Ateliers de Technologies des Grosses Données

Résumé

Statistique, fouille ou Science des Données, les appellations changent le volume et la diversité des données explosent, les technologies se succèdent, les modèles et algorithmes se complexifient. L'estimation devient un apprentissage, la prévision remplace l'explication.

Le parcours pour devenir data scientist *est structuré en quatre parties :*

Retour *à l'introduction générale*

Saison 1 *(L3) Statistique élémentaire, descriptive* vs. *inférentielle.*

Saison 2 *(M1) Statistique Exploratoire multidimensionnelle et apprentissage non supervisé.*

Saison 3 *Apprentissage Statistique / Machine supervisé.*

Saison 4 *(M2) Technologies pour la Science des (grosses) Données.*

plus des réflexions sur : Statistique et Déontologie scientifique.

1 Motivations

1.1 Objectif

L'objectif de cette saison est d'ouvrir une réflexion sur le choix des meilleures mises en œuvre de techniques d'apprentissage face à des données massives. Cet objectif ambitieux se heurte à de nombreuses difficultés : il présuppose une définition de ce que sont des *données massives*, donc les spécificités de l'environnement de travail et le choix de critères de comparaison.

De façon triviale, des données deviennent massives lorsqu'elles excèdent la capacité de la mémoire vive (RAM) de l'ordinateur (quelques giga-octets), puis réellement massives lorsqu'elles excèdent celle du disque dur (quelques tera-octets) et doivent être distribuées sur plusieurs disques voire machines. Cette définition dépend évidemment de l'environnement matériel mais les principaux problèmes méthodologiques et algorithmiques sont soulevés lorsque les données sont effectivement distribuées sur plusieurs machines même virtuelles. Le principal objectif est alors d'éviter des transferts coûteux de données entre ordinateurs ou même entre disques et unités centrales ; analyse et calculs sont déportés sur le lieu de stockage et doivent se limiter à une seule lecture des données.

L'environnement de travail peut-être un poste de travail personnel avec plus ou moins de mémoire, de processeurs, de cartes graphiques (GPU), l'accès à un puissant serveur de calcul intensif ou encore, c'est la caractéristique originale de la prise en compte de données massives, à un *cluster* de calcul réel ou virtuel, privé ou loué. Pouvoir transférer les calculs d'un poste personnel utilisé pour la réalisation d'un prototype, au déploiement, passage à l'échelle, d'un grand *cluster* à l'aide du même code est un élément essentiel d'efficacité car le coût de développement humain l'emporte largement sur celui de location d'un espace de calcul.

Les évaluations et comparaisons classiques de performances en apprentissage sur des données de faible taille se compliquent encore avec l'explosion combinatoire du nombre d'implémentations des algorithmiques ainsi qu'avec la diversité des technologies de parallélisation et celle des architectures matérielles. Le nombre de paramètres à considérer rend ces comparaisons beaucoup trop complexes ou soumises à des choix bien arbitraires. Face à cette complexité, l'objectif est nécessairement réduit. Loin de pouvoir discuter la réelle optimalité des technologies d'analyse de données massives, on se propose d'aborder les intérêts respectifs de quelques stratégies autour de trois cas d'usage devenus classiques afin d'illustrer, ne serait-ce que la complexité des problèmes et la pertinence de certains choix.

L'objectif est donc réduit à la comparaison des implémentations, dans trois environnements très répandus ou en pleine expansion : R, Python (*Scikit-learn*) et *Spark-MLlib*, de trois méthodes parmi les plus utilisées : les forêts aléatoires (Breiman 2001)[2]) en classification supervisée, la factorisation non négative de matrices (NMF) pour apprendre des recommandations et enfin la régression logistique faisant suite au pré-traitement d'un corpus volumineux de textes.

D'autres approches, d'autres technologies, d'autres méthodes, certes importantes et faisant par exemple appel à l'apprentissage profond (*deep lear-*

ning) pour l'analyse d'images, à des méthodes d'optimisation stochastique ou de décision séquentielle... nécessiteraient des développements spécifiques conséquents ; elles sont volontairement laissées de côté à l'exception d'une introduction à l'"apprentissage profond en utilisant la librairie keras sur des données textuellesles données très classiques (MNIST) de reconnaisance de caratères.

Il s'agit d'illustrer, sur des cas d'usage, le choix qui se pose au statisticien de passer, ou non, à un environnement technologique plus complexe : Python *Scikit-Learn* (Pedregosa et al. (2011)[10], *Spark* (Zaharia et al. 2012)[15] que celui R[12]) qu'il maîtrise depuis de longues années.

1.2 Nouvelle Science des Données

La fouille de données (*data mining*) des années 90 a largement promu de nouveaux métiers et surtout de nouvelles suites logicielles commerciales intégrant gestion, transformations et analyse statistique des données pour s'adresser à de plus vastes marchés, principalement le marketing. L'objectif était déjà la valorisation (gestion de la relation client) de données acquises par ailleurs et sans planification expérimentale spécifique à une démarche statistique traditionnelle. Une deuxième étape, en lien avec l'accroissement du volume et la complexité des données, fut l'explosion de leur très grande dimension avec la multitude des omiques (génome, transcriptome, protéome...) produite par les technologies de séquençage. Ce fut l'expansion de la Bioinformatique et, pour le statisticien confronté au fléau de la dimension, la nécessaire recherche de modèles parcimonieux. Une troisième étape est liée au développement d'internet, du commerce en ligne et des réseaux sociaux. Ce sont les principales sources de production et d'analyse de données massives (*big data analytics*). Comme pour la fouille des données des années 90, mais avec une ampleur sans commune mesure, l'histoire se répète avec l'explosion du marché des espaces publicitaires (*advertasing*) en ligne (*Google, Facebook...*) et celles des services (*Amazon Web Service...*) associés.

Friedman (1997)[5] soulevait déjà la question : *Is data mining an intellectual discipline ?* Les enjeux actuels font *changer d'échelle* pour s'interroger sur la naissance plus prestigieuse d'une "nouvelle" *Science, des Données*, accompagnée d'un considérable battage médiatique. Celui-ci glorifie les exploits de la ruée vers le nouvel eldorado des investissements et emplois mais stigmatise aussi les risques éthiques, juridiques ou les fiascos annoncés. Cette évolution, tant historique que méthodologique, et son influence sur les programmes académiques est décrite par Besse et Laurent (2016)[1]. Elle n'est pas reprise ici mais nous en retenons le principal élément. Celui-ci s'apparente à l'émergence d'un nouveau paradigme provoquée par les changements d'échelles de volume, variété, vélocité des données qui bousculent les pratiques, tant en Statistique qu'en Apprentissage Machine.

L'estimation d'un modèle prédictif, qu'il soit statistique ou d'apprentissage supervisé, est la recherche d'un compromis optimal entre biais (*erreur d'approximation*) et variance (*erreur d'estimation*) donc d'un modèle parcimonieux évitant le sur-ajustement. À cette question centrale en Apprentissage, vient s'ajouter un nouveau problème d'*Optimisation* sous la forme d'un troisième terme d'erreur. Les ressources (mémoire, temps) sont contraintes ; comment minimiser le terme d'erreur d'optimisation dû, soit à un sous-échantillonnage pour satisfaire aux contraintes de mémoire, soit à la limitation des calculs ou méthodes utilisables sur une base d'apprentissage très volumineuse et distribuée ? Une fonction objectif globale à minimiser pourrait être définie à la condition d'évaluer le coût induit par des erreurs de prévision dues à l'échantillonnage, à mettre en balance avec le coût de l'estimation avec données massives sur des serveurs les supportant. Le problème général étant bien trop complexe, nous nous restreignons à des comparaisons plus qualitatives.

1.3 Contenu

La section 2 introduit l'environnement technologique de référence *Spark* (Zaharia et al. 2012)[15] associé à la gestion de données massive *Hadoop distributed file system*. L'importance de la notion de *résilience* des bases de données est illustrée par l'exemple de l'algorithme élémentaire de Forgy (1965)[4] modifié pour satisfaire aux contraintes des fonctionnalités *MapReduce* d'*Hadoop*. Sont introduites également les librairies *MLlib, SparkML* développées pour exécuter les méthodes de modélisation et apprentissage dans ce contexte.

Les sections suivantes abordent trois cas d'usage sur des données publiques ou rendues publiques : reconnaissance de caractères (MNIST), recommandation de films (MovieLens), catégorisation de produits (Cdiscount) avec l'objectif de comparer les performances (temps, précision) des environnements R, Python et Spark sur ces exemples de taille raisonnable.

Les comparaisons ont été opérées sur des machines de faible taille / coût : Lenovo X240 (Windows 7) avec processeur 4 cœurs cadencé à 1,7 GHz et 8Go de RAM, MacBook Pro (OS X Yosemite) avec processeur 4 cœurs cadencé à 2,2 GHz et 16Go de RAM, cluster (Linux, Spark) avec au plus 1 nœud maître et 8 nœuds excécuteurs de 7Go de RAM. Ces configurations ne sont pas très réalistes face à des données massives. Néanmoins les résultats obtenus, sont suffisamment explicites pour illustrer les contraintes du compromis taille des données *vs.* précision pour une taille de mémoire fixée.

La dernière section conclut sur l'opportunité des choix en présence.

2 Principaux environnements technologiques

Les choix de méthodes offertes par les librairies et les performances d'une analyse dépendent directement de l'environnement matériel et logiciel utilisé. La profusion des possibilités rendent ces choix difficiles. Il serait vain de chercher à en faire une synthèse, mais voici quelques éléments de comparaison parmi les environnements les plus pratiqués ou au mois les plus médiatisés et accessibles car sous la licence de l'*Apache Software Fondation*.

2.1 Nouveau modèle économique

Rappelons tout d'abord que le déluge des données, conséquence de la *datafication* de notre quotidien, entraîne un profond changement des modèles économiques en lien avec l'analyse de ces données. Changement qui impacte directement les développements et donc disponibilités des environnements accessibles aux entreprises et aux établissements académiques. Les équipements trop chers sont loués, les langages et logiciels sont libres mais les concepts et méthodes complexes à assimiler et mettre en œuvre sont des services (formations, plateformes) monnayables. Plus généralement, tout un ensemble de ces services et une nomenclature associée se développent avec l'industrialisation, la commercialisation du *cloud computing* : *software as a service* (SaaS), *infrastructure as a service* (IaaS), *platform as a service* (PaaS), *desktop as a service* (DaaS)...

À titre d'illustration, citons seulement quelques entreprises surfant sur la vague des nouvelles technologies : *Enthought* (Canopy) et *Continuum analytics* (Anaconda) proposent des distributions libres de Python et, c'est important, faciles à installer ainsi que des environnements plus élaborés payants et de la formation associée. *Horthonworks* et *Cloudera* diffusent des environnements de *Hadoop* incluant *Spark*. Les créateurs de *Spark* ont fondé *databricks* : *Data science made easy, from ingest to production*, pour principalement vendre de la formation et une certification. Trevor Hastie et Ron Tibshirani conseillent *Oxdata* qui développe (*H20*) avec notamment une forme d'interface entre R et le système *Hadoop*.

Dans le même mouvement ou cherchant à ne pas perdre trop de terrain par rapport à *Amazon Web Services*, les grands éditeurs ou constructeurs intègrent, avec plus ou moins de succès, une offre de service à leur modèle économique d'origine : *Google Cloud platform, IBM Analytics, Microsoft Azure, SAS Advanced Analytics...*

2.2 *Hadoop, MapReduce, Spark*

Hadoop (*distributed file system*) est devenu la technologie systématiquement associée à la notion de données considérées comme massives car distribuées. Largement développé par *Google* avant de devenir un projet de la fondation *Apache*, cette technologie répond à des besoins spécifiques de centres de données : stocker des volumétries considérables en empilant des milliers de cartes d'ordinateurs et disques de faible coût, plutôt que de mettre en œuvre un supercalculateur, tout en préservant la fiabilité par une forte tolérance aux pannes. Les données sont dupliquées et, en cas de défaillance, le traitement est poursuivi, une carte remplacée, les données automatiquement reconstruites, sans avoir à arrêter le système.

Ce type d'architecture génère en revanche un coût algorithmique. Les nœuds de ces serveurs ne peuvent communiquer que par couples (clef, valeur) et les différentes étapes d'un traitement doivent pouvoir être décomposées en étapes fonctionnelles élémentaires comme celles dites de *MapReduce*. Pour des opérations simples, par exemple de dénombrement de mots, d'adresses URL, des statistiques élémentaires, cette architecture s'avère efficace. Une étape *Map* réalise, en parallèle, les dénombrements à chaque nœud ou exécuteur (*workers*) d'un *cluster* d'ordinateurs, le résultat est un ensemble de couples : une clef (le mot, l'URL à dénombrer) associée à un résultat partiel. Les clefs identiques sont regroupées dans une étape intermédiaire de tri (*shuffle*) au sein d'une même étape *Reduce* qui fournit pour chaque clef le résultat final.

Dans cette architecture, les algorithmes sont dits *échelonnables* de l'anglais *scalable* si le temps d'exécution décroît linéairement avec le nombre d'exécuteurs dédiés au calcul. C'est immédiat pour des dénombrements, des calculs de moyennes, ce n'est pas nécessairement le cas pour des algorithmes itératifs complexes. La méthode des k plus proches voisins n'est pas échelonnable au contraire des algorithmes de classification non-supervisée par réallocation dynamique (*e.g.* Forgy, k-means) qui opèrent par itérations d'étapes *MapReduce*.

L'exemple de l'algorithme de Forgy (1965)[4] est très révélateur.

- **Initialisation** de l'algorithme par définition d'une fonction de distance et désignation aléatoire de k centres.
- **Jusqu'à** convergence :
  - L'étape **Map** calcule, en parallèle, les distances de chaque observation aux k centres courants. Chaque observation (vecteur de valeurs) est affectée au centre (clef) le plus proche. Les couples : (clef ou numéro de centre, vecteur des valeurs) sont communiqués à l'étape *Reduce*.
  - Une étape intermédiaire implicite **Shuffle** adresse les couples de même clef à la même étape suivante.
  - **Pour** chaque clef désignant un groupe, l'étape **Reduce** calcule les nouveaux barycentres, moyennes des valeurs des variables des individus partageant la même classe c'est-à-dire la même valeur de clef.

Principal problème de cette implémentation, le temps d'exécution économisé par la parallélisation des calculs est fortement pénalisé par la nécessité d'écrire et relire toutes les données entre deux itérations.

2.3 *Spark*

C'est la principale motivation du développement de la technologie *Spark* (Zaharia et al. 2012)[15] à l'université de Berkeley. Cette couche logicielle au-dessus de systèmes de gestion de fichiers comme *Hadoop* introduit la notion de base de données *résiliente* (*resilient distributed dataset* ou RDD) dont chaque partition reste, si nécessaire, présente en mémoire entre deux itérations pour éviter réécriture et relecture. Cela répond bien aux principales contraintes : des données massives ne doivent pas être déplacées et un résultat doit être obtenu par une seule opération de lecture.

Techniquement, ces RDDs sont manipulées par des commandes en langage *Java* ou *Scala* mais il existe des API (*application programming interface*) acceptant des commandes en Python (*PySpark*) et en R. *Spark* intègre beaucoup de fonctionnalités réparties en quatre modules : *GRAPHX* pour l'analyse de graphes ou réseaux, *streaming* pour le traitement et l'analyse des flux, *SparkSQL* pour l'interrogation et la gestion de bases de tous types et la librairie *MLlib* pour les principaux algorithmes d'apprentissage. En plein développement, cet environnement comporte (version 1.6) des incohérences. *SparkSQL* génère et gère une nouvelle classe de données *DataFrame* (similaire à R) mais qui n'est pas connue de *MLlib* qui va progressivement être remplacée par *SparkML* dans les versions à venir...

Dans la présentation qui en est faite, *Spark* apparaît donc comme un cadre général (*framework*) permettant de connecter la plupart des technologies développées sous forme de projet *Apache*. Tous types de fichier (JSON, csv, RDDs...) et types de données structurées ou non, tous types de flux de données (objets connectés, tweets, courriels...) peuvent ainsi être gérés, agrégés par des opérations classiques de sélection, fusion à l'aide de commandes utilisant une syntaxe dérivée de SQL (*structured querry language*) avant d'être utilisés pour des modélisations.

2.4 Librairie *MLlib*

Ce chapitre est focalisé sur l'utilisation de quelques méthodes de transformation et modélisation sous *Spark* dont celles de *MLlib* décrites par Pentreath (2015)[11]. Cette librairie regroupe les algorithmes d'apprentissage adaptés à des bases de données résilientes et qui supportent donc le *passage à l'échelle* du volume des données. Un programme mis au point sur de petits échantillons et un poste de travail personnel peut en principe s'exécuter en l'état sur un *cluster* de calcul (*e.g. Amazon Web Service*) pour analyser des données massives même si la *scalabilité* n'est pas strictement atteinte.

Pour cette librairie en plein développement et aussi en migration vers *SparkML*, seule la documentation en ligne, malheureusement fort succincte, est à jour concernant la liste des méthodes disponibles et leurs options d'utilisation. En voici un rapide aperçu (version 1.6) :

Statistique de base : Univariée, corrélation, échantillonnage stratifié, tests d'hypothèse, générateurs aléatoires, transformations (standardisation,

quantification de textes avec hashage et TF-IDF pour *vectorisation*), sélection ($\chi 2$) de variables (*features*).

Exploration multidimensionnelle Classification non-supervisée (k-means avec version en ligne, modèles de mélanges gaussiens, LDA (*Latent Dirichlet Allocation*), réduction de dimension (SVD et ACP mais en Java ou Scala pas en Python), factorisation non négative de matrice (NMF) par moindres carrés alternés (ALS).

Apprentissage Méthodes linéaires : SVM, régression gaussienne et binomiale ou logistique avec pénalisation L1 ou L2 ; estimation par gradient stochastique, ou L-BFGS ; classifieur bayésien naïf, arbre de décision, forêts aléatoires, boosting (*gradient boosting machine* en Scala).

Les sections qui suivent proposent une étude comparative des environnements R, Python (*Scikit-Learn*) et *Spark MLlib* sur trois cas d'usage de données publiques. Les données ne sont pas excessivement volumineuses mais sont considérées comme telles en mettant en œuvre des technologies d'algorithmes distribués capables en principe de passer sans modification à des échelles plus importantes. Elles le sont pas ailleurs suffisamment pour mettre en défaut des implémentations pas ou mal optimisées comme la librairie *randomForest* de R.

3 Cas d'usage

3.1 Introduction à Spark

Le dépôt github décrit brièvement l'environnent Spark et les calepins proposés pour s'initier à la pratique de cet environnement : les objets de Spark et la préparation des données, statistiques élémentaires avec Spark, Spak SQL et le type *data frame*.

3.2 Reconnaissance de caractères MNIST

Le problème

La reconnaissance de caractères manuscrits, notamment les chiffres de codes postaux, est un vieux problème de classification supervisée qui sert depuis de nombreuses années de base de comparaison entre les méthodes. Le site *MNIST DataBase* fournit des données (Le Cun et al. (1998)[7] et une liste de références, de 1998 à 2012 proposant des stratégies d'analyse avec des taux d'erreur de 12% à 0,23%. Les méthodes les plus récentes s'affrontent toujours sur ce type de données (Lee et al. 2016)[8] ainsi que dans un concours *Kaggle*.

De façon très schématique, plusieurs stratégies sont développées dans une vaste littérature sur ces données.

- Utiliser une méthode classique : k plus proches voisins, forêt aléatoire... sans trop raffiner mais avec des temps d'apprentissage rapides conduit à un taux d'erreur autour de 3%.
- Ajouter ou intégrer un pré-traitement des données permettant de recaler les images par des distorsions plus ou moins complexes.
- Construire une mesure de distance adaptée au problème car invariante par rotation, translation, homothétie, puis l'intégrer dans une technique d'apprentissage adaptée comme les k plus proches voisins. Simard et al. 1998)[13] définissent une distance dite tangentielle entre les images de caractères possédant ces propriétés d'invariance.
- D'autres pistes peuvent être explorées notamment celles des machines à noyaux par Muller et al. (2001)[9] qui illustrent ce type d'apprentissage par les données de MNIST sans semble-t-il faire mieux que la distance tangentielle.
- Les réseaux de neurones, renommés apprentissage profond (*deep learning*), et implémentant une architecture modélisant une convolution (*convolutional neural network*) impliquant les propriétés recherchées d'invariance, sont les algorithmes les plus compétitifs et les plus comparés (Wan et al. (2013)[14]. Noter le coût de calcul assez prohibitif pour l'estimation et l'optimisation de ces architectures complexes qui nécessitent des moyens matériels et des librairies sophistiquées reprenant toutes ce même jeu de données dans leur tutoriel. C'est le cas de *torch* en langage *Lua* ou de *Lasagne* (Theano) en Python. Même chose pour *H2O*, dont le tutoriel, très commercial, optimise les paramètres à partir de l'échantillon test malgré un risque évident de surapprentissage, ou encore pour *Tensor Flow* rendu (12-2015) accessible par *Google*.
- Toujours en connexion avec le *deep learning* et illustrée par les mêmes données, Brun et Mallat (2013)[3] utilisent une décomposition, invariante par transformations, des images sur des bases d'ondelettes par *scattering*.
- ...

Attention, l'objectif de cette section n'est pas de concourir avec les meilleures solutions publiées. Les données sont simplement utilisées pour comparer les performances des implémentations de solutions élémentaires dans l'objectif d'un passage à l'échelle volume.

Les données

Les données représentent 60 000 images en niveaux de gris de chiffres manuscrits écrits par plusieurs centaines de personnes sur une tablette. Un ensemble de pré-traitements restreint les dimensions des images à $28 \times 28 = 784$ pixels. Celles-ci ont ensuite été normalisées par des transformations élémentaires. L' échantillon test indépendant est constitué de la même manière de 10 000 images.

Ces données ne sont pas réellement massives, elles tiennent en mémoire, mais leur volume suffit à montrer les limites, notamment en temps de calcul ou occupation mémoire, de certains algorithmes.

3.3 Recommandation de films

Le problème

Il s'agit d'un problème de recommandation classique par filtrage collaboratif où seules sont prises en compte des informations croisant clients et films. Il s'agit d'une très grande matrice très creuse contenant les notes attribués par les clients aux quelques films qu'ils ont vus. Il s'agit donc d'un problème de *complétion de matrices* car les valeurs absentes de la matrice creuses sont des données manquantes et pas des notes nulles.

Les données

Des données réalistes croisant plusieurs milliers de clients et films, sont accessibles en ligne. Il s'agit d'une extraction du site `movielens.org` qui vous aide à choisir un film. Quatre tailles de matrices très creuses sont proposées contenant seulement les appréciations connues d'un client sur un film.

`100k` 100 000 évaluations de 1000 utilisateurs de 1700 films.

`1M` Un million d'évaluations par 6000 utilisateurs sur 4000 films.

`10M` Dix millions d'évaluations par 72 000 utilisateurs sur 10 000 films.

`20M` Vingt deux millions d'évaluations par 138 000 utilisateurs sur 27 000 films.

3.4 Catégorisation de textes

Le problème

Il s'agit d'un problème récurrent du commerce en ligne qui se présente sous la forme suivante. Un commerçant partenaire d'un site en ligne souhaite proposer l'ensemble d'un catalogue d'articles à la vente, chacun décrit par un court texte en langage naturel. Pour assurer le maximum de visibilité des produits, ce site doit assurer une catégorisation homogène des produits malgré leurs origines très variées : les commerçants partenaires. A partir de la liste des articles déjà présents sur le site (base d'apprentissage), il s'agit de déterminer la catégorie d'un nouvel article, c'est-à-dire permettre de l'introduire dans l'arborescence des catégories et sous-catégories du site ; c'est donc encore d'un problème de discrimination ou classification supervisée mais appliquée à de la fouille de textes.

Il s'agit donc d'un problème de fouille de texte qui enchaîne des étapes classiques de nettoyage puis vectorisation ou quantification des textes, de façon à remplacer les mots par des nombres d'occurrences ou plutôt par les valeurs prises par une liste de variables (*features*) mesurant des fréquences relatives d'une liste déterminée de regroupements de mots (TF-IDF). Enfin, une fois construite une matrice, généralement creuse, différentes méthodes d'apprentissage sont testées dans la 3ème étape afin de prévoir, au mieux, la catégorie des articles d'un échantillon test.

Les données

Il s'agit d'une version simplifiée du concours proposé par *Cdiscount* et paru sur le site `datascience.net`. Les données d'apprentissage sont accessibles sur demande auprès de *Cdiscount* dans le forum de ce site et les solutions gagnantes ont été présentées aux journées de Statistique de Montpellier par Gutorbe et al. (2016)[6]. Comme les solutions de l'échantillon test du concours ne sont pas et ne seront pas rendues publiques, un échantillon test est donc extrait pour l'usage de cet exemple. L'objectif est de prévoir la catégorie d'un produit à partir de son descriptif. Seule la catégorie principale (1er niveau de 47 classes) est modélisée au lieu des trois niveaux demandés dans le concours (5789 classes). L'objectif n'est pas de faire mieux que les

solutions gagnantes basées sur des *pyramides* complexes de régressions logistiques programmées en Python mais de comparer les performances des méthodes et technologies en fonction de la taille de la base d'apprentissage ainsi que d'illustrer, sur un exemple réel, le pré-traitement de données textuelles. La stratégie de sous ou sur-échantillonnage des catégories très déséquilibrées qui permet d'améliorer la prévision n'a pas été mise en œuvre.

Les données se présentent sous la forme d'un fichier texte de 3.5 Go. Il comporte quinze millions de lignes, un produit par ligne contenant sa catégorie et le descriptif. La nature brute de ces données, leur volume, permet de considérer toute la chaîne de traitement et pas seulement la partie apprentissage.

D'autres à venir

Références

[1] P. Besse et B. Laurent, *De Statisticien à Data Scientist – Développements pédagogiques à l'INSA de Toulouse*, Statistique et Enseignement **7(1)** (2016), 75–93.

[2] L. Breiman, *Random forests*, Machine Learning **45** (2001), 5–32.

[3] Joan Bruna et S. Mallat, *Invariant Scattering Convolution Networks*, IEEE Transactions on Pattern Analysis and Machine Intelligence **35** (2013), n° 8, 1872–1886, ISSN 0162-8828.

[4] R. Forgy, *Cluster Analysis of Multivariate Data : Efficiency versus Interpretability of Classification*, Biometrics (1965), n° 21, 768–769.

[5] J. H. Friedman, *Data Mining and Statistics. What's the connection ?*, Proc. of the 29th Symposium on the Interface : Computing Science and Statistics, 1997.

[6] B. Goutorbe, Y. Jiao, M. Cornec, C. Grauer et Jakubowicz J., *A large e-commerce data set released to benchmark categorization methods*, Journées de Statistique de Montpellier, Société Française de Statistique, 2016.

[7] Yann LeCun, Léon Bottou, Yoshua Bengio et Patrick Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE, 1998, p. 2278–2324.

[8] Chen Yu Lee, Patrick W. Gallagher et Zhuowen Tu, *Generalizing Pooling Functions in Convolutional Neural Networks : Mixed, Gated, and Tree*, 2016.

[9] Klaus Robert Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda et Bernhard Schölkopf, *An introduction to kernel-based learning algorithms*, IEEE TRANSACTIONS ON NEURAL NETWORKS **12** (2001), n° 2, 181–201.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot et E. Duchesnay, *Scikit-learn : Machine Learning in Python*, Journal of Machine Learning Research **12** (2011), 2825–2830.

[11] N. Pentreath, *Machine Learning with Spark*, Packt puplishing, 2015.

[12] R Core Team, *R : A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2016, http://www.R-project.org.

[13] Patrice Simard, Yann Le Cun, John Denker et Bernard Victorri, *Transformation Invariance in Pattern Recognition - Tangent Distance and Tangent Propagation*, Lecture Notes in Computer Science, Springer, 1998, p. 239–274.

[14] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L. Cun et Rob Fergus, *Regularization of Neural Networks using DropConnect*, Proceedings of the 30th International Conference on Machine Learning (ICML-13) (Sanjoy Dasgupta et David Mcallester, réds.), t. 28, JMLR Workshop and Conference Proceedings, mai 2013, p. 1058–1066.

[15] M. Zaharia, M. Chowdhury, D. Das, A. Dave, J. Ma, M. McCauly, S. Franklin, M. J.and Shenker et I. Stoica, *Resilient Distributed Datasets : A Fault-Tolerant Abstraction for In-Memory Cluster Computing*, Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), USENIX, 2012, p. 15–28.