2013 AMALTHEA REU Program

Large-scale Clustering for Big Data Analytics:

A MapReduce Implementation of Hierarchical Affinity Propagation

Dillon Mark Rose[1], Jean Michel Rouly[2], Rana Haber[1], Nenad Mijatovic[1], Adrian M. Peter[1]

[1]Florida Institute of Technology, [2]George Mason University

Problem Statement

Clustering algorithms are a fundamental part of Machine Learning (ML), used to extract the underlying structure of unknown datasets. ML has the potential to provide meaningful insight for large datasets. However, many traditional implementations of clustering algorithms are hindered because they are inefficient and incapable of handling Big Data. Thus there is a need within the ML community to develop massively scalable and computationally efficient implementations.

Objective

To develop a robust data mining framework in a cloud-computing environment, capable of processing massive data quantities for the extraction of actionable intelligence.

At a Glance

Acquire Data
- Ingest Massive Data Volumes

Map to Mathematical Representation
- Map Data Content to Numerical Constructs

Apply Machine Learning Analytics
- Extract Latent Patterns and Actionable Intelligence

Visualize
- Presentation of Information in a Succinct and Meaningful Manner

Hierarchical Affinity Propagation

Hierarchical Affinity Propagation is an efficient, parallelizable exemplar-based clustering algorithm, used to extract the underlying structure from an unlabeled dataset.

Objective: Select exemplars in order to maximize the similarities between every data point in a cluster and that cluster's exemplar.

Algorithm: Hierarchical Affinity Propagation

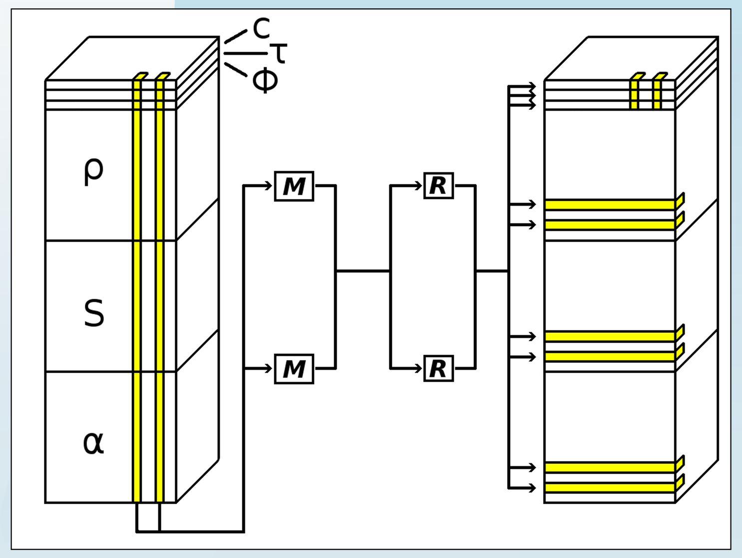
1: **Input:** Similarity, Levels, Iterations, λ
2: **Initialize:** Messages, Preferences
3: **for** $iter = 1 \to$ Iterations **do**
4:     Update Responsibilty
5:     Update Availability
6:     Update Inter-Level Messages
7:     Update Cluster Preferences
8:     **Optional** Update Similarity
9: **end for**
10: Extract Cluster Assignments

MapReduce Design and Development

Our methodology for parallelizing Hierarchical Affinity Propagation (HAP) in MapReduce was motivated by viewing the major update equations for HAP as tensorial mathematical constructs. The HAP algorithm can be parallelized because all updates to the various tensors require only a subset of the total information provided. Therefore, the updates can be split into parallel jobs where each job receives the subset of data it needs to evaluate the update.

In our parallelization scheme, HAP is broken down into three separate MapReduce jobs. The first job handles updating ρ, c, and τ. The second job handles updating α and φ. These first two jobs loop for a set number of iterations. At the end of the iterations, the final job extracts the cluster memberships on each level.

In the figure on the right, the tensors have been stacked to show how the indices line up in the parallelization scheme. The yellow strips on the left represent information being passed to mappers, one strip per mapper. The information is then passed through reducers. The resulting output is now ready for use by the next job.



Parallelization Scheme

Comparisons / Benchmarking

Hierarchical Affinity Propagation vs. Hierarchical K-Means

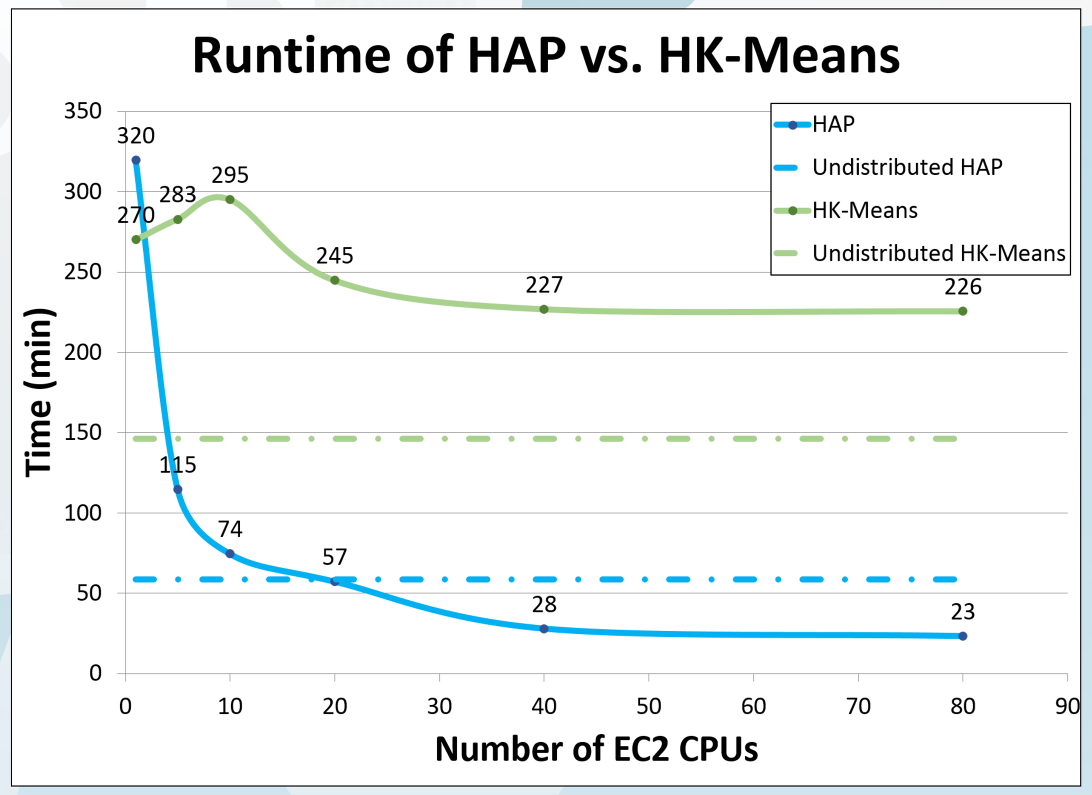
Benchmark	HAP	HK-Means
Small† Cluster Runtime	320m	270m
Big‡ Cluster Runtime	23m	226m
Runtime Speedup (minutes)	300m	45m
Runtime Speedup (%)	94%	16%
Undistributed Runtime	59m	146m
Runtime Plateau	20m	225m

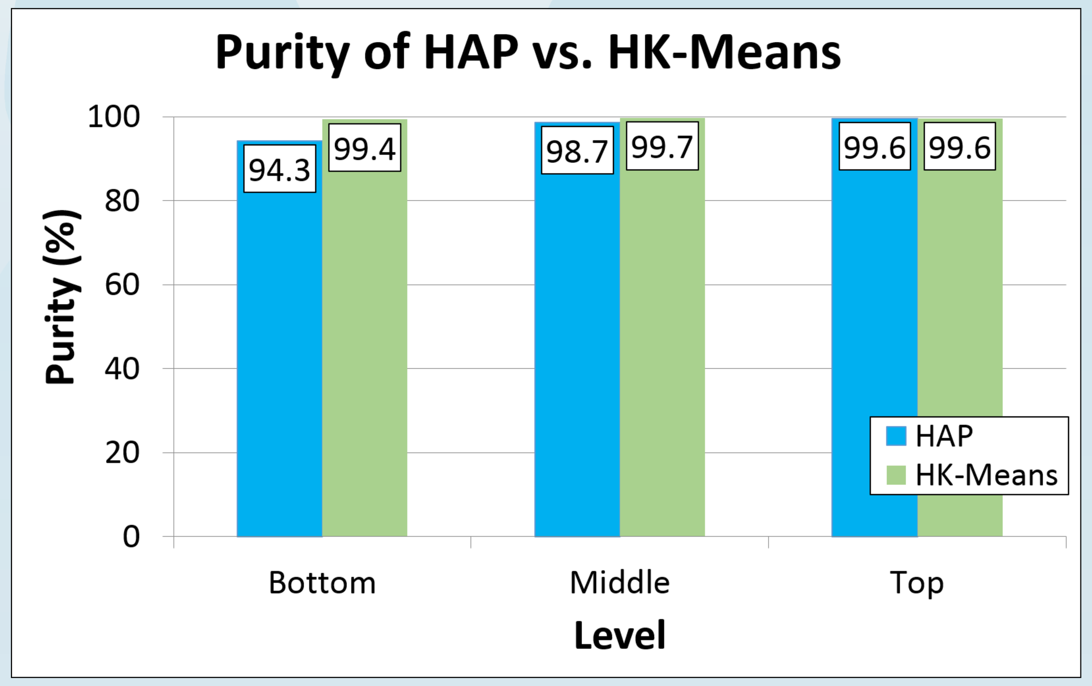
†: 1 AWS ECU, ‡: 80 AWS ECU

It is apparent from the table above and the figures to the right that HAP consistently exceeds the performance of HK-Means. Due to its superior parallel design, HAP, indicated by blue in the figures, is more receptive to benefits from parallelization on increasingly powerful Hadoop clusters than HK-Means, colored green in the figures.

Through parallelization, HAP is able to process the tensors at every level in a single step. In contrast, the Mahout implementation of K-Means is parallelized for each level, but creating the HK-Means "Top Down" structure requires sequential executions of K-Means.

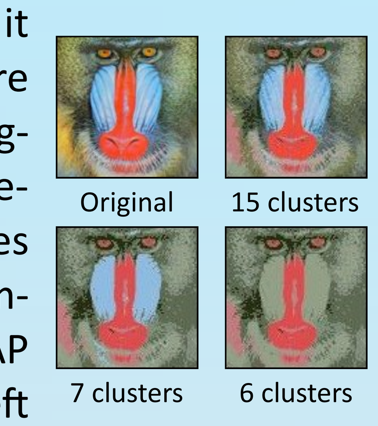
With significantly faster runtimes, HAP still posts purity levels competitive with HK-Means. This combination of speed and high performance is ideal for processing Big Data in a large-scale cloud computing environment.



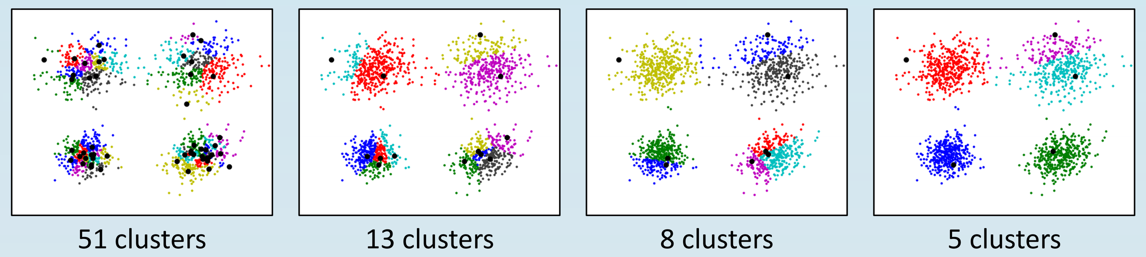


Results

HAP has shown great results when it comes to deriving the underlying structure of unknown data. When processing images, each pixel is its own data point represented as a RGB vector. In the images shown to the right, HAP has performed image segmentation. In the plots below, HAP clustered 2-D points by distance. From left to right, the sub-clusters group together into subsequent hierarchical levels.



128 x 128 image
16,384 pixels



1,600 2-D points from 4 Gaussian Distributions

Future Work

The final goal for this project is to use the cluster membership assignments learned from HAP in combination with semantic metadata mined from the input data to create a semantically rich, interactive user environment. In order to attain the metadata, the input data must be preprocessed. For example, sentiment analysis can be used for text, texture analysis can be used for images, etc. Because HAP is an unsupervised algorithm, the goal is to gather as much information as possible from preprocessing. This simulation of a user interface shows how our solution can present meaningful information about an initially unknown dataset in an easy-to-use, easy-to-understand, portable, and scalable web interface.

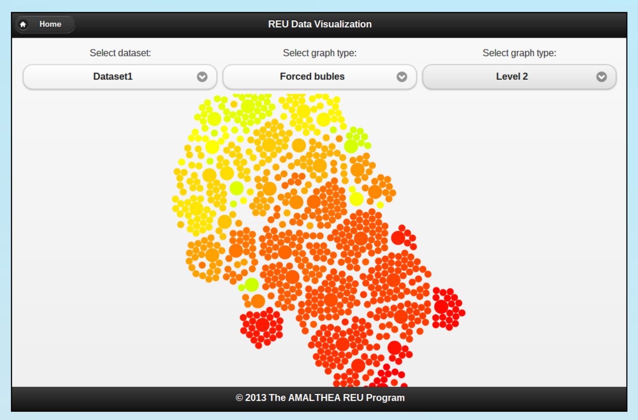


Image of User Interface

References

[1] I. E. Givani, C. Chung, and B. J. Frey. Hierarchical Affinity Propagation. *rXiv preprint arXiv* 1202. 3722 (2012)

[2] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation - Volume 6, USENIX Association,* 2004, 10-10

[3] J. A. Hartigan and M. A. Wong. Algorithm AS 136: A k-means clustering algorithm. Applied Statistics,28:100-108, 1978.

[4] N. Sahoo, J.Callan, R. Krishnan, G. Duncan, and R. Padman. Incremental hierarchical clustering of text documents. In Proceedings of the 15th ACM international conference on Information and knowledge management, CIKM '06, pages 357-366, New York, NY, USA, 2006. ACM.

This material is based upon work/research supported in part by the National Science Foundation under Grant No. 1263011.