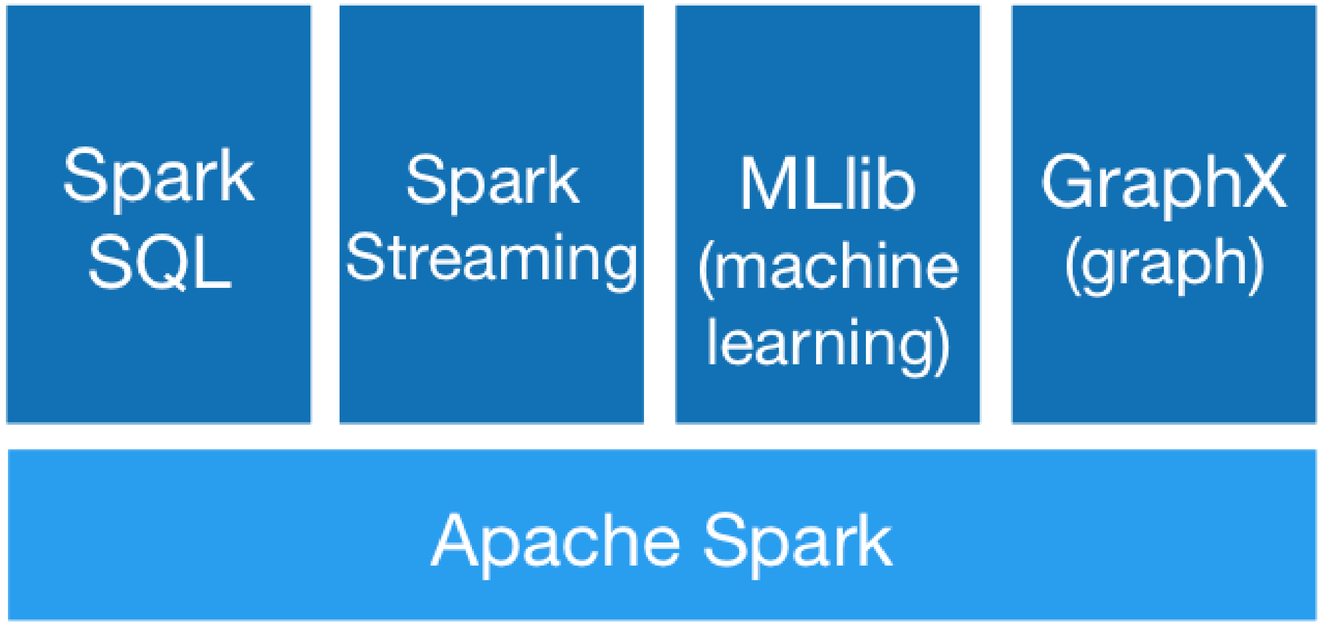
Big Data Analytics

« Pilotage de la performance pour une bonne gouvernance des entreprises »

CHAPITRE 5 – Spark MLlib

Spark Stack



- ***Spark SQL :*** pour le traitement de données (SQL et non structuré)
- ***Spark Streaming :*** traitement de flux de données en direct (live streaming)
- ***MLlib :*** Algorithmes Machine Learning
- ***GraphX :*** Traitement de graphes

Classification supervisée / non supervisée

Classification supervisée

- L'apprentissage/classification supervisé consiste à apprendre une fonction de prédiction à partir d'exemples annotés

Principe : On dispose d'éléments déjà classés
Exemple : articles en rubrique cuisine, sport, culture…
On veut classer un nouvel élément
Exemple: lui attribuer un nom parmi cuisine, sport, culture.

Applications : Vision par ordinateur, Reconnaissance de formes, Reconnaissance de l'écriture manuscrite, Reconnaissance vocale, Traitement automatique de la langue, Bio-informatique, …

Intelligence artificielle

- **Supervisé** : apprendre à classer un nouvel individu parmi un ensemble de classes prédéfinies: on connaît les classes à priori.
- **Non-supervisé :** le nombre et la définition des classes n'étant pas données à priori

Classification non supervisée

- L'apprentissage/classification non supervisé consiste à apprendre sans superviseur.
- Il s'agit d'extraire des classes ou groupes d'individus présentant des caractéristiques communes.

Principe : On dispose d'éléments non classés
Exemple : une fleur
On veut les regrouper en classes
Exemple: si deux fleurs ont la même forme, elles sont en rapport avec une même plante correspondante

Applications : le partitionnement de données, l'estimation de densité de distribution, …

Algorithmes Machine Learning

think big data

regression

Ordinary Least Squares Regression (OLSR)
Linear Regression
Logistic Regression
Stepwise Regression
Multivariate Adaptive Regression Splines (MARS)
Locally Estimated Scatterplot Smoothing (LOESS)
Jackknife Regression

bayesian

Naive Bayes
Gaussian Naive Bayes

deep learning

Deep Boltzmann Machine (DBM)
Deep Belief Networks (DBN)
Convolutional Neural Network (CNN)
Stacked Auto-Encoders

neural networks

Self Organizing Map
Perceptron
Back-Propagation
Hopfield Network
Radial Basis Function Network (RBFN)
Backpropagation
Autoencoders
Hopfield networks
Boltzmann machines
Restricted Boltzmann Machines
Spiking Neural Networks
Learning Vector quantization (LVQ)

associated rule

Apriori
Eclat
FP-Growth

ensemble

Logit Boost (Boosting)
Bootstrapped Aggregation (Bagging)
AdaBoost
Stacked Generalization (blending)
Gradient Boosting Machines (GBM)
Gradient Boosted Regression Trees (GBRT)
Random Forest

...and others

Support Vector Machines (SVM)
Evolutionary Algorithms
Inductive Logic Programming (ILP)
Reinforcement Learning (Q-Learning, Temporal Difference, State-Action-Reward-State-Action (SARSA))
ANOVA
Information Fuzzy Network (IFN)
Page Rank
Conditional Random Fields (CRF)

esprit

Familles d'algorithmes en ML

- **Régression :**
  - Ce type d'algorithmes supervisés va trouver une valeur continue (un nombre réel) qui est la prédiction de la valeur d'une nouvelle observation donnée.
  - Exemple : prédiction de prix de maison en fonction de ses caractéristiques (superficie, nb pièces, ...).
- **Classification :**
  - Ces algorithmes vont classifier une donnée dans une catégorie. Il s'agit également d'un algorithme supervisé.
  - Exemple : classifier un mail en Spam ou non, décider si une tumeur est maligne ou bénigne.
- **Clustering :**
  - Il s'agit d'une famille d'algorithmes non supervisés. Par conséquent, les données n'ont pas d'étiquettes (Non-labeled Data). L'algorithme va regrouper les données par similarité.
  - Exemple : on fournit un ensemble de photos de plantes (sans qu'on dise de quels plantes il s'agit). L'algorithme va regrouper les photos des roses ensemble et celle des tulipes ensemble etc...

NB : on considère que les problèmes de prédiction d'une variable quantitative sont des problèmes de régression tandis que les problèmes de prédiction d'une variable qualitative sont des problèmes de classification.

Spark MLlib, c’est quoi ?

- Spark MLlib est un sous projet de Spark
- Contribution initiale AMPLab, UC Berkeley,
- Livré avec Spark depuis la version 0.8.
- Algorithmes :
  - ***Classification :*** logistic regression, linear support vector machine (SVM), naive Bayes
  - ***Régression :*** generalized linear regression (GLM)
  - ***Collaborative filtering :*** alternating least squares (ALS)
  - ***Clustering :*** k-means
  - ***Décomposition :*** singular value decomposition (SVD), principal component analysis (PCA)

Spark MLlib, pourquoi ?

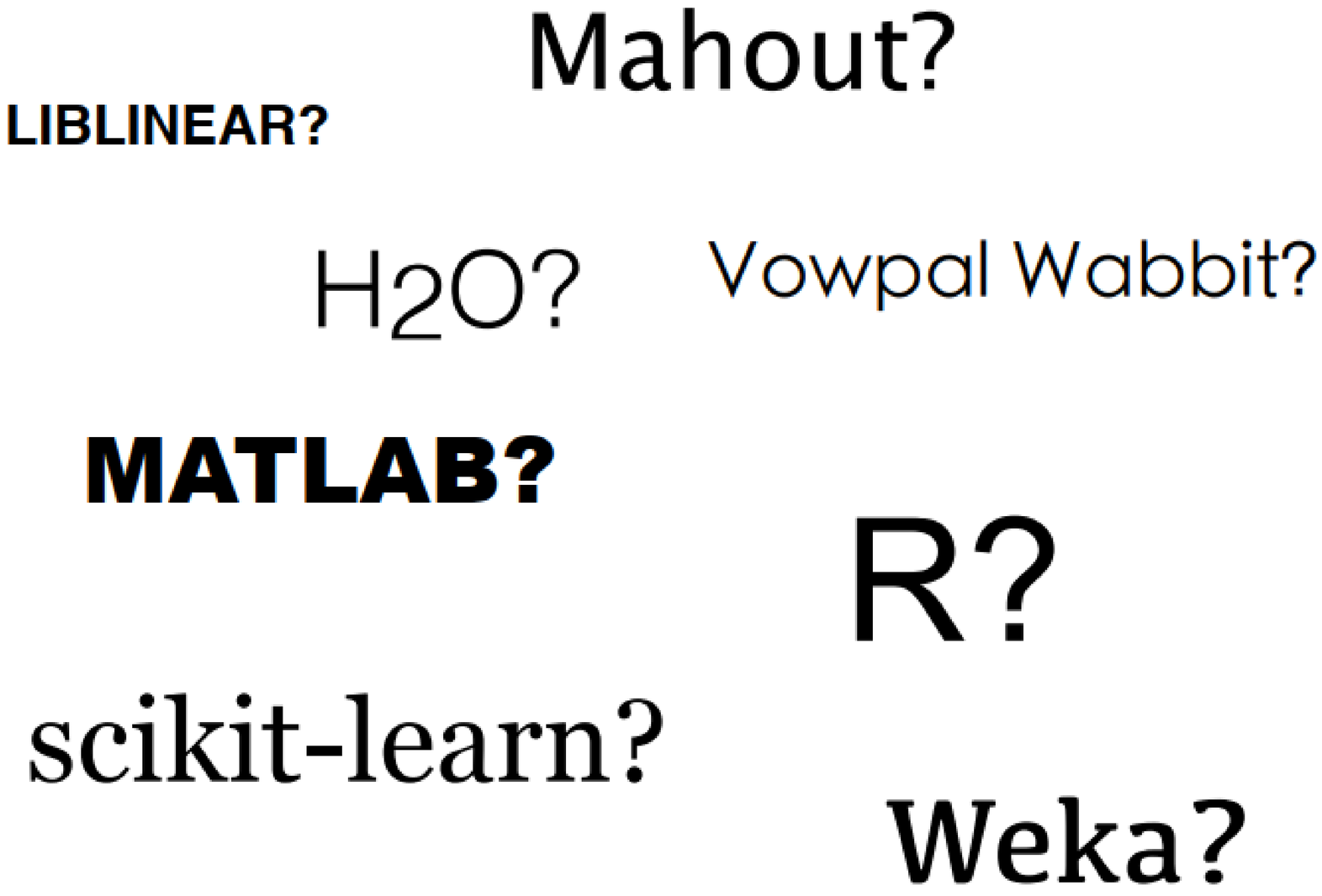
scikit-learn ?

- Classification: SVM, nearest neighbors, random forest, ...
- Régression: support vector regression (SVR), ridge regression, Lasso, logistic regression, ...
- Clustering: k-means, spectral clustering, ...
- Décomposition: PCA, non-negative matrix factorization (NMF), independent component analysis (ICA), ...

Mahout ?

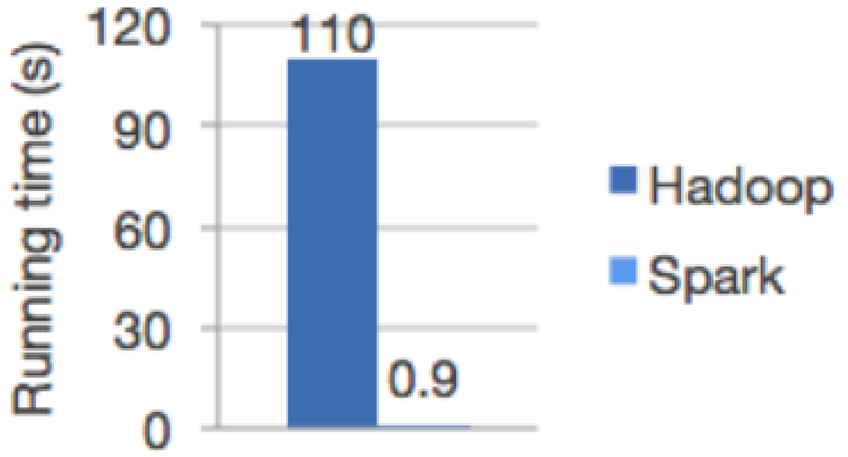
- Classification: logistic regression, naive Bayes, random forest, ...
- Collaborative filtering: ALS, ...
- Clustering: k-means, fuzzy k-means, ...
- Décomposition: SVD, randomized SVD, ...

Spark MLlib, pourquoi ?



Spark MLlib, pourquoi ?

- Repose sur Apache Spark, un moteur rapide pour le traitement de données à grande échelle :
  - Exécute des programmes 100 fois plus rapidement que Hadoop MapReduce en mémoire ou 10 fois plus vite sur le disque.
- Possibilité d'écrire rapidement des applications en Java, Scala ou Python.
- MLlib est comparable, voire meilleur, que d'autres bibliothèques spécialisées dans l'apprentissage automatique à grande échelle.
- Spark Mllib :
  - Evolutivité
  - Performance
  - APIs User-friendly
  - Intégration avec Spark



k-means (scala)

```scala
// Load and parse the data.
val data = sc.textFile("kmeans_data.txt")
val parsedData = data.map(_.split(' ').map(_.toDouble)).cache()
// Cluster the data into two classes using KMeans.
val clusters = KMeans.train(parsedData, 2, numIterations = 20)
// Compute the sum of squared errors.
val cost = clusters.computeCost(parsedData)
println("Sum of squared errors = " + cost)
```