International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:11, No:7, 2017

FCNN-MR: A Parallel Instance Selection Method Based on Fast Condensed Nearest Neighbor Rule

Lu Si, Jie Yu, Shasha Li, Jun Ma, Lei Luo, Qingbo Wu, Yongqi Ma, Zhengji Liu

***Abstract*—**Instance selection (IS) technique is used to reduce the data size to improve the performance of data mining methods. Recently, to process very large data set, several proposed methods divide the training set into some disjoint subsets and apply IS algorithms independently to each subset. In this paper, we analyze the limitation of these methods and give our viewpoint about how to divide and conquer in IS procedure. Then, based on fast condensed nearest neighbor (FCNN) rule, we propose a large data sets instance selection method with MapReduce framework. Besides ensuring the prediction accuracy and reduction rate, it has two desirable properties: First, it reduces the work load in the aggregation node; Second and most important, it produces the same result with the sequential version, which other parallel methods cannot achieve. We evaluate the performance of FCNN-MR on one small data set and two large data sets. The experimental results show that it is effective and practical.

***Keywords*—**Instance selection, data reduction, MapReduce, kNN.

I. Introduction

THE process of knowledge discovery and data mining has six steps: problem specification, problem understanding, data preprocessing, data mining, evaluation and result exploitation [1]. Data mining is the essential process and a large number of well-known techniques (such as classification, regression, clustering and so on) have been proposed in many applications. However, the quality of data, which serves as the input for data mining, has a great impact on the mining performance. Low-quality data will lead to low-quality mining results [2]. In the real world, raw data sets may contain a lot of redundant, erroneous, missing and noisy values and not perfectly suitable for data mining task. Data preprocessing techniques are applied before mining and will improve the quality of data and mining results. In this paper, we focus on data reduction (DR), one of the data preprocessing techniques, aiming to reduce the size of training sets for k nearest neighbor (kNN) decision rule [3].

The kNN technique is a simple classification method and is widely used in the field of data mining. It has high storage requirement and is computationally expensive due to the comparison of every instance in the training set to find the closest neighbors. In order to reduce the runtime and storage requirements, several methods have been proposed [4]–[7] to reduce the size of the stored data for the kNN, such as instance selection, feature selection and feature extraction. Among these techniques, instance selection techniques aim to choose a subset from the total available data by removing instances that are noisy and/or redundant. A successful algorithm can significantly reduce the size of training set without a significant reduction of generalization accuracy [8].

Instance selection is usually divided into three approaches: *selection*, *abstraction* and *hybrid*. *Selection* algorithms maintain a subset of the original instances and *abstraction* algorithms modify the instances using a new representation. *Hybrid* algorithms are based on the combination of *selection* and *abstraction* methods. In comparing instance selection algorithms, reduction rate and classification accuracy are two main evaluation criteria. However, with the exponential growth of data in many application domains such as industry, medicine and financial businesses, processing very large scale data sets for instance selection is becoming a major limitation and a fast execution speed of reduction is definitively needed. The traditional sequential methods lack enough scalability to cope with data sets with millions of instances even though they have already been performed over the previous smaller data set.

Recent improvements in this field cover the stratification of data and redesign the algorithms and their inclusions in parallel environments [9]–[13]. Among these methods, [10], [11] implement the algorithms with MapReduce [14] framework which was proposed by Google and has been widely used to process big data problems on computing clusters.

In this paper, we analyze the limitation of these methods and propose a large data sets instance selection method by redesigning fast condensed nearest neighbor (FCNN) rule [15], [16] in a MapReduce way, namely FCNN-MR. Unlike other proposed parallel methods that apply IS on disjoint subsets independently and finally merge partial results together, we parallel the prediction process and select informative instances in the aggregation node. It reduces the work load in the final single aggregation node and can produce the same result with the sequential version.

The rest of this paper is organized as follows: Section II discusses the related work and preliminaries. In Section III, we analyze the limitation of previously proposed methods and give out our viewpoint about how to divide and conquer for instance selection. Section IV illustrates the design and implementation of FCNN-MR. The performance evaluation results of FCNN-MR are given in Section V. We discuss the limitation and future work in Section VI. Finally, the paper is concluded in Section VII.

Lu Si is with the National University of Defense Technology, China (e-mail: lusi@ubuntukylin.com).

Jie Yu, Shasha Li, Jun Ma, Lei Luo and Qingbo Wu are with the National University of Defense Technology, China.

Yongqi Ma and Zhengji Liu are with the Institute of Computer Application, China Academy of Engineering Physics, China.

II. Related Work and Preliminaries

In this section, we briefly review some of the most relevant methods (Section II. A) and describe two preliminary works, FCNN (Section II. B) and MapReduce (Section II. C). In this paper, we use T as original instances in the training set and S represents the subset of T. That is to say, instance selection is searching for a subset S of instances to keep from training set T.

A. Instance Selection

The condensed nearest neighbor (CNN) rule [17] is the first and probably the simplest instance selection strategy. It begins with an empty subset S, and adds each instance from training set T to S if the instance is misclassified using only the instances in S. However, noisy instance will usually be misclassified by their neighbors, and thus will be retained in S. Generalized condensed nearest neighbor (GCNN) rule [18] is a recent extension of CNN. GCNN assigns an instance to S if it satisfies an absorption criterion according to a threshold. It can reduce the size of S compared to CNN. Reduced nearest neighbor (RNN) rule [19] starts with $S = T$ and removes each instance from S if the removal does not cause other instances in T to be misclassified by the remaining instances in S. It is able to remove noisy instances, but is more expensive in terms of learning time compared to CNN. Edited nearest neighbor (ENN) rule [20] proposed by Wilson also starts with $S = T$, and then removes any instance that would be misclassified by its k nearest neighbors (with k = 3, typically). This algorithm removes noisy and close border instances. Repeated ENN (RENN) rule applies the ENN algorithm repeatedly until all instances in S have a majority of their k nearest neighbors with the same class. Five decremental reduction optimization procedure algorithms (DROP1 - DROP5) [4] were proposed by Wilson and Martinez. Among them, DROP3 is the most successful method. In DROP3, each instance has k nearest neighbors called associates. The algorithm removes it if at least as many of its associates in T would be classified correctly without it (where $S = T$ originally). Furthermore, DROP3 uses ENN to remove noisy instances.

Recently, some new instance selection methods were proposed by different researchers. Based on outlier pattern analysis and prediction, Lin et al. [21] proposed an approach to detect the representation instances from large data sets. IRAHC [8] maintains a hyperrectangle and removes interior instances and keeps border and near border ones. [22] utilizes the fuzzy-rough instance selection method based on weak gamma evaluator to remove redundant and erroneous instances. [23] proposed a fast instance selection method for large data sets by clustering. This algorithm selects border instances and some non-border instances. Evolutionary algorithm [24] is one type of instance selection. In this method, some initial sets of instances are represented by chromosome strings. According to a fitness function, the individuals are evaluated and the best chromosomes are selected after a specific number of iterations.

Nowadays, in order to deal with huge data sets that additionally involve millions of instances, some distributed algorithms [9]–[13] were proposed. [9], [12], [13] divide the original training set into small subsets where the instance selection algorithms are applied and then merge them into one subset. [10], [11] proposed MapReduce-based frameworks for nearest neighbor classifier to deal with the classification problems of large data sets. They partition the large data sets into some small subsets and selects informative instances in Map phase with traditional instance selection algorithms. In Reduce phase, they collect the selected instances from different nodes to rejoin a subset. We will analyze the limitations of these parallel methods in Section III.

B. Fast Condensed Nearest Neighbor

Fast condensed nearest neighbor (FCNN) rule [15], [16] proposed by F. Angiulli is one of the state-of-the-art instance selection algorithms and exhibits excellent performance in either maintaining classification accuracy, reduction rate, or execution speed. FCNN retains border instances and order independent. It has sub-quadratic time complexity and requires few iterations to converge.

Algorithm 1 Fast Condensed Nearest Neighbor Rule

Input:
Original training set T
Output:
The reduced set S
$S = \varnothing$
$\Delta S = \varnothing$
Add centroids of each class to ΔS
repeat
  $S = S \cup \Delta S$
  $\Delta S = \varnothing$
  for each instance $I \in S$ **do**
    Add the nearest enemy inside it's Voronoi region to ΔS
  end for
until $\Delta S = \varnothing$

FCNN searches for informative instances in an incremental manner. That is to say, it begins with an empty subset S, and adds each instance in T to S if it fulfills the criterion. Algorithm 1 shows the complete procedure of FCNN. First of all, the subset S is initialized by the centroids of each class in T. The centroid of each class is the instance which is closest to the geometrical center of the class region. Then, during each iteration, for each instance I in S, its nearest enemy inside its Voronoi region (consists of all instances closer to it than to any ohter.) is added to a temporary set ΔS. The process is performed repeatedly until ΔS is empty (no enemy is found), i.e., all instances in T are correctly classified using S. Further more, FCNN algorithm can be extended to the case in which the kNN rule are taken into account. In this case, it leverages k nearest neighbors in S to predict the instance class in T - S.

C. MapReduce

MapReduce [14] was proposed by Google in 2004 and has become a popular distributed and parallel computing model for large-scale data-intensive computations since then. In MapReduce, the input data for a job are split into some

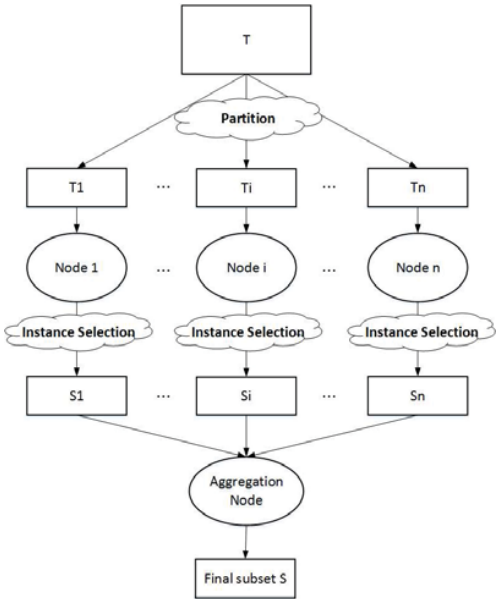

Fig. 1 Flowchart of other proposed parallel instance selection methods

data chunks called *inputsplit* and stored in a distributed file system [25]. A MpaReduce operation essentially includes two phases: Map and Reduce. Each one has ⟨key, value⟩ pairs as input and output. In Map phase, each *inputsplit* is assigned to one map worker which invokes map functions to compute and generate a set of ⟨key, value⟩ intermediate pairs. In Reduce phase, a user-defined reduce function is applied to all intermediate values associated with the same intermediate key and produces a list of output values.

Hadoop is one of the most popular implementation of MapReduce and is released under the umbrella of the Apache Software Foundation. It is an open source framework and written in Java. In this work, we implement the proposed algorithm on Hadoop platform.

III. How to Divide and Conquer

Selecting informative instances from large-scale data sets is a challenging task. Divide and conquer algorithm can break down a problem into two or more sub-problems that are simple enough to be solved. Then, the solutions to the sub-problems are combined to give a final solution to the original problem. Inspired by the divide and conquer methodology, [9]–[13] divide the original training set T into many disjoint subsets T_i, and then apply traditional instance selection algorithm over each subset independently to get a partial reduced set S_i. Finally, they merge all partial reduced sets S_i into a final set S (as described in Fig. 1).

However, the traditional instance selection algorithms have to know the whole data set and select the nearest neighbor of an instance. These distributed previous methods lack the information of the whole problem and may lead to the introduction of noisy and redundant instances. Like [15], [16], we generated a data set composed by 9,000 instances with 2 attributes. Instances were randomly distributed into the unit square and partitioned in two classes by a circle of diameter 0.5. We divided the data set into n splits, applied FCNN algorithm on them independently, and merged the results together. As we can see in Fig. 2, with the number of split increasing, more noisy and redundant instances are remained.

To mitigate this problem, [9] repeats the divide and aggregation procedure in a recursive manner; [11] introduced a voting mechanism to select informative instances from partial result sets S_i; [10] leverages a filter to remove noisy and redundant instances in rejoining phase. However, these remedies can relieve the problem to some extent, they do not actually implement the traditional instance selection algorithm in a parallel way. More fatally, the size of rejoined set is increasing with the split number. The final process of aggregation and selecting requires to perform in a single compute node. In [10], they perform IS methods again to reduce the merged subset, the compute node may become a bottleneck in the distributed system.

Instead of applying the instance selection algorithm independently, FCNN-MR uses the whole training set and parallels the procedure of searching nearest neighbors. In FCNN-MR, the reduced result is the same with sequential version of FCNN. Meanwhile, it reduce the work load in aggregation node and has a good performance of scalability. In the next section, we will introduce the implementation of our approach.

IV. FCNN-MR

This section describes the proposed MapReduce approach for instance selection, FCNN-MR. FCNN-MR algorithm is based on FCNN to select representative instances from training set. During each iteration of FCNN, the nearest enemy inside the Voronoi region of each instance in S is found and added to S. We redesign FCNN and parallelize this algorithm following a MapReduce procedure. The working way of FCNN-MR is illustrated in Fig. 3.

First, FCNN-MR initializes the subset S by the centroids of each class. Second, the splitting procedure of MapReduce divides the training set T into n disjoint subsets (T_i) of instances. Then, FCNN-MR allocates T_i and S to every compute nodes, and each T_i subset is processed independently producing several intermediate ⟨label, instance⟩ pairs by the corresponding Map task. During the Shuffle phase, the system combines the instances with the same label to minimize network overloading and routes them to the reduce node. Finally, the reduce phase collects all intermediate inputs and uses only one Reduce task to form the final output for one iteration. This above procedure is repeated until the criterion is fulfilled (see Algorithm 2).

The mapper and reducer function are described in Algorithm 3 and Algorithm 4 respectively. For each input instance from T_i, the mapper function finds whose Voronor region it locates in and emit a ⟨key, value⟩ pair if it is the enemy of the region. The key of the pair is the label of the Voronor region. Briefly speaking, the mapper function predicts the class of the instance from T_i using the nearest neighbors in

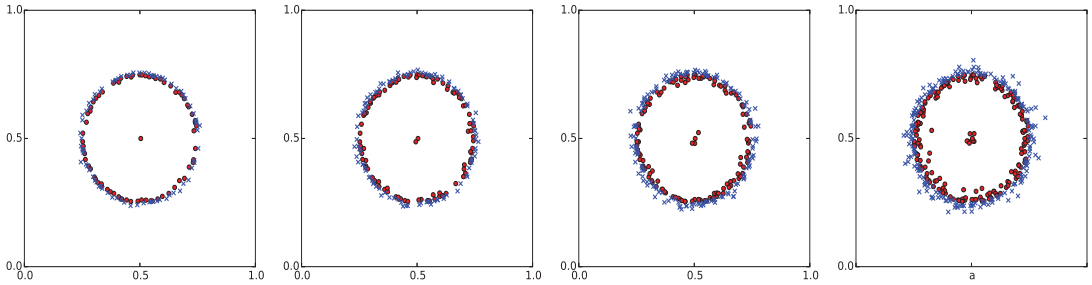

Fig. 2 Example of subset computed by FCNN with different split number

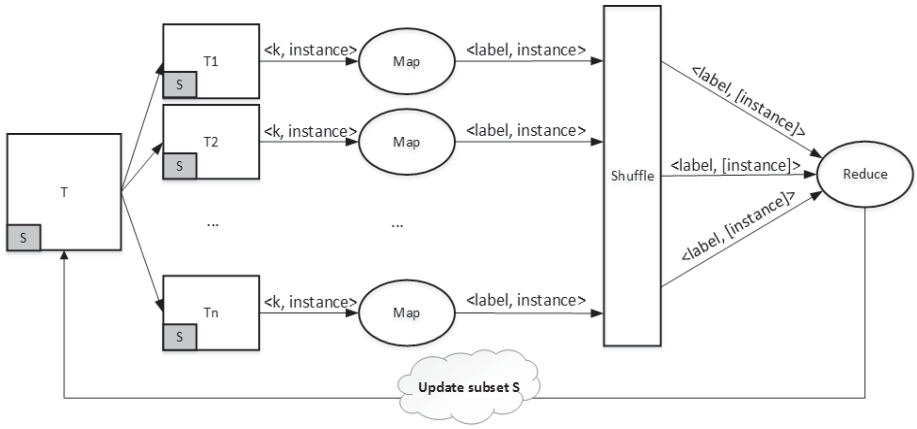

Fig. 3 Flowchart of FCNN-MR algorithm

Algorithm 2 FCNN-MR

Input:
Original training set T
Output:
The reduced set S
$S = \varnothing$
$\Delta S = \varnothing$
Add centroids of each class to ΔS
repeat
  $S = S \cup \Delta S$
  $\Delta S = \varnothing$
  for each instance $I \in S$ **do**
    Invoke a mapper function (Algorithm 3) to process
  end for
  Merge all intermediate results and invoke a reducer function (Algorithm 4) to generate ΔS
until $\Delta S = \varnothing$

Algorithm 3 The mapper function for FCNN-MR

Input:
$\langle$k, instance$\rangle$ pairs, subset S
Output:
null or $\langle$label, instance$\rangle$ pair
Get the class C of instance I from $\langle$label, instance$\rangle$
Predict the class C^* of I by the nearest neighbor in S
if $C \neq C^*$ **then**
  emit $\langle$label, instance$\rangle$ pair
end if

Algorithm 4 The reducer function for FCNN-MR

Input:
$\langle$label, [instance]$\rangle$ pairs, subset S
Output:
A temporary set ΔS
$\Delta S = \varnothing$
for each $\langle$label, [instance]$\rangle$ pair **do**
  Add the nearest enemy to ΔS
end for

S and emit a $\langle$key, value$\rangle$ pair if the prediction is wrong. The reducer function computes the nearest one to the instance in S with the same label from each $\langle$label, [instance]$\rangle$ and adds them into a temporary set ΔS. The time complexity of this procedure is $\mathcal{O}(m)$ (m is the number of instances routed to the Reduce node). If ΔS is empty, the algorithm is finished and FCNN-MR will return the final output S. Otherwise, it will update S by absorbing ΔS to prepare the next iteration.

It is noteworthy that the update of subset S does not mean modifying the file in the distributed file system. It just creates a new file to store the new merged set S which will be loaded and allocated among the cluster before the next iteration.

V. Experiments

In this section, we present results obtained by the experiments. Section V. A describes the experiment environment and the data sets chosen. Section V. B shows the accuracy and reduction rate for FCNN-MR. Section V. C presents and discusses the impact of split.

TABLE I
THE CONFIGURATION OF NODES

Items	Configuration
Operating system	Ubuntu 15.10
JDK	JDK-7u71-linux
MapReduce	Hadoop 2.6.0
Maximum map tasks	512
Maximum reduce tasks	1

TABLE II
SUMMARY OF THE USED DATA SETS

Data set	#instances	#dimension	#classes
Page-blocks	5,473	10	5
Poker-hand	1,025,010	10	10
Mnist	70,000	784	10

A. Experimental Setup

The experiments for this paper were carried out on nine nodes in a cluster: eight compute nodes and a master. It should be noted that we used three physical machines as hosts and each of them contains three virtual machines. The master processes, NameNode and JobTracker, are hosted in the master node and a number of TaskTrackers and DataNodes are hosted in the compute nodes. The configuration of nodes in the cluster is given in Table I.

We selected a small data set (Page-blocks) and two large data sets (Poker and Mnist) to evaluate the performance of our approach. Page-blocks and Poker are from the UCI Machine Learning Repository and Mnist is from Mnist database. Table II summarizes the main characteristics of these data sets. In Table II, #instances, #dimension, and #classes denote the number of instances, the number of attributes and the number of classes in the data sets respectively.

Page-blocks comes from 54 distinct documents and classifies all the blocks of the page layout of a document. All attributes are numeric. In Poker-hand data set, each record is an example of a hand consisting of five playing cards from a standard deck of 52 and is composed of 10 attributes totally. MNIST database has a training set (60,000 examples) and a test set (10,000 examples). The digits have been size-normalized and centered in a 28×28 image. Each image has 784 attributes each of which is associated with a distinct pixel (ranges from 0 to 255).

B. Accuracy and Reduction Rate Results

To test the reduced sets selected by FCNN-MR, we used kNN classifier and the Euclidean distance. The reduction rate is the ratio of the number of removed instances to the number of instances in the original data set T. Based on the results shown in Fig. 4, we can see that FCNN-MR can keep the accuracy with the number of mapper. Further more, we list the numeric results in Table III, including the accuracy, reduction rate and iteration. We stress that we do not devote to optimize the classification accuracy and the reduction rate for a specific problem, but the parallel method can produce the same outputs with the sequential version.

TABLE III
RESULTS FOR THREE DATA SETS

Data set		Results		
		Accuracy	Reduction Rate	Iteration
Mnist	k = 1	0.9449	0.8954	20
	k = 3	0.9428	0.8876	19
	k = 5	0.9259	0.8908	19
Poker-hand	k = 1	0.5664	0.4275	30
	k = 3	0.4856	0.3658	31
	k = 5	0.4397	0.3502	32
Page-blocks	k = 1	0.9617	0.9100	41
	k = 3	0.9690	0.8972	36
	k = 5	0.9617	0.8810	32

C. Work Load in Aggregation Node

As we clarified in Section III, divide the training set into disjoint chunks and perform instance selection algorithm independently can lead to the introduction of noisy and redundant since the lack of the whole information. To alleviate these problems, MRPR [10] proposes three aggregation alternatives (*join*, *filtering*, and *fusion*) in the Reduce phase. *Join* simply concatenates all partial results S_i into a final set S. *Filterinng* and *fusion* remove noisy or redundant instances respectively and have better performance with respect to accuracy and reduction rate. Actually, *filtering* and *fusion* apply some other instance selection algorithms again on the merged subset. As described in MRPR, the accuracy and reduction rate decrease according as the number of available instances in the used training set is reduced. Besides these, the impact on performance of the Reduce node cannot be neglected since only one single node can be used. It may become a bottleneck in the system. Fig. 5 shows the number of instances stored in the Reduce node during the execution of MRPR with the data sets above described. We can see that the size of storage increases with the mapper number.

Table IV summarizes the result details and lists the storage requirement in FCNN-MR. Note that FCNN is an incremental algorithm and the size of subset S stored in the Reduce node increases with each iteration. Table IV lists the the size of S in the last iteration in which S is the largest. We can see that FCNN-MR requires lesser storage space which means less load in the Reduce node.

VI. DISCUSSION

FCNN also has some variants (FCNN2 - FCNN4) that can be redesigned following the proposed distributed model. During each iteration, FCNN2 adds the centroids of the enemies in each Voronoi region; FCNN3 only adds one closest enemy which belongs to the Voronoi region with most enemies; FCNN4 only adds one centroid which belongs to the Voronoi region with most enemies. They can use the same way as what FCNN-MR do (Algorithm 3) to predict each instance class in the Map phase and select different informative instances in the Reduce phase. The time complexity for these selecting procedures are $\mathcal{O}(m)$ (m is the number of $\langle$key, value$\rangle$ pair routed to the Reduce node).

Note that the distributed model is applicable to any instance selection method which only need to maintain a small part of the training set, like RMHC [26] and MCNN [27]. We plan to implement these methods based on the proposed model and evaluate the performance of them in the future work.

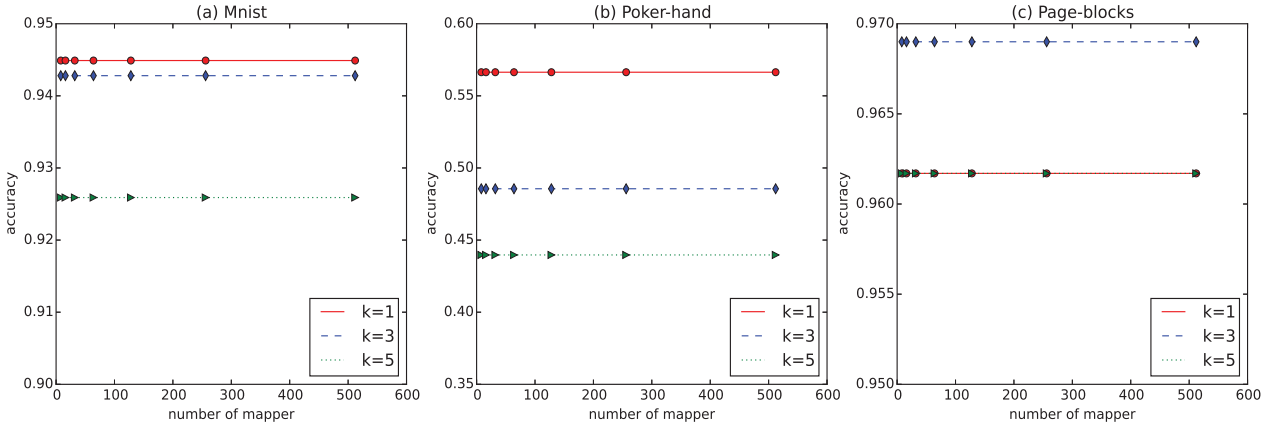

Fig. 4 The accuracy with different number of mapper

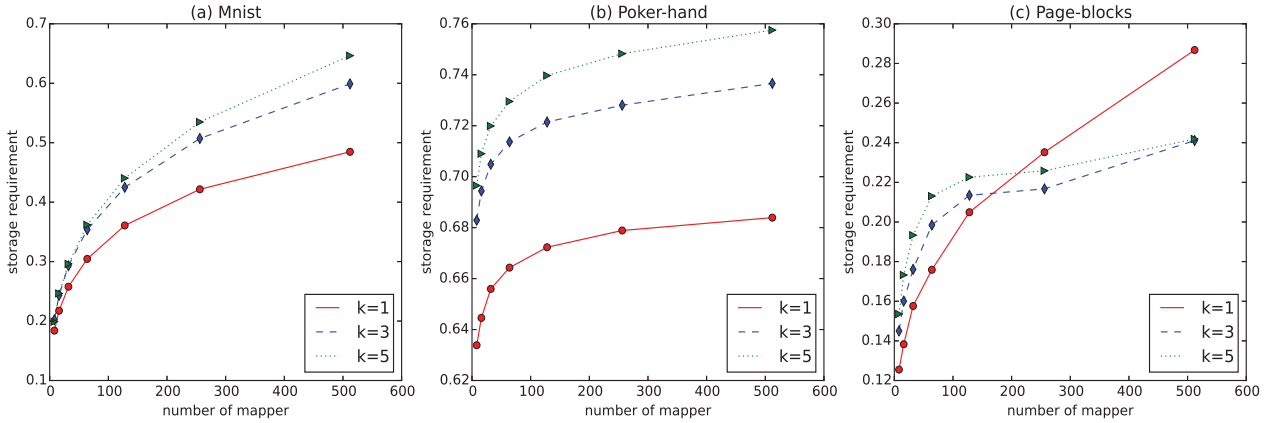

Fig. 5 Storage requirement in Reduce node with different number of mapper

TABLE IV
THE NUMBER OF INSTANCE STORED IN REDUCE NODE

Data set		Storage requirement							
		8	16	32	64	128	256	512	FCNN-MR
Mnist	k = 1	11034	13043	15464	18269	21646	25296	29079	6275
	k = 3	12132	14601	17575	21225	25494	30421	35933	6744
	k = 5	11934	14787	17766	21711	26416	32087	38780	6533
Poker-hand	k = 1	633859	644570	655943	664305	672290	678866	683913	572508
	k = 3	682903	694360	704845	713648	721447	728038	736599	634227
	k = 5	696499	708991	719889	729523	739670	748266	757521	649827
Page-blocks	k = 1	618	681	776	866	1009	1158	1412	443
	k = 3	714	788	867	977	1051	1067	1187	506
	k = 5	756	853	952	1049	1096	1112	1191	586

There are also some limitations. First, for decremental methods which start with $S = T$, it is impossible to allocate the whole very large data set into every compute node. However, they are applicable to other parallel models [9]–[13]. Second, each iteration in FCNN-MR will start a new task in the MapReduce system, it may spend more time to prepare and allocate jobs.

VII. CONCLUSIONS

Instance selection is an important preprocess task, specially for instance-based learning methods, since it can reduce the runtime in the classification stage. In this paper, the purpose is not to optimize the classification accuracy and the reduction rate with an instance selection method. We focus on the learning speed which is usually neglected but definitively needed when facing the very large scale data set. We propose a novel parallel instance selection method, namely FCNN-MR. Unlike other distributed methods [9]–[13] which simply apply instance selection methods on disjoint subset independently and merge the partial results together, FCNN-MR redesigns the prediction procedure of each instance in the Map phase and selects informative instances in the Reduce phase. Compared with other proposed parallel methods, FCNN-MR can reduce the work load in the aggregation node in the cluster and has a good performance of scalability. The key property of this parallel method is the same outputs with the sequential version.

REFERENCES

[1] S. Garcła, J. Luengo, and F. Herrera, "Data preprocessing in data mining," *Computer Science*, vol. 72, 2015.
[2] B. J. Han, "Data mining. concepts and techniques. 3rd ed," *Data Mining Concepts Models Methods & Algorithms Second Edition*, vol. 5, no. 4, pp. 1 – 18, 2000.
[3] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.
[4] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Machine Learning*, vol. 38, no. 3, pp. 257–286, 2000.
[5] M. Kudo and J. Sklansky, "Comparison of algorithms that select features for pattern classifiers," *Pattern Recognition*, vol. 33, no. 1, pp. 25–41, 2000.
[6] H. Liu and H. Motoda, "Feature extraction construction and selection: A data mining perspective," *Springer International*, vol. 94, no. 448, p. 014004, 1999.
[7] I. Triguero, J. Derrac, S. Garcia, and F. Herrera, "A taxonomy and experimental study on prototype generation for nearest neighbor classification," *Systems Man & Cybernetics Part C Applications & Reviews IEEE Transactions on*, vol. 42, no. 1, pp. 86–100, 2012.
[8] J. Hamidzadeh, R. Monsefi, and H. S. Yazdi, "Irahc: Instance reduction algorithm using hyperrectangle," *Pattern Recognition*, vol. 48, no. 5, pp. 1878–1889, 2015.
[9] Haro-Garc, A. Aida, Garc, and N. A-Pedrajas, "A divide-and-conquer recursive approach for scaling up instance selection algorithms," *Data Mining and Knowledge Discovery*, vol. 18, no. 3, pp. 392–418, 2009.
[10] I. Triguero, D. Peralta, J. Bacardit, S. Garcła, and F. Herrera, "Mrpr: A mapreduce solution for prototype reduction in big data classification," *Neurocomputing*, vol. 150, no. 150, p. 331C345, 2015.
[11] J. Zhai, X. Wang, and X. Pang, "Voting-based instance selection from large data sets with mapreduce and random weight networks," *Information Sciences*, vol. 367, pp. 1066–1077, 2016.
[12] H. Liu and H. Motoda, "On issues of instance selection." *Data Mining and Knowledge Discovery*, vol. 6, no. 2, pp. 115–130, 2002.
[13] J. R. Cano, F. Herrera, and M. Lozano, "Stratification for scaling up evolutionary prototype selection," *Pattern Recognition Letters*, vol. 26, no. 7, pp. 953–963, 2005.
[14] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters." in *Conference on Symposium on Opearting Systems Design & Implementation*, 2004, pp. 107–113.
[15] F. Angiulli, "Fast condensed nearest neighbor rule," in *International Conference*, 2005, pp. 25–32.
[16] Angiulli, "Fast nearest neighbor condensation for large data sets classification," *IEEE Transactions on Knowledge & Data Engineering*, vol. 19, no. 11, pp. 1450–1464, 2007.
[17] B. P. E. Hart, "The condensed nearest neighbor rule," in *IEEE Trans. Information Theory*, 1968.
[18] C. H. Chou, B. H. Kuo, and F. Chang, "The generalized condensed nearest neighbor rule as a data reduction method," vol. 2, pp. 556–559, 2006.
[19] G. W. Gates, "The reduced nearest neighbor rule," *IEEE Transactions on Information Theory*, vol. 18, no. 3, pp. 431 – 433, 1972.
[20] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems Man & Cybernetics*, vol. 2, no. 3, pp. 408–421, 1972.
[21] W. C. Lin, C. F. Tsai, S. W. Ke, C. W. Hung, and W. Eberle, "Learning to detect representative data for large scale instance selection," *Journal of Systems & Software*, vol. 106, no. C, pp. 1–8, 2015.
[22] A. Onan, "A fuzzy-rough nearest neighbor classifier combined with consistency-based subset evaluation and instance selection for automated diagnosis of breast cancer," *Expert Systems with Applications*, vol. 42, no. 20, pp. 6844–6852, 2015.
[23] J. A. Olvera-Lpez, J. A. Carrasco-Ochoa, and J. F. Martłnez-Trinidad, "A new fast prototype selection method based on clustering," *Pattern Analysis and Applications*, vol. 13, no. 2, pp. 131–141, 2010.
[24] J. R. Cano, F. Herrera, and M. Lozano, "Using evolutionary algorithms as instance selection for data reduction in kdd: an experimental study," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 6, pp. 561–575, 2004.
[25] D. Borthakur, "The hadoop distributed file system: Architecture and design," *Hadoop Project Website*, vol. 11, no. 11, pp. 1 – 10, 2007.
[26] D. B. Skalak, "Prototype and feature selection by sampling and random mutation hill climbing algorithms," *Machine Learning Proceedings*, pp. 293–301, 1994.
[27] V. S. Devi and M. N. Murty, "An incremental prototype set building technique," *Pattern Recognition*, vol. 35, no. 2, pp. 505–513, 2002.