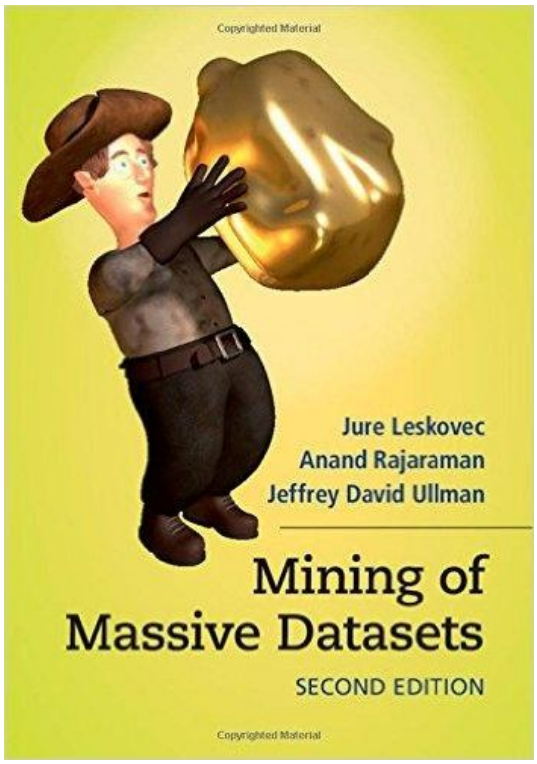
Mining of Massive Datasets (2nd Edition)

Jure Leskovec, Anand Rajaraman, Jeff Ullman

Cambridge University Press, Novembre 2014.

http://www.mmds.org/



« Mining of Massive Datasets », littéralement « Fouille de données massives », s'inscrit dans l'air du temps. Le contexte, maintes fois évoqué, est bien connu aujourd'hui : la profusion des données et la multiplication des sources, exacerbées par les outils de communication et notre mode de vie, induisent de nouveaux défis et opportunités pour le Data Mining.

Par rapport aux très nombreuses références qui existent, le livre de Leskovec, Rajaraman et Ullman présente une double particularité : il est basé sur des enseignements dispensés à l'Université de Stanford (CS246 : Mining Massive Datasets et CS354A : Data Mining), c'est dire s'il a fait ses preuves ; le livre au format PDF ainsi que tout le matériel pédagogique associé (les diaporamas relatifs à chaque chapitre en Powerpoint et PDF) sont librement accessibles sur le web (http://www.mmds.org/). C'est Byzance ! Sachant par ailleurs que l'ouvrage imprimé est disponible dans les librairies (mais pas gratuitement).

Plusieurs aspects m'ont incité à le lire attentivement : le contexte « données massives », qui s'inscrit dans la déferlante « big data », est aujourd'hui incontournable, et le sera de plus en plus à l'avenir ; des chapitres abordent des thèmes qui m'intéressent particulièrement pour faire évoluer mes enseignements à l'Université (ex. systèmes de recommandation, analyse des réseaux sociaux, etc.) ; le discours adopté est raisonnablement accessible, on nous fait grâce de l'arsenal mathématique qui rassure souvent ceux qui ne comprennent pas très bien de quoi ils parlent, et qui s'avère rédhibitoire pour nos étudiants (bien sûr qu'il en faut, mais il n'y a pas que ça, il nous revient - à nous enseignants - de trouver la bonne mesure compte tenu du public auquel nous nous adressons) ; enfin, par rapport à des supports que nous pourrions glaner ici ou là sur le web, un ouvrage présente toujours une structure pédagogique qui permet de mettre en perspective les thèmes abordés.

Les auteurs se positionnent dans le cadre de la fouille sur des données tellement volumineuses qu'elles ne tiennent pas en mémoire vive (Préface, page ix). En réalité, ils abordent différents sujets « à la mode » qui s'enracinent dans le traitement des données en provenance du web au sens large, cadre dans lequel la volumétrie joue effectivement un rôle important. Ainsi, certains chapitres y font explicitement référence (ex. parallélisation des algorithmes via MapReduce, méthodes de machine learning adaptées à l'appréhension des grandes bases, etc), alors que d'autres prennent leur origine dans les opportunités que nous offre les nouvelles pratiques du web (ex. Page Rank, réseaux sociaux, systèmes de recommandation, etc.). La cohérence de l'ensemble n'est pas toujours évidente. Mais a contrario, la lecture indépendante des chapitres est possible.

Notons une bonne habitude des auteurs : un résumé décrivant les principales notions abordées est proposé à la fin de chaque chapitre. Il est assez bien fait. Un lecteur pressé pourrait se contenter de le lire. Il m'a énormément servi en tous les cas pour vérifier que j'avais bien saisi la teneur des thèmes développés.

1. Data Mining

Le premier chapitre introduit les principales notions du data mining. Il consiste à « extraire des modèles à partir des données ». Plusieurs aspects sont abordés (statistique, machine learning), des concepts clés sont présentés : le danger de l'artefact statistique ; l'indicateur TF.IDF pour indiquer la pertinence des termes dans le *text mining* ; les fonctions de

hachage permettant d'associer un objet quelconque à une valeur entière ; les index permettant de retrouver très rapidement des objets dans une base ; etc.

Pris individuellement, ces thèmes sont intrinsèquement intéressants. Mais regroupés dans un même chapitre, on saute un peu du coq à l'âne d'une section à l'autre, rendant le tout un peu confus. Je me suis rendu compte par la suite qu'ils revenaient à plusieurs reprises dans l'ouvrage. Si on ne les maîtrise pas, il faut faire l'effort de comprendre au moins les idées qui les sous-tendent.

La lecture attentive de la section 1.4 (page 15 et suivantes) qui décrit les thèmes abordés dans chaque chapitre est conseillée pour disposer d'une vision globale de l'ouvrage.

2. MapReduce and the New Software Stack

Le chapitre 2 est consacré à la parallélisation des algorithmes de data mining via le paradigme MapReduce dont le principe est bien explicité (Figures 2.2 et 2.3). L'augmentation de la volumétrie ne peut pas être appréhendée par une amélioration des performances des machines. Il faut passer à l'informatique distribuée. Les auteurs décrivent les concepts clés du domaine : l'organisation des machines en clusters et nœuds ; les systèmes de fichiers distribués, dont HDFS de Hadoop en est une déclinaison ; la gestion des défaillances ; les coûts de transfert des données d'une machine à l'autre.

« MapReduce ne résout pas tous les problèmes, et tous les problèmes ne profitent pas (de la même manière) de MapReduce » (page 28). Cette remarque est particulièrement édifiante. À mon sens, MapReduce permet déjà de simplifier la programmation parallèle qui, nativement, requiert de très bonnes compétences en informatique. La synchronisation des tâches, l'accès aux ressources partagées, sont autant de problèmes que l'on peut évacuer en se plaçant dans le cadre de MapReduce. Il permet aussi de gérer très simplement (on n'a pas à s'en occuper, il n'y a pas plus simple) la répartition des tâches sur des machines distantes. Il n'en reste pas moins que certains problèmes sont plus faciles à mettre dans le moule MapReduce que d'autres. L'ouvrage nous propose plusieurs exemples (ex. calcul matriciel, opérations d'algèbre relationnel, etc.), avec en filigrane les questions clés : que doit-on mettre dans la partie MAP ? Et dans la partie REDUCE ?

La bibliographie commentée permettra au lecteur curieux de se documenter sur les très nombreuses technologies big data qui nous submergent d'acronymes plus ou moins abscons (section 2.8).

3. Finding Similar Items

Ce chapitre s'intéresse à la similarité sous l'angle de l'intersection entre 2 éléments. L'indice de Jaccard est mis en lumière. Les applications sont nombreuses. L'idée sous-jacente repose sur la recherche de similarité, notamment entres documents textuels, pour la détection de plagiat par exemple, ou encore la recherche de sources communes (ex. des brèves que l'on retrouve sur plusieurs sites de journaux en ligne ont pour origine la même dépêche Afp).

Les auteurs s'intéressent ensuite au « shingling » de documents (Section 3.2). Je ne connaissais pas le terme. Mais en lisant attentivement la section, je me suis rendu compte qu'il s'agissait simplement d'identifier des termes (k-shingles) sous l'angle de « n-gramme » définis sur les lettres. Un document serait alors caractérisé par les termes qui lui sont associés. Nous avons là une version simplifiée de la construction d'une matrice documents-termes fondamentale en *text mining*. À la différence que les termes ne correspondent pas à des mots mais à des suites de caractères de longueur « k ». La similarité entre deux documents peut être mesurée à l'aide de l'indice de Jaccard, explicitant le degré d'intersection entre les termes associés aux documents. Jusque-là, rien de plus classique.

Mais on peut aller plus loin en s'appuyant sur les fonctions de hachage (section 3.2.3 - « Hashing Shingles »). Chaque terme (une suite de k-caractères) est transformé en une valeur entière par la fonction de hachage. Ainsi, un document peut être identifié par un ensemble d'entiers. Il en résulte une meilleure efficacité calculatoire et une réduction de l'espace occupée en mémoire vive.

Malgré tout, surtout si le nombre de documents est très important, non seulement l'espace mémoire occupé par la représentation documents-termes peut s'avérer rédhibitoire, mais le calcul des similarités entre les documents nécessaires à certains algorithmes de *data mining* s'avèrerait infaisable. Dans les sections qui suivent, les auteurs présentent des approches pour dépasser ces inconvénients. La première (Section 3.3, « Similarity-Preserving Summaries of Sets ») consiste à remplacer l'ensemble de termes par une « signature » de taille réduite en occupation mémoire (MinHash en l'occurrence). Un exemple didactique

permet de retracer la démarche. À la sortie, nous disposons d'une information permettant d'estimer avec plus ou moins de précision, selon la taille de la signature choisie, l'indice de Jaccard entre les documents. La seconde (Section 3.4, « Locality-Sensitive Hashing for Documents ») permet de retrouver les documents ayant une forte similarité, sans avoir à calculer les indices de tous les documents pris deux à deux. L'efficacité du système repose sur la capacité de la fonction de hachage à placer dans les mêmes paquets (*buckets*) les items « proches ». Un peu comme le ferait un algorithme de clustering. De fait, ne seront calculés que les indices de Jaccard des documents voisins. Le processus complet est résumé dans la section 3.4.3. Il rend possible le traitement d'un très grand nombre de documents, impossible à réaliser si nous passions par une matrice documents-termes classique.

L'approche est approfondie dans les sections suivantes où l'on explore les autres familles de fonctions (autres que MinHash) qui permettent de produire des paires candidates proches, qui pourraient s'étendre à d'autres mesures de distance que celle de Jaccard (définie par 1 - similarité de Jaccard) telle que la distance euclidienne, la distance cosinus, etc.

La lecture de ce chapitre est quand même un peu ardue. Il faut le reconnaître. Pour ma part, j'ai commencé à comprendre la démarche et les calculs lorsque que j'ai essayé de reproduire les exemples, et réaliser les exercices qui émaillent le chapitre.

4. Mining Data Streams

Le chapitre 4 est consacré à la fouille de flots de données. Leur principale particularité est qu'elles arrivent en continu, avec une grande rapidité, il n'est pas possible de tous les stocker pour les analyser et, de toute manière, leur obsolescence est rapide. De fait, les données sont perdues si elles ne sont pas exploitées immédiatement.

Les auteurs évoquent les requêtes sur un moteur de recherche ou les clics sur un site web à titre d'exemple. Pour ma part, je donne l'exemple de l'indicateur d'autonomie affiché par les ordinateurs de bord des véhicules lorsque j'aborde ce sujet en cours. Il y a bien une forme de modélisation puisqu'une projection est réalisée, elle est mise à jour en fonction de notre mode de conduite, en donnant une importance de moins en moins forte aux données anciennes. Nous avons là une illustration du principe du lissage exponentiel, bien connu en traitement des séries temporelles, que nos auteurs présentent dans un cadre différent dans la section 4.7, « Decaying Window ».

Le chapitre commence par une présentation de la particularité des flots de données par rapport à une base de données classique. Les flux d'informations peuvent être stockés dans des bases d'archivage (*archival store*), pour mémoire ou pour un traitement a posteriori. En revanche, espérer pouvoir effectuer des requêtes sur cette base pour des traitements en temps réel (ou quasi temps réel) n'est pas envisageable. Une base de travail (*working store*) peut être mise en place. Elle est de capacité réduite, les données sont effacées au fur et à mesure. Le support peut être la mémoire vive si la rapidité est un enjeu crucial. Elle constituera la source d'information utilisée pour la modélisation à la volée.

Deux approches sont souvent mises en avant : maintenir un résumé des données que l'on met à jour au fur et à mesure que de nouvelles informations arrivent ; ou appliquer les traitements sur une fenêtre glissante incluant les versions les plus récentes des données.

Une remarque importante a retenu mon attention : du fait des caractéristiques des données (flots) et des contraintes sur les traitements (données en mémoire vive de taille limitée, vitesse d'analyse cruciale), il est souvent plus efficace d'obtenir une réponse approchée mais réalisable plutôt que de prétendre à une réponse exacte impossible à fournir compte tenu du cadre imposé (page 135).

Différents problèmes relatifs au traitement des flots de données sont ensuite présentés : l'échantillonnage dans les flots de données (section 4.2), avec comme enjeu fort la représentativité de l'échantillon extrait ; le filtrage des flots (section 4.3), qui est simple si un calcul permet de définir la condition de sélection, il devient compliqué si cette dernière nécessite une requête dans une base de données ; le comptage d'éléments distincts dans un flux (section 4.4), par exemple pour obtenir la liste des numéros IP des machines accédant à un site, les fonctions de hachage peuvent être de nouveau mis à contribution dans ce contexte ; l'estimation des moments (section 4.5) qui passe par le comptage des éléments ; le comptage d'apparition de 1 dans un flot de bits (section 4.6).

On ne parle pas vraiment des algorithmes de data mining dans ce chapitre (un peu quand même avec les comptages). Les sujets abordés mettent surtout en lumière les défis posés par le traitement des flots de données.

5. Link Analysis

Ce chapitre se focalise sur le mode de calcul du « PageRank » introduit par Google. Le préambule est captivant (section 5.1.1). On nous y explique que les anciens moteurs de recherche se basaient exclusivement sur le contenu pour mettre en évidence les pages pertinentes par rapport à une requête. Les spammeurs (au sens large ici) ont rapidement tourné à leur avantage le dispositif en truffant leurs pages de mots-clés judicieusement placés, ou encore en recopiant les pages populaires et en les insérant en mode invisible (texte blanc sur fond blanc par exemple) dans leurs propres productions. Google a supplanté ses concurrents en introduisant la notion de *PageRank*, reposant sur deux principales idées : une page intéressante est une page souvent citée, il faut donc tenir compte du nombre de liens qui y font référence ; les termes contenus dans la page sont importants, son auteur peut la manipuler à sa guise, les termes utilisés dans les liens qui y mènent doivent peser également, d'autant plus que l'éventuel spammeur a peu de prise dessus.

Pour comptabiliser l'importance d'une page, le processus de « surfeur aléatoire » est mis en place. Un robot démarre à partir d'une page quelconque. Il saute sur la suivante en choisissant au hasard un lien parmi l'ensemble des références disponibles. Il navigue ainsi sur la totalité du web. Une page souvent visitée sera importante parce que de nombreux liens extérieurs y mènent. Les calculs et la résolution du système sont très bien expliqués sur un exemple de web à 4 pages (section 5.1.2).

Tout serait facile si la structure du web était simple et parfaite. Ce n'est pas le cas bien évidemment. Le reste du chapitre est consacré aux solutions pour dépasser les problèmes potentiels (une page peut ne pas avoir de liens sortants, elle peut être isolée de l'ensemble du web, etc.) ; au calcul efficace du système sur de très grandes dimensionnalités (un web à 4 pages est un exercice de style, on l'a bien compris), sachant que la matrice de transition est très « sparse » (avec beaucoup de zéros, les pages sont peu connectées entres elles finalement) ; à l'introduction d'une pondération tenant compte des thèmes (*topics*) associés aux pages ; aux remèdes à apporter quand des spammeurs essaient de tourner à leur avantage le mécanisme du *PageRank*, notamment en passant par l'authentification des pages.

Le contenu n'est pas oublié. En dernier lieu, il faudra combiner le score de référencement (*PageRank*) avec un score relatif aux informations délivrées par la page. Il paraîtrait

invraisemblable qu'une page ne contenant aucun des mots-clés introduits dans la requête apparaisse dans les premières positions. Mais nous n'avons pas d'information là-dessus (section 5.1.6). La formule sous-jacente est aussi secrète que celle du Coca-Cola visiblement.

6. Frequent Itemsets

Le chapitre suivant traite de la recherche des itemsets fréquents. Le sujet est intrinsèquement intéressant (quels sont les produits qui sont achetés ensemble), mais qui constitue également un préalable indispensable à l'extraction des règles d'association (l'achat de tel produit va entraîner l'achat de tel autre).

Plusieurs notions fondamentales sont introduites : transaction, panier, item, itemset, support, itemset fréquent. L'exemple illustratif est issu du *text mining*. Cela nous change du panier d'achats usuellement utilisé dans ce type d'exposé. Et c'est heureux car les domaines d'application de la technique dépassent largement l'analyse des emplettes de la ménagère de moins de 50 ans (section 6.1.2).

À l'instar de la très grande majorité des références qui traitent du sujet, l'algorithme Apriori d'Agrawal et Sikrant (1994) est disséqué. L'originalité ici est que les auteurs étudient la faisabilité de l'approche sur des grandes bases en s'intéressant à l'accès aux données, au nombre d'accès aux données, et à l'occupation mémoire. Enfin, la dernière section 6.5 « Counting Frequent Items in a Stream » enrichit la partie dévolue au traitement des flots de données (chapitre 4) avec le comptage d'itemsets fréquents dans ce contexte très particulier.

Encore un chapitre sur les itemsets fréquents me suis-je dit en entamant le chapitre. Force est de reconnaître que les auteurs ont su rendre le thème intéressant en se positionnant différemment, en approfondissant des aspects que l'on retrouve peu par ailleurs. Le lecteur informaticien y trouvera clairement son compte.

7. Clustering

Le chapitre 7 est consacré aux algorithmes de classification automatique (apprentissage non supervisé, typologie). L'application de cette famille de méthodes de *machine learning* dans une configuration de très grande volumétrie fait partie des sujets qui m'ont amené à m'intéresser à cet ouvrage. Les auteurs annoncent d'ailleurs la couleur en indiquant

s'intéresser aux très grandes dimensionnalités (en nombre d'observations et/ou de descripteurs), avec des données qui ne tiennent pas en mémoire vive.

La première section rappelle les concepts de la typologie. L'objectif est d'identifier des groupes homogènes de « points » au sens d'un certain critère. L'approche hiérarchique de type CAH (classification ascendante hiérarchique) est détaillée (section 7.2). On sait qu'elle est gourmande en ressources car nécessite le calcul des distances entre les points pris deux à deux. Une section spécifique aux espaces non-euclidiens (section 7.2.4) pose la question de la situation où il n'est pas possible de « résumer » un groupe à l'aide de son barycentre. L'idée de « point(s) représentatif(s) » des groupes est mise en avant. Elle peut se combiner avec des règles alternatives d'agrégation des groupes (saut minimum, saut maximal, distance moyenne, etc.).

La section 7.3 s'attaque à la méthode très populaire des K-Means. La présentation ne diffère guère de ce qu'on peut lire dans d'autres références. Nous ferons une mention particulière pour la variante BFR qui introduit une hypothèse forte d'orthogonalité deux à deux des descripteurs pour pouvoir traiter efficacement les très gros volumes de données. Un cluster peut être ainsi résumé avec seulement 3 informations : le nombre d'objets, la somme des valeurs par variable, la somme des carrés des valeurs. Nous pouvons en déduire les barycentres et les inerties des groupes.

Les sections suivantes décrivent des méthodes moins connues. CURE est détaillée dans la section 7.4. La méthode ne fait pas d'hypothèses implicites sur la forme des clusters. Sa faisabilité sur de très grandes bases repose sur la possibilité de travailler à partir d'échantillons qui tiennent en mémoire vive. GRGPF (du nom de ses auteurs) est adaptée aux espaces non euclidiens (section 7.5). Comme CURE, elle maintient une structure hiérarchique et représente un cluster à l'aide d'un ensemble de points.

La section 7.6 regroupe deux thèmes qui ne se rejoignent pas forcément. La première est consacrée à la recherche de clusters dans les flots de données. La question n'est absolument pas triviale si l'on considère que la structure sous-jacente des données peut évoluer au fil du temps, les groupes peuvent se subdiviser, se contracter, s'élargir ou encore être fusionnées (page 271). L'algorithme BDMO (Babcock, Datar, Motwani, O'Callaghan) est présenté. Au-delà de sa compréhension, nous percevons en filigrane les enjeux posés par le traitement de ce type de données. La seconde partie de la section traite de la parallélisation des

algorithmes de classification automatique via le paradigme MapReduce. Je ne vois pas trop pourquoi ce sujet est placé dans la même section que le traitement des flots de données. Peut-être parce que les auteurs ont peu de choses à dire, du moins pour l'instant, puisqu'il s'étend sur une douzaine de lignes seulement. J'imagine que cette partie sera développée dans les prochaines éditions de l'ouvrage.

8. Advertising on the Web

Le chapitre 8 nous parle de la publicité sur Internet. Que vient faire la publicité dans un ouvrage de data mining ? Le fait est que la publicité à tout crin est inefficace, voire contre-productive. Afficher une réclame sur des jantes alliages sur un site de lingerie ou suite à une requête sur la recette du bœuf bourguignon ne sert pas à grand-chose, parce que personne ne la lira. La même annonce sur un site de vente de voitures d'occasion ou suite à une requête sur des pneus aura nettement plus d'impact. Il faut donc contextualiser la publicité en tenant compte de caractéristiques qui restent à définir. Et pour cela, on se sert des données disponibles.

Dans le cas des publicités de type AdWords suite à une requête, nous devons faire face à un double pari : maximiser les revenus de la régie, maximiser le taux de clics pertinents pour l'annonceur ; tout cela en tenant compte du budget de l'annonceur. L'optimisation est relativement simple si l'on dispose déjà de toutes les données (*off-line algorithm*) c.-à-d. des clics effectués par les internautes lorsque l'annonce est mise en ligne sur toute sa durée de vie. Mais en pratique, ce n'est pas réalisable. Lorsque l'AdWord est mise en ligne, on ne sait pas encore comment l'internaute va réagir. Les auteurs parlent alors d'algorithme en-ligne (*on-line algorithm*), une forme particulière de l'apprentissage sur flots de données où l'on doit donner une réponse tout de suite (afficher ou pas l'AdWord) en espérant maximiser la fonction objectif. Le rapport entre les solutions off-line et on-line est appelé « competitive ratio ». Le problème des AdWords est défini formellement dans la section 8.4.2. Les auteurs décrivent alors des scénarios de solutions en partant d'une situation simplifiée vers une situation réaliste.

Le sujet est en soi intéressant. Mais j'avoue avoir été un peu dérouté à la lecture de ce chapitre. Nous ne sommes pas vraiment dans le cadre habituel du data mining où l'on vise à produire des modèles à partir des données disponibles. Il s'agit plutôt d'optimisation ici.

9. Recommandations Systems

Le chapitre 9 est plus en phase avec notre acception habituelle de la fouille de données. On nous parle des systèmes de recommandations en ligne auxquels, en tant qu'internautes, nous sommes confrontés tous les jours. En allant sur des sites de ventes en ligne d'ouvrages ou d'autres produits, nous avons tous remarqué qu'en choisissant un produit (le livre « Mining of Massive Datasets » par exemple), une flopée d'autres produits « similaires » nous est conseillée, nous incitant à les acheter. Personne n'apprécie qu'on lui force la main. Mais je reconnais que les propositions sont parfois (souvent) pertinentes. Ces systèmes de recommandation reposent essentiellement sur deux technologies : ceux fondés sur la caractérisation des produits « content-based systems », vous avez cliqué sur un livre de philosophie, on vous propose d'autres références du même acabit ; et ceux fondés sur la similarité entre les produits (*items*) ou les utilisateurs (*users*), le fameux « les personnes qui ont acheté tel produit, ont aussi acheté tels autres produits ».

La section 9.2 traite de la recommandation par le contenu. L'idée est de projeter les produits dans un nouveau espace de représentation décrit par les principales caractéristiques (*features*) pertinentes pour la prescription. Il s'agit là d'une forme de réduction de dimensionnalité. Prenons l'exemple des films, ils peuvent être décrits par la liste des acteurs, du réalisateur, de l'année de sortie, de leur genre. Ces *features* peuvent être fournies par expertise, mais elles peuvent être également extraites des produits lorsqu'elles sont associées à des descriptions textuelles (4ème de couverture des livres par exemple) (section 9.2.2) ou décrits à l'aide de tags (qui sont des textes descriptifs) comme nous pouvons le faire sur des images (section 9.2.3). Une fois le tableau de données constitué, différentes approches sont possibles : trouver les groupes d'items aux caractéristiques similaires, trouver les groupes d'individus aux goûts similaires (ayant choisi les mêmes produits), trouver les meilleures associations utilisateurs – produits pour la recommandation. Les techniques de fouilles de données peuvent s'exprimer pleinement.

Le filtrage collaboratif (section 9.3) s'appuie sur la matrice d'utilité (section 9.1.1). Elle croise les individus et les produits, elle décrit la préférence (une note) des premiers pour les seconds. La matrice est composée de nombreuses cellules vides. Un des enjeux est justement de les combler, pas toutes forcément, en priorité celles qui sont susceptibles de prendre des valeurs élevées (page 309). Un autre enjeu possible est d'essayer de déterminer

des similarités entre les individus avec l'idée sous-jacente : « A » et « B » ont des profils proches en termes de préférences, « A » a acheté le « produit 1 », proposons-le à « B », il y a de fortes chances qu'il en fasse l'acquisition également.

Les applications sont nombreuses. Mais il faut composer avec le caractère particulier de la matrice d'utilité (grandes dimensions avec de nombreuses valeurs manquantes). La section 9.4 est dévolue à la réduction de la dimensionnalité dans ce contexte, la décomposition U-V d'une matrice, un cas particulier de la décomposition en valeurs singulières détaillée dans le chapitre 11, est décrite de manière approfondie. Dans l'espace réduit, en évacuant le bruit et en ne conservant que l'information « utile », on s'attend à ce que les proximités soient mesurées de manière plus fiable.

Les recherches sur les systèmes de recommandations ont connu une avancée significative avec le challenge NetFlix où la société éponyme avait proposé un prix de 1.000.000 de dollars pour toutes personnes ou organismes qui proposerait un dispositif meilleur que le sien (section 9.5). Les données étaient composées de notes attribuées par, approximativement, un demi-million d'utilisateurs à 17.000 films (http://www.netflixprize.com/).

10. Mining Social-Network Graphs

On s'attelle à l'analyse des réseaux sociaux dans le chapitre 10. Facebook est l'exemple typique de réseau social que l'on peut représenter sous forme de graphe : les individus sont les nœuds, ils sont reliés par une arête s'ils sont amis. Twitter et le mécanisme des *followers* peut également constituer un réseau social. La détection de communautés dans les graphes sociaux est l'application phare du domaine. Mais il est possible de mettre en évidence d'autres propriétés des graphes, représentatives de relations entretenues par les individus ou des informations qu'ils s'échangent.

La section 10.1 explique dans quelle mesure un réseau social peut s'exprimer sous la forme d'un graphe. Un réseau social est constitué d'une collection d'entités, des personnes souvent, qui maintiennent une forme de relation entre elles. Elle peut être binaire, les entités sont connectées ou pas, mais elle peut aussi être valorisée par une valeur indiquant un degré de proximité (ex. relation familiale vs. professionnelle) ou une intensité de la liaison (ex. nombre de messages échangés sur une période). Enfin, on fait l'hypothèse que

les connexions ne sont pas aléatoires. Des personnes partageant des caractéristiques communes ou certaines accointances ont tendance à être connectées entres elles. La notion de réseau social dépasse largement le cadre du web. Il s'applique dès lors que nous disposons d'un ensemble d'individus qui peuvent se cliver selon différentes formes d'affinités.

Les sections suivantes détaillent les différentes approches pour détecter des communautés. L'algorithme hiérarchique de Grivan-Newman (section 10.2) repose sur la notion de *betweenness* qui traduit le degré de centralité d'un nœud du graphe par rapport aux autres, elle peut s'étendre à la relation entre deux nœuds. Nous disposons ainsi d'un tableau avec une gradation de valeurs de proximité propices à la partition progressive des entités. L'approche basée sur minimisation du nombre d'arêtes dans les sous-groupes permet également de partitionner efficacement le graphe (section 10.4). La recherche des cliques permet l'attribution d'un individu – à des degrés divers – à différentes communautés (section 10.3). Le problème est a priori NP-Complet, mais il est possible d'obtenir des résultats de bonne qualité via des heuristiques s'appuyant sur la recherche des itemsets fréquents (chapitre 6). Enfin, deux individus qui partagent des centres d'intérêts communs - c.-à-d. qui appartiennent aux mêmes communautés – ont plus de chances d'êtres « amis ». Et inversement. On doit pouvoir en tenir compte (section 10.5).

Les nœuds d'un graphe ne sont pas forcément du même type. Pour une communauté scientifique, les entités peuvent être des auteurs et/ou des articles. En citant un article, nous faisons référence à un groupe d'auteurs. L'analyse est enrichie, encore faut-il savoir exploiter à bon escient cette particularité. L'approche « Simrank » (section 10.6) s'applique aux graphes à nœuds hétérogènes, elle mesure les similarités entre les nœuds de même type. Une fois la matrice des similarités calculée, les méthodes de data mining usuels peuvent s'appliquer. Bien sûr, l'approche se généralise aux graphes homogènes. Elle s'appuie sur la notion du marcheur aléatoire, un peu à la manière du surfeur aléatoire de *PageRank*. La méthode ne s'applique qu'aux graphes de taille limitée du fait de la lourdeur des calculs.

Une autre approche pour mesure les proximités repose sur le comptage des « triangles » dans le graphe (section 10.7). En effet, si « A » et « B » sont amis, et que « A » est ami avec « C », il y a de fortes chances que « B » et « C » le soient également, sauf si les amitiés sont

distribuées au hasard. La densité des triangles traduit l'existence d'une communauté mature.

Enfin, la section 10.9 nous explique les propriétés de voisinage des graphes. Des notions importantes telles que le diamètre d'un graphe, la fermeture transitive, la taille des voisinages sont explicités.

Franchement, j'ai du m'accrocher pour lire attentivement et assimiler ce chapitre. Je connaissais (pensais bien connaître) les graphes, mais là il m'a fallu réviser un peu pour apprécier pleinement les techniques étudiées, et je ne suis pas sûr d'avoir tout compris complètement à ce stade. Mais je me dis quand même qu'il y a matière à faire des choses intéressantes dans mes prochains cours. L'analyse des réseaux sociaux est une compétence incontournable aujourd'hui pour un statisticien / data miner.

11. Dimensionality Reduction

Le chapitre 11 est dédiée à la réduction de dimensionnalité. Il s'agit surtout de construction d'un espace de représentation simplifié préservant certaines propriétés de la description initiale. Le chapitre débute avec l'extraction des valeurs et vecteurs propres d'une matrice carrée symétrique à l'aide de la méthode de la puissance itérée. Ces notions sont importantes pour la compréhension de l'analyse en composantes principales (ACP) présentée dans la section suivante (section 11.2). Un exemple particulièrement didactique (section 11.2.1) permet de comprendre toutes les étapes. Je le vois bien en sujet d'examen pour les étudiants. On notera que travailler dans l'espace des individus ou des variables est équivalent (section 11.2.3).

La section suivante (section 11.3) s'intéresse à la décomposition en valeurs singulières (*singular value decomposition*) d'une matrice. L'intérêt est surtout dans la possibilité de mettre en relation cette partie avec la précédente. On sait qu'il est possible d'obtenir les résultats de l'ACP : soit via la diagonalisation de la matrice des corrélations, soit via la décomposition en valeurs singulières du tableau des données centrées réduites (voir « Analyse en Composantes Principales - Principes et pratique de l'ACP », Ricco Rakotomalala ; sections 2 et 3).

On rentre de plain-pied dans la fouille des données massives avec la méthode CUR (section 11.4). Elle permet de traiter les matrices à haute dimensionnalité, avec pour particularité de

très nombreuses cellules avec des valeurs 0 (on parle de matrice creuse). La décomposition obtenue n'est pas exacte, mais elle est raisonnablement précise si l'on souhaite produire un nouvel espace avec un nombre réduit de facteurs. Les champs d'applications sont nombreux. Les matrices en *text mining* notamment présentent ces caractéristiques de dimension et de remplissage.

12. Large-Scale Machine Learning

Le chapitre 12 conclut l'ouvrage. Il est consacré aux algorithmes d'apprentissage automatique adaptés aux grandes dimensions ou qui ont des potentialités en matière de parallélisation. « Machine Learning » est générique et couvre à la fois l'apprentissage supervisé et non supervisé. En réalité, les auteurs se focalisent sur les méthodes d'analyse prédictive (section 12.1). La démarche globale est résumée dans la figure 12.3 : un modèle est entraîné à partir d'un échantillon d'apprentissage (*training set*), ses performances sont par la suite évaluées sur un échantillon test (*test set*). La distinction est faite entre le *batch-learning* et l'*on-line learning*. Dans le premier cas, toutes les données sont disponibles, on crée le modèle prédictif (le classifieur) une fois pour toutes. Dans le second, les données arrivent en flux, la question de la maintenance des modèles est posée.

Plusieurs techniques sont par la suite décrites : le perceptron (section 12.2), pour lequel on notera une variante simplifiée où toutes variables sont binaires (algorithme WINNOW), situation qui peut se présenter en *text mining* par exemple (pondération présence/absence) ; les supports vector machines (SVM, section 12.3), par rapport à la présentation ultra-classique que l'on retrouve partout, on notera l'appréhension des très grandes bases par la descente du gradient (section 12.3.4) qui ouvre de nouvelles portes à cette approche dont on connaît les bonnes propriétés par ailleurs ; la méthode des plus proches voisins (section 12.4). Pour les différentes approches, l'implémentation parallèle via le paradigme MapReduce est évoquée très rapidement. Un peu trop rapidement à mon goût.

Outre les performances prédictives, les auteurs proposent plusieurs critères pour comparer les méthodes : le type de variables prédictives gérées (numériques, catégorielles, etc.), la capacité à traiter les fortes dimensionnalités ; la compréhensibilité des classifieurs produits. Le texte est émaillé de nombreux commentaires qui peuvent interpeller les spécialistes du

data mining. Mais clairement, les personnes qui souhaitent s'initier à ces techniques auront du mal à se projeter dans la lecture de ce chapitre.

13. Conclusion

Je suis un peu mitigé quand je fais le bilan. Peut-être parce que j'avais une idée bien arrêtée de ce que je cherchais en entamant la lecture de cet ouvrage. Je m'intéressais avant tout aux méthodes de *machine learning* qui s'appliquent sur de très gros volumes de données, soit parce qu'elles ont été sciemment développées à cet effet, soit via un aménagement judicieux des méthodes existantes. Nous avons bien des réponses ici ou là. Je ne connaissais pas par exemple la méthode de clustering CURE qui permet d'appréhender de très grandes volumétries en ne montant qu'un échantillon de données en mémoire centrale. Je ne connaissais pas non plus la méthode CUR pour la réduction de la dimensionnalité lorsque les matrices sont creuses, situation fréquemment rencontrée en text mining par exemple. J'ai été très intrigué également par la résolution des SVM à travers une descente du gradient (ou même descente du gradient stochastique avec un mécanisme d'échantillonnage) qui étend son aptitude à traiter des grandes bases. Il reste que je suis globalement resté sur ma faim. J'aurais aimé que les auteurs aillent plus loin, en présentant un éventail plus large de méthodes peut-être, ou encore en approfondissant des aspects tels que la parallélisation des algorithmes.

Ma satisfaction est ailleurs. L'ouvrage m'a permis de mettre un pied dans des domaines que je connaissais d'un peu loin et que j'ai pu explorer plus en détail en me repérant par rapport aux jalons proposés par les auteurs. J'ai ainsi pu approfondir la notion de *PageRank*, comprendre les enjeux et les méthodes de la fouille de flots de données, identifier les calculs sous-jacents aux systèmes de filtrage collaboratifs et de recommandations, les détections de communautés dans les réseaux sociaux. Nous n'avons pas bien sûr toutes les réponses dans une présentation qui s'apparente plus à un survol (le data stream mining à lui seul pourrait faire l'objet d'un ou plusieurs manuels, idem pour l'analyse des réseaux sociaux, etc.). Mais au moins, nous disposons des bons repères et pointeurs nous permettant d'approfondir par nous-mêmes. De ce point de vue, je ne regrette absolument pas d'avoir passé du temps sur ce livre.