Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 144 (2018) 140–152

Procedia Computer Science

www.elsevier.com/locate/procedia

INNS Conference on Big Data and Deep Learning 2018

Big Data Analytics based on PANFIS MapReduce

Choiru Za'in[a], Mahardhika Pratama[b], Edwin Lughofer[c], Meftahul Ferdaus[d], Qing Cai[b], Mukesh Prasad[e,f]

[a] *La Trobe University, Melbourne, Australia*
[b] *Nanyang Technological University, Singapore, Singapore*
[c] *Johannes Kepler University, Linz, Australia*
[d] *University of New South Wales, Canberra, Australia*
[e] *University of Technology Sydney, Sydney, Australia*
[f] *Center for Artificial Intelligence, School of Software, FEIT, University of Technology Sydney, Australia*

Abstract

In this paper, a big data analytic framework is introduced for processing high-frequency data stream. This framework architecture is developed by combining an advanced evolving learning algorithm namely Parsimonious Network Fuzzy Inference System (PANFIS) with MapReduce parallel computation, where PANFIS has the capability of processing data stream in large volume. Big datasets are learnt chunk by chunk by processors in MapReduce environment and the results are fused by rule merging method, that reduces the complexity of the rules. The performance measurement has been conducted, and the results are showing that the MapReduce framework along with PANFIS evolving system helps to reduce the processing time around 22 percent in average in comparison with the PANFIS algorithm without reducing performance in accuracy.

Keywords: Big data stream analytic, Distributed evolving algorithm, Scalable real-time data mining, Parallel learning, Rule merging strategy

1. Introduction

The rapid growth of data generated through the Internet, which causes big data, attracts great attention from many stakeholders. This phenomenon takes place in many areas in the real life such as business, management, medical, government, and society administration. Big data is unique of its 4Vs characteristics: volume, velocity, variety, and veracity. Volume related to the number of data generated in the storage, which is associated with the scale of data. Velocity indicates the flow rate of continuous data which is associated with the data streams. Variety of data is char-

* Choiru Za'in
E-mail address: C.Zain@latrobe.edu.au

1877-0509
10.1016/j.procs.2018.10.514

acterized by the number of different format of the data, whereas veracity shows the uncertainty of data where the data sources need to be validated (trustees, accuracy, and data quality).

Big data provides enormous opportunities for government/organizations in discovering and extracting valuable information/knowledge of their system which can be beneficial for them in the decision-making processes. In the business area for example, Wal-Mart collaborated with Hewlett Packard trace every purchase record belonging to their customers from their point-of-sales terminals, where their transactions reach around 267 million per day. This valuable transaction data becomes a key basis for company to improve their benefits by applying pricing strategy and advertising campaigns [10, 6]. In this case, the decision could be made by applying some techniques such as data mining, which is extensively used for many decision-making problems in many real-life applications. However, discovering meaningful insight of big data is challenging due to its 4V's characteristics, which lie on the difficulties in data capture, data storage, data analysis and data visualization [37, 6]. Therefore, an advanced data mining techniques and technologies are highly necessary to process and analyze big data.

Big data is often stored in cloud to support the extensibility and scalability of local storage refers to one characteristic of big data, namely volume. In order to extract valuable information of big data efficiently, there is an urgent demand to modify existing data mining techniques to be scalable in processing large-scale dataset. This issue led to a necessity to the development distributed or parallelize scenario in processing big data. In addition, big data are also generated by the arrival of new instances continuously in either by batches or one by one, known as data stream, emerging from real-world applications[9, 33]. Therefore, it is necessary for machine learning algorithm to adapt to rapidly changing non-stationary data streams. Note that stream processing/mining in the web news domain has been conducted in [39] using eT2Class [31], which is able to handle streaming data[4]. This phenomenon triggers the development of evolving learning algorithms, which are able to learn big data continuously [13] by evolving its model to adjust the shift and drift of the big data pattern.

There are some frameworks have been conducted in big data analytic, exploiting the characteristics of big data, including large volumes, velocity, and variety of data. Initially, some frameworks have been developed to cope with the volume of big data by increasing the scalability of machine learning algorithms as discussed in [36, 34, 26, 7]. Lately, much attentions have been paid to the development of big data analytic by considering the velocity characteristic in addition to volume due to streaming nature of data. For example, big data stream clustering [16, 1] processed big data by distributing the data stream to be processed in many processors/nodes to speed up the computational time. Despite the increase of computational speed, these works still suffer from coping with the data pattern changing, which in turn, possibly lead to fusion of non-related clusters [14]. It means that the final model generated does not represent the knowledge of the data, which can lead to the misleading inference in prediction and clustering result.

The development of algorithms and techniques which are scalable for handling large-scale data stream and, in the same time, able to cope with the changing environment is crucial. This procedure enable data stream to be processed in parallel mode by distributing them into many processors, where each processor processes/learnes a block of data stream (data chunk). In addition, it also processes every data chunk in online mode with the capability to change its data pattern incrementally.

To the best of our knowledge, a few works have addressed an online big data processing and analytic in both distributed and online mode. Parallel_TEDA was introduced in [15], which is based on parallel mode, which processes every data chunk in single pass mode using evolving algorithm. Its evolving characteristic is inherited from an evolving algorithm, namely Typicality based Empirical Data Analysis (TEDA) [2], which supports the recursive form of calculation and is parameter-free. Eventhough, Parallel_TEDA has applied a parallel strategy and evolving learning algorithm to improve the scalability and speed of learning process, its model is still generated using a simple fusion technique, which can cause a redundancy problem. In addition, it only applies on a simple parallel technique.

In this paper, we propose a novel big data stream analytic based on a newly developed evolving algorithm namely PANFIS [28] by incorporating a well-known MapReduce framework with it. PANFIS is a seminal evolving neuro fuzzy systems (ENFS), which inherits all benefits of evolving learning. It starts learning from the first data with an empty rule base. During its learning, the generated rules adapt incoming data in the single pass mode. The main characteristic of PANFIS is its capability in generating ellipsoids in arbitrary direction, respecting local correlation between variables. In addition, a new fusion technique called rule-merging strategy is also proposed to avoid redundancy and increase the diversity of the model. Futhermore, MapReduce is introduced to improve the scalability of evolving learning algorithm since it is known as one of the most efficient platform for big data analytic due to its

simplicity, generality, and maturity with regard to the previous works in [5, 27, 35]. Our online distributed big data stream analytic framework has the following characteristics:

1. It processes the highly rate of big data stream with a scalable distributed online learning.
2. It updates the rules' structure by merging all the rules and parameters to cope with the changing environment.

The rest of the paper is organized as follows. Section 2 discusses the related researches: MapReduce framework and PANFIS algorithm. Section 3 describes our proposed approach which specifically explains the data flow in processing big data including the rule merging policy. Section 4 discusses experimental setup and results in evaluating big data analytic and section 5 will conclude the paper.

2. PRELIMINARIES

This chapter discusses the underlying techniques of analyzing big data using one of the most widely used parallel learning called MapReduce with PANFIS evolving learning algorithm.

2.1. MapReduce Framework

The MapReduce framework was introduced in 2008 [11], which aims to perform highly computational real-world application tasks by dividing large dataset into many chunks. MapReduce has been applied to various domains such as data mining and machine learning in both multiple cores within a single CPU or multiple CPU. Mapreduce consists of two main phases: mapping and reducing. Mapping phase performs precursory calculation, which in data mining is known as a training of data chunks, whereas reducing phase aggregates the result of training phase.

Under MapReduce framework, data mining procedure is initiated by a procedure in the mapping phase which call data chunks randomly and learn each chunk in a node/processor, after an initial variable pair of *key-value mapping* (K_1, V_1) is assigned to each data chunk. After the learning process, the results are stored in the new variable pair as an object in the intermediate phase, namely *KeyValueStore* (K_1, V_1'). While the keys K_1 in *KeyValueStore* are identical with the keys K_1 in *key-value mapping*, the values V_1' in *KeyValueStore* are the learning outcome from the values V_1 in *key-value mapping*.

In general, the simple MapReduce framework is defined as follows:

$$Mapping : (K_1, V_1) \rightarrow (K_1, V_1') \tag{1}$$

$$Intermediate(Storing\ and\ Grouping) : (K_1, V_1') \rightarrow (K_2, V_2) \tag{2}$$

$$Reducing : (K_2, V_2) \rightarrow (K_3, V_3) \tag{3}$$

Intermediate phase is a bridge between mapping phase and reducing phase. The object in the intermediate phase namely *KeyValueStore* records all information of the learning result V_1' such as models and other parameters assosiated with the data chunk being learnt in the node. In the next step, the learning results V_1' are grouped based on the unique key becomes the object called *key-value intermediate* (K_2, V_2). This object is then used in the reducing phase by aggregating or reducing the values in V_2 becomes V_3. In this case the object *final-key-value* is generated (K_3, V_3). *Final key-value* comprises keys K_3 and the final value V_3. The learning results *final key-value* from the reducing phase represents the number of rules generated from all data chunks. The rule merging procedure described in subsection 3.1 is then taken place to process *final key-value* becomes *final rules*. The complete architecture of MapReduce framework comprising mapping phase and reducing phase is explained in the section 3.

2.2. PANFIS

PANFIS is a seminal evolving neuro fuzzy systems (ENFS), which extends a hybrid human-like reasoning of fuzzy system and neural network architecture that is known as neuro fuzzy system (NFS) [21] in evolving manner. ENFS

is built based on the concept of incremental learning which updates the model components (rules) to handle concept drift. As other evolving learning algorithms characteristic, ENFS have capabilities to learn data from scratch with no initial structure. It reorganizes its rules to presence of the untouched data region by adding new rules, and remove or merge its rules on demand in the single pass learning mode. In addition, an evolving learning algorithm can typically work with minimum or mostly without manual operator supervision, which is desirable in an online real-time learning environment.

In PANFIS, fuzzy rules are extracted and removed based on a new incoming training pattern and fuzzy rule contribution to the system output. The correlation between variable in PANFIS is represented by ellipsoidal in arbitrary position, which connected with a new projection concept to form the antecedent parts in terms of linguistic term (fuzzy sets). This is the extension of work in [20] which draws all operations in multidimensional level nonaxis parallel, which is not interpretable for an expert/user. However, PANFIS still use high dimensional representation for the inference scheme.

In PANFIS, both rule growing and pruning methodology are inspired by SAFIS [32]. For pruning method, an extended rule significance (ERS) concept is utilized to deal with multivariate Gaussian rule. This is carried out by integrating hyperplanes consequents and generalizing to ellipsoids in arbitrary position, which allow to prune the rules in the high-dimensional learning space.

The rules evolution is based on datum significance (DS) criterion, which represents potential of data point being learned. Initialization of new rules is carried out by considering ε-completeness of the rule base and fuzzy partition to ensure an adequate input space coverage. Original PANFIS utilizes ESOM to update the rule premise of PANFIS. The underlying drawback of this method is the need of reinversion which causes instability if the inverse covariance matrix is ill-conditioned (e.g., due to redundant input features). Hence, we adopt the rule adaptation formula of GENEFIS [30], pClass [29] and GEN-SMART-EFS [25]. The enhanced recursive least square (ERLS), an extension of conventional recursive least square (RLS), is utilized to adjust the fuzzy consequences. ERLS has been widely used in the ENFS community and proven to underpin the convergences of the system error and so the weight vector being updated.

The rule base management of PANFIS comprises some learning modules: 1) Rule growing which utilizes the DS concept with ε-completeness criterion to determine the antecedent of new fuzzy rule. Rule premise adaptation is also discussed in this subsection; 2) Rule pruning and merging methodology which allow rules to be pruned and coalesced

2.2.1. *Fuzzy rules growing using DS and fuzzy rule premise adaptation*

The rules growing is triggered by two conditions: 1) datum evokes a high system error and 2) the model traverses outside the existing fuzzy rules' region in the input space. This new growing rule becomes the rule base for the next datum which falls close to it. In PANFIS, the DS concept is formed based on statistical contribution which represents the potential of injected datum when the number of observations approaches to infinity. The approximation of the DS is carried out for both current condition until the end of the training process.

DS criterion algorithm is initialized by the work of [17] and [32] as high-potential datum to cope with the data streams. However, in order to adapt it to PANFIS algorithm which explores hyperplane consequents and rules in arbitrary position in the learning space, some adjustment need to be done. This is due to its nature of DS criterion which is composed in the learning platform of hyperspherical rules and constant output parameters. The mathematical definition of the DS in adjustment to PANFIS is initiated as follows:

$$D_n = |e_n| \int_x \exp\left(\frac{(X - X_n)\Sigma_i^{-1}(X - X_n)}{(X - C_i)\Sigma_i^{-1}(X - C_i)}\right)\frac{1}{S(X)} \tag{4}$$

where X_n represents the latest datum in the u dimension input space $X_n \epsilon \Re^u$, $S(X)$ represents the range of input X. The system error is denoted as e_n which represents the degree of accuracy of PANFIS when the nth training observation is carried out. The system error e_n can be defined as follows:

$$|e_n| = |t_n - y_n| \tag{5}$$

where y_n and t_n are the PANFIS' output and target value of the nth training observation respectively. By performing u-fold numerical integration, we may obtain as follows:

$$D_n = |e_n| \frac{\det(\Sigma_{win})^u}{S(X)} \tag{6}$$

The new datum's significance represents its statistical contribution to PANFIS's output. Therefore, the sampling data distribution $S(X)$ can be further modified as the total volumes of existing fuzzy rules, which can be written as follows:

$$D_n = |e_n| \frac{\det(\sum_{win})^u}{(\sum_{i=1}^{r+1} \det(\sum_i)^u)} \tag{7}$$

The new datum can be recruited as a new rule if such condition is satisfied: $g < D_n$, where g is a constant specified by the user. This condition represents that the datum significance of the nth training observation D_n is measured to have a high descriptive power and generalization potential. Vice versa, if $g \geq D_n$ condition is fulfilled, the existing fuzzy rules are sufficient in covering available training stimuli. In this case, the winning rule's premise will adapt to the current datum's properties.

The formula of rule premise adaption of the winning fuzzy rule in respect to a current datum is defined as follows:

$$C_{win}^{new} = \frac{N_{win}^{old}}{N_{win}^{old} + 1} C_{win}^{old} + \frac{X - C_{win}^{old}}{N_{win}^{old} + 1} \tag{8}$$

$$\begin{aligned} \Sigma_{win}(new)^{-1} &= \tfrac{\Sigma_{win}(old)^{-1}}{1-\alpha} + \tfrac{\alpha}{1-\alpha} \\ \left(\Sigma_{win}\,(old)^{-1}\left(X - C_{win}^{new}\right)\right)&\left(\Sigma_{win}\,(old)^{-1}\left(X - C_{win}^{new}\right)\right)^T \end{aligned} \tag{9}$$

$$N_{win}^{new} = N_{win}^{old} + 1 \tag{10}$$

where $\alpha = 1/\left(N_{win}^{old} + 1\right)$, X represents the focalpoint of the current datum, and N_{win}^{old} represents the number of training samples lie on the winning cluster. This scenario features direct update of the inverse covariance matrix improving robustness and computation speed of adaptation process.

A) ε-completeness criterion definition for new fuzzy rule's initialization

The automation of fuzzy rule generation also follows the ε-completeness criterion in selecting the new antecedent of fuzzy set in order to achieve an adequate coverage to accommodate the universe discourse. The ε-completeness criterion is proposed by [19], where ε is set to 0.5 as defined in [8]. Therefore, for any input injected at least there exists one fuzzy rule with match degree greater than ε. For the new recruited datum, PANFIS will set this datum as a new focal point of the new ellipsoidal rule. The new ellipsoidal region is formed by defining the width of the Gaussian function in order to assure ε-completeness, thus:

$$C_{i+1} = X^n \tag{11}$$

$$diag(\sum_{i+1}) = \left(\frac{\max(|C_i - C_{i-1}|, |C_i - C_{i+1}|)}{\sqrt{\ln(\frac{1}{\varepsilon})}} \right) \tag{12}$$

2.2.2. Fuzzy rules pruning and merging of inconsequential rules and similar fuzzy sets

This subsection exploring the rule pruning and merging of PANFIS. These methods aim to form the effective (economical and interpretable) rule base, and also to achieve a good predictive quality.

A) *ERS concept to prune inconsequential fuzzy rules*

Rule base simplification of PANFIS is inherited from generalized growing and pruning radial basis function (GGAP-RBF) [17] and SAFIS [32]. However, this method is not suitable for PANFIS as it is only suitable for zero-order TSK fuzzy system and unidimensional membership function environment. The contribution of ith rule to the

overall systems output for an input vector X_n has been modified in order to suit PANFIS learning scenario. This contribution is defined alike to the method to determine online input sensitivity analysis and an input selection used in [18] as follows:

$$E(i,n) = |\delta_i| \frac{R_i(x_n)}{\sum_{i=1}^{r} R_i(x_n)} = |\delta_i| E_i \tag{13}$$

where $\delta_i = \sum_{k=1}^{u+1} a_0 + \omega_{1i} + \omega_{2i} x_i + ... + \omega_{ki} x_k + ... + \omega_{u+1,i} x_u$ represents the ith fuzzy rule's output contribution. The fuzzy rule's input contribution is represented by E_i. Moreover, it is notable that the rule significance can be seen as the statistical contribution when the number of observation approaches infinity. To sum up, using simplification process and u-fold numerical integration as explained in [28], the input contribution E_i can be derived as follows:

$$E_i \approx \frac{\pi^{u/2} \det(\sum^i)^u}{S(X)} \tag{14}$$

In this situation, the size of range X ($S(X)$) is insignificant and time consuming to be computed. Therefore, for the sake of flexibility, $S(X)$ can be replaced by the overall fuzzy rule's contribution, which is determined by the overall cluster volumes. In this case, the statistical contribution estimation can be computed as the volume of single fuzzy rule over the total volume of overall fuzzy rules, which is expressed as follows:

$$E_{inf}(i) = \frac{\det(\sum_i)^u}{\sum_{i=1}^{r} \det(\sum_i)^u} \tag{15}$$

The rule pruning is carried out upon the threshold value of k_{err} in order to reduce rule base complexity. The fuzzy rules are considered to be pruned when its contribution vector contains the fuzzy rule contribution below or equal the k_{err} threshold.

B) *Merging of similar fuzzy sets*

Some fuzzy sets could possibly be overlapped if their membership functions are very similar. In this case, the fuzzy sets can be merged. Furthermore, the multivariate Gaussian rule presents less transparent rule semantic because the atomic clauses vanishes from the rule representation. The merging procedure is carried out based on similarity calculation between fuzzy rules membership function. By incorporating a kernel-based metric method in comparing the widths and centers of two fuzzy sets [24], the similarity of two Gaussian membership functions A and B can be expressed as follows:

$$S_{ker}(A,B) = e^{-|C_A - C_B| - |\sigma_A - \sigma_B|} \tag{16}$$

The equation above has the following properties:

$$\begin{aligned} S_{ker}(A,B) = 1 &\Leftrightarrow |C_A - C_B| - |\sigma_A - \sigma_B| = 0 \\ &\Leftrightarrow C_A = C_B \wedge \sigma_A = \sigma_B \end{aligned} \tag{17}$$

$$S_{ker}(A,B) < \varepsilon \Leftrightarrow |C_A - C_B| > \delta \vee |\sigma_A - \sigma_B| > \delta \tag{18}$$

The Eq. (17) shows that two fuzzy sets are identical, which means that the degree of similarity value is 1. On the other hand, Eq. (18) indicates that the embedded set is not identically match, having different width and focal point. The condition of both fuzzy sets can be merged, that is, if the similarity degree above the threshold value of $S_{ker} \geq 0.8$ based on [24]. The final merging formula is defined as follow:

$$c_{new} = (max(U) + min(U))/2 \tag{19}$$

$$\sigma_{new} = (max(U) - min(U))/2 \tag{20}$$

where $U = \{c_A \pm \sigma_A, c_B \pm \sigma_B\}$, the underlying construct is to reduce the approximate merging of two Gaussian kernels. In order to do that the $\alpha - cut$ variable is applied to the exact merging, which value is α, where $\alpha = e^{-1/2} \approx 0.6$. The value of α denotes the membership degree of the inflection points $c \pm \sigma$ of Gaussian kernel. Note that this scenario is carried out at the last stage when presenting the fuzzy rule to the operators. Moreover, the projection strategy to craft the fuzzy set representation from the high-dimensional cluster as presented in PANFIS [28] has to be carried out first before carrying out this procedure. The second method of [28] is used because it offers a fast projection process well-suited for big data analytics.

3. The Proposed Approach

This section will explain the details of our proposed approach especially the flow of data streams from the beginning to the end of MapReduce. It also explains merging process which utilizes rule-merging strategy to reduce rules redundancy which is discussed in the subsection 3.1.

Suppose that the data stream described in the feature space R^2 is denoted as $\{x_1, x_2, x_3, ..., x_k, ...\}$, where k denotes the number of instance at the current state. The number of processors (nodes) is denoted as i, where $i = 1, .., P$. A chunk of data sample entered at the ith node is denoted as $\{x_1^i, x_2^i, \ldots, x_{K-1}^i, x_K^i\}$, where K denotes the size of chunk. The number of key is denoted by l where $l = \{1, 2, ..., M\}$. In this experiment, the number of nodes i, the number of key l, and the size of data chunk K is set by user. The number of data stream k and the number of data chunk N are infinity as data arrives continuosly. As the data arrives contiously, the number of chunk is also continue denoted by o where $o = \{1, 2, ...N..\}$.

This approach processes data chunks in several phases: 1) Mapping phase; 2) Storing phase; 3) Grouping phase; 4) Reducing phase; 5) Merging phase. From five phases mentioned, the first four phases are conducted inside MapReduce framework, whereas the last phase is carried out using rule merging scenario as described in more details in subsection 3.1.

The first three phases of this approach is depicted in Fig. 1. The first phase, mapping phase, is a learning phase which assigns each chunk with its key and value denoted as *key-value mapping* denoted as $< K_1, V_1 >$, where $K_1 = \{1, .., M\}$. In this phase, some data chunks are processed in some processors randomly. Each processor trains data chunk V_1 as described in Fig. 2 using PANFIS evolving algorithm [28] and the results are stored as V_1'. V_1' contains the models (rules) which are stored in the object called *KeyValueStore*. The models V_1' are collected along with previous key K_1 as *KeyValueStore* $< K_1, V_1' >$. The grouping phase will group *KeyValueStore* $< K_1, V_1' >$ by key in the second block of intermediate phase becomes *key-value intermediate*$< K_2, V_2 >$. Note that storing phase and grouping phase are part of intermediate phase. Group of *key-value intermediate* are then passed into reducing phase. In this work, reducing phase will collect only the main properties of learning result to be processed in the rule merging phase outside the MapReduce. The illustration of MapReduce process in reducing phase onward is illustrated in Fig. 3. The collection of *key-value* pairs in this phase is denoted as *key-value final* $< K_3, V_3 >$. Up to this phase, MapReduce has successfully generated pairs of *key-value final* $< K_3, V_3 >$ which contain a group of models (rules) generated from learning of data chunks.

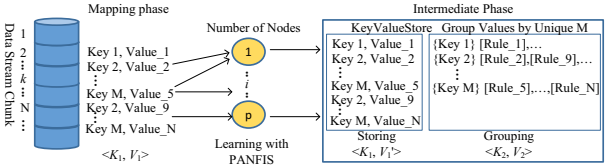

Fig. 1. Mapping phase and intermediate phase

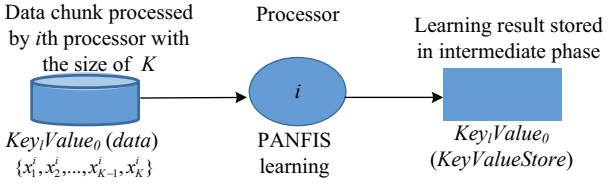

Fig. 2. A data chunk V_1 is learnt in the processor and the value $V_1'$is stored in the storing block of intermediate phase

After reducing phase, the last phase which is essential in this this framework is merging phase, which merges V_3 (grouped and reduced rules from all data chunks), to from the *key-value final* $< K_3, V_3 >$. There are some steps needed to be undertaken to merge the values of V_3 to obtain the final rules. The first step is selecting the highest statistical contribution among rules as the *winning* cluster. The *winning* cluster is determined by comparing all rules statistical contribution using Eq. (15). The second step is comparing *winning* rule with other rules using three criterions which

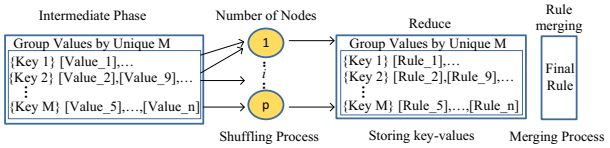

Fig. 3. Reducing phase

need to be satisfied to execute the rule merging process. Thus, the second step comprises 3 substeps: 1) comparing winning rule with other rules based on the criterion of Eq. (21); 2) ensuring that two rules merged form a homogeneous shape and direction using Eq. (23); 3) Ensuring that the merged rule should be smaller or equal of the number of input dimension p times the total volume of both rules using Eq. (24). These three criterions are summarized in Eq. (25)). The third step of rule merging process is applying rule merging policy which is described in the subsection 3.1.1.

3.1. Rule Merging of Final Value generated in MapReduce Process

In the world of evolving fuzzy system, rules which represents the cluster are evolving based on the data pattern. At some points, some rules may become overlapping which reflects the redundancy between the rules [22]. Such rules could always be coalesced in the extreme cases as the overlapping rules represents the similar local region as explained in subsection 2.2.2. In this case, merge of overlapping rules is essential to avoid ambiguous rules and to assure the readability and transparency [23]. We refer to the work explained in [25] to extend the work demonstrated in the PANFIS rule merging. Therefore, there are three criterions used implemented due this reason. The first criterion aims to inspect the rules to be merged if the rules are lying nearby, touching or even slightly overlapping. In order to maintain minimal complexity to resolve non-linearities in the regression problem depicted by multivariate Gaussian rules of PANFIS, a Bhattacharyya distance [3][12] is employed to calculate the degree of overlapping between clusters in MapReduce *final value*, which represents the result of chunks learning. The overlap degree between target cluster (rule) expressed by *win* and the other clusters, $k = \{1, ..., C\}\backslash\{win\}$ is expressed as follow :

$$olap(win, k) = \frac{1}{8}(c_{win} - c_k)^T \sum^{-1}(c_{win} - c_k) + \frac{1}{2}\ln\left(\frac{\det \sum^{-1}}{\sqrt{\det \sum_{win}^{-1} \det \sum_k^{-1}}}\right) \tag{21}$$

where $\Sigma^{-1} = (\Sigma_{win}^{-1} + \Sigma_k^{-1})/2$. In this work, the target cluster *win* can be determined by choosing the cluster which has the highest statistical contribution over other clusters using Eq. (15). The distance of two ellipsoidal cluster is equal to 0 if the two ellipsoids are touching, >0 for overlapping situation and <0 for disjoint situation. Thus, a threshold feasible for cluster merging is 0 with at least the clusters are toching each other in respect to the conditions mentioned.

The second similarity criterion of clusters can be merged based on the degree of deviation in the hyper-planes's gradient information. This criterion is based on the dihedral angle of the two hyper-planes they span [25]. This criterion is expressed as follows :

$$\phi = arccos\left(\frac{a^T b}{|a||b|}\right) \tag{22}$$

where $a = (w_{win;1}w_{win;2}w_{win;3} - 1)^T$ and $b = (w_{k;1}w_{k;2}w_{k;3} + 1)^T$ the normal vectors of the two planes corresponding to rules *win* and k, showing into the opposite direction with respect to target y (-1 and +1 in the last coordinate). The two planes are regarded to be in the same trend of approximation curve if the output ϕ is close to (π). The similarity criterion of two hyperplanes is defined as:

$$S_{cons}(w_{win}, w_k) = \frac{\phi}{\pi} \tag{23}$$

The last similarity criterion for clusters to be merged is based homegeneity of adjacent rules. This criterion aims to ensure that both merged rules should form a homogeneous shape and direction, which reflects the accurate representation of the two local data clouds. In order to do this, the blow-up effect is applied to trigger homogenous joint regions. This condition is expressed as follows:

$$V_{merged} \leqslant p(V_{win} + V_k) \tag{24}$$

where p is the dimension of input attribute. V_{merged}, V_{win}, and V_k represent the volume of merged rules, winning rule, and compared rule respectively. The volume of rule can be determined using Eq. (15). Finally, the merging condition can be carried out by below conditions:

$$(Eq.(21) \geq thr) | (Eq.(21) \geq 0 \wedge Eq.(24) \wedge Eq.(23) \geq 0.5) \tag{25}$$

where thr is set to 0.8 based on empirical experience in [24].

3.1.1. Rule merging policy in the merging phase

In the merging phase, after the winning rule is selected and three criterions are satisfied, rule merging policy is carried out based on the work in [25]. The merging policy is summarized as follows:

$$\begin{gathered} c_j^{new} = \frac{c_j^{win} k_{win} + c_j^k k_k}{k_{win} + k_k} \\ k_{new} - k_{win} + k_k \\ \sum_{md}^{-1} = \frac{k_{win} \sum_{win}^{-1} + \frac{k_{win} k_k}{k_{win} + k_k} diag(\frac{1}{((c^{new} - c^i)(c^{new} - c^i))}) * I}{k_{win} + k_k} \end{gathered} \tag{26}$$

with $(c^{new} - c^i)$ is a row vector with $i = \arg\min(k_{win}, k_k)$. The merging of the rule consequent functions, which follows the idea of Yager's participatory learning concept [38] and also applied by Lughofer [24]:

$$w_{new} = w_j + \alpha \cdot Cons(j,k) \cdot (w_k - w_j) \tag{27}$$

where the basic learning rate is defined by $\alpha = k_k/(k_j + k_k)$ and the compatibility measure of two rules $Cons(j,k)$ within the participatory learning context. Here j represents the index of the more supported rule, i.e. $k_j > k_k$.

4. Experimental setup and results

This section describes the environmental setup in which the PANFIS MapReduce framework has been executed. In order to compare the framework performance, PANFIS MapReduce will be compared with other solely online algorithm algorithms in processing big data. The datasets used in this experiment are SUSY, HIGGS, HEPMASS data which has 5, 11.5, and 11 millions row of table respectively. The description of all datasets are explained in the Table 5. All of the datasets: SUSY, HEPMASS, and HIGGS are generated from Monte Carlo simulations and are used to predict 2 classes label. SUSY comprises of 8 low-level features which describes the kinematic properties and 10 high-level features. In HIGGS dataset, the first 21 features are the high-level features while the last seven features are the functions derived from the first 21 features. HEPMASS dateset comprises the first 27 normalized features (22 low-level features then 5 high-level features) and the last 1 feature represents mass feature.

Table 1. The description of datasets.

Name of Dataset	#features	#row	#training data	#testing data
SUSY	18	5 millions	4.5 millions	0.5 millions
HEPMASS	28	11 millions	7 millions	3 millions
HIGGS	28	11.5 millions	7 millions	3.5 millions

4.1. Experiments

The experimental framework is carried out under the following computer specifications: Intel (R) Core (TM) i7-6700 CPU @3.40 GHz, and Memory 16GB x64-based processor. The software used to run the experiment is MATLAB R2017b 64-bit. The MapReduce environmentment is setup using MapReduce framework available in MATLAB 2017b. In this experiment we conduct 4 algorithms: PANFIS in MapReduce environment, PANFIS, eTS, and Simpl_eTS.

The underlying reason in choosing these algorithms for comparison is as follows:

1. PANFIS is the base evolving algorithm which has capability to learn data in online manner, thus it will provide a benchmark againts the proposed method (MapReduce PANFIS).
2. eTS and Simp_eTS also can be seen as an evolving classifier. Although both of them apply the type-1 classifier architecture, it still deserve to be a benchmarked algorithm as they work in single pass mode learning.

The classification scenarios will measure the running time and the accuracy to evaluate the performance of the algorithm. A summary of our experiment is described in Table 2, Table 3 and Table 4. Each classification task will train and test the number of instance described in Table 5. The classifier performance is measured by calculating the classification rate and running time.

4.2. Results

This subsection details the performance of every classifier in classfying the dataset namely SUSY, HEPMASS, and HIGGS. The classification performance of running time is depicted in the Table 2 and 3. The accuracy performance is depicted in the Table 4. The number of rules generated during the learning process for each iteration on SUSY, HEPMASS, and HIGGS datasets using all methods are shown in Fig. 4, Fig. 5, and Fig. 6 respectively. Furthermore, Table 5 describes the number of rules generated in PANFIS MapReduce before and after rule merging process.

Table 2. The running time performance using full training data.

Method/Dataset	SUSY(second)	HEPMASS(second)	HIGGS(second)
PANFIS MapReduce	7425	19939	16223
PANFIS	10280	23100	22820

Table 3. The running time performance using partial training data

Datasets	#rows	Running time for all methods (second)			
		PANFIS MapReduce	PANFIS	eTS	Simpl_eTS
SUSY	10k	14.43	15,82	74.10	21.05
	20k	30.19	32.02	304.95	104.73
	30k	43.55	47.42	2005.74	762.92
HEPMASS	10k	35.06	35.38	117.12	19.03
	20k	63.96	71.49	461.89	92.29
	30k	96.53	103.43	2931.38	695.43
HIGGS	10k	28.22	34.45	78.81	16.14
	20k	51.25	64.4	324.07	78.11
	30k	72.63	102.63	78.11	656.65

In terms of running time, Table 2 shows that MapReduce environment speeds up the learning performance of PANFIS classifier by around 23, 14, and 29 percent respectively for SUSY, HEPMASS, and HIGGS dataset. This experiment is carried out using full training data: SUSY (4.5 millions), HEPMASS (7 millions), and HIGGS (7 millions). It is shown that MapReduce enviroment can improve the speed around 22 percent. The similar result also can be seen in Table 3, where MapReduce environment also improves the speed around 15 percent for every scenario in partial training data. Note that, the time measurement in PANFIS MapReduce is carried out by accumulating the total time for every processors in processing every data chunk.

For eTS and Simp_eTS, the running time performance can be investigated in Table 4. The reason we separate the running performance of eTS and Simpl_eTS from the result in the Table 2 is due their time consuming in learning the

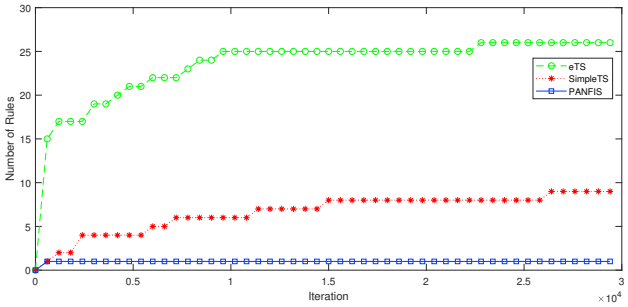

Fig. 4. Number of rules generated in processing SUSY dataset using all methods for each iteration

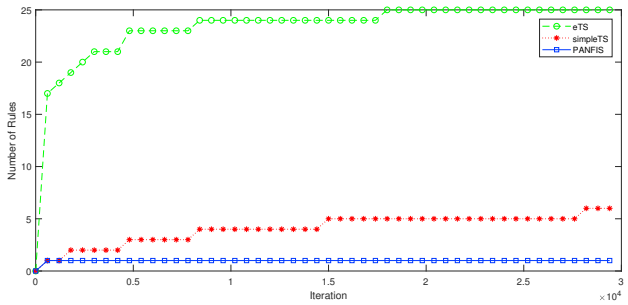

Fig. 5. Number of rules generated in processing HEPMASS dataset using all methods for each iteration

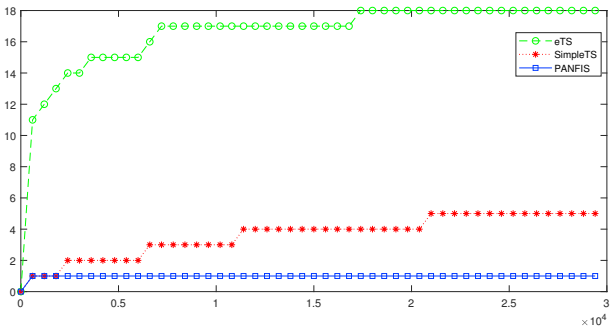

Fig. 6. Number of rules generated in processing HIGGS dataset using all methods for each iteration

large dataset. For example in Table 3, the running time for eTS and Simp_eTS in running the SUSY dataset increases significantly from 74.10 second and 21.05 second to 2005.74 second and 762.92 second for 10 thousands and 30 thousands data. It is shown that the increase number of data is not linear with the time required in processing the data due to the increase of rules generated in their evolving system as shown in Fig. 4, Fig. 5, and Fig. 6. For that reason, we eliminate the experiment of eTS and Simpl_eTS up to 30 thousands of dataset. In contrast, PANFIS algorithm produces running time performance linear with the increase of data. Note that for the number of rules generated in all algorithms, we only compare 3 base algorithms PANFIS, eTS, and Simpl_eTS as PANFIS MapReduce performs its learning in chunk by chunk basis so that the number rules generated is resulted from the accummulation of all learning in every chunks.

In terms of accuracy, PANFIS, eTS, and Simp_eTS provide similar results for the SUSY, HEPMASS, and HIGGS dataset with eTS performs the highest result with 77.05, 82.32, and 64.69 percent. However, PANFIS and Simp_eTS

Table 4. The classification accuracy

Method/DATASET	Accuracy		
	SUSY	HEPMASS	HIGGS
PANFIS MapReduce	76.80	83.35	63.48
PANFIS	75.42	83.32	63.94
eTS	77.05	82.32	64.69
Simpl_eTS	70.93	81.22	60.17

Table 5. The rule evolution of PANFIS MapReduce learning.

#Rules generated	SUSY	HEPMASS	HIGGS
Accumulated rules before merging	179	350	438
After merging	1	1	1

provide comparable result with eTS. From the Table 4, we can see that the effect of MapReduce environment followed by the merging process does not deteriorate the performance of accuracy.

5. Conclusions

Analyzing big data in online environment is challenging due to its characteristic of 4Vs: volume, velocity, variety, and veracity. There are many areas to optimize and improve its speed and scalability of big data analytic. In terms of speed, PANFIS evolving system provides a promising solution for processing big data stream solution due to its capability to process data in real time environment. In addition, PANFIS evolving system has the ability to reduce the system complexity by reducing the superfluous rules and merging similar rules. In this work, the learning performance has improved further by processing them in distributed environment, which increases the scalability of machine learning algorithm. It can be seen in the subsection 4.2, the speed performance increases around 22 percent in average for all datasets, where full training data are utilized without reducing the accuracy performance.

6. Acknowledgement

The third author acknowledges the support of the Austrian COMET-K2 programme of the Linz Center of Mechatronics (LCM), funded by the Austrian federal government and the federal state of Upper Austria. This publication reflects only the authors' view.

References

[1] Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 81–92. VLDB Endowment, 2003.
[2] P Angelov, X Gu, D Kangin, and J Principe. Teda: typicality based empirical data analysis. *Submitted to Information Sciences*, 2016.
[3] Anil Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distribution. *Bull. Calcutta Math. Soc*, 1943.
[4] Albert Bifet and Richard Kirkby. Data stream mining a practical approach. 2011.
[5] Yingyi Bu, Bill Howe, Magdalena Balazinska, and Michael D Ernst. Haloop: Efficient iterative data processing on large clusters. *Proceedings of the VLDB Endowment*, 3(1-2):285–296, 2010.
[6] CL Philip Chen and Chun-Yang Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, 275:314–347, 2014.
[7] Rong Chen, Krishnamoorthy Sivakumar, and Hillol Kargupta. Collective mining of bayesian networks from distributed heterogeneous data. *Knowledge and Information Systems*, 6(2):164–187, 2004.
[8] Stephen L Chiu. Fuzzy model identification based on cluster estimation. *Journal of Intelligent & Fuzzy Systems*, 2(3):267–278, 1994.

[9] Lior Cohen, Gil Avrahami-Bakish, Mark Last, Abraham Kandel, and Oscar Kipersztok. Real-time data mining of non-stationary data streams from sensor networks. *Information Fusion*, 9(3):344–353, 2008.
[10] Jared Dean. *Big data, data mining, and machine learning: value creation for business leaders and practitioners*. John Wiley & Sons, 2014.
[11] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.
[12] Abdelhamid Djouadi, Oe. Snorrason, and FD Garber. The quality of training sample estimates of the bhattacharyya coefficient. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):92–97, 1990.
[13] João Gama. *Knowledge discovery from data streams*. CRC Press, 2010.
[14] João Gama, Pedro Pereira Rodrigues, and Raquel Sebastião. Evaluating algorithms that learn from data streams. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1496–1500. ACM, 2009.
[15] Xiaowei Gu, Plamen P Angelov, German Gutierrez, Jose Antonio Iglesias, and Araceli Sanchis. Parallel computing teda for high frequency streaming data clustering. In *INNS Conference on Big Data*, pages 238–253. Springer, 2016.
[16] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):515–528, 2003.
[17] Guang-Bin Huang, Paramasivan Saratchandran, and Narasimhan Sundararajan. A generalized growing and pruning rbf (ggap-rbf) neural network for function approximation. *IEEE Transactions on Neural Networks*, 16(1):57–67, 2005.
[18] Nikola K Kasabov and Qun Song. Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE Transactions on Fuzzy Systems*, 10(2):144–154, 2002.
[19] Chuen-Chien Lee. Fuzzy logic in control systems: fuzzy logic controller. i. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2):404–418, 1990.
[20] Andre Lemos, Walmir Caminhas, and Fernando Gomide. Multivariable gaussian evolving fuzzy modeling system. *IEEE Transactions on Fuzzy Systems*, 19(1):91–104, 2011.
[21] Chin-Teng Lin et al. *Neural fuzzy systems: a neuro-fuzzy synergism to intelligent systems*. Prentice hall PTR, 1996.
[22] Edwin Lughofer. *Evolving fuzzy systems-methodologies, advanced concepts and applications*, volume 53. Springer, 2011.
[23] Edwin Lughofer. On-line assurance of interpretability criteria in evolving fuzzy systems–achievements, new concepts and open issues. *Information Sciences*, 251:22–46, 2013.
[24] Edwin Lughofer, Jean-Luc Bouchot, and Ammar Shaker. On-line elimination of local redundancies in evolving fuzzy systems. *Evolving Systems*, 2(3):165–187, 2011.
[25] Edwin Lughofer, Carlos Cernuda, Stefan Kindermann, and Mahardhika Pratama. Generalized smart evolving fuzzy systems. *Evolving Systems*, 6(4):269–292, 2015.
[26] Dijun Luo, Chris Ding, and Heng Huang. Parallelization with multiplicative algorithms for big data mining. In *IEEE 12th International Conference on Data Mining (ICDM)*, pages 489–498. IEEE, 2012.
[27] Peter Mika. Flink: Semantic web technology for the extraction and analysis of social networks. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2):211–223, 2005.
[28] Mahardhika Pratama, Sreenatha G Anavatti, Plamen P Angelov, and Edwin Lughofer. Panfis: A novel incremental learning machine. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):55–68, 2014.
[29] Mahardhika Pratama, Sreenatha G Anavatti, Meng Joo, and Edwin David Lughofer. pclass: an effective classifier for streaming examples. *IEEE Transactions on Fuzzy Systems*, 23(2):369–386, 2015.
[30] Mahardhika Pratama, Sreenatha G Anavatti, and Edwin Lughofer. Genefis: toward an effective localist network. *IEEE Transactions on Fuzzy Systems*, 22(3):547–562, 2014.
[31] Mahardhika Pratama, Jie Lu, and Guangquan Zhang. Evolving type-2 fuzzy classifier. *IEEE Transactions on Fuzzy Systems*, 24(3):574–589, 2016.
[32] Hai-Jun Rong, N Sundararajan, Guang-Bin Huang, and P Saratchandran. Sequential adaptive fuzzy inference system (safis) for nonlinear system identification and prediction. *Fuzzy Sets and Systems*, 157(9):1260–1275, 2006.
[33] Moamar Sayed-Mouchaweh and Edwin Lughofer. *Learning in non-stationary environments: methods and applications*. Springer Science & Business Media, 2012.
[34] John Shafer, Rakesh Agrawal, and Manish Mehta. Sprint: A scalable parallel classi er for data mining. In *Proceedings of the 22nd International Conference on Very Large Data Bases*, pages 544–555, 1996.
[35] Apache Mahout Team. Apache mahout: Scalable machine-learning and data-mining library, 2011.
[36] Daniel Warneke and Odej Kao. Nephele: efficient parallel data processing in the cloud. In *Proceedings of the 2nd workshop on many-task computing on grids and supercomputers*, page 8. ACM, 2009.
[37] Xindong Wu, Xingquan Zhu, Gong-Qing Wu, and Wei Ding. Data mining with big data. *IEEE transactions on knowledge and data engineering*, 26(1):97–107, 2014.
[38] Ronald R Yager. A model of participatory learning. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(5):1229–1234, 1990.
[39] Choiru Za'in, Mahardhika Pratama, Edwin Lughofer, and Sreenatha G Anavatti. Evolving type-2 web news mining. *Applied Soft Computing*, 54:200–220, 2017.