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/

de stocker les données en mémoire. Lors de l'exécution d'un algorithme d'apprentissage les données sont chargées en mémoire (depuis le disque) dès le début de l'apprentissage, par la suite elles seront relues depuis la mémoire. Ainsi, les gains de temps sont très significatifs lors des nombreuses itérations.

Dans ce papier nous proposons une nouvelle approche globale de biclustering basé sur les cartes auto-organisatrices et le calcul distribué. Le modèle de biclustering BiTM (Biclustering using Topological Maps) a deja donnée lieu à une publication dans Chaibi et al. (2014). Dans ce papier nous proposons une adaptation de ces travaux a l'architecture MapReduce avec une implémentation de BiTM sous Spark, une technologie open source de calcul distribué incluant plusieurs paradigmes de programmation. Les principales problématiques abordées dans ce papier sont la minimisation de la fonction et la taille des données en entrée et en sortie des fonctions primitive (Map et Reduce) d'un algorithme de biclustering topologique.

Références

Chaibi, A., H. Azzag, et M. Lebbah (2014). Pondération de blocs de variables en bi-partitionnement topologique. In *14èmes Journées Francophones Extraction et Gestion des Connaissances, EGC 2014, Rennes, France, 28-32 Janvier, 2014*, pp. 317–328.

Dean, J. et S. Ghemawat (2008). Mapreduce : simplified data processing on large clusters. *Commun. ACM 51*(1), 107–113.

Govaert, G. et M. Nadif (2008). Block clustering with Bernoulli mixture models : Comparison of different approaches. *Computational Statistics and Data Analysis 52*, 3233–3245.

Labiod, L. et M. Nadif (2011). Co-clustering under nonnegative matrix tri-factorization. In *Proceedings of the 18th international conference on Neural Information Processing - Volume Part II*, ICONIP'11, Berlin, Heidelberg, pp. 709–717. Springer-Verlag.

Lee, D. D. et H. S. Seung (1999). Learning the parts of objects by nonnegative matrix factorization. *Nature 401*, 788–791.

Zaharia, M., M. Chowdhury, M. J. Franklin, S. Shenker, et I. Stoica (2010). Spark : cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, HotCloud'10, Berkeley, CA, USA, pp. 10–10. USENIX Association.

Summary

Biclustering is a main task in a variety of areas of machine learning providing simultaneous observations and features clustering. Biclustering approches are more complex compared to the traditional clustering particularly those requiring large dataset and Mapreduce platforms. We propose a new approach of biclustering based on popular self-organizing maps for cluster analysis of large dataset. We have designed scalable implementations of the new biclustering algorithm using MapReduce with the Spark platform. We report the experiments and demonstrated the performance public dataset using different cores. Using practical examples, we demonstrate that our algorithm works well in practice. The experimental results show scalable performance with near linear speedups across different data and 120 cores.