Available online at http://www.journalcra.com

International Journal of Current Research
Vol. 7, Issue, 04, pp.14455-14459, April, 2015

INTERNATIONAL JOURNAL OF CURRENT RESEARCH

ISSN: 0975-833X

RESEARCH ARTICLE

EFFICIENT CLUSTERING OF BIG DATA USING GRAPH METHOD

***Kameshwaran, K. and Malarvizhi, K.,**

Department of Computer Science and Engineering, Coimbatore Institute of Technology, Coimbatore, Tamil Nadu, India

ARTICLE INFO

Article History:
Received 25th January, 2015
Received in revised form
16th February, 2015
Accepted 20th March, 2015
Published online 28th April, 2015

Key words:

Big data,
Clustering, PIC,
Parallel Computing,
Data Mining.

ABSTRACT

The data mining process is to extract the information from a large data set and transform the extracted data into an understandable structure for further use. Clustering is a main task of exploratory data analysis and data mining applications. Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups (clusters). Power Iteration Clustering (PIC) algorithm is recently identified algorithm which helps to create a good quality of cluster. PIC is simple and scalable graph clustering technique. In PIC the embedding is an approximation to a eigen value-weighted linear combination of all the eigenvectors of an normalized similarity matrix. This embedding turns out to be very effective for clustering. This work shows that PIC consistently outperformed when the process is parallelized and it achieves fault tolerance by branch and bound algorithm. Parallelization minimizes computation cost, this algorithm works on all lower end commodity computers.

INTRODUCTION

Spectral clustering is an effective and elegant clustering method based on the pair wise similarity between objects. Here we present a fast and simple spectral-clustering like technique called power iteration clustering (William *et al.,* 2010). As in spectral clustering, points are embedded in a low-dimensional subspace derived from the similarity matrix for the data points; however, while in spectral clustering, the subspace is derived from the bottom eigenvectors of the Laplacian of an affinity matrix, in PIC, the subspace is an approximation to a linear combination of these eigenvectors. We show that Parallel PIC (Yana *et al.,* 2013) obtains comparable or better clusters than existing power iteration methods; however, the most important advantages of the method are its simplicity and scalability. In PIC the subspace used is a one-dimensional subspace formed by using the power iteration with early stopping on a normalized affinity matrix (the power iteration is normally run to convergence in order to find the dominant eigenvector of a matrix); this technique is simple, scalable, easily parallelized, fault tolerance and well-suited to very large datasets.

POWER ITERATION CLUSTERING

For description clarity, Table 1 summarizes the notations and abbreviations (William *et al.,* 2010) used in the paper. Spectral clustering has become an active research area in machine learning. It often out-performs traditional clustering algorithms (e.g. k means) when the clusters are not Gaussian distributed. Consider a set of data points: {x1, x2,…, xn}, where x is a d-dimensional vector, and some notion of similarity, for example,

$$S(x_i, x_j) = exp\left(-\frac{\|x_i - x_j\|_2^2}{2\sigma^2}\right) \quad \text{..... (1)}$$

Table 1. Notation and Abbreviation used

n	Number of samples/cases in data file
k	Number of iterations before stopping
p	Number of processors
$s(x_i, x_j)$	Similarity function
r_i^1, r_i^2	Data starting & ending indices for ith processor
A	Similarity matrix
A_i	Similarity sub-matrix calculated on ith processor
D	Diagonal matrix with $D(i,i) = \sum_j A_{ij}$
W	Normalized similarity matrix, $W = D^{-1}A$
R	Row sum vector
v^0	Initial vector
v^{t-1}, v^t	Vector at iteration $t-1$ and t, respectively
γ	Normalizing constant

Where σ is a scaling parameter controls the kernel width. We can build an affinity matrix A with $a_{ij} = s(x_i, x_j)$ if $i \neq j$ and $a_{ij} = 0$ if $i = j$. The degree matrix associated with A is denoted by D. Diagonal matrix D has a diagonal entries which equal to the row sums of A, i.e., $D_{ii} = \sum_j A_{ij}$ Aij. A normalized random-walk Laplacian matrix L is defined as $L = \Delta - D^{-1}A$, where Δ is an identity matrix. Intrinsic clustering structure is often

****Corresponding author: Kameshwaran,***
Department of Computer Science and Engineering,Coimbatore Institute of Technology, Coimbatore, Tamil Nadu, India.

revealed by representing data in the basis composed of the smallest eigenvectors of L not the very smallest one, is a constant vector that doesn't have discriminative power. If we define another matrix $W = D^{-1}A$, its largest eigenvector is the smallest eigenvector of L. Power Iteration (PI) is a well-known method for computing the largest eigenvector of a matrix, which randomly initializes a N-dimensional vector v^0 and iteratively updates the vector by multiplying it with

$$Wv^t = \gamma Wv^{t-1}, t = 1, 2, \ldots, \quad \ldots\ldots\ldots (2)$$

Where γ is a normalizing constant to keep v^t numerically stable. Lin and Cohen (William *et al.,* 2010) discover an interesting property of the largest eigenvector of W: before the elements of v^t converge to the constant value, they first converge to local centers that correspond to the clusters in the data. Therefore, the largest eigenvector, v^t, which is discarded in spectral clustering algorithms, becomes a useful tool for clustering. They then developed a systematic way to determine when to stop the iteration, i.e. when the elements have achieved local convergence but have not yet achieved global convergence. They called their algorithm power iteration clustering (PIC). Algorithm 1 (William *et al.,* 2010) summarizes the key steps of the PIC algorithm. PIC is computationally efficient since it only involves iterative matrix–vector multiplications and clustering of the one dimensional embedding of the original data, which is relatively easy to do. However, similar to other spectral clustering algorithms, a bottleneck for PIC when applied to large datasets lies in the calculation and storage of big matrices.

For example, the computational complexity of computing pair-wise distance/similarity for all data points is O (n2). Also, it is infeasible to have a n×n matrix (W) stored in memory when n is large. These practical constraints motivate us to consider parallel strategies that distribute the computation and data across multiple machines.

Algorithm 1. Power Iteration Clustering Algorithm

Input: A dataset $x=\{x_1, x_2, \ldots, x_n\}$ and similarity function $s(x_i, x_j)$

//similarity matrix calculation and normalization

1 construct similarity matrix, $A \in R^{n \times n}$

2 Normalize similarity matrix by dividing each element by its row sum, $W=D^{-1}A$

//iterative matrix-vector multiplication

3 Generate initial vector, $V^0=R/\|R\|_1$, where R is the row sum of W

4 repeat the following

5 Calculate new vector and new velocity, $v^t = \gamma W v^{t-1}$, $\delta^t = |v^t - v^{t-1}|$

6 Until stopping criterion is met, $|\delta^t - \delta^{t-1}| \approx 0$

//clustering

7 Cluster on the final vector, v^t

8 Output the clusters

Parallel Power Iteration Clustering

The PIC algorithm has slightly better capability in handling large data. However, it still requires both the data and the similarity matrix fit into computer memory, which is not a feasible for massive datasets. These constraints motivate us to consider parallel strategies (Yana *et al.,* 2013), distribute the data and computation across multiple machines. The implementation of parallel PIC algorithm was done using message passing interface as the programming model due to its efficiency and performance for data communications in distributed cluster environments. There are many different parallel programming frameworks available (He *et al.,* 2009). The message passing interface (MPI) is a message passing library interface for performing communications in parallel programming environments (Yana *et al.,* 2013). The algorithm for parallel PIC is described in Algorithm 2 and Algorithm 3. The problem in parallel PIC is fault tolerance; if any one node (Slave) is failed then the job given to that particular node cannot be done so the data sent to that particular node doesn't produce any similarity matrix and cluster.

Algorithm 2. Steps for Master Processor

1 Determine # of splits of the data to cluster and get the starting and end indices of the cases

2 Broadcast indices to each of the slave processors

3 Read in one case at time and broadcast to all processors

4 Receive $R_i, i = 1,2,\ldots,p$ from all slave processors and concatenate them to obtain *the overall row sum, R*

5 Obtain the initial vector, $v^0 = R/\|R\|_1$

6 Broadcast the vector, v^0, to all processors

7 Receive sub vectors $v_i^t, i = 1,2,\ldots,p$ from all processors and concatenate them to obtain a new vector, v^t, and normalize it by its element sum

8 Check the stop criteria, $|\delta^t - \delta^{t-1}| \approx 0$

9 if not, go to step 6

10 if yes, stop

Algorithm 3. Steps for Slave Processor

1 Get starting and end indices of cases from master processor

2 Read in the chunk of case data and also get a case broadcasted from master

3 Calculate similarity sub-matrix, A_i ,a *n/p*-by-*n* matrix

4 Calculate the row sum, R_i, of the sub-matrix and send it to master

5 Normalize sub-matrix by the row sum, $W_i = D_i^{-1}A_i$

6 Receive the initial vector from the master, v^{t-1}

7 Obtain sub-vector by performing matrix-vector multiplication, $v_i^t = \gamma W_i v^{t-1}$,

8 Send the sub-vector, v_i^t , to master

Parallel Power Iteration Clustering With Fault Tolerance

Fault Tolerance Hierarchy – Branch and Bound (FTH-B & B) is an FT parallel B&B algorithm based on the Hierarchical Master paradigm in order to deal with the FT issue while ensuring scalability in large scale environments. The proposed approach is composed of several FT Master/Worker-based sub-B & B algorithms, where inside each sub-B&B one master manages several workers and performs failure recovery. The sub-B & Bs are launched in parallel and act independently on different sub-problems. They are organized hierarchically into several levels. The root node and the inner nodes of the hierarchy designate masters and the leaves designate workers. The masters perform a parallel recursive branching in order to decrease the size of sub-problems until reaching sufficiently fine-grained sub problems which are explored sequentially by the workers. In addition, they locally store the branched and assigned sub-problems for further rescheduling. Each sub-B&B is composed of one master and a group of workers. Where one master manages a dynamic group of communicating workers. Therefore, the algorithm is composed of communicating groups of FT Master based sub- B & B processes.

The algorithm for Parallel PIC with fault tolerance is described in Algorithm.4.

Algorithm 4. Steps for Fault Tolerance using Branch and Bound

1 **Initialization:** The master node should be declared to be alive.

2 **Visit:** The failure node criteria decide which of the live nodes is to process next.

3 **Replacement:** The chosen node is removed from the set of live nodes, and sample data set has to be sent to new live slave node.

4 **Iteration:** The visitation and replacement steps are repeated until stopping criteria met.

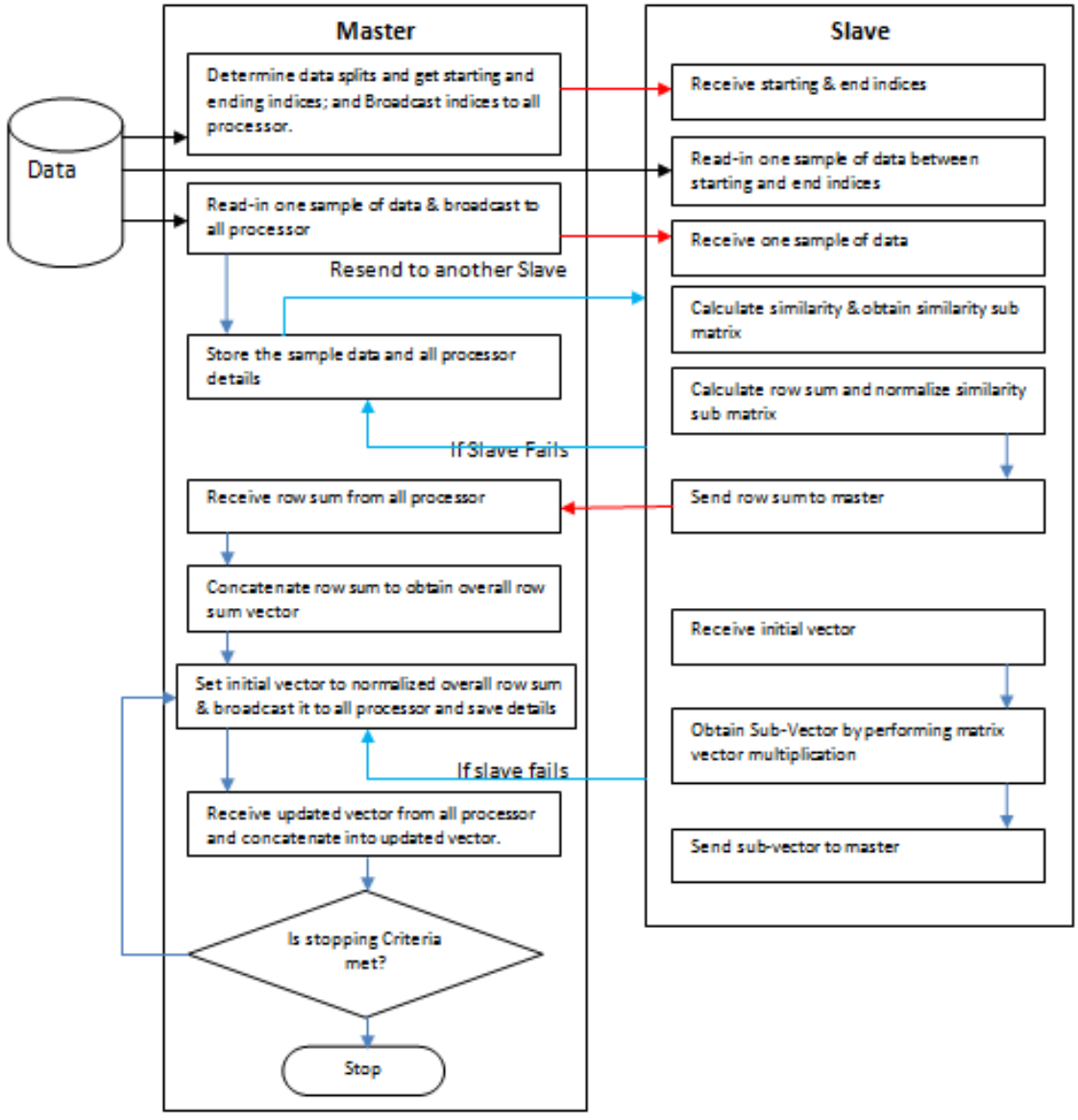

Figure 1. Overview of Parallel PIC with Fault Tolerance

The Hadoop https://hadoop.apache.org/ setup for single node follows by multi node configuration. The prerequisites for the hadoop installation are JDK. Commodity computers or laptops (nodes) with password-less logins and Security Shell (SSH) enabled. Install Hadoop release 1.0.3 and Operating system may be fedora or ubuntu.

Single Node Setup in Hadoop https://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/

- Extract hadoop-1.0.3 in a required destination folder.
- Change the properties of core-site.xml which is present in hadoop1.0.3/conf/core-site.xml
- Change the properties of hdfs-site.xml which is present in hadoop1.0.3/conf/hdfs-site.xml
- Change the properties of mapred-site.xml which is present in hadoop1.0.3/conf/mapred-site.xml.
- Enable JAVA-HOME path in hadoop1.0.3/conf/hadoop-env.sh
- Setup password-less security SHell (SSH).
- Format the Hadoop Distributed File System (HDFS) via the name node.
- Finally start the java process to run the program.

MULTI NODE SETUP IN HADOOP https://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/

- Choose one of the node to be the Master Node (for HDFS).
- To network master and slaves update the IP address of master node and slave nodes in /etc/hosts.conf
- Enter the master node`s IP in hadoop-1.0.3/conf/masters (only on master machine)
- Enter both master and slave node IPs in conf/slaves (only on master node).
- Enable SSH for master to master and master to slave.
- Change the property value of fs.default.name in hadoop-1.0.3/conf/core-site.xml to master node's IP (on both master and slave machine)
- Change the value of mapred.job.tracker in hadoop-1.0.3/conf/mapred-site.xml to master node's IP (on both master and slave machine)
- Change the property value of dfs.replication to 2 (since we have 2 nodes if there several slave node then value depend on slave node i.e. slave nodes + master node , it's better to replicate the HDFS blocks) in hadoop-1.0.3/conf/hdfs-site.xml (on both the nodes)
- Format the Hadoop Distributed File System (HDFS) via the name node in master node.
- Start the HDFS layer id the master node and check for java process in both master and slave node.

Experimental Results

To demonstrate the data scalability of parallel PIC algorithm, we execute the algorithm on a number of synthetic datasets. The size (n) of the datasets varies from 200 to 5000 rows, with the number of attributes being 24. Initially we test the datasets using PIC which consumes more time and space to create a single node cluster in hadoop. Similarly the same datasets evaluated by replacing Parallel PIC with the help multi nodes that reduce the execution time, space and fault tolerance is achieved using the methodology called the branch and bound problem.

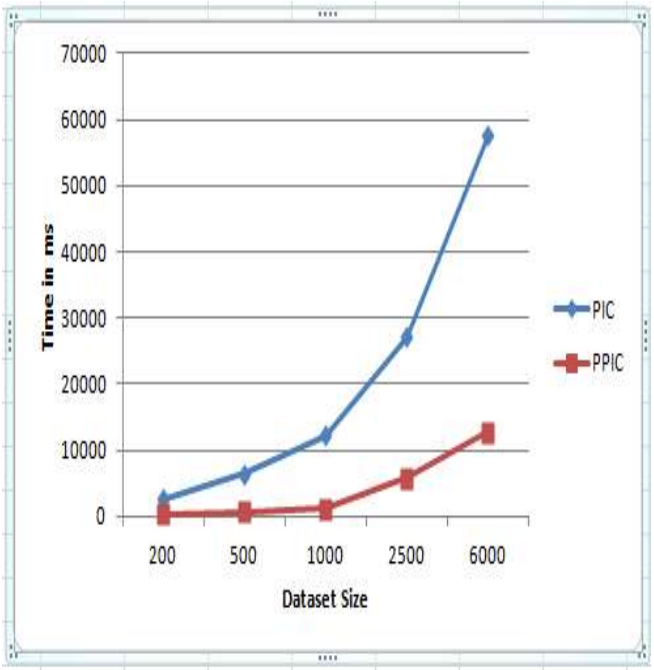

Figure.1. Sample Output of PIC and PPIC with fault tolerance

Conclusion and Future Work

Power Iteration Clustering (PIC) algorithm is recently identified algorithm which helps to create a good quality of cluster. PIC is simple and scalable graph clustering technique. In PIC the embedding is an approximation to a Eigen value-weighted linear combination of all the eigenvectors of an normalized similarity matrix. This embedding turns out to be very effective for clustering. This work shows that PIC consistently outperformed when the process is parallelized and node failure achieved by branch and bound algorithm. Parallelization minimizes computation cost, this algorithm works on all lower end commodity computers. We would also like to explore efficient data transmission methods, following the work by Hsu et al., for data-intensive scientific applications over cloud computers. In addition we would like to investigate other parallel programming frameworks

Acknowledgement

I would like to thanks the Department of Computer Science & Engineering of Coimbatore Institute of Technology, Coimbatore, Tamil Nadu, India.

REFERENCES

Yana, W *et al*. 2013. "P-PIC: Parallel-Power Iteration Clustering for big data", Models and Algorithms for High-Performance Distributed Data Mining. Volume 73, Issue 3, March

Xu, R. and Wunsch, D. 2005. "Survey of clustering algorithms", IEEE Transactions on Neural Networks 16 (3).

Kameshwaran, K. and K. Malarvizhi, 2014. "Survey on Clustering Techniques in Data Mining", International Journal of Computer Science and Information Technologies, Vol. 5 (2) , 2272-2276.

Chen, W.Y., Song, Y, Bai, H. and Lin. C. 2011. "Parallel spectral clustering in distributed systems", IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (3) 568–586

Single Node Installation, https://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/

Multi Node Installation, https://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/

Apache Hadoop Tutorial, https://hadoop.apache.org/

He, Q., Ma, H., Zhao, W. 2009. "Parallel K-means clustering based on MapReduce", Journal of Cloud Computing 5931 674–679.

William W. and Frank Lin, 2010. "Power Iteration Clustering", International Conference on Machine Learning" , Haifa, Israel,

****