EXPLORING PEER-TO-PEER DATA MINING

Andrea Marcozzi - Gianluca Mazzini

Lepida SpA, Bologna, Italy
andrea.marcozzi@lepida.it - g.mazzini@ieee.org

ABSTRACT

The emerging widespread use of Peer-to-Peer computing is making the P2P Data Mining a natural choice when data sets are distributed over such kind of systems. The huge amount of data stored within the nodes of P2P networks and the bigger and bigger number of applications dealing with them as p2p file-sharing, p2p chatting, p2p electronic commerce etc.., is moving the spotlight on this challenging field. In this paper we give an overview of two different approaches for implementing primitives for P2P Data Mining, trying then to show differences and similarities. The first one is based on the definition of Local algorithms; the second one relies on the Newscast model of computation.

KEYWORDS

distributetd data mining; local algorithms; gossiping

1. INTRODUCTION

Open peer-to-peer networks have become very popular for several kinds of applications and they have emerged as an important paradigm for distributed computing, due to their potential for the involvement of millions of peers in the process of sharing and collaboration. One of the most interesting features of P2P networks is their ability for direct resource sharing among dynamic, decentralized client peers. Due to the improvement of the technologies relating networks connectivities, digital storages and devices, these kind of server-less networks storing a huge amount of varying data are growing very fast; mining such relevant amount of data can be of great importance for several purposes,[1] that's why investigating methods for P2P data mining has become a very interesting research field.

Traditional approaches see data mining systems download all the relevant data stored in a P2P network into a centralized location and then perform the classical data mining operation. In the scenario we have described above, this kind of solution results to be not always appropriate, hence Distributed Data Mining has been introduced.

---

[1]Commercial, scientific and medical purposes.

Jan Zizka et al. (Eds) : CCSEIT, AIAP, DMDB, MoWiN, CoSIT, CRIS, SIGL, ICBB, CNSA-2016
pp. 165–174, 2016. © CS & IT-CSCP 2016 DOI : 10.5121/csit.2016.60613

Distributed Data Mining deals with data analysis in those environments in which data are distributed as for peer-to-peer networks and offers an alternative way to address this problem.

Researchers have developed several approaches for computing primitive operations (sum, average, max) on P2P networks. In this report we are going to introduce two different approaches: the first one is based on the concept of *local algorithms*[1], algorithms computing their results just with communications between immediate neighbors; the second one is based on the *Newscast model of computation* [4][3], a probabilistic epidemic protocol for information and membership dissemination.

In the next sections we will first give a brief overview on P2P data mining, its motivations and goals; then (section 3) we will introduce the concept of *local algorithms* [1] giving some examples; in section 4 we will introduce the *Newscast model* and give an idea of how it works along with some practical primitive implementations; finally in the last section we will draw some conclusions.

2. GOAL IN PEER-TO-PEER DATA MINING

One of the main goal of P2P data mining is to achieve as closer as possible the same results that can be obtained with a centralized approach without moving any data from the original location. That's why such algorithms must be highly scalable, tolerant to crashes and to peer “churn"[2] and mainly they must be able to calculate the results in-network instead of loading all the data in an unique system and then apply to itself the traditional data mining techniques.

As just said, there are several properties that are required by a peer-to-peer data mining algorithm:

scalability has been already mentioned and it is the foremost requirement; algorithms for peer-to-peer networks must be independent of the size of the network or at most they must be dependent of the log of the size;

anytimeness means that, since in some application the rate of the data change may be higher than the rate of computation, the algorithm must be able to provide a good and ad hoc solution at any time;

asynchronism is also crucial requirements for P2P algorithms: P2P networks are often huge, that means that any attempt to synchronize between the entire network is vane due to network latency or limited bandwidth;

decentralization means that the computations must be done in network, hence no centralized coordination must be used;

fault-tolerance is an other issue we have already mentioned; in a large P2P network can often happen that nodes crash or that they want to leave or join the network. That's why P2P algorithms

---

[2]By "Churn" we mean the situation in which some nodes leave the network and are suddenly replaced with brand new ones.

should be able to recover from these situations.

In the next sections we will present some primitive designed for Peer-to-Peer Data Mining purposes which show to comply with the above mentioned needs.

3. LOCAL ALGORITHMS

Approaches to P2P data mining have focused on developing data mining primitive operations over the network as well as more complicated data analysis algorithms.

Datta in [1] proposes some algorithms for calculating such primitives as sum and average, basing on the definition of *local algorithms*.

Given a constant k, for any network dimension, we can say that an algorithm is a local algorithm if there is a part of the input for which the algorithm terminates with communications expended per peer no greater than k and on the rest of the inputs, the communication expended per peer is of the order of the network size. Them can be divided into two categories: *exact local alg*orithm and *approximate local algorithms*: the former always terminate granting the same results that can be obtained with centralized methods; the latter instead, they can not give such level of accuracy.

Of course exact local algorithms can give better results, but it is not possible to develop them for every kind of situation.

In the next subsections will be given examples of both exact an approximate local algorithms.

3.1 Exact local algorithms (The Majority voting problem)

The Majority voting problem is a typical example of exact local algorithm which represents a situation in which each peer (P_i) of a P2P network has a number b_i which may be 0 or 1 and a threshold $1 > \tau > 0$ [3] (each node has the same τ). Peers want to collectively determine if the sum of all b_i is greater than $n\tau$, where n is the number of peers in the network.

Addressing this problem can serve as a primitive for several kind of data mining algorithms and can be used as a primitive for more complicated exact local algorithms.

P_i is a generic node in the network, N ei_i is one of its neighbors, C_i represents an estimate of the number of the nodes in the network and S_i is an estimate of the global sum. All the peers can only communicate with its neighbors and it is through this way that the just mentioned estimates are updated. We also define the threshold belief of P_i when such peer believes or not the majority threshold is met $(S_i - C_i\tau > 0)$.

Hence this threshold belief depends on the exchange of informations between neighbors. The crucial point of this approach concerns in deciding whether P_i needs to send a message to its neighbor P_j from which it has just received information on C and S. P_i will not send such a message if and only if it can be certain that such information cannot modify the threshold belief

---

[3] Actually in the original paper [1] was just indicated $\tau > 0$

of P_j . On the other end, if it cannot be certain of this, a message must be sent. This decision is taken on the basis of the estimate P_i makes on the values of C_j and S_j together with its values (C_i and S_i). When a node P_i decides to send a message, then it sends all of its informations about S and C, excluding those sent from P_j .

This approach is considered to be robust to data and network change: when a peer P_i changes its data, then P_i recomputes C_i and S_i and applies those conditions to all of its neighbors; if a peer P_j leaves the network, then P_i recomputes S_i and C_i without taking into account the informations from P_j .

Discussion

Exact local algorithms can be very useful in solving data mining problems in P2P networks but unfortunately they are very limited. The scope of such algorithms is restricted to functions that can have a local representation in the given network and they are limited to those problems which can be reduced to threshold predicates (as for the majority voting problem). An example of application is given in [1], where an exact local algorithm (based on the majority voting problem) is used for monitoring a K-means clustering of data distributed over a peer-to-peer network. In this application K-means clustering is performed in the traditional way (on a centralized system). After this, results are sent to the peers of the network and the local algorithm for monitoring the K-means clustering is executed: this algorithm just raises an alert if the centroids needs to be updated.

It is quite impossible to develop an exact local algorithm to compute the average of a set of data distribute over the network, that is why it is impossible to solve some data mining problems with exact local algorithms (as for example P2P K-means clustering). For addressing this, two different mechanism are proposed: the first one (section 3.2) describes an approximate local algorithm to perform the K-means clustering over a P2P network [2]; the second one (section 4) describes the Newscast model of computation for calculating means over data distributed on P2P overlay networks.

3.2 Approximate local algorithms (P2P K-means clustering)

The P2P K-means clustering algorithm [2] is an iterative algorithm requiring only local communications and synchronization at each iteration: nodes exchange messages and synchronize only with their neighbors. The goal is for each node to converge on a set of centroids that are as close as possible to the centroids that would have been produced if the data from all nodes was first centralized, then K-means was run.

The algorithm is initiated with a set of starting centroid selected at random. P_1, P_2,...., P_n denote the nodes in the network and X^i denotes the data set held by each node; the global data set is denoted as a X which equals to $\bigcup_{i=1}^{n} X^i$; the list of neighbors of a generic node i is denoted by $Nei^{(i)}$. Each node stores: a set $\{w_{j,k}^{(i)} : 1 \leq j \leq K\}$ indicating the centroids (*local centroids*) held by the node *i* at the beginning of cycle *l*; a termination threshold $\gamma > 0$; and a cluster count $|(w_{j,l}^{(i)})|$ which is the number of tuple in X^i for which $w_{j,l}^{(i)}$ is closer than any other $w_{h,l}^{(i)}, h \neq j$.

Each iteration of the algorithm is divided in two steps: the first one is similar to the centralized K-means in which peer P_i assigns each of its points to the nearest centroid; in the second one peer P_i sends a poll message to its immediate neighbors and waits for a respond. This request consists of a pair $\langle k, l \rangle$ (id, current iteration number) which is done in order to make the neighbors to respond with their local centroids and cluster count for iteration *l*. Once they have all responded, P_i updates its j^{th} centroid at the beginning of iteration $l + 1$. The update is a *weighted average*[4] of the local centroids and counts received from all immediate neighbors (for their iteration *l*). Then it moves to the next iteration of the K-means algorithm and repeats the whole process. If the maximum change in position of the new centroid after an iteration remains above the defined threshold γ, then P_i goes on iteration $l + 1$.

The key point is how do the peers respond to those requests. Suppose peer P_i receives a poll message $\langle k, \hat{l} \rangle$ from node k at its iteration $\hat{l}$: if $\hat{l} < l$, P_i sends its local centroid and cluster count to peer P_k; if $\hat{l} > l$, that means P_i does not have local centroids for iteration $\hat{l}$ and in such case, the poll message is put in the poll table of P_i; if $\hat{l} = l$, P_i checks if it contains local centroids and cluster counts for P_k, if so, they are sent to P_k, else the poll message is put in the poll table. Finally P_i will check its poll table at each iteration and will respond to any message it can.

At the end of each iteration *l*, if no important changes are detected on the cluster centroids (the maximum change in position is below the user defined threshold γ). each node could enter a termination state. In that case, such peer no longer updates its centroids and sends poll messages. However, it does responds to polling messages from its neighbors. Once all peers enter into the terminated state, the algorithm is terminated.

Experiments results and discussion

The P2P K-means clustering algorithm [1] presented in the previous section is a very important example which gives the idea of how is important to implement good primitives for data mining in distributed environments. The algorithm is in fact based on a primitive derived from the majority voting problem which is a primitive developed for peer-to-peer systems.

Datta in its works [1][2] performed several experiments with its P2P K-means clustering algorithm which seems to achieve good results. Experiments were conducted in both static and dynamic environments with a network of 1000 nodes: in both cases was calculated the accuracy with respect to the classic centralized K-means and the communication cost. In the static environment high accuracy is found (less than 3% of points per node misclassified on average) while no significant impact on this has had the method of assigning data points to node (uniformly or non-uniformly). However this has had a significant impact on communication costs: the number of bytes received per node increases slowly with network size for uniform assignment; the cost increases more sharply for non-uniform assignment.

Experiments were also conducted on a dynamic environment (with nodes leaving and joining the network) and even here good accuracy was found (less than 3:5% misclassified on average) which remains stable when the network evolves. Even increasing the network size did not seems to change the accuracy significantly: this proves that the algorithm is highly scalable.

[4]Such weighted average is calculated by primitive implemented with local algorithms.

4. DATA MINING THROUGH THE NEWCAST MODEL OF COMPUTATION

As already said, researchers working on distributed data mining have been focusing on investigating techniques for calculating primitives as *average*, *maximum* etc.. in distributed environment. Even Kowalczyk and Jelasity in [4] focused on this, but their approach is slightly different from the Datta's one [2].

First of all they adopted two important constraints: the first is that all the nodes (peers) store as few as one single data instances (in [1] each peer held several data instances); the second is that there is practically no limit on the number of nodes (even in [1] there was no potential limit on this, but experiments were always conducted on small networks of the order of thousand nodes). Furthermore, as in [1] nodes can leave and join the network as in a dynamic environment. For this, in this approach are needed resources that scale directly with the size of the network, which is a feature distinguishing it from local algorithms.

In the next subsections we are going to first introduce the Newscast model of computation, then we will propose some primitive for distributed Data Mining within this model and finally we will draw some conclusions.

4.1 The Newscast model (an overview)

Newscast [3] is a *gossip-based* topology manager protocol. Its aim is to continoulsy rewire the (logical) connections between hosts. The rewiring process is designed in such a way that the resulting overlay is very close to a random graph. The generated topology is thus very stable and provides robust connectivity.

As in any large P2P system, a node only knows about a small fixed set of other nodes (due to scalability issues), called *neighbors*. In Newscast, the neighborhood is represented by a partial, fixed c size view of node *descriptors* composed by a node address and a logical *time-stamp* (e.g., the descriptor creation time).

The protocol behavior performs the following actions: selects first a neighbour from the local view, exchanges the view with the neighbor, then both participants update their actual view according to the received view. The data actually sent over the network by any Newscast node is represented by the node's own descriptor plus its local view.

In Newscast, the neighbor selection process is performed in a random fashion by the SELECTPEER() method. The UPDATE() method is the Newscast core behavior. It merges ($\bigcup$ operation) a received view (sent by a node using SENDSTATE()) with the current peer view in a temporary view list. Finally, Newscast trims this list to obtain the new c size view. The node descriptors discarded are chosen from the most \old" ones, according to the descriptor time-stamp. This approach changes continuously the node descriptors hold in each node view; this implies a continuous rewiring of the graph defined by the set of all node views.

The protocol always tends to inject new informations in the system and allows an automatic elimination of old node descriptors using the aging approach. This feature is particularly desirable to remove crashed node descriptors and thus to repair the overlay with minor efforts.

Newscast is also cheap in terms of network communication. The traffic generated by the protocol involves the exchange of a few hundred bytes per cycle for each peer.

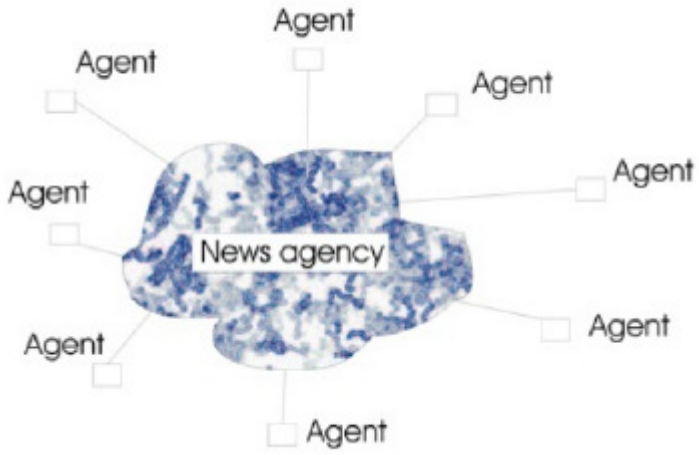

Figure 1: Conceptual model of the collective of agents and the news agency (from [4]).

Trying to talk about this model on an higher level, we can say that it is based on two main concepts: the collective of agents and the news agency (see Figure 1). The agents communicate through the news agency. Although the news agency plays the role of a server orchestrating the communication schedule, it is a virtual entity implemented in a fully distributed P2P solution. The communication schedule is organized into cycles and in each of them the news agency collect exactly one news item from all the agents. At the same time it prepares for each agent a randomly chosen set of news item from the previous cycle and delivers these set to the agents. In the next subsection we present an example of averaging primitives implemented within the model.

4.2 Basic Statistics

The ability of calculating the mean is central for implementing some basic data mining algorithms in Newscast. As we said the Newscast communication schedule is organized into cycles so what we want is an algorithm able to calculate in few cycles the average of the values held by the nodes of the Newscast Network. In this section we present three averaging algorithms developed on Newscast.

4.2.1 Basic Averaging (BA)

The *Basic Average algorithm* [4] is the simplest way to achieve this. During the first cycle each agent publishes its value so that the news agency get a copy of all the values that must be averaged. Next all agents whenever they receive news, they make the average of these values and then publish it. An important observation must be done: in every cycle the news agency receives a set of values that on average has the same mean of the original set, but the variance will be getting smaller and smaller with the number of cycles. This is the most important result of this algorithm. From the experiments performed, it can be shown that after k iterations of the "averaging operations", the variance drops to $(1 - 1/n)^k$ of its initial value.

As already said this is probably the simplest averaging algorithm that we can think on the Newscast model, but its simplicity pays the lack of adaptation. In fact if we think to a network where nodes leave and join, change their values and where nodes can temporary or permanently crash, the BA is not able to fit with these dynamical needs. To address this, the *Systematic Averaging algorithm* [4] is proposed (next section).

4.2.2 Systematic Averaging (SA)

The SA algorithm [4] achieves adaptation by constantly propagating agents current values and temporal averages through the news agency. Therefore, any change in the incoming data will quickly affect the final result.

A small positive integer d is fixed and it is used to control the depth of the propagation process. The news items are vectors of $d + 1$ elements: the first element of a news item X is x_0 and contains the agent value (called 0-order estimate of the mean); x_1 is the means of two 0-order estimates and it is called 1-order estimation mean; x_d, which is the last element, is the average of two estimates of order d -1 and is called an estimate of order d. In this way consecutive elements of X will be *balanced*. The result of this propagation is represented by x_d.

Even the SA algorithm has the ability to drop the variance: it decreases in an exponential way. Moreover the system can react to changes in the input data within d iterations.

4.2.3 Cumulative Averaging (CA)

The two algorithm we have just seen have the ability of reducing the variance very fast, but, due to the randomness characterizing the Newscast engine, the output value could be different from the true mean.

This problem is solved by the CA algorithm [4]. It runs two processes in parallel: in the first one agents updates their estimate of the mean of the incoming data, while in the second one the mean of these estimates is collectively calculated by a BA procedure.

4.3 Experiments configurations and results

The experiments we are going to describe [4] relates to the three averaging algorithms we have just mentioned. These are based on tests with different network sizes (from 10000 to 50000) and different data set. For each configuration were executed 100 independent runs. Were also used three different kind of data sets: Gaussian, where the value of each agent was taken independently from a Gaussian distribution; half-half, where half of the agents had value 0 and the other half 1; and peak where all but one agents hold the value 0.

With respect to the convergence rate the BA algorithm was fastest (20-30 iteration), the SA was slower (50 iterations) and the CA was the slowest (100 iterations). With respect to accuracy, the situation was inverted: BA was worst, SA better and CA best. The deviation from the true mean depends on the used distribution. Peak distribution has the intent of showing the "true power of the averaging algorithm", in fact we can note from the results with such distribution that the mean and variance with BA are respectively 0.935 and 0.656, while with SA they are 0.98 and 0.265.

4.4 Applications

As already seen with the primitives calculated with local algorithms, even here the averaging primitives implemented through the Newscast model can be used in several data mining tasks as for example for classification techniques. In [4] is reported an example in which the above mentioned averaging algorithms, with a little modification, are used for finding the Naive Bayes classifier for data that are arbitrarily distributed among the nodes in a P2P Network.

Kowalczyk *et al*. have implemented the Naive Bayes algorithm using BA as the base averaging method. As they had to maintain estimates of several means, they had to represent news items by vectors of the same length as the number of the means and to run BA on all coordinates at the same time. They have then tested the performances of that algorithm performing several experiments and then comparing the results with a classical Naive Bayes centralized algorithm. In most cases, although the model parameters were slightly different, no difference in the classification rate were found.

Most statistics that are used by other classification algorithms are defined in terms of ratios (or probabilities) that have the same form as described above. Consequently they can be implemented within the newscast framework.

5. CONCLUDING DISCUSSION

In this paper we have described in brief two interesting approaches to Data Mining in Peer-to-Peer Systems. The first one from Datta et al. [1] is based on the concept of *local algorithms* and the second one from Kowalczyk *et al*. [4] uses the *Newscast model of computation*.

Both the authors are intent to supply techniques for calculating primitives for P2P networks, primitives that form the basis for more complicated Data Mining algorithms. Both the approaches result very interesting even though they differs in some aspects.

Calculating primitives through the Newscast model of computation, resulted a winning strategy. The main task the authors wanted to deal with, was seeking a model for data spread over a number of agents; this was addressed through Newscast which is based on an epidemic protocol for disseminating information and group membership. The fact that the model lies on such robust and highly scalable protocol, states the goodness of the model itself, which inherits all these good features.

One important thing the two approaches have in common is that in both cases peers communicates only with their immediate neighbors but the second one (Newscast) does this in an epidemic-style manner so that results can be spread very quickly and all the agents can hear about the final solution in a short amount of time.

An important difference between the Newscast model and local algorithms which derives from what we have just said, is that in the Newscast model the terminations is reached once all agents have heard about the final results: although there is no signal that informs the agents that the result is found, using the theory of epidemic algorithms (which works as a broadcasting mechanism), all agents will hear about the final solution very quickly. Local algorithms instead, they terminates once a certain threshold is met: when an agent has reached an user defined

threshold (in the P2P K-means clustering it related the change in position of the centroids), it enters the terminated state; once all agents have reached such state, the whole process stops.

An other main dfference between the two approaches is that the News cast model requires resources that scale directly with the size of the network. So the resources required by the algorithm are dependent from the size of the system. In spite of this, the model results very scalable and robust. Local algorithms instead, computes their results using informations from a handful of nearby neighbors which leads to a good level of scalability too. It has also been proved that they are very good at adjusting to failure and changes in the input locally (see section 3.1). Even with the Newscast model it is possible to achieve these properties: we have seen the *Systematic Average algorithm* which is able to adjust to changes in the value of the agents on-the-fly and we have also seen that the tendency of the protocol to insert new informations in the system, allowing an automatic elimination of old node descriptors, is particularly desirable to remove crashed node descriptors and thus to repair the overlay with minor efforts.

In light of this, we can't say if one of the two approaches is better than the other one. We can certainly state (basing on the provided results) that both are able to fit with the peer-to-peer networks requirements: they are highly scalable, robust to node crashes and data set changes, decentralized and asynchronous. Once applied to real P2P Data Mining algorithms as K-means clustering and Naive Bayes classification, they have also shown a good level of accuracy and convergence to the results obtained with traditional centralized techniques.

In spite of this, data analysis in P2P systems still offers lot of challenges for the researchers. The experiments on the primitive and the distributed data mining algorithms we have described in this report, have shown good results but they come from lots of simulations done on P2P networks testbeds, hence we have “no mathematical proofs" of their absolute validity. As future work, it would be interesting and challenging at the same time to test this approaches on a platform like PlanetLab (a reliable testbed for overlay networks)[5], or even better on a real Peer-to-Peer Overlay Network.

REFERENCES

[1] Datta, S., Bhaduri, K., Giannella, C., Wol, R. (2005) Distributed Data Mining in Peer-to-Peer Networks. Invited submission to the IEEE Internet Computing special issue on Distributed Data Mining.

[2] Datta, S., Giannella, C., Kargupta, H. (2006) K-Means Clustering Over a Large, Dynamic Network. Accepted paper in SIAM2006 Data Mining Conference.

[3] Jelasity, M., van Steen, M. (2002) Large-Scale Newscast Computing on the Internet. Internal Report IR-503, Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, The Netherlands.

[4] Kowalczyk, W., Jelasity, M., Eiben, A. (2003) Towards Data Mining in Large and Fully Distributed Peer-to-Peer Overlay Networks. Technical Report IR-AI-003, Vrije Univeriteit Amsterdam, Department of Computer Science, Amsterdam, The Netherland.

---

[5]http://www.planet-lab.org, Last visited March 2016