Modèle de Biclustering dans un paradigme "Mapreduce"

Tugdual Sarazin* Hanane Azzag * Mustapha Lebbah *

* Université Paris 13, LIPN UMR 7030
99, avenue Jean-Baptiste Clément
93430 Villetaneuse, France
sarazin.tugdual, hanane.azzag, mustapha.lebbah@lipn.univ-paris13.fr

1 Introduction

Le BiClustering consiste à réaliser un clustering simultanément sur les observations et les variables. Govaert et al ont introduit une adaptation de l'algorithme k-means au biclustering nommée "Croeuc" qui permet de découvrir tous les biclusters en même temps. Dans Labiod et Nadif (2011), les auteurs ont proposés une approche de factorisation CUNMTF, qui généralise le concept de la NMF Lee et Seung (1999). D'autres modèles probabiliste de biclustering sont proposés dans Govaert et Nadif (2008). Le Biclustering a de nombreuses applications et devient un challenge de plus en plus important avec l'augmentation des volumes de données. Cependant les bons algorithmes de clustering sont encore extrêmement utiles, il est donc nécessaire de les adapter aux nouvelles architectures massivement distribuées utilisant le paradigme MapReduce.

Le paradigme MapReduce Dean et Ghemawat (2008) et l'écosystème qui en découle sont actuellement l'une des solutions les plus adaptées au traitement des larges volumes de données. Ce modèle de programmation est simple, il permet au développeur de s'affranchir des problématiques de gestion de la mémoire et de communication entre les processus/machines. Le développement d'un programme MapReduce consiste à écrire une ou plusieurs fonctions primitives Map et Reduce. Les fonctions de Map servent généralement à extraire l'information utile d'une partie des données (phrase d'un texte, vecteur d'une matrice, ...). Les fonctions de Reduce sont généralement utilisées pour agréger les données en sortie de la fonction de Map. Les fonctions de Map et Reduce peuvent être exécutées de façon autonomes sur toutes les machines du cluster simultanément. L'architecture MapReduce se charge de leur répartition et du transfert des données à l'entrée et à la sortie des fonctions. Hadoop [1] est sûrement l'une des architectures MapReduce les plus populaire pour le traitement des gros volumes de données. Bien qu'Hadoop est démocratisé et simplifié les problématiques de traitement et d'analyse des gros volumes ça reste une méthode peu performante dans le domaine de l'apprentissage automatique. En effet la majorité des algorithmes d'apprentissage - et plus particulièrement les algorithmes de Clustering - lisent plusieurs fois les données afin d'optimiser un paramètre or cette lecture des données est assez lente sur une architecture Hadoop. Le framework de traitement de données distribué Spark [2] Zaharia et al. (2010) résout ce problème en permettant

1. www.hadoop.com
2. http ://spark-project.org/