

Technologies des Grosses Data

Philippe Besse & Brendan Guillouet

Université de Toulouse
INSA – Dpt GMM
Institut de Mathématiques – ESP
UMR CNRS 5219

Définition des *Grosses Données*

- **Volume** : croissance exponentielle
- **Variété** : signaux, images, graphes
- **Vélocité** : décision séquentielle, optimisation stochastique
- **Validité, Valorisation** —> Prévision —> Apprentissage

Confusion de domaines très variés

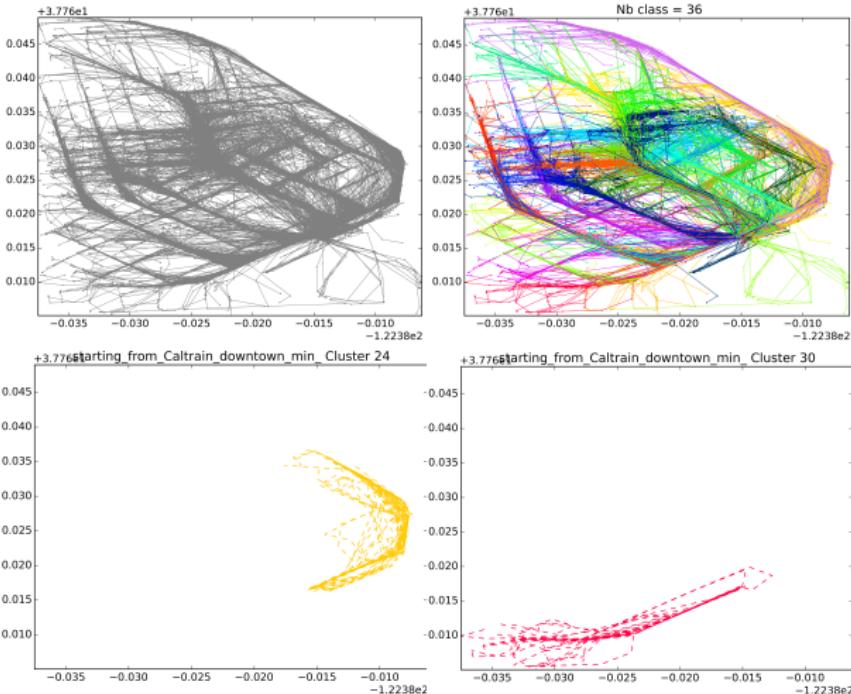
- **E-commerce** : recommandations et **Réseaux sociaux**
- **Publique** : administrations, santé et (*open data*)
- **Recherche** Météo, Biologie, Astronomie...
- **Industrie** : défaillance, fraudes, maintenance...

Introduction

Technologies des données massives
Fonctionnement des ateliers

Définitions, objectifs

Nouvelle Science des Données



Variété : Classification de trajectoires GPS

Réellement massives ?

- **Seuils** technologiques (RAM, Disque)
- **Préparation** (*munging*) des données (Python)
- Données **distribuées** :
- **Hadoop, MapReduce, scalability**

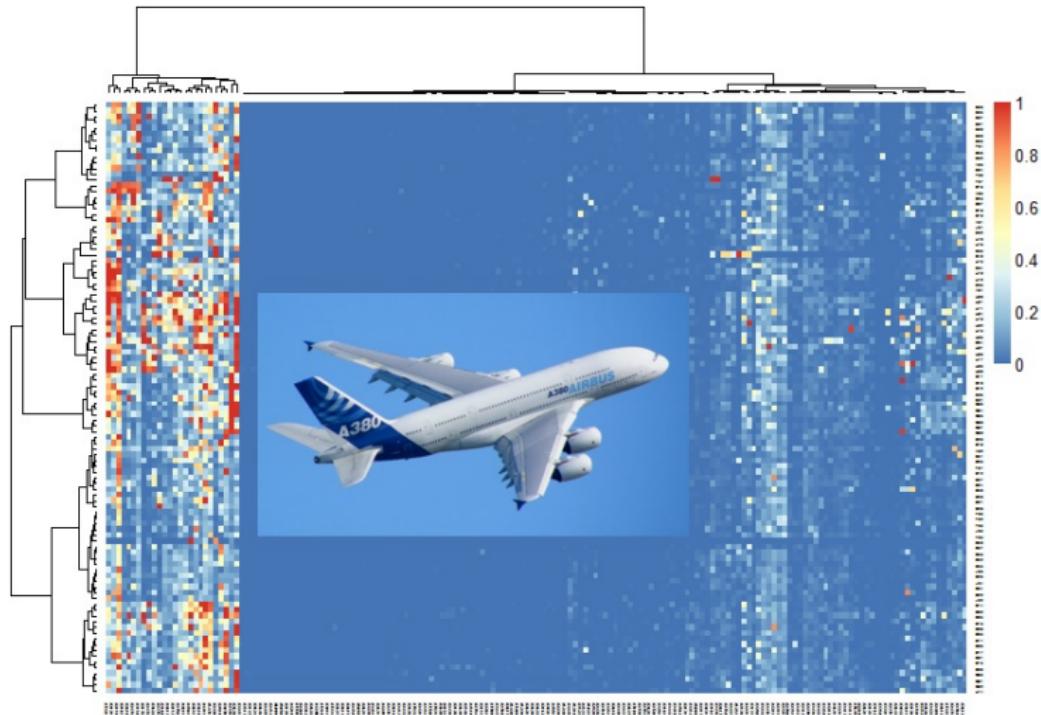


Ferme de données

Objectif

- Apprentissage sur données massives
- Question : Quelles (meilleures) technologies utiliser ?
- Matériel : Poste personnel, GPU, serveur(s), *Cloud*
- Données distribuées (*Hadoop*) ?
- Point de vue du "statisticien"
 - Prototype puis passage à l'échelle du même code
 - bICas d'usage : MovieLens, Cdiscount, MNIST, HAR, Cats/Dogs, Courbes
 - Algorithmes : *munging*, NMF, Logit, RF, SVM, *Gradient boostitng*, *Deep Learning*
 - environnements : R, Python, Spark, XGBoost, TensorFlow, Keras
- Comparaisons de difficiles à impossibles
- Scripts dans <http://github.com/wikistat>





Albus : Analyse des messages d'incidents en vol (700 000 en 6 mois)

Nouvelle (?) Science des Données

- 1995 *Data Mining* : GRC & suites logicielles
- 2010 *Data Science* : Publicité en ligne, *cloud computing*
 - Données **préalables** (fouille), dimensions (omiques) $p \gg n$,
 - Pas de **nouvelles** méthodes, seules celles **échelonnables**
 - **Data** pas toujours *Grosses* mais *datification* du quotidien
 - **Éthique** et virtualisation / transparence de la décision
- "Science" des données : **nouveau** terme d'erreur
 - **Erreur d'optimisation** + Compromis biais / variance
 - **Optimisation** stochastique (Robbins Monro) ;
non différentiable (parcimonie)
 - **Parallélisation** et/ou distribution des calculs

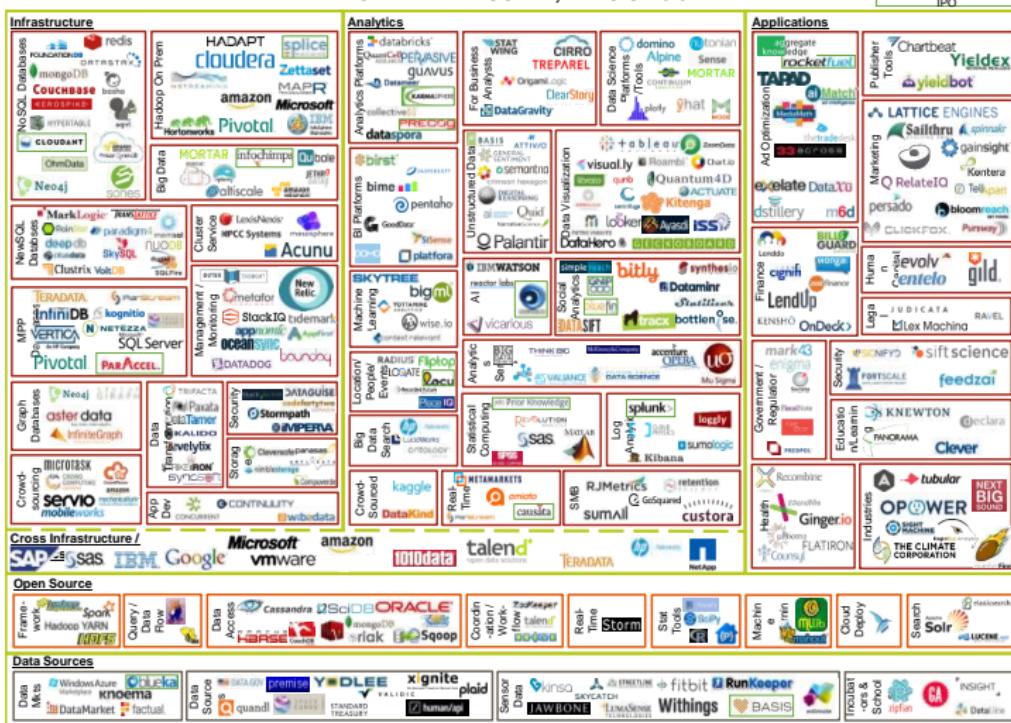
Nouveau modèle économique

- Marges réduites sur matériels (IBM)
- Logiciels sous licence GNU, MIT, Apache... (Microsoft, SAS)
- Vendre du *service* :
 - Enthought ([Canopy](#)), Continuum analytics ([Anaconda](#)),
 - Horton Works, Cloudera ([Hadoop](#), [Spark...](#))
 - Databricks ([Spark](#)), Oxdata ([H2O](#))
 - Revolution Analytics ([RHadoop](#)) – Microsoft
- Nouveau marché du *cloud computing*
 - Platform [aaS](#), Software [aaS](#), Service [aaS...](#)
 - Amazon Web Service
 - Microsoft Azure, Google Cloud Computing
 - IBM Analytics, SAS Advanced Analytics...
- Développement *industriel* vs. Recherche *académique*

Introduction
Technologies des données massives
Fonctionnement des ateliers

Écosystème Hadoop, MapReduce MapReduce pour les nuls Spark

BIG DATA LANDSCAPE, VERSION 3.0

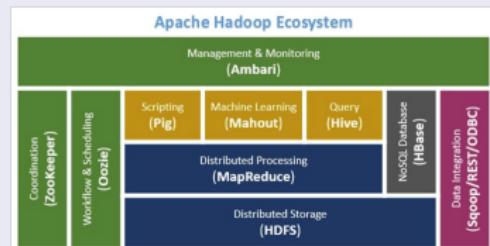


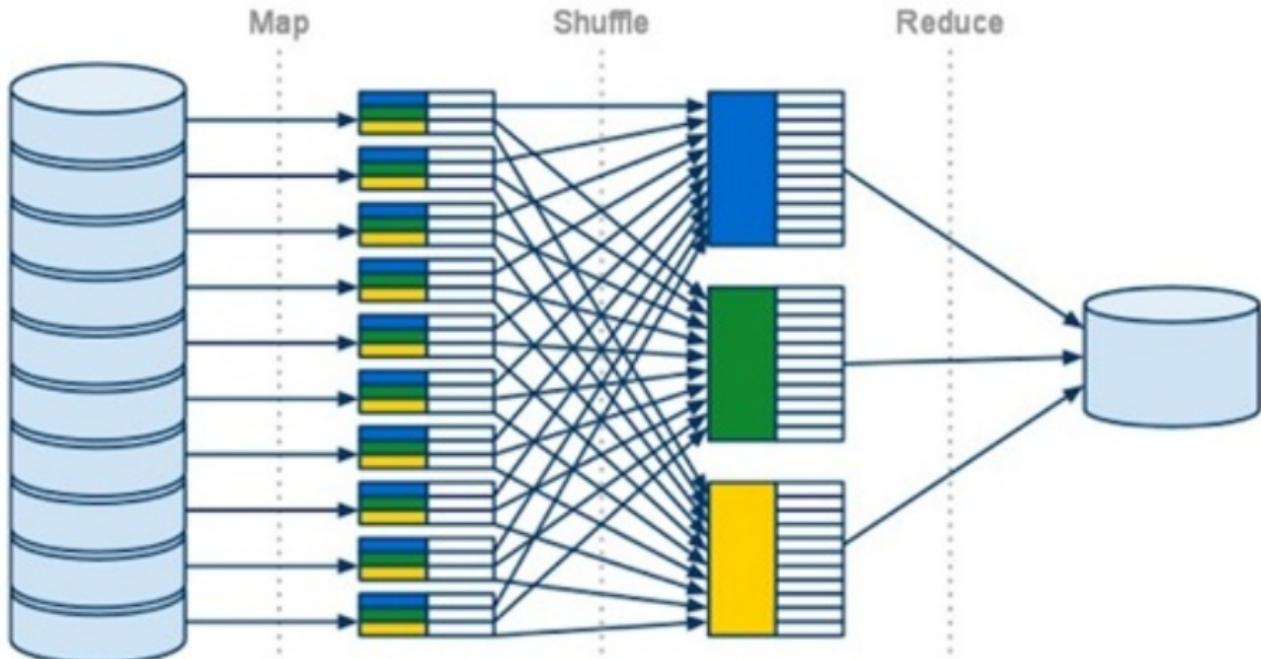
© Matt Turck (@mattturck), Sutian Dong (@sutiandong) & FirstMark Capital (@firstmarkcap)

Écosystème des données massives (Turck, 2014)

Hadoop, MapReduce

- Environnement : *Google* puis *Apache* (2009)
- *Hadoop Distributed File System* (HDFS)
- Données hétérogènes **distribuées**
- Tolérance aux pannes matérielles
- Scalabilité (multiplier les nœuds)
- **Parallélisation** : *Map Reduce*
- Communication par (**clef, valeur**)
- Déplacer les algorithmes, pas les données
- Données *immuables*
- Lecture unique ou *streaming*

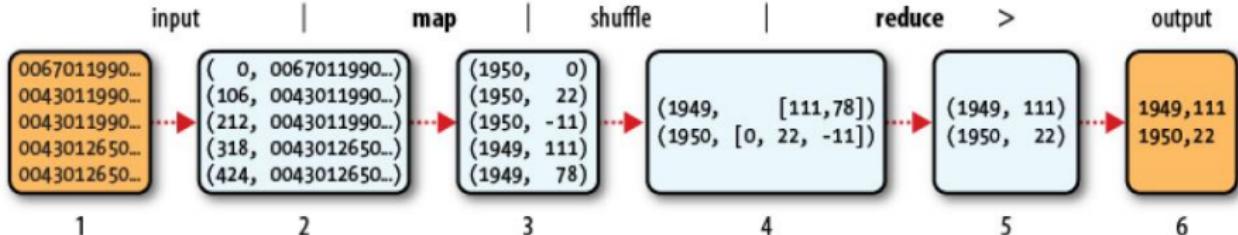




Hadoop Distributed File System (HDFS) & MapReduce

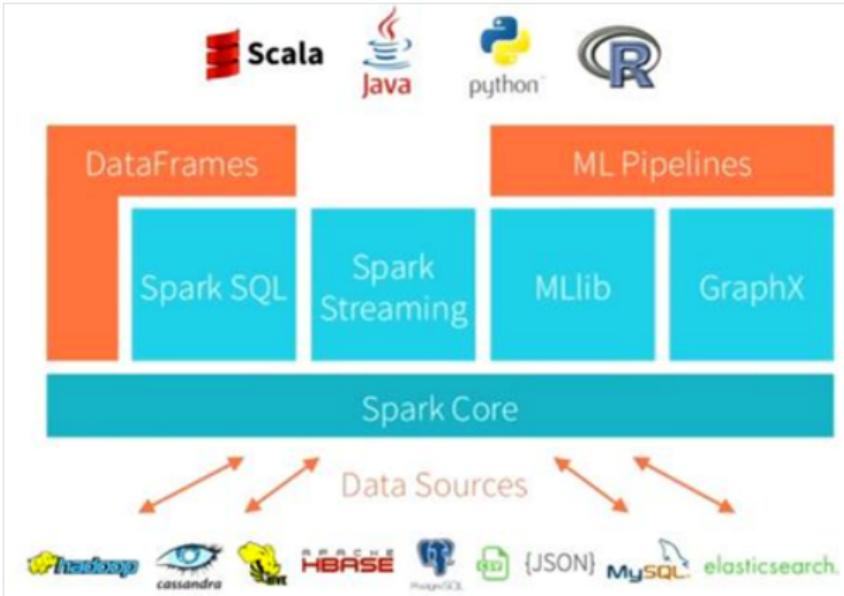
MapReduce

- ① Input \Rightarrow list (k1, v1)
- ② Map () \Rightarrow list (k2, v2)
- ③ *Shuffle, Combine* \Rightarrow (k2, list(v2))
 - Implicite
 - Clefs identiques vers même nœud réducteur
- ④ Reduce () \Rightarrow list (k3, v3)



Classification par centres mobiles ($\approx k\text{-means}$)

- Définition d'une **distance** euclidienne
- Algorithme de **Forgy** (1965)
 - Initialisation des k centres
 - Itération des étapes **MapReduce**
 - **Map** : Affectation de chaque individu (**valeur**) au centre (**clef**) le plus proche
 - **Reduce** : Calcul des **centres** des individus de même **clef**
 - Mise à jour des centres
- **Problème** : accès disques à chaque itération
- **Solution actuelle de Spark** : *Resilient Distributed Dataset*,
(Zaharia *et al.*, 2012)



La technologie **Spark** et son écosystème

Librairie MLlib

- **Spark** 2.2
- **MLlib** (Resilient Distributed Dataset) : *k-means*, SVD, NMF (ALS), Régression linéaire et logistique (l_1 et l_2), SVM linéaires, Classifieur Bayésien Naïf, Arbre, Forêt Aléatoire, Boosting
- Évolution de **SparkML** : *DataFrame*, *pipeline*
- Peu de méthodes mais passage à l'échelle "Volume"

Progression "pédagogique"

- Statistique de **petites** données avec **R**
- Traitement et stat de données plus **grosses** : **Python**
- Traitement et stat de données **distribuées** : **PySpark**
- Autres **algorithmes** avec Python : *XGBoost, deep learning*

Matériels

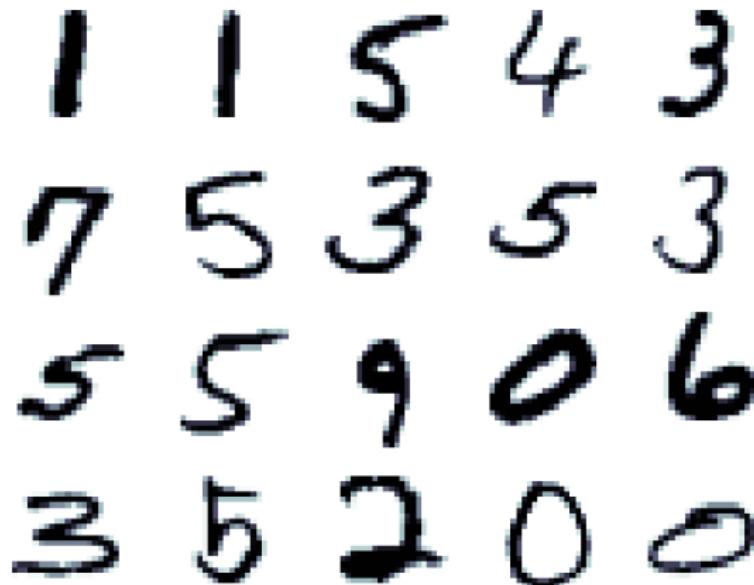
- **Ordinateurs personnels** avec R, Python 2.7 (Anaconda)
- **CSN** : R, Python 2.7 & 3.6, Spark 2 (Hadoop virtuel)
- **GEI** salle 103 avec GPU
- **Hupi.fr** Spark, Maître + 8 exécuteurs (*workers* 7Go de RAM)
- Autres : Amazon Web Services, Google cloud ?

Résultats attendus

- Performance au Défi grosses data 2018
- Soutenance
 - Solution optimale
 - Solution qui passe à l'échelle volume
- Utiliser un *cloud* à partir d'un container de *docker*?

Rappel de l'objectif principal

Apprendre à s'auto-former à des technologies
en perpétuelle (r)évolution



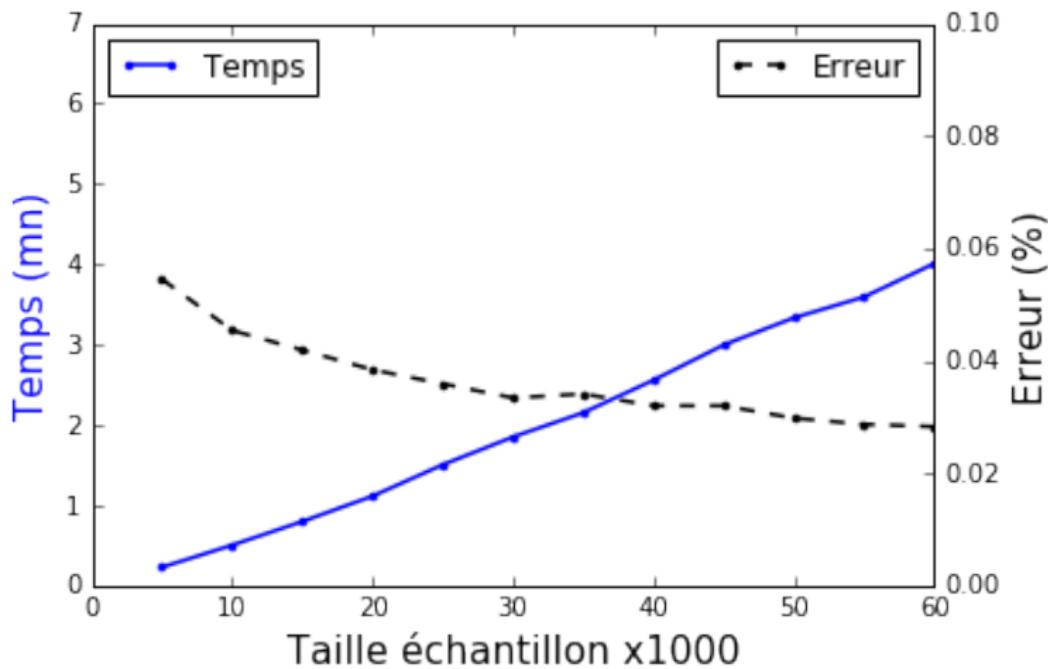
MNIST : quelques exemples d'images de caractères

MNIST : État de l'art

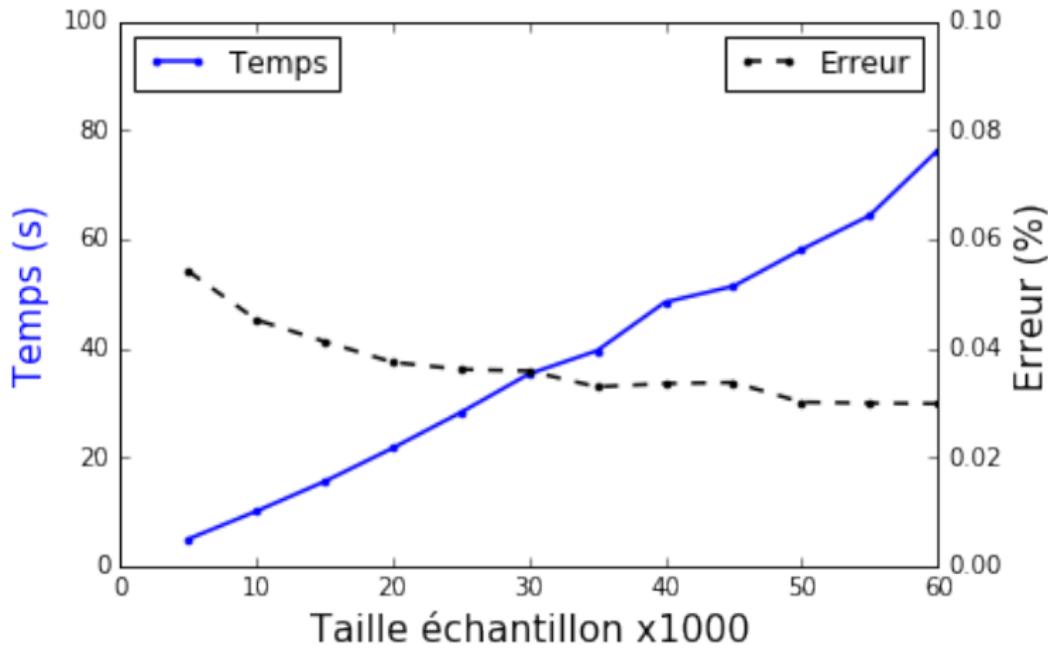
- Site de Yann le Cun
- 60 000 caractères, $28 \times 28 = 784$ pixels
- Test : 10 000 images
- Méthodes classiques (*k*-nn, RF)
- Prétraitement de normalisation des images
- Distance spécifique (tangentielle) avec propriétés d'invariance
- Ondelettes et *scattering* (Stéphane Mallat)
- Apprentissage Profond : *TensorFlow*, *Lasagne*, *Keras*
- ...
- Comparer l'usage des méthodes classiques

Implémentations de Random Forest

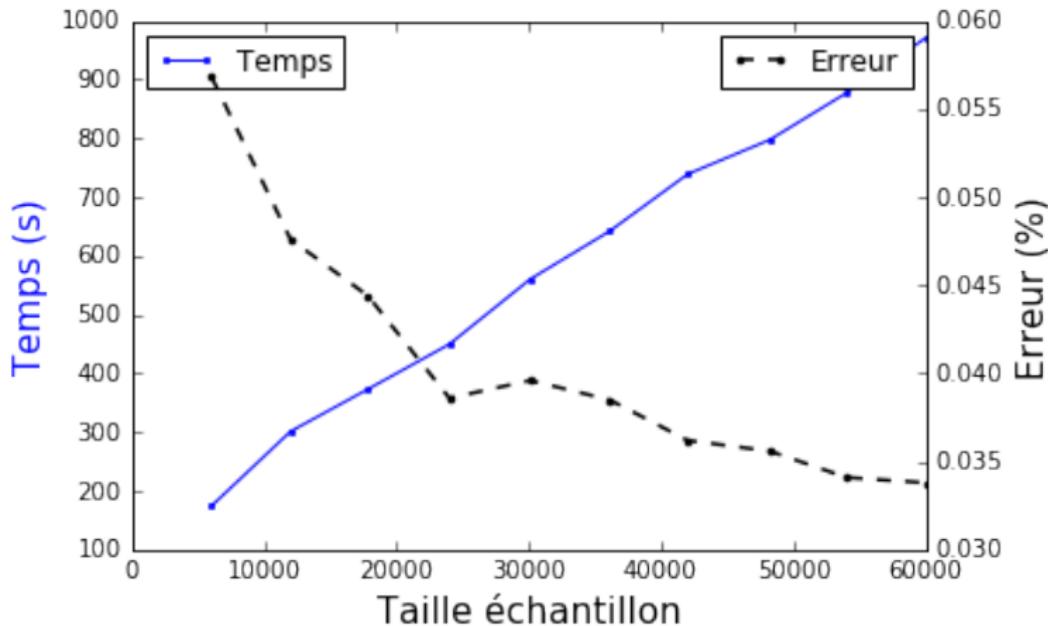
- R : randomForest, ranger
- Python : Scikit-learn
- MLLib : arbres du projet Google PLANET
 - Élagage d'un arbre
 - maxBins = 32
 - maxDepth et problèmes de mémoire
- *Bootstrap* sans remise ?



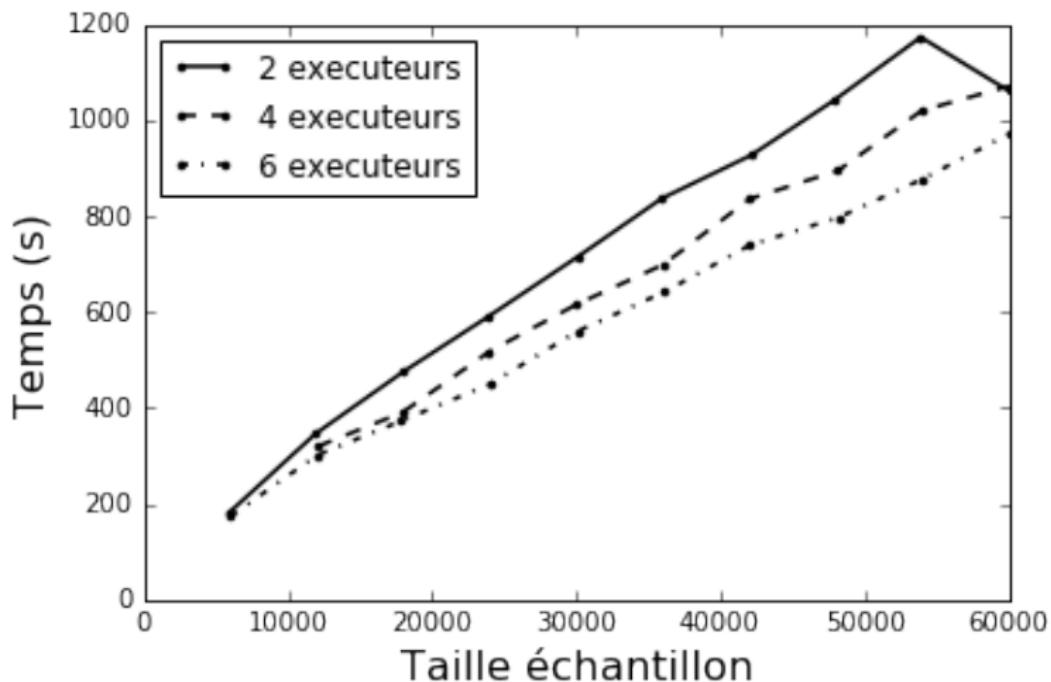
MNIST : forêts aléatoires avec R (ranger pas randomForest)



MNIST : forêts aléatoires avec Python (Scikit-learn)



MNIST : forêts aléatoires avec *Spark (MLlib)*; maxdepth=15



MNIST : Forêts aléatoires (Spark) avec 2, 4 ou 6 exécuteurs

MNIST : discussion

- R ranger (sauf Windows) ou Python Scikit-learn
- Spark : problèmes de mémoire limitent maxDepth, nb arbres
- Spark : *scalability* pas vérifiée
- Plateau de l'erreur fonction de la taille de l'apprentissage
- Architecture intégrée préférable à distribuée

E-commerce et Systèmes de Recommandation

- Gestion de la relation client (score d'appétence)
- *Google double click, Criteo*, enchères
- Commerce en ligne et **filtrage** de produits (Cf. **Vignette**)
 - Filtrage **adaptatif** : algorithme de bandit
 - Filtrage **collaboratif** : concours *Netflix*
 - Basé sur les seules informations **client × produit**
 - **Modèle** de voisinage
 - **Facteurs latents**
 - Méthodes mixtes ou hybrides

Problèmes délicats

- **Évaluation** d'une recommandation :
RMSE, clics, ventes, *AB testing*
- Initialisation : *cold start*

Filtrage collaboratif par voisinage

- Grande matrice creuse **clients** × **produits**
 - **Nombre** ou présence de ventes
 - **Appréciation** ou note de 1 à 5
- **Basé** sur le voisinage client **ou** produit
 - Définir une **similarité** (corrélation linéaire ou rang)
 - Trouver un **sous-ensemble** S_i de proches de i
 - **Prévoir** une note ou un achat
 - **Combinaison** linéaire des notes de S_i

Filtrage collaboratif par facteurs latents

- Matrices clients \times produits très creuses
- Nombre d'achats ou de clics *vs.* appréciation ou note
- Valeur nulle *vs.* valeur manquante
- Modèle par factorisation *vs.* CompléTION
 - Candes et Tao (2010)
 - $\min_{\mathbf{M}} (\|\mathbf{P}_{\Omega}(\mathbf{X} - \mathbf{M})\|_2^2 + \lambda \|\mathbf{M}\|_*)$
 - $\mathbf{P}_{\Omega}(\mathbf{X})$: "restriction" de la matrice \mathbf{X} à l'ensemble des valeurs connues

CompléTION de matrice

- Netflix ou MovieLens : problème de **compléTION**
- Sujet à la bibliographie explosive
- Deux méthodes facilement accessibles (R et Spark)
 - Librairie R **softImpute** : Algorithme hybride associant SVDs seuillées et moindres carrés alternés (Hastie et al. 2015)

$$\min_{\mathbf{A}_{n \times r}, \mathbf{B}_{p \times r}} \|P_\Omega(\mathbf{X} - \mathbf{AB}^T)\|_2^2 + \lambda (\|\mathbf{A}\|_2^2 + \|\mathbf{B}\|_2^2)$$

- Librairie Spark MLLib : **CompléTION** par *Non negative Matrix Factorisation* (NMF)

Non negative Matrix Factorisation

$$\min_{\mathbf{W}, \mathbf{H} \geq 0} [c(\mathbf{X}, \mathbf{WH}) + P(\mathbf{W}, \mathbf{H})]$$

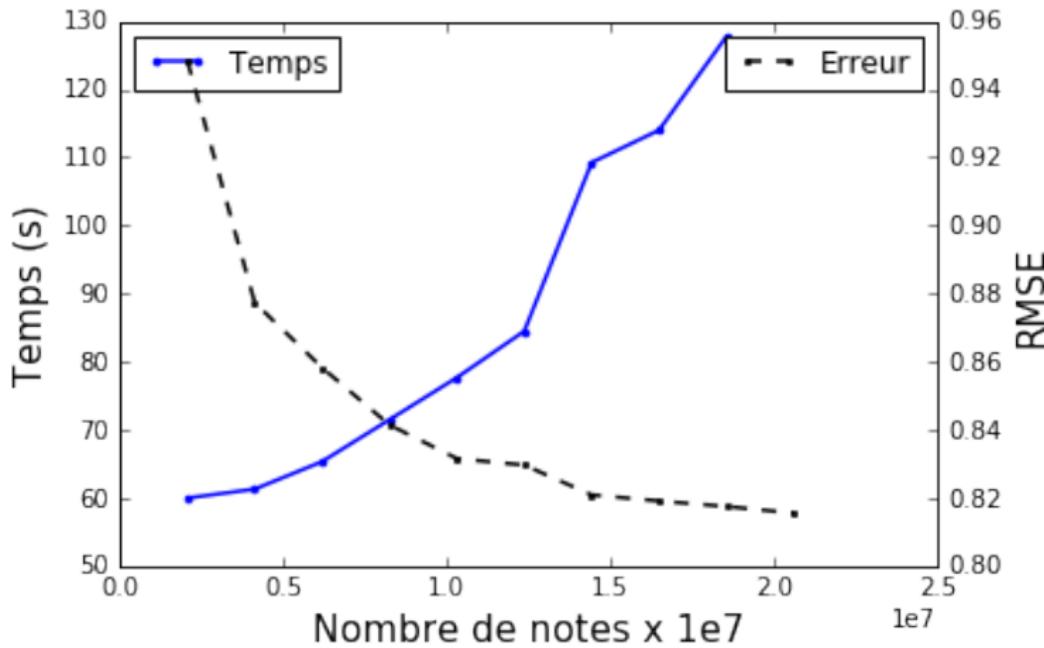
- Similaire à la SVD mais avec contrainte de rang sur \mathbf{W}, \mathbf{H}
- c : moindres carrés ou Kulback Leibler
- P : régularisation
- Nombreux algorithmes dont **ALS**
- Convergence locale
- Nombreuses initialisations disponibles
- Optimiser le **rang** de \mathbf{W} et \mathbf{H}
- Optimiser le coefficient de **régularisation**
- **MLlib** : deux options factorisation ou **complémentation** : $P_\Omega(\mathbf{X})$

Données MovieLens

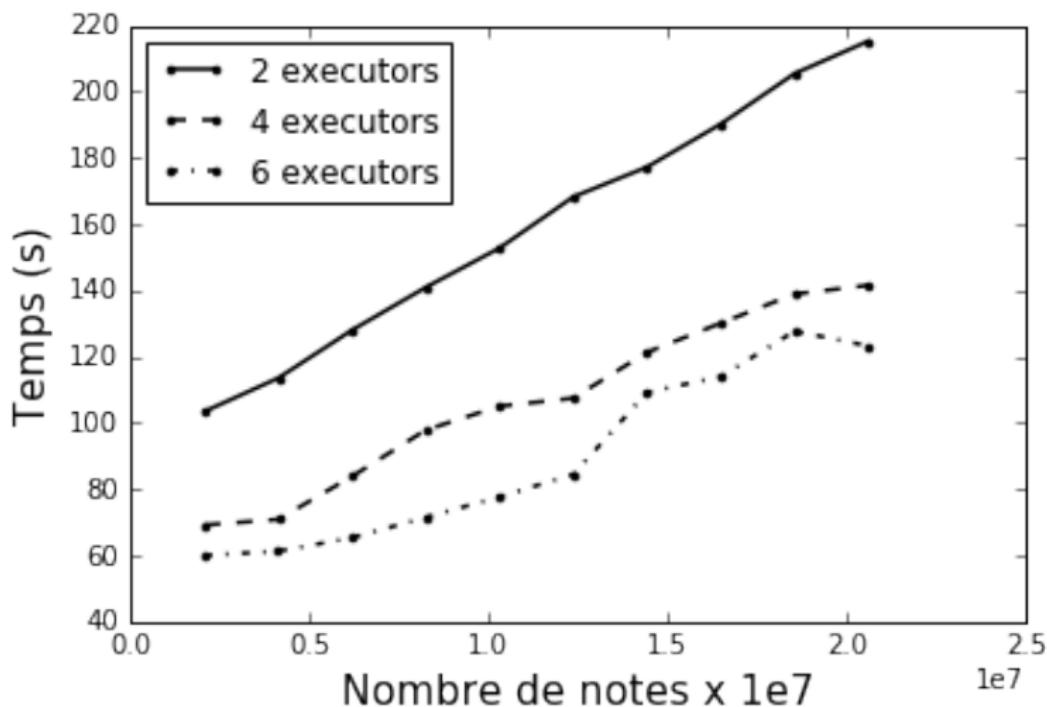
- 100k** 100 000 évaluations de 1000 utilisateurs de 1700 films.
- 1M** Un million d'évaluations par 6000 utilisateurs sur 4000 films.
- 10M** Dix millions d'évaluations par 72 000 utilisateurs sur 10 000 films.
- 20M** Vingt deux millions d'évaluations par 138 000 utilisateurs sur 27 000 films.

Rang Max	λ	Temps	RMSE
4	1	5.6	1.07
10	10	12.6	1.020
10	20	12.2	1.033
15	10	19.4	1.016
20	1	26.9	1.020
20	10	26.1	1.016
20	15	24.4	1.018
20	20	27.0	1.016
30	20	40.1	1.020

MovieLens : Optimisation du rang et de la régularisation de softImpute (ALS)



MovieLens : compléction par NMF (MLlib)



MovieLens : compléTION par NMF (MLlib)

MovieLens : discussion

- NMF mais pas de compléTION dans Scikit-learn
- Moins bon RMSE de softImpute : pas de contrainte ?
- MLlib : pas tout à fait scalable
- Architecture distribuée adaptée aux matrices creuses

Catégorisation de produits (Cdiscount)

- Préparation ou *data munging*
 - Nettoyages (ponctuation, erreurs de code, casse...)
 - Suppression des mots "vides" (*stopwords*)
 - Racinisation (*stemming*) : $\text{card}(\text{Dictionnaire}) = N$
- Vectorisation de (grosses ?) données
 - Hashage (*hashing trick*) : $n_hash < N$

$$i = h(j) \quad h : \{1, \dots, N\} \longmapsto \{1, \dots, n_hash\}$$

- Xgram : *h* appliquée à un mot ou couple (bigram) ou...
- TF-IDF
- Apprentissage

TF-IDF

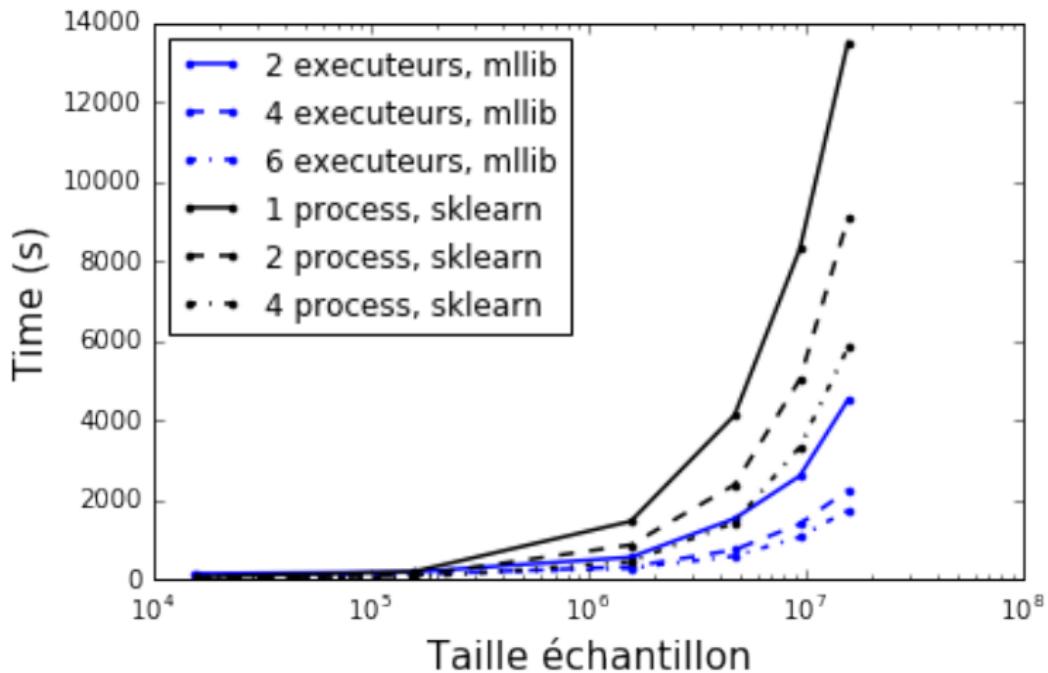
- Importance relative de chaque mot (ou xgram) dans un document par rapport à l'ensemble des documents.
- D : nombre de documents
- $TF(m, d)$: nombre d'occurrences du mot m dans le document d
- $f(m)$: nombre de documents contenant le mot m
- $IDF(m) = \log \frac{D+1}{f(m)+1}$ (version smooth)
- TF-IDF : nouvelles variables ou *features* par pondération des effectifs conjoints :

$$V_m(d) = TF(m, d) \times IDF(m)$$

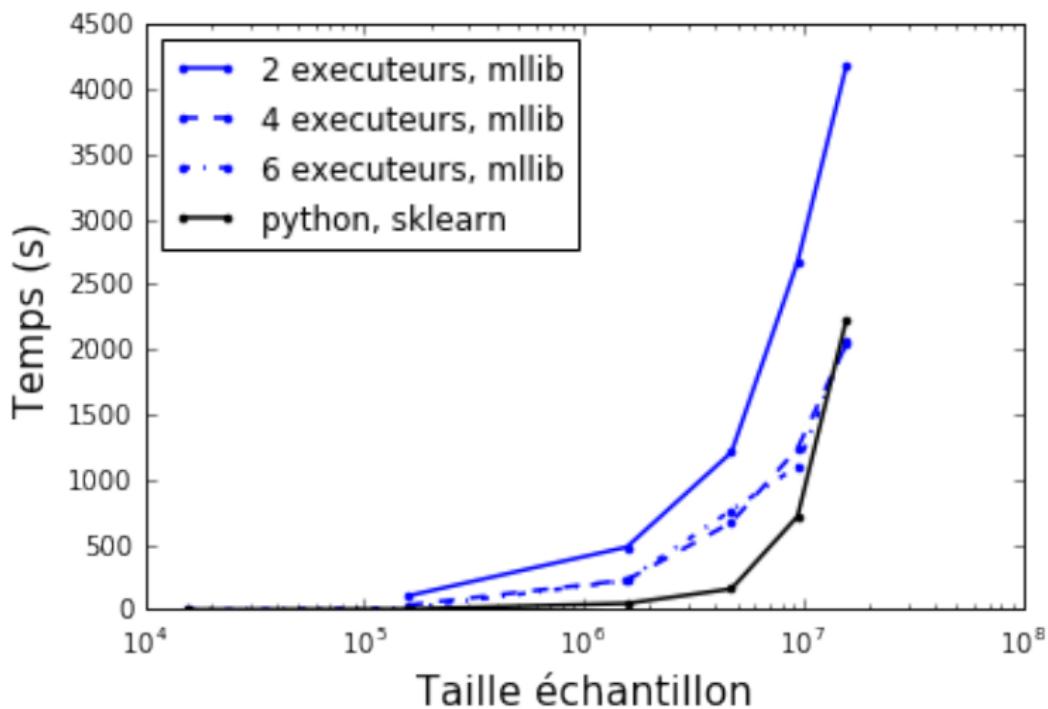
- Même vectorisation (hashage, TF-IDF) sur le test

Cdiscount

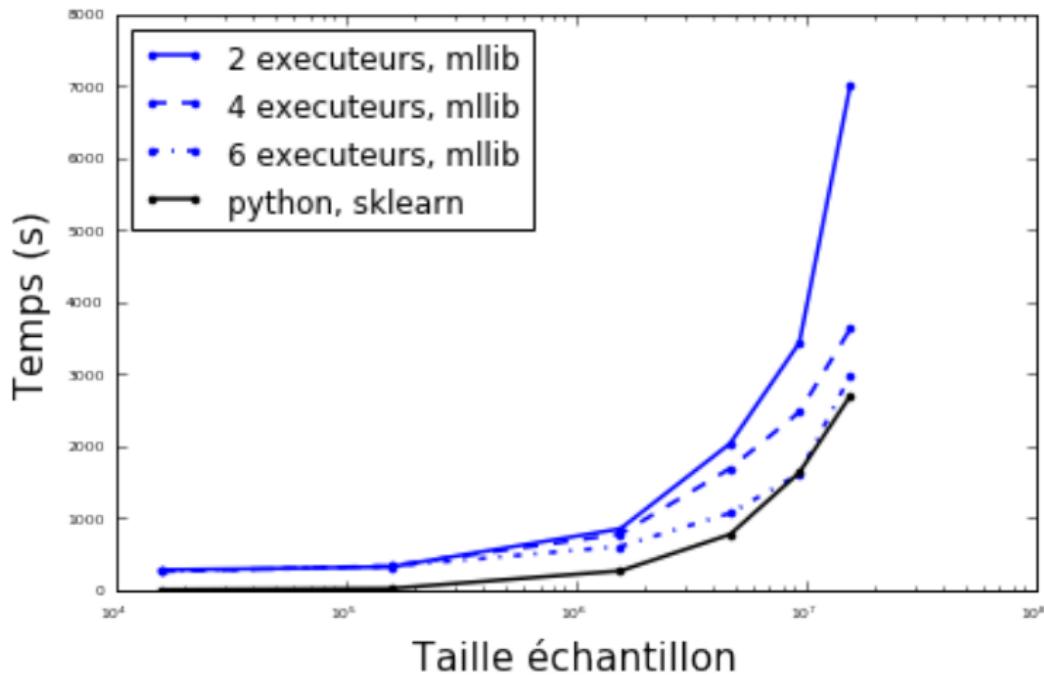
- Données publiques du concours [datascience.net](#)
- **15M** de produits (3.5 Go), 3 niveaux : 5789 classes
- Classes [déséquilibrées](#)
- [Solution gagnante](#) (Goutorbe et al. 2016)
- [Pyramide](#) (Python) de régressions logistiques
- Simplifier : prévoir le [1er niveau](#) : 47 classes
- Comparaison de Python Scikit-learn [vs.](#) SparkML
- [Trois phases](#) : Nettoyage, Vectorisation, Apprentissage



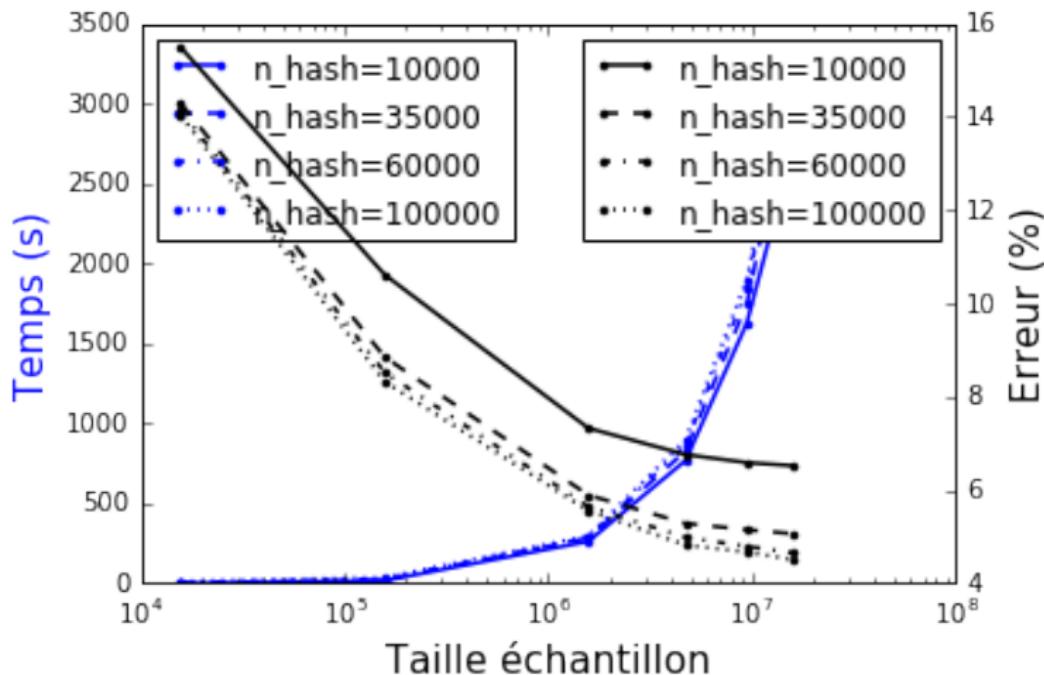
Cdiscount : nettoyage des données



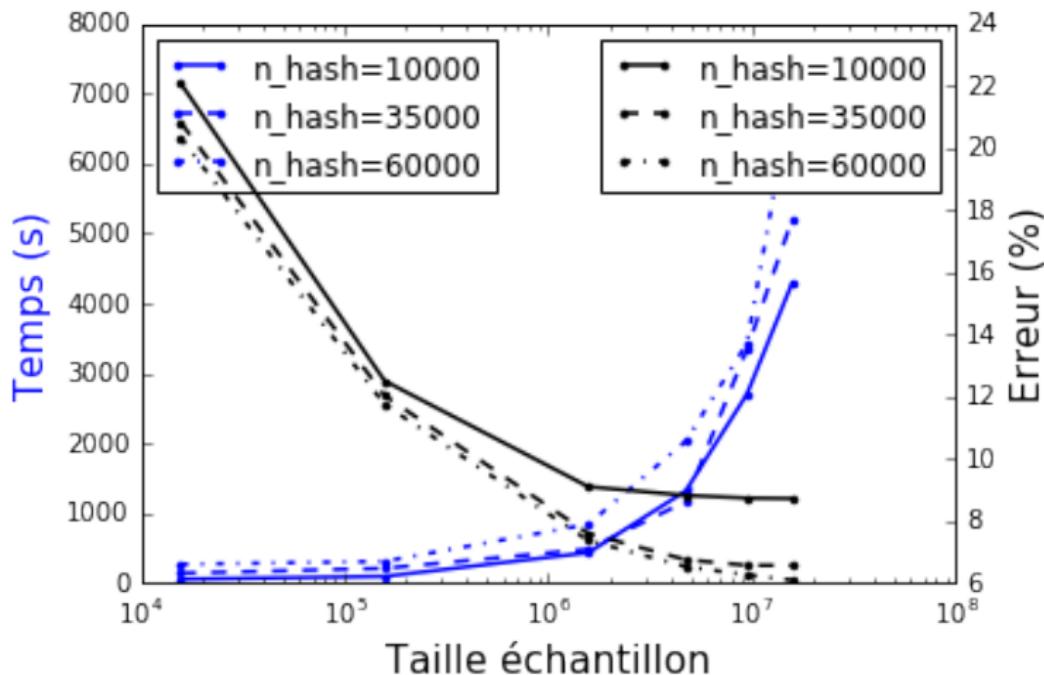
Cdiscount : vectorisation avec `n_hash` = 60 000



Cdiscount : apprentissage avec Spark, Python (Scikit-learn) et n_hash = 60 000



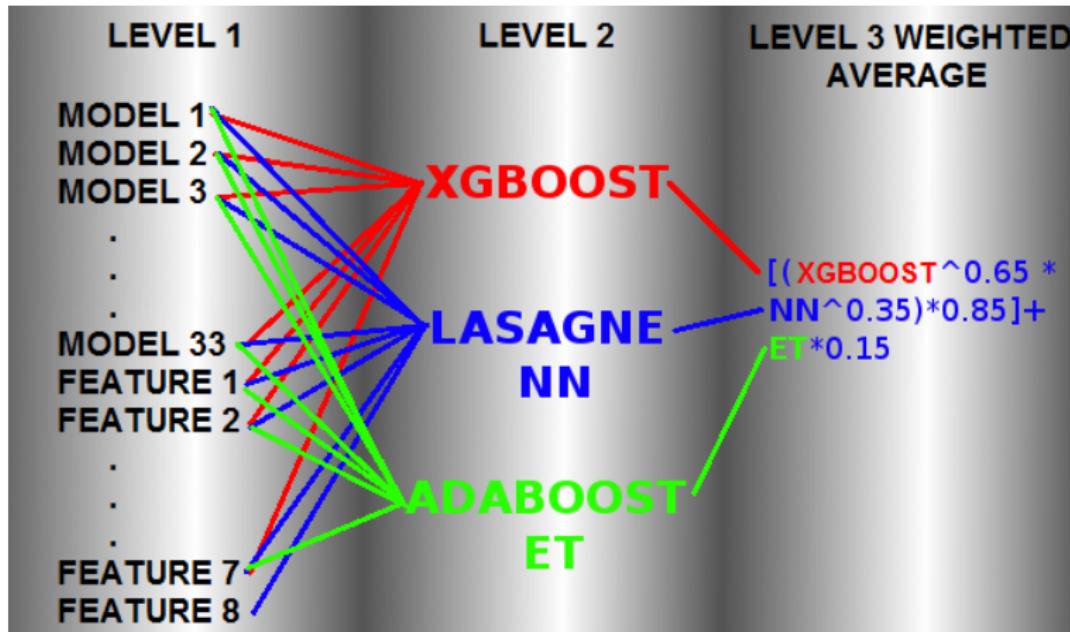
Cdiscount : Apprentissage avec Python Scikit-learn



Cdiscount : Apprentissage avec Spark MLlib

Conclusion

- Comparaison entre architectures distribuée *vs.* intégrée
- Problème de maturation des technologies
- Trois étapes :
 - *Data munging, streaming* : Spark, SparkSQL
 - *Vectorisation* : Python, Scikit-learn, Lucene...
 - *Apprentissage* : grosses data et gros modèles
- R *vs.* Python Scikit-learn *vs.* SparkSQL, SparkML
- Nettoyage et Apprentissage *en ligne* ?
- *But?* Publication, Concours, Industriel
- Cdiscount, Critéo, Deepky, Tinyclues, Hupi (ppml)...
- *Ne pas oublier* : fiabilité, représentativité des données



Exemple de concours Kaggle : Identify people who have a high degree of Psychopathy based on Twitter usage.