L'informatique des entrepôts de données

Daniel Lemire

SEMAINE 13
L'exploration des données

13.1. Présentation de la semaine

L'exploration de données (ou data mining) est souvent associée à l'intelligence d'affaires. Nous présenterons cette semaine les notions essentielles du domaine. Nous distinguerons aussi l'exploration de données d'une approche guidée par l'utilisateur telle qu'OLAP.

13.2. Qu'est-ce que l'exploration de données?

Il n'existe pas de définition universellement acceptée de l'exploration de données. Cependant, le point de départ est toujours la présence d'une quantité appréciable de données. L'exploration de données repose ensuite sur des méthodes automatiques ou semi-automatiques pour le

traitement des données. On cherche généralement à extraire automatiquement des connaissances générales à partir des données. Pour y arriver, on utilise un mélange de statistiques et d'intelligence artificielle.

L'exploration de données (*data mining*) est un processus d'analyse de grands ensembles de données visant à découvrir des motifs, des corrélations ou des tendances cachées, souvent à l'aide de techniques d'intelligence artificielle, d'apprentissage automatique et de statistiques. Elle permet d'extraire des informations exploitables pour la prise de décision dans des domaines comme le marketing (segmentation de clients), la finance (détection de fraudes) ou la santé (prédiction de maladies). Les étapes clés incluent le nettoyage des données, la sélection de modèles (comme les arbres de décision ou les réseaux neuronaux), et l'interprétation des résultats. Des outils comme RapidMiner, Weka ou des bibliothèques Python (scikit-learn) facilitent ce processus.

Il faut cependant distinguer l'exploration de données de la statistique traditionnelle. Bien qu'il n'y ait pas de distinction toujours claire entre les deux, le volume de données constitue une indication. Ainsi, si on procède à partir d'un échantillon (comme dans le cas d'un sondage), on fait alors des statistiques conventionnelles.

Dans le même ordre d'idée, toute forme d'intelligence artificielle n'est pas nécessairement pertinente en exploration de données. En effet, l'intelligence artificielle se construit souvent avec un modèle de la réalité élaboré à partir de règles définies par des experts. Cela n'est pas un cas d'apprentissage machine.

Une caractéristique fréquente en exploration de données est le très grand volume de données. Ainsi, les algorithmes utilisés en exploration de données doivent souvent être très efficaces. Il faut parfois utiliser des moyens considérables pour le traitement des données. Il n'est pas rare de faire appel à des grilles de calcul.

Il n'est pas rare d'utiliser Hadoop pour le stockage des données volumineuses. Hadoop est un *framework* open-source de la fondation Apache, conçu pour le traitement distribué et le stockage de grandes quantités de données sur des *clusters* de serveurs à faible coût. Il repose sur deux composants principaux : **HDFS** (*Hadoop Distributed File System*), qui stocke les données de manière distribuée et tolérante aux pannes, et **MapReduce**, un modèle de programmation permettant le traitement parallèle des données à grande échelle. Hadoop est particulièrement adapté aux charges de travail analytiques, comme l'exploration de données ou l'analyse de *logs*, dans des domaines tels que la finance, les télécommunications ou les réseaux sociaux. Complété par des outils comme Hive (pour les requêtes SQL-like), Pig (pour le traitement des données), et HBase (pour l'accès aléatoire), Hadoop a

été un pilier du *Big Data*, bien que sa complexité et l'émergence de solutions *cloud* comme Spark ou Snowflake aient réduit son adoption récente.

On utilise l'exploration de données à plusieurs fins : pour mieux comprendre les données ou même pour faire des prédictions. Par exemple, la plupart des logiciels de courriel utilisent une classification des courriels selon qu'ils sont légitimes ou qu'ils constituent du pourriel. Le plus souvent, cette classification s'effectue sur la base d'une exploration des données. Par exemple, on peut utiliser la classification bayésienne. Le filtrage bayésien du *spam* est une technique d'apprentissage automatique utilisée pour détecter les courriels indésirables (*spam*) en appliquant le théorème de Bayes. Cette méthode analyse le contenu d'un courriel en calculant la probabilité qu'il soit un *spam* ou un courriel légitime (*ham*), en se basant sur la fréquence des mots ou des motifs dans des ensembles d'entraînement préalablement étiquetés. Par exemple, des mots comme "gratuit" ou "offre spéciale" augmentent la probabilité de *spam*, tandis que des termes comme "réunion" ou "projet" suggèrent un courriel légitime. Le filtrage bayésien est efficace grâce à son adaptabilité aux nouveaux types de *spam* et sa capacité à s'améliorer avec l'ajout de données, bien que des techniques comme l'obfuscation (ex. : "V1@GR@") puissent poser des défis. Populaire dans des outils comme SpamAssassin, il reste une référence malgré l'essor de méthodes plus complexes basées sur les réseaux neuronaux.

La différence fondamentale entre OLAP et l'exploration de données est que les systèmes OLAP visent à répondre directement aux interrogations de l'analyste. À l'inverse, l'exploration de données n'est généralement pas entièrement guidée par un analyste. Ce sont plutôt les algorithmes qui déterminent le résultat.

Il y a une multitude de problèmes intéressants en exploration des données. Nous en présenterons brièvement trois exemples importants : le partitionnement, règles d'association et arbres de décision.

13.3. Partitionnement des données

Le partitionnement des données consiste à classifier automatiquement des données. Par exemple, un marchand pourrait vouloir classer automatiquement ses clients en ensembles homogènes.

L'algorithme sans doute le plus utilisé est K-means. Il est plus facile de le comprendre si on imagine des points dans l'espace. Le calcul procède généralement de la manière suivante. On choisit aléatoirement k éléments. Ils forment les centres de k noyaux. (Le paramètre k est fixé à l'avance.) On répartit les points restants dans l'un des k noyaux

en choisissant selon la technique des plus proches voisins. Ensuite, pour chaque noyau, on calcule la moyenne des points. Chaque moyenne forme alors un nouveau centre à partir duquel nous allons construire un nouveau noyau. Et ainsi de suite. Au bout d'un certain temps, l'algorithme se stabilisera. On trouve plusieurs animations intéressantes illustrant l'algorithme K-means.

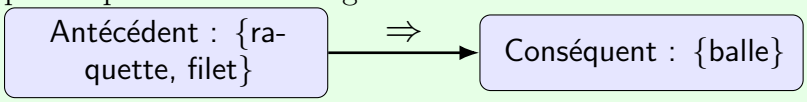
Bien que très intéressant, le partitionnement a ses limites. Par exemple, différents algorithmes vont donner différents partitionnements. Comment alors choisir la bonne version?

13.4. Les règle d'association

Une technique très commune en exploration de données est la recherche de règles d'association. Un exemple simple de règle d'association au sein d'un supermarché pourrait être la constatation que la plupart des gens qui achètent de la bière, achètent aussi des couches pour bébé. Sur la base de cette association, trouvée automatiquement, un marchand pourra adopter une disposition des produits visant à en bénéficier.

Exemple 1: Règle d'association

Un marchand observe que lorsqu'un client achète une raquette et un filet, il achète aussi une balle. Il peut alors en déduire la règle d'association suivante : (raquette, filet) $\Rightarrow$ (balle). On peut représenter cette règle d'association de la manière suivante :



Formellement, on définit les règles d'association comme suit. Étant donné un ensemble d'événements $A_1, A_2, \ldots, A_i$ et un autre ensemble d'événements $B_1, B_2, \ldots, B_j$, on dit qu'il y a une règle d'association d'un ensemble vers l'autre, $A_1 \wedge \ldots \wedge A_i \rightarrow \wedge B_1 \wedge \ldots \wedge B_j$ si, lorsque les événements $A_1, A_2, \ldots, A_i$ se produisent, on constate aussi l'ensemble des événements $B_1, B_2, \ldots, B_j$. Par exemple, si le fait de prendre l'avion et d'être une femme est associé avec le fait d'être blonde et enceinte, on dira que nous avons la règle d'association (avion, femme) $\rightarrow$ (blonde, enceinte). Évidemment, toutes les règles d'association ne sont pas égales. On détermine la qualité d'une règle à l'aide de son support et de sa confiance. Le support est défini comme étant la probabilité que la règle s'applique (noté $P(B_1 \wedge \ldots \wedge B_j \wedge A_1 \wedge \ldots \wedge A_i)$)). En effet, s'il est rare que les femmes enceintes et blondes prennent l'avion,

alors la règle portant sur les femmes qui prennent l'avion n'aura peut-être pas une grande valeur pour un analyste. La confiance d'une règle est définie comme étant la probabilité que le second ensemble d'événements se produira étant donné que le premier ensemble s'est produit : $P(B_1 \wedge \ldots \wedge B_j | A_1 \wedge \ldots \wedge A_i) = \frac{P(B_1 \wedge \ldots \wedge B_j \wedge A_1 \wedge \ldots \wedge A_i)}{P(A_1 \wedge \ldots \wedge A_i)}$. Afin d'illustrer ces mesures, considérons le tableau suivant :

Monster Species	Color	vegetarian?
Ziziz	Red	yes
YiYoz	Blue	yes
Filoufoul	Red	no
Coucou	Red	yes
Passpass	Blue	yes

À partir de cette table, nous pouvons déterminer que la règle selon laquelle les monstres bleus sont végétariens a un support de 40% et une confiance de 100%. La règle selon laquelle les monstres rouges sont végétariens a un support de 40% et une confiance de 66.7%.

Dans la pratique, on fixe souvent des contraintes sur le support et la confiance. Par exemple, on ne s'intéressera qu'aux règles qui ont une confiance d'au moins 80% et un support d'au moins 1%. Une règle d'association qui a à la fois une bonne confiance et un bon support est dite une règle forte.

En pratique, la recherche des règles fortes dans une grande base de données est un problème difficile. Un des algorithmes les plus communs est Apriori [1].

L'algorithme Apriori est une méthode d'exploration de données utilisée pour identifier des règles d'association dans de grands ensembles de données, notamment pour découvrir des motifs fréquents dans les transactions, comme dans l'analyse des paniers d'achat. Introduit par Rakesh Agrawal et Ramakrishnan Srikant en 1994, il repose sur le principe que si un ensemble d'items est fréquent, tous ses sous-ensembles le sont également. L'algorithme procède par itérations : il génère d'abord des ensembles d'items candidats (*itemsets*) de taille croissante, puis évalue leur fréquence (*support*) dans les données, éliminant ceux en-dessous d'un seuil défini. Par exemple, dans un supermarché, il peut révéler que "pain et beurre" sont souvent achetés ensemble. Bien que simple et efficace, Apriori peut être coûteux en calcul pour des ensembles volumineux, ce qui a conduit à des optimisations comme FP-growth. Il reste fondamental en *business intelligence* et en recommandation.

En général, les algorithmes visant à trouver des règles d'association fortes fonctionnent de la manière suivante. On cherche d'abord à répondre à une contrainte sur le support. Parmi tous les cas possibles satisfaisant cette contrainte sur le support, on cherche des règles ayant une forte confiance.

Il n'est cependant pas facile de distinguer automatiquement les règles qui sont utiles de celles qui ne sont d'aucune autre utilité (pour un marchand, par exemple). Ainsi, malgré leur potentiel apparemment immense, les règles d'association demeurent relativement peu utilisées dans le commerce.

Il existe aussi une relation entre les règles d'association et les requêtes de type iceberg. Considérons l'exemple suivant où l'on donne la répartition des achats combinés de pain et de produits laitiers :

	white cheese	yellow cheese	skim milk	1% milk	2% milk	fat milk
whole wheat	12	0	43	13	22	0
white bread	8	16	432	3304	4343	444
brown bread	0	32	2	99	441	4324

On remarque immédiatement la présence d'articles fréquemment associés [3] :

- (white bread, 1% milk) : 3304
- (white bread, 2% milk) : 4343
- (brown bread, fatty milk): 4324

On peut mettre à jour de telles associations simples avec une requête iceberg [2]. Ils correspondent à des règles d'association (par ex. white bread $\rightarrow$ 1% milk).

13.5. Activité pratique

Pour approfondir votre compréhension des règles d'association, nous vous proposons une activité pratique. Rendez-vous à l'adresse suivante: https://github.com/lemire/python_association_rule/. Vous y trouverez un exemple de code Python illustrant la recherche de règles d'association à partir d'un ensemble de données. Vous pourrez ensuite expérimenter avec l'algorithme Apriori et découvrir comment extraire des règles d'association à partir de données réelles.

13.6. Les arbres de décision

Les arbres de décision peuvent être vus comme une généralisation des règles d'association lorsqu'il y a plus d'une variable. Imaginons un marchand qui découvre la séquence logique suivante. Si un client entre seul, alors il va acheter pour moins de 50$. S'il entre avec quelqu'un d'autre, mais, qu'il a plus de 50 ans, il va acheter pour moins de 100$,

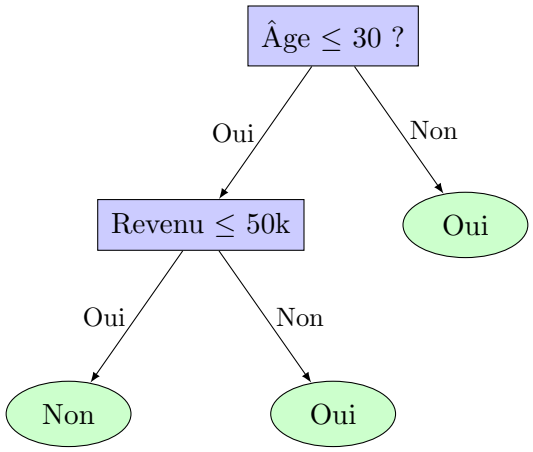
mais sinon, il va probablement acheter pour plus de 200$. On voit ici que comme les règles s'appliquent les uns à la suite des autres, nous avons effectivement une forme d'arbre. Wikipedia a un excellent article sur ce sujet :

Les arbres de décision ont l'avantage d'être facilement compréhensibles (par l'humain). Il existe plusieurs algorithmes efficaces pour en effectuer automatiquement le calcul.

Les arbres de décision sont des modèles d'apprentissage automatique utilisés pour résoudre des problèmes de classification et de régression en représentant des décisions sous forme d'une structure arborescente. Chaque nœud de l'arbre correspond à une condition sur une caractéristique, chaque branche à une réponse possible, et chaque feuille à une décision ou une prédiction. Populaire pour leur simplicité et leur interprétabilité, ils sont largement utilisés dans des domaines comme la finance (évaluation du risque de crédit), la médecine (diagnostic) ou le marketing (segmentation client). Cet article explore leurs principes, leurs applications, et leurs limites.

Un arbre de décision est construit en divisant récursivement l'espace des caractéristiques en régions basées sur des critères, comme le gain d'information (basé sur l'entropie) ou l'indice de Gini, pour maximiser la séparation des classes ou minimiser l'erreur de prédiction. L'algorithme sélectionne la caractéristique et le seuil qui optimisent ce critère à chaque nœud, créant des branches jusqu'à atteindre un critère d'arrêt (par exemple, profondeur maximale ou pureté des feuilles). Les principaux algorithmes incluent CART (*Classification and Regression Trees*), ID3 et C4.5, mais le plus important est sans doute CART. Les arbres peuvent être *élagués* pour éviter le surajustement, en supprimant les branches peu significatives.

Considérons un problème de classification pour prédire si un client achètera un produit en fonction de son âge et de son revenu. Un arbre de décision pourrait être construit comme suit :



Dans cet exemple :

- Le nœud racine teste si l'âge est inférieur ou égal à 30 ans.
- Si oui, un second test vérifie si le revenu est inférieur ou égal à 50 000$.
- Les feuilles indiquent la prédiction : "Oui" (achète) ou "Non" (n'achète pas).

Un client de 25 ans avec un revenu de 40 000$ suivrait le chemin "Âge $\leq$ 30" $\rightarrow$ "Revenu $\leq$ 50k" $\rightarrow$ "Non", prédisant qu'il n'achètera pas.

Les critères de division sont formalisés mathématiquement. L'entropie d'un ensemble S avec c classes est donnée par $H(S) = -\sum_{i=1}^{c} p_i \log_2 p_i$, où p_i est la proportion de la classe i dans S. Le gain d'information pour un attribut A est $IG(S, A) = H(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H(S_v)$, où S_v est le sous-ensemble de S pour la valeur v de A. L'indice de Gini pour S est $Gini(S) = 1 - \sum_{i=1}^{c} p_i^2$, et le gain pour A est $Gini(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} Gini(S_v)$. Ces mesures guident la sélection de l'attribut optimal à chaque nœud, en maximisant la pureté des sous-ensembles résultants.

L'algorithme CART (Classification and Regression Trees) a été développé en 1984 par Leo Breiman, Jerome Friedman, Richard Olshen et Charles Stone, marquant une avancée significative dans les méthodes d'apprentissage automatique pour la classification et la régression. Contrairement aux approches antérieures, CART produit systématiquement des arbres binaires, en divisant les données de manière récursive sur la base d'un critère d'impureté tel que l'indice de Gini pour la classification ou l'erreur quadratique moyenne pour la régression, afin de minimiser l'hétérogénéité dans les nœuds enfants. Ce processus commence par la sélection de l'attribut et du seuil optimal qui maximisent la réduction d'impureté, et se poursuit jusqu'à un critère d'arrêt comme une

Entrées : Un ensemble de données D avec des exemples étiquetés (classification) ou valeurs cibles (régression), une liste d'attributs A
Sorties : Un arbre de décision
if *D est vide ou respecte un critère d'arrêt (ex. : profondeur max, taille min)* **then**
  Retourner une feuille avec la prédiction majoritaire (classification) ou moyenne (régression);
end
if *tous les exemples dans D ont la même valeur cible* **then**
  Retourner une feuille avec cette valeur;
end
Sélectionner l'attribut *best* et le seuil *threshold* qui minimisent l'impureté (Gini ou MSE);
Créer un nœud racine avec *best* et *threshold*;
$D_{left} \leftarrow$ sous-ensemble de D où $best \leq threshold$;
$D_{right} \leftarrow$ sous-ensemble de D où $best > threshold$;
Ajouter le sous-arbre gauche : CART(D_{left}, A);
Ajouter le sous-arbre droit : CART(D_{right}, A);
return *l'arbre construit, suivi d'un élagage si nécessaire*;

Algorithme 1 : Algorithme CART pour la construction d'un arbre de décision

profondeur maximale ou une taille minimale de nœud, suivi souvent d'un élagage basé sur la complexité pour éviter le surajustement. Voir Algorithme 1 pour une description formelle de l'algorithme CART.

Les arbres de décision offrent plusieurs avantages :

- **Interprétabilité** : Leur structure visuelle est facile à comprendre, même pour les non-experts.
- **Flexibilité** : Applicables à la classification et à la régression, avec des données numériques ou catégoriques.
- **Préparation minimale** : Peu de prétraitement requis (pas besoin de normalisation).

Cependant, ils présentent des limites :

- **Surajustement** : Sans élagage, ils peuvent mémoriser les données d'entraînement, réduisant la généralisation.
- **Instabilité** : De petites variations dans les données peuvent produire des arbres très différents.
- **Biais vers les caractéristiques dominantes** : Les caractéristiques avec plus de valeurs possibles peuvent être favorisées.

Ces limites sont souvent atténuées par des techniques comme les forêts aléatoires (*random forests*), qui combinent plusieurs arbres pour améliorer la robustesse.

Les arbres de décision sont des outils puissants et intuitifs pour l'apprentissage automatique, grâce à leur capacité à modéliser des décisions complexes de manière interprétable. Leur simplicité les rend idéaux pour des applications où la transparence est cruciale, comme le diagnostic médical ou l'analyse de risques. Cependant, leur tendance au surajustement et leur sensibilité aux données nécessitent des ajustements, comme l'élagage ou l'utilisation d'ensembles. En combinaison avec des approches modernes, comme les forêts aléatoires ou les *gradient boosting*, les arbres de décision restent un pilier de l'analyse de données et de la prise de décision automatisée.

13.7. Implantation

Il est parfois relativement facile de programmer ses propres algorithmes d'exploration de données. Malheureusement, ce n'est pas toujours le cas, surtout si le volume de données est important. Heureusement, il existe un vaste choix de solutions logicielles :

- Weka (Open Source) http://www.cs.waikato.ac.nz/~ml/weka/
- Mahout (Open Source) http://mahout.apache.org/
- Data Mining Extensions (Microsoft, SQL Server) http://msdn.microsoft.com/en-us/library/ms132058.aspx

13.8. Activité pratique

Pour approfondir votre compréhension des arbres de décision. nous vous proposons une activité pratique. Rendez-vous à l'adresse suivante: https://github.com/lemire/python_tree_example. Vous y trouverez un exemple de code Python illustrant la construction d'un arbre de décision à partir d'un ensemble de données.

13.9. Questions d'approfondissement

(a) Est-ce qu'OLAP est une forme d'exploration de données?
(b) Étant donné l'exemple de monstres, est-ce qu'on pourrait appliquer K-means pour partitionner l'ensemble des monstres?
(c) Étant donné une table comportant deux colonnes, comment procéderiez-vous pour trouver toutes les règles d'association fortes allant de la première colonne à la seconde?

(d) Étant donné un arbre de décision sur 10 variables, quelle sera la hauteur maximale de l'arbre?

13.10. Réponses suggérées

(a) Non. Dans un système OLAP conventionnel, les requêtes proviennent d'un analyste. S'il y a extraction de règles ou de motifs, c'est plutôt dans l'esprit de l'analyste que le travail se fera.

(b) Certainement, mais il faudrait alors définir une mesure de distance entre les monstres.

(c) Chaque item respectent la contrainte sur le support. Il suffit ensuite de calculer sa confiance.

(d) On pourrait tout d'abord chercher les items fréquents, c'est-à-dire les paires de valeurs (une par colonne) qui fréquent forme une règle d'association. donc une hauteur maximale de 10.

Dans un arbre de décision conventionnel, chaque variable n'intervient qu'une seule fois. L'arbre de décision aura

Votre avis compte !

Chers étudiants,
Si vous repérez une erreur dans ce document ou si vous avez une suggestion pour l'améliorer, nous vous invitons à remplir notre **formulaire anonyme**.
Votre contribution est précieuse pour nous !

Cliquez ici pour accéder au formulaire

BIBLIOGRAPHIE

1. R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *ACM SIGMOD Record*, 22(2):207–216, 1993.
2. K. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cube. *SIGMOD Rec.*, 28(2):359–370, 1999.
3. J. Park, M. Chen, and P. Yu. An effective hash-based algorithm for mining association rules. *ACM SIGMOD Record*, 24(2):175–186, 1995.