Hierarchical Clustering on Massive Datasets

K.Maheswari[1] and M.Ramakrishnan[2]

[1]Research Scholar, Department of Computer Science, Bharathiyar University, Coimbatore, Tamil Nadu,India
[1] magzhere@gmail.com
[2]Chairperson, School of Information Technology, Madurai Kamaraj University, Madurai, Tamil Nadu, India,
[2]ramkrishod@gmail.com

ABSTRACT

This paper presents hierarchical clustering of massive data sets using Support vector machines. SVMs are capable of providing classification and regression analysis since they are contain solid mathematical foundations which convey several salient properties that other methods fail to provide. But SVM are not much utilized for huge volume of data and many real-world data mining applications involve millions or billions of data records where even multiple scans of the entire data are too expensive to perform. Hence we have applied SVM in clustering huge volume of data. Experimental results show that SVM based hierarchal clustering is highly scalable with high accuracy.

Keywords— Hierarchical Clustering, SVM, Large Volume of Data, Map Reduce

1. INTRODUCTION

In hierarchical clustering a series of partitions takes place, which may run from a single cluster containing all objects to n clusters that each contains a single object. Hierarchical Clustering can be either agglomerative methods, which proceed by a series of fusions of the n objects into groups or divisive methods, which separate n objects successively into finer groupings. Commonly used is agglomerative techniques and this is the method implemented in XLMiner. Hierarchical clustering may be represented by a two-dimensional diagram known as a dendrogram, which illustrates the fusions or divisions made at each successive stage of analysis[1].

The hierarchical micro-clustering algorithm denotes a statistically summarized representation of a group of data which are so close together that they are likely to belong to the same cluster. Our hierarchical micro-clustering algorithm has the following characteristics. It builds a micro-cluster tree, called CF (Clustering Feature) tree, in one scan of the data set given a limited amount of resources by incrementally and dynamically clustering incoming multidimensional data points. Since the single scan of the data does not allow backtracking, localized inaccuracies may exist depending on the order of data input[2]. CF tree captures the major distribution patterns of the data and provides enough information. Another feature of hierarchical micro-clustering is that it handles noise or outliers effectively as a by-product of the clustering. Hierarchical clustering plays a vital role in data mining and hence we present a paper on hierarchical clustering of massive data set using SVM machines. The paper is structured as follows: Paper starts by giving introduction about clustering. Literature survey is presented in section 2 and section 3 briefs related workings. Our proposed method is detailed in section 4 and experimental results are presented in section 5. The paper ends by briefing the findings as conclusion in section 6.

2. BACKGROUND WORK

Plenty of algorithms and implementation techniques have been proposed for improving SVMs efficiency since the running time of the standard QP algorithms grows too fast. Generally to speed up, SVM training divides the original QP problem into small pieces, thereby reducing the size of each QP problem. Chunking, decomposition and sequential minimal optimization are most well-known techniques[3].

SVM approximations have been attempted to improve the computational efficiency of SVMs by altering the QP formulation to the extent that it keeps a similar semantic of the original SVM while it is faster to be solved by a QP solver.

But new formulations are still not proven to be efficient and reliable enough to work with very large data sets[4].

On-line SVMs or incremental and decremental SVMs have been developed to handle dynamically incoming data efficiently. SVM model is incrementally constructed and maintained, the newer data have a higher impact on the SVM model than older data[5].

Selective sampling or active learning is to intelligently sample a small number of training data from the entire data set that maximizes the degree of learning, i.e., learning maximally with a minimum number of data points. The core of the active learning technique is to select the data intelligently such that the degree of learning is maximized by the data[6].

A common active learning paradigm iterates a training and testing process works in series of steps namely, construct a model by training an initially given data, test the entire data set using the model, by analyzing the testing output, select the data that will maximize the degree of learning for the next round, accumulate the data to the training data set, and train them to construct another model[7].

The selective sampling of SVMs selects the data close to the boundary in the feature space at each round because the data near the boundary have higher chances to be SVs in the next round. They iterate until there exists no data nearer to the boundary than the SVs[8]. But it generates too much I/O cost for very large data sets. Few random sampling techniques were developed to reduce the training time of SVMs for large data sets. But it is also based on selective sampling

3. RELATED WORK

3.1. Clustering Feature and CF Tree

The clustering feature (CF) tree is the core of the hierarchical micro-clustering algorithm which makes the clustering incremental which avoids expensive computations. CF summarizes the information that a CF tree maintains for a cluster. From the CF definition and additivity theorem, the CF vectors of clusters can be stored and calculated incrementally and accurately as clusters are merged. The centroid and the radius of each cluster can be also computed from the CF of the cluster[9]. CF of the sub cluster is represented by the child. A leaf entry, the entry in a leaf node, only has a without a child pointer. So, a leaf or a non leaf node represents a cluster made up of all the sub clusters represented by its entries. The threshold is a constraint for the leaf entries to satisfy such that the radius of an entry in a leaf node[10]

The CF is a summary of a cluster–a set of data points. Managing only this CF summary is efficient, saves spaces significantly, and is sufficient for calculating all the information for building the hierarchical micro-clusters which will facilitate computing an SVM boundary for a very large data set. A CF tree is a height-balanced tree with two parameters: branching factor and threshold[11].

The tree size is a function of t. If t is large, the tree is small. The branching factor can be determined by a page size such that a leaf or non-leaf node fit in a page. This CF tree is a compact representation of the data set because each entry in a leaf node is not a single data point but a sub cluster[12].

4. PROPOSED ALGORITHM

A CF tree is built up dynamically as new data objects are inserted. The insertion of data into the correct sub cluster, merges leaf nodes, and manages non leaf nodes are similar to those in a B+-tree, which follows the below algorithm:

- Identifying the appropriate leaf: Starting from the root, it descends the CF tree by choosing the child node whose centroid is the closest.
- Modifying the leaf without violating the threshold conditions
- Modifying the path to the leaf which causes an insertion of a new non-leaf into the parent node.

A highly skewed input could cause two sub clusters since we have limited number of entries in a node that should have been in one cluster.

4.1. Determination of threshold

The choice of the threshold is crucial for building the tree in the right size which fits in the available memory because if is too small, we run out of memory before all the data are scanned. The original BIRCH algorithm initially sets very low, and iteratively increases until the tree fits in the memory[13]. After the construction of a CF tree, the leaf entries that contains far fewer data points than average are considered to be outliers. A low setting of outlier threshold can increase the classification performance of CB-SVM especially when the number of data is relatively large compared to the number of dimensions and the type of boundary functions are simple because the non-trivial amount of noise in the training data which may not be separable by the simple boundary function prevents the SVM boundary from converging in the quadratic programming. For this reason, we enabled the outlier handling

with a low threshold in our experiments in Section 5 because the type of data we are targeting is of large number of data points with relatively low dimensions, and the type of the boundary functions is linear with VC dimension m+1, where m is the number of dimensions[14].

A CF tree that fits in a memory can have the $\frac{M}{P}$ nodes at maximum where M is the size of memory and P is the size of a node. The height of a tree is $logb\left(\frac{M}{P}\right)$ which is independent of the size of the dataset. The algorithm for

Algorithm

Input: Data set *P*, negative data set *N*
Output: - a boundary function *h*
Notation:
- HC(S): hierarchical clustering algorithm from tree T and data set S
- getRootEntries(T): return the root entries of a tree T
- getChildren(S):return the children entries of an entry set
- getLowMargin(h,S):return the low margin entries from a set which are close to the boundary h

1. T_p = *HC(P);* T_n=*HC(N);*
2. *S=getRootEntries(*T_p*)*$\cup$*getRootEntries(*T_n*);*
3. *Do loop*
   h=train(S);
   S'=getlowmargin(h,S);
   S'=getchildren(S');
   If S'=ϕ*, exit*
   S=S$\cup$*S'*
4. *Return h;*

5. EXPERIMENTAL EVALUATION

To test the effectiveness of the proposed CB-SVM in realistic environments while providing visualized results, we perform binary classifications on two-dimensional data sets. Once the characteristics of each cluster are determined, the data points for the cluster are generated according to a 2-d independent normal distribution.

The class label of each data is inherited from the label of its parent cluster. Note that due to the properties of the normal distribution, the maximum distance between a point in the cluster and the center is unbounded. We enabled the shrinking heuristics for fast training. The results are tabulated as follows:

Table 1. Experimental Results

S-Rate in percentage	No. of data in data sets	No. of errors	T-Time	S-Time
0.0001	23	6348	0.001457	847.42
0.001	214	2378	0.009147	841.78
0.01	2147	1478	0.125	834.15
0.1	24871	1047	9.547	837.14
1	247899	1128	784.96	841.08
5	1741859	1020	1897.23	847.57
ASVM	298	843	54968.21	54968.21
Our method	3657	752	1.784	2387.687

From the above table, it is quite clear that our method performs well when compared to ASVM method.

6. CONCLUSION

In this paper we proposed hierarchical clustering of huge volume of data using SVM. The method called CB-SVM (Clustering-Based SVM) integrates a scalable clustering method with an SVM method and effectively runs SVMs for very large data sets. The existing SVMs are not feasible to run such data sets due to their high complexity on the data size or frequent accesses on the large data sets causing expensive I/O operations. CB-SVM applies a hierarchical micro-clustering algorithm that scans the entire data set only once to provide an SVM with high quality micro-clusters that carry the statistical summaries of the data such that the summaries maximize the benefit of learning the SVM. CB-SVM tries to generate the best SVM boundary for very large data sets given limited amount of resource based on the philosophy of hierarchical clustering where progressive deepening can be conducted when needed to find high quality boundaries for SVM. Our experiments on synthetic and real data sets show that CB-SVM is very scalable for very large data sets while generating high classification accuracy. But CB-SVM is currently limited to the usage of linear kernels since the hierarchical micro-clusters would not be isomorphic to a new high-dimensional feature space

REFERENCES

[1] L. Kaufman and P.J. Rousseeuw, "Finding Groups in Data: An Introduction to Cluster Analysis", New York: John Wiley & Sons, 1990.

[2] A. Hinneburg and D. A. Keim, "An Efficient Approach to Clustering in Large Multimedia Databases with Noise", KDD'98, New York, Aug. 1998.

[3] M. Ester, H.-P. Kriegel, and J.Sander, "Spatial Data Mining: A Database Approach", SSD'97, Berlin, Germany, July 1997.

[4] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases", SIGMOD'96, Montreal, Canada, June 1996.

[5] R. Ng and J. Han, "Efficient and Effective Clustering Method for Spatial Data Mining", VLDB'94, Santiago, Chile, Sept. 1994.

[6] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification", SSD'95, Portland, ME, Aug. 1995.

[7] P. Bradley, U. Fayyard, and C. Reina, "Scaling Clustering Algorithms to Large Databases", KDD'98, New York, Aug. 1998.

[8] M. Dash, H. Liu, X. Xu, "1+1>2: Merging Distance and Density Based Clustering", DASFAA, 2001.

[9] J. MacQueen, "Some Methods for Classification and Aanalysis of Multivariate Observations", Proc. 5th Berkeley Symp. Math. Statist. Prob., 1:281-297, 1967.

[10] Qin Ding, Maleq Khan, Amalendu Roy, and William Perrizo, "P-tree Algebra", ACM Symposium on Applied Computing, Madrid, Spain, 2002.

[11] Maleq Khan, Qin Ding, William Perrizo, "K-Nearest Neighbor Classification of Spatial Data Streams using P- trees", PAKDD-2002, Taipei, Taiwan, May 2002.

[12] Qin Ding, Qiang Ding, William Perrizo, "Association Rule Mining on Remotely Sensed Images using P-trees", PAKDD-2002, Taipei, Taiwan, 2002.

[13] Qin Ding, William Perrizo, Qiang Ding, “On Mining Satellite and other RSI Data”, DMKD-2001, Santa Barbara, CA, 2001.

[14] A. Roy, “Implementation of Peano Count Tree and Fast P-tree Algebra”, M. S. thesis, North Dakota State University, 2001.

