INDICIa: a new distributed clustering protocol

Mar Callau-Zori
Universidad Politécnica de Madrid, Spain
mcallau@fi.upm.es

ABSTRACT

Many data streaming applications produces massive amounts of data that must be processed in a distributed fashion due to the resource limitation of a single machine. We propose a distributed data stream clustering protocol. Theoretical analysis shows preliminary results about the quality of discovered clustering. In addition, we present results about the ability to reduce the time complexity respect to the centralized approach.

1. INTRODUCTION

In the last years, the data streaming paradigm has emerged to deal with continuous processing in near-real time of applications like online targeted publicity, anomaly detection in cloud data centers, etc. Many of these applications require data mining tasks which cannot be satisfied by the query language provided by data streaming platforms. We focus in clustering, an important data mining task, where data is grouped according to different dimensions. A motivating scenario is targeted publicity in social networks, user information (e.g. tweets in Twitter) is grouped to drive the targeted publicity. Streaming scenarios are characterized by massive amounts of data demanding scalable distributed solutions that can leverage the cloud computing power. Our research work focuses precisely on this topic, how to perform clustering in a distributed system over data streams. In the social network application, this technique could be applied to clusters user messages among studding how trending topic emerges and evolves.

In this research abstract, we present INDICIa (*INnocent DIstributed ClusterIng*), a distributed clustering protocol that processes data as a single set in a distributed fashion. Our protocol aggregates the computational power of many nodes in order to share the load of the clustering processing. We show some preliminary results about the clustering quality and time complexity of the distributed protocol with respect a centralized approach. The clustering quality quantifies the ability of the algorithm to discover representative groups. The time complexity quantifies the cost of producing the clustering on a per tuple basis.

Prior work. Existing solutions for distributed clustering over streams focus mainly on producing intermediate clusterings computed close to the data sources and combining them [2, 1, 4]. These approaches introduce an important bias due to the hierarchical processing of the clustering.

2. PROBLEM STATEMENT

Given a data stream $S = \{s_n\}_{n\in\mathbb{N}} \subset X$, we focus on the k-means problem over each sliding window of size N: the goal is to find a set of k centers $\mathcal{C}_n \subset X$ that minimizes the quality function $cost(W_n, \mathcal{C}_n)$, measuring the sum of all the distances from each tuple in the window W_n to the nearest center in $\mathcal{C}_n$. Formally, a window is formed by the most recent N tuples starting from s_n, $W_n = \{s_n, \ldots, s_{n-N+1}\}$, and the function cost $cost(W_n, \mathcal{C}_n) = \sum_{s\in W_n} \min_{c\in\mathcal{C}_n} d(s, c)$. An algorithm gives an (a, b)-approximation to the k-means problem, if the center set has a number of centers in $[a, ak]$ with a cost at most b times the minimum cost. We refer to the *weight* of a center c as the number of tuples assigned to c, i.e., $\omega_n = card(\{s \in W_n : c = \arg\min_{\tilde{c}\in\mathcal{C}_n} d(s, \tilde{c})\})$. Being the weighted set $\mathcal{W}_n$ the weight of each center in $\mathcal{C}_n$.

The addressed problem is to distribute the computation of a k centers set $\mathcal{C}_n$ over each window W_n based on two algorithms: 1) algorithm $\mathcal{A}$ computes an (a, b)-approximation clustering over non-complete windows and the clusters weight in a $O(f(N))$ processing time where N is the maximum number of tuples received of a window; and 2) algorithm $\mathcal{A}_\mathcal{F}$ computes k centers over a weighted set $(\mathcal{C}, \mathcal{W})$ with approximation factor c in a $O(f(M))$ processing time where M is the length of the input weighted set.

3. PROPOSED APPROACH

We present here INDICIa, a protocol to distribute the computation over multiple nodes of the centralized algorithm $\mathcal{A}$ (Fig. 1). There are three kinds of nodes: 1) *load balancer*(LB) which distributes tuples according to a round-robin process; 2) *m instances of algorithm $\mathcal{A}$* $(\mathcal{A}_1, \ldots, \mathcal{A}_m)$ which compute a set of intermediate weighted clustering centers; and 3) *clustering aggregator* (AC) which computes

the final clustering $\mathcal{C}_n$ based on the algorithm $\mathcal{A}_{\mathcal{F}}$ over the weighted centers set from all the $\mathcal{A}$ instances.

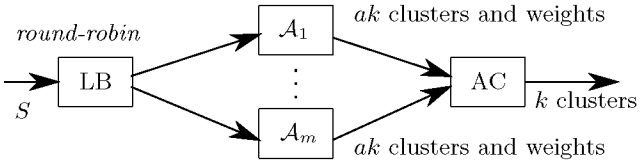

Figure 1: INDICIa

The addition of new stream sources is managed with a new LB at each stream source. It takes care of distributing the data across the $\mathcal{A}$ instances without introducing any bias on the sources of the data.

Elasticity. This challenge lies in how to provision and decommission instances. If clustering can be partitioned, a consistent protocol requires the ability of transferring state between instances. In this case, when the number of instances needs to be scaled up or down, some clusters are transferred from some instances to others. This approach has not impact on the final clustering. We provide an alternative approach if state cannot be partitioned. A new provisioned node will start performing a new intermediate clustering, initially with the empty set, with not impact in the final clustering. Moreover, during decommissioning, the intermediate clusters associated to the instances to be decommissioned will not be updated anymore. Hence, in the AC there are two kinds of intermediate clusterings depending on whether the $\mathcal{A}$ instance is still active. In this case, the AC component computes two different final clusterings from the intermediate clusterings according to this classification: one for the active instances and other one over the non-active instances.

4. PRELIMINARY RESULTS

In this section we present some preliminary results about INDICIa along two lines: 1) the clustering quality and 2) the improvement in the time complexity (reduction of processing time per tuple).

Clustering quality. In this section, we measure two source of error in INDICIa: 1) the window is split and several approximated clusterings are computed, and 2) the intermediate clusterings are not synchronized. Firstly, the impact of the partitioning in the quality has been studied in [3] (Thm. 4) with a different goal. We reformulate it to our case attaining an approximation cost factor $(c(b+1)+b)$. On the other hand, the non-synchronized instances arise from the fact that each incoming tuple is only sent to one $\mathcal{A}_i$. Focusing on the instance $\mathcal{A}_{old}$ that has received the oldest tuple, s_{old}, we have that the number of tuples outside the window W_n that have been considered in the clustering computation N_e is at most $\mathcal{E}_{old} = n - old$. As each $\mathcal{A}_i$ receives a tuple with probability $1/m$, then the number $\mathcal{E}_{old}$ is a random variable following a geometric distribution. Due to the geometric distribution measures the number of Bernoulli trials to get the first success (a success is "route to $\mathcal{A}_{old}$"). Applying the one-side Chebyshev's inequality over $\mathcal{E}_{old}$, we obtain the result.

PROPOSITION 4.1. *The number of tuples outside the window that are considered into the clustering, N_e, satisfies:*

$$Pr\{N_e \leq (1+\sqrt{1/\delta - 1}(m-1)\} \geq 1-\delta$$

Time complexity. In this section, we present a preliminary result about the reduction of the time processing per tuple. In INDICIa, $O(1)$ time is spent to route a tuple to an $\mathcal{A}$ instance. Due to the balancing load each $\mathcal{A}$ instance receives N/m tuples per window, computing the intermediate clustering in $O(f(N/m))$ time. Finally, the final clustering is computed from all the intermediate clusterings in a $O(f(akm))$ time. Hence, INDICIa processes a tuple in $O(f(N/m)+f(akm))$. At this point, we are interested into the constraints of the centralized algorithm $\mathcal{A}$ in order to improve the time complexity under INDICIa. The following result summarizes this idea.

PROPOSITION 4.2. *Given $\varphi(\cdot)$ the complexity set function respect to the number of $\mathcal{A}$ instances, i.e., $\varphi(1) = O(f(N))$ and $\varphi(m) = O(f(N/m)+f(akm))$, $\forall m \geq 2$. We have:* **C1)** *If $f(x) = x^p$, $p \geq 1$, then $\varphi(m) \subseteq \varphi(1)$, $N \geq 2akm$.* **C2)** *If $f(x) = \log(x)$, then $\varphi(m) \supseteq \varphi(1)$.*

PROOF. We remark that if $f(x) \leq g(x)$, then $O(f(x)) \subseteq O(g(x))$. Let us look at the function cases. **C1)** We need to prove $N^p \geq N^p/m^p + a^p k^p m^p$. Given the interesting points $m^p_{\pm} = \frac{N^p \pm \sqrt{N^{2p} - 4a^p k^p N^p}}{2a^p k^p}$, hence, it is enough prove that $m^p \in [m^p_-, m^p_+]$. As $N \geq 2akm$, then $m^p_+ \geq \frac{N^p}{2a^p k^p} \geq m^p$. Due to $m > 2$, hence it is enough to prove $m^p_- \leq 2^p \leftrightarrow \frac{2^{2p}}{2^p-1} a^p k^p \leq N^p \leftarrow \frac{2^{2p}}{2^p-1} a^p k^p \leq 2^{2p} a^p k^p \leftrightarrow \frac{2^{2p}}{2^p-1} \leq 2^{2p} \leftrightarrow 2^p - 1 \geq 1$ which is right with $p \geq 1$. **C2)** It is derived from $\log(N) \leq \log(N) - \log(m) + \log(ak) + \log(m)$. □

5. CONCLUSIONS AND FUTURE WORK

We have presented a protocol to distribute the clustering computation over data streams with some preliminary results about the clustering quality and the time complexity. As future work, we will focus on: 1) finding a metric about the impact in the clustering quality of the number of expired tuples, 2) a deeper study about the reduction of the time complexity, 3) designing of a smarter protocol that considers data locality to reduce space redundancy between instances, and 4) parallelizing the clustering aggregator AC. Finally, a real implementation using social network data (such as Twitter) will be conducted.

6. REFERENCES

[1] Beringer and Hüllermeier. Online clustering of parallel data streams. *DKE*, 2006.

[2] Dai, Huang, Yeh, and Chen. Clustering on demand for multiple data streams. In *ICDM '04*.

[3] Guha, Meyerson, Mishra, Motwani, and O'Callaghan. Clustering data streams: Theory and practice. *TKDE*, 2003.

[4] Hassani, Müller, and Seidl. EDISKCO: energy efficient distributed in-sensor-network k-center clustering with outliers. In *SensorKDD '09*.

7. ACKNOWLEDGEMENTS

This work was partially supported by the Spanish Research Agency: CloudStorm project TIN2010-19077 and the FPI PhD-fellowship BES-2008-009249; by the Madrid Regional Research Council: CLOUDS project and by the European Commission: MASSIF project FP7-257475.