International Journal of Engineering Research and General Science Volume 4, Issue 3, May-June, 2016
ISSN 2091-2730

# Data Mining of Genomic Data using Classification Algorithms on MapReduce Framework: A Survey

Rajarshi Banerjee[1], Ravi Kumar Jha[1], Aditya Neel[1], Rituparna Samaddar (Sinha) [1] and Anindya Jyoti Pal[1]

[1]Department of Information Technology
Heritage Institute of Technology
Kolkata-700107, INDIA
aj.pal@heritageit.edu, rajarshibanerjee1112907@gmail.com, 91-9674038667

**Abstract**-The advancement of Next Generation Sequencing (NGS) technology has led to a huge deluge of genomic data. In order to classify this data, the algorithms that are in use are struggling because of the enormity of the datasets. Hence parallelization of these classification algorithms using a MapReduce implementation is an efficient solution that tackles these issues such as computational cost, spatial complexity and running time. In this survey paper we have presented the various DNA sequence alignment algorithms that are most commonly used. Then we have discussed the various classification algorithms that are used in data mining genomic databases and presented a relative study of their merits and demerits. And finally we have focused on Big Data and MapReduce which helps in the increasing the efficiency of the classification algorithms mentioned in this survey paper.

**Keywords**- NGS, kNN, SVM, Naïve Bayes, C4.5, HDFS, Big Data, MapReduce, Data Mining, Classification

## 1. INTRODUCTION

Using data obtained from sequence alignment of DNA we can obtain a lot of information about a person's heath and makeup. This is critical in cases such as detection of diseases as well as targeted and personalized drug delivery mechanisms. After this information is obtained, it has to be classified. Classification algorithms are essentially utilized to predict which class a data point would lie based on the training dataset given to the supervised algorithm. Classification algorithms are useful in mining biological database and genomic data and classifying patients or organisms according to risk of developing certain diseases, hereditary or otherwise. An example would be one which divides patients into classes according to diseases that they may be susceptible to. When checking for lung cancer in a group of patients, we can divide them into two classes. One class will contain the patients that have a high risk for developing cancer. The other is a low cancer risk class. If we are dealing with more than one type of cancer then we can have multiple classes like 'low lung cancer risk', 'high lung cancer risk', 'low pancreatic cancer risk', 'high pancreatic cancer risk'. If we have medium risk patients, we can also create a class and set its parameters such that patients with medium risk will be classified into it. Now the datasets that will be supplied for processing are of two kinds. There is a training dataset in case the algorithm is supervised. There are also test datasets that check the accuracy and precision that is being generated by the classification algorithm. Some of the shortcomings of standard implementations of these algorithms are inaccurate predictions when datasets are large as well as slow running time. The issue of high computational cost is also significant. We can alleviate these problems by parallelizing these algorithms using MapReduce. Thus a Big Data approach can make our task much faster and more reliable and with a greater degree of accuracy.

## 2. DNA SEQUENCE ALIGNMENT ALGORITHMS

### 2.1 Burrows-Wheeler Alignment (BWA)

According to Li, H. et al. [1] the Burrows–Wheeler transform is used to modify a character string and produce rows of repeated characters. This becomes a very useful method for compressing a text. There is also no need to store the data and it can be reversed without any issues if required. It greatly improves the accuracy of text compression algorithms. The output string has various instances where a single character is repeated multiple times in a row when the original string contains repeated substrings. The use of Burrows–Wheeler transformation in NGS is when DNA is broken down into small pieces and the initial pieces are sequenced. This leads to the formation of reads which are around 30 to 500 base pairs long. The results obtained by this process were accurate but not very fast. These reads are then aligned to a reference genome. The memory requirement of this algorithm is also not high. The slow speed of this algorithm can be overcome by parallelizing the method by a MapReduce implementation that will help in overcoming this problem.

## 2.2 Smith-Waterman (SW) Algorithm

As stated in Farrar, M. [2], the Smith–Waterman (SW) algorithm is used for obtaining local alignment of sequences. Here the entire sequence is not compared; only the local segments of small but variable sizes are compared. It utilizes dynamic programming and is a modification of the Needleman–Wunsch algorithm. Since it uses DP it finds the optimal alignment with 100% accuracy. But because of this it is also very time consuming and for two sequences of length x and y it takes O(xy) magnitude of time. In order to execute faster but with less accuracy, heuristic methods like BLAST and FASTA can be used. This algorithm can be sped up using graphics processing units (GPUs). This slow speed of this algorithm can be remedied by parallelizing the method by a MapReduce implementation that will help in overcoming this problem. Nvidia's CUDA platform supports programs that can provide further speedup. It is usually accurate for smaller sequences but less so for larger ones.

## 2.3 BLAST Algorithm

Schatz, M. [3] states that BLAST stands for Basic Local Alignment Search Tool. It is also a local alignment algorithm. This alignment occurs by assigning scores to each alignment and only retaining scores that cross a certain threshold while at the same time eliminating scores that don't. BLAST is not a full alignment algorithm but a heuristic one and hence is much faster but less accurate than the Smith–Waterman algorithm. The query and database sequences thus are not optimally aligned. BLAST is also less time consuming than FASTA because it considers regions of higher significance. BLAST misses matches which are outliers and hence are harder to locate. BLAST also offer space requirement advantages over Smith-Waterman. And even when SW is sped up using GPUs, a parallel implementation of BLAST on MapReduce such as BlastReduce is significantly faster. This processing speed of this algorithm can be remedied by parallelizing the method by a MapReduce implementation that will help in overcoming this problem.

## 3. CLASSIFICATION ALGORITHMS USED IN DATA MINING

Classification algorithms help in predicting which class a data point would lie based on the training dataset given to the supervised algorithm. Classification algorithms are used in mining biological database and genomic data and classifying patients or organisms according to risk of developing certain diseases. If we wish to divide patients into classes according to diseases that they may be susceptible to. When checking for pancreatic cancer in a group of patients, we can divide them into two classes. One class will contain the patients that have a high risk for developing pancreatic cancer. The other is a low pancreatic cancer risk class. If we are dealing with more than one type of cancer then we can have multiple classes like 'low breast cancer risk', 'high breast cancer risk', 'low pancreatic cancer risk', 'high pancreatic cancer risk'. If we have medium risk patients, we can also create a class and set its parameters such that patients with medium risk will be classified into it. There are two types of datasets – a training dataset in case the algorithm is supervised and test datasets that check the accuracy and precision that is being generated by the classification algorithm. The disadvantages of standard implementations of these algorithms are inaccurate predictions when datasets are large as well as slow running time. High computational cost is also a significant disadvantage. We can alleviate these problems by parallelizing these algorithms using MapReduce framework.

### 3.1 K Nearest Neighbor

According to Ayma, V.A. et al. [4], K Nearest Neighbor is a supervised algorithm that uses an analogical model. However it is a lazy classifier. This means that when we supply it with a labelled training set, it only stores the data but makes no attempt to classify it. It starts classification only when we provide the test dataset. kNN is most comfortable when the neighbours all belong to the same class. When that is not the case, either a vote is taken with the new data point going to the class with most votes or else nearer neighbours are given more weightage and further neighbours less and a weighted vote is polled. The value of the user defined constant k is not very high usually in order of 10. A MapReduce implementation of kNN improves its performance. The method of implementation is not complicated. Also speed of the training phase is high. However space complexity is significant. It is also highly noise sensitive. Testing phase is time consuming. The shortcomings of standard implementation are various. First the dimension of k determines the quality of prediction. Also large datasets are not dealt with efficiently. Moreover it is sensitive to noisy data and is expensive from a computational standpoint. The advantages of parallelization via MapReduce is multiple like communication cost which is greatly decremented. Also accuracy and speed is improved significantly.

### 3.2 Support Vector Machine

As stated in He, Q. et al. [5], Support Vector Machine is perhaps the most accurate algorithm in general but the training phase of this supervised algorithm is considerable. SVM finds a separating hyper plane between classes by maximizing the margin between the two classes. This optimal classification can then be generalized for unknown datasets. The time and spatial complexity of SVM is directly proportional to the size of the dataset. SVM is a binary linear classifier. It uses a probabilistic model and when new data is supplied to it outputs the class to which the data point belongs. The hyper plane providing optimal separation gives maximum distances to the closest data point of any class in the training set. This helps in the most accurate generalization when new data points are supplied. This method is also robust at handling a limited number of outliers. A kernel trick can be used in cases where the data is projected to higher dimensions when the hyper plane cannot be efficiently generated in lower dimensions and then translated back. In case of very large datasets this method is not very efficient because when the number of outliers is high, the hyper planes may be not be easily generated. The computation can be sped up using parallel implementation via MapReduce. This also allows us to decrease the requirement of memory. The results provided are the most accurate. Nonlinear data points can be tackled. The issue of over-fitting has less of an impact on the results. These disadvantages can be overcome by the process of parallel implementation on MapReduce.

According to Chu, F. et al. [6], support vector machines help in detection of carcinoma inferring from genomic data and protein prediction. Results produced by this algorithm were highly accurate with a smaller gene pool in comparison with currently prevalent processes of detection. But protein prediction with SVMs did not achieve any advantages. Here the SVMs helped in eliminating redundant genes. The disadvantages are that overhead of computation is high. The results obtained were such that choice of the right kernel function is pivotal to the quality of results obtained. The training phase is also critically time consuming. The results from SVM are useful when there are two classes. But when the number of classes is greater, it requires modification. The shortcomings of standard implementation are space complexity is high when large number of data points are present. Also processing time is significant including training time for large number of data points. The results obtained by parallelization which helped were such that less time required for training phase. Also the processing time decreases. Finally accuracy increases because of fault tolerance in HDFS.

### 3.3 Naive Bayes classifier

According to Pakize, S. et al. [7] Naïve Bayes theorem is based on Bayes' theorem. It is called naïve because it assumes that the attributes or the variable are independent. This is not always the case and can lead to problems. The theorem probabilistically predicts which class a data point belongs to. It is used to process strings such as words in a document and hence can also be used to classify mapped reads. It deals with joint probability distribution of these words and helps us in classifying them. But because the order of the words or the sequences is ignored, it may sometimes give less than satisfactory results. But in certain cases like spam filtering the method works surprisingly well. It works well with large datasets and where attributes are several in number. It is also very simple and easy to implement. The method can also be parallelized without hassle. The two stages are training and prediction respectively. After this supervised algorithm has passed through the training phase, its output can then be utilized to analyze data in the prediction phase. A parallelized implementation of Naïve Bayes is both accurate and fast. The advantage is that computation process is simple. Also when datasets are large, the accuracy and speed of this algorithm is higher than others. Since it assumes independence among the variables, it gives less accurate results when the attributes are dependent. The shortcomings are such that the variables are assumed to be independent. Also sometimes problems occur in large datasets when the variable are strongly correlated. We can remedy these shortcomings by parallelizing via MapReduce. This process leads to a state such that less time required for training phase. Accuracy increases because of fault tolerance in HDFS. Also parallelized Naïve Bayes is able to handle large sets of data points better than other classification algorithms.

### 3.4 C4.5 (Decision Tree)

As elaborated in Nandakumar, A. et al. [8], C4.5 is used to classify data points by constructing a decision tree. The tree is incremented by using a divide and conquer approach. The advantage that C4.5 provides is in cases where it is not a binary classification problem but there are more than two sets of data. If we are dealing with a data set on cancer then we can have multiple classes like 'low lung cancer risk', 'high lung cancer risk', 'low pancreatic cancer risk', 'high pancreatic cancer risk'. Since we are dealing with more than two classes, here the standard decision tree algorithm is not applicable. In such cases we require C4.5. A shortcoming is that with large data sets this method becomes very slow. A parallelized implementation of this algorithm is also not as efficient as parallelized versions of the other algorithms as each subtree will need to be combined in the final decision tree. Decision trees are also much easier to interpret as the output is user readable and can be instinctively understood by people lacking in technical knowledge. The advantage of this method is that it can be constructed when availability of information about the dataset is low. Also the outcomes of the

decisions are assigned exact numerical values and this reduces ambiguity of interpretation. Also higher dimensional data can be handled with ease. Moreover non-technical people can interpret it as the output is user readable. Finally besides numerical data, categorical data can also be handled. The major disadvantage is that the process can be considerably time consuming. Similar to other methods this can also be remedied by a parallelized approach on MapReduce which allows us to attain better speedup as well as accuracy.

According to Dai, W. et al. [9], the C4.5 algorithm is converted to mapper and reducer programs. The communication overhead is also reduced. In this algorithm the ratio of information gain was selected rather than the value of information gain in order to choose which branch to select at a decision point. Both discrete and continuous data can be utilized. Missing data points can be accounted for with a certain degree of robustness. Also over fitting can be managed through a process of pruning. The utilization of specialized data structures helped in the reduction of communication overhead. The mapping and reducing programs were pipelined for efficient processing. The results obtained via output can contain only one attribute. The output is also categorical. It is not very stable and is dependent on the input dataset. The tree generated becomes very complex in case the dataset is numeric in nature. Outcomes of the tree are based on expectation and hence actual outcomes may vary. With more decisions, the complexity increases and the accuracy decreases. All these can be fixed by parallelization. By this method the algorithm becomes highly scalable. The method is more robust against outliers and wrong/missing values. It can handle numerical data as well as categorical data points.

## 4. AN INTRODUCTION TO BIG DATA, MAPREDUCE AND HDFS

According to Grolinger, K et al. [10] Big Data consists of datasets so large in size that they cannot be efficiently stored, processed and analyzed by existing standardized practices. The data can be both numerical and categorical in nature. Biological data, especially human genetics generates a huge deluge of data. Some of these databases can contain around 20 petabytes of data including redundant copies for backup. More importantly this data is at least doubling every year and very soon the existing storage capacity will be exhausted. So this data needs to be reduced in sized and then analyzed and this is the essence of MapReduce. Apache Hadoop is the open-source implementation of MapReduce. The cluster of machines provides us with a highly available system. Because such a large cluster leads to higher probability of failure, there is redundancy in terms of backup nodes. These nodes are also monitored in the application layer. The advantages are that it is faster, more reliable. The shortcoming seen in the results is that there is some overhead due to network latency. This can be remedied by reducing the latency so that optimal speedup can be obtained.

As elaborated in Dean, J. et al. [11], MapReduce is a technique that allows us to 'map' a large dataset into key-value pairs and then reduce the volume of data into sorted sets which allow us to infer about and reach certain conclusions regarding the original set of data. The programmer has to supply a map function consisting of a key/value pair as an input and that is converted to a set of intermediate key/value-pairs. The reducer function then merges the intermediate values associated with the same intermediate key. In the mapping phase the dataset is divided among and processed by mappers that operate on different nodes of a cluster or maybe multiple processing cores of the same computer. The intermediate (key, value) pairs are then passed on the reducers. The data having the same key are sorted into the same classes by the reducers and then they combine different values with the same key and sum them up and the resultant values of the different classes are generated. The results are then written to the HDFS i.e. the Hadoop Distributed File System. The mappers and reducers operate in parallel by dividing the large dataset and hence we can achieve significant parallelism in terms of processing. The overhead that is generated by network latency and other issues prevents the attainment of optimal speedup.

According to Shvachko, K. et al. [12], The Hadoop Distributed File System is a distributed file system. It allows us to divide and store large datasets over several nodes in a cluster which is an essential requirement of any MapReduce program. It uses commodity hardware and hence occurrences of failure is large. Hence backup nodes provide support in cases of failure. This introduces redundancy but is essential in such cases. The results obtained in this method are obtained much faster. They can reduce runtimes from around 2-3 days to even 2-3 hours. The throughput is also significant because of the parallelization of the task. The advantages are that it is faster, more reliable. The shortcoming seen in the results is that there is some overhead due to network latency. This can be remedied by reducing the latency so that optimal speedup can be obtained. The speedup depends on the number of parallelizing machines which can be the number of nodes in the cluster or maybe even the number of cores in the CPU. If there are 4 cores the ideal speedup should be 4x. But network latency is such that it is usually less than 4x. The fix for this is to use a greater bandwidth between the machines.

As mentioned in Kishor, D. [13] the user sends a key value pair to the mapper. This consists of one line of input from the HDFS file. This usually consists of strings of data like log files where there is a store and the value of a transaction or a website and the number of hits to the website. Every such line leads to the intermediate generation of new key/value pairs by dividing the line after each token. The token is assigned to a word and defines a new value. As a key the store name or the website is used. All values for a particular log file or document are sent to the reducer after the mapping phase. An inverted index may be used to provide a list of terms with all files

in which they are present. Then it produces separate classes and according to the class in which each item belongs we assign it accordingly. We iterate over the entire set of values and they are added up into a list that corresponds to that class. The ultimate output consists of keys and their corresponding lists. The output is then written to the Hadoop Distributed File System. The advantages are that it is faster, more reliable. The shortcoming seen in the results is that there is some overhead due to network latency. This can be remedied by reducing the latency so that optimal speedup can be obtained.

## 5. CONCLUSION

Some of the significant points we have noted after surveying the papers are that to make sure the processing is not slowed down we have to remove the outliers and inappropriate attributes. In order to select the appropriate attributes we can utilize feature selection methods. Also it has been seen after studying the papers that there is no single method that gives best results for every set of data. Thus before selecting a particular method we use small test datasets and compare their performances. It has been seen that better results are obtained when the same data sets are used for both training and testing phases. We also see that classification algorithms fail when little to no information is present about the datasets since training phase cannot be initiated. In such cases clustering algorithms are the only option. Classification algorithms suffer from a shortcoming that they do not report the degree of association among the attributes. Hence a combination of classification and association algorithms can be used in a hybridized manner to achieve desired outcomes We have seen that out of the 4 classification algorithms in general SVM was the most accurate and efficient. It only suffered when the dataset grew very large. In those cases Naïve Bayes gave better results. kNN was the worst performing algorithm in most cases. C 4.5 was useful when user readability of output was critical which was achieved through the generation of a decision tree.

After analyzing all existing work we come to the conclusion that the use of SVM leads to the most efficient and elegant solutions to this class of problems. But classifying data from the genomic databases by non-parallel implementations is a very time consuming process. So we are suggesting a method in which we are utilizing the MapReduce paradigm to implement SVM via SciKit using python code. We have identified that by mapping the data to higher dimensions using a kernel trick and modifying the C value accordingly we can obtain a hyper plane in those instances where generating one would be impossible in two dimensions. After the generation of the hyper plane we are retranslating the data back to two dimensions. This would enable us to attain optimal speedup as well as reliability that is obtained from the fail-safe backup nodes in the cluster. We will also try to use higher bandwidth between the nodes in the cluster so that in case N machines are used we can attain results with up to Nx speedup or at least very close to it.

**REFERENCES:**

[1] Li, H., Durbin, R., 2009. Fast and accurate short read alignment with Burrows–Wheeler Transform. Wellcome Trust Sanger Institute, Wellcome Trust Genome Campus, Cambridge, CB10 1SA, UK. Journal of Bioinformatics. pp. 56-67.

[2] Farrar, M., 2006. Striped Smith–Waterman speeds database searches six times over other SIMD implementations. Journal of Bioinformatics. pp 34-42.

[3] Schatz, M., 2014. BlastReduce: High Performance Short Read Mapping with MapReduce. University of Maryland Center for Bioinformatics and Computational Biology. International Journal of Innovative Research in Advanced Engineering, pp 345-359.

[4] V. A. Ayma, R. S. Ferreira, P. Happ, D. Oliveira, R. Feitosa, G. Costa, A. Plaza, P. Gamba 2015. Classification Algorithms for Big Data Analysis, a MapReduce approach.TIAPR, Munich, Germany, pp. 17-21.

[5] He, Q., Zhuang, F., Li, J., and Shi, Z., 2010. Parallel Implementation of Classification Algorithms Based on MapReduce. International Conference on Rough Set and Knowledge Technology, pp. 655-662.

[6] Chu, F., Jin, G., Wang, L., 2015. Cancer Diagnosis and Protein Secondary Structure Prediction Using Support Vector Machines. School of Electrical and Electronic Engineering Journal. Pp. 133-145.

[7] Pakize, S., and Gandomi, A., 2014. Comparative Study of Classification Algorithms Based On MapReduce Model. International Journal of Innovative Research in Advanced Engineering, 1(7), pp. 215-254.

[8] Nandakumar, A., and Yambem, N., 2014. A Survey on Data Mining Algorithms on Apache Hadoop Platform. International Journal of Emerging Technology and Advanced Engineering, 4(1), pp. 563-565.

[9] Dai, W., and Ji, W., 2014. A MapReduce Implementation of C4.5 Decision Tree Algorithm. International Journal of Database Theory and Application, 7(1), pp. 49-60.

[10] Grolinger, K., Hayes, M., Higashino, W., L'Heureaux, A., Allison, D., and Capretz, M., 2014. Challenges for MapReduce in Big Data. IEEE 10th World Congress on Services, pp. 182-189.

[11] Dean, J., and Ghemawat, S., 2006. MapReduce: Simplified Data Processing on Large Clusters. Proceedings of the 6th Conference on Symposium on Opearting Systems Design and Implementation, 6, pp. 137-149.

[12] Shvachko, K., Kuang, H., Radia, S., and Chansler, R., 2010. The Hadoop Distributed File System. IEEE 26th Symposium on Mass Storage Systems and Technologies, pp. 1-10.

[13] Kishor, D., 2013. Big Data: The New Challenges in Data Mining. International Journal of Innovative Research in Computer Science & Technology, 1(2), pp. 39-42.