne** ou **statique**. Dans les situations où il existe un flot continu d'informations auxquels l'algorithme doit s'adapter, cette hypothèse est naturellement fausse. Les algorithmes qui ajustent leur fonction prédictive f au fur et à mesure que les données leur parviennent, sont dit **incrémentaux** ou **online**.

Un exemple typique d'un système d'apprentissage en ligne est un mécanisme de filtrage antispam qui doit ajuster ses critères de tri à chaque fois qu'un utilisateur lui signale un nouveau cas positif ou négatif.

7.2.6 Modèle géométrique ou probabiliste ?

Construire une fonction de prédiction à partir d'un ensemble d'observations se fait à partir d'un modèle, c'est-à-dire d'un ensemble d'hypothèses simplificatrices quant au lien qui lie la variable cible y aux variables prédictives $\mathbf{x}$.

Les hypothèses les plus intuitives ont souvent un caractère géométrique. À titre d'exemple considérons l'algorithme de classification des k plus proches voisins, voir section 7.3.2. Pour prédire la classe d'une nouvelle observation, cet algorithme propose de regarder parmi les k plus proches voisins (k est un nombre fixé par avance, d'ordinaire inférieur à 10) la classe la plus représentée et de l'attribuer comme prédiction à notre observation. En bref, chaque observation ressemble à ses voisins. Parler de proximité présuppose en l'occurrence une notion de métrique qui est bien une notion géométrique. Il s'agit donc d'un **modèle géométrique**.

Bien que leur simplicité les rende séduisants, les modèles géométriques laissent beaucoup à désirer car ils sont incapables de fournir et de justifier des estimations des probabilités d'erreur du type : « *Quelle est la probabilité que ma prédiction soit erronée ?* ». On leur préfère donc souvent des **modèles probabilistes** qui présupposent que les valeurs des variables prédictives et des variables cibles obéissent à certaines lois de probabilité[1].

1. On distingue en fait deux catégories de modèles de ML probabilistes. Pour les introduire il faut cependant faire intervenir la notion de **probabilité conditionnelle** $P(A|B)$, qui indique les chances pour qu'un évènement A se produise si l'on sait par ailleurs qu'un évènement B s'est produit. Les deux catégories de modèles de ML probabilistes sont alors :
Les **modèles génératifs** : ils sont définis par une distribution de probabilité conjointe $P(y,\mathbf{x})$ pour les variables prédictives $\mathbf{x}$ et pour la variable cible y. Les probabilités conditionnelles $P(y|\mathbf{x})$ qui permettent de calculer la fonction de prédiction f, sont alors calculées à partir de ce $P(y,\mathbf{x})$. L'avantage de ces modèles est qu'ils modélisent simultanément les variables cibles et prédictives et permettent donc de simuler des jeux de données. La classification naïve bayésienne est un algorithme de cette espèce.
Les **modèles discriminants** : ils sont défini directement les probabilités conditionnelles $P(y|\mathbf{x})$ sans passer par une distribution conjointe $P(y,\mathbf{x})$. Par l'économie de moyens qu'ils mettent en œuvre, ils offrent souvent de meilleures performances que les modèles génératifs. La régression logistique appartient à cette catégorie.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

7.3 LES PRINCIPAUX ALGORITHMES

Puisqu'en vertu du théorème du *No Free Lunch* aucun algorithme universel n'existe, le data scientist devra faire son marché. Sans être nécessairement un statisticien chevronné, il ne pourra se contenter d'utiliser les algorithmes de manière aveugle mais devra se forger une intuition sur leur fonctionnement et sur leur domaine de validité. Cette lucidité est indispensable pour être capable d'optimiser un modèle.

La brève liste qui termine ce chapitre ne prétend nullement à l'exhaustivité et n'a pas vocation à se substituer à un cours de ML en bonne et due forme. Elle décrit succinctement et sans formalisme les principes de fonctionnement des algorithmes de ML les plus utilisés ainsi que les contextes où ils sont utiles, leurs avantages et leurs limitations. Pour une description plus détaillée des algorithmes on se reportera par exemple à ces deux références en ligne StatSoft, scikit-learn[1].

7.3.1 La régression linéaire

Description

La **régression linéaire** est l'un des modèles de *machine learning* supervisé et paramétrique les plus simples qui soient. L'un de ses principaux mérites est de fournir une illustration pédagogique élémentaire des différents concepts du ML. Il suppose que la fonction de prédiction f qui lie les variables prédictives $x_1,...,x_p$ à la variable cible a la forme : $f(\mathbf{x}) = a_1 x_1 + ... + a_p x_p + b = \mathbf{a} \cdot \mathbf{x} + b$. On préfère parfois remplacer le terme constant b par l'expression $a_0 x_0$, où $x_0 \equiv 1$ est une « variable constante » ce qui confère à l'expression un caractère strictement linéaire.

L'apprentissage du modèle consiste en l'occurrence à calculer les coefficients a et b qui minimisent les erreurs de prédiction sur un jeu de données d'apprentissage. Le plus souvent l'erreur est définie comme la somme des carrés des écarts entre les valeurs prédites $f(\mathbf{x}^{(i)})$ et les valeurs observées $y^{(i)}$. On parle à ce titre de **méthode des moindres carrés**. Les justifications de cette définition de l'erreur sont nombreuses. Au niveau intuitif le fait de prendre le carré évite que les erreurs positives et négatives ne puissent se compenser. À un niveau plus fondamental, le carré permet de fonder la régression linéaire sur un modèle probabiliste[2].

La régression linéaire est utilisée pour l'estimation de certaines tendances en économétrie et dans le marketing lorsqu'on a des raisons de penser qu'il existe une relation linéaire entre la variable explicative et la cible. Établir la relation entre l'augmentation du prix d'un produit et sa demande, évaluer l'impact d'une campagne publicitaire en fonction des frais engagés sont des exemples d'utilisation.

1. StatSoft : http://www.statsoft.com/textbook
sckit-learn : http://scikit-learn.org/stable/user_guide.html

2. Le modèle en question est le modèle dit normal ou gaussien. Il est alors possible d'associer la régression linéaire à un modèle génératif ou discriminatif. Le modèle discriminatif revient à postuler que les variables cibles sont distribuées selon une courbe de Gauss autour d'une valeur moyenne qui dépend linéairement des valeurs prédictives.

Copyright © 2015 Dunod.

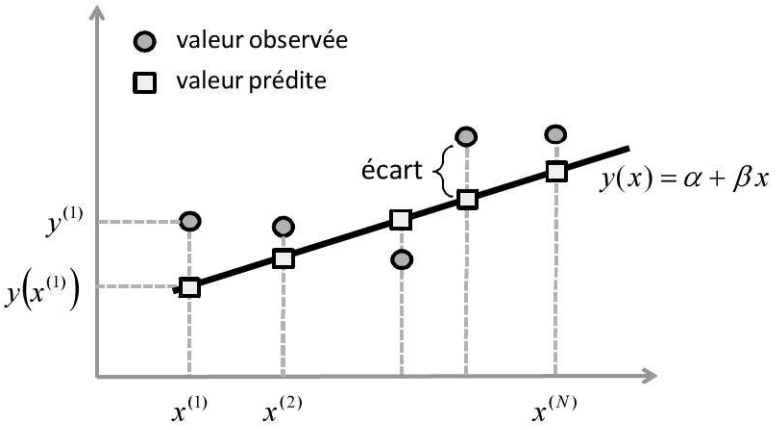


Figure 7.6 — La régression linéaire. L'erreur de prédiction est définie comme la somme des carrées des écarts entre les observations et les prédictions.

Remarquons que si la variable cible est une variable quantitative, les variables prédictives peuvent, elles, être quantitatives ou qualitatives.

Avantages

- Le principal avantage de ce modèle est que l'apprentissage se résume à l'inversion d'une matrice construite à partir des données d'apprentissage. Il existe donc une expression mathématique explicite qu'il suffit de calculer, aucun algorithme numérique complexe n'intervient si bien le calcul d'une prédiction est très rapide.
- Le modèle est en général simple à interpréter.

Inconvénients

- Il faut avoir une bonne raison de croire que la relation que l'on souhaite mettre en évidence est effectivement linéaire.
- Dans sa version élémentaire présentée ici, le modèle est particulièrement sensible aux **valeurs aberrantes** (*outliers*) des données d'apprentissage. Pour pallier à cet inconvénient il existe des méthodes dites de **régularisation**[1] qui pénalisent les valeurs trop grandes des coefficients *ai* et *b*, c'est-à-dire les configurations trop complexes qui caractérisent une situation de surapprentissage.
- Le caractère linéaire du modèle néglige de fait toutes les interactions entre les variables prédictives.

7.3.2 Les *k* plus proches voisins

Description

L'algorithme des ***k* plus proches voisins** (KNN pour *K Nearest Neighbors*) est, dans sa forme élémentaire, un algorithme de classification supervisé et non paramétrique.

1. Les méthodes les plus utilisées sont la régression *Ridge* et la *méthode du lasso*.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

L'idée est simple, elle consiste à supposer qu'une observation est similaire à celle de ses voisins. On suppose chaque observation de l'ensemble d'entraînement représentée par un point dans un espace comportant autant de dimensions qu'il y a de variables prédictives. On suppose par ailleurs qu'il existe une notion de distance dans cette espace et l'on cherche les k points les plus proches de celui que l'on souhaite classer. La classe de la variable cible est, par exemple, celle qui est majoritaire parmi les k plus proches voisins. Certaines variantes de l'algorithme pondèrent chacune des k observations en fonction de leur distance à la nouvelle observation, les observations les plus distantes étant considérées comme les moins importantes.

Une variante de cet algorithme est utilisée par *NetFlix*[1] pour prédire les scores qu'un utilisateur attribuera à un film en fonction des scores qu'il a attribués par le passé à des films similaires.

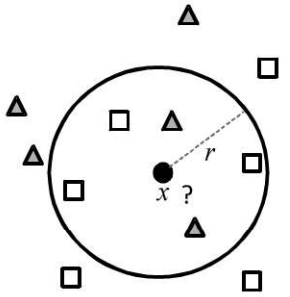


Figure 7.7 — La sélection de k plus proches voisins d'une observation, le point central, que l'on souhaite classer. Les points proches sont ici ceux contenus dans le disque de rayon r. La classe des carrés étant majoritaires, on affectera à l'observation x les caractéristiques que celles des observations représentées par un carré.

Avantages

- On ne présuppose rien sur la distribution des données si ce n'est qu'il existe une notion de similarité.
- Comme pour la régression linéaire, l'une des principales vertus est sa simplicité conceptuelle qui en fait un bon exemple pédagogique.

Inconvénients

- L'algorithme est très sensible au bruit.
- L'apprentissage est uniquement local et ne tient pas compte des effets collectifs à longue distance.
- Lorsque le nombre de variable est grand (> 10) le calcul de la distance peut s'avérer très coûteux. Il faut alors utiliser des techniques de réduction dimensionnelle, voir section 7.3.9.

1. http://cs229.stanford.edu/proj2006/HongTsamis-KNNForNetflix.pdf

Copyright © 2015 Dunod.

7.3.3 La classification naïve bayésienne

Description

La **classification naïve bayésienne** (*Naïve Bayes Classifier*) est un algorithme de classification supervisé très performant en dépit de sa grande simplicité. Pour expliquer son principe, examinons l'exemple d'un filtre antispam. Pour chaque message, le filtre devra prédire son appartenance à l'un ou l'autre des deux catégories « spam » ou « non-spam ». À l'évidence, les fréquences de certains mots diffèrent selon que l'on se trouve dans l'une ou l'autre classe. Ces fréquences sont par ailleurs faciles à estimer lorsqu'on dispose d'un échantillon de messages dont on connaît a priori la classification correcte. Sans hypothèse supplémentaire, les formules qui permettent de prédire la probabilité d'appartenance à une classe en fonction des fréquences observées s'avèrent tellement complexes qu'elles sont inexploitables[1] en pratique. Une hypothèse sauve pourtant la mise, elle consiste à postuler qu'au sein d'une classe, spam ou non-spam, les fréquences des mots sont indépendantes. Sous cette hypothèse, les formules se simplifient à l'extrême et la détermination de la classe la plus probable devient alors aisée[2].

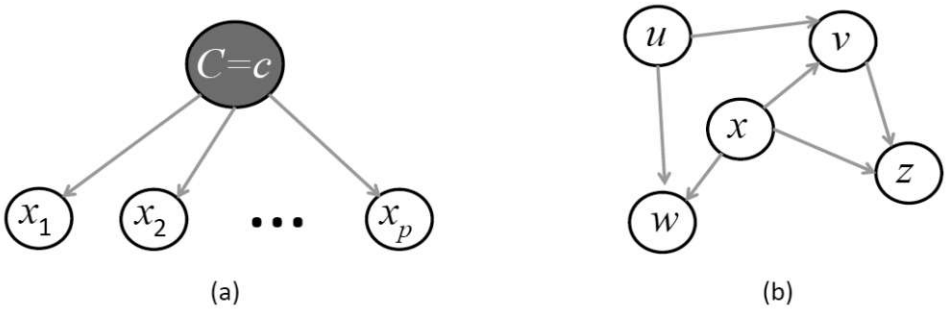


Figure 7.8 — (a) La classification naïve bayésienne repose sur l'hypothèse de l'indépendance des probabilités conditionnelles des variables prédictives (les fréquences de mots dans notre exemple) sachant qu'une catégorie est fixée (spam ou non spam dans l'exemple). (b) Les réseaux bayésiens généralise cette idée à des relations de dépendance plus complexes.

Point remarquable : l'hypothèse d'indépendance précédente est le plus souvent *fausse* en pratique ! Raison pour laquelle on utilise d'ailleurs l'adjectif « naïf ». Malgré tout, l'expérience montre que l'algorithme conduit à de très bonnes prédictions.

La classification naïve bayésienne se généralise à des situations où les relations de dépendances conditionnelles sont plus complexes que celle mentionnée ici. Les modèles associés s'appellent les **réseaux bayésiens**.

1. Elles font intervenir un grand nombre de probabilités conditionnelles où interviennent simultanément la classe et les fréquences des mots, probabilités qu'il est impossible d'estimer en pratique à cause du fléau de la dimension.

2. Si C désigne la classe et F_i les fréquences, on trouve que $P(C|F_1,...F_k)$ est proportionnelle à $P(C)$ $P(F_1|C)...$ $P(F_k|C)$, chaque facteur étant mesurable sur l'échantillon d'apprentissage. Il suffit alors de trouver la classe C qui maximise cette expression.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

Avantages

- La simplicité de l'algorithme lui assure de bonnes performances.
- L'algorithme reste utile et prédictif même dans les situations où l'hypothèse d'indépendance des variables explicatives conditionnées sur les classes est difficile à justifier.

Inconvénients

- Les prédictions de probabilités pour les différentes classes sont erronées lorsque l'hypothèse d'indépendance conditionnelle est invalide.

7.3.4 La régression logistique

Description

La **régression logistique** est un modèle de classification linéaire qui est le pendant de la régression linéaire dans une situation où la variable cible y n'est susceptible de prendre que deux valeurs, disons 0 ou 1. Comme il s'agit d'un modèle linéaire on utilise une fonction de score S des variables prédictives : $S(x) = a_1 x_1 + ... + a_p x_p$. L'idée est de chercher des coefficients $a_1, ..., a_p$ tels que le score $S(x)$ soit positif lorsque les chances d'appartenir au groupe 1 sont grandes et tel que le $S(x)$ soit négatif lorsque les chances d'appartenir au groupe 0 sont grandes. Entre les deux on utilise pour calculer la probabilité $P_1(\mathbf{x})$ d'appartenance au groupe 1 une fonction d'interpolation logistique qui a la forme indiquée dans la figure 7.9. Sa forme explicite est *logit* $(S) = 1/[1+\exp(-S)]$. La théorie des probabilités permet de justifier dans de nombreuses situations que cette forme est bien légitime[1].

Il s'agit donc d'un modèle probabiliste paramétrique et discriminant puisque l'on postule une certaine forme pour la probabilité conditionnelle qui nous intéresse, à savoir $P(y = 1|x) = logit\ (S(x))$.

La recherche des paramètres optimaux $a_1, ..., a_p$ doit se faire par des méthodes numériques, contrairement à la régression linéaire, il n'existe pas de formule explicite.

Les domaines d'application de la régression logistique sont très variés. En médecine il est possible de prédire le taux de mortalité provoqué par certaines affections en fonction de différentes caractéristiques des patients. En ingénierie, on va pouvoir l'utiliser pour prédire des probabilités de pannes pour des systèmes complexes. Enfin dans le marketing la régression logistique permet de prédire la probabilité qu'un individu achète un produit et résilie un contrat en fonction de paramètres connus.

Copyright © 2015 Dunod.

1. C'est le cas en particulier lorsque les variables prédictives $x_1, ..., x_p$ sont distribuées normalement, séparément dans chaque groupe. La validité de l'hypothèse logistique est toutefois plus générale.

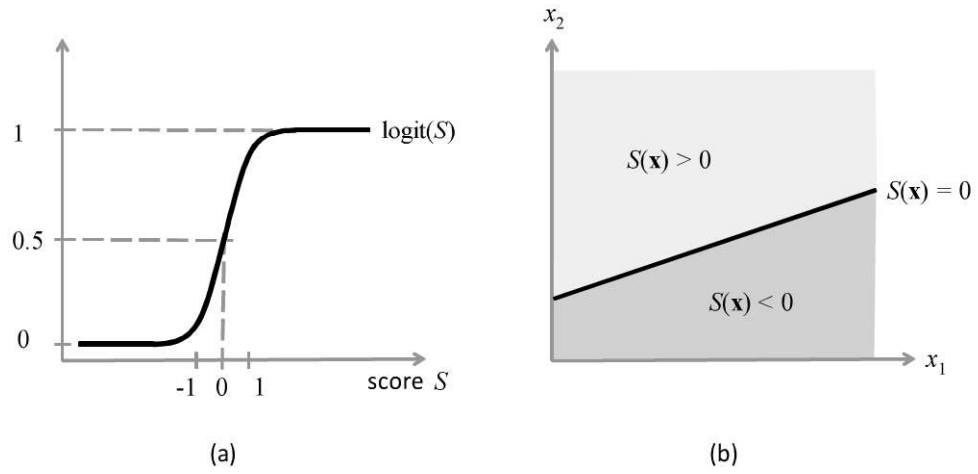


Figure 7.9 — La régression logistique permet de prédire l'appartenance à une parmi deux catégories au moyen d'une fonction score *S*, qui dépend du problème examiné, et de la fonction logistique représentée en (a). Les valeurs positives et négatives de $S(\mathbf{x})$ partitionnent l'espace des observations en deux.

Avantages

- La classification d'une nouvelle observation est extrêmement rapide puisqu'elle se résume essentiellement à l'évaluation d'une fonction de score linéaire S.
- La simplicité même de l'algorithme fait qu'il est peu susceptible de surapprentissage.
- L'interprétation des coefficients a_i du score est aisée. En effet lorsque l'on augmente x_i d'une unité le rapport $P_1(\mathbf{x}) / P_0(\mathbf{x})$ des probabilités d'appartenance aux deux groupes est simplement multiplié par $\exp(a_i)$[1], un fait couramment utilisé en médecine par exemple.

Inconvénients

- L'hypothèse de linéarité du score empêche de tenir compte des interactions entre variables.
- La phase d'apprentissage peut être longue car l'opération numérique d'optimisation des coefficients $a_1,..., a_p$ est complexe.
- L'algorithme est a priori limité aux variables cibles binaires. Un enchaînement de plusieurs régressions logistiques permet cependant, en principe, de surmonter cette limitation.

7.3.5 L'algorithme des *k*-moyennes

Description

L'**algorithme des k-moyennes** est le seul parmi notre liste à appartenir à la catégorie des algorithmes non supervisés. À partir d'un ensemble de *N* observations $x^{(1)},..., x^{(N)}$,

1. Puisque $P_1(\mathbf{x}) / P_0(\mathbf{x}) = \exp[S(\mathbf{x})]=\exp[a_1 x_1 +...+ a_p x_p]$ d'après les définitions.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

chacune étant décrite par p variables, cet algorithme crée une partition en k classes ou clusters homogènes. Chaque observation correspond à un point dans un espace à p dimensions et la proximité entre deux points est mesurée à l'aide de la distance qui les sépare (dans le sens euclidien usuel). Pour être précis, l'algorithme a pour objectif de trouver la partition des N points en k clusters de manière à ce que la somme des carrés des distances des points au centre de gravité du groupe auquel ils sont affectés soit minimale, voir figure 7.10.

Ainsi posé le problème est extrêmement complexe à résoudre[1]. La seule voie praticable est un algorithme heuristique qui fournit une solution approximative. Après avoir choisi initialement, au hasard, k points centraux (ou prototypes), l'algorithme procède par itérations successives de deux étapes :

1. On affecte chacun des N points au plus proche des k points centraux. On a ainsi défini k clusters intermédiaires.
2. On calcule les centres de gravité des k clusters intermédiaires qui deviennent les nouveaux points centraux.

L'itération se poursuit jusqu'à stabilisation des points centraux. Il n'existe pas de garantie théorique de convergence de l'algorithme mais en pratique celle-ci est aisée à vérifier.

Parmi les applications pratiques on trouve en particulier la segmentation d'un marché en niches et les systèmes de recommandation.

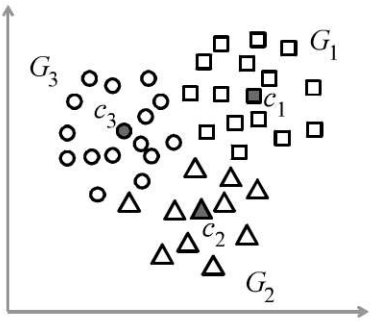


Figure 7.10 — L'algorithme des k-moyennes effectue une partition des données en k groupes homogènes, ici $k = 3$. Les trois centres de gravités sont représentés en foncé.

Avantages

- En pratique l'algorithme est très rapide si bien qu'il est possible de le lancer plusieurs fois pour vérifier la plausibilité de la partition trouvée.
- Se laisse implémenter pour des grands volumes de données.

1. On montre en théorie de la complexité que le problème des k-moyennes est NP-complet ce qui signifie que si l'on trouvait un algorithme efficace pour le résoudre on pourrait en déduire un algorithme efficace pour une grande classe de problèmes difficiles appelés problèmes NP.

Copyright © 2015 Dunod.

- Utilisé conjointement avec un autre algorithme de classification supervisé l'algorithme des *k*-moyennes permet de découvrir des catégories auxquelles on affectera les futures observations.

Inconvénients

- Le nombre *k* de classes relève du choix de l'utilisateur, il n'est pas découvert. Si ce choix ne correspond pas à un partitionnement « naturel » les résultats seront dépourvus de sens.
- La solution obtenue dépend en règle générale des *k* points centraux choisis initialement.

7.3.6 Les arbres de décision

Description

Les **arbres de décision** sont des modèles de ML supervisés et non paramétriques extrêmement flexibles. Ils sont utilisables aussi bien pour la classification que pour la régression. Nous décrirons ici brièvement les principes utilisés pour la classification.

Les arbres de décision utilisent des méthodes purement algorithmiques qui ne reposent sur aucun modèle probabiliste. L'idée de base consiste à classer une observation au moyen d'une succession de questions (ou **critères de segmentation**) concernant les valeurs des variables prédictives x_i de cette observation. Un exemple de critère de sélection pourrait être : « *La variable* x_3 *est-elle plus grande que 45.7 ?* ». Chaque question est représentée par un nœud d'un arbre de décision. Chaque branche sortante du nœud correspond à une réponse possible à la question posée. La classe de la variable cible est alors déterminée par la feuille (ou nœud terminal) dans laquelle parvient l'observation à l'issue de la suite de questions.

La phase d'apprentissage consiste donc à trouver les bonnes questions à poser pour ranger correctement un ensemble $\mathbf{x}^{(1)},\ldots, \mathbf{x}^{(N)}$ de N observations dotées d'étiquettes $y^{(1)},\ldots, y^{(N)}$. Trouver un arbre optimal, dans le sens où il minimise le nombre de questions posées, est un problème difficile au sens de la théorie de la complexité[1] et nous en sommes donc réduits à trouver une méthode heuristique.

Les différentes variantes des arbres de décisions utilisent des stratégies différentes pour associer des critères de segmentation aux nœuds. L'objectif cependant est toujours de parvenir à ce que les feuilles soient aussi homogènes que possible dans le sens où elles ne contiennent idéalement que des observations $\mathbf{x}^{(i)}$ appartenant à une seule et même classe $y^{(k)}$. La stratégie consiste dès lors à associer aux nœuds des critères de segmentation qui, décision après décision, accroîtront progressivement cette homogénéité[2].

1. Comme pour l'algorithme des *k*-moyennes il s'agit d'un problème NP-complet. Voir http://barbracoco.dyndns.org/eiyou/data/NPComplete.pdf.

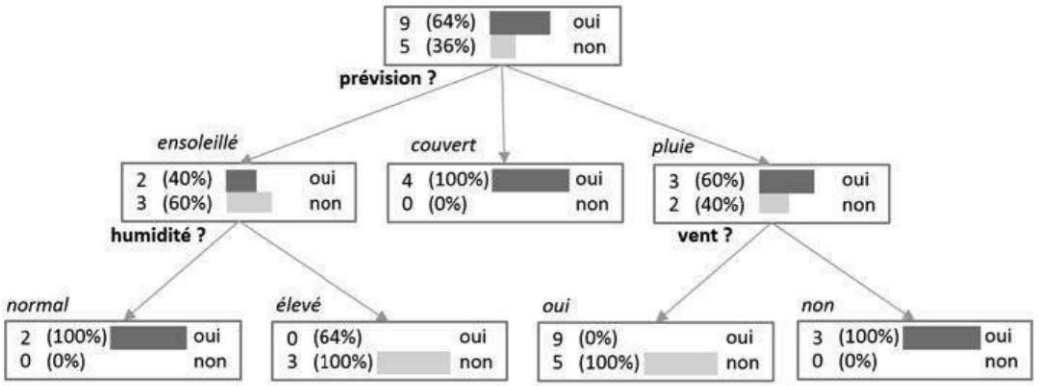
2. Il existe des métriques standards d'homogénéité avec lesquelles on peut mesurer l'homogénéité d'une distribution de valeurs. Les plus courantes sont l'entropie de Shannon et l'indice de diversité de Gini.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

Copyright © 2015 Dunod.

	prévision	température	humidité	vent	oui
1	couvert	froid	normal	oui	oui
2	couvert	chaud	élevé	non	oui
3	couvert	chaud	normal	non	oui
4	couvert	moyen	élevé	oui	oui
5	pluvieux	froid	normal	oui	non
6	pluvieux	moyen	élevé	oui	non
7	pluvieux	froid	normal	non	oui
8	pluvieux	moyen	élevé	non	oui
9	pluvieux	moyen	normal	non	oui
10	ensoleillé	chaud	élevé	non	non
11	ensoleillé	chaud	élevé	oui	non
12	ensoleillé	moyen	élevé	non	non
13	ensoleillé	froid	normal	non	oui
14	ensoleillé	moyen	normal	oui	oui

(a)



(b)

Figure 7.11 — Construction d'un arbre de décision :
(a) l'ensemble d'apprentissage contient 14 observations décrites par quatre variables prédictives et une variable cible que l'on veut prédire.
(b) la succession de questions posées homogénéise progressivement la répartition des observations comme le montrent les histogrammes.

La figure 7.11 illustre un exemple dans lequel il s'agit de déterminer si un individu décide de pratiquer ou non une activité de plein air en fonction de cartes variables prédictives de nature météorologique. La variable cible est la décision de jouer ou non. Les histogrammes attachés à chaque nœud montrent la progression du processus d'homogénéisation sur les données d'apprentissage à mesure que l'on s'approche des feuilles.

La question de la profondeur de l'arbre qu'il faut retenir est délicate et est directement liée au problème du surapprentissage. Exiger que toutes les observations soient parfaitement rangées peut rapidement mener au surapprentissage. Pour cette raison on décide généralement de ne plus rajouter de nœuds lorsque la profondeur de l'arbre excède un certain seuil, qui caractérise la complexité maximale de l'arbre de décision, ou lorsque le nombre d'observations par feuille est trop faible pour être représentatif des différentes classes (on parle de pré-élagage). On pratique aussi des opérations d'**élagage** a posteriori (*prunning*) sur des arbres dont les feuilles sont homogènes en utilisant un jeu de données distinct (*prunning set*) de celui qui a permis la construction de l'arbre original.

Il existe également des versions des arbres de décision adaptée aux problèmes régression[1] mais nous ne les examinerons pas ici.

Les arbres de décisions interviennent par ailleurs comme brique de base de l'algorithme plus sophistiqué des forêts aléatoires que nous présenterons au paragraphe suivant.

Avantages

- Les variables explicatives en entrée peuvent être aussi bien qualitatives que quantitatives.
- La phase de préparation des données est réduite. Ni normalisation, ni traitement des valeurs manquantes ne sont indispensables.
- On tient compte des interactions entre variables car aucune hypothèse de linéarité n'est nécessaire.
- On traite les problèmes de classification dans toute leur généralité (le nombre de classes n'est pas limité à deux comme dans la régression logistique p.ex.)
- À condition que l'arbre soit peu profond, il est souvent possible d'interpréter le processus de classification comme l'application d'un ensemble de règles intelligibles.

Inconvénients

- Il existe un risque de surapprentissage si l'arbre n'est pas correctement élagué.
- Le critère de segmentation affecté au premier nœud possède une très grande influence sur le modèle de prédiction, si bien qu'un jeu de données peu représentatif pourra mener à des conclusions totalement erronées.

1. On rencontre souvent l'acronyme anglais CART pour *Classification And Regression Tree*.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

7.3.7 Les forêts aléatoires

Description

Le but de l'algorithme des **forêts aléatoires** est de conserver la plupart des atouts des arbres de décision tout en éliminant leurs inconvénients, en particulier leur vulnérabilité au surapprentissage et la complexité des opérations d'élagage. C'est un algorithme de classification ou de régression non paramétrique qui s'avère à la fois très flexible et très robuste. De conception récente (2001), il est toujours l'objet de travaux de recherche[1].

L'algorithme des forêts aléatoires repose sur trois idées principales :

1. À partir d'un échantillon initial de N observations $\mathbf{x}^{(1)},\ldots, \mathbf{x}^{(N)}$, dont chacune est décrite au moyen de p variables prédictives, on crée « *artificiellement* » B nouveaux échantillons de même taille N par tirage avec remise. On appelle cette technique le **bootstrap**. Grâce à ces B échantillons, on entraîne alors B arbres de décisions différents.
2. Parmi les p variables prédictives disponibles pour effectuer la segmentation associée au nœud d'un arbre, on n'en utilise qu'un nombre $m < p$ choisies « *au hasard* ». Celles-ci sont alors utilisées pour effectuer la meilleure segmentation possible.
3. L'algorithme combine plusieurs algorithmes « faibles », en l'occurrence les B arbres de décisions, pour en constituer un plus puissant en procédant par vote. Concrètement, lorsqu'il s'agit de classer une nouvelle observation $\mathbf{x}$, on la fait passer par les B arbres et l'on sélectionne la classe majoritaire parmi les B prédictions. C'est un exemple d'une **méthode d'ensemble**.

Le nombre B d'arbres s'échelonne généralement entre quelques centaines et quelques milliers selon la taille des données d'apprentissage. Le choix du nombre m de variables à retenir à chaque nœud est le résultat d'un compromis. Il a été démontré que les prédictions d'une forêt aléatoire sont d'autant plus précises que les arbres individuels qui la composent sont prédictifs et que les corrélations entre prédictions deux arbres différents sont faibles. Augmenter le nombre m de variables augmente la qualité de prédiction des arbres individuels mais accroît aussi les corrélations entre arbres. Une valeur m de l'ordre de $\sqrt{p}$ constitue un bon compromis.

Comparées aux autres algorithmes que nous avons décrits jusque-là, les forêts aléatoires apportent des informations inédites et précieuses :

- **Estimation de l'erreur de prédiction** : la démarche de validation croisée est incorporée à l'algorithme, et n'a donc pas besoin d'être menée séparément. Dans chacun des B échantillons de *bootstrap*, environ un tiers des observations est réservé pour estimer l'erreur de prédiction.
- **Estimation de l'importance des variables** : pour estimer l'importance d'une variable x_j sur la classification on permute ses valeurs aléatoirement parmi le

1. Une bonne introduction au sujet est disponible en ligne ici : https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm.

Copyright © 2015 Dunod.

tiers des observations réservées. On évalue ensuite l'impact moyen de cette permutation sur les prédictions des B arbres. Plus l'impact est significatif plus la variable sera considérée comme importante.

- **Définition de proximités entre observations** : à chaque couple d'observations $(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ est associé un compteur. On parcourt les B arbres et à chaque fois qu'un couple d'observations est rangé dans la même feuille on incrémente le compteur associé. La moyenne de chaque compteur sur les B arbres constitue alors une mesure de proximité.

Avantages

- C'est l'un des meilleurs algorithmes disponibles à l'heure actuel pour ce qui est de la précision des prédictions.
- Il ne souffre pas de surapprentissage.
- Il est très utile en phase exploratoire pour se faire une idée du pouvoir prédictif des données et de l'importance respective des différentes variables prédictives.
- Il fournit une notion objective de similarité entre observation que l'on peut utiliser dans une démarche d'apprentissage non supervisée.
- Il incorpore une étape de validation croisée.
- Il peut traiter des données avec des milliers de variables prédictives.
- Il conserve une bonne puissance prédictive même lorsqu'une grande partie des données est manquante.

Inconvénients

- La complexité de l'algorithme rend son implémentation délicate. Il est préférable d'avoir recours à une implémentation d'une librairie existante.
- On ne conserve pas le caractère intelligible des arbres de décisions.

7.3.8 Les machines à vecteurs de support

Description

Les machines à vecteurs de support, ou séparateurs à vaste marge (SVM), sont des algorithmes de classification binaire non linéaires extrêmement puissants. De conception récente[1], ils remplacent dans beaucoup de circonstances les systèmes de neurones beaucoup plus coûteux en temps d'apprentissage.

Le principe des SVM consiste à construire une bande séparatrice non linéaire de largeur maximale qui sépare deux groupes d'observations comme le montre la figure 7.12 et à l'utiliser pour faire des prédictions. L'astuce qu'utilisent les SVM pour y parvenir consiste à utiliser une transformation φ non linéaire qui envoie les points $\mathbf{x}^{(1)},\ldots, \mathbf{x}^{(N)}$ de l'espace original à p dimensions (p étant le nombre de variables prédictives) vers des nouveaux points $\varphi(\mathbf{x}^{(1)}),\ldots, \varphi(\mathbf{x}^{(N)})$ dans un espace de dimension

1. Une bonne introduction aux SVM est disponbile ici : http://www.work.caltech.edu/~boswell/IntroToSVM.pdf.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

beaucoup plus grande que p, où ils seront « plus faciles à séparer », et dans lequel il sera alors possible de trouver une bande séparatrice *linéaire*.

Trouver un séparateur linéaire de largeur maximale est une tâche simple pour laquelle de nombreux algorithmes efficaces ont été conçus depuis des décennies. Ainsi, le problème difficile original est remplacé par un problème plus simple au prix du passage à une dimension plus élevée.

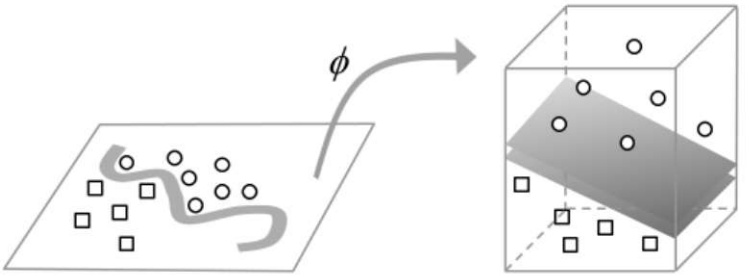


Figure 7.12 — Les SVM permettent de trouver une bande séparatrice non linéaire de largeur maximale entre deux groupes d'observations. L'astuce consiste à trouver une transformation φ non linéaire qui représente les points originaux dans un espace de dimension beaucoup plus grande où il sera plus facile de trouver un séparateur linéaire.

À partir de là, la difficulté consiste à trouver une transformation φ appropriée. Par chance, la connaissance explicite de φ n'est en fait pas nécessaire et ceci pour une raison simple. Il se trouve en effet que le calcul d'une bande de séparation linéaire de largeur maximale ne fait intervenir que les distances et les angles[1] entre les points que l'on souhaite séparer (et non pas leur coordonnées explicites). Ces points sont en l'occurrence les images $\varphi(\mathbf{x}^{(1)}),\ldots, \varphi(\mathbf{x}^{(N)})$ des points originaux. Par conséquent, la connaissance de la seule fonction $K(\mathbf{x},\mathbf{y}) \equiv \varphi(\mathbf{x}) \cdot \varphi(\mathbf{y})$, qui donne les produits scalaires entre points transformés, que l'on appelle **noyau**, suffirait donc pour trouver notre bande séparatrice linéaire dans l'espace de grande dimension et, par contrecoup, la bande séparatrice non linéaire dans l'espace original. On appelle ce tour de passe-passe le **kernel trick**. Reste encore à trouver le noyau K ! Le choix de la fonction K ne peut-être totalement arbitraire puisqu'elle doit représenter un produit scalaire. Heureusement pour nous, des fonctions K convenables et utiles en pratique ont été répertoriées par les statisticiens. Les deux plus utilisées par les SVM sont le noyau polynomial et le noyau gaussien[2].

Il est possible de rendre l'algorithme précédent plus flexible en introduisant des **marges souples** qui admettent qu'une certaine proportion des observations, que l'on peut définir à l'aide d'un paramètre supplémentaire C, puisse être mal classée.

1. Les produits scalaires en fait.
2. Le noyau polynomial vaut $K(\mathbf{x},\mathbf{y})=(\mathbf{x} \cdot \mathbf{y}+1)^d$ et le noyau gaussien $K(\mathbf{x},\mathbf{y})=\exp[-||\mathbf{x}-\mathbf{y}||/2\sigma^2]$.

Copyright © 2015 Dunod.

Avantages

- Il permet de traiter des problèmes avec un très grand nombre de dimensions et d'éviter le fléau de la dimension.
- Il permet de traiter des problèmes de classification non linéaire complexes.
- Les SVM constituent une alternative aux systèmes de neurones car ils répondent au même besoin de classification non linéaire tout en étant beaucoup plus faciles à entraîner que ces derniers.

Inconvénients

- Le choix de la fonction noyau K est délicat et possède un caractère un peu mystérieux qui ne peut être étayé que par l'expérience.
- Bien que l'algorithme puisse être entraîné avec des ensembles de données de plusieurs dizaines de milliers d'observations il n'est cependant pas scalable. On estime que sa complexité croit comme N^3 où N est le nombre d'observations.
- Il est souvent moins performant que les forêts aléatoires.

7.3.9 Techniques de réduction dimensionnelle

Il existe plusieurs raisons pour chercher à réduire le nombre de variables explicatives d'un ensemble d'observations :

- Le temps d'exécution d'un algorithme d'apprentissage dépend simultanément de la taille N de l'échantillon d'apprentissage et du nombre p de variables explicatives.
- La taille N d'un échantillon significatif croît elle-même avec le nombre de ces variables, c'est le fléau de la dimension que nous avons déjà mentionné à la section 7.1.4.
- Un modèle prédictif avec peu de variables est plus facile à interpréter et plus facile à visualiser.

Plusieurs stratégies ont été conçues dans ce but. Elles permettent toutes de diminuer le nombre de variables tout en conservant l'essentiel du pouvoir prédictif des variables originales. Les trois principales sont les suivantes :

- L'**analyse factorielle** (FA, au sens anglo-saxon du terme : *factor analysis*) consiste à expliquer les p variables observées $x_1,...,x_p$ au moyen d'un nombre $k < p$ de facteurs $f_1,...f_k$ explicatifs que l'on appelle des **variables latentes**. En pratique on cherche à exprimer les x_i comme combinaison linéaire des f_j.
- L'**analyse en composantes principales** (ACP) est dans son principe l'inverse de l'analyse factorielle. Elle consiste à chercher un nombre restreint de combinaisons linéaires des variables originales $x_1,...,x_p$ pour créer des composantes $c_1,...,c_k$ non corrélées qui expliquent l'essentiel de la variabilité des variables originales. En pratique elle fournit des résultats proches de l'analyse factorielle.
- Les approches de type **forward selection** consistent à rajouter, une à une, des variables prédictives au modèle en choisissant à chaque étape parmi les variables

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

non encore utilisées celle qui est la plus corrélée à la variable cible et en stoppant ce processus lorsque ce niveau de corrélation descend en dessous d'un certain seuil fixé par avance.

Aucune de ces techniques n'est propre au *machine learning*, elles sont présentées dans tout ouvrage d'introduction à la statistique[1].

Certains algorithmes qui se basent sur les distances entre les points représentatifs, comme les *k*-moyennes ou les *k* plus proches voisins peuvent être utilisés conjointement avec une technique dite de projection aléatoire[2] qui permet de représenter un ensemble de points dans un espace de grande dimension dans un espace de dimension plus petit tout en préservant avec une grande précision les distances entre toutes les paires de points.

7.4 ILLUSTRATIONS NUMÉRIQUES

Le but de cette dernière section est d'illustrer quelques-uns des concepts et des algorithmes qui viennent d'être présentés au moyen d'exemples. Toutes les analyses et les visualisations de cette section ont été faites à l'aide du logiciel *Data-Science-Studio* (*DSS*) édité par la société *Dataiku* dont la *Community Edition* est disponible gratuitement[3].

Le contexte métier utilisé comme fil rouge est celui d'un site de e-commerce pour lequel on souhaite prédire les revenus que vont générer de nouveaux clients.

L'ensemble de données utilisé possède 10 000 lignes[4], les variables qui décrivent les clients sont décrites dans la figure 7.13.

Deux datasets additionnels, `CountryGDP` et `CountryPopulation`, indiquent respectivement le PIB et la population du pays correspondant à l'adresse IP de la requête du client.

Copyright © 2015 Dunod.

1. StatSoft, http://www.statsoft.com/Textbook.
2. Le résultat de base est le lemme de Johnson-Lindenstrauss : http://en.wikipedia.org/wiki/Johnson%E2%80%93Lindenstrauss_lemma.
3. La version Community Edition de Data Science Studio est disponible gratuitement ici : http://www.dataiku.com/dss/editions/.
Les jeux de données utilisés dans ce chapitre sont disponibles ici : http://downloads.dataiku.com/training/livre-big-data/.
4. Le dataset utilisé est `customer`.

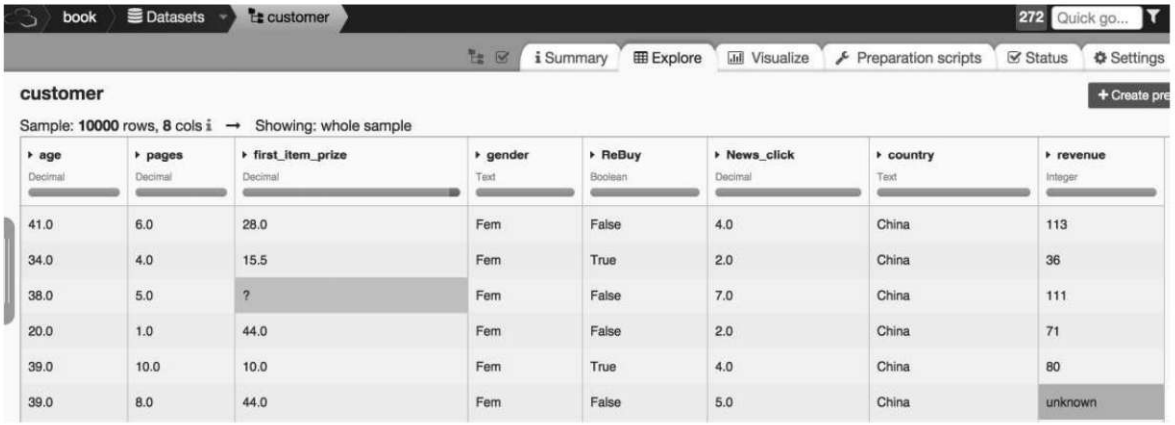


Figure 7.13 — Un fichier clients dont les variables prédictives sont `age` (l'âge), `pages` (le nombre de pages du site visité), `first_item_prize` (le prix du premier article acheté), `gender` (masculin ou féminin), `ReBuy` (le client a-t-il acheté le premier article plus d'une fois ?), `News_click` (nombre de fois où le client a cliqué sur une campagne de publicité du site), `country` (le pays dont provient l'adresse IP), `revenue` (le revenu généré par le client sur le site). Le nom du dataset est `customer`. © Dataiku.

7.4.1 Nettoyage et enrichissement des données

Bien que cette étape ne fasse pas partie à proprement parler du *machine learning*, nous la décrivons brièvement car elle illustre bien le type de tâches que doit mener un data scientist avant même d'entreprendre l'analyse des données, voir chapitres 5 et 6. En l'occurrence elle se réduit à deux opérations :

- Le **nettoyage** : il s'agit de repérer puis de supprimer les lignes mal renseignées. Le plus commode est d'utiliser pour cela un script *Python*, voir figure 7.14.
- L'**enrichissement** : au dataset de base on rajoute les informations liées au PIB et la population du pays de provenance de la requête, ceci au moyen d'un script *Python* ou d'un script *SQL*.

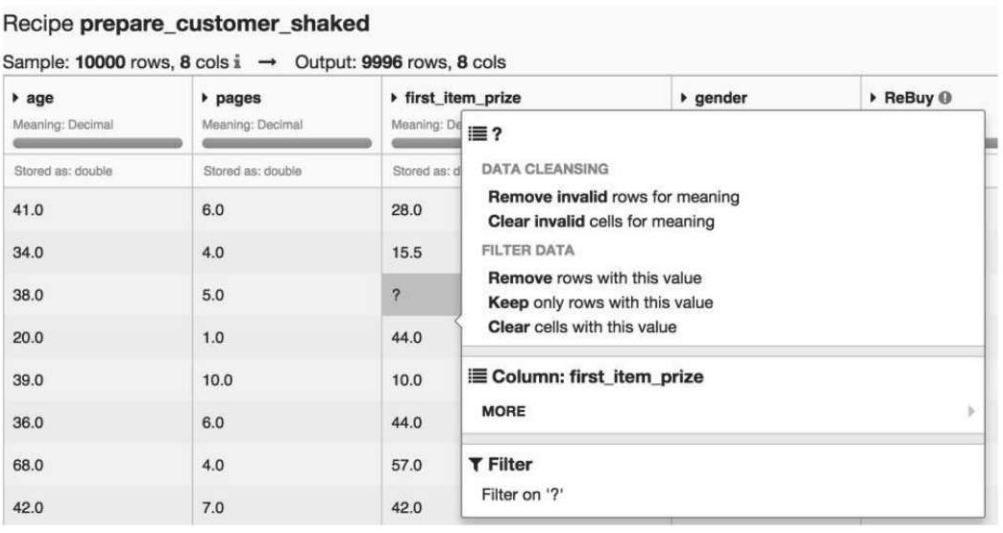


Figure 7.14 — Suppression dans DSS des lignes comportant des valeurs manquantes ou des données invalides. © Dataiku.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Quelques lignes de code Python, que l'on peut saisir directement dans DSS, suffisent pour enrichir les données initiales de deux nouveaux attributs : `CountryGDP` et `CountryPopulation` :

```
# -*- coding: utf-8 -*-
from dataiku import Dataset
import pandas as pd
# Input datasets
customer_shaked = Dataset("customer_shaked").get_dataframe()
CountryGDP = Dataset("CountryGDP").get_dataframe()
CountryPopulation = Dataset("CountryPopulation").get_dataframe()
CountryGDP.columns=['country','GDP_inhab']
CountryPopulation.columns=['country','population']
customer_shaked=pd.merge(customer_shaked,CountryGDP)
customer_shaked=pd.merge(customer_shaked,CountryPopulation)
# Output datasets
customer_enriched = Dataset("customer_enriched")
customer_enriched.write_with_schema(customer_shaked)
```

Pour plus d'informations sur ces deux types d'opération, le lecteur pourra se référer à la documentation fournie avec *DSS*.

7.4.2 Profondeur d'un arbre et phénomène de surapprentissage

Notre première illustration concerne le surapprentissage. Dans un premier temps nous renonçons à prédire le revenu attendu d'un nouveau client et nous nous contenterons de prédire une variable binaire qui indique si un client est susceptible ou non de générer plus de revenus que la moyenne. Il nous faut donc remplacer la variable revenu par cette nouvelle variable binaire. Voici le script Python utilisé à cet effet :

```
# -*- coding: utf-8 -*-
from dataiku import Dataset
# Input datasets
customer_enriched = Dataset("customer_enriched").get_dataframe()
customer_enriched['revenue_HigherMean']=
customer_enriched['revenue']> customer_enriched['revenue'].mean()
del customer_enriched['revenue']
# Output datasets
customer_classif = Dataset("customer_classif")
customer_classif.write_with_schema(customer_enriched)
```

Pour l'algorithme de décision, nous utiliserons un arbre de décision[1]. La création d'un modèle de prédiction est aisée dans DSS. Dans un premier temps, il suffit de spécifier les données d'entraînement et la variable cible souhaitée. Celle-ci est en l'occurrence binaire. Dans un deuxième temps, DSS propose une liste d'algorithmes adaptés, entre autres : les forêts aléatoires, la régression logistique et la machine à vecteurs de support. Tous sont des algorithmes de décision binaire, voir figure 7.15.

1. Dans DSS on utilise pour cela une forêt aléatoire à un seul arbre.

Copyright © 2015 Dunod.

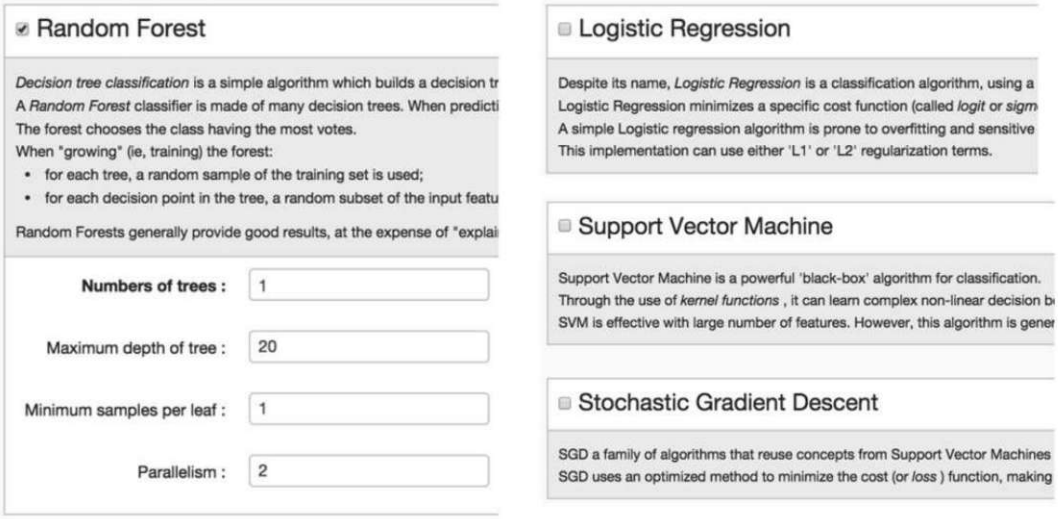


Figure 7.15 — La liste des algorithmes de décision binaires proposés par DSS. L'algorithme retenu, celui des forêts aléatoires, demande à être paramétré. © Dataiku.

DSS indique que `country` est une variable nominale à 50 niveaux. En tenir compte nécessiterait d'introduire autant de variables binaires, ce qui n'est pas conseillé à ce stade de l'analyse.

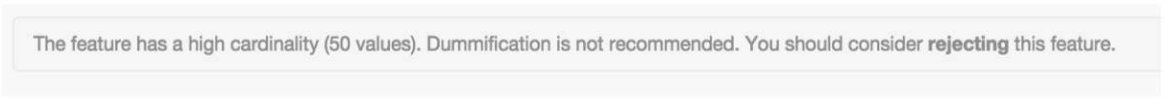


Figure 7.16 — La notification de DSS sur le grand nombre de niveau de la variable nominative `country`. © Dataiku.

De plus, `country` est vraisemblablement fortement corrélée aux variables numériques `GDP_inahb` et `population` qui ont été rajoutées durant la phase d'enrichissement et qui contiennent déjà l'essentiel de l'information. Dans un premier temps on va donc exclure cette variable du modèle.

Bien choisir la profondeur d'un arbre de décision est essentiel pour éviter le surapprentissage. D'expérience, une profondeur maximale égale à 20 s'avère en général un premier choix raisonnable. Notre objectif dans cette section sera d'optimiser la valeur de ce paramètre.

Lorsque l'on lance la phase d'apprentissage dans *DSS*, l'outil scinde automatiquement les données en un jeu d'apprentissage (80 % des données) et un jeu de validation (20 % des données) à partir duquel l'outil évalue les performances de l'algorithme, voir figure 7.17.

Pour représenter graphiquement les performances d'un prédicteur binaire (notre arbre de décision en l'occurrence) on utilise d'ordinaire ce que l'on appelle une **courbe ROC**, le concept sera décrit en toute généralité au chapitre 8. Décrivons-la rapidement dans notre contexte.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

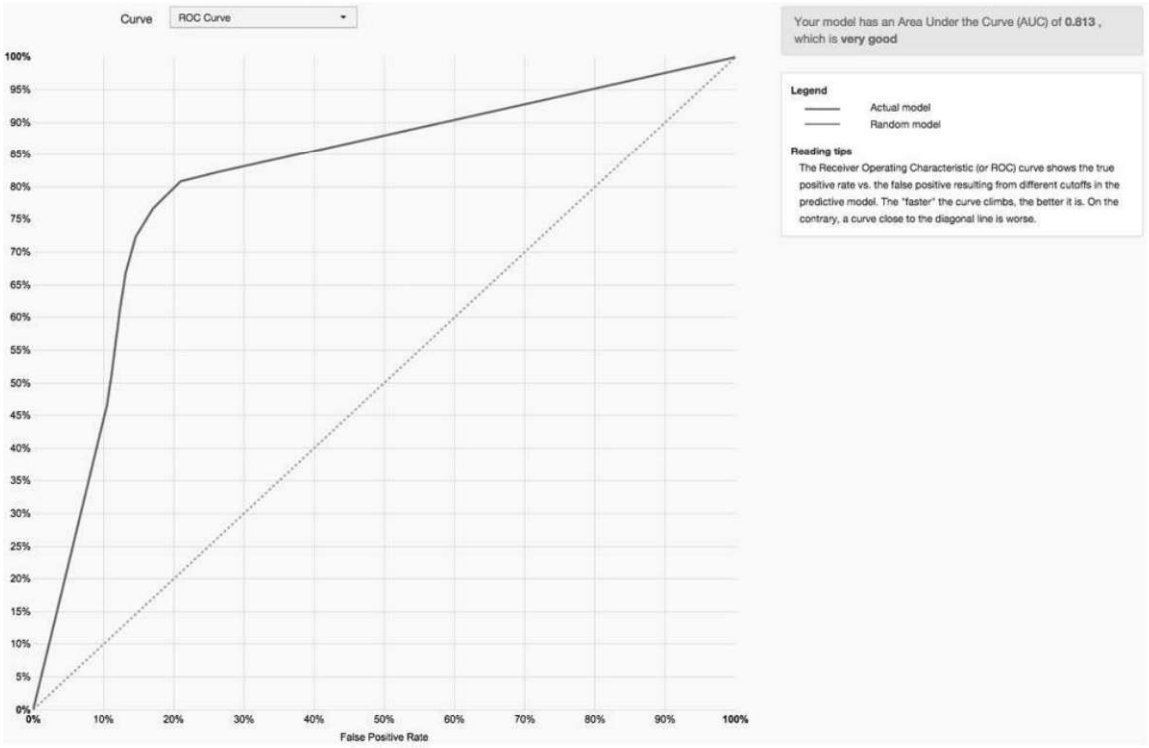


Figure 7.17 — La courbe ROC représente en ordonnée le taux de vrais positifs (la proportion de clients qui génèrent un revenu supérieur à la moyenne) et le taux de faux positifs (la proportion des clients dont l'algorithme a prédit à tort qu'ils généreraient un revenu supérieur à la moyenne) en fonction d'un seuil de décision. © Dataiku.

Pour chaque client, notre arbre de décision génère en interne, en fonction des valeurs des variables prédictives, un nombre p compris entre 0 et 1 qui est une sorte d'estimation des chances que le client soit effectivement un bon client. On peut alors décider, par exemple, qu'un client sera un bon client à partir du moment où p dépasse le seuil de 0,5. Mais, selon les besoins du métier, on peut aussi faire varier ce seuil s et remplacer 0,5 par d'autres valeurs qui correspondent à différents niveaux d'exigence quant à la définition de ce qu'est un bon client. Une valeur de seuil s proche de 1 revient par exemple à être très exigeant. Pour construire la courbe ROC, on fait varier ce seuil s entre 0 et 1 et, pour chaque valeur de s, on reporte en ordonnée la proportion réelle de clients qui génèrent un revenu supérieur à la moyenne, on les appelle les « **vrais positifs** » et en abscisse la proportion de clients prédite à tort par l'algorithme comme générant des revenus supérieurs alors que ce n'est pas vrai, on les appelle les « **faux positifs** ».

Un algorithme idéal qui ne commettrait aucune erreur (0 % de faux positifs et 100 % de vrais positifs) correspondrait à une courbe qui possède un coude à angle droit. L'efficacité d'un algorithme de prédiction binaire, comme l'est celui des forêts aléatoires, peut par conséquent se mesurer au moyen d'un score qui compare la surface sous la courbe (**AUC** pour *Area Under the Curve*) à la surface sous la courbe idéale. Dans notre exemple ce score vaut 0,813, ce qui est plutôt très bon.

Copyright © 2015 Dunod.

Se pose dès lors la question : « *Est-il possible d'améliorer ce score en jouant sur la profondeur de l'arbre ?* ». Pour y répondre on reporte l'évolution du score AUC en fonction de la profondeur de l'arbre. *DSS* permet de tracer cette courbe au moyen de script *Python*.

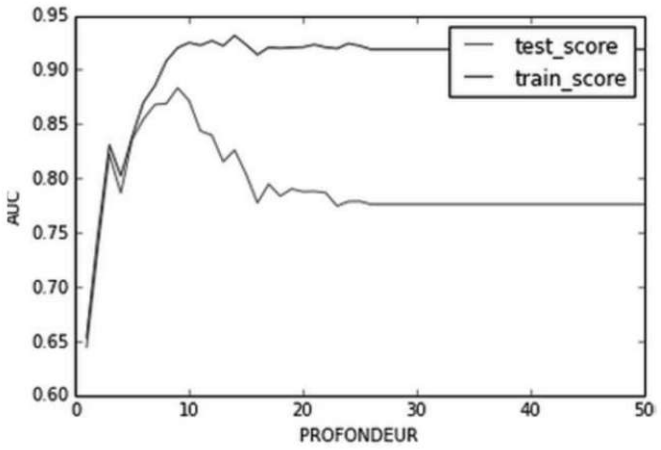


Figure 7.18 — Évolution du score de performance des prédictions en fonction de la profondeur de l'arbre de décision pour le jeu de données de test et pour les données d'entraînement.

Si l'on augmente indéfiniment la profondeur de l'arbre, on observe sur la figure 7.18 que les performances de prédiction de l'algorithme sur les données d'entraînement se rapprochent de la valeur idéale d'un AUC = 1. Mais, dans le même temps, l'AUC sur le jeu de test augmente dans un premier temps avant de diminuer avec une valeur optimale pour la profondeur située autour de 10. Cela signifie qu'au-delà de cette valeur optimale, l'algorithme se contente d'« apprendre par cœur » les données d'apprentissage et perd en capacité de généralisation. C'est le phénomène de surapprentissage !

La profondeur 20 que nous avions choisie initialement était donc dans la zone de surapprentissage. En la faisant passer cette profondeur à 10 on gagnera en performance, voir figure 7.19.

On a effectivement amélioré légèrement notre score, AUC = 0,848 > 0,813.

7.4.3 Apport du « feature engineering »

Dans cette section, nous allons montrer pourquoi les algorithmes linéaires sont limités lorsqu'il s'agit de modéliser des phénomènes non linéaires. Cependant nous montrerons aussi comment il est parfois possible de contourner ces limitations en construisant de nouvelles variables à partir de celles dont on dispose déjà, une démarche que l'on appelle le **feature engineering**.

Nous nous placerons cette fois-ci dans un contexte où l'on veut prédire le revenu exact généré par un client et non plus seulement identifier les bons clients[1].

1. Les données utilisées sont tirées du dataset `customer_enriched`.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

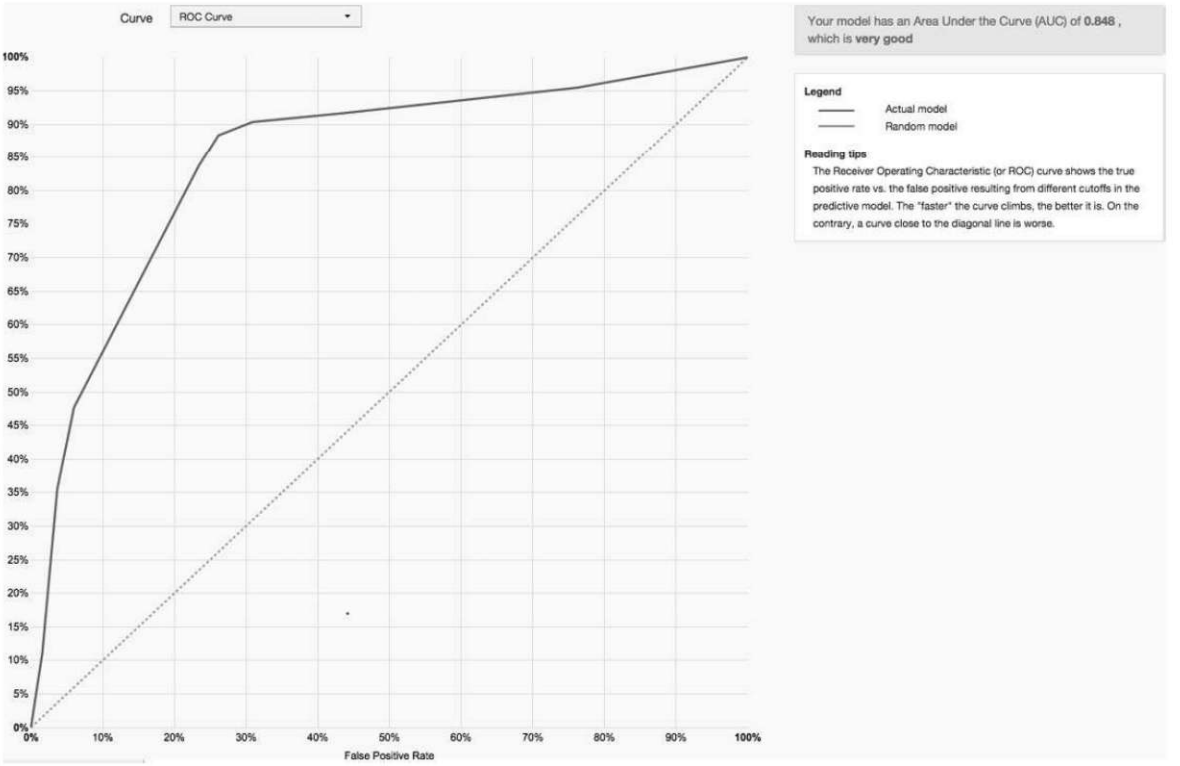


Figure 7.19 — Amélioration du score AUC de l'arbre de décision par un choix optimal de la profondeur de l'arbre. © Dataiku.

Nous allons entraîner puis comparer les prédictions d'une régression linéaire avec celle d'un arbre de décision qui est évidemment non linéaire. Comme précédemment, nous n'utiliserons pas la variable `country` et nous laisserons *DSS* effectuer la séparation entre données d'entraînement et données de test.

Afin de comparer la qualité des deux algorithmes nous comparerons leurs prédictions respectives avec les valeurs réelles en les reportant sur deux diagrammes de dispersion (ou *scatterplots*, voir section 8.2). Visuellement, une bonne prédiction se traduira par un nuage de points concentrés le long de la diagonale $y=x$. On constate sur la figure 7.20, que l'alignement des points pour l'arbre de décision est nettement meilleur que pour la régression où l'on décèle un alignement légèrement incurvé.

Quantitativement, la qualité de cet ajustement peut être évaluée au moyen d'un coefficient de corrélation[1] ρ compris entre 0, lorsqu'il n'existe aucune corrélation, et 1 lorsque la corrélation est parfaite (et positive). En l'occurrence $\rho = 0.62$ pour la régression linéaire et $\rho = 0.82$ pour l'arbre de décision.

1. Il s'agit du coefficient de corrélation linéaire de Pearson compris entre –1 et 1. Des valeurs proches de zéro signifient une corrélation faible. Des valeurs proches de 1 ou –1 indique des corrélations positives ou négatives significatives.

Copyright © 2015 Dunod.

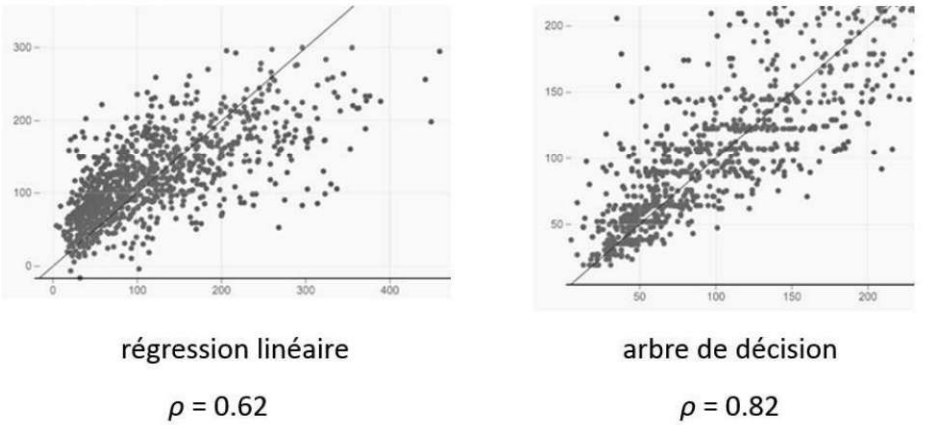


Figure 7.20 — Deux diagrammes de dispersion qui représentent le revenu prédit, en ordonnée, en fonction du revenu observé, en abscisse, pour la régression linéaire et pour un arbre de décision optimal. On constate que le coefficient de corrélation pour l'arbre de décision est plus élevé ce qui indique une meilleure qualité de prédiction. © Dataiku.

Reste encore à comprendre pourquoi l'arbre de décision obtient un meilleur score que la régression linéaire. Rappelons qu'il est possible d'évaluer l'importance d'une variable prédictive au moyen de l'algorithme des forêts aléatoire, une fonctionnalité très utile implémentée par *DSS*, voir figure 7.21.

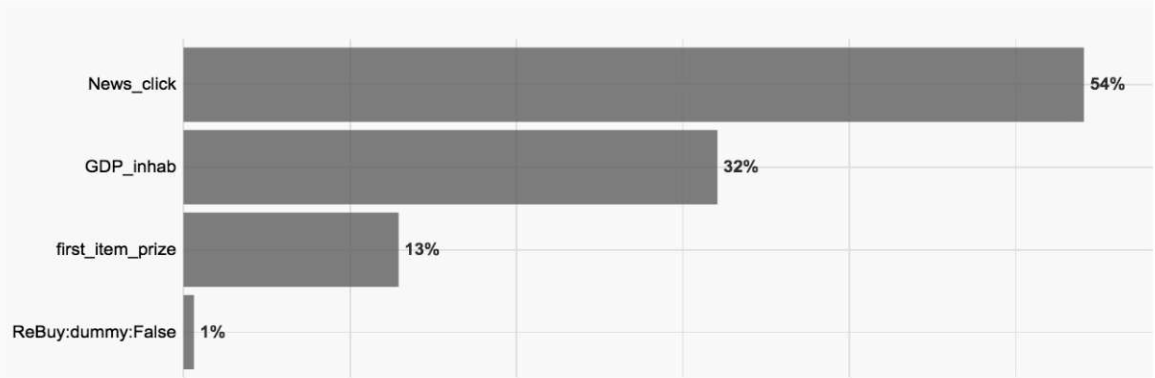


Figure 7.21 — L'histogramme produit par DSS qui représente l'importance des variables prédictives pour notre estimation des revenus générés. La variable `News-click` est de loin la plus significative. © Dataiku.

On constate sur l'histogramme de la figure 7.21 que la variable la plus importante pour notre arbre de décision est, de loin, `News-click` qui correspond au nombre de fois qu'un client a cliqué sur un bandeau publicitaire du site.

Imaginons qu'après recherche et discussion avec le marketing, nous soyons amenés à formuler l'hypothèse que cette variable présente un effet de seuil : au-dessous de quatre clics les clients ne feraient que passer alors qu'un nombre supérieur à quatre correspondrait à l'amorce d'un comportement addictif susceptible de mener à un achat. Voilà un effet qu'une régression linéaire ne saurait capter au contraire d'un arbre de décision.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Pour vérifier cette hypothèse, on procède en deux étapes. On commence par définir une nouvelle variable booléenne `News_click_seuil`[1] qui prend la valeur 1 sitôt que `News_click` dépasse la valeur de seuil égale à 4, c'est le *feature engineering*. On entraîne ensuite un modèle linéaire (sic) en substituant la nouvelle variable `News_click_seuil` à l'ancienne variable `News_click`. Après entraînement de ce nouveau modèle linéaire affiche une corrélation de 0,69 améliorant le score du modèle linéaire original sans toutefois parvenir au niveau de performance de l'arbre de décision.

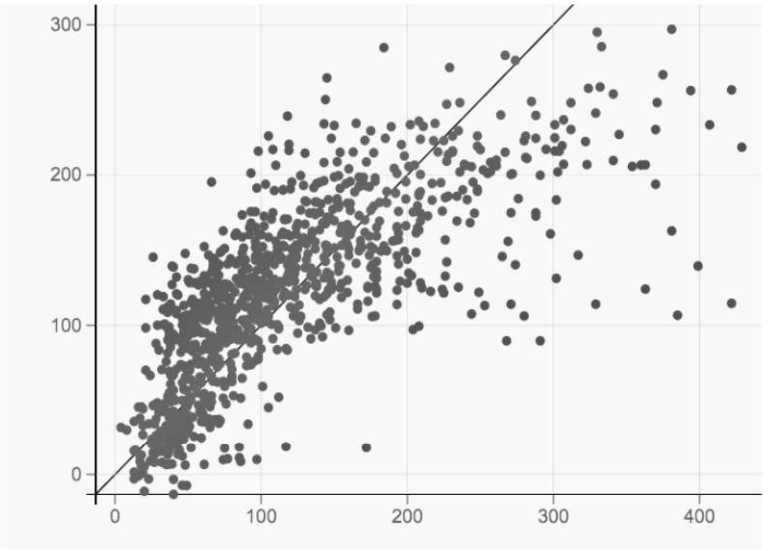


Figure 7.22 — Diagramme de diffusion des prédictions du nouveau modèle linéaire utilisant la nouvelle variable `News_click_seuil`. © Dataiku.

Il est vraisemblable que les autres variables d'importances (`GDP_inhab`, `first_item_prize`...) sont elles aussi non linéaires et qu'un travail similaire nous rapprocherait encore des performances de l'arbre de décision. C'est là toute l'importance d'un bon *feature engineering*.

Si l'on arrive à créer les variables adéquates, un algorithme linéaire peut donner des résultats aussi bons qu'un algorithme non linéaire a priori plus complexe.

Choisir un modèle prédictif linéaire basé sur des variables non linéaires (par rapport à des variables initiales) n'est pas une simple coquetterie académique mais peut se justifier de deux manières. La première raison est que les modèles linéaires, nous l'avons vu, sont extrêmement facile à entraîner. La seconde est liée au fait que les coefficients de ces modèles sont souvent directement interprétables alors que des modèles non paramétriques plus sophistiqués ne possèdent pas toujours une interprétation, métier ou physique, évidente.

1. Ce que *DSS* permet aisément de réaliser au moyen d'un script *Python*.

Copyright © 2015 Dunod.

7.4.4 Sensibilité de l'algorithme KNN au bruit

Le but de cet exemple est de souligner la sensibilité au bruit de l'algorithme des *k* plus proches voisins et d'illustrer par la même occasion l'intérêt d'une technique de la réduction dimensionnelle.

On se place à nouveau dans le contexte d'une classification binaire où l'on cherche à prédire si le revenu généré sera au-dessus ou en dessous de la moyenne. L'algorithme utilisé est KNN avec cinq plus proches voisins. Supposons qu'aux variables prédictives réelles nous ajoutions deux variables fictives de pur bruit, c'est-à-dire ne comportant aucune information relative au revenu généré. La figure 7.23 montre les deux courbes ROC associées aux observations initiales et à celles auxquelles l'on a rajouté deux variables fictives de bruit. On constate que l'AUC = 0,71 de la courbe ROC des observations bruitées est nettement inférieur à l'AUC = 0,82 de la courbe ROC des observations non bruitées.

```
Area under the ROC curve with noisy features: 0.714985
Area under the ROC curve without noisy features: 0.829050
<matplotlib.legend.Legend at 0x10d217850>
```

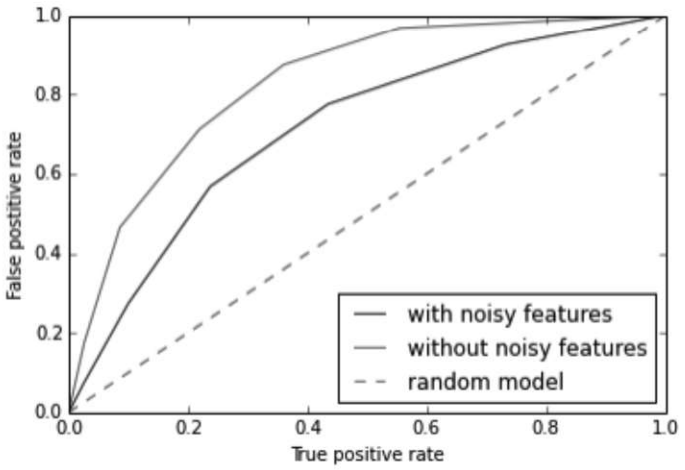


Figure 7.23 — En noir, la courbes ROC d'un algorithme KNN en présence d'observations bruitées. On constate que son AUC est moindre que celle de la courbe claire correspond aux observations non bruitées.

Les deux variables de bruit font perdre 0,1 point d'AUC à l'algorithme KNN ce qui est conséquent. L'interprétation de ce phénomène est simple : le bruit fausse considérablement le calcul des distances entre points représentatifs des observations et donc l'identification des plus proches voisins sur lesquels se base la prédiction.

Dans une situation réaliste où nous ignorerions que deux variables ne contiennent aucune information utile à notre prédiction, il nous faudrait le découvrir. Ne sélectionner que les variables les plus utiles à la prédiction est l'essence même de la **réduction dimensionnelle**. Une méthode bien adaptée à la situation décrite consisterait à rajouter, une à une des variables prédictives, en testant à chaque fois laquelle parmi les variables restantes est la plus corrélée avec le revenu généré et en stoppant ce processus

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

lorsque cette corrélation descend en dessous d'un certain seuil fixé par avance. On parle alors de stratégie de type **forward selection**.

7.4.5 Interprétabilité de deux modèles

L'objectif de ce paragraphe est de montrer comment il possible d'interpréter les résultats d'apprentissage des deux algorithmes envisagés précédemment, l'arbre de décision et la régression linéaire, pour leur conférer une valeur ajoutée commerciale.

Commençons par l'arbre de décision. Lorsque sa profondeur est modeste (< 5), un tel arbre se laisse parfois se traduire en règles métiers intuitives. Il suffit pour cela de « descendre » dans l'arbre à partir du nœud racine et de relever les règles de séparation que l'algorithme a apprises. La figure 7.24 illustre un arbre de profondeur 3 (il est tiré de la série des arbres de décision générés pour illustrer plus haut le problème du surapprentissage).

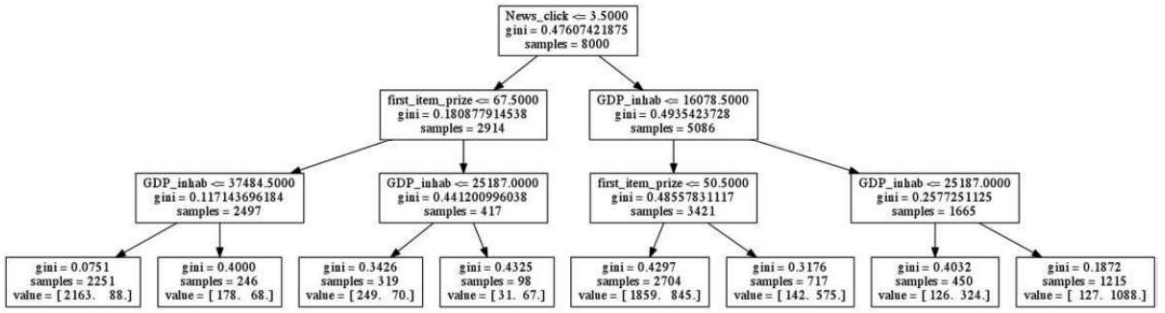


Figure 7.24 — Un arbre de profondeur 3 qui se laisse interpréter au moyen de règles métiers.

On constate tout d'abord que les variables retenues par l'algorithme pour séparer les données, `News_click`, `first_item_prize` et `GDP_inhab`, sont précisément celles qui avaient été identifiées dans la section sur le *feature engineering*, voir figure 7.21. Par ailleurs la variable `News_click` dont nous avions identifié à cette occasion le caractère non linéaire est précisément celle qu'a retenue l'arbre de décision pour son premier critère de décision. Le seuil de décision égal à 3,5 retenu par l'algorithme est lui aussi cohérent avec la valeur 4 que nous avions identifiée.

En suivant le chemin le plus à gauche de l'arbre on note la succession de critères de décisions suivante :

```
''News_click < 3.5" → ''first_item_prize < 65.5"
    → ''GDP_inhab < 37484.5"
```

Dans la feuille correspondante on trouve 2 163 clients ayant généré un revenu inférieur à la moyenne et 88 clients ayant généré un revenu supérieur à la moyenne. On peut donc en déduire, en termes métiers, que les clients qui suivent peu les campagnes de publicité, qui achètent un premier article à bas prix et qui ne viennent pas de pays extrêmement riches sont peu susceptibles de générer de gros revenus, ce qui paraît plutôt cohérent.

Copyright © 2015 Dunod.

Venons-en maintenant à la régression linéaire. Celle-ci est effectuée après une normalisation initiale des données, ceci pour pouvoir légitimement comparer entre eux les coefficients de la régression.

```
Revenue = constante + 0.002 * GDP_inhab
   + 1.1 * first_item_prize + 0.09 * age + 0.09 * pages
   + 13.6 * News_click - 0.00000002 * population
   - 18.4 * ReBuy_value_False - 8.9 * gender_value_Fem
```

On peut commencer par regarder les signes des différents coefficients. S'ils sont positifs cela traduit le fait qu'une augmentation de la variable associée se traduit par une augmentation du revenu et inversement. On voit par exemple qu'un client qui n'a pas racheté une deuxième fois son premier article (`ReBuy_value_False`) génèrera moins de revenu, tandis qu'un niveau de prix élevé pour ce premier article (`first_item_prize`) est un bon indicateur d'un client rémunérateur. Rien de surprenant à cela, mais il est toujours bon de vérifier la cohérence de l'algorithme.

Le fait que la variable `News_click` soit associée au plus grand coefficient (13.6) est, là encore, en pleine cohérence avec notre identification précédente de cette variable comme étant la plus importante. À l'inverse, la population du pays d'origine de la personne ne joue quasiment aucun rôle.

Lorsque l'algorithme s'y prête, il est toujours intéressant d'essayer d'en interpréter les résultats grâce à ce type d'analyse. Identifier les variables les plus significatives est crucial pour entreprendre les bonnes actions en vue d'améliorer les objectifs. L'analyse qui précède incite, par exemple, à focaliser l'attention sur l'amélioration des campagnes publicitaires qui favorisent la curiosité des clients et en définitive l'achat.

7.4.6 Bénéfices de l'approche ensembliste

Une approche ensembliste donne presque toujours de meilleurs résultats qu'un simple estimateur. Elle évite souvent le surapprentissage et introduit une diversité intéressante pour la prédiction. En guise de métaphore on pourrait dire qu'un collège d'experts donnera toujours une meilleure prédiction qu'un seul de ces experts. Pour illustrer cela, nous avons lancé un algorithme de forêt aléatoire sur les données utilisées dans la partie illustrant le surapprentissage.

Pour une forêt de 50 arbres, on obtient un AUC de 0,925, ce qui est largement supérieur au meilleur arbre de décision qui avait obtenu un AUC de 0,848.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

En résumé

La *machine learning* est un sujet vaste en permanente évolution. Les algorithmes qu'il met en œuvre ont des sources d'inspiration variées qui vont de la théorie des probabilités aux intuitions géométriques en passant par des approches heuristiques comme celle des forêts aléatoires.

La difficulté du sujet, mais aussi son intérêt, tient au fait qu'il n'existe aucune panacée qui permettrait de résoudre tous les problèmes, c'est le *No Free Lunch*. Ceci est d'autant plus vrai dans un contexte Big Data ou le fléau de la dimension est une préoccupation permanente.

Le data scientist devra faire son marché dans un éventail d'algorithmes dont chacun a des spécificités qu'il doit connaître pour être à même d'optimiser les modèles qu'il construit. La Rolls-Royce des algorithmes du *machine learning* est à l'heure actuelle l'algorithme des forêts aléatoires qui apporte une somme d'informations particulièrement utiles en phase exploratoire.

Copyright © 2015 Dunod.

8

La visualisation des données

Objectif

La visualisation des données, aussi appelée « *dataviz* », concerne aussi bien le data scientist que l'expert métier ou le décideur. Au premier, elle dévoile ce que les statistiques ne révèlent pas. Aux experts et aux décideurs, elle fournit le support de l'intuition et la contextualisation indispensable à une prise de décision.

La première section de ce chapitre explique pourquoi les statistiques à elles seules ne suffisent pas pour juger de l'adéquation d'un modèle prédictif. Une deuxième section explique comment mettre en évidence graphiquement des corrélations entre plusieurs variables, une tâche au cœur du *machine learning* puisque cette discipline cherche précisément à tirer profit de ces corrélations pour construire des prédictions. Enfin, une dernière section aborde la problématique de la représentation de données complexes. Les principes d'encodage visuels statiques et dynamiques sont traités séparément. Des critères seront proposés pour juger de la qualité d'une représentation graphique.

8.1 POURQUOI VISUALISER L'INFORMATION ?

8.1.1 Ce que les statistiques ne disent pas

Une grande partie des algorithmes de *machine learning* présentés au chapitre précédent repose sur l'utilisation des modèles statistiques. Un algorithme de régression linéaire fournit par exemple une estimation statistique $y_{\text{estimé}}$ d'une variable cible en fonction

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

des valeurs observées **x** des variables prédictives d'une nouvelle observation. Pour autant que l'apprentissage ait été mené sur un échantillon significatif, cette prédiction sera correcte *en moyenne*. Explicitement, cela signifie que, sur un échantillon d'observations caractérisées par les mêmes valeurs **x**, les valeurs effectivement observées $y^{(1)},..., y^{(n)}$ seront dispersées à proximité de la prédiction $y_{estimé} = f(\mathbf{x})$. Une vérification sommaire de l'adéquation d'un tel modèle peut se faire en utilisant certains tests statistiques élémentaires qui comparent le degré de dispersion des valeurs observées $y^{(1)},..., y^{(n)}$ à la variabilité intrinsèque du modèle[1]. Une trop forte dispersion est par exemple le signe d'un modèle peu prédictif comme l'illustre la figure 8.1.

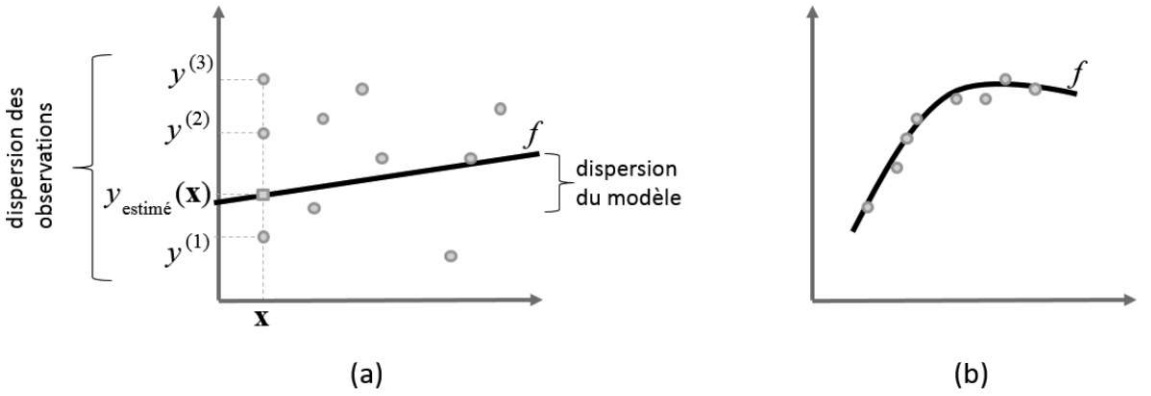


Figure 8.1 — (a) une dispersion relative importante implique un modèle peu prédictif. (b) une dispersion faible correspond à un modèle prédictif.

Cependant, la vérification de l'adéquation d'un modèle à un jeu de données ne se résume pas toujours à l'évaluation de quelques statistiques élémentaires comme la moyenne ou la variance des prédictions. Comment vérifier par exemple la validité de l'hypothèse d'une liaison linéaire entre les y et les **x** ? Ou comment détecter des valeurs aberrantes parmi les observations ? Pour ce genre de tâche l'outil le plus performant n'est pas la statistique mais le système visuel humain. C'est l'un des objectifs de la visualisation que d'en tirer le meilleur parti.

Illustrons notre propos au moyen d'un exemple célèbre : celui du *quartet d'Anscombe*[2]. Il s'agit de quatre jeux de données fictifs construits en 1973 par le statisticien Francis Anscombe dans le but de démontrer l'importance de la visualisation des données *avant* même d'entreprendre une analyse statistique. Chaque jeu de données du quartet d'Anscombe contient le même nombre (onze) de points $(x^{(i)}, y^{(i)})$. Les moyennes des $y^{(i)}$, la moyenne des $x^{(i)}$ ainsi que leurs dispersions respectives sont toutes parfaitement identiques entre les quatre jeux, de même que trois autres statistiques comme le montre le tableau 8.1.

1. Cette variabilité est donnée en l'occurrence par la pente de la droite de régression. Un des tests possibles est le test dit du *t* de *Student*.
2. *The Visual Display of Quantitative Information*, Edward Tufte.

Copyright © 2015 Dunod.

Tableau 8.1 — Les caractéristiques communes des quatre jeux de données du quartet d'Anscombe

Statistique	Valeur
Moyenne des $x^{(i)}$	9,0
Variance[a] des $x^{(i)}$	10,0
Moyenne des $y^{(i)}$	7,5
Variance des $y^{(i)}$	3,75
Corrélation[b] entre les $x^{(i)}$ et les $y^{(i)}$	0,816
Coefficient de la régression linéaire $y = f(x)$	$y = 3 + 0{,}5x$
Somme des carrées des erreurs de prédiction	110,0

a. La *variance* mesure la dispersion d'un ensemble de valeur autour de la moyenne. Elle est égale à la somme des carrés des écarts à la moyenne.

b. La *corrélation* mesure le degré de liaison linéaire entre deux collections de nombres. Elle est comprise entre –1 et +1. Lorsqu'elle vaut +1 ou –1, les deux collections sont parfaitement liées linéairement, de manière croissante ou décroissante. Lorsqu'elle vaut zéro il n'y a aucune liaison linéaire.

Sans visualisation, l'égalité des sept statistiques pourrait inciter à conclure hâtivement que ces quatre jeux de données sont très similaires, voir interchangeables. Ce qu'une visualisation vient pourtant démentir comme l'illustre la figure 8.2.

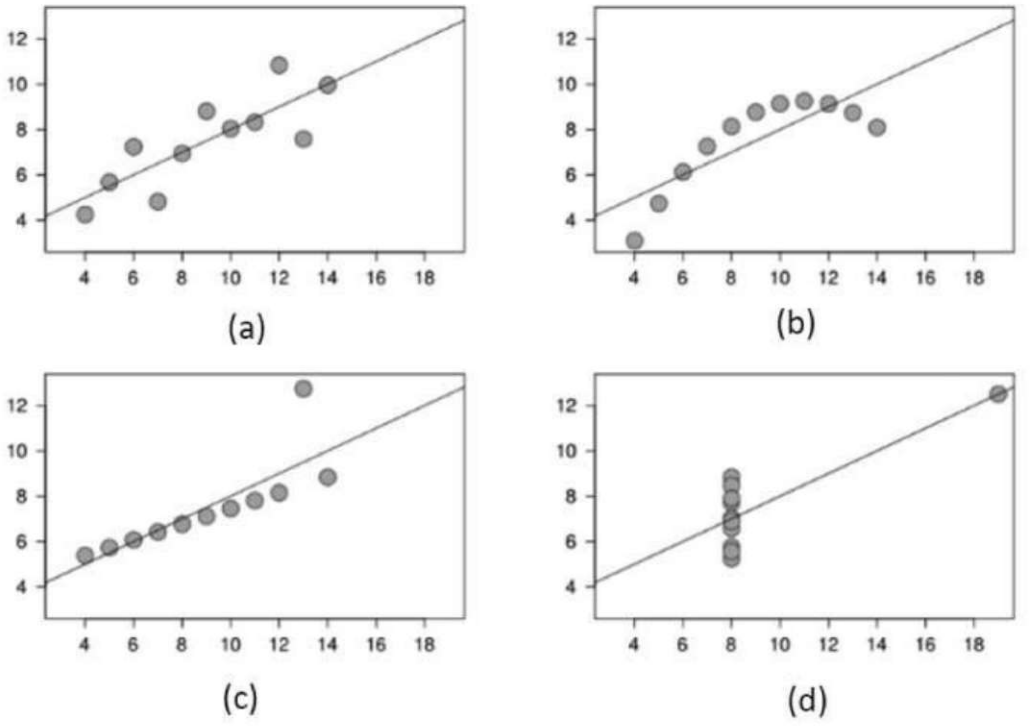


Figure 8.2 — La visualisation des quatre jeux de données du quartet d'Anscombe.

L'ensemble (a) apparaît essentiellement dépourvu de toute structure. L'ensemble (b) bien que non corrélé linéairement est parfaitement rangé sur une courbe non

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

linéaire, une parabole. L'ensemble (c) présente une corrélation parfaite, et devrait donc afficher une corrélation très proche de un, mais une seule valeur aberrante ramène ce coefficient à la valeur 0,816. L'ensemble (d) est un ensemble dans lequel il n'existerait aucune corrélation entre les valeurs $x^{(i)}$ et $y^{(i)}$ si l'on omettait l'unique valeur aberrante en haut à droite.

Notons enfin que dans aucun des quatre cas la régression linéaire n'aurait été un bon choix de modèle, à l'exception du cas (c) dont il aurait fallu exclure la valeur aberrante.

8.1.2 Les objectifs de la visualisation

L'exemple précédent montre tout l'intérêt de la visualisation pour détecter les valeurs aberrantes et vérifier certaines hypothèses avant de choisir un modèle. Mais il ne s'agit pas, loin s'en faut, des seules motivations pour visualiser des données. En réalité la visualisation intervient à toutes les étapes du travail d'un data scientist :

- **Préparation des données** : dans cette phase la visualisation pourra servir à détecter des valeurs aberrantes ou des valeurs manquantes.
- **Choix d'un modèle** : à ce stade, on pourra valider certaines hypothèses comme la linéarité de certaines relations par exemple.
- **Apprentissage** : un data scientist peut guider l'apprentissage en supprimant certaines variables peu prédictives ou en construisant de nouvelles variables plus prédictives (voir section 7.4.3). La visualisation pourra ainsi contribuer à identifier quels sont les couples de variables les plus corrélées, nous y reviendrons dans la section 8.2.
- **Optimisation d'un modèle** : ici il s'agit par exemple de vérifier que le modèle n'a pas subi de surapprentissage. Des courbes qui montrent l'évolution parallèle des erreurs de prédictions sur les ensembles d'apprentissage et de tests en fonction des paramètres ajustables du modèle permettront de trouver les paramètres optimaux, voir la figure 7.18. Les courbes ROC qui permettent d'optimiser un modèle de classification sont un autre exemple, voir la figure 7.17.
- **Prédiction** : la visualisation permet de partager l'information tout en la rendant intelligible à des experts métiers sans connaissances statistiques particulières.

Pour les experts métiers et pour les décideurs la visualisation fournit le **support de l'intuition** et la **contextualisation** essentiels à toute prise de décision.

Le terme **dataviz** recouvre tous ces aspects de la représentation graphique. Dans tous les cas, la visualisation doit viser les objectifs suivants :

- **Intelligibilité** : les données doivent être interprétables sans effort excessif et sans ambiguïté pour l'audience à laquelle elles sont destinées. La visualisation doit être une aide au jugement humain.
- **Pertinence** : la visualisation doit répondre à un objectif métier précis défini par avance.

Copyright © 2015 Dunod.

- **Cohérence** : la visualisation ne doit pas représenter, ou même seulement suggérer, des informations erronées ni en omettre d'essentielles.
- **Stimulation** : la visualisation doit stimuler la réflexion et inciter l'utilisateur à découvrir toutes les facettes d'un jeu de données.

Terminons cette section par une remarque de bon sens. Si la visualisation s'avère souvent plus efficace que les statistiques lorsqu'il s'agit de se faire une idée de la structure globale d'un jeu de données, elle ne saurait en revanche se substituer au *machine learning*. Imaginons à titre d'exemple qu'une première analyse visuelle ne révèle aucune corrélation entre deux variables x_1 et y. Il serait pourtant erroné d'en conclure que la variable x_1 est inutile pour effectuer des prédictions car les corrélations entre x_1, une autre variable x_2 et y pourraient rendre x_1 utile. C'est ce type de corrélations complexes non intuitives, qu'un modèle de *machine learning* est capable d'apprendre et c'est ce qui fait son utilité.

La visualisation et les statistiques sont donc des outils complémentaires qui doivent entretenir un dialogue permanent. La visualisation stimule l'imagination et forge les intuitions que les modèles statistiques du ML permettent de quantifier.

8.2 QUELS GRAPHES POUR QUELS USAGES ?

Considérant la grande diversité des types de données susceptibles d'intervenir dans un problème de ML, il est difficile de proposer une méthode infaillible qui associerait un certain type de représentation à chaque combinaison de variables prédictives et de variable cible. Dans les cas simples, il existe cependant quelques bonnes pratiques identifiées de longue date par les statisticiens ainsi que des usages courants qui, sans être des diktats, constituent un bon point de départ pour construire des graphes intelligibles.

Les quatre sections qui suivent suggèrent quelques bonnes pratiques tirées de la statistique descriptive bidimensionnelle. L'objectif est de proposer une représentation adaptée à la détection de corrélations entre deux variables x et y sur un jeu de N observations $(x^{(i)}, y^{(i)})$. Il s'agit d'une première étape dans la sélection d'un modèle de ML. Les représentations seront différentes selon que l'une ou l'autre des deux variables est soit qualitative soit quantitative (voir section 7.1). Nous envisagerons successivement les quatre combinaisons possibles.

Une dernière section décrit un graphe utile, les courbes ROC pour l'évaluation d'un modèle de classification dont nous avons déjà vu un exemple dans la section 7.4.

Variable prédictive et variable cible qualitatives

Considérons pour commencer le cas où la variable prédictive et la variable cible sont toutes deux qualitatives. Les exemples de ce type abondent : la relation entre l'achat d'un produit et le sexe de l'acheteur, la relation en l'exercice d'une certaine profession

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

et un cursus d'études, etc. Le graphe adapté à ce cas est le **diagramme en barres** (*mosaic plot*).

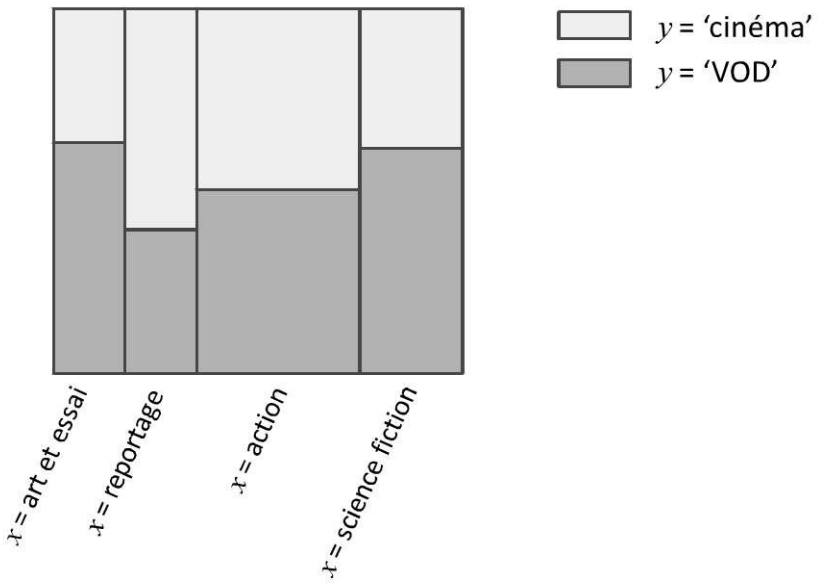


Figure 8.3 — Un exemple de diagramme en barres. On veut représenter le mode de visionnement d'un film par catégorie. La variable prédictive *x* décrit quatre types de film, la variable cible *y* les deux modes de visionnement.

La variable prédictive *x* est associée à l'axe horizontal qui est subdivisé en autant de segments qu'il existe de catégories pour *x*. La largeur de chaque colone est proportionnelle à la fréquence relative des observations dans la catégorie correspondante de *x*. Chaque colonne, associée à une catégorie de la variable *x*, est à son tour subdivisée en autant de rectangles qu'il y a de catégories pour la variable cible y. La hauteur d'un rectangle est proportionnelle à la fréquence relative des observations dans la catégorie de y correspondante pour la valeur de *x* associée à la colonne en question.

On obtient ainsi une mosaïque où chaque rectangle possède une surface proportionnelle au nombre d'observations caractérisées par un couple de catégories (*x*,y) comme l'illustre la figure 8.3.

Remarquons que dans un diagramme en barres les axes ne sont pas symétriques. L'idée en effet est de présenter y comme une réponse à une variable *x*. Si la variable cible y est peu corrélée avec la variable prédictive *x*, les segments horizontaux se situeront environ à la même hauteur. Si, à l'inverse, y et *x* sont fortement corrélés les barres de séparations horizontales seront situées à des hauteurs significativement différentes.

Pour améliorer l'intelligibilité du graphe on utilise parfois un code à deux couleurs pour représenter le niveau de signification de la corrélation, positive ou négative, entre les variables y et *x*, la couleur étant d'autant plus saturée que l'écart est important. On peut aussi se contenter d'encoder les différentes catégories de y à l'aide de couleurs.

Copyright © 2015 Dunod.

Variable prédictive quantitative, variable cible qualitative

Pour représenter la dépendance d'une variable cible y qualitative en fonction d'une variable x quantitative l'idée consiste à représenter la répartition des valeurs de x pour chaque catégorie de y. On utilise pour cela des **diagrammes en boîtes** aussi appelées **boîtes à moustaches** (*box plot*) dont la figure 8.4 donne un exemple. Remarquons au passage que, contrairement à l'usage habituel, la variable prédictive x correspond ici à l'axe vertical et la variable cible y à l'axe horizontal !

Pour N observations $x^{(1)},\ldots, x^{(N)}$ d'une variable x, un diagramme-boîte représente la position des quartiles : le minimum, le quantile[1] à 25 %, la médiane, le quantile à 75 % et enfin le maximum. Les valeurs aberrantes sont quant à elles représentées séparément, comme des points isolés. Chaque boîte fournit ainsi une représentation schématique de la répartition des valeurs $x^{(i)}$ autour de la valeur centrale donnée par la médiane.

Une forte corrélation entre la variable prédictive x et la variable cible y se traduira par une importante disparité des positions et des tailles des boîtes associées à différentes valeurs de y.

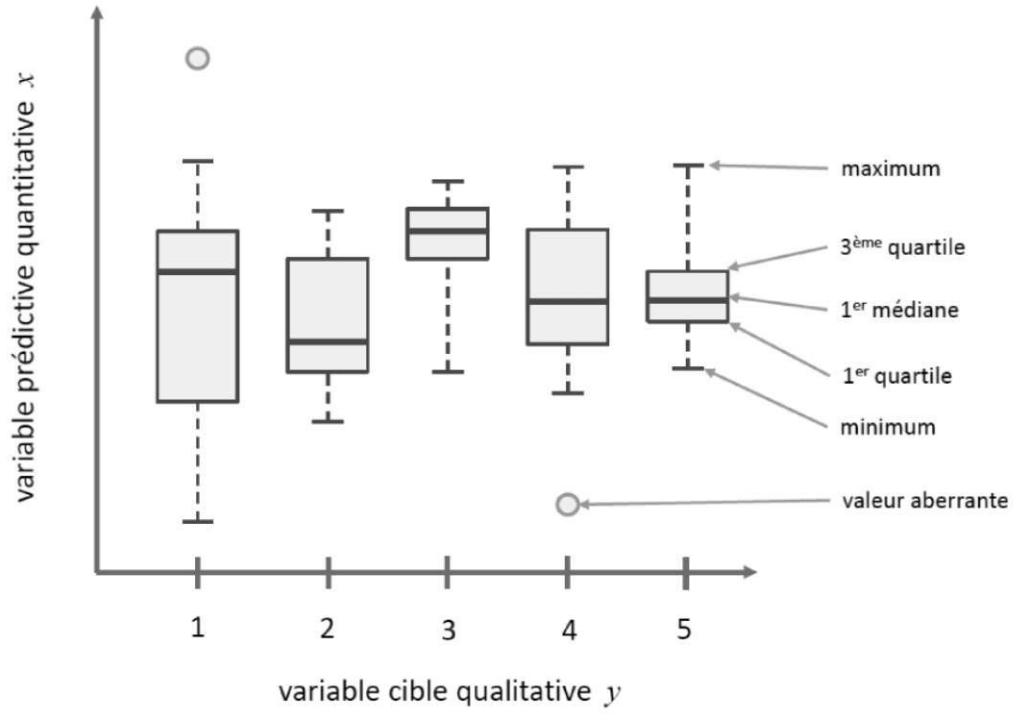


Figure 8.4 — Un exemple de diagrammes en boîtes.
Une disparité importante entre les boîtes indique une forte corrélation entre les variables x et y.

1. Le quantile à p % d'un échantillon de valeurs correspond à la valeur en dessous de laquelle se trouvent p % des observations. Les quartiles correspondent à p = 25, 50, 75. La médiane est un nom donné au quartile p = 50.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

Variable prédictive qualitative, variable cible quantitative

Dans la situation inverse de celle décrite dans la section précédente, c'est-à-dire lorsque la cible y est quantitative et la variable prédictive x est qualitative, on peut également utiliser les diagrammes en barres, peut-être même sont-ils plus naturels dans ce cas puisque l'axe vertical représente alors la cible y.

Une autre possibilité est d'utiliser des **courbes de densité**, similaires aux histogrammes, comme le montre la figure 8.5.

Pour chaque catégorie de la variable prédictive x on construit un histogramme lissé, que l'on appelle courbe de densité, des valeurs prises par la variable cible y. À un mince intervalle de valeurs de y correspond une surface sous la courbe proportionnelle au nombre d'observations pour lesquels y se situe dans l'intervalle et x est dans la catégorie spécifiée.

Une forte disparité dans le profil des courbes de densité indique une forte corrélation entre les variables x et y.

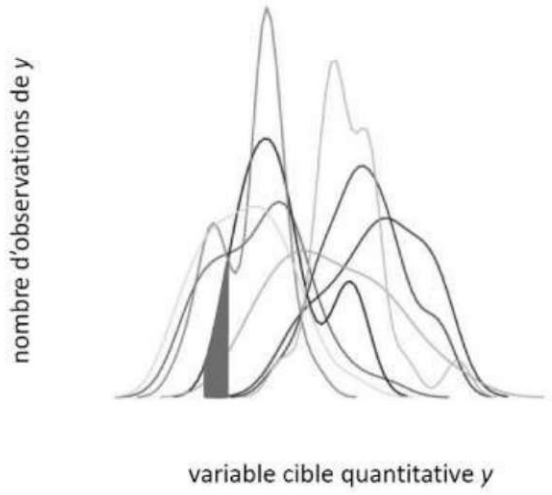


Figure 8.5 — Un exemple de courbes de densité. À chaque catégorie de la variable prédictive x est associée une courbe de densité des valeurs de la variable cible y. La surface sous une courbe est proportionnelle au nombre d'observations dans l'intervalle correspondant de la variable cible y.

Variable prédictive et variable cible quantitatives

Enfin, terminons par le cas le plus simple, celui où les deux variables sont quantitatives. Chaque couple d'observation (x,y) est simplement représenté par un point sur un diagramme qu'on appelle **diagramme de dispersion** qui permet de détecter des corrélations de différentes natures entre les deux variables comme le montre la figure 8.6.

Copyright © 2015 Dunod.

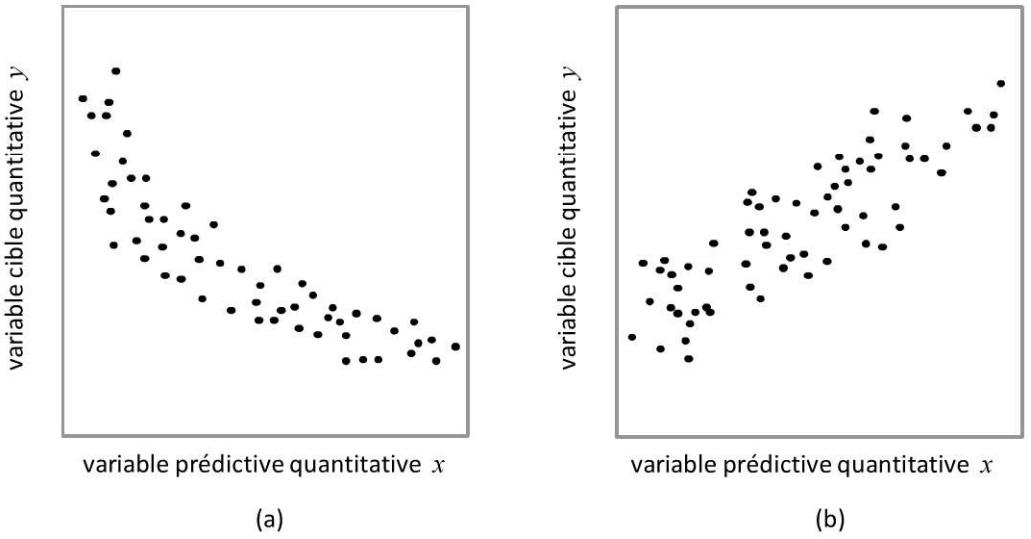


Figure 8.6 — Deux exemples de diagramme de dispersion.
Le diagramme (a) indique une dépendance non linéaire entre x et y.
Le diagramme (b) indique une dépendance linéaire entre x et y.

Courbe ROC et évaluation des modèles de classification

Le but du diagramme décrit dans cette section est de représenter la performance d'un algorithme de classification binaire. Il est donc d'une autre nature que les quatre diagrammes que nous venons de présenter dont l'objectif était de détecter une éventuelle corrélation entre deux variables.

Pour fixer les idées, imaginons que nous souhaitions mettre en place un dispositif de dépistage d'une maladie grave en utilisant un modèle de classification binaire comme la régression logistique, voir section 7.3.4. Un certain nombre de variables prédictives $\mathbf{x} = (x_1, x_2, x_3,...)$, comme le rythme cardiaque, l'âge ou les pays visités récemment, seront utilisées pour prédire la probabilité $P_{\text{malade}}(\mathbf{x})$ qu'un individu soit atteint de la maladie. C'est en fonction de cette probabilité, comprise entre 0 et 1, qu'il conviendra de prendre la décision d'affecter l'individu $\mathbf{x}$ au groupe des individus sain S ou à celui des porteurs de la maladie M. Dans une approche naïve on pourrait convenir d'affecter $\mathbf{x}$ à S si $P_{\text{malade}}(\mathbf{x}) < 0.5$ et à M si $P_{\text{malade}}(\mathbf{x}) \geqslant 0.5$.

Pourtant, à bien y réfléchir, un tel choix n'est guère acceptable car les risques encourus ne sont nullement symétriques vis-à-vis des erreurs commises selon qu'elles concernent le groupe S ou M. Il est beaucoup plus grave en effet d'affecter par erreur un individu malade au groupe sain que l'inverse. Dans le premier cas une vie est un jeu, dans le second l'individu en sera quitte pour une inquiétude de durée limitée.

Pour limiter le risque de ne pas détecter la maladie lorsqu'elle est présente il faut abaisser le seuil de $s = 0{,}5$ à une valeur plus faible, par exemple $s = 0{,}1$, et affecter l'individu $\mathbf{x}$ à M sitôt que $P_{\text{malade}}(\mathbf{x}) \geqslant 0{,}1$, quitte à commettre davantage d'erreurs sur des individus sains. En effet, abaisser le seuil s, conduit à attribuer $\mathbf{x}$ plus souvent au groupe M, que cette décision soit correcte ou fausse.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

Dans un tel contexte on utilise généralement le vocabulaire suivant pour caractériser les décisions possibles (l'adjectif positif se réfère à un signal positif de détection de la maladie) :

- **vrai positif** : on a affecté un individu malade au groupe malade.
- **faux positif** : on a affecté un individu sain au groupe malade (pas grave).
- **vrai négatif** : on a affecté un individu sain au groupe sain.
- **faux négatif** : on a affecté un individu malade au groupe sain (décision lourde de conséquences qu'il s'agit de minimiser !).

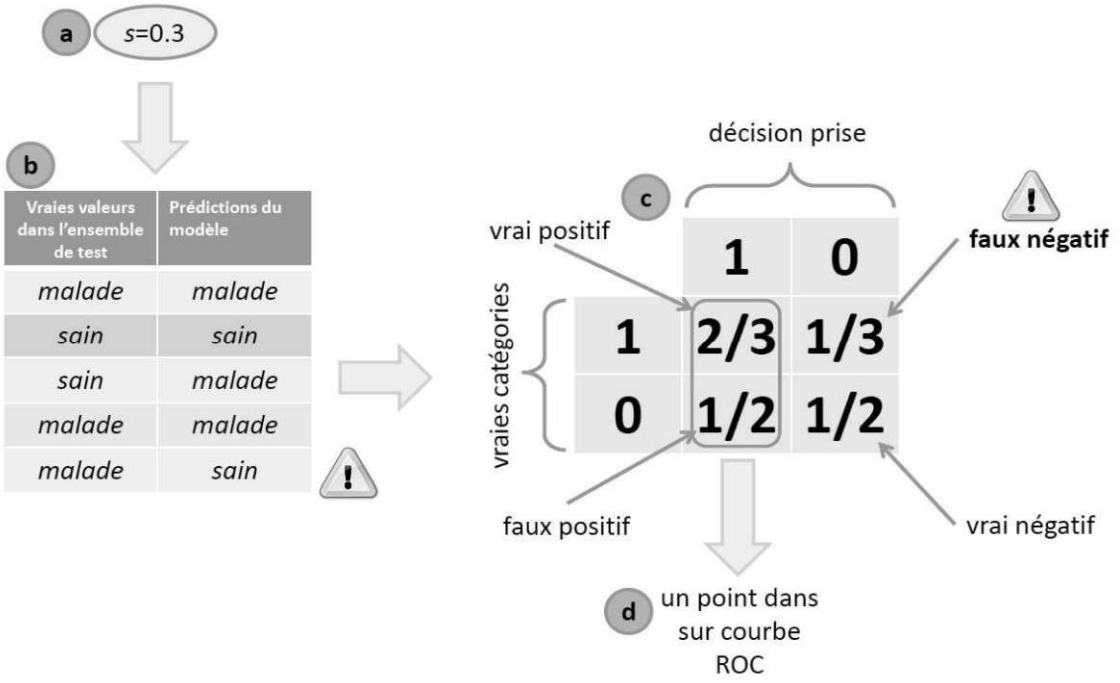


Figure 8.7 — Les étapes de la construction d'un point sur la courbe ROC : (a) choix d'un seuil *s* au-delà duquel on affecte l'individu à la catégorie malade, (b) utilisation de ce seuil pour faire des prédictions, (c) construction de la matrice de confusion associée, la valeur 1 correspond à « malade » et la valeur 0 correspond à « sain ». Les éléments de matrices représentent les fréquences des décisions correspondantes. Les valeurs de la première colonne permettent de comparer les probabilités des « vrai positif » aux « faux positifs », (d) définition d'un point sur la courbe ROC, voir figure 8.8.

Pour représenter les prédictions d'un classificateur binaire on utilise un tableau à quatre éléments, que l'on appelle la **matrice de confusion**, et qui représente les probabilités des prédictions d'un modèle de ML à partir d'un échantillon de test. La figure 8.7 en donne un exemple.

Remarquons que les sommes par ligne valent toutes 1, si bien que minimiser la probabilité des faux négatifs, qui est notre objectif, revient à maximiser la probabilité des vrais positifs. Pour la même raison, toute l'information d'une matrice de confusion est contenue dans les deux éléments de sa première colonne.

Copyright © 2015 Dunod.

Pour trouver le seuil s qui réalise un compromis acceptable on reporte sur un graphe la proportion de vrais positifs en fonction de la proportion des faux positifs. Chaque point correspond alors à une matrice de confusion. C'est la **courbe ROC**[1] ou **courbe de sensibilité**. La figure 8.8 donne un exemple.

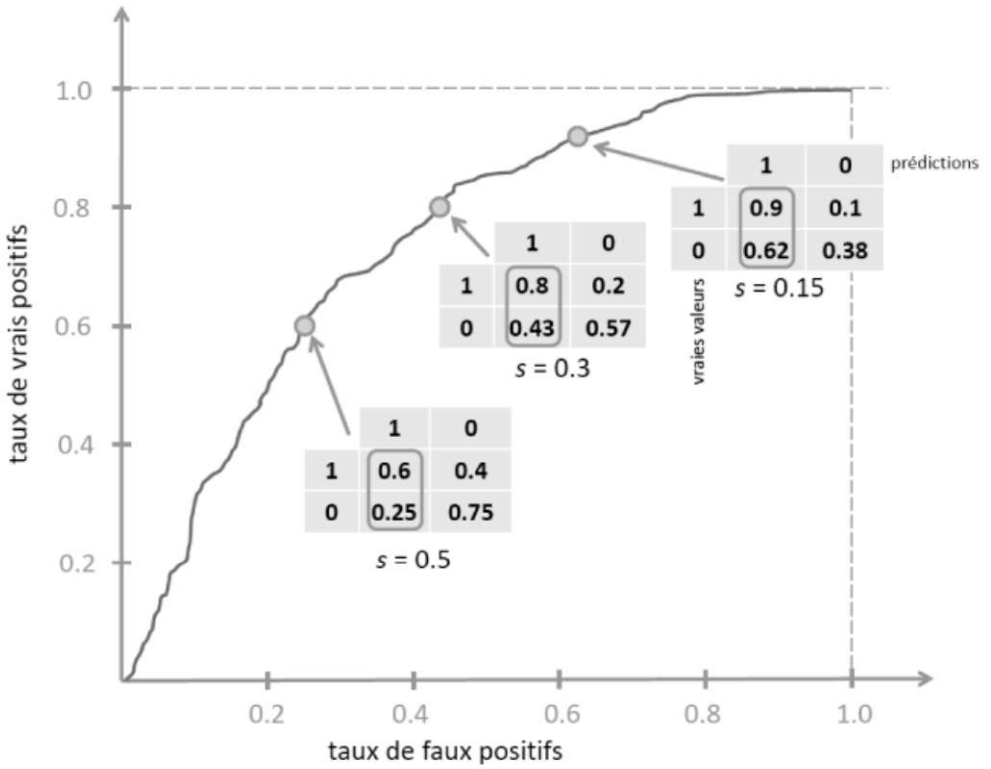


Figure 8.8 — Un exemple de courbe ROC qui représente le taux de vrais positifs en fonction du taux de faux positifs pour différentes valeurs du seuil s. Les matrices de confusions associées à trois points sont également représentées pour différentes valeurs du seuil s.

Pour un test de dépistage comme celui que nous envisageons, il est donc préférable de se trouver dans la partie supérieure droite du diagramme où les faux négatifs sont peu nombreux.

Un algorithme idéal ne commettrait aucune erreur, son taux de vrais positifs vaudrait 1 alors que celui des faux positifs vaudrait 0. La « courbe » ROC associée formerait un angle droit. Plus une courbe ROC s'approche de cet idéal meilleur sera l'algorithme. Ainsi, dans la figure 8.9, le prédicteur associé à la courbe ROC n° 2 est meilleur que celui associé à la courbe ROC n° 1. Une mesure possible de la qualité d'un prédicteur binaire est l'aire située sous la courbe ROC qui lui est associée. Un score de qualité d'un algorithme binaire s'obtient en divisant l'aire sous la courbe ROC par l'aire du rectangle idéal. On désigne généralement ce score par le sigle **AUC** pour « *Area Under the Curve* ».

1. De l'anglais *Receiver Operating Characteristic*.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

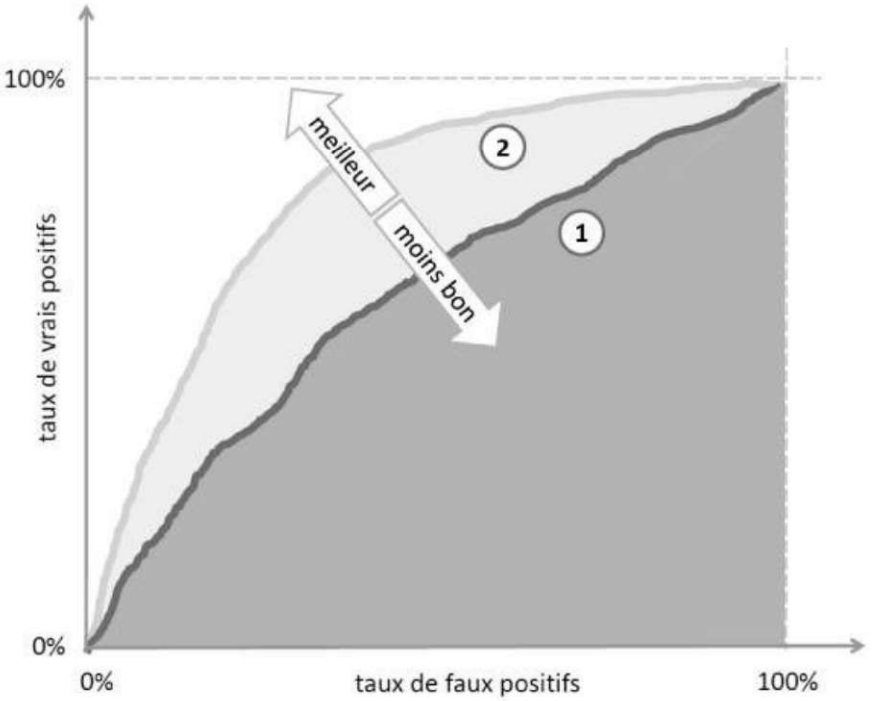


Figure 8.9 — Deux courbes ROC associées à deux prédicteurs binaires différents. La courbe n° 2 est associée à un meilleur prédicteur que la courbe n° 1 car elle est plus proche de l'angle droit idéal.

8.3 REPRÉSENTATION DE DONNÉES COMPLEXES

8.3.1 Principes d'encodage visuel

Les deux dimensions d'un écran ou d'une feuille de papier permettent de représenter deux variables x_1 et x_2 d'une observation **x** en les utilisant comme coordonnées d'un point. Pour aller au-delà, et représenter un plus grand nombre de variables, il faut par conséquent recourir à d'autres attributs visuels que la position d'un point sur un plan. Si l'on s'en tient aux diagrammes statiques, les attributs visuels utilisables en pratique ont été identifiés et qualifiés par le cartographe Jacques Bertin[1]. Ils sont répertoriés sur la figure 8.10 :

- La **position** d'un point ou d'un symbole.
- La **taille** d'un point représentatif ou de tout autre symbole.
- La **saturation** d'une couleur ou des niveaux de gris d'un symbole.
- Le **grain** ou la texture d'une surface.
- La **couleur** d'une surface
- L'**orientation** d'un symbole représentatif d'un point
- La **forme** d'un symbole représentatif d'un point

1. *La graphique et le traitement graphique de l'information*, Paris, Flammarion, 1977, 273 p.

Copyright © 2015 Dunod.

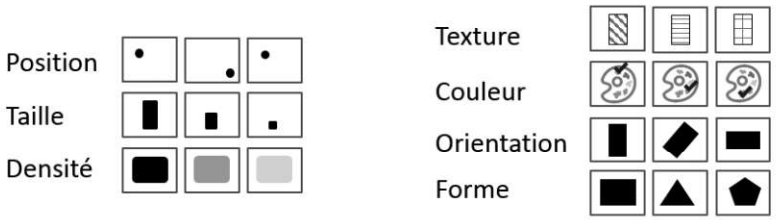


Figure 8.10 — Les sept attributs visuels de Bertin

Les caractéristiques perceptuelles de ces septs attributs s'avèrent très différentes dans la mesure où ils ne sont pas tous perçus par l'œil humain avec la même précision[1]. Des études de ces mécanismes perceptifs ont même permis de les ranger par ordre décroissant de précision perceptuelle (figure 8.11).

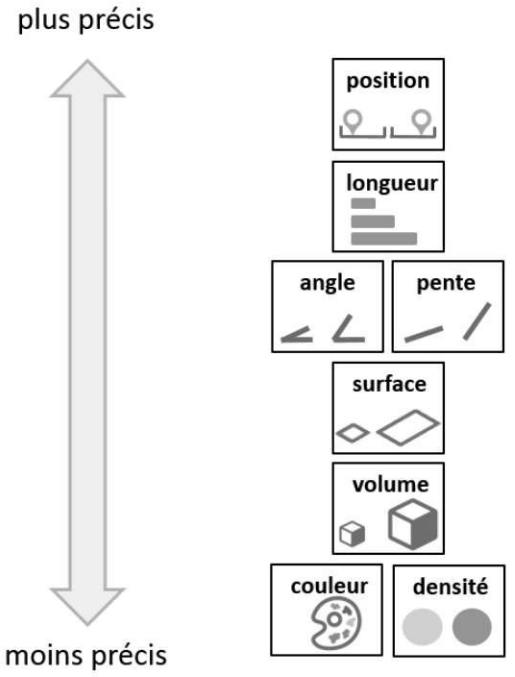


Figure 8.11 — Classement des sept attributs visuels de Bertin selon la précision avec laquelle ils sont perçus.

La position et la taille d'un symbole, perçues de manière très précises, doivent par conséquent être réservées à la représentation des variables les plus importantes. La couleur et la densité en revanche, ne devraient être utilisées que pour des variables moins significatives. Par ailleurs, certains attributs comme la texture ou la couleur se prêtent mal à la représentation des variables quantitatives ou ordinales, le tableau 8.2 indique les meilleurs usages.

1. Mackinlay, APT (A Presentation Tool) 1986. http://www2.parc.com/istl/groups/uir/publications/items/UIR-1986-02-Mackinlay-TOG-Automating.pdf

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

Tableau 8.2 — Les bonnes pratiques pour la représentation des différents types d'attributs.

Attributs visuels	Types de variables		
Position	quantitative	ordinale	nominale
Taille	quantitative	ordinale	nominale
Densité		ordinale	nominale
Texture			nominale
Couleur			nominale
Orientation			nominale
Forme			nominale

Enfin, pour évaluer la qualité d'une visualisation, voici une check-list de questions à poser :

1. Le graphique est-il esthétiquement plaisant ?
2. Est-il immédiatement intelligible ou nécessite-t-il un effort cognitif important ?
3. Apporte-t-il une information qui n'était pas décelable dans la représentation originale ?
4. Le diagramme se prête-il aisément aux comparaisons ?
5. Le diagramme est-il cohérent dans le sens où l'information qu'il véhicule ne distord par la réalité ?
6. Les principes d'encodage visuels du tableau de la figure 8.11 et du tableau 8.2 ont-ils été respectés ?

8.3.2 Principes de visualisation interactive

L'un des objectifs de la visualisation est la stimulation de l'imagination et de l'envie d'explorer toutes les facettes d'un jeu de données. Pour y parvenir, le meilleur moyen est de rendre cette exploration interactive, un aspect que nous avons ignoré jusque-là.

Les outils de visualisation interactive sont actuellement en plein essor et ont été largement démocratisés par l'avènement de logiciels en mode SAAS comme *ClikView* et *Tableau*. Objet de recherches actives, les principes perceptifs de la visualisation interactive ne sont pas encore aussi clairement identifiés que les principes d'encodage visuel statique. Le but de cette dernière section est de décrire succinctement les techniques qu'utilisent aujourd'hui les outils les plus flexibles.

Copyright © 2015 Dunod.

Selon l'article d'*acmqueue* Interactive Dynamics for Visual Analytics[1], il est possible de distinguer trois types de tâches que doivent assumer les bons outils de visualisation interactive.

La spécification d'une visualisation

Les outils de visualisation interactive peuvent combiner deux stratégies opposées pour permettre à un analyste de sélectionner les données qu'il souhaite visualiser et de choisir comment les visualiser.

La première consiste à offrir un **catalogue de graphes** prédéfinis comme ceux que nous avons mentionnés à la section 8.1 : histogrammes, diagrammes en barres, diagrammes de boîtes, diagrammes de dispersions, etc. L'avantage de cette approche est qu'elle paraîtra d'emblée naturelle aux utilisateurs de tableurs de type *Excel*. Son principal inconvénient reste son manque de flexibilité.

À l'autre extrémité des possibilités, l'utilisation de **grammaires formelles** offre une deuxième stratégie pour une construction de graphes personnalisés. C'est par exemple ce qu'offre un environnement d'analyse statistique comme *R*. La flexibilité quasi infinie de cette approche se paie cependant par une courbe d'apprentissage beaucoup plus raide que pour les graphes sur étagère. À tel point, qu'à défaut d'une formation spécialisée, elle restera inaccessible à une majorité d'utilisateurs métiers peu familiers avec les concepts de la programmation.

Certains outils comme *Tableau* tentent de concilier les deux approches précédentes en permettant à l'utilisateur de choisir, par glissé-déposé, les attributs visuels qu'il souhaite utiliser pour l'encodage des différentes variables.

La **sélection de variables** à représenter au moyen de filtres ajustables en temps réel, au moyen de glissières ou de cases à cocher, est un atout supplémentaire pour l'exploration incrémentale des données complexes.

Les fonctions habituelles de tri des variables ordinales sont parfois enrichies d'autres méthodes d'organisation qui s'appliquent par exemple aux structures en graphe propres aux réseaux sociaux. Elles permettent, par exemple, de révéler la proximité et la nature du lien social entre deux individus, comme l'illustre la figure 8.12.

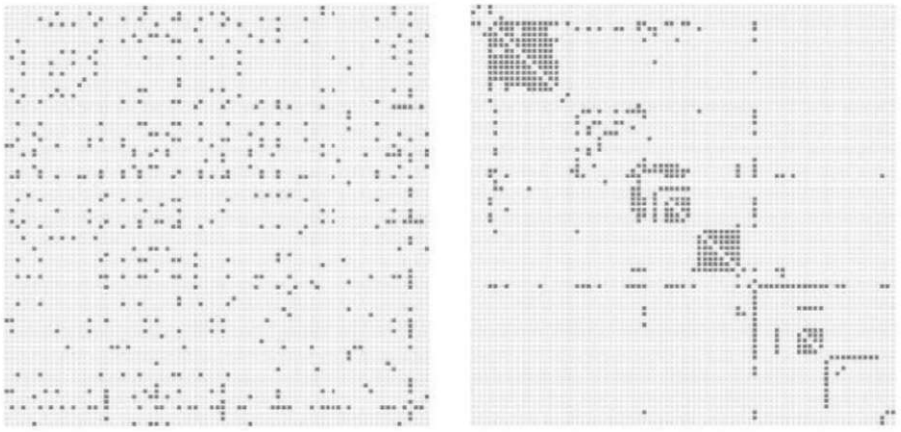
Enfin, la possibilité de définir de manière flexible et de visualiser en temps réel de nouvelles variables dérivées des variables originales est actuellement un domaine de recherches actives.

La manipulation des vues

Une fois la vue des données spécifiées, il est important de pouvoir la manipuler de manière interactive, une opération qui n'est pas indépendante de la sélection de variables évoquée précédemment.

1. *Interactive Dynamics for Visual Analysis* de Jeffrey Heer et Ben Shneiderman, acmqueue (2012) : http://queue.acm.org/detail.cfm?id=2146416.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.



(a) (b)

Figure 8.12 — Une matrice qui représente la connectivité entre individus dans un réseau social avec en (a) un classement par ordre alphabétique et en (b) un classement qui révèle la structure en communautés du réseau.

L'un des principaux défis à relever par les concepteurs d'outils de visualisation dynamique est de proposer des outils qui permettent tout à la fois de sélectionner un sous-ensemble des données à analyser en détail et de contextualiser ces détails au sein d'une vue plus globale. Le processus de navigation naturel dans un ensemble complexe de données se déroule généralement en trois étapes :

1. **recherche de données** qui correspondent à certains critères,
2. **affichage du contexte** de ces données particulières,
3. **affichage à la demande des détails** concernant ces données particulières.

Un exemple typique est celui de données géographiques : on commence par rechercher des territoires qui répondent à certains critères sociaux, géologiques ou météorologiques. Ces zones sont alors mises en évidence dans le contexte d'un territoire plus large. Dans les zones sélectionnées, on choisit ensuite d'afficher certains détails : la nature des sols, le niveau de précipitations, des statistiques sociales détaillées, etc.

Lorsque la représentation utilise des vues multiples qu'il s'agisse de tables, de graphes ou de cartes, il est essentiel que celles-ci soient coordonnées au sens où la sélection ou le masquage d'une information dans l'une des vues soit immédiatement répercutée dans les autres.

Le partage de l'information

Lorsque le processus d'analyse de donnée est collaboratif il est essentiel de pouvoir partager, non seulement les données et les résultats d'une analyse, mais encore tout le paramétrage d'un ensemble de vues et de graphes.

Dans les cas où la spécification et la manipulation d'une vue sont complexes, il est particulièrement utile aussi de pouvoir enregistrer le processus de construction des

Copyright © 2015 Dunod.

vues ainsi que l'interaction avec les données pour pouvoir les rejouer ultérieurement. L'analyse devient alors un processus incrémental.

Les outils modernes comme *Tableau* offrent par ailleurs la possibilité d'intégrer de manière dynamique des tableaux de bord complexes, constitués d'une collection de graphes coordonnés, dans n'importe quel site web ou blog. Ainsi toute modification des données sera automatiquement répercutée dans toutes les publications où elle est utilisée sans qu'il soit nécessaire de procéder à une mise à jour manuelle.

En résumé

La visualisation des données est l'opération par laquelle se révèle le sens des données. Alors que les modèles statistiques utilisés par le *machine learning* autorisent des prédictions quantitatives, la visualisation permet la contextualisation d'un jugement humain. Au data scientist, elle révélera les valeurs aberrantes, la nature et l'amplitude des corrélations entre variables et l'aidera par conséquent à sélectionner les variables prédictives pertinentes. Dynamique, elle favorisera l'exploration des données et la découverte d'aspects insoupçonnés dans un jeu de donnée. Pour l'expert métier, elle constituera le support de l'interprétation et de la comparaison entre différentes hypothèses.

Qu'il s'agisse d'évaluer le degré de liaison entre variables, la performance d'un modèle de classification ou de représenter des données complexes, une représentation de qualité doit impérativement tenir compte des caractéristiques du système visuel humain. Certains attributs visuels sont en effet perçus de manière plus précise que d'autres. L'exploration interactive de jeux de données complexes exige quant à elle une réactivité en quasi temps réel pour être en phase avec le traitement de l'information visuelle par le cerveau humain. Le domaine de la visualisation est aujourd'hui en plein essor notamment grâce à la disponibilité d'outils performant en mode SAAS comme *QlikView* ou *Tableau* qui, outre les fonctions de visualisation proprement dites, intègrent des fonctions sociales et des options de partage et d'intégration avancées.

Copyright © 2015 Dunod.

Copyright © 2015 Dunod.

TROISIÈME PARTIE

Les outils du Big Data

Cette partie traite des plateformes utilisées dans les projets Big Data. On y présente d'abord la plateforme Hadoop et ses principaux outils. Ensuite, on traite de la question de l'utilisation du Big Data dans les situations de type temps réel, c'est-à-dire quand le temps écoulé entre la demande d'information et la réponse doit être le plus court possible.

Cette partie est organisée en quatre chapitres :

- Le chapitre 9 est consacré à la plateforme **Hadoop** et à son écosystème.
- Le chapitre 10 montre comment on peut analyser des fichiers de logs au moyen de **Pig** et **Hive**.
- Le chapitre 11 est consacré à la présentation des **architectures λ**, un nouveau concept qui permet de répondre aux contraintes du temps réel.
- Le chapitre 12 présente le mode de fonctionnement d'**Apache Storm**.

Copyright © 2015 Dunod.

Copyright © 2015 Dunod.

9

L'écosystème Hadoop

Objectif

Initialement conçu pour fournir un environnement d'exécution distribué pour le pattern MapReduce, Hadoop se diversifie aujourd'hui en une myriade de projets satellites. L'objet de ce chapitre est de présenter les principaux systèmes qui constituent Hadoop[1] ainsi que ceux qui viennent étendre ses fonctionnalités.

Deux directions se dessinent parmi les innovations récentes. Il s'agit d'une part de s'affranchir des rigidités du paradigme MapReduce pour parvenir à des vitesses de traitements compatibles avec les exigences d'une exploration interactive de grands volumes de données. De plus en plus c'est cette réactivité qu'attendent aussi bien les data scientists que les experts métiers pour qui elle est un enjeu primordial surtout lorsqu'ils sont face à des clients. D'autre part, au vu de la complexité des développements MapReduce de bas niveau, il s'agit de proposer des systèmes qui permettent l'exploration des données au moyen de requêtes SQL standards et non d'un langage de script exotique.

Les quatre principales distributions de Hadoop seront aussi présentées dans ce chapitre.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

1. Voir également http://blog.ippon.fr/2013/05/14/big-data-la-jungle-des-differentes-distributions-open-source-hadoop/

9.1 LA JUNGLE DE L'ÉLÉPHANT

9.1.1 Distribution ou package

Comme nous l'avons vu au chapitre 4 *Hadoop* est un projet open source de la fondation Apache dont l'objectif principal est de développer des outils qui facilitent la construction d'applications scalables distribuées sur des clusters de serveurs bon marchés. Plus précisément, Hadoop dans sa version originale peut être conçu comme un framework d'exécution distribuée pour le pattern MapReduce.

Il en va avec Hadoop comme avec la plupart des systèmes open source. La gratuité des briques logicielles n'implique en rien, loin s'en faut, la gratuité du service qu'elles apportent. S'agissant de Hadoop, le niveau de technicité requis est particulièrement important, qu'il s'agisse de déployer le framework sur un cluster, de le configurer ou de l'intégrer dans un SI existant. Deux raisons principales à cela :

- L'installation de Hadoop dans un environnement distribué est une tâche complexe qui exige l'écriture de nombreux fichiers de configuration, de scripts et qui demande de spécifier de nombreuses variables d'environnement.
- Les différentes briques logicielles du projet Hadoop évoluent sans cesse et existent donc en plusieurs versions où « *releases* » qui ne sont pas toujours compatibles entre elles. Connaître ces relations de compatibilité exige une veille technologique active que peu d'individus ou d'organisation peuvent mener par eux-mêmes.

Il existe cependant plusieurs solutions pour éviter de sombrer dans les sables mouvants de la jungle Hadoop, sans pour autant avoir à devenir soi-même un expert.

La première solution consiste à utiliser ce qu'on appelle une **distribution Hadoop**. Une distribution est un ensemble cohérent des différentes briques qui constituent l'écosystème Hadoop packagées par un fournisseur. Outre le packaging des composants, le fournisseur peut également vendre ses propres outils graphiques pour simplifier le déploiement, la configuration et le *monitoring* du *framework*. Il assure généralement un support commercial pour sa distribution ainsi qu'une documentation complète et systématiquement maintenue à jour. Les trois principales distributions sont aujourd'hui celles de *Cloudera*, de *Hortonworks* et de *MapR*. Elles seront décrites plus loin. En mode cloud, c'est la distribution *Elastic MapReduce* (EMR) d'Amazon qui, pour l'heure, fait référence.

La seconde possibilité est d'utiliser un **package** Big Data. Un tel package s'appuie sur une ou plusieurs distributions Hadoop, auxquelles viennent s'ajouter des outils de productivité comme :

- des **plugin** intégrés aux environnements de développement comme Eclipse ;
- des **générateurs de code** qui produisent du code MapReduce à partir d'une représentation graphique des services ;
- des **connecteurs** capables de récupérer l'information à partir de supports variés, qu'il s'agisse de SGBDR, de systèmes NoSQL ou de média sociaux comme Twitter ou Facebook, et de les charger dans Hadoop.

Copyright © 2015 Dunod.

Certains éditeurs comme *Oracle* par exemple proposent des *appliances* qui combinent logiciels et matériels.

9.1.2 Un monde de compromis

Pour l'heure, le système qui permettrait l'interrogation en langage naturel en temps réel d'un pétaoctet de données non structurées relève encore de la science-fiction. Pour cette raison, les différents systèmes qui cherchent à améliorer ou à concurrencer Hadoop sont tous contraints d'effectuer des compromis sur différents paramètres :

- les temps de réponse du système,
- la flexibilité du modèle de données que l'on peut interroger,
- la complexité du langage de requête,
- la compatibilité avec les systèmes existants.

À titre d'illustration, voici quelques exemples qui illustrent qu'il n'existe pas de panacée :

- Le langage et la plateforme *R* constituent une solution éprouvée extrêmement puissante pour l'exploration, la manipulation et la visualisation des données. Cependant le type de programmation *R* est très éloigné des langages que connaissent les développeurs IT comme Java, PHP ou C# et ceci constituera une barrière d'apprentissage infranchissable pour beaucoup d'entre eux. Par ailleurs *R* n'est pas forcément adapté à l'industrialisation des tâches de nettoyage et de préparation des données qui précèdent le travail d'analyse proprement dit car il présuppose (du moins dans sa version de base) que les données puissent tenir dans la mémoire vive d'une seule machine.
- *Python* est un langage doté d'excellentes librairies de *machine learning* comme *ScikitLearn* ou *Pandas* mais celles-ci ne sont pas conçues pour analyser des volumes de données qui excèdent ce que peut héberger une seule machine en mémoire.
- MapReduce est un algorithme extrêmement performant pour traiter en mode batch de très grands volumes de données mais n'est cependant pas adapté au traitement itératif qu'exige souvent le *machine learning*. Par ailleurs, le développement MapReduce de bas niveau exige des profils pointus difficiles à trouver.
- Le système *Hive* permet de compiler du code pseudo-SQL en batch MapReduce mais ce code, qui n'est pas strictement compatible avec SQL, comporte de nombreuses subtilités et ne permet pas (encore) les requêtes interactives.
- Des systèmes de base de données en mémoire offrent des temps de réponses inférieures à la seconde mais ils sont propriétaires, extrêmement coûteux et ne permettent pas encore le traitement des très grands volumes de données comme ceux que permettent MapReduce sur un cluster de mille serveurs.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Pour y voir plus clair, on gardera à l'esprit qu'il existe très schématiquement trois types de besoins en analyse de données selon les temps de réponses attendus ou tolérables :

- Des traitements de type ETL sur de très grands volumes de données (allant jusqu'à plusieurs pétaoctets) qui peuvent être effectués en mode batch. La complexité du code n'est pas un enjeu en l'occurrence.
- Des analyses exploratoires et interactives effectuées par des data scientists ou des experts métiers. L'aspect itératif est ici essentiel, de même que la possibilité d'interroger le système au moyen d'un langage standard comme SQL. Les temps de réponses de l'ordre de quelques dizaines de secondes sont en revanche encore acceptables.
- Enfin, les systèmes qui exigent un délai de réponse inférieur à la seconde comme ceux qu'utilisent des agents commerciaux en contact direct avec une clientèle à laquelle ils doivent une réponse sur le champ.

9.1.3 Les services autour de Hadoop

Pour s'orienter dans la jungle de l'éléphant, de nombreux guides proposent leurs services :

- **Communautés open source** – Pour les plus courageux, ou les plus ambitieux si l'on préfère, il y a tout d'abord la communauté Apache Hadoop. Très active, celle-ci offre de nombreux forums, wiki, FAQ et blogs où s'échangent les dernières informations techniques sur la plateforme. Si l'information est naturellement gratuite, elle s'adresse toutefois à une audience de développeurs chevronnés et d'administrateurs dotés d'un fort niveau technique et rompus aux codes des communautés de logiciel libre.
- **Intégrateurs** – Viennent ensuite les intégrateurs et les fournisseurs de distribution Hadoop comme *Cloudera*, *Hortonworks* ou *MapR* ou encore des éditeurs comme *Talend* ou *IBM*. Ces sociétés offrent un accès payant à guichet unique pour toutes les problématiques liées à Hadoop : conseil technologique, recherche de cas d'utilisation Big Data, assistance au déploiement, à l'intégration et à la personnalisation de produits open source.
- **Éditeurs** – Ils offrent des extensions propriétaires pour la plateforme Hadoop. La visualisation des données, l'analyse exploratoire et le *machine learning* sont les domaines d'activité de prédilection de ces sociétés. Une société leader dans ce domaine est *Karmasphere*. Elle propose un environnement collaboratif dédié à l'analyse itérative et à la visualisation de données traitées dans Hadoop.

9.2 LES COMPOSANTS DE APACHE HADOOP

Les principaux modules qui constituent le projet Hadoop ainsi que d'autres projets Apache qui lui sont directement reliés sont décrits dans cette section conçue comme un simple lexique qui n'exige pas d'être lu en ordre séquentiel. Les librairies de *machine*

Copyright © 2015 Dunod.

learning seront abordées dans la section 9.5. La figure 9.1 représente l'articulation des différents composants.

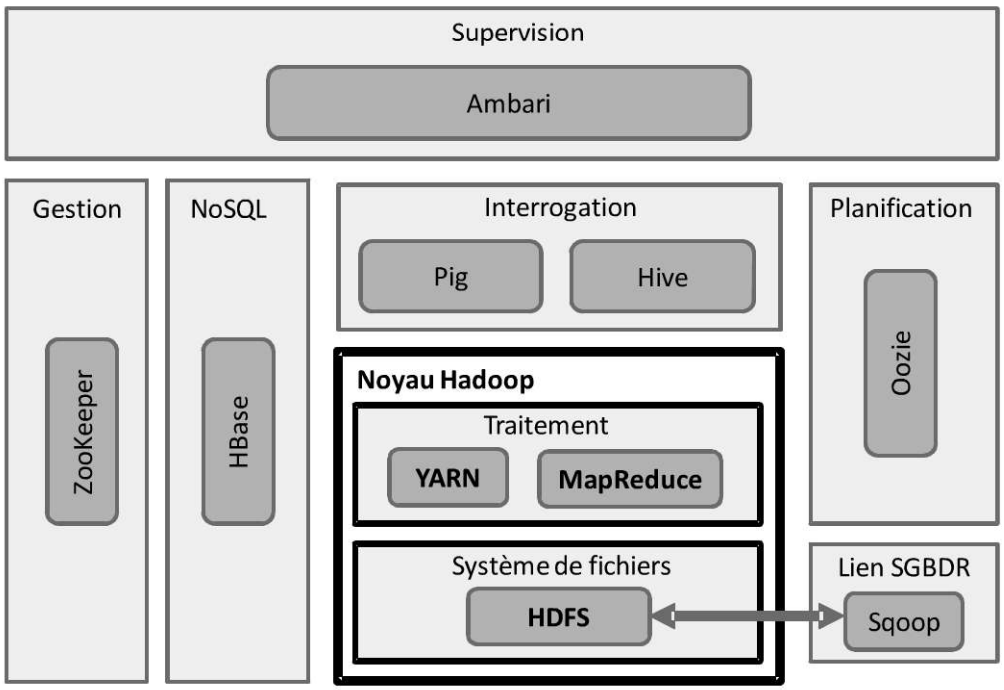


Figure 9.1 — Les différents composants des projets Apache autour de Hadoop qui sont décrit dans ce chapitre.

9.2.1 Hadoop Distributed File System

Hadoop intègre un système de fichiers distribués qui prend en charge toutes les fonctions de réplication des données et de tolérance aux pannes sur un cluster. Il se nomme HDFS pour *Hadoop Distributed File System*.

HDFS a été conçu avant tout pour héberger des fichiers de très grande taille auxquels on accède le plus souvent en lecture seule. Comme tout système de fichiers, HDFS définit une notion de bloc qui correspond à la plus petite quantité de données qu'il est possible de lire ou d'écrire sur le disque. Cette taille est cependant beaucoup plus grande (environ 64 Mo) que dans les systèmes de fichiers ordinaires (512 octets), ceci dans l'objectif d'optimiser les temps de transfert pour des fichiers volumineux. Comme rien n'oblige les différents blocs qui constituent un fichier de résider sur une seule machine, un fichier pourra donc bénéficier de tout l'espace disponible sur le cluster.

L'architecture de base de HDFS est celle d'un système maître-esclave. Au sein d'un cluster HDFS, il existe un nœud maître, appelé *NameNode*, qui gère l'espace de nommage et qui associe les blocs aux différentes machines du cluster. Il gère toutes les métadonnées du système de fichier. Des processus appelés *DataNode*, associés à chaque nœud, gèrent quant à eux les opérations de stockage locales. Sur instruction du NameNode, les DataNode procèdent aux opérations créations, de suppression ou de réplication des blocs.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Remarque : hormis les mécanismes de tolérance aux pannes, communs à tout système de fichiers distribués, HDFS se distingue par une caractéristique qui lui confère un avantage décisif pour l'exécution de MapReduce : chaque fichier est en effet « conscient » de sa localisation au sein du cluster. De cette manière, lorsqu'un *jobtracker* s'apprête à affecter une tâche à un *tasktracker* (voir chapitre 4), il peut le faire en tenant compte de la localisation des données. La bande passante étant en général une denrée rare dans un cluster, ce mécanisme permet alors de déplacer les processus de traitements plutôt que les données ce qui serait beaucoup trop couteux en bande passante.

Notons encore que Hadoop n'impose pas obligatoirement l'usage de HDFS mais propose une API qui permet l'intégration avec d'autres systèmes de fichiers distribués, comme le système S3 d'Amazon par exemple.

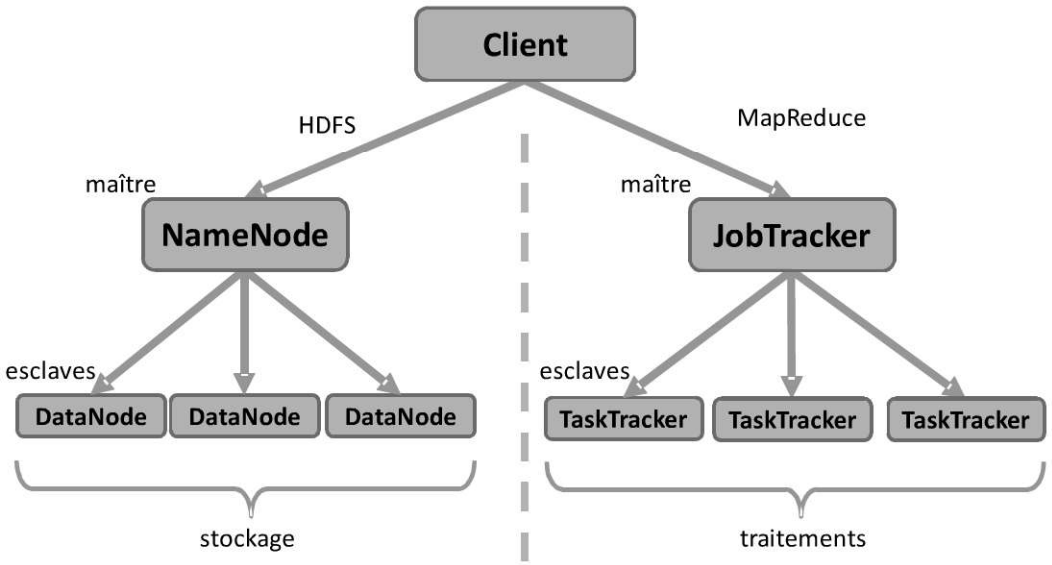


Figure 9.2 — L'articulation des *NameNode* et des *DataNode* au sein d'un cluster Hadoop procède selon un schéma maître esclave similaire à celui des traitements MapReduce.

9.2.2 MapReduce et YARN

Comme nous l'avons mentionné au chapitre 4, le modèle de calcul en batch qui correspond à MapReduce est aujourd'hui challengé par des modèles de calcul en quasi-temps réels plus flexibles et plus performants. Pour faire face à cette exigence de flexibilité, Hadoop 2.0 également nommée YARN (du nom du projet Apache associé), repose sur un principe de dissociation des deux fonctions principales du JobTracker (voir section 4.4.1) à savoir d'un côté la planification des tâches et de l'autre la gestion des ressources du cluster. L'atout de cette nouvelle approche est qu'elle s'ouvre à d'autres modèles de calcul que les batchs MapReduce classiques tout en capitalisant sur l'infrastructure de gestion des ressources distribuées qu'offre Hadoop.

Copyright © 2015 Dunod.

L'architecture de YARN repose sur deux processus principaux : un gestionnaire de ressources partagées par toutes les applications, nommé le *ResourceManager*, et, pour chaque application, un *ApplicationMaster* chargé de négocier les ressources auprès du ResourceManager. La planification des tâches utilisée par l'algorithme MapReduce original devient dès lors un cas particulier de planification par le ResourceManager.

Une API de compatibilité est mise à disposition des développeurs pour l'exécution des jobs MapReduce classiques. Une simple recompilation du code suffira.

9.2.3 HBase

Inspirée de la base de données propriétaire *Bigtable* de Google, *Apache HBase* est une base NoSQL distribuée appartenant à la catégorie des bases de données orientées colonnes(voir section 3.3.3). C'est la base de données utilisée par défaut dans Hadoop. Elle s'appuie en particulier sur le système de fichier HDFS. HBase permet de gérer des très grandes tables constituées de plusieurs milliards d'enregistrements avec plusieurs millions de colonnes. Le modèle est celui des tables creuses (*sparse* en anglais) c'est-à-dire que les valeurs « *null* » ne consomment aucun espace de stockage.

Les principales caractéristiques de HBase sont :

- Une scalabilité linéaire.
- Une utilisation native du système de fichiers distribué HDFS.
- Contrairement à beaucoup d'autres systèmes NoSQL, le modèle de cohérence de HBase n'est pas la consistance finale (*eventual consistency*) mais un modèle de cohérence strict en lecture et en écriture. L'ordre d'écriture est par ailleurs garanti et toutes les opérations de lectures ne voient que les données les plus récentes.
- La tolérance aux pannes est inclue.
- Une API Java facile à utiliser est mise à disposition des applications clientes.
- Une conception lui permettant d'être aussi bien une source qu'une destination d'écriture des jobs MapReduce.
- Une absence de tout mécanisme d'indexation pour les colonnes. Les enregistrements sont toutefois ordonnés selon la valeur de leur clé.
- HBase peut s'exécuter en mode standalone sur une machine pour les besoins d'un développement.

9.2.4 ZooKeeper[1]

À l'origine un sous-projet du projet Apache, *ZooKeeper* est désormais un projet indépendant. L'objectif de ZooKeeper est de fournir un service de coordination de processus distribués au moyen d'un espace de nommage partagé hautement disponible en lecture. Un haut niveau de performance est assuré par une disponibilité des données

1. http://zookeeper.apache.org/doc/trunk/zookeeperOver.html

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

en mémoire. Beaucoup d'applications distribuées utilisent des services de synchronisation de ce type et ZooKeeper évite d'avoir à ré-implémenter ces mécanismes particulièrement délicats à concevoir. Il peut être utilisé dans de nombreux contextes comme :

- l'utilisation d'un service de nommage,
- la mise en place d'un système de configuration distribué,
- l'utilisation d'un mécanisme de synchronisation distribué,
- l'utilisation d'un système de file de message,
- la mise en place d'un système de notifications.

Son fonctionnement est similaire à celui d'un système de fichiers. Des registres de données de taille limitée (moins d'un mégaoctet) appelés **znodes** sont identifiés au moyen d'un chemin. Contrairement à un système de fichiers ordinaire, il est possible d'associer des données à chacun de ces nœuds, comme si un fichier pouvait aussi être un répertoire et inversement. ZooKeeper n'a pas été conçu comme une base de données mais comme un outil de coordination qui permet de partager des données de configuration, de statut, de session etc.

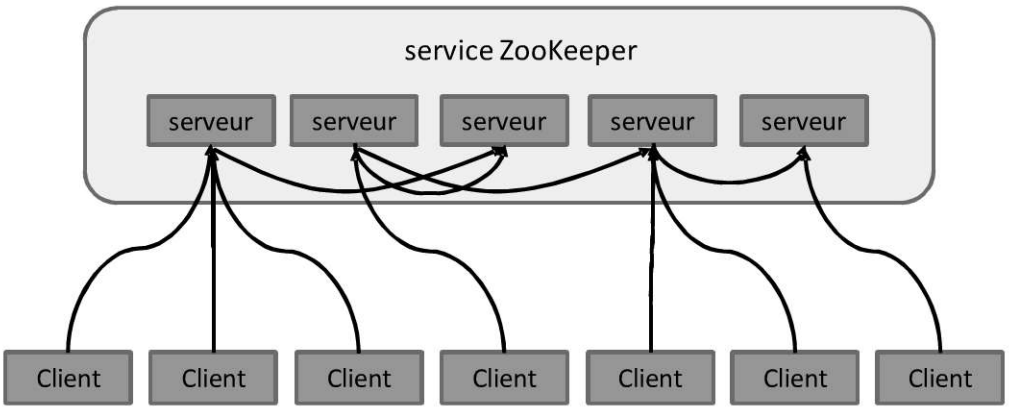


Figure 9.3 — Le service ZooKeeper

Zookeeper permet à une application cliente de définir des notifications spécifiques à certains znodes, appelés des « *watches* ». Sitôt que le contenu du nœud en question change, le client en sera notifié et le watch associé sera supprimé.

Un aspect important de ZooKeeper est la garantie qu'il offre que l'ordre selon lequel les mises à jour sont effectuées est le même que celui dans lequel les ordres ont été transmis. Parmi les autres garanties offertes par ZooKeeper citons encore :

- L'**atomicité** : chaque mise à jour est nécessairement soit réussie soit est considérée comme un échec, il n'y a pas d'état intermédiaire possible.
- L'**unicité de l'image** : un client voit la même image du système ZooKeeper quel que soit le serveur par le biais duquel il y accède.
- La **persistance** : tout changement effectué sur le contenu d'un nœud persiste tant qu'aucun client n'effectue une nouvelle mise à jour.

Copyright © 2015 Dunod.

- La **ponctualité** : la vue du système par un client contient les dernières mises à jour, à un intervalle de temps garanti près.

Chaque client souhaitant utiliser le service ZooKeeper doit connaître la liste des serveurs sur lesquels il est installé. De même, tous les serveurs qui participent au service ZooKeeper doivent se connaître les uns les autres. Le système reste opérationnel tant qu'une majorité des serveurs est disponible.

En bref, ZooKeeper est un framework Java qui contient des briques logicielles conçues pour développer de services de synchronisation et de coordination distribués, robustes et performants.

9.2.5 Pig

Le développement d'une succession d'opérations *map* et *reduce* est une tâche complexe et délicate qui demande un haut niveau d'expertise en programmation et en conception. Augmenter le niveau d'abstraction utilisé pour décrire le processus de traitement est en l'occurrence la solution la plus naturelle. C'est la voie qu'emprunte *Pig*. Ce système comprend d'une part un langage de script, nommé *Pig Latin*, qui permet de définir un flot de transformations de données, un peu à la manière d'un ETL. D'autre part Pig comprend aussi l'environnement d'exécution de ces scripts. Ceux-ci seront automatiquement compilés en une succession de jobs MapReduce, dispensant ainsi le développeur d'avoir à acquérir l'expertise en programmation MapReduce. Parmi les transformations disponibles figurent notamment les jointures que nous avions examinées au chapitre 4.

Le langage Pig Latin comporte de nombreuses similitudes avec SQL et inclus en particulier des opérateurs familiers comme `JOIN`, `GROUP BY` ou `UNION`. Toutefois il existe aussi des différences. Alors que SQL est un langage déclaratif qui est converti en plan d'exécution par le SGBDR (voir section 3.4), le langage Pig Latin s'apparente plutôt à un langage procédural de description d'un flot de données. Comme MapReduce, Pig Latin est conçu pour décrire des opérations de batch et non des requêtes en temps réel. Chaque opération Pig Latin est une transformation sur l'ensemble complet des données. En réalité, l'écriture d'un script Pig Latin s'apparente davantage à l'écriture d'un plan d'exécution de SGBDR qu'à l'écriture d'une requête SQL. Le développeur spécifie explicitement comment les données sont transformées plutôt que le résultat souhaité.

Une autre différence significative avec les SGBDR est que Pig ne comporte pas de processus de chargement de données, celles-ci sont lues directement à partir du système de fichier HDFS. L'économie de temps ainsi réalisée est substantielle.

Comme la plupart des systèmes NoSQL, Pig n'exige pas de schéma de données rigoureux. Un schéma peut toutefois être décrit au moment de l'exécution du script.

Enfin, Pig Latin est un langage hautement extensible. Quasiment toutes les fonctions qu'il s'agisse de jointure, de groupement, de tri ou de filtrage sont redéfinissables par l'utilisateur. Ces extensions peuvent alors constituer des librairies de fonctions

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

réutilisables. L'expérience a montré que celles-ci s'avèrent souvent plus utiles que les librairies de fonctions *map* et *reduce*.

9.2.6 Hive

Comme Pig, *Hive* est une infrastructure de datawarehouse construite au-dessus de Hadoop en vue de simplifier l'analyse d'ensembles de données très volumineux. Hive a été conçu à l'origine par les ingénieurs chez Facebook qui ne disposait pas de suffisamment de développeurs spécialisés dans la conception d'algorithmes MapReduce. L'objectif dès lors était de créer un système de traitement de données dont le langage de script serait très aussi proche que possible du langage SQL ordinaire. Charge ensuite au système Hive de les compiler en jobs MapReduce. La plateforme d'exécution Hive et le langage HiveQL sont le résultat de ces travaux. C'est désormais un projet Apache qui bénéficie de contributions d'autres sociétés comme *Netflix*.

Comme Pig, Hive n'est pas vraiment adapté à l'analyse transactionnelle de données en temps réels ni à des mises à jour au niveau de granularité d'un seul enregistrement. Son temps de latence typique, même pour de petits ensembles de données, se chiffre en minutes. Hive excelle en revanche dans des situations où il s'agit de traiter en mode batch de grandes quantités de données immutables, comme des logs par exemple. Plus proche du SQL standard que Pig Latin, HiveQL est aussi plus facile à apprendre.

Hive propose différents mécanismes d'indexation pour accroître les performances. Comme Pig, Hive donne la possibilité aux développeurs de définir leurs propres fonctions (UDF = *User Defined Functions*).

9.2.7 Oozie

Apache *Oozie* est un outil de planification de jobs Hadoop. Il permet de combiner plusieurs types d'opérations : MapReduce, Pig, Hive ou Sqoop (voir ci-dessous) en une seule unité logique. C'est donc essentiellement un outil conçu pour construire des combinaisons complexes et reproductibles d'opérations de transformations.

9.2.8 Flume[1]

Apache *Flume* est une solution qui permet d'agréger en temps réel différents flux de logs des serveurs web. Les données sont alors écrites dans le système de fichier HDFS. L'un des rôles principaux de Flume est d'isoler les applications clientes des pics de charge durant lesquels la quantité de données reçues excède celle qui peut être écrite sur disque. La conception de Flume garantit qu'aucune donnée n'est jamais perdue. Pour assurer sa scalabilité, Flume est distribué.

1. http://blog.octo.com/introduction-a-flume-ng/

Copyright © 2015 Dunod.

9.2.9 Sqoop

Apache Sqoop est un outil qui permet de transférer efficacement de grands volumes de données entre Hadoop et un SGBDR, un besoin prioritaire pour de nombreuses entreprises. Sqoop est compatible avec les principaux SGBDR comme MySQL, Oracle, IBM DB2 ou Microsoft SQL Server. Sqoop peut transférer les données issues d'un SGBDR soit directement vers le système de fichier HDFS ou alors vers la base orientée colonne HBase ou encore vers Hive.

9.3 LES PRINCIPALES DISTRIBUTIONS HADOOP

Les quatre distributions Hadoop décrites dans ce paragraphe ne sont liées à aucun logiciel ni aucun matériel spécifique. Elles sont par conséquent utilisables dans n'importe quel contexte technique.

9.3.1 Cloudera

Cloudera a été fondée par des experts Hadoop et Doug Cuttings, le concepteur de Hadoop a rejoint les équipes Cloudera en 2009. Cloudera vend aussi bien des licences que du support et des formations. Cloudera propose par ailleurs une version open source de sa plateforme.

Au noyau Hadoop de Apache, Cloudera ajoute ses propres interfaces graphiques d'administration, de gestion et de suivi, des outils de déploiement, des outils de gestion de la sécurité ainsi que des solutions d'intégration à un large éventail de produits. La suite est entièrement testée et documentée par Cloudera.

À ce jour, c'est la distribution de Hadoop qui a le plus de déploiement référencés.

La figure 9.4 montre l'essentiel du contenu de la distribution proposée par Cloudera. Hormis *Impala* dont nous parlerons dans la section 9.4.1, toutes les autres composantes ont été décrites précédemment.

9.3.2 Hortonworks

Hortonworks a été fondée en 2011 par des spécialistes de Hadoop chez Yahoo. C'est la seule dans notre liste à utiliser 100 % des composants open source Apache sans n'y rajouter aucune modification. L'objectif de Hortonworks est de faciliter l'adoption de la plateforme Hadoop d'Apache. Hortonworks est aussi un grand contributeur au noyau de la plateforme Hadoop. Le modèle économique de Hortonworks consiste essentiellement à vendre du support et des formations.

Les composants qui constituent la distribution Hortonworks sont présentés dans la figure 9.5. HCatalog est une couche de gestion de stockage de données qui incorpore une notion de table utilisable conjointement avec MapReduce, Hive et Pig.

Utile pour débuter avec la plateforme Hadoop, Hortonworks propose une machine virtuelle où sont préinstallés tous les composants Hadoop.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

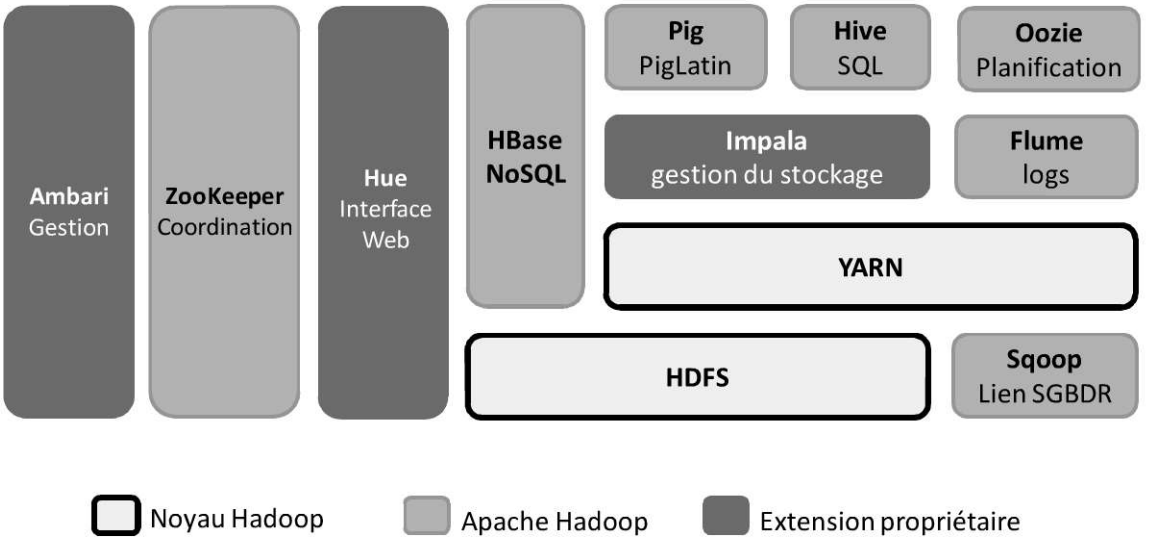


Figure 9.4 — La plateforme Hadoop de Cloudera

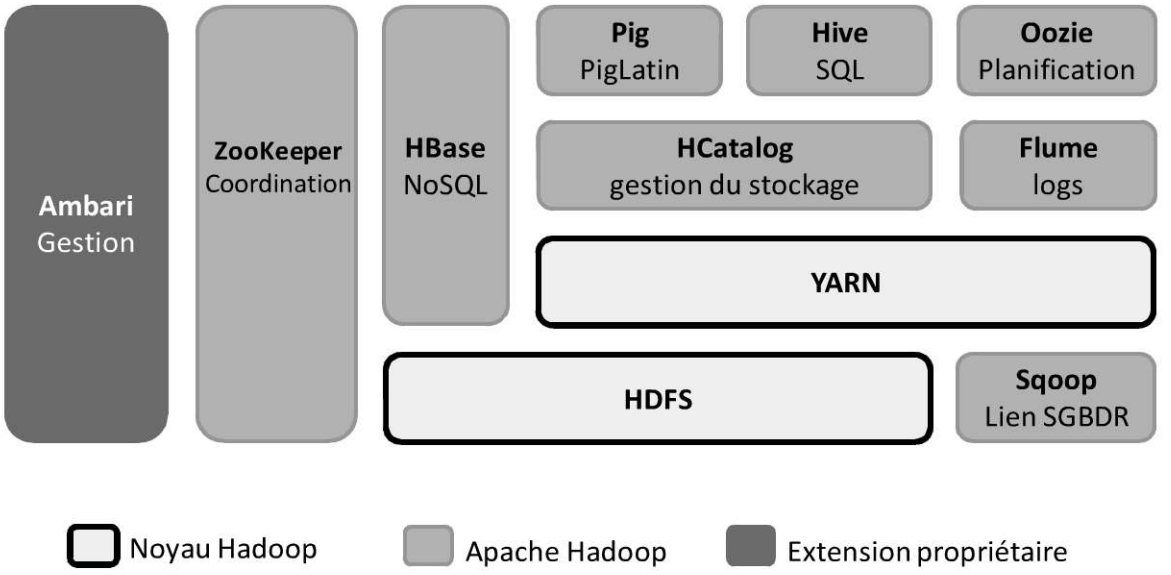


Figure 9.5 — La plateforme Hadoop de Hortonworks

9.3.3 MapR

MapR est une société californienne, fondée en 2009 par des collaborateurs de Google. Elle propose une distribution de Hadoop qui se veut particulièrement facile d'utilisation. Elle enrichit le noyau Hadoop de solutions propriétaires et propose trois distributions : M3 qui est gratuite, M5 qui ajoute des fonctionnalités de haute disponibilité et M7 enfin qui intègre une base de donnée haute disponibilité qui réimplémente l'API de HBase. Elle utilise un système de fichiers propriétaires, MapR FS au lieu du système HDFS de Apache pour accroître les performances et propose également sa propre implémentation de MapReduce : MapR MapReduce.

Copyright © 2015 Dunod.

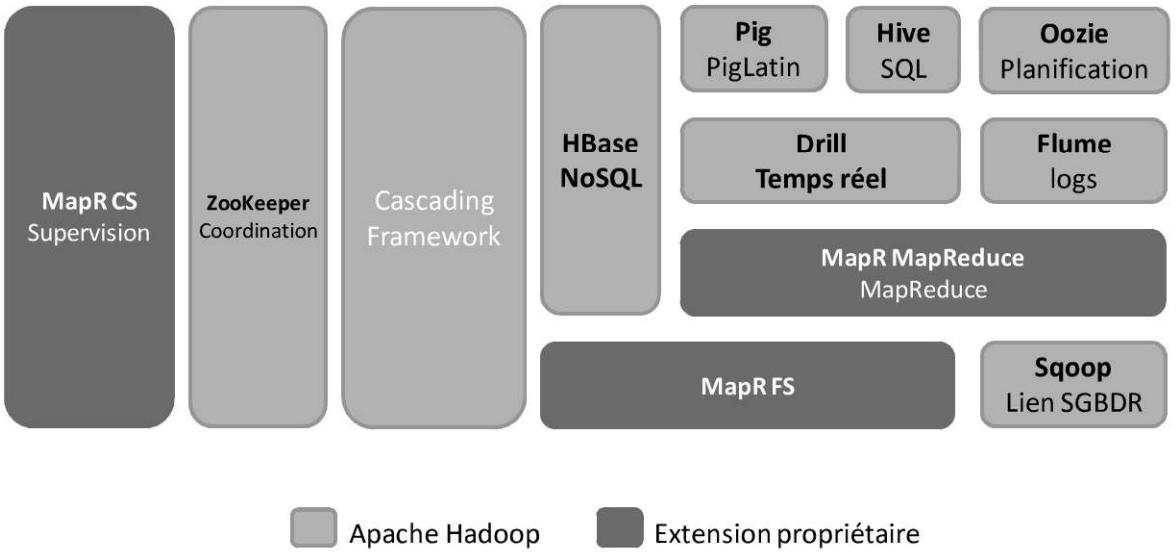


Figure 9.6 — Lla plateforme Hadoop de MapR

MapR a rencontré de nombreux succès commerciaux depuis sa fondation. C'est la distribution choisie par Amazon pour son service cloud Elastic MapReduce (voir ci-dessous). MapR est un partenaire technologique de Google pour sa plateforme IAAS Google Compute Engine.

MapR contribue à de nombreux projet Apache Hadoop comme HBase, Pig Hive, ZooKeeper. Enfin MapR est à la tête du projet Apache *Drill* dont l'objectif est de permettre l'utilisation de requêtes SQL standards sur des données Hadoop et d'offrir des traitements en temps réel (voir ci-dessous).

9.3.4 Amazon Elastic MapReduce

Elastic MapReduce (EMR) est une solution Hadoop dans le cloud proposée par Amazon depuis 2009. Elle est hébergée sur l'infrastructure cloud scalable d'Amazon EC2 et tire profit du service de stockage Amazon S3. EMR est une solution idéale dans les situations où l'on cherche à disposer ponctuellement d'une importante puissance de traitement et de stockage. La solution permet d'économiser aussi bien les coûts liés à l'acquisition des compétences nécessaires au déploiement de Hadoop que l'achat des serveurs. Au prix toutefois d'une limitation des fonctionnalités puisque parmi les composants Hadoop seuls Pig et Hive sont inclus dans le package EMR. On aura tout intérêt par ailleurs à effectuer un rapide calcul du coût du transfert et d'hébergement des données si l'on veut éviter de mauvaises surprises.

Le principal intérêt d'un tel service en mode cloud est son élasticité. Nul besoin de prévoir par avance la capacité de traitement et de stockage nécessaire, celles-ci peuvent être ajustées à la volée et la tarification se base sur les ressources effectivement consommées.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Parmi les inconvénients potentiels dont il faut être conscient, mentionnons les temps de latence élevés pour les entrées sorties sur S3, inévitablement plus importants que sur une installation Hadoop à domicile.

EMR peut être utilisé avec une large panoplie d'outils tiers, qu'il s'agisse de l'environnement de développement, du transfert des données, de la visualisation ou de la BI. Enfin, notons qu'il est également possible d'utiliser la distribution MapR sur EMR.

9.4 LES BRIQUES ANALYTIQUES À VENIR

9.4.1 Impala

Comme nous l'avons mentionné à plusieurs reprises, les deux principales limitations de l'algorithme MapReduce sont d'une part un modèle de programmation inhabituel qui requiert des compétences pointues rares, et d'autre part l'inadéquation de ce modèle pour les traitements interactifs. Les briques Pig et Hive que nous avons déjà décrites résolvent la moitié de ce problème au moyen d'un langage de script, Pig Latin pour le premier, et d'un langage plus proche de SQL pour le second : HiveQL. Pour autant, aucun des deux systèmes n'est adapté aux requêtes interactives, ceci pour une raison simple : les requêtes doivent au préalable être compilées en une séquence de jobs MapReduce avant d'être exécutés sur l'infrastructure Hadoop.

Pour pallier cet inconvénient, le projet *Impala*, initié en 2012 par Cloudera vise à développer une implémentation open source d'un moteur SQL massivement parallèle qui bénéficie de l'infrastructure en cluster de Hadoop, *sans* toutefois utiliser MapReduce. Grâce à Impala, le système Hadoop peut être utilisé aussi bien pour exécuter des batchs volumineux, grâce à MapReduce, que pour répondre à des requêtes interactives, tout en bénéficiant d'un ensemble cohérent de métadonnées pour ces deux tâches. Plus besoin désormais de procéder à de laborieuses conversions ni aucun transfert de données entre un SGBDR et Hadoop. Impala peut utiliser indifféremment HDFS ou HBase pour le stockage des données.

Impala est donc un système concurrent de Hive. Le premier est supporté par Cloudera alors que Hortonworks favorise la modernisation de l'architecture Hive, par le biais notamment de son initiative *Stinger*. Celle-ci est plus en phase avec la philosophie de YARN qui vise aujourd'hui à transformer Hadoop en une plateforme de traitement massivement parallèle capable d'utiliser d'autres modèles de traitements que MapReduce. Il faut cependant garder à l'esprit que ses intérêts des deux sociétés sont très différents. Alors que Cloudera cherche à favoriser ses propres extensions, Hortonworks mise sur son label de société 100 % compatible Apache. À ce jour le match n'est pas encore joué entre les versions modernisées de Hive et Impala.

9.4.2 Drill

Enfin le meilleur, ou du moins le plus ambitieux, pour la fin : le projet Apache *Drill*. Comme pour Impala, l'objectif principal de Drill est de permettre l'analyse

Copyright © 2015 Dunod.

exploratoire et interactive de quantité de données se chiffrant en pétaoctets sur clusters de plusieurs dizaines de milliers de serveurs. Drill utilisera une version de SQL entièrement conforme au standard ANSI. Le projet Drill est inspiré du système *Dremel* de Google, accessible comme IaaS sous le nom de Google *BigQuery*. Bien que Drill soit conçu pour s'exécuter dans l'environnement Hadoop, il ne lui est pas lié et pourra s'exécuter dans n'importe quel environnement distribué sur lequel est installé ZooKeeper (voir section 9.2.4). À l'heure actuelle (fin 2014) Drill est en phase d'incubation chez Apache.

Le cœur de l'architecture de Drill est un environnement d'exécution distribué spécifiquement conçu pour le traitement de données à très grande échelle. Chaque nœud du cluster exécute un service, appelé un *Drillbit*, qui se charge d'intercepter la requête d'un client, de la traiter localement puis de lui renvoyer le résultat. Il n'existe aucune relation maitre-esclave entre les différents nœuds.

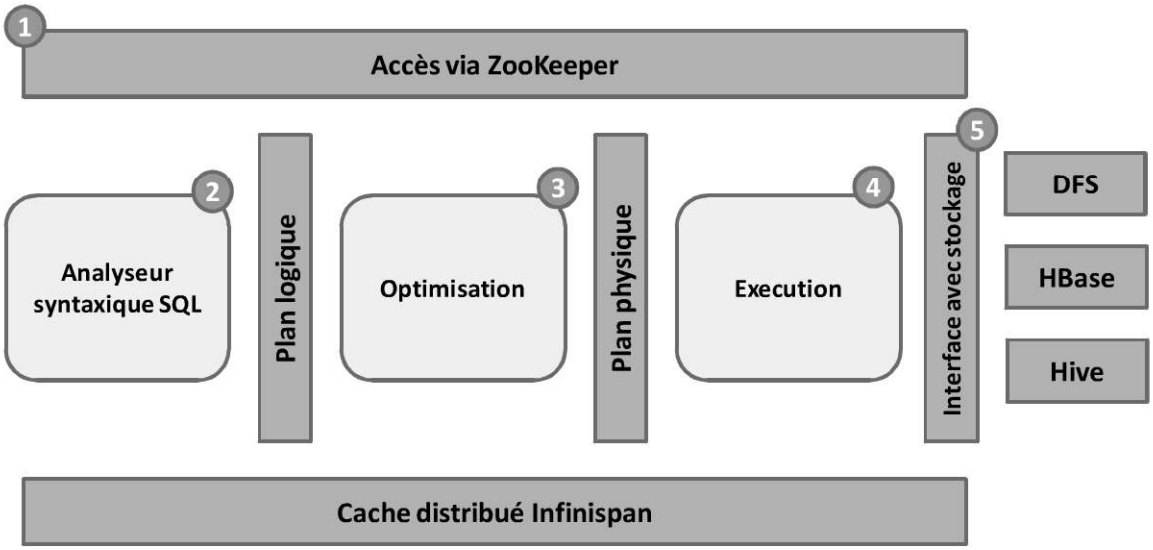


Figure 9.7 — Les principaux composants d'un Drillbit :
(1) le point d'accès à Drill se fait via ZooKeeper ;
(2) l'analyseur syntaxique renvoie un plan logique d'exécution de la requête ;
(3) l'optimiseur renvoie un plan d'exécution physique distribué qui incorpore à la fois les optimisations usuelles des SGBDR et la prise en compte des considérations de localité des données ;
(4) le moteur d'exécution massivement parallèle (MPP) exécute les traitements planifiés ;
(5) Drill utilise différents plugins pour interagir avec les solutions de stockage les plus courantes.

Le processus d'exécution d'une requête Drill se déroule schématiquement de la manière suivante, voir figure 9.7.

1. Un client envoie une requête à Drill par l'intermédiaire du système de coordination ZooKeeper, ce qui le dispense d'avoir à gérer lui-même l'accès aux nœuds individuels du cluster et d'avoir à se préoccuper de l'ajout ou de la suppression de nœuds.
2. Le Drillbit à qui la requête a été transmise par ZooKeeper analyse la requête, l'optimise puis génère un plan d'exécution distribué et optimisé.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

3. Ce Drillbit devient le nœud pilote le temps de l'exécution de la requête. Il détermine une liste de Drillbit disponibles en maximisant l'utilisation de données locales.
4. Le Drillbit planifie l'exécution des multiples fragments de requête sur ces nœuds.
5. Les nœuds effectuent le travail et renvoient la réponse au nœud initiateur.
6. Le nœud initiateur renvoie la réponse au client.

Drill utilise en interne un modèle de données hiérarchique et orienté colonnes qui permet de représenter des structures de données complexes et dynamiques comme ceux utilisés par les systèmes NoSQL. Le modèle de données relationnel n'est qu'un cas particulier simple de ce modèle général. Drill n'impose pas de définir un schéma avant l'exécution d'une requête. Ce modèle est découvert à la volée pendant le traitement à partir des informations fournies par les plugins correspondant aux différentes sources de données.

En conclusion, Drill sera un système concurrent de Hive, Pig et MapReduce utile dans l'exploration interactive de très grands volumes de données. À ce jour les optimisations ne permettent pas encore d'envisager des usages de type OLTP pour lesquels les temps de réponses doivent être inférieurs à la seconde.

9.5 LES LIBRAIRIES DE CALCUL

9.5.1 Mahout

Comme nous l'avions indiqué au chapitre 7 l'une des caractéristiques souhaitable d'un algorithme de *machine learning* est qu'il soit parallélisable pour pouvoir passer à l'échelle. Le projet Apache *Mahout* vise précisément à fournir une librairie Java d'implémentations parallélisables des principaux algorithmes de ML. Mahout contient également des primitives statistiques pour construire de nouvelles implémentations. À ce jour, nombre de ces implémentations utilisent le pattern MapReduce. Récemment cependant (avril 2014), le projet Mahout a officiellement abandonné le paradigme MapReduce pour se tourner vers d'autres modèles de programmation plus flexibles, à l'image de YARN que nous avons évoqué précédemment. Les anciennes implémentations MapReduce continuent toutefois à être maintenues.

Mahout propose d'une part des implémentations d'algorithmes généralistes comme ceux que nous avons présentés au chapitre 7. Ceux-ci sont répartis en trois catégories :

- Les algorithmes de **classification** (p. ex. : *random forest* voir section 7.3.7, régression logistique voir section 7.3.4, naïve Bayes voir section 7.3.3). Ils appartiennent tous à la catégorie des algorithmes de ML supervisés (voir section 7.2.1). Parmi les applications, on trouve la classification automatique de documents, la reconnaissance de caractères ou de visages ou l'attribution d'un morceau de musique à une playlist.

Copyright © 2015 Dunod.

- Les algorithmes de **clustering** (p. ex. : k-means voir section 7.3.5). Ils appartiennent à la catégorie des algorithmes non supervisés (voir section 7.2.1) qui, pour l'essentiel, permettent de regrouper des entités similaires et d'identifier des structures. Le domaine d'application typique est la segmentation d'un marché en niches.
- Les techniques de **réduction dimensionnelle** (p. ex. : analyse en composantes principales voir section 7.3.9). Rappelons que l'objectif ici est de réduire le nombre de caractéristiques des observations pour ne retenir que celles qui sont utiles d'un point de vue prédictif et éviter le fléau de la dimension (voir section 7.14).

Mahout implémente également des algorithmes plus spécifiques au e-business comme le **filtrage collaboratif** (FC) utilisé par la plupart des systèmes de recommandation comme ceux que proposent Amazon, Netflix ou Meetic. Dans ce contexte, le FC désigne un ensemble d'outils statistiques qui permettent de faire des prédictions concernant les centres d'intérêts d'un individu à partir des préférences collectées auprès d'un grand nombre d'individus. L'idée de base du FC est que si deux personnes A et B partagent les mêmes opinions sur un certain nombre de sujets/produits connus, A et B auront également la même opinion sur un nouveau sujet/produit. La prédiction concerne donc un utilisateur en particulier mais elle est faite à partir de l'observation du comportement de beaucoup d'individus, d'où l'adjectif « collaboratif ».

9.5.2 MLlib de Spark

Spark est un projet open source d'analyse de données. Il a été initié par l'université de Berkeley en 2009 et fait partie des projets prioritaires de la fondation Apache depuis février 2014. Comme YARN ou Impala, Spark utilise l'infrastructure distribuée de Hadoop, en particulier son système de fichier HDFS, sans toutefois faire appel au pattern MapReduce. Les performances annoncées sont jusqu'à cent fois plus importantes que celles de MapReduce grâce en particulier à une architecture *in-memory*. Spark se prête donc particulièrement bien à l'analyse interactive et au *machine learning*. Il intègre aussi bien les traitements au fil de l'eau de flots de données que les requêtes SQL. Sa librairie *MLlib* intègre des implémentations scalables des principaux algorithmes de ML que nous avons présenté au chapitre 7 : SVM, la régression logistique, l'algorithme k-means, la régression linéaire, et l'algorithme naïve Bayese. MLlib est utilisable en Java, en Python et en Scala. Spark est désormais intégré à certaines distributions Hadoop comme MapR.

Il existe trois modes de déploiement de Spark dans Hadoop : le mode standalone au-dessus de HDFS, le mode au-dessus de YARN dans Hadoop 2.0 et même à l'intérieur de MapReduce dans Hadoop 1.x comme le montre la figure 9.8.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

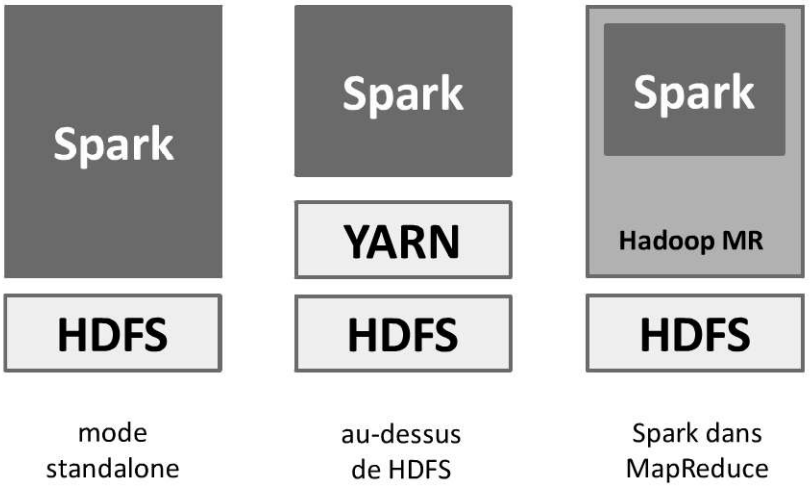


Figure 9.8 — Les trois modes de déploiement de Spark dans Hadoop.

9.5.3 RHadoop

Les outils décrits jusque-là s'adressent tous aux spécialistes IT ou aux experts métiers qui souhaitent bénéficier d'outils de traitement et d'analyse à très grande échelle. *RHadoop*, créé par *Revolution Analytics*, appartient à une autre catégorie puisqu'il s'adresse aux statisticiens chevronnés dont l'outil de prédilection est *R* et aux développeurs MapReduce. Rappelons que *R* est à la fois un langage et un environnement de développement conçu par des statisticiens pour l'analyse statistique. *R* bénéficie d'une communauté estimée aujourd'hui à deux millions d'individus à travers le monde. Conçu au début des années 1990, donc bien avant l'avènement du Big Data, *R* n'est cependant pas prévu pour analyser de très gros volumes de données, celles-ci devant entièrement résider dans la mémoire d'une seule machine. C'est précisément pour pallier à cette limitation que RHadoop a été conçu.

Aux statisticiens programmeurs *R*, RHadoop donne accès au paradigme MapReduce ainsi qu'à tous les bénéfices d'une plateforme distribuée comme Hadoop : traitements massivement parallèles, tolérances aux pannes, réplication de données etc.

Aux développeurs MapReduce, elle permet de simplifier le codage des jobs MapReduce au moyen de librairies dédiées à ce modèle.

RHadoop est constitué de trois packages :

- `rhdfs` : une librairie de fonctions qui donnent accès au système de fichiers distribué HDFS à partir de l'environnement *R*.
- `rmr` : une librairie de fonctions qui donnent accès à MapReduce à partir de *R*.
- `rhbase` : des fonctions qui donnent accès à la base de données distribuées HBase à partir de *R*.

Ces packages sont testés avec les distributions de Cloudera et de Hortonworks.

Copyright © 2015 Dunod.

En résumé

L'écosystème Hadoop est un monde en pleine effervescence. Il a pour l'instant le charme sauvage des terres inexplorées et ravira par conséquent les esprits pionniers. Distinguer les effets d'annonces de la réalité des performances et de la complexité de ces nouveaux systèmes n'est pas toujours une chose aisée. En effet, les intérêts des promoteurs de ces systèmes, qu'il s'agisse de communautés open source technophiles ou d'éditeurs dans le feu de la compétition, coïncident rarement, ni avec l'objectivité, ni avec la clarté des propos tenus.

Deux tendances se dessinent parmi ces extensions, qui parfois se chevauchent au sein d'une même solution. D'une part, avec des systèmes comme YARN, Spark ou Drill, il s'agit de s'affranchir des rigidités du modèle MapReduce qui permet, certes, le traitement de très grands volumes de données mais n'autorise pas l'interactivité. Une course au temps réel est donc engagée. En attendant la disponibilité de systèmes qui offriront des temps de réponse inférieurs à la seconde sur de très grands volumes de données, des systèmes permettant une authentique analyse interactive sont d'ores et déjà disponibles. Un autre enjeu est de proposer un langage d'interrogation des données qui soit aussi proche que possible du SQL standard. C'est ce que proposent des systèmes concurrents comme Hive et Impala par exemple.

Copyright © 2015 Dunod.

Copyright © 2015 Dunod.

10

Analyse de logs avec Pig et Hive

Objectif

Pour illustrer le traitement de données Big Data non structurées, ce chapitre décrit comment il est possible d'exploiter les logs d'un serveur web avec *Pig* et *Hive* pour en extraire des informations utiles sur le comportement des visiteurs d'un site d'e-commerce. Le principal objectif de ces analyses est la détection et l'analyse des signaux avant-coureurs qui précèdent un acte d'achat ou une résiliation de service. Ces informations pourront ensuite être exploitées pour optimiser le *business model*, pour découvrir les parcours de navigation qui amènent un bon taux de conversion ou, plus directement, pour entrer en contact avec les clients hésitants ou sur le point de passer à la concurrence.

10.1 POURQUOI ANALYSER DES LOGS ?

Chaque interaction d'un internaute sur un site web se traduit par l'écriture de dizaines voire de centaines de lignes dans les logs (les « journaux ») des serveurs. Il est aujourd'hui possible de « suivre » ces interactions de manière très fine qu'il s'agisse de saisie de texte, de clic sur des liens ou de *scrolling* d'une sous-fenêtre. Ces données essentiellement techniques ne sont pas exploitables en l'état et demandent à être traitées puis enrichies pour révéler toute l'information qu'elles recèlent. Pour faire une comparaison avec le commerce traditionnel, c'est un peu comme s'il était possible d'analyser durant plusieurs mois le parcours détaillé de tous les clients d'un centre

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

commercial. Chaque parcours inclut non seulement des informations de position dans le magasin mais également les temps de séjours dans chaque rayon ou même devant chaque produit.

Une meilleure compréhension du comportement des clients ouvre naturellement un vaste champ d'actions qui va de l'amélioration de la navigation du site afin d'optimiser le taux de conversion jusqu'à la prise de contact directe avec les clients indécis ou insatisfaits, en passant par l'évaluation de clients à risques (dans le cas d'une souscription d'un crédit par exemple).

L'exploitation des logs comporte deux facettes. La première relève de l'analyse statistique. La connaissance du comportement d'une grande population de clients potentiels constituera en effet une source utile pour alimenter les modèles de *machine learning* : anticipation des revenus générés, identification de niches de marché, détection de tendances par *clustering*, etc. La seconde facette relève du contact individualisé avec la clientèle. L'objectif ici est d'établir une relation de confiance basée sur la prise en compte d'intérêts client très spécifiques.

10.2 POURQUOI CHOISIR PIG OU HIVE ?

Les systèmes *Pig* et *Hive* ont été présentés au chapitre précédent 9. Tous deux permettent de créer, rappelons-le, des jobs MapReduce sur Hadoop sans qu'il soit nécessaire de les coder explicitement. Ce sont donc des outils particulièrement bien adaptés au traitement en mode batch de quantités massives de données peu structurées, comme les logs. Données pour lesquelles les SGBDR sont souvent à la fois inadaptés et d'un coût prohibitif. Ajoutons encore que les langages Pig Latin et Hive sont d'un accès aisé pour quiconque connait quelques rudiments de SQL.

Il n'est pas rare pour des sites à gros trafic d'être confronté à des volumes de données de logs qui dépassent d'un facteur 100 voire 1 000 le volume de données hébergées dans un SGBDR classique ou dans un CRM. On peut typiquement les chiffrer en dizaines voire en centaines de téra-octets. Ces volumes importants s'expliquent à la fois par la granularité du tracking consigné dans les logs et par le nombre de visites. Concrètement, une ligne de log est générée pour chaque vue d'une page, pour chaque interaction sur une page (affichage d'un bloc de contenu, réponse à une question d'un formulaire) et même pour chaque micro-évènement (scrolling, déplacement d'un curseur). C'est notamment ce tracking intra-page qui multiplie facilement les volumes de logs d'un facteur 10 ou 100.

Venons-en rapidement à ce qui distingue *Pig* et *Hive*. Bien que le choix d'utiliser l'un ou l'autre relève souvent du goût ou des compétences d'un développeur, on peut cependant relever quelques différences objectives.

Pig avec son langage de script *Pig Latin* est d'un maniement similaire aux anciennes procédures stockées. Ce caractère procédural en fera un outil de choix pour les tâches lourdes de types ETL sur des données peu structurées. Dans l'analyse de log et dans le ML en général, *Pig* pourra être utilisé avec profit dans les phases de préparation des données avant le ML proprement dit : nettoyage des données, mise en conformité,

Copyright © 2015 Dunod.

traitement des données manquantes, etc. Il permet de passer des données brutes à des données mises en forme. Il est utile pour construire par exemple une série d'indicateurs en aval. Pig peut être combiné avec profit avec un outil de *workflow* comme *Oozie*, voir section 9.2.7.

Hive et son langage *HiveQL* se rapproche davantage du langage SQL standard. Ce sera par conséquent un outil idéal dans un contexte BI pour procéder, par exemple, à des agrégations et des jointures sur des données partiellement structurées avant de les représenter visuellement sous forme de rapport. Sa proximité avec SQL le rend facile d'accès pour un grand nombre de développeurs et même d'analystes métier. C'est un outil adapté à la construction d'indicateurs agrégés. Enfin, il apporte une interopérabilité avec les outils BI tout en offrant une scalabilité quasi infinie.

10.3 LA PRÉPARATION DES DONNÉES

L'objectif de la préparation des données lorsqu'on part de fichiers de logs consiste à générer des variables, ou des colonnes si l'on préfère, qui permettent de décrire le comportement des internautes sur un site d'e-commerce : pré-achat, achat, visite par curiosité etc. En d'autres termes, il s'agit de passer d'une vue « par évènement » (qui est celle des logs) à une vue « par internaute ou par client », voir figure 10.1.

vue par évènement

ID_visiteur (cookie)	date	action (URL)
8593DJ81	04-11-13	URL1.display.do
9001KA54	30-12-13	URL2.order.do
0428OA43	23-02-14	URL3.order.d

vue par internaute

ID_visiteur (cookie)	page vue	temps passé (secondes)
8593DJ81	3	73
9001KA54	5	32
0428OA43	1	15

Figure 10.1 — Passage d'une vue par évènement à une vue par client.

Pour aller au-delà du simple temps passé par page et par visiteur on pourra synthétiser sur une seule ligne de la description par internaute une information riche comme :

« *M. Dupond a effectué 13 visites ces 6 derniers mois, il a vu 4 pages en moyenne, la dernière fois il y a passé 50 % de temps de plus que la moyenne lors des visites précédentes et a acheté le produit n° 7.* »

Concrètement la préparation des données de logs va donc consister à procéder à des agrégations au niveau des utilisateurs et à calculer des indicateurs pour chacun d'eux. On va s'intéresser par exemple à des variables liées à la séquence des pages vues, aux catégories de pages consultées (institutionnelles, catalogue, page de soumission de commande, etc.) à des informations de durée et à leur déviation par rapport à un comportement moyen par catégorie de visiteur (femmes de moins de 40 ans).

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

10.3.1 Le format des lignes de logs

À titre d'exemple, les données brutes en entrée sont constituées des lignes de logs d'un serveur Apache. Parmi les champs utilisables, on trouvera ceux listés ci-dessous. Chacun devra faire l'objet d'une première analyse syntaxique élémentaire (*parsing*) :

- `remote_ip` : l'adresse IP de la machine à l'origine d'une requête ou d'un clic à partir de laquelle on peut procéder à la géolocalisation.
- `request_datetime` : la date et l'heure à laquelle a eu lieu la requête. `request` : l'URL qui a été demandée ainsi que d'autres informations sur les modalités de la requête (`GET` ou `POST`).
- `status code` : le code renvoyé par le serveur (comme 200, 404, etc.) pour indiquer si la requête a abouti ou non.
- `size` : la taille de la page renvoyée par le serveur.
- `referer` : l'URL de la page à l'origine de la requête présente, utile pour comprendre l'enchaînement des différentes actions.
- `user_agent` : un amalgame d'informations diverses dont la marque et la version du navigateur utilisé (*Chrome*, *Safari*, *IE*, *Firefox*), le type de terminal (smartphone, tablette, PC), le système d'exploitation (*Windows*, *Linux*, *Mac OS*), l'architecture 32/64 bits du processeur, etc.

10.3.2 L'enrichissement des logs

À partir des données précédentes on va procéder à divers enrichissements par croisement des données tirées des logs avec des données annexes. C'est l'un des aspects du *feature engineering* décrit dans la section 7.4.3.

Enrichissements temporels

À partir du champ `request_datetime` qui contient l'heure d'un clic on pourra tirer nombre d'informations implicites supplémentaires. Ainsi on saura par exemple si la page a été consultée durant les heures de bureau ou durant une pause. Ce type d'information pourrait améliorer un modèle de prédiction s'il s'agit d'acheter un produit de loisir ou de réserver des vacances. De même, on pourra par croisement avec des calendriers externes déterminer si le clic est intervenu un jour de semaine, durant le week-end ou durant une période de vacances scolaires.

Enrichissements géographiques

Les enrichissements géographiques à partir des données contenues dans `remote_ip` peuvent s'avérer plus délicats car la mise en correspondance d'une adresse IP avec un couple latitude-longitude n'a pas partout la même précision, les résultats étant en général moins précis dans les zones rurales que dans les grandes agglomérations. Des bases toponymiques publiques comme celles d'*Open Street Map* permettent ensuite de passer du couple longitude-latitude à des noms de villes, de département ou de région.

Copyright © 2015 Dunod.

Ces informations géographiques sont riches d'enseignement pour les taux de conversions, les habitudes d'achat, la fréquence de transit d'un lieu à un autre. De plus, elles ouvrent la voie à d'autres enrichissements par :

- des données à caractère socioéconomique : revenu moyen par habitant, niveau de chômage, taux d'imposition local, etc. qui permettront d'ajuster le niveau d'une offre des services ;
- des listes de points d'intérêts (ou POI) touristiques auquel un service pourra être rattaché.

Enrichissements météorologiques

Si on dispose simultanément du champ de position `remote_ip` et de l'horodatage `request_datetime`, on pourra aisément récupérer des données météorologiques qui seront utiles pour corréler par exemple les ventes de glaces ou de parapluies à des volumes de ventes à la température et au niveau d'hygrométrie.

Enrichissements avec des données Open Data

Les données ouvertes, telles que celles que mettent à disposition le gouvernement sur `data.gouv.fr`, la SNCF ou la RATP, constituent une source non négligeable d'enrichissements sur des sujets aussi variés que la nature des sols, les informations culturelles, économiques, la consommation énergétique, le débit ADSL moyen d'une commune, le cadastre, le logement, la santé ou les transports publics.

Des listes de POI pourront être constituées à partir de ces données : quel est le nombre de restaurants dans un rayon de 500 mètres ?

Enrichissements par calculs

Le plus souvent il s'agit de données qui résument le comportement d'un utilisateur au cours du temps, sa manière de parcourir un site par exemple. Parmi les informations utiles calculables citons :

- La page d'entrée et la page de sortie d'un site.
- La séquence des pages vues à laquelle on pourra retrancher les pages qui ne sont pas porteuses de sens (page d'authentification).
- Le nombre de pages vues sur différentes plages de temps.
- Le nombre de pages vues par catégorie de produit.
- Le délai entre deux pages visitées.
- Le nombre de produits ou le prix total des produits mis dans un panier.
- Des ratios significatifs tels que les nombres de pages vues pendant une période spécifique (soldes) par rapport à une période de référence.

Des dizaines de variables supplémentaires peuvent être créées de la sorte.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

10.3.3 La reconstruction des sessions

Un niveau d'agrégation des données particulièrement pertinent pour un site d'e-commerce est le regroupement de données par **session** utilisateur. Par session on entend généralement la visite par un internaute d'une succession pages d'un site pour laquelle le temps écoulé entre deux pages consécutives n'excède pas une certaine durée fixée par avance. Par convention, on accepte souvent la valeur de 30 minutes.

D'un point de vue technique, différents cas se présentent selon la manière dont la session est identifiée :

1. Un cookie de session identifie la session.
2. L'utilisateur s'est authentifié auprès du site.
3. Aucun des deux cas précédents.

Dans le premier cas les informations de session sont déjà disponibles et aucune reconstruction n'est nécessaire. Le deuxième et le troisième cas nécessitent en revanche de reconstruire la session. Dans le troisième cas on pourra considérer que la concaténation de l'adresse IP `remote_ip` et du `user_agent` constitue une bonne approximation d'un identifiant utilisateur.

Voici un exemple de code *Pig* utilisé pour réaliser ces tâches :

```
-- On charge le schéma
logs = LOAD 'logs.tsv' AS (request_datetime:chararray, member_id:int,
url:chararray);
-- On rajoute une colonne  session ID  à chaque ligne par regroupement
sessions = FOREACH(GROUP logs BY member_id){
   ordered = ORDER logs BY request_datetime;
   GENERATE FLATTEN(Sessionize(ordered)) AS
      (request_datetime, member_id, session_id);
};
-- Ici on ajoute le rang temporel dans la session de telle ou telle ligne
sessions_final = FOREACH(GROUP d BY session_id){
   ordered ) ORDER sessions BY request_datetime;
   GENERATE FLATTEN(Enumerate(ordered)) AS
      (request_datetime, member_id, session_id, request_rank)
};
```

10.3.4 Agrégations et calculs

Une fois que les sessions ont été reconstruites, on peut calculer d'indicateurs qui résument le parcours d'un utilisateur durant la session. Pour chaque session on extrait des informations telles que :

- le temps passé dans la session,
- le navigateur utilisé,
- la zone géographique,
- le nombre de clics sur certaines pages.

Copyright © 2015 Dunod.

Voici un exemple en Hive qui calcule, pour chaque session, le nombre total de pages, les temps passés dans la session, la longitude et la latitude, la marque du navigateur et enfin le nombre d'URL similaires à la page « sale confirmation » (confirmation de vente).

```
CREATE TABLE session_aggregated as
SELECT session_id,
   count(*) as nb_pages,
   sum(page_time) as total_time,
   max(latitude) as latitude, max(longitude) as longitude,
   max(user_agent_brand) as user_agent_brand,
   sum(case when url like '%sale_confirmation%' then 1 else 0 end) as hit_sales
FROM logs_sessionized
GROUP BY session_id
HAVING nb_pages > 1 -- drop length 1 session
```

Voici un autre exemple où l'on calcul cette fois le temps de session moyen par navigateur (Chrome, Safari, IE), toutes sessions confondues :

```
SELECT user_agent_brand,
   AVG(total_time) as mean_total_time
FROM session_aggregated
GROUP BY user_agent_brand
```

Et voici l'histogramme associé :

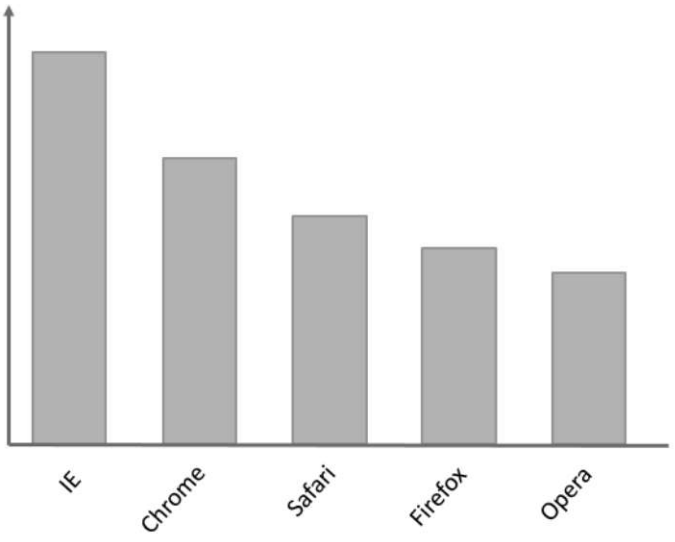


Figure 10.2 — Le temps moyen par individu passé sur un site en fonction du navigateur utilisé.

On constate sur la figure 10.2 que les parcours durent significativement plus longtemps sur certains navigateurs. Les questions qu'on peut alors se poser sont :

- Y a-t-il une relation entre le navigateur et le type de terminal utilisé (ordinateur, tablette ou smartphone) ? Auquel cas il serait normal de constater qu'une consultation en mode nomade dure un peu moins longtemps.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

- Si tel n'est pas le cas, peut-être que la différence de temps de parcours est révélatrice d'un problème de performance ou de compatibilité JavaScript sur un site mal optimisé pour certains navigateurs ?

Dans un second temps, on va agréger certaines informations sur plusieurs sessions. Ainsi, pour un même utilisateur, pourra-t-on demander :

- Quelle est la fréquence de visite (nombre de sessions par mois par exemple) du site pour tel utilisateur ?
- Tel utilisateur a-t-il déjà été sur le point de mener un acte d'achat à son terme ?

10.4 L'ANALYSE DES PARCOURS CLIENTS

C'est à partir des regroupements par utilisateur, par session, par navigateur ou par zone géographique qu'on va essayer de comprendre la dynamique des clients sur un site.

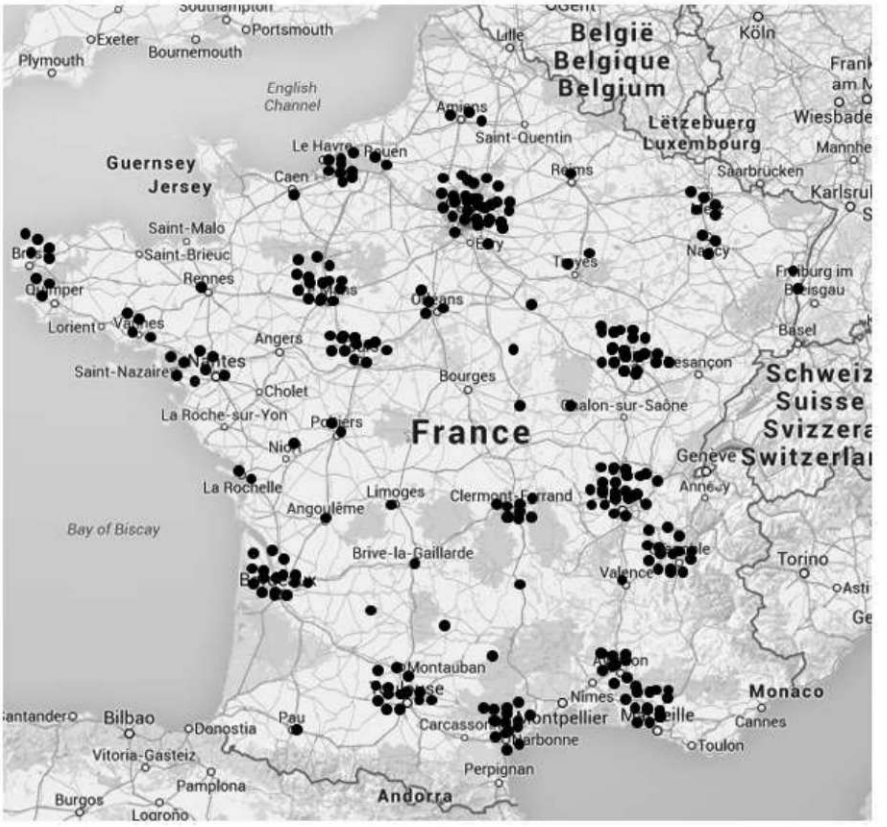


Figure 10.3 — Une représentation de la répartition géographique des clics.

À partir d'une visualisation géographique du nombre de clics on peut par exemple identifier un problème technique ponctuel lié à un serveur régional si le nombre de clics n'est pas proportionnel à la densité de population. Sur la figure 10.3, la corrélation entre densité de population et nombre de clics semble a priori cohérente. Une palette de couleurs pourrait être utilisée pour représenter le nombre de pages vues par session.

Copyright © 2015 Dunod.

Une fois les sessions reconstruites on pourra s'intéresser à la chronologie des transitions entre pages pour les différentes catégories de clients. Les questions auxquels on va souhaiter répondre sont du type :

- Combien de clics sont nécessaires pour arriver à une certaine URL décisive pour l'acte d'achat par exemple ?
- Peut-on identifier différentes catégories d'utilisateurs selon le type de transitions qu'ils effectuent entre pages ?

On peut également s'intéresser aux transitions entre pages et chercher à savoir, pour une page cible donnée comme la page panier/achat par exemple, d'où vient l'utilisateur. Ou, à l'inverse, pour une page de départ fixée, comprendre où va l'utilisateur. L'objectif global de l'analyse pourra être d'identifier des parcours types pour parvenir à une page donnée pour associer par la suite des catégories d'utilisateurs à chacun d'eux.

Voici un exemple de code *Hive* qui permet de calculer les transitions entre pages :

```
SELECT url,
   prev_url,
   counts,
   counts/(sum(counts)) over(partition by url) as percent_global,
   counts/(sum(counts)) over(partition by prev_url) as percent_from
FROM (
   SELECT url, prev_url, count(*) as counts
   from logs_session
GROUP BY url, prev_url
) s1 ;
```

Voici un autre exemple de code Hive qui permet de calculer au bout de combien de clics on parvient à une page donnée :

```
SELECT url,
   myrank, -- position of url in the session
   counts,
   counts/(sum(counts) over() as percent_global,
   counts/(sum(counts) over(partition by ulr)) as url_distrib,
   counts/(sum(counts) over(partition by rank)) as percentage_at_fixed_rank
FROM (
   SELECT url, myrank, count(*) as counts
   from logs_session
GROUP BY url, myrank
) s1 ;
```

La figure 10.4 propose un graphe qui représente la distribution du nombre de clics pour quatre pages accessibles depuis la page « produit » du site.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

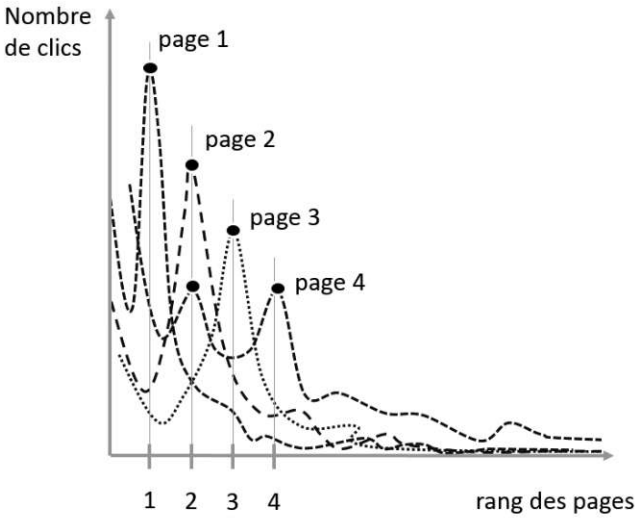


Figure 10.4 — Distribution du nombre de clics pour quatre pages d'un site d'e-commerce. Chaque courbe correspond à une page et représente le nombre clics en fonction de la longueur du chemin parcouru pour y parvenir.

Un tel histogramme permet de confronter ce que l'on imagine a priori de la visite d'un site, une lecture séquentielle des pages par exemple, à la réalité mesurée. En l'occurrence on constate d'une part que les pics se décalent vers la droite, ce qui indique que les quatre pages sont majoritairement visitées dans un ordre précis. On constate d'autre part que la hauteur des pics diminue d'une page à l'autre ce qui indique que l'on perd une fraction des visiteurs à chaque transition. Enfin, pour la page 4 on constate un pic secondaire pour un rang égal à 2, ce qui indique que certains utilisateurs y parviennent par un raccourci. Ce comportement est-il volontaire ? Est-il favorable pour le taux de conversion ? Voilà autant de questions sur lesquels devront plancher les designers du site et les experts marketing.

Une analyse plus ambitieuse pour chercher à identifier des catégories d'utilisateurs selon leur comportement : les curieux, les hésitants, les impulsifs, les raisonnables, etc.

En résumé

Depuis des décennies les analystes marketing utilisent des données structurées des CRM et des systèmes transactionnels pour comprendre le comportement des individus. Depuis peu, il est possible de mettre à profit les données semi-structurées contenues dans les logs techniques des serveurs web. Ces données sont souvent extrêmement volumineuses et, de prime abord, peu parlantes. Pour les exploiter, il faut recourir à des nouveaux outils comme Pig et Hive de l'écosystème Hadoop pour les stocker et pour faire les calculs d'agrégation qui les rendront intelligibles. Ces opérations sont très similaires à ceux qu'effectuent les analystes avec SQL.

Copyright © 2015 Dunod.

11

Les architectures λ

Objectif

Ce chapitre a pour objectif de présenter les enjeux, les problèmes et les solutions de l'utilisation du Big Data dans les situations où il est nécessaire d'obtenir un résultat dans un délai très court. À cette fin, un nouveau concept appelé architectures λ a vu le jour pour permettre la mise en œuvre du Big Data dans les contextes proches du temps réel. Ces nouvelles architectures tentent de pallier les lacunes des solutions précédemment décrites dans cet ouvrage (e.g. Hadoop) et qui ne donnent pas toujours satisfaction en raison de temps de traitement trop longs.

11.1 LES ENJEUX DU TEMPS RÉEL

11.1.1 Qu'est-ce que le temps réel ?

L'expression « temps réel » peut avoir plusieurs significations en fonction du contexte dans lequel il est utilisé. Il faut donc préciser ce que l'on entend par temps réel sur des projets de Big Data.

Le temps réel des systèmes embarqués ou du matériel médical signifie que les temps de réponses des systèmes doivent être de l'ordre de la microseconde et que généralement la défaillance du système (c'est-à-dire son incapacité à effectuer le traitement dans les temps) peut avoir des conséquences catastrophiques, voire mettre des vies en danger : systèmes embarqués dans des avions, ordinateur de bord des voitures, matériel de bloc opératoire...

À côté du temps réel des systèmes embarqués, le temps réel que l'on appellera « temps réel de l'informatique de gestion » se retrouve dans des situations où un

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

système informatique doit répondre en continu aux sollicitations qu'il reçoit. Dans ces cas-là, les temps de réponse ne sont plus de l'ordre de la milliseconde mais de la seconde, voire de la dizaine de secondes. De plus, la défaillance (un temps de traitement excessif) n'entraîne habituellement pas de conséquences catastrophiques mais peut néanmoins porter préjudice : insatisfaction du client, perte de chiffre d'affaires, amende pour non-conformité. Pour ce type de temps réel on distinguera les cas où le délai de réponse attendu est de l'ordre de la seconde et ceux pour lesquels il peut atteindre plusieurs secondes, voire dizaines de secondes.

Le tableau 11.1 récapitule les deux différentes formes de temps réel.

Tableau 11.1 — Les différents types de temps réel.

Type de temps réel	Usages	Latence
Temps réel embarqué	Aéronautique Automobile Matériel médical	Microseconde
Temps réel de gestion	Opérations de bourse Systèmes de réservation	Secondes
Quasi temps réel de gestion	Systèmes de tracking Systèmes de gestion de stocks Réseaux sociaux (Twitter)	De quelques secondes à une dizaine de secondes

Dans le cadre des projets de type Big Data, c'est bien sûr uniquement les situations de type temps réel de gestion qui seront rencontrées et en aucun cas des situations de type temps réel embarqué. On aura donc dans la plupart des cas au moins quelques secondes pour répondre aux requêtes.

11.1.2 Quelques exemples de cas réels

Il existe de nombreuses situations où l'on souhaite obtenir une réponse rapide à des traitements complexes dont les sources de données changent très rapidement :

- **La lutte contre la fraude** – C'est un grand classique de l'analyse temps réel. Il s'agit des situations où pour découvrir une situation de fraude, il est nécessaire d'analyser un grand volume de données dans un laps de temps très court. On peut citer ici en exemple la fraude aux péages autoroutiers où il faut combiner l'analyse du ticket et celles des données collectées sur le parcours (e.g. des images des plaques d'immatriculation). Il faut traiter ces informations **très** rapidement pour ne pas ralentir les véhicules sortants et ne pas provoquer d'embouteillages.
- **L'analyse des données de micro-messaging** – Les utilisateurs des réseaux sociaux tels que Twitter émettent des centaines de milliers de messages chaque seconde. Pour pouvoir les monétiser, le réseau social doit analyser les messages dans un temps très court pour en comprendre le contenu et vendre par l'intermédiaire de sa régie publicitaire des liens ou tweets sponsorisés dépendant du contexte de l'utilisateur aux annonceurs.
- **L'analyse des données de géolocalisation** – Les smartphones embarquent désormais des mécanismes de géolocalisation qui fonctionnent à la fois en

Copyright © 2015 Dunod.

extérieur (GPS) et en intérieur (Bluetooth LE, iBeacon...). Si l'utilisateur le permet, il est donc possible pour les applications installées sur le smartphone de suivre les déplacements du client, par exemple dans un centre commercial. Le système d'information peut ainsi analyser en temps réel le parcours du client et lui envoyer des informations où des offres promotionnelles (e.g. un coupon de réduction) en temps réel.

Jusqu'à présent la plupart des projets Big Data portaient sur des traitements de données en différé. On commence néanmoins à rencontrer des nouveaux projets qui nécessitent cette capacité temps réel de gestion.

11.2 RAPPELS SUR MAPREDUCE ET HADOOP

On a pu constater dans les chapitres 4 et 9 consacrés à l'algorithme MapReduce et à son implémentation de référence Hadoop que son mode de fonctionnement était basé sur l'exécution de traitements batch en parallèle dont les résultats intermédiaires étaient ensuite agrégés. Cette forme de traitement n'est clairement pas adaptée aux situations où les temps de réponse doivent être le plus court possible.

Une première piste pour accélérer les temps de traitement est apparue avec Hadoop 2, elle a été présentée au chapitre 9 avec les outils YARN et Spark. Ces solutions permettent l'accélération des traitements mais cela reste insuffisant dans de nombreux cas. Il faut donc trouver un autre type d'architecture.

11.3 LES ARCHITECTURES λ

Le concept d'architecture λ a été mis au point par Nathan Marz lorsqu'il travaillait en 2009 pour la startup BlackType spécialisée dans les solutions d'agrégation de données venant des réseaux sociaux et dans les outils de recherche. L'équipe de Nathan Marz rencontrait des difficultés pour la partie temps réel des traitements. C'est ainsi qu'est né le concept des **architectures λ**. BlackType a ensuite développé un produit appelé Storm qui est basé sur ce concept.

La société BlackType a ensuite été rachetée en 2011 par Twitter qui était intéressé par les capacités temps réel de ses produits. En 2013 Twitter a proposé Storm à Apache qui l'a ensuite intégré dans ses solutions.

Le concept d'architecture λ est donc très récent et les produits qui le mettent en œuvre peu nombreux à ce jour. Il connaît néanmoins un succès croissant et le nombre d'entreprises qui l'utilisent, généralement au travers d'Apache Storm, est assez impressionnant.

L'ambition des architectures λ est de fournir un mécanisme de calcul fonctionnant en temps réel sur un volume arbitraire de données. Pour cela, un nouveau modèle d'architecture de calcul a été conçu. Ce modèle ne remet pas fondamentalement en cause les mécanismes déjà décrits dans cet ouvrage et basés sur l'algorithme Map

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Reduce mais les complète en leur ajoutant une dimension temps réel. La figure 1.1 introduit les trois couches du modèle.

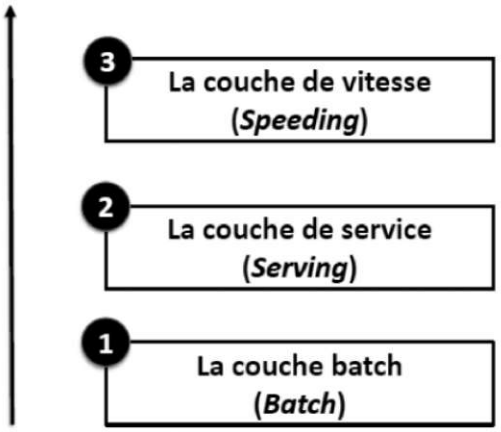


Figure 11.1 — Les trois couches des architectures λ.

Puisqu'il n'est pas possible de calculer une requête dans un délai suffisamment court avec une architecture traditionnelle (comme Hadoop par exemple) les architectures λ utilisent une approche basée sur le pré-calcul des résultats. À chaque fois qu'une nouvelle donnée entre dans le système, les résultats ou des résultats intermédiaires sont recalculés, de telle sorte que lorsqu'une nouvelle requête arrive dans le système la quantité de traitements à effectuer soit la plus faible possible. Si le résultat est déjà pré-calculé, il ne reste plus qu'à le récupérer dans une base et à l'envoyer au demandeur. Si c'est un résultat intermédiaire qui a été stocké, il reste alors à effectuer quelques opérations simples pour calculer le résultat final. Dans les deux cas cela prend très peu de temps et permet d'envoyer la réponse dans un délai très réduit.

Dans les paragraphes suivants, on se basera sur l'exemple fournit par Nathan Marz pour illustrer les architectures λ. Il s'agit d'une solution de Web Analytics, c'est-à-dire d'une application qui analyse l'activité des sites web pour produire des données statistiques sur sa fréquentation. Bien entendu, il s'agit ici de produire les statistiques en temps réel.

11.3.1 La couche batch

La couche batch fonctionne selon le mode traditionnel des applications Big Data, c'est-à-dire sous la forme de batchs qui sont lancés périodiquement, par exemple tous les jours ou toutes les heures.

Dans la plupart des cas, la couche batch repose sur l'utilisation d'outils classiques tels qu'Hadoop par exemple. On attend de cette couche qu'elle soit capable de stocker les données entrantes (dans notre exemple ce sont les logs des sites web) sous une forme brute et d'effectuer périodiquement des calculs sur ces données. La figure 1.2 illustre le fonctionnement de la couche batch.

Copyright © 2015 Dunod.

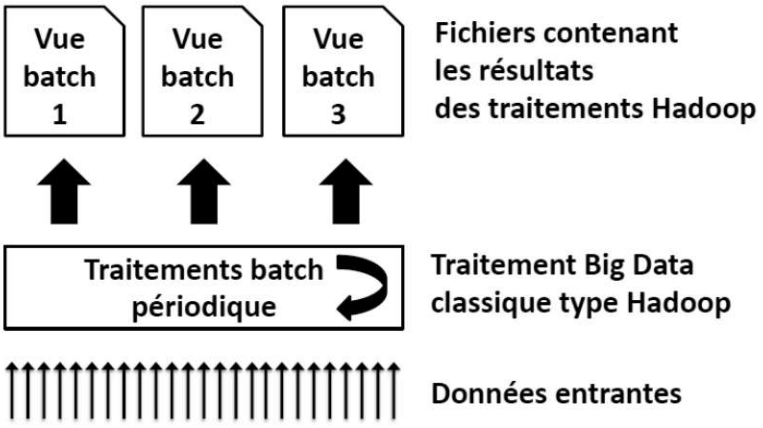


Figure 11.2 — Le fonctionnement de la couche batch.

Tout d'abord, des données entrantes arrivent en continu dans des fichiers de logs. Elles sont la résultante de l'activité des utilisateurs sur les sites web. Ces données s'accumulent dans les fichiers de logs. Périodiquement, par exemple toutes les heures, un batch récupère ces données pour les consolider. Après cette phase de récupération, un traitement est lancé sur une plateforme Hadoop pour produire les statistiques. Ce traitement utilise toutes les données collectées précédemment ainsi que les nouvelles données récupérées lors de la dernière heure. Si le volume de données devient trop important, il suffit alors de rajouter des serveurs de calcul pour réduire le temps de traitements. Ce traitement va produire des résultats, par exemple le nombre de pages vues pour chaque URL. Ces résultats seront stockés dans des fichiers appelés « Vue batch ». Ce mode de fonctionnement est très classique et n'apporte rien de nouveau par rapport à ce qui a été présenté dans les précédents chapitres.

11.3.2 La couche de service

La couche de service a pour rôle de rendre exploitable les résultats calculés par la couche batch. En effet, cette dernière consomme des fichiers de logs bruts et produits des fichiers de statistique. Il s'agit maintenant de les charger dans une base de données qui permettra d'effectuer des requêtes de façon performante. Pour cela, il faut choisir une base de données avec les caractéristiques suivantes :

- D'excellentes performances en lecture.
- Une capacité d'alimentation par batch.

Il n'est en effet pas nécessaire que la base de données retenue propose des opérations d'écriture, car les seules opérations qui s'effectueront seront de type lecture. Elle doit en revanche permettre une alimentation d'un seul tenant par batch d'importation des données.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

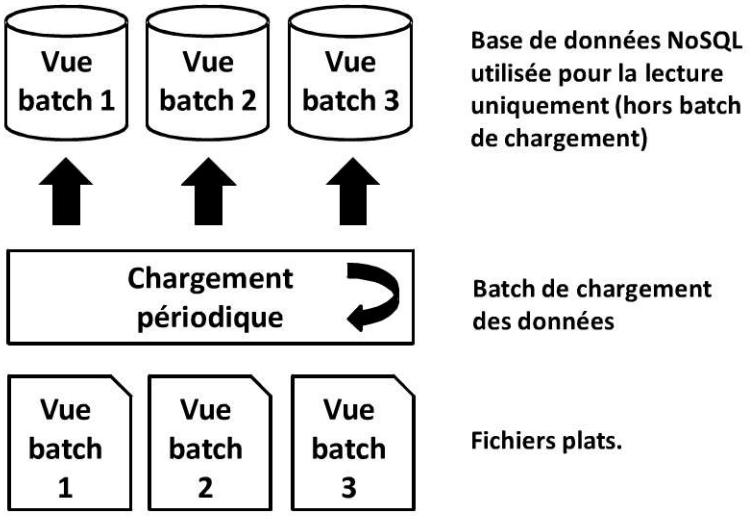


Figure 11.3 — Le fonctionnement de la couche de service.

La performance en lecture sera obtenue par un mécanisme classique d'indexation des données. La base n'étant pas utilisée en écriture, le mécanisme d'indexation ne posera aucun problème de performance. Le créateur des architectures λ d'Apache Storm a également développé une base de données répondant spécifiquement aux besoins de la couche de service. Il s'agit du projet ElephantDB disponible en open source sur la plateforme GitHub (https://github.com/nathanmarz/elephantdb) qui est une base de données très simple mais robuste et scalable.

11.3.3 La couche de vitesse

La couche de vitesse est chargée de traiter les nouvelles données au fur et à mesure de leur arrivée. Cela signifie que cette couche ne fonctionne pas en mode batch mais en continu. Chaque nouvelle donnée qui arrive est immédiatement traitée contrairement à la couche batch qui ne tourne que périodiquement et qui laisse donc s'accumuler les nouvelles données avant de les traiter. La couche de vitesse réalise alors les mêmes traitements que ceux réalisés par la couche batch, mais elle le fait unitairement pour chaque nouvelle donnée entrante. À l'issue de chaque traitement, la couche de vitesse met à jour sa base de données. La figure 11.4 illustre le fonctionnement de la couche de vitesse.

La couche de vitesse met à jour en continu sa base de données. Cette dernière contient donc les résultats des traitements pour les données reçues récemment. À chaque déclenchement de la couche batch, les données de la couche de vitesse sont effacées puisque ces dernières sont désormais prises en compte par le dernier traitement batch. La taille de la base de données de la couche de vitesse est donc nécessairement assez réduite par rapport à celle de la couche de service puisqu'elle ne contient que les résultats des traitements pour les données reçues depuis la dernière exécution de la couche batch.

Copyright © 2015 Dunod.

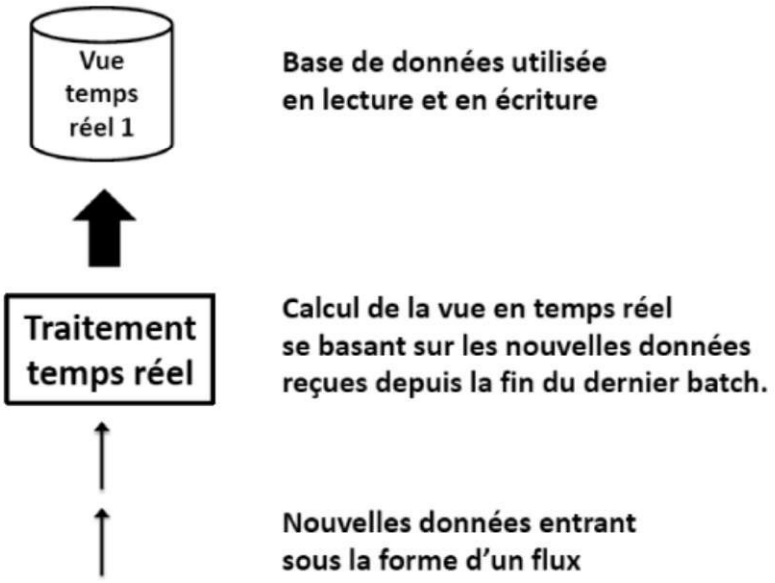


Figure 11.4 — Le fonctionnement de la couche de vitesse.

11.3.4 La fusion

Lorsqu'une requête arrive dans le système, il est nécessaire pour y répondre d'interroger à la fois la base de données de la couche de service qui contient les résultats des traitements sur tout l'historique de données, et la base de données de la couche vitesse qui contient quant à elle les résultats des traitements sur les données récentes qui n'ont pas encore été prises en compte par la couche batch. La figure 11.5 illustre ce mécanisme.

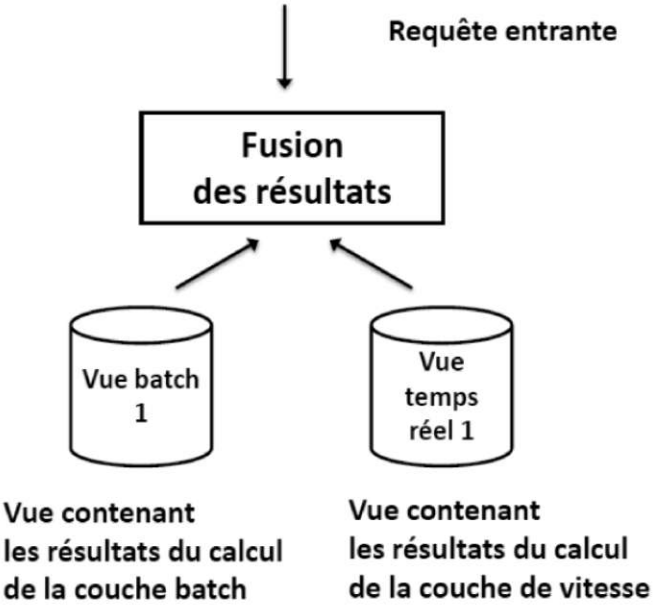


Figure 11.5 — La fusion des résultats contenus dans les couches de service et de vitesse.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Puisque les données sont stockées dans deux bases différentes, il est nécessaire de réaliser la fusion de résultats avant d'envoyer une réponse aux requêtes. Ce traitement peut être plus ou moins complexe en fonction de la nature des opérations. Dans l'exemple proposé, ce traitement est relativement simple. Il suffit dans ce cas d'additionner pour une URL donnée les fréquentations (hits) contenues dans la vue batch et celles contenues dans la vue temps réel.

11.3.5 Les architectures λ en synthèse

Une architecture λ est constituée de trois couches distinctes qui se coordonnent pour mettre à disposition des résultats de traitements quelconques en temps réel. La figure 11.6 récapitule leur fonctionnement.

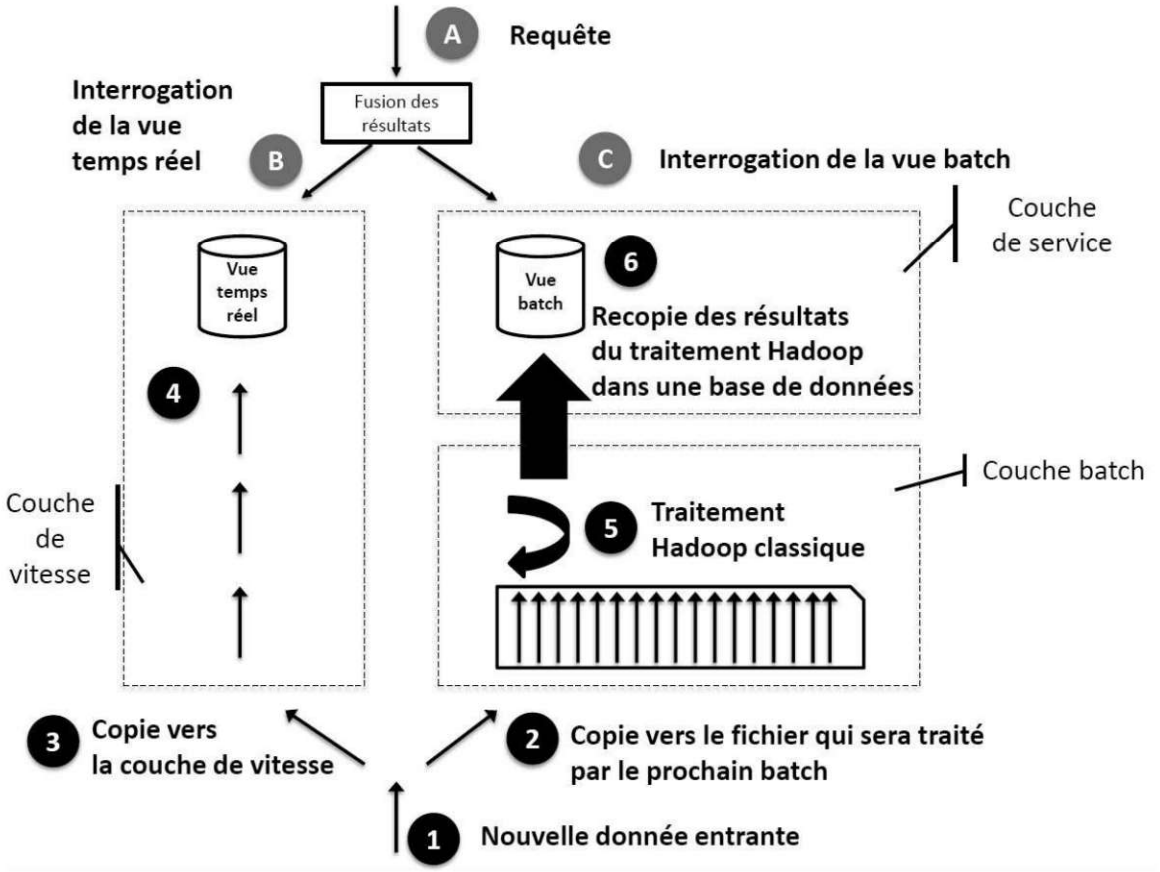


Figure 11.6 — Vue générale du fonctionnement d'une architecture λ.

Lorsqu'une nouvelle donnée arrive dans le système (1) elle est dupliquée pour être transmise à la fois à la couche de vitesse (3) et à la couche batch (2). La couche de vitesse fait alors un pré-calcul sur cette donnée et la stocke (4) dans sa base de données. De son côté, la couche batch accumule les données entrantes dans un fichier et lance périodiquement un traitement de recalcul global du stock de données (5), c'est-à-dire s'appliquant aux nouvelles données mais aussi au stock de données anciennes. Cela se fait généralement à l'aide d'une plateforme Hadoop classique. Une fois le recalcul terminé, celles-ci sont injectées (6) dans une base de données optimisée pour la lecture

Copyright © 2015 Dunod.

dans la couche de service et la base de données de la couche de vitesse est purgée. Lorsqu'une requête arrive, il faut interroger les bases de données de la couche vitesse et de la couche de service et fusionner les résultats avant d'envoyer la réponse.

Il est à noter que l'on peut concevoir des solutions basées uniquement sur la couche de vitesse et donc sans les couches batch et service. Cela correspond aux situations particulières où le stock de données est de petite taille et où l'historisation des données n'a pas d'importance soit parce que les données sont stockées dans un autre système, soit parce que la durée de vie des données (c'est-à-dire leur intérêt pour le business) est particulièrement courte.

En résumé

Ce chapitre a présenté les cas d'utilisation du Big Data temps réel ainsi que les principes des architectures λ qui sont utilisés pour y répondre. On verra dans le chapitre suivant comment fonctionne précisément la couche de vitesse avec la plateforme Apache Storm.

Copyright © 2015 Dunod.

Copyright © 2015 Dunod.

12

Apache Storm

Objectif

Ce chapitre a pour objectif d'introduire la solution de traitement Big Data temps réel Apache Storm et de montrer un exemple de mise en œuvre sur un cas simple de comptage. La présentation se limitera ici aux concepts mis en œuvre par Storm sans rentrer dans les détails tels que la gestion des flux, le paramétrage ou les techniques de programmation associées.

12.1 QU'EST-CE QUE STORM ?

Apache Storm est un framework destiné au traitement de données en temps réel développé par Nathan Marz de la start-up BlackType. La société a ensuite été rachetée par Twitter qui l'a ensuite publié en open source sous licence Apache. Storm est un framework qui permet d'organiser les traitements sur des flux de données. On entend ici par flux, des sources de données qui arrivent en continu au fil de l'eau, ce qui les différencie des traitements de type batch qui travaillent sur des blocs de données bien délimités.

Un flux de données n'a donc par définition pas de taille fixe, puisqu'à chaque instant arrivent de nouveaux blocs de données. Storm propose une architecture pour traiter chacun de ces blocs de données dès leur arrivée dans le système. L'exemple le plus connu d'utilisation de Storm est la détermination des sujets de conversion les plus mentionnés (*trending topics*) dans des messages sur la plateforme Twitter. Ces sujets de conversation sont disponibles ici : https://mobile.twitter.com/trends.

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

Il est à noter que Storm est un framework de traitement de données et qu'il ne fournit pas de solutions de transports de données. Il s'intègre cependant avec n'importe quel middleware de messaging, par exemple Apache Kafka ou RabbitMQ. De la même façon, Storm ne fournit pas de solution particulière pour stocker les résultats des calculs. Une fois ceux-ci effectués, il faudra utiliser un autre outil pour stocker les résultats. Storm est donc simplement un composant de calcul que l'on insère dans une chaîne de traitement entre la source d'information en amont et un système de stockage en aval.

12.2 POSITIONNEMENT ET INTÉRÊT DANS LES ARCHITECTURES λ

On a présenté dans le chapitre précédent la notion d'architecture λ qui est structurée autour d'un modèle en trois couches. Apache Storm est un framework qui se situe dans la troisième couche, c'est-à-dire la couche de vitesse. La figure 12.1 positionne Storm par rapport aux autres couches de l'architecture.

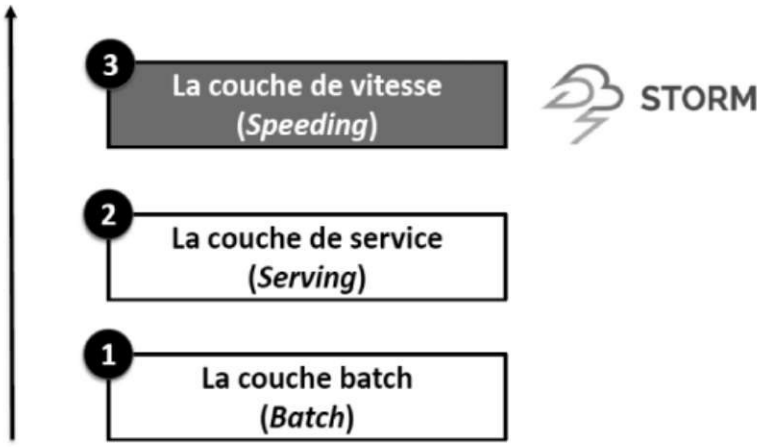


Figure 12.1 — Le positionnement d'Apache Storm dans les architectures λ.

Apache Storm dispose de toutes les fonctionnalités nécessaires pour implémenter la couche de vitesse, hors middleware de transport pour récupérer les données en entrée et système de stockage en sortie.

Il est bien sûr possible d'implémenter soi-même la couche de vitesse mais Apache Storm permet de gagner beaucoup de temps en fournissant tous les mécanismes de base tels que la distribution des calculs sur plusieurs nœuds, la garantie que chaque bloc de données ne sera traité qu'une seule fois, ou encore la scalabilité. De plus, bien que Storm utilise la plateforme Java, il est tout à fait possible d'utiliser d'autres langages pour effectuer les traitements.

Copyright © 2015 Dunod.

12.3 PRINCIPES DE FONCTIONNEMENT

12.3.1 La notion de tuple

Le premier concept utilisé par Storm est la notion de « tuple ». Dans Storm chaque paquet de données entrant dans le système est appelé tuple. Un tuple est une liste ordonnée de valeurs. Ces valeurs peuvent être de n'importe quel type, entier, flottant, chaîne de caractères, tableau ou même objet. La seule contrainte est que Storm soit capable de sérialiser/désérialiser toutes les valeurs d'un tuple. La figure 12.2 présente un tuple composé de deux chaînes de caractères et de deux entiers.

```
("user", "http://examples.com/test ", 200, 50)
```

Figure 12.2 — La notion de tuple.

Avec Storm, l'envoi d'un tuple de données est appelé une émission. Dans la pratique l'émetteur du tuple devra également donner un nom à chacune des données contenues dans le tuple.

12.3.2 La notion de stream

Le second concept introduit par Storm est la notion de flux, aussi appelé « stream ». Un stream se définit comme une suite non limitée de tuples. La figure 12.3 présente un stream. Les streams sont utilisés pour relier les différents nœuds qui composent une solution construite avec Storm.

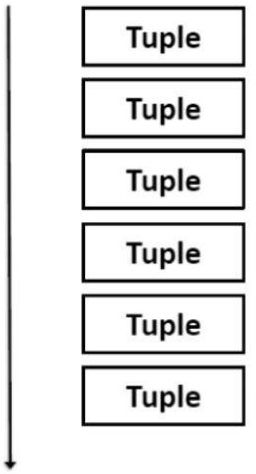


Figure 12.3 — La notion de stream (flux).

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

12.3.3 La notion de spout

Dans Storm, le « spout » est l'élément par lequel les données entrent dans le système. Le rôle d'un spout est d'alimenter un stream avec des tuples construits à partir d'une source externe. Celle-ci peut être de n'importe quel type :

- une base de données,
- des fichiers dans un répertoire,
- un middleware orienté message (RabbitMQ, JMS...),
- un socket réseau...

Un spout n'est pas nécessairement limité à l'alimentation d'un seul stream et dans certains cas il est possible que le spout en alimente plusieurs en même temps. Il est à noter que le spout n'est pas censé faire des traitements autres que ceux strictement nécessaires au chargement des données et à leur retransmission sous la forme de tuples. La figure 12.4 illustre le fonctionnement d'un spout qui alimente deux streams.

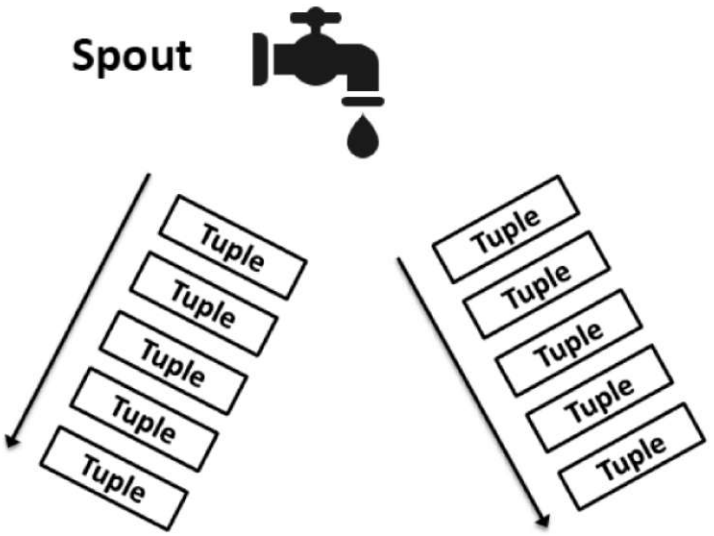


Figure 12.4 — La notion de spout.

12.3.4 La notion de bolt

Le « bolt » est le composant de base pour faire des traitements élémentaires avec Storm. Il est possible de faire n'importe quel type de traitement avec un bolt, par exemple filtrer des données, faire appel à des web services externes, agréger des contenus ou encore se connecter à une base de données.

Un bolt reçoit des tuples depuis un ou plusieurs streams, effectue un traitement particulier puis renvoie le résultat sous la forme de tuples dans un autre stream. Dans certains cas, comme pour le spout, il est possible que le bolt émette des tuples en sortie sur plusieurs streams. Dans d'autres cas, le bolt ne renvoie aucun résultat. C'est en particulier le cas des bolts en terminaison dont le rôle est de faire sortir les résultats de Storm, par exemple pour les sauvegarder dans une base de données.

Copyright © 2015 Dunod.

La figure 12.5 montre un bolt alimenté par deux streams différents et qui alimente en sortie un troisième stream.

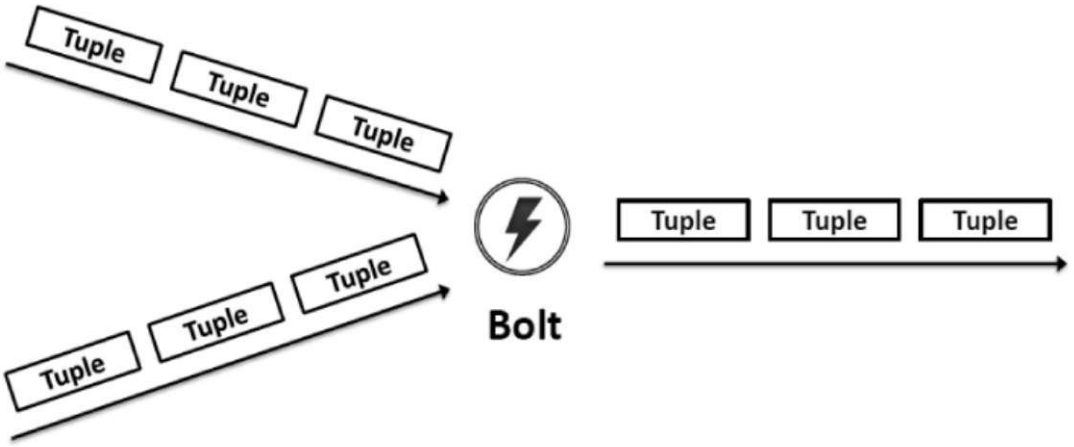


Figure 12.5 — La notion de bolt.

12.3.5 La notion de topologie

Dans Storm, la notion de topologie correspond au graphe de la logique de circulation des données dans le système. C'est l'équivalent de la notion de job dans la plateforme Hadoop, à ceci près que dans Storm la topologie n'a jamais de fin prévue puisqu'elle tourne en continu. La topologie comprend l'ensemble des spouts et bolts utilisés ainsi que les streams qui les relient.

Il n'y a pas d'outil graphique pour construire une topologie Storm, c'est donc programmatiquement que le développeur la construit en utilisant les classes Java mises à disposition par le framework.

La figure 12.6 illustre une topologie Storm constituée de deux spouts, cinq bolts et six streams. On voit clairement que les deux bolts de droite sont des terminaisons car ils ne transmettent pas d'information vers des streams. Il est donc probable qu'ils exportent des données dans des systèmes externes tels qu'une base de données ou un middleware orienté message.

La topologie décrit la logique de circulation des flux de données indépendamment des choix de déploiement. En effet, Storm permet ensuite de définir le nombre d'instances de chaque nœud (les spouts et les bolts) ainsi que les différents serveurs sur lesquels ils doivent être répartis pour bien monter en charge. Il existe aussi des mécanismes permettant de contrôler finement la façon dont les nœuds communiquent entre eux. Comme pour la définition de la topologie, ceci est réalisé principalement par programmation et paramétrage.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

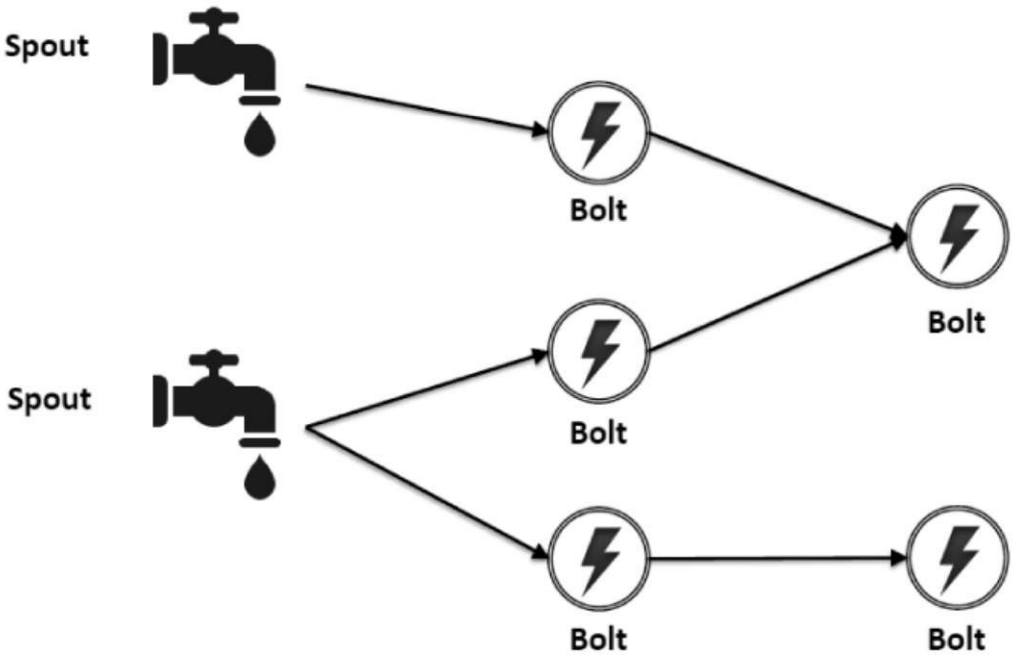


Figure 12.6 — La notion de topologie.

12.4 UN EXEMPLE TRÈS SIMPLE

Pour illustrer la mise en œuvre de Storm, voici un exemple très simple d'utilisation dont l'objectif est de réaliser le comptage des mots à l'intérieur de fichiers. Pour commencer on définit un spout qui sera chargé de lire les fichiers dans un répertoire particulier et de réémettre chaque ligne sous la forme d'un stream. Ce stream est ensuite envoyé vers un bolt dont le rôle est de découper chaque ligne reçue en mots. Il renvoie ensuite chaque mot vers un autre bolt qui est chargé quant à lui de maintenir à jour dans une base de données le comptage de chacun des mots trouvés dans les fichiers. La figure 12.7 illustre la topologie mise en œuvre dans cet exemple.

Cet exemple est évidemment très simple et n'est pas représentatif de la complexité des architectures qui sont déployées en production dans le monde réel. Il permet néanmoins d'appréhender facilement la mise en œuvre des concepts de base.

Apache Storm n'en reste pas moins un outil assez complexe à déployer et à administrer et qui nécessite une bonne compréhension des middlewares sur lesquels il s'appuie, en particulier Apache Zookeeper qui est utilisé pour la synchronisation des différents nœuds d'un cluster.

Copyright © 2015 Dunod.

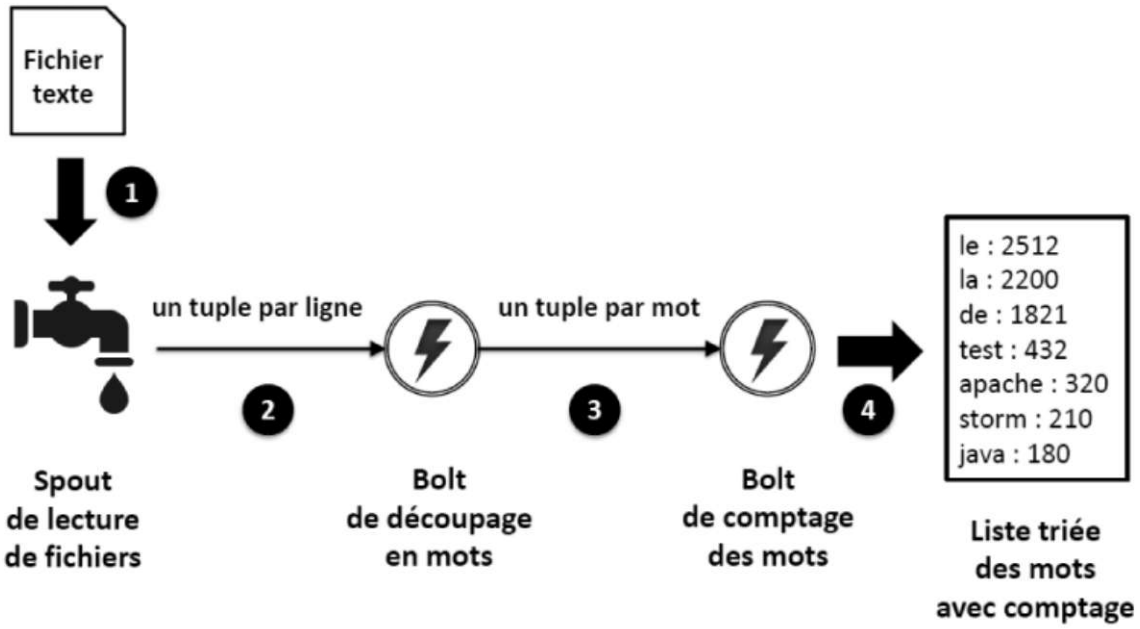


Figure 12.7 — Un exemple de topologie Storm pour le comptage des mots dans un flux de fichiers textes.

En résumé

Ce chapitre a présenté l'outil Apache Storm, une solution open source permettant le traitement de données sous forme de flux avec des faibles latences. Storm est utilisé en particulier pour implémenter la couche de vitesse dans les architectures λ.

Copyright © 2015 Dunod.
© Dunod – Toute reproduction non autorisée est un délit.

Copyright © 2015 Dunod.

Index

Symboles

3V 15

A

ACP (analyse en composantes principales) 133
agrégation 28, 100
algèbre des relations 28
algorithme
  d'apprentissage 85
  d'apprentissage statistiques 11
  de classification 182
  de clustering 183
  des *k*-moyennes 125
  linéaire 118
ALL 45
analyse
  en composantes principales (ACP) 133
  factorielle 133
apprentissage 110
  hors ligne 119
  non supervisé 117
  supervisé 117
arbres de décision 127
architecture λ 199
atomicité 174
attributs visuels 158
AUC (*Area Under the Curve*) 138

B

bases de données
  NoSQL 36
  orientées agrégats (BDOA) 37
  orientées document (BDOD) 40
  orientées graphes 46
  orientées graphes (BDOG) 38
  relationnelles (SGBDR) 28
BDOA (bases de données orientées agrégats) 37
BDOD (bases de données orientées document) 40
BDOG (bases de données orientées graphes) 38
Big Data 3
  définition 7
  exemples 8
blob 38
boîte à moustache 96, 153
bolt 210
bootstrap 130
box plot 96, 153
bruit $\varepsilon(x)$ 109

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

C

capacity planners 10
classification naïve bayésienne 123
cloud computing 14
Cloudera 168, 177
clustering 117
clusters 126
Codd E. F. 28
cohérence à terme (*eventual consistency*) 33
cohérence (*consistency*) 33
cohortes 101
combiner 65
community manager 22
connecteurs 168
consistance forte 40
contextualisation 150
corrélation 149
    fictive 115
couche
    batch 200
    de service 201
    de vitesse 202
courbe
    de densité 97, 154
    de sensibilité 157
    ROC 137, 155, 157
crawling 91
critères de segmentation 127

D

data lab 79
data scientist 71
datamasse 3
DataNode 171
dataviz 150
dataviz 147
détecteurs de désir 24
diagramme
    de dispersion 140, 154
    en barres 152
    en boîtes 153
disponibilité (*availability*) 33
distribution Hadoop 168
données
    d'entraînement 113
    de test 113
données personnelles 23
Dremel 181
Drill 67, 180

E

écart-type 96
ECV (entrepôt clé-valeur) 38
élagage 129
Elastic MapReduce 168
Elastic MapReduce (EMR) 179
émission 209
EMR (*Elastic MapReduce*) 179
ensemble d'apprentissage 110
entraînement 110
entrepôt clé-valeur (ECV) 38
exécution spéculative 64
exemple étiqueté 117
extraction 100

F

FA (*factor analysis*) 133
facteur de réplication 39
factor analysis (FA) 133
familles de colonnes 42
feature engineering 84, 139, 142
filtrage collaboratif 183
fléau de la dimension 115
Flume 176
fonction
    de prédiction 109
    F(x) 109
forêts aléatoires 130
formats d'échange 92
forward selection 133, 144
fuzzy join 103

Copyright © 2015 Dunod.

G

géocodage 102
grammaires formelles 161
grid-search 87

H

Hadoop 52, 62, 168, 199
    distribution 168
Hadoop Distributed File System (HDFS) 171
HANA 67
HBase 173
HDFS 63
HDFS (*Hadoop Distributed File System*) 171
heartbeat call 63
histogramme 97
Hive 56, 169, 176
HiveQL 176
Hortonworks 168, 177

I

Impala 180
industrialisation 87
interprétabilité 85

J

job MapReduce 62
jobtracker 62, 65, 172
jointure 28
    approximative 103
    exacte 102
Jubatus 116

K

k plus proches voisins 121
Karmasphere 170
kernel trick 132
KNN (*K Nearest Neighbors*) 121

L

langage déclaratif 29
logs 90

M

machine learning 107, 169
machine learning 67, 73, 151
machines à vecteurs de support 131
Mahout 67, 116, 182
map 53, 61
mapper 53
MapR 168, 178
MapReduce 52, 169, 199
marges souples 132
matrice
    creuse 60
    de confusion 156
médiane 96, 153
méthode
    d'ensemble 130
    des moindres carrés 120
mismatch d'impédance 31
modèle
    de classification 118
    de cohérence à terme (*eventual consistency*) 39
    de régression 118
    géométrique 119
    non paramétrique 118
    paramétrique 85, 118
    probabiliste 119
mosaic plot 152
moyenne arithmétique 96

N

Naïve Bayes Classifier 123
NameNode 171
No Free Lunch Theorem 117
nœud
    maître 171
    terminal 127

© Dunod – Toute reproduction non autorisée est un délit.
Copyright © 2015 Dunod.

NoSQL 11, 27, 48, 93
bases de données 36
noyau 132
gaussien 132
polynomial 132

O

objets connectés 90
ONE 45
Oozie 176
open data 4, 91
outliers 121

P

Pandas 169
parallélisation 11
partition horizontale 40
passage à l'échelle 74, 87
pattern de répartition des tâches 52
persistance 174
Pig 56, 175
Pig Latin 175
POC (*Proof Of Concept*) 14
ponctualité 175
prédiction 108
projection 28
aléatoire 134
prunning 129
Python 169

Q

quantile 96, 153
quartet d'Anscombe 95, 148
quartile 96, 153
quorum
d'écriture 39
de lecture 39
QUORUM 45

R

R 184
recherche contextuelle 8
reduce 53, 61
reducers 53
réduction dimensionnelle 84, 115, 143, 183
régression
linéaire 120
logistique 118, 124
régularisation 121
réseaux
bayésiens 123
sociaux 91
résistance au morcellement (*partition tolerance*) 33
résolution d'adresse IP 102
RHadoop 184

S

scalabilité 48
scaling 74
scatterplots 140
ScikitLearn 169
score 118
sélection 28
séparateurs à vaste marge (SVM) 131
séparation 100
SGBDR (bases de données relationnelles) 28
sharding 38, 40
shuffle 53, 64
Spark 67, 183
split 53, 64
spout 210
Sqoop 177
Storm 199, 207
stream 209
structures de données 92
super-colonnes 42

Copyright © 2015 Dunod.

surapprentissage 113, 136
SVM (séparateurs à vaste marge) 131

T

tableaux croisés dynamiques 96, 97
tâches
    map 62
    reduce 62
tasktracker 62, 65, 172
temps réel 197
théorème CAP 33
transaction 29
transformation 100
trending topics 207
tuple 209

U

unicité de l'image 174
union 28

V

valeurs aberrantes 121
validation croisée 114
variable
    cible 81, 109
    latente 133
    nominale 118
    ordinale 118
    prédictive 81, 84
    prédictives 109
    qualitative 118
    quantitative 118
variance 96, 149
variété 15, 16
vectoriser 9
vélocité 15, 16
vie privée 23
visualisation des données 147
volume 15, 16
VRM (*Vendor Relationship Management*) 24

W

watch 174
Web Analytics 200

Y

YARN 172

Z

znodes 174
ZooKeeper 173