RHDOFS: A Distributed Online Algorithm Towards Scalable Streaming Feature Selection

Chuan Luo, Sizhao Wang, Tianrui Li, *Senior Member, IEEE*, Hongmei Chen, Jiancheng Lv, *Senior Member, IEEE*, Zhang Yi, *Fellow, IEEE*

Abstract—Feature selection is an important topic in data mining and machine learning, which aims to select an optimal feature subset for building effective and explainable prediction models. This paper introduces Rough Hypercuboid based Distributed Online Feature Selection (RHDOFS) method to tackle two critical challenges of Volume and Velocity associated with big data. By exploring the class separability in the boundary region of rough hypercuboid approach, a novel integrated feature evaluation criterion is proposed by examining not only the explicit patterns contained in the positive region but also the useful implicit patterns derived from the boundary region. An efficient online feature selection method for streaming feature scenario is developed to identify relevant and nonredundant features in an incremental iterative fashion. Furthermore, a parallel optimization mechanism by combining both data and computational independence is further employed to accelerate the original sequential implementation. An efficient distributed online feature selection algorithm is presented and implemented on the Apache Spark platform to scale for massive amount of data by exploiting the computational capabilities of multicore clusters. Encouraging results of extensive experiments indicate the superiority and notable advantages of the proposed algorithm over the relevant and representative online feature selection algorithms. Empirical tests on scalability and extensibility also demonstrate our distributed implementation significantly reduces the computational times requirements while maintaining the prediction accuracy, and is capable of scaling well in volume of data and number of computing nodes.

Index Terms—Apache spark, parallel computing, scalability, feature selection, rough hypercuboid, online learning.

1 Introduction

Feature selection, also known as variable selection and attribute selection in data mining and machine learning society, refers to select a minimal subset of features that can retain the discriminating ability of features maximally while maintaining the physical meanings of the whole features for efficient model construction [1]. By removing irrelevant and redundant features and selecting relevant and informative ones, feature selection is capable of improving the generalization ability and interpretability as well as the compactness of the learned models [2]. Up to now, various types of feature selection methods have been proposed and found applications in many domains such as bioinformatics, text mining, genetic and microarray analysis, etc. These methods can be broadly classified into three categories from the perspective of selection strategy as wrapper, filter and embedded methods [3]. The wrapper methods employ a predetermined learning algorithm to evaluate all possible feature subsets in the whole feature space, the embedded methods embed the feature selection process into the model training, and the filter methods evaluate the feature subsets based on the quantitative criteria without involving any specific classifiers.

In many real-time data-intensive applications, data analysis tasks are typically operated in a dynamic environment whose data are being continuously evolved through the periodical or occasional updating [4]. Usually, acquiring all helpful training data for model learning is too difficult and expensive. For example, the conduction of labour-intensive wet laboratory experiments in bioinformatics is time consuming and expensive [5]. In such situation, it is impractical to activate the feature selection process stalled until the generation and preparation of the full set of candidate features [6]. Despite the extensive investigation of the feature selection, most existing methods are performed in an offline or batch manner which are limited to the problem setting that all the candidate features are given a priori. Different from standard feature selection algorithm, streaming feature selection assumes not all the features are present and the features are dynamically flowed into the predictive model one by one, and the selection of representative features is performed by the arrival of features. Due to the major characteristic of velocity in the context of big data, a surge of research interests on streaming feature selection have been attracted from industry and academia recently. Some representative works include Grafting [7] Alpha-investing [8], OSFS [9], Fast-OSFS [10], SAOLA [11], OFS-A3M [12] and OFS-Gapknn [13]. Although the existing online feature selection approaches address the issue of streaming features, the main drawback of these algorithms is that they only consider the explicit patterns of classification while ignoring the use of implicit patterns to measure the discriminating ability of a feature or a subset of features. The ignorance of implicit patterns in these methods may cause the absence of useful information which may lead to degradation in the prediction performance. Furthermore, these methods are performed in a sequential computing manner, none of exist-

- *C. Luo, S.Z. Wang, J.C. Lv, and Z. Yi are with the College of Computer Science, Sichuan University, Chengdu 610065, China (e-mail: cluo@scu.edu.cn; wangsizhao@stu.scu.edu.cn; {lvjiancheng, zhangyi}@scu.edu.cn).*
- *T.R. Li and H.M. Chen are with School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu 611756, China (e-mail: {trli,hmchen}@swjtu.edu.cn).*
(Corresponding author: Chuan Luo.)

ing streaming feature selection methods can well exploit the parallel computational capabilities of modern distributed systems. Hence, those methods select features are extremely time-consuming and become inapplicable when processing a large amount of data. Acceleration of feature selection algorithms using parallel computing technologies has attracted much attention. To alleviate the difficulties of developing distributed programs, many parallel programming frameworks have appeared to facilitate efficient distribution of work over computing clusters, such as Apache Hadoop and Spark. Up to now, many batch feature selection methods are parallelized on cloud-based systems using Hadoop and Spark for big data [14–20]. However, to the best of our knowledge, this is the first effort that considers parallel online approach for streaming feature selection in a distributed fashion, a significantly more challenging problem than the conventional streaming feature selection.

Motivated by these observations, we address the issue of large-scale streaming feature selection and propose a novel distributed online algorithm called RHDOFS based on the rough hypercuboid approach in this paper. The unique contributions that distinguish the proposed work from existing approaches are fourfold: 1) A novel distance metric-assisted feature evaluation criterion in the framework of rough hypercuboid approach is proposed, which not only considering the explicit patterns contained in the positive region but also the useful implicit patterns derived from the boundary region. 2) We develop an effective online feature selection algorithm for streaming feature scenario which incorporates the proposed novel integrated feature evaluation criterion. 3) A scalable optimization framework under data parallelism is presented to enable the proposed online algorithm that can access distributed computing clusters. 4) Parallel online feature selection algorithm is then implemented following several Apache Spark RDD operations that used to ease the code parallelization effectively. A comprehensive view of performance evaluation based on extensive empirical study illustrates that the proposed novel online feature selection algorithms offer superior performance than the state-of-the-art representative online algorithms, and exhibit high scalability and extensibility for processing large-scale datasets.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 describes the theoretical foundations of rough hypercuboid approach. Section 4 presents the novel online feature selection algorithm which incorporates the proposed distance metric-assisted feature evaluation criterion. The parallel implementation of the online feature selection algorithm on Spark is demonstrated in Section 5. Section 6 shows the experimental results and evaluations. We conclude our work in Section 7.

2 Related Work

Our work is closely related to the studies of streaming feature selection and distributed feature selection in literature. Below we review the state-of-the-art representative related works in both areas.

2.1 Online Feature Selection with Streaming Features

Generally, online feature selection assumes that the full feature space is unknown in advance, and the features arrive in by streams. Up to now, several research efforts have been made to address the streaming feature challenge. Perkins and Theiler proposed the Grafting algorithm by employing a stagewise gradient descent strategy to select feature subset in an incremental iterative manner by the arrival of features [7]. Tuning an optimal regularization parameter in Grafting requires the knowledge of the global feature space, which leads to a poor performance in dealing with infinitely streaming features. Zhou et al. proposed information-investing and Alpha-investing methods to dynamically adjust the threshold on the error reduction during the streamwise feature selection process [8]. However, Alpha-investing selects the relevant features without considering the redundancy among them, which will significantly influence the quality of the selected feature subset. By checking the redundancy of the selected feature subset with the arrival of new features, Wu et al. presented a novel online feature selection method (OSFS) to select the strongly relevant and nonredundant feature subset [9]. OSFS offers outstanding prediction performance with fewer selected features while suffering from computational burden. Wu et al. further developed Fast-OSFS method to accelerate online feature selection by determining the redundancy of the newly arrived feature and identifying the redundancy of the already selected features with the inclusion of the new feature [10]. Yu et al. provided a theoretical analysis on bounds of the pairwise correlations between features, and proposed a scalable and accurate online feature selection approach (SAOLA) by performing pairwise comparisons online to efficiently calculate correlations between features [11]. The difficulty in obtaining the optimal relevance threshold is one major drawback of SAOLA. The use of rough set theory for the task of feature selection has proven remarkably popular and has been applied to many real world applications in recent years. Rough set theory provides an information granulation oriented formal methodology for feature selection by employing simple set operations without user specified parameter [21]. It requires no additional domain knowledge other than the data itself, and preserves the underlying semantics of the original feature space. The conventional rough sets can only be applied to the nominal or discrete data, and the optimal discrete partition of numerical feature values is an NP-hard problem [22]. In this context, generalized theories for hybridization of rough computing has emerged as an important topic, several successful models for handling numerical feature selection problem have been developed, such as fuzzy rough sets [23], neighborhood rough sets [24], rough hypercuboid approach [25], etc. To address the challenge of feature selection for online learning, a few of new online feature selection with streaming features have been developed recently. Eskandari and Javidi presented a conventional rough sets based online feature selection algorithm in the framework of QUICKREDUCT [26] by eliminating redundant features in terms of the positive region [27]. Combining dependency, significance criteria with relevance criterion, Zhou et al. proposed an online feature selection method (OFS-A3M) to achieve features with high correlation, high dependency and low redundancy based on neighborhood rough sets with adapted neighbors [12]. Li et al. developed a novel neighborhood rough sets method for online feature selection based on the k-nearest and the Gap

neighborhood (OFS-Gapknn) to perform well on the unevenly distributed object space [13]. The problem of multi-label streaming feature selection has been addressed in [28], a novel neighborhood rough sets based online multi-label feature selection framework was proposed via the online importance selection and online redundancy update. The online feature selection problem from a dynamic decision perspective was investigated in [29], a novel online scalable streaming feature selection framework was proposed based on three-way decisions of features (i.e., selecting, discarding and delaying), so as to minimize the overall decision risks when selecting dynamic features under uncertainty. Zhou et al. proposed an efficient online feature selection algorithm by early terminating the selection of features before the end of feature streaming arrived [30]. They further proposed a generalized assembly streaming feature selection framework including irrelevant feature discarding, relevant feature selecting, and non-significant feature removing [31]. Existing online feature selection approaches can evaluate features dynamically for streaming feature scenario, but they suffer from a common limitation: they only consider the explicit patterns of classification and overlook the implicit patterns to measure the discriminating ability of features. Simply discarding the implicit patterns in feature selection may degrade performance. Unlike the existing studies, we aim to proposed a novel efficient online feature selection algorithm by effectively exploring the useful implicit patterns contained in the boundaries of classification.

talks about implicit patttern and trying to solve it.

2.2 Parallel and Distributed Feature Selection

It was ascertained from the review of previous work that several previous works have addressed the online implementation of different feature selection algorithms. Unfortunately, these existing solutions are sequential computing algorithms which are impractical for processing the large volumes of data due to limited computational and storage resources. Therefore, a scalable adaptation of the online algorithm is required to be able to apply the feature selection algorithm to large-scale streaming data. The adequacy of distributed and parallel computing technique by efficient exploitation of computing clusters to accelerate feature selection are increasingly becoming critical and popular in recent years [32, 33]. Zhang et al. developed a distributed version of positive region based feature selection algorithm by incremental updating and parallel merging the similarity relation matrices [14]. Qian et al. described a parallel hierarchical attribute reduction by improving the parallelism of feature-significance computation via the data and task parallel optimization [15]. Hu et al. proposed a MapReduce based parallel multimodality attribute reduction method in the framework of multikernel fuzzy rough sets [16]. The method assumes that the entire data set is available at each node of a cluster which greatly hinder the time-efficiency and scalability. Chen et al. proposed a parallel algorithm for selecting minimal dominance discernibility feature subset in ordered classification problem with monotonicity constraint [17]. Dagdia et al. presented a scalable positive region based filter feature selection algorithm by splitting the data vertically [18]. Kong et al. used the dynamic data decomposition and summarization strategies to implement a distributed feature selection algorithm based on fuzzy rough sets [19]. Although this algorithm has the advantage of avoiding intermediate data transfer across the nodes, the overheads with respect to computations over overlapping subsets make the algorithm computationally expensive. Dagdia et al. developed a novel distributed feature selection algorithm based on the locality sensitive hashing technique, which maps similar objects into the same bucket in low dimensional cases and partitions the feature search space in a more appropriate way [20]. Luo et al. proposed vertical and horizontal partitioning strategies based parallel feature selection algorithms underpinned by rough hypercuboid approach [34]. A large-scale meta-heuristic feature selection approach was further developed in [35] based on the hybridization of the rough hypercuboid approach and binary particle swarm optimization algorithm. Although existing parallel and distributed methods greatly relieve the burden of processing large-scale feature selection, most of them are conducted in an off-line manner based on a fixed set of candidate features which assume that all features are available in advance. However, since the prior knowledge of global feature space is not always available, existing studies on distributed feature selection could not perform well in the streaming feature scenario. In contrast, we address the online distributed feature selection problem in this work. To enable online approach for streaming feature selection that can access distributed computational clusters, we propose a parallel online feature selection algorithm for tasks with large-scale data that is implemented on the Apache Spark platform.

3 Rough Hypercuboid Approach

Rough hypercuboid based feature selection methods have demonstrated their effectiveness [25, 36]. Let $DT = (U, C \cup D)$ be a decision table, where $U = \{x_1, x_2, ..., x_n\}$ is a set of n objects, $C = \{f_1, f_2, ..., f_m\}$ and $D = \{d\}$ are the non-empty finite sets of conditional features and decision features. The equivalence relation induced by D partitions U into c equivalence classes, which can be denoted as $U/D = \{\beta_1, \beta_2, \cdots, \beta_c\}$. The interval $[L_i, U_i]$ represents the value range of objects that belongs to $\beta_i \in U/D$ under $f_k \in C$. Then, the rough hypercuboid equivalence partition matrix of f_k can be defined as $\mathrm{H}(f_k) = [h_{ij}(f_k)]$, where

$$h_{ij}(f_k) = \begin{cases} 1, & L_i \leq x_j(f_k) \leq U_i; \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

Here, $x_j(f_k)$ is the k-th conditional feature value of object x_j, and $h_{ij}(f_k)$ indicates the membership degree of x_j that can be classified into the class β_i, which satisfies $\forall i$, $1 \leq \sum_{j=1}^{n} h_{ij}(f_k) \leq n$ and $\forall j$, $1 \leq \sum_{i=1}^{c} h_{ij}(f_k) \leq c$. Given two conditional features f_k and f_l, the hypercuboid equivalence partition matrix of $\{f_k, f_l\}$ can be calculated as $\mathrm{H}(\{f_k, f_l\}) = \mathrm{H}(f_k) \cap \mathrm{H}(f_l)$, where $h_{ij}(\{f_k, f_l\}) = h_{ij}(f_k) \cap h_{ij}(f_l)$.

Based on the concept of hypercuboid equivalence partition matrix $\mathrm{H}(f_k)$, the misclassified objects from implicit hypercuboids can be identified by the confusion vector $\mathrm{V}(f_k) = [v_1(f_k), v_2(f_k), ..., v_n(f_k)]$, where

$$v_j(f_k) = \min\left\{1, \sum\nolimits_{i=1}^{c} h_{ij}(f_k) - 1\right\}. \quad (2)$$

In Eq. (2), if an object x_j belongs to exactly one class β_i, then x_j belongs to the lower approximation of β_i and $v_j(f_k) = 0$. Otherwise, x_j is encompassed by the implicit hypercuboids and belongs to the boundary region of multiple classes. Let $B = \{f_k\}$, the lower and upper approximations of β_i using the knowledge contained in B can be defined as follows.

$$\begin{aligned}&\underline{\mathrm{B}}(\beta_i) = \{x_j | h_{ij}(f_k) = 1 \text{ and } v_j(f_k) = 0\} \text{ and}\\ &\overline{\mathrm{B}}(\beta_i) = \{x_j | h_{ij}(f_k) = 1\}.\end{aligned} \tag{3}$$

The boundary region of β_i is thus denoted as:

$$\mathrm{BND}_B(\beta_i) = \{x_j | h_{ij}(f_k) = 1 \text{ and } v_j(f_k) = 1\}. \tag{4}$$

Then, the positive, negative and boundary regions of a decision partition U/D can be defined as follows.

$$\begin{aligned}&\mathrm{POS}_B(D) = \cup_{\beta_i \in U/D} \underline{\mathrm{B}}(\beta_i),\\ &\mathrm{NEG}_B(D) = U - \cup_{\beta_i \in U/D} \overline{\mathrm{B}}(\beta_i),\\ &\mathrm{BND}_B(D) = \cup_{\beta_i \in U/D} \overline{\mathrm{B}}(\beta_i) - \cup_{\beta_i \in U/D} \underline{\mathrm{B}}(\beta_i).\end{aligned} \tag{5}$$

Combining Eqs. (1), (2) and (5), the relevance between a feature $f_k \in C$ and the decision feature set D is given by:

$$\begin{aligned}\gamma_{f_k}(D) &= \frac{1}{n} \sum\nolimits_{i=1}^{c} \sum\nolimits_{j=1}^{n} h_{ij}(f_k) \cap [1 - v_j(f_k)]\\ &= 1 - \frac{1}{n} \sum\nolimits_{j=1}^{n} v_j(f_k),\end{aligned} \tag{6}$$

where $0 \leq \gamma_{f_k}(D) \leq 1$.

Hence, the dependency between a feature subset $B \subseteq C$ and the decision feature set D is given by:

$$\gamma_B(D) = 1 - \frac{1}{n} \sum\nolimits_{j=1}^{n} v_j(B), \tag{7}$$

where $h_{ij}(B) = \bigcap_{f_k \in C} h_{ij}(f_k)$ and $0 \leq \gamma_B(D) \leq 1$.

And, the significance of feature f_k ($f_k \notin B$) with respect to B is thus defined as:

$$\begin{aligned}\sigma_B(D, f_k) &= \gamma_{B \cup \{f_k\}}(D) - \gamma_B(D)\\ &= \frac{1}{n} \sum\nolimits_{j=1}^{n} [v_j(B) - v_j(B \cup \{f_k\})],\end{aligned} \tag{8}$$

where $0 \leq \sigma_B(D, f_k) \leq 1$.

By integrating Eqs. (6), (7) and (8), a heuristic filter algorithm using sequential search approaches can be formalized for feature selection by maximizing feature relevance and minimizing feature redundancy.

4 Online Feature Selection

Rough hypercuboid approach provides three efficient quantitative measures to describe the relevance, dependency and significance of features in approximation spaces. However, only the objects which can be classified with certainly (i.e., the objects belong to the positive region) are considered to define these quantitative measures, while ignoring the additional useful information contained in boundary regions which may contribute to identify more discriminative features. In this section, a novel distance metric assisted quantitative measure to evaluate the discriminating ability of a feature or a subset of features is developed, which not only considering the explicit patterns contained in the positive region but also the useful implicit patterns derived from the boundary region. An online feature selection algorithm based rough hypercuboid approach is further developed by incorporating the proposed integrated evaluation criterion.

4.1 Distance Metric Assisted Rough Hypercuboid Approach

The class separability degree of objects from different classes in boundary regions reflects the distinguishing ability of feature subsets to a certain extent. Considering an example with a conditional feature set $\{f_1, f_2, f_3\}$ and two classes $\{\mathrm{A}, \mathrm{B}\}$ shown in Fig. 1, where the rough hypercuboids of class A and B are denoted by the blue and red rectangles respectively, the intersection of two rough hypercuboids indicates the boundary region in each subgraph, and the center points of class A and B are represented by the blue and red pentagrams, respectively.

We can find that the positive regions in Figs. 1(a) and 1(b) are exactly the same, which implies that $\gamma_{\{f_1,f_2\}}(D) = \gamma_{\{f_1,f_3\}}(D)$ holds according to Eq. (7). Given feature f_1, then features f_2 and f_3 have indistinguishable performance when performing selection in terms of the significance criterion as defined in Eq. (8). However, there actually exist obvious differences between the boundary regions in Figs. 1(a) and 1(b). That is, the objects belong to different classes in the boundary region induced by $\{f_1, f_3\}$ are more separable and easier to be classified explicitly in comparison with that in the boundary region induced by $\{f_1, f_2\}$. Evidently, f_3 should own higher significance than f_2 if f_1 has already been selected. Hence, the class separability can give insights into the boundary region and is conducive to find more relevant and discriminative features.

The class separability of boundary region can be measured by exploring the relationship between the membership and distance of each object with respect to the decision classes under features [37]. Given $U/D = \{\beta_1, \beta_2, \cdots, \beta_c\}$, we use $CP = \{CP_1, CP_2, ..., CP_c\}$ to represent the set of center points of decision classes under the feature subset $B \subseteq C$, where $CP_i = [cp_i(f_1), cp_i(f_2), ..., cp_i(f_{|B|})](i = 1, 2, ..., c, f_k \in B)$, and $cp_i(f_k)$ denotes the average value of feature f_k with respect to the objects that belong to the decision class β_i. Let $\mu_B(x_j, \beta_i)$ be the fuzzy membership of object x_j with respect to the class β_i, abbreviated as μ_{ji}. And the distance of object x_j with respect to class β_i is denoted as $d_B(x_j, \beta_i)$. The relationship between the fuzzy membership $\mu_B(x_j, \beta_i)$ and the distance $d_B(x_j, \beta_i)$ can be obtained as follows.

$$\begin{aligned}\mu_{ji} &= \left(\frac{1}{\sum_{k=1}^{c} \frac{1}{d_B(x_j,\beta_k)^{\frac{2}{m-1}}}}\right)\left(\frac{1}{d_B(x_j,\beta_i)^{\frac{2}{m-1}}}\right)\\ &= \frac{1}{\sum_{k=1}^{c} \frac{d_B(x_j,\beta_i)^{\frac{2}{m-1}}}{d_B(x_j,\beta_k)^{\frac{2}{m-1}}}} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_B(x_j,\beta_i)}{d_B(x_j,\beta_k)}\right)^{\frac{2}{m-1}}},\end{aligned} \tag{9}$$

where m is a weighting exponent and $m > 1$. Eq. (9) shows that μ_{ji} has a negative correlation with $d_B(x_j, \beta_i)$, i.e., a smaller distance results in larger fuzzy membership, which further indicates a higher class separability degree of objects in the boundary region. The distance $d_B(x_j, \beta_i)$ can be calculated as:

$$d_B(x_j, \beta_i) = \sqrt[p]{\sum_{f \in B} |x_j(f) - cp_i(f)|^p}, \tag{10}$$

where $p \geq 1$. Specially, $d_B(x_j, \beta_i)$ is the Manhattan distance if $p = 1$, the Euclidean distance if $p = 2$ and the Chebychev

distance if $p = \infty$. Here, we choose the Euclidean distance and let m equals 2 to improve computational efficiency.

Based on Eq. (9), the class separability degree of boundary region induced by $B \subseteq C$ is defined as follows.

$$\delta_B(D) = \frac{1}{n}\sum_{j=1}^{n} \min\{\mu_B(x_j, \beta), v_j(B)\}, \tag{11}$$

where β is the decision class to which object x_j belongs.

Combining the dependency criterion defined in the positive region and the class separability degree of boundary region, a novel integrated criterion is proposed by not only considering the explicit patterns contained in the positive region but also the implicit separable patterns derived from the boundary region as follows.

$$\begin{aligned}\rho_B(D) &= \gamma_B(D) + \delta_B(D) \\ &= 1 - \frac{1}{n}\sum\nolimits_{j=1}^{n} v_j(B) + \delta_B(D) \\ &= \frac{1}{n}\sum_{\beta_i \in U/D}\sum_{x_j \in \beta_i} \max\{\mu_B(x_j, \beta_i), 1 - v_j(B)\}.\end{aligned} \tag{12}$$

Table 1 provides a toy example to illustrate the difference between the criterion of dependency shown in Eqs. (7) and (12), which contains twelve objects with three features and two classes. Let $B_1 = \{f_1, f_2\}$ and $B_2 = \{f_1, f_3\}$, the confusion vectors can be obtained as $\mathrm{V}(B_1) = \{0,0,0,0,1,1,0,0,0,0,1,1\}$ and $\mathrm{V}(B_2) = \{0,0,0,0,1,1,0,0,0,0,1,1\}$. Based on Eq.(7), the dependency between D and B_1, B_2 can be calculated as $\gamma_{B_1}(D) = \gamma_{B_2}(D) = 8/12 = 0.6667$. On the other hand, we can find that the centroids of classes '1' and '2' under B_1 are $CP_1^{B_1} = (0.3617, 0.2483)$ and $CP_2^{B_1} = (0.5867, 0.4117)$. Then, we can calculate the distance and fuzzy membership for each object in the boundary region, which are shown in Table 2. Taking the object of x_5 as an example, according to Eq. (10), the distance between x_5 and $CP_1^{B_1}$, $CP_2^{B_1}$ can be calculated as $d_{B_1}(x_5, \beta_1) = \sqrt{(0.45 - 0.3617)^2 + (0.37 - 0.2483)^2} = 0.1504$ and similarly, $d_{B_1}(x_5, \beta_2) = 0.1429$. From Eq. (9), the fuzzy membership of object x_5 with respect to class β_1 is calculated as $\mu_{5,1} = 1/\left((\frac{0.1504}{0.1504})^2 + (\frac{0.1504}{0.1429})^2\right) = 0.4745$. Similarly, $\mu_{5,1} = 0.5255$. The class separability degree of boundary region and the proposed integrated criterion can then be calculated as $\delta_{B_1}(D) = (\mu_{5,1} + \mu_{6,1} + \mu_{11,2} + \mu_{12,2})/12 = 0.1054$ and $\rho_{B_1}(D) = 0.6667 + 0.1054 = 0.7721$. Similarly, we have $\delta_{B_2}(D) = 0.2152$ and $\rho_{B_2}(D) = 0.8819 > \rho_{B_1}(D)$. Consequently, it is concluded that B_2 is more discriminative than B_1.

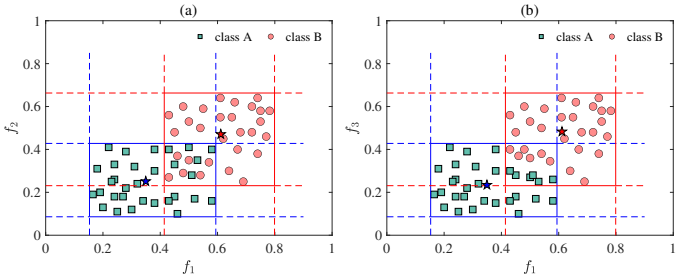

Fig. 1: A toy example to show the class separability in the boundary region of rough hypercuboid approach.

TABLE 1: A simple decision table.

U	f_1	f_2	f_3	D	U	f_1	f_2	f_3	D
x_1	0.22	0.38	0.42	1	x_7	0.43	0.56	0.24	2
x_2	0.30	0.10	0.70	1	x_8	0.71	0.64	0.16	2
x_3	0.18	0.13	0.67	1	x_9	0.69	0.25	0.55	2
x_4	0.52	0.16	0.64	1	x_{10}	0.77	0.46	0.34	2
x_5	0.45	0.37	0.53	1	x_{11}	0.44	0.27	0.44	2
x_6	0.50	0.35	0.54	1	x_{12}	0.48	0.29	0.45	2

TABLE 2: Distance and fuzzy membership values of boundary objects in Table 1 under $B_1 = \{f_1, f_2\}$.

$BND_{B_1}(D)$	$d_{B_1}(x_i, \beta_1)$	$d_{B_1}(x_i, \beta_2)$	$\mu_{B_1}(x_i, \beta_1)$	$\mu_{B_1}(x_i, \beta_2)$
x_5	0.1504	0.1429	0.4745	0.5255
x_6	0.1717	0.1064	0.2774	0.7226
x_{11}	0.0813	0.2039	0.8629	0.1371
x_{12}	0.1255	0.1618	0.6245	0.3755

4.2 Rough Hypercuboid Approach based Online Feature Selection

Streaming feature selection assumes that features exist in a stream model that arrives one by one over time. Compared with the batch feature selection, the entire feature spaces of datasets in streaming scenarios cannot be specified in advance, which requires the feature selection algorithms are capable of constructing discriminative feature subset incrementally by selecting newly arrived significant features and eliminating the redundant features contained in the existing feature subset. Formally, let $DT_{t-1} = (U, C_{t-1} \cup D)$ be a decision table at time $t-1$, where $C_{t-1} = \{f_1, f_2, ..., f_{t-1}\}$. At time t, we have $C_t = C_{t-1} \cup \{f_t\}$ as features flow in one by one in streaming scenarios.

Given the selected feature subset S_{t-1} at time $t-1$, and an incoming feature f_t at time t, the significance of f_t can be defined based on the dependency criterion as follows.

$$\sigma_{S_{t-1}}(D, f_t) = \gamma_{S_{t-1}\cup\{f_t\}}(D) - \gamma_{S_{t-1}}(D), \tag{13}$$

where $\gamma_{S_{t-1}}(D) = \frac{|POS_{S_{t-1}}(D)|}{|U|}$. If $\sigma_{S_{t-1}}(D, f_t) > 0$, the feature f_t can be deemed significant with respect to S_{t-1} and should be selected. If $\sigma_{S_{t-1}}(D, f_t) = 0$, then f_t is regarded as redundant for S_{t-1}.

The mutually exclusive subset B of S_{t-1} with respect to f_t satisfies the following condition.

$$\begin{aligned}\sigma_{S_{t-1}\cup\{f_t\}-B}(D, B) &= \gamma_{S_{t-1}\cup\{f_t\}}(D) \\ &\quad - \gamma_{S_{t-1}\cup\{f_t\}-B}(D) = 0.\end{aligned} \tag{14}$$

Eq. (14) indicates that there may exist a subset B of S_{t-1} that appears to be redundant if f_t is selected. To find the exclusive subset B, the NON-SIGNIFICANT function shown in Algorithm 1 can be employed, where the backward elimination method is used to determine whether a feature in $B - \{f\}$ is redundant with respect to the remaining features in the current subset S [27].

By introducing the proposed integrated criterion of dependency into the relevance and redundancy analysis based on rough hypercuboid approach, a novel online streaming feature selection algorithm is developed in Algorithm 2 which consists of two components: online relevance selection and online redundancy elimination. In the algorithm, steps 4-6 perform the relevance analysis for each feature

Algorithm 1 NON-SIGNIFICANT function

Input: S: The feature set; f: The incoming feature.
Output: B: The redundant feature set.
1: $B = \{\}, T = S - \{f\}$;
2: **while** $(|T| \neq 0)$ **do**
3: g=Random$\{f_i \in T, i = 1, 2, ..., |T|\}$;
4: **if** $\sigma_{S-\{g\}](D, g) == 0$ **then**
5: $B = B \cup \{g\}, S = S - \{g\}$;
6: **end if**
7: $T = T - \{g\}$;
8: **end while**
9: **return** B.

Algorithm 2 Rough hypercuboid approach based online feature selection (RHOFS)

Input: $(U, C \cup D)$: The decision table.
Output: S: The selected feature subset.
1: $S = \{\}, i = 1, \theta = 0$;
2: Get a new feature f_{t_i} of C at time stamp t_i;
3: **repeat**
4: **if** $\gamma_{f_{t_i}}(D) < \theta$ **then** discard f_{t_i} and go to step 18;
5: **end if**
6: **if** $\sigma_S(D, f_{t_i}) > 0$ **then** $S = S \cup \{f_{t_i}\}, \theta = \frac{1}{|S|}\gamma_S(D)$;
7: **else if** $\sigma_S(D, f_{t_i}) == 0$ **then**
8: $S = S \cup \{f_{t_i}\}$, B=NON-SIGNIFICANT(S, f_{t_i});
9: **if** $|B| == 0$ **then** $S = S - \{f_{t_i}\}$, and go to step 18;
10: **end if**
11: **if** $|B| > 1$ **then** $S = S - B, \theta = \frac{1}{|S|}\gamma_S(D)$;
12: **else if** $|B| == 1$ **then**
13: **if** $\gamma_S(D) < 1$ **then** $f' = \arg \max_{f \in B \cup \{f_{t_i}\}} \delta_{S-\{f\}}(D)$;
14: **else if** $\gamma_S(D) == 1$ **then** $f' = \arg \min_{f \in B \cup \{f_{t_i}\}} \rho_f(D)$;
15: **end if**
16: $S = S - \{f'\}$;
17: **end if**
18: **end if**
19: **until** no more features are available;
20: **return** S.

incrementally at its arrival. Specifically, step 4 calculates the dependency of feature f_{t_i} in terms of $\gamma_{f_{t_i}}(D)$ and discard it when the dependency degree is smaller than the threshold θ. In order to meet the maximal-dependency criterion which aims to find a feature subset with the largest dependency and smallest cardinality, the relevance threshold θ is set to be $\frac{1}{|S|}\gamma_S(D)$ so as to filter out low-relevance features. Steps 6 evaluates the significance of f_{t_i} with respect to the selected feature set S. If $\sigma_S(D, f_{t_i})$ is greater than 0, then f_{t_i} is deemed to be significant and should be selected, the threshold θ is then updated correspondingly. If $\sigma_S(D, f_{t_i})$ equals to 0, f_{t_i} will not be immediately discarded, but will sent to the stage of online redundancy elimination for further analysis. Based on the features achieved by the online relevance selection, we further consider the redundancy criterion which used for searching mutually exclusive features. The NON-SIGNIFICANT function shown in Algorithm 1 is used to select the mutually exclusive subset B with the maximum number of features in step 8. Three-way decisions for f_{t_i} are activated in steps 9-17 in terms of the cardinality of B. (1) If $|B|$ is equal to 0, then f_{t_i} should be discarded from S and waits for the next upcoming feature. (2) If $|B|$ is greater than 1 in step 11, the mutually exclusive subset B should be replaced with f_{t_i} to guarantee the number of features in S is as small as possible while keeping the dependency of S fixed. Then B is discarded from S and the relevance threshold θ is updated because of the change in S. (3) If $|B|$ is equal to 1, it is necessary to evaluate the discriminating ability of f_{t_i} and check the redundancy of B. If the dependency degree of S is smaller than 1 and the class separability degree of boundary region satisfies $\delta_{S-\{f_{t_i}\}} > \delta_{S-B}$ in step 13, which means that keeping B in S is more able to distinguish objects belonging to different classes in the boundary regions, then f_{t_i} should be discarded from S in step 16. By contrary, previously selected feature in B will be discarded. And when $\delta_{S-\{f_{t_i}\}}$ equals δ_{S-B}, remove one of f_{t_i} and B randomly. Step 14 evaluates the significance of f_{t_i} and the features from B based on the proposed integrated criterion as shown in Eq. (12) if $\gamma_S(D)$ is equal to 1. The above iterations will continue until no more incoming features are available, and an optimal subset S will be returned in step 20.

In our algorithm, the computational complexity to calculate the dependency of feature f_{t_i} and compare it with θ in step 4 is $O(nc)$, where n and c are the number of objects and classes respectively. Similarly, the complexity to calculate the significance of f_{t_i} is also $O(nc)$ in steps 6-7. Because the redundancy of each feature in S needs to be checked by NON-SIGNIFICANT function, the time complexity of step 8 is $O(|S|nc)$. In order to obtain the class separability degree of boundary region, the centroid of each class under each feature and the distance from each object in boundary region to each centroid need to be calculated. Hence, steps 13-14 have $O(n + n'c)$ time complexity, where n' is the number of objects in boundary regions. In effect, the total computational complexity of RHOFS is $O(m^2nc)$ in the worst case, where m is the number of conditional features.

5 Distributed Online Feature Selection

In this section, we present a distributed online feature selection algorithm with streaming features based on rough hypercuboid approach (RHDOFS). The proposed RHDOFS algorithm has been implemented under the Apache Spark framework to ensure that three major iterative steps in RHDOFS (i.e., calculations of the hypercuboid partition matrix, the dependency criterion of features, and the class separability degree of boundary region) are optimized to their fullest degree of parallelism in the distributed environment.

Based on the definition of the hypercuboid partition matrix, it is necessary to calculate the minimum and maximum values of each incoming feature with respect to each class firstly. Given a decision table $DT_t = (U, C_t \cup D)$ be at time t, the new feature $f_t \in C_t$ and $U/D = \{\beta_1, \beta_2, \cdots, \beta_c\}$, the value range vector LU_t of f_t can be defined as follows.

$$\mathrm{LU}_t = \{(L_{1t}, U_{1t}), (L_{2t}, U_{2t}), ..., (L_{ct}, U_{ct})\}, \quad (15)$$

where $L_{it} = \min_{x_j \in \beta_i}\{x_j(f_t)\}$ means the minimum value of f_t with respect to class β_i and $U_{it} = \max_{x_j \in \beta_i}\{x_j(f_t)\}$ means the maximum value of f_t with respect to class β_i.

The parallelization method of calculating the value range vector of one new feature f_t at time stamp t is shown in Algorithm 3. Data with incoming features are partitioned into multiple chunks horizontally and stored in the distributed memory as a Resilient Distributed Datasets (RDD) on Spark. The input of Algorithm 3 is an RDD of objects characterized by the new coming feature and the decision feature, which can be obtained via a textFile operator. Steps 2-5 use the map function to transform each object x_j into a $<$ key, value $>$ pair in parallel, where key is the index of the decision class to which x_j belongs and value is a two-tuple initialized to the feature value of x_j under f_t. The reduceByKey function can aggregate the two-tuples with the same class, and then calculate the minimum value of their first elements and the maximum value of the second elements, so as to obtain the value range of the new feature with respect to each class in steps 6-10. The results will be collected as an array in Step 11 and transformed into a new array, where the i-th element is the value range of the new feature with respect to the i-th class in steps 12-15.

Algorithm 3 Calculate the value range vector of the newly arrived feature(*CalLUVector*)

Input: DT_t: RDD of objects with the new feature f_t and the decision feature at time stamp t.
Output: $LUVec_t$: Value range vector of f_t.

1: $\mathrm{LU}_t =$
2: **map** $< j, x_j > \in DT_t$
3: Identify the i-th decision class to which x_j belongs;
4: ***EMIT*** $< i, (x_j(f_t), x_j(f_t)) >$
5: **end map**
6: **reduceByKey** (val_1, val_2) **do**
7: $val_{min} = \min\{val_1.\_1, val_2.\_1\}$;
8: $val_{max} = \max\{val_1.\_2, val_2.\_2\}$;
9: ***EMIT*** (val_{min}, val_{max})
10: **end reduce**
11: **collect**
12: $\mathrm{LUVec}_t =$ new Array(c)
13: **for** $< i, (val_{min}, val_{max}) >$ **in** LU_t **do**
14: $LUVec_t(i) = (val_{min}, val_{max})$;
15: **end for**
16: **return** LUVec_t.

To address the global data communication issue for calculating hypercuboid partition matrix, a horizontal partitioning method is proposed to accommodate Spark MapReduce paradigm. We decompose the hypercuboid partition matrix, and define a novel concept of distributed hypercuboid partition vector for streaming feature scenario as follows.

Given a decision table $DT_t = (U, C_t \cup D)$ at time t and a new arrival feature $f_t \in C_t$. Suppose the value range vector f_t is LU_t, and the previously selected feature subset is S_{t-1}, the distributed hypercuboid partition vector of an object $x_j \in U$ under f_t is defined as $\mathrm{DV}_j(f_t) = \mathrm{H}(f_t)[:, j] = \{h_{ij}(f_t)\}$, and

$$h_{ij}(f_t) = \begin{cases} 1, & L_{it} \le x_j(f_t) \le U_{it}; \\ 0, & \text{otherwise}, \end{cases} \tag{16}$$

where $i = 1, 2, ..., c$ and $M[:, j]$ indicates the j-th column of a matrix M.

Eq. (16) shows that the calculation of the distributed rough hypercuboid equivalence partition vector for any pairs of objects are independent of each other, and can be performed in a completely local way without communication among partitions. Algorithm 4 describes the detailed parallelization process, which has also depicted graphically in Fig. 2. In Algorithm 4, the map function transforms each object with the new feature into its distributed hypercuboid partition vector under that feature in parallel. And the result from the map function is cached in the distributed memory of Spark cluster by the persist operator. The RDDs of the distributed hypercuboid partition vectors of each object under the arrived features make up a RDD, which will be cached in the cluster and named after $\mathrm{DV}_{S_{t-1}}$ before time stamp t. Each record of the RDD is an array that contains the distributed hypercuboid partition vector of one object under each selected feature. The above process can be implemented by the join operator in Spark, which merges two RDDs by joining their records via the one-to-one correspondence combination.

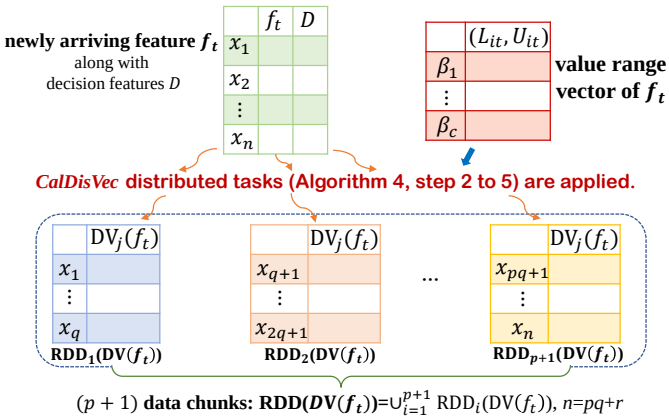

Fig. 2: Parallel calculation process of the distributed hypercuboid partition vector of each object under the newly arriving feature f_t.

Based on Eq. (16), the dependency between the new arrival feature f_t and the decision feature set D can be calculated by summarizing the accumulated vector values of $\mathrm{DV}_j(f_t)$ as follows.

$$\begin{aligned} \gamma_{f_t}(D) &= 1 - \frac{1}{n}\sum\nolimits_{j=1}^{n} v_j(f_t) \\ &= \sum\nolimits_{j=1}^{n}[1 - \min\{1, \mathrm{sum}\{\mathrm{DV}_j(f_t)\}\}]/n. \end{aligned} \tag{17}$$

And, the dependency between the feature subset $B_t \subseteq S_{t-1} \cup \{f_t\}$ and the decision feature set D can be calculated as follows.

$$\gamma_{B_t}(D) = \sum\nolimits_{j=1}^{n}[1 - \min\{1, \mathrm{sum}\{\mathrm{DV}_j(B_t)\}\}]/n, \tag{18}$$

where $\mathrm{DV}_j(B_t) = \bigcap_{f' \in B_t} \mathrm{DV}_j(f')$ and $h_{ij}(B_t) = \bigcap_{f' \in B_t} h_{ij}(f')$ for $i = 1, 2, ..., c$.

Algorithm 4 Calculate the distributed hypercuboid partition vector of each object under the newly arrived feature(*CalDisVec*)

Input: DT_t: RDD of objects with the new feature f_t and the decision feature at time stamp t;
LUVec_t: Value range vector of f_t.
Output: DV_{f_t}: RDD of the distributed rough hypercuboid equivalence partition vector of each object under f_t.

$\text{DV}_{f_t} =$
map $< j, x_j > \in DT_t$
  Calculate the distributed rough hypercuboid equivalence partition vector $\text{DV}_j(f_t)$ using Eq. (16);
  EMIT $< j, \text{DV}_j(f_t) >$
end map
persist
return DV_{f_t}.

Algorithm 5 Calculate the dependency of candidate feature subset by the arrival of feature(*CalDepen*)

Input: B_t: Feature subset of the union of the new feature f_t and the selected feature set S_{t-1};
$\text{DV}_{S_{t-1}\cup\{f_t\}}$: RDD of $Array\{\text{DV}_j(f)\}$s, which are arrays of $\text{DV}_j(f), (f \in S_{t-1} \cup \{f_t\})$.
Output: $Depen$: Dependency value of B_t.

$Depen =$
map $< j, Array\{\text{DV}_j(f)\} > \in \text{DV}_{S_{t-1}\cup\{f_t\}}$
  $vec =$ new onesVector(c);
  for f' **in** B_t **do**
    $vec = vec \cap \text{DV}_j(f')$;
  end for
  $val = [1 - \min\{1, \text{sum}\{vec\}\}]/n$;
  EMIT val
end map
reduce (val_1, val_2) **do**
  EMIT $val_1 + val_2$
end reduce
return $Depen$.

In Eq.(18), the calculation of the term $[1 - \min\{1, \text{sum}\{\text{DV}_j(B_t)\}\}]/n$ only concerns the object x_j itself and independent of other objects, which is amenable to be parallelized. Algorithm 5 describes the dependency computation through MapReduce process, where the map function is used to calculate the term $[1 - \min\{1, \text{sum}\{\text{DV}_j(B_t)\}\}]/n$ of Eq. (17) for each object, and the results are eventually aggregated by the reduce function so as to obtain the dependency of a feature subset. Fig. 3 explains the parallel calculation of the feature dependency with respect to $B_t \subseteq S_{t-1} \cup \{f_t\}$. It is worth noting that if $B_t = \{f_t\}$, the dependency of f_t can be obtained by Algorithm 5 and compared with the relevance threshold θ in the step 4 of RHOFS algorithm. If $B_t = S_{t-1}$ or $B_t = S_{t-1} \cup \{f_t\}$, the dependency values of S_{t-1} and $S_{t-1} \cup \{f_t\}$ should be calculated to further achieve the significance and check the redundancy of f_t. Similarly, if $B_t = S_{t-1}$ or $B_t = S_{t-1} - \{g\}$ where $g \in S_{t-1}$, the significance of g with respect to S_{t-1} would be calculated to find the mutually exclusive subset in S_{t-1} by using the NON-SIGNIFICANT function.

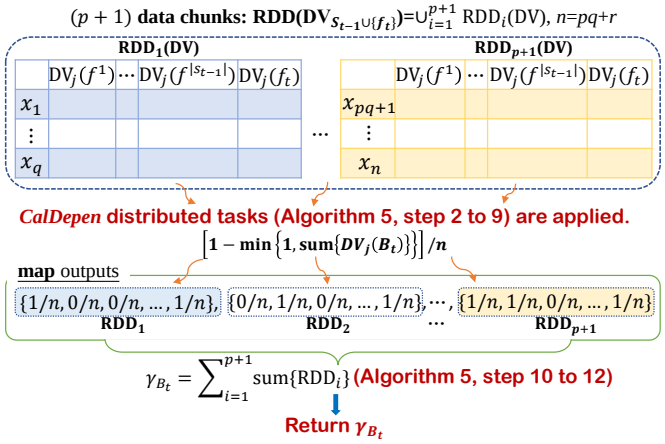

Fig. 3: Parallel calculation process of the dependency of the feature subset $B_t \subseteq S_{t-1} \cup \{f_t\}$.

Algorithm 6 Calculate the centroid of the newly arrived feature with respect to each class(*CalCentroid*)

Input: DT_t: RDD of objects with the new feature f_t and the decision feature at time stamp t.
Output: $CPArr_t$: Array of the center point of each class under f_t.

$CP_t =$
map $< j, x_j > \in DT_t$
  Identify the i-th decision class to which x_j belongs;
  EMIT $< i, (1, x_j(f_t)) >$
end map
reduceByKey (val_1, val_2) **do**
  $val_1 = val_1.\_1 + val_2.\_1$;
  $val_2 = val_1.\_2 + val_2.\_2$;
  EMIT (val_1, val_2)
end reduce
collect
$CPArr_t =$ new Array(c)
for $< i, (val_1, val_2) >$ **in** CP_t **do**
  $CPArr_t(i) = val_2/val_1$;
end for
return $CPArr_t$.

Prior to the computation of class separability degree of boundary region for redundancy analysis, the centroid of each class needs to be identified. Algorithm 6 is employed to obtain the centroid of the new incoming feature with respect to each class. In the algorithm, the map function transforms each object x_j into a $< \text{key}, \text{value} >$ pair, where key is the index of the decision class to which x_j belongs and value is a two-tuple consists of the number of "1" and the new feature value of x_j, respectively. In this way, the reduceByKey function aggregates all the two-tuples with the same class and counts the number of appearances of each combination, so as to achieve the cardinality of each decision class and their accumulated feature values under the new feature. Steps 12-15 calculate the center point of each class and cache them as an array, in which the i-th element denotes the centroid of the i-th decision class under the new feature. At time stamp t, the arrays from all arrived features make up a matrix *CPMat*, which contains

the centroid of each class under each arrived feature.

Based on the distributed hypercuboid partition vector $\mathrm{DV}_j(f)$ of each object $x_j \in U$ under each feature f in $S_{t-1} \cup \{f_t\}$, the class separability degree of boundary region under the feature subset $B_t \subseteq S_{t-1} \cup \{f_t\}$ can be calculated as follows.

$$\begin{aligned}\delta_{B_t}(D) &= \frac{1}{n}\sum_{j=1}^{n} \min\{\mu_{B_t}(x_j,\beta),\ v_j(B_t)\} \\ &= \sum_{j=1}^{n} \min\{\mu_{B_t}(x_j,\beta),\ \min\{1, \mathrm{sum}\{\mathrm{DV}_j(B_t)\}\}\}/n,\end{aligned} \tag{19}$$

where β is the decision class to which object x_j belongs. The summation term in Eq. (19) means that if x_j falls into the boundary region, the term is equal to $\mu_{B_t}(x_j,\beta)$, otherwise, it equal to "0". Calculation of the summation term can be applied to each object independently, and the horizontal partitioning method allows different objects to be calculated in parallel.

Algorithm 7 outlines the detailed process for parallel calculating the class separability degree of boundary region. In the input of Algorithm 7, the RDD of $\mathrm{DV}_{S_{t-1}\cup\{f_t\}}$ is used to calculate the confusion vector $v_j(B_t)$ of each object and its membership to boundary region, the RDD of DT_t is used to obtain the fuzzy membership of object with respect to each class. The join operator is employed to combine DT_t and $\mathrm{DV}_{S_{t-1}\cup\{f_t\}}$ RDDs in order to make a one-to-one correspondence between the distributed hypercuboid partition vectors and objects. The map function transforms each object in boundary regions into the ratio of its fuzzy membership to the number of objects using Eqs. (9) and (10) in steps 2-14. Eventually, the reduce function summarizes all the results to obtain the class separability degree of boundary region under the feature subset B_t. Apparently, Algorithm 7 is mainly used to compare the discriminating ability of the new arrival feature and its mutually exclusive subset in steps 12-17 of the sequential RHOFS algorithm. Fig. 4 further explains the parallel calculation of the class separability degree by using Algorithm 7.

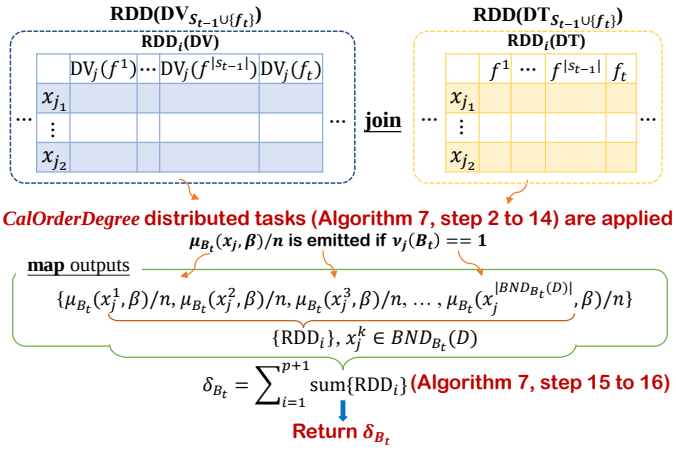

Fig. 4: Parallel calculation process of the class separability degree of feature subset $B_t \subseteq S_{t-1} \cup \{f_t\}$.

The overview of our distributed online streaming feature selection (RHDOFS) framework is illustrated in Fig. 5, where several major iterative steps in the sequential RHOFS algorithm are optimized to their fullest degree of parallelism

Algorithm 7 Calculate the class separability degree of feature subset(*CalOrderDegree*)

Input: B_t: Feature subset of the union of the new feature f_t and the selected feature set S_{t-1};
DT_t: RDD of objects with the new feature f_t and the decision feature at time stamp t.
$\mathrm{DV}_{S_{t-1}\cup\{f_t\}}$: RDD of $Array\{\mathrm{DV}_j(f)\}$s, which are arrays of $\mathrm{DV}_j(f), (f \in S_{t-1} \cup \{f_t\})$;
CPMat: Matrix containing the centroid of each class under each arrived feature.
Output: $Order$: Order degree of B_t.

1: $Order =$
2: **map** $<j, (Array\{\mathrm{DV}_j(f)\}, x_j)> \in \mathrm{DV}_{S_{t-1}\cup\{f_t\}}$ **join** DT_t
3: $vec =$ new onesVector(c);
4: **for** f' **in** B_t **do**
5: $vec = vec \cap \mathrm{DV}_j(f')$;
6: **end for**
7: $v_j(B_t) = \min\{1, \mathrm{sum}\{vec\}\}$;
8: **if** $v_j(B_t) == 1$ **then**
9: Identify the centroid of each class under B_t using *CPMat*;
10: Calculate the distances between object x_j and the centroid of each class under B_t using Eq. (10);
11: Calculate the fuzzy membership $\mu_{B_t}(x_j,\beta)$ of object x_j with respect to its decision class using Eq. (9);
12: ***EMIT*** $\mu_{B_t}(x_j,\beta)/n$
13: **end if**
14: **end map**
15: **reduce** (val_1, val_2) **do**
16: ***EMIT*** $val_1 + val_2$
17: **end reduce**
18: **return** $Order$.

in the distributed environment, and the computational efficiency is thus improved by transferring the heavy computational burden to Spark clusters. RHDOFS consists of three parts: data source, online feature selection and distributed memory. The data source will constantly generate data with newly arrived features, which are indicated by gray graphics at every time, such as the data of new feature f_t at time t. The original data and distributed hypercuboid partition vectors of objects under f_t and the selected subset S_{t-1} are persisted in the distributed memory of clusters. And, the data related to f_t and S_{t-1} are represented by green and blue cylinders respectively. Fig. 5 introduces the online streaming feature selection process of f_t using RHDOFS at time t, where the orange and golden rectangles indicate online relevance selection and redundancy elimination stages respectively, the red rectangles denote different possible feature selection results, and the black bold dashed lines refers to the operation of caching and freeing RDDs in distributed memory. During the online relevance selection process, the value range vector of f_t is calculated by Algorithm 3 firstly. Then the distributed hypercuboid partition vector of each object under f_t can be further obtained by Algorithm 4. To facilitate data reuse, results of Algorithm 4 and the original data of f_t are cached in the distributed memory. Next, Algorithm 5 is used to calculate the relevance of f_t. If the relevance is smaller than the threshold θ, f_t will be

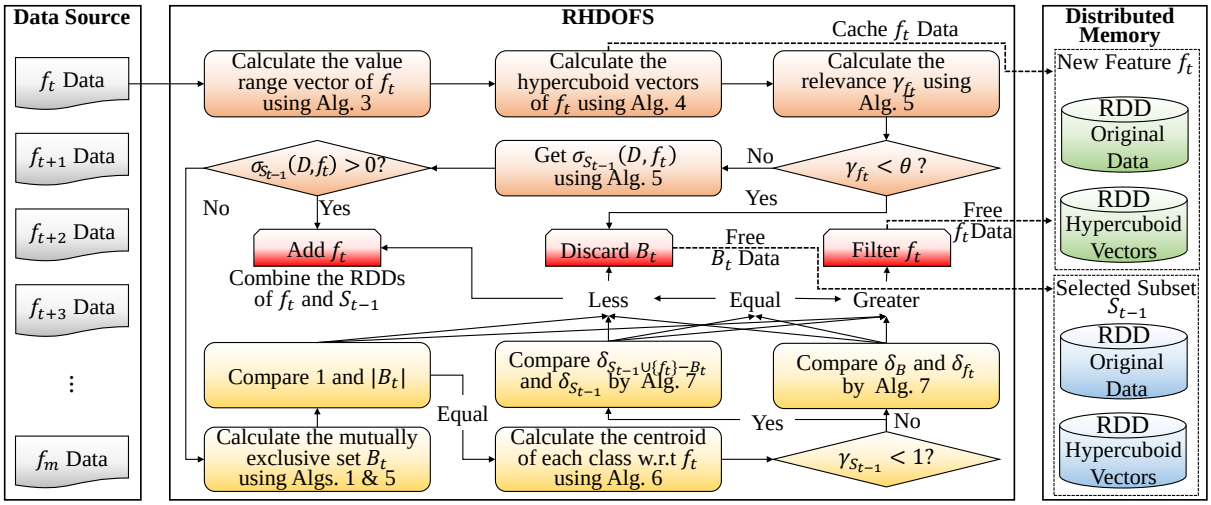

Fig. 5: Framework of the proposed distributed online feature selection method (RHDOFS).

filtered out as a low relevance feature, the data related to f_t in distributed memory will be released, and RHDOFS will continue to process the next incoming feature. Otherwise, Algorithm 5 is employed to calculate the dependency of the feature subset $S_{t-1} \cup \{f_t\}$ so as to obtain the significance of f_t. If the significance is greater than zero, f_t will be added into S_{t-1} as a relevant feature, the RDDs corresponding to f_t and S_{t-1} will be combined by using join operator, and the next incoming feature will be considered. Otherwise, f_t enters the redundancy elimination stage. As mentioned earlier, the calculation of feature significance in the NON-SIGNIFICANCE function also utilizes Algorithm 5, hence the mutually exclusive subset B_t in S_{t-1} of feature f_t can be identified by Algorithms 1 and 5. If the number of redundant features in B_t is smaller than one, f_t will be filtered out directly. If B_t contains more than one redundant feature, it will be removed from S_{t-1}, the cached data related to B_t in the distributed memory will be released, and f_t is selected. Otherwise, if there is only one feature in B_t, the center point of each class with respect to f_t is calculated using Algorithm 6, B_t and f_t are then evaluated. If the dependency of the current selected subset is smaller than one, the degree of class separability in boundary regions induced by $S_{t-1} \cup \{f_t\} - B_t$ and S_{t-1} are calculated as $\delta_{S_{t-1}\cup\{f_t\}-B_t}$ and $\delta_{S_{t-1}}$ using Algorithm 7, respectively. Then the feature which is capable of distinguishing more objects in boundary regions should be selected, the other features are removed and their cached data are freed. For example, when the degree of class separability $\delta_{S_{t-1}\cup\{f_t\}-B_t}$ is greater than $\delta_{S_{t-1}}$, f_t will be added into S_{t-1} and B_t will be removed. If the dependency of the current selected subset equals to one, Algorithm 7 is used to calculate the degree of class separability δ_{B_t} and δ_{f_t}, and the evaluation values of B_t and f_t can be obtained as ρ_{B_t} and ρ_{f_t}. Finally, the feature with a greater value will be selected, the other features will be filtered out.

6 Experimental Results

In this section, various comparative and evaluative experiments are conducted to validate the effectiveness and superiority of the proposed novel online feature selection algorithm (RHOFS), and test the efficiency and scalability of the distributed online feature selection algorithm (RHDOFS). The experiments regarding to RHOFS were carried out in Matlab R2019b and run on a computer with an Intel Core i9-9900 CPU@3.10GHz and 16 GB memory space. The programming language used in the RHDOFS implementation is Scala, which is a statically typed programming language that incorporates both functional and object-oriented programming. Besides, a Spark cluster composes of one master node and six slave nodes was used to evaluate the parallel scalability of RHDOFS. The configuration of each node is 8 Intel(R) Platinum 8269CY CPU@2.50GHz, 16 GB memory space, and installed with Ubuntu 18.04 64-bit, Apache Spark 2.4.5, Apache Hadoop (HDFS) 2.7.7, Jdk 1.8, Scala 2.11.12, and the network bandwidth is 1.2 Gbps. The SSH protocol facilitates data transmission among the nodes, and the HDFS file system is controlled by the Spark Master. The RPC protocol facilitates communication among nodes in the Spark cluster, including resource application, task distribution, etc. RHOFS and RHDOFS are publicly available to download from https://github.com/SCU-Cloud/RHFS.

The detailed description of experimental datasets is listed in Table 3, which contains nine Microarray[1] datasets (*Colon*, *CNS*, *Leukemia*, *Leukemia_3c*, *Leukemia_4c*, *MLL*, *SRBCT*, *Lung*, *Ovarian*), three UCI[2] datasets (*Arcene*, *Madelon*, *Isolet*), and one LIBSVM[3] dataset (*Epsilon*). These experimental datasets were taken as our testbeds by observing features one by one to simulate the situation of streaming features, each feature is inputted and evaluated by the feature selection algorithm individually. After performing feature selection over all streaming features, the feature evaluation indices could then be computed. The features of the datasets were min-max normalized in the experiments. The *Colon* dataset consists of 62 microarray experiments collected from colon cancer patients with 2000 gene expression levels, in which 40 tumor biopsies are from tumors and 22 biopsies

1. https://csse.szu.edu.cn/staff/zhuzx/Datasets.html
2. http://archive.ics.uci.edu/ml/datasets.php
3. https://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/

are from healthy parts of the colons. The *CNS* dataset is used for distinguishing between failed and succeed treatment outcomes in molecular investigation of treatment effectiveness for embryonal tumors, which contains 60 patients with 7129 genes, where 21 patients are survivors and 39 patients are failures. The *Leukemia* dataset consists of 7129 features and 72 samples to be categorized into two classes of ALL and AML. *Leukemia_3c* and *Leukemia_4c* are multi-class versions of *Leukemia*. The *MLL* dataset consists of 72 samples of 3 subtypes of leukemia cancer classes, and the dimension of *MLL* is 12582. The *SRBCT* dataset is the small round blue-cell tumor data set, which has 83 samples of four classes (EWS, BL, NB and RMS). Each samples has 2308 gene expression levels. The *Lung* dataset is used for lung cancer classification consists of 181 samples, among which 31 samples belong to MPM and 150 samples belong to ADCA. Each sample is described by 12533 genes. The *Ovarian* cancer dataset consists of 15154 genes, categorized as cancer and normal classes, having 253 samples include 162 cancer and 91 normal samples. The *Arcene* dataset is used for distinguishing between cancer and normal patterns from mass spectrometric data, which comprises 200 samples and 10000 features. *Madelon* is an artificial dataset used for the problem of random data classification with sparse binary input variables. Both *Arcene* and *Madelon* were used by the NIPS 2003 feature selection challenge. The *Isolet* dataset consists of several spectral coefficients of utterance of English alphabets by 150 speakers. There are 617 real features with 7797 samples and 26 classes. The *Epsilon* dataset is an artificial binary dataset created for the Pascal Large Scale Learning Challenge in 2008, which contains 500000 samples described by 2000 features. The first eleven smaller datasets in Table 3 were used to establish the effectiveness of our RHOFS algorithm over the existing representative online feature selection algorithms. The last four large-scale datasets were applied to evaluate the efficiency and parallel performance of our RHDOFS algorithm, where *Isolet-1000* and *Isolet-2000* were obtained by expanding the objects of *Isolet* by 1000 and 2000 times respectively. Similarly, *Epsilon-10* and *Epsilon-20* were obtained by replicating *Epsilon* dataset 10 and 20 times, respectively.

TABLE 3: Description of experimental datasets.

No.	Datasets	Objects/Size	Features	Classes
1	Colon	62	2,000	2
2	CNS	60	7,129	2
3	Leukemia	72	7,129	2
4	Leukemia_3c	72	7,129	3
5	Leukemia_4c	72	7,129	4
6	MLL	72	12,582	3
7	SRBCT	83	2,308	4
8	Lung	181	12,533	2
9	Ovarian	253	15,154	2
10	Arcene	200	10,000	2
11	Madelon	2,000	500	2
12	Isolet-1000	7,797,000/29.2G	617	26
13	Isolet-2000	15,594,000/58.4G	617	26
14	Epsilon-10	5,000,000/141.3G	2,000	2
15	Epsilon-20	10,000,000/282.6G	2,000	2

Six representative online methods, including Alpha-investing [8], OSFS [9], Fast-OSFS [10], SAOLA [11], OFS-A3M [12] and OFS-Gapknn [13] were chosen as comparison algorithms to shown the superiority of the proposed RHOFS algorithm for streaming feature selection. The parameters for Alpha-investing are set as their default settings in [8]. The significance level α for Fast-OSFS, OSFS and SAOLA algorithms is set to be 0.01. Naive Bayes, JRip and J48 classifiers using ten-fold cross-validation are performed to evaluate the quality of feature subsets achieved by different algorithms on each datasets. For the sake of fair comparison, the training and test datasets of all algorithms remain the same for each fold. OFS-A3M, OFS-Gapknn and RHOFS algorithms ran ten times on each dataset independently owing to the randomness of results in their redundancy analysis stages.

6.1 Impact of Streaming Features Order

In this section, we explore the influences of different order of streaming features on the performance of RHOFS from the aspects of running time, the cardinality and classification accuracy of selected feature subsets. And, three types of streaming order including the original, inverse and random streaming orders are considered in the experiments.

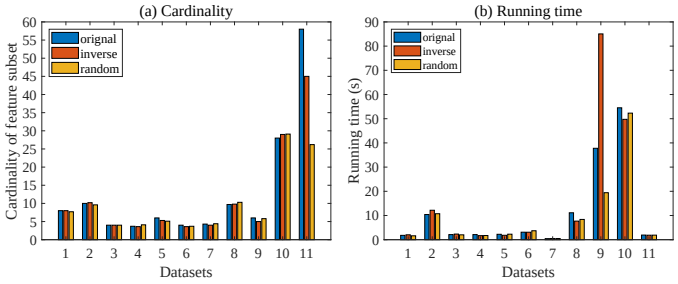

Fig. 6: Cardinality of feature subset and running time of RHOFS with different streaming orders (in second).

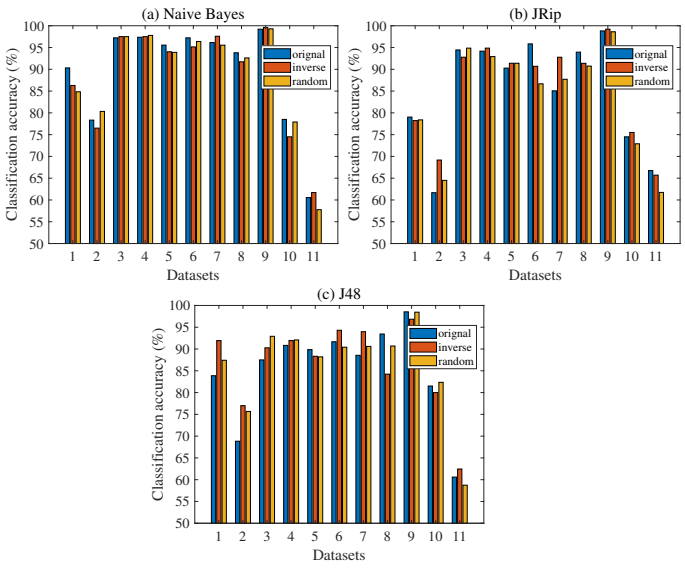

Fig. 7: Average classification accuracy of RHOFS with different streaming orders.

Figs. 6 and 7 present the cardinality of the selected feature subset, running time and average classification accuracy of our RHOFS algorithm under Naive Bayes, JRip,

and J48 classifiers with three different orders of streaming features on the eleven smaller datasets. The datasets are represented numerically in order in Figs. 6 and 7. It can be seen from the results that the cardinality of feature subset and the running time of RHOFS with different streaming orders are almost the same except for the 11-th dataset (*Madelon*) in Fig. 6(a) and the 9-th dataset (*Ovarian*) in Fig. 6(b). Furthermore, Fig. 7 also shows that the average classification accuracies achieved by three classifiers fluctuate a little in response to changes in the orders of streaming features. Friedman test in statistics at a significance level $\alpha = 0.05$ was performed to verify whether there exists a significant difference between the cardinality of feature subset, running time and average classification accuracy of the original, inverse and random streaming orders. And, the p-values of the observed comparison results are summarized in Table 4. It appears that all p-values shown in Table 4 are greater than 0.05, which means that the null hypothesis is accepted that the influences of different streaming orders on the proposed RHOFS algorithm in terms of the cardinality of the selected feature subset, running time and average classification accuracy have no significant differences.

TABLE 4: p-values of the comparison among three feature stream orders (O: Original, I: Inverse, R: Random).

Evaluation Measure	O vs. I	O vs. R	I vs. R
Naive Bayes	0.7630	0.3657	1.0000
JRip	0.7630	0.3657	0.1317
J48	0.3657	0.7630	0.7630
Number of features	0.3173	0.5271	0.5271
Running time	0.7630	0.3657	0.3657

6.2 Comparison with Relevant Algorithms

In this section, we compare our RHOFS algorithm with six representative online feature selection algorithms to appraise the effectiveness of reduced feature subset for classification. Table 5 summarizes the average cardinalities of the reduced feature subset of seven algorithms on eleven datasets. As it can be seen, the proposed RHOFS method is inferior to Fast-OSFS and OSFS algorithms, but outperforms Alpha-investing, SAOLA and OFS-A3M algorithms in terms of the average cardinality of feature subset and the average rank. Compared with OFS-Gapknn, RHOFS can achieve more compact subsets on most of the datasets, although the average number of features of RHOFS is larger.

Tables 6-8 further represent the classification performance of comparative algorithms using Naive Bayes, JRip and J48 classifiers in terms of accuracy, precision, and recall, where W/L/T (win/loss/tie) indicates the number of datasets on which RHOFS has higher (or lower, equal to) classification performance than the corresponding comparative algorithm. The Wilcoxon signed-rank test to compare RHOFS against other comparative algorithms in a pairwise manner at a 5% significance level are performed to evaluate the significant advantages of RHOFS, and the corresponding p values are listed in Table 9, in which if the p value is smaller than 0.05 (not underlined) indicates that the performance achieved by the corresponding algorithm is significantly different from RHOFS.

TABLE 5: Average cardinalities of the selected feature subset by seven algorithms on eleven datasets.

Datasets	Alpha-investing	Fast-OSFS	OSFS	SAOLA	OFS-A3M	OFS-Gapknn	RHOFS
Colon	9	3	**2**	6	26	11.5	8
CNS	4	3	**2**	9	18	14.8	10
Leukemia	16	6	**3**	21	11.1	8.2	4
Leukemia_3c	8	5	**2**	16	12.2	8.8	3.7
Leukemia_4c	13	3	**2**	16	18.9	12.7	6
MLL	14	6	**3**	35	7.6	9	4
SRBCT	26	7	**3**	17	6	7.6	4.3
Lung	40	9	**6**	36	57.5	17.9	9.7
Ovarian	84	4	**3**	8	3.1	**3**	6
Arcene	19	6	**2**	18	30.2	19.6	28
Madelon	4	3	3	3	**2**	**2**	58
Average	21.55	5.00	**2.82**	16.82	17.51	10.46	12.88
Avg. Rank	(5.36)	(2.82)	(**1.32**)	(5.27)	(5.23)	(4.27)	(3.73)

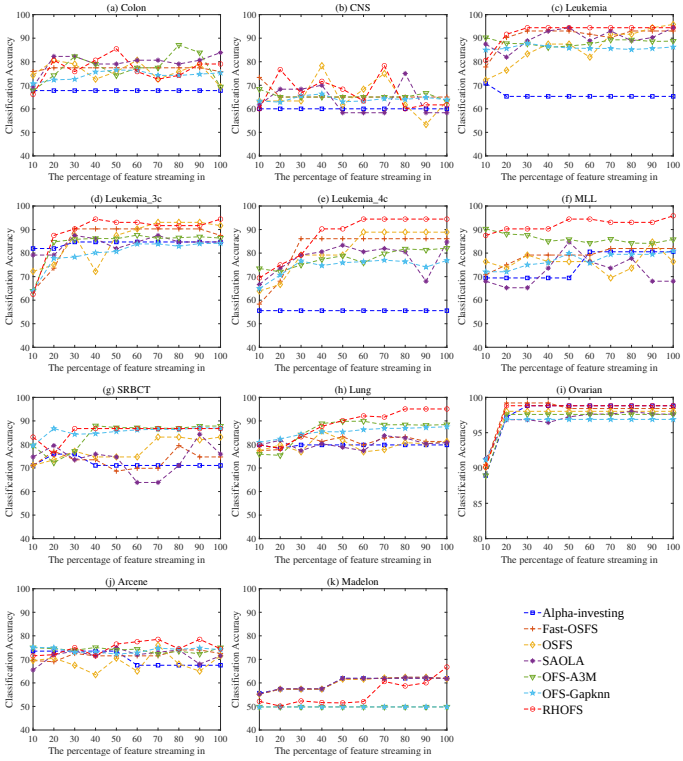

Fig. 8: Average classification accuracy of the selected feature subsets during features streaming with JRip.

A common observation can be perceived from Tables 6-8 that our RHOFS algorithm performs very well and owns higher performance in terms of classification performance than the other six comparative algorithms, which has been reflected in the achieved maximum average classification metrics and minimum average rank under Naive Bayes, JRip and J48 classifiers. Compared with Alpha-investing, the average accuracy, precision and recall of RHOFS are better than that of Alpha-investing on almost all datasets. And the average difference between the two algorithms is more than 10% using three classification metrics under three classifiers. Table 9 also shows that there is a significant difference between RHOFS and Alpha-investing using three different classifiers. That's because Alpha-investing only select the first few features of streaming features and

TABLE 6: Average classification performance of seven algorithms on eleven datasets with Naive Bayes.

Datasets	Average accuracy						
	Alpha-investing	Fast-OSFS	OSFS	SAOLA	OFS-A3M	OFS-Gapknn	RHOFS
Colon	59.6774	85.4839	74.1935	85.4839	75.8065	73.0645	**90.3226**
CNS	56.6667	66.6667	76.6667	75.0000	**78.3333**	62.6667	**78.3333**
Leuk	73.6111	98.6111	**100.000**	95.8333	97.7778	93.1944	97.2222
Leuk_3c	80.5556	97.2222	91.6667	94.4444	96.9444	93.4722	**97.3611**
Leuk_4c	62.5000	87.5000	84.7222	90.2778	90.1389	90.5556	**95.5555**
MLL	94.4444	93.0556	86.1111	94.4444	92.7778	91.8056	**97.2222**
SRBCT	91.5663	90.3614	87.9518	93.9759	96.0241	96.0241	**96.1446**
Lung	89.1626	90.6404	85.2217	92.1182	88.1281	89.8522	**93.7931**
Ovarian	98.0237	**100.000**	99.6047	99.6047	99.3676	98.8933	99.2095
Arcene	77.0000	78.5000	65.0000	**83.0000**	70.0000	69.5000	78.5000
Madelon	60.5500	60.9000	**61.2500**	**61.2500**	49.6500	49.6500	60.5500
Average	76.7053	86.2674	82.9444	87.7666	84.9953	82.6071	**89.4740**
Avg. Rank	(5.73)	(3.27)	(4.82)	(2.91)	(4.05)	(5.09)	(**2.14**)
W/L/T	10/0/1	7/3/1	8/3/0	8/3/0	8/2/1	11/0/0	-
Datasets	Average precision						
	Alpha-investing	Fast-OSFS	OSFS	SAOLA	OFS-A3M	OFS-Gapknn	RHOFS
Colon	61.7460	84.3786	74.5000	83.9459	74.9739	73.1709	**90.2381**
CNS	55.2189	62.2222	75.4209	72.6077	**76.1962**	61.3022	75.1250
Leuk	73.6111	98.9583	**100.000**	95.0669	97.3013	92.9395	96.9362
Leuk_3c	85.3992	**97.8408**	89.3129	92.9012	96.3545	92.4954	96.7393
Leuk_4c	50.9303	88.6251	81.9420	90.3056	87.0835	86.9966	**96.0911**
MLL	94.2200	92.6341	85.4510	94.2641	92.4821	91.6927	**96.9697**
SRBCT	92.4845	88.6056	85.3544	94.5083	96.6714	**97.0965**	96.1773
Lung	87.2919	88.4598	80.7513	**91.3913**	79.3483	82.0117	88.5131
Ovarian	97.5579	**100.000**	99.4565	99.6933	99.2934	98.6302	99.1419
Arcene	76.6640	78.4428	66.6184	**82.7579**	70.6119	70.4565	78.2214
Madelon	60.5503	60.9000	**61.2519**	**61.2519**	49.6500	49.6500	60.5629
Average	75.9704	85.5516	81.8236	87.1540	83.6333	81.4947	**88.6106**
Avg. Rank	(5.73)	(2.91)	(4.68)	(2.68)	(4.05)	(5.32)	(**2.64**)
W/L/T	11/0/0	6/5/0	7/4/0	7/4/0	7/4/0	10/1/0	-
Datasets	Average recall						
	Alpha-investing	Fast-OSFS	OSFS	SAOLA	OFS-A3M	OFS-Gapknn	RHOFS
Colon	62.6136	83.6364	66.7045	85.6818	77.1591	74.3182	**88.4091**
CNS	55.6777	60.0733	**77.6557**	73.0769	76.7399	62.2711	74.5421
Leuk	76.0426	98.0000	**100.000**	95.8723	97.9234	93.0085	96.9362
Leuk_3c	77.4152	95.4191	90.9981	93.6647	**95.9637**	92.0292	95.7439
Leuk_4c	47.4154	79.7097	85.3279	76.4985	77.1528	78.5495	**91.7058**
MLL	94.5635	92.4206	84.6429	94.3651	92.3849	91.3929	**97.6190**
SRBCT	90.2036	89.4476	85.8178	91.8147	95.4980	94.9150	**95.8551**
Lung	78.3294	75.6157	78.6184	81.3020	87.7652	82.5312	**90.6898**
Ovarian	98.2160	**100.000**	99.6914	99.4505	99.3376	99.0154	99.1419
Arcene	76.6640	78.8555	66.3149	**83.1169**	70.7792	70.4424	78.6323
Madelon	60.5500	60.9000	**61.2500**	**61.2500**	49.6500	49.6500	60.5500
Average	74.3355	83.0980	81.5474	85.0994	83.6685	80.7385	**88.1659**
Avg. Rank	(5.77)	(3.64)	(4.14)	(3.41)	(3.59)	(5.05)	(**2.41**)
W/L/T	10/0/1	7/4/0	7/4/0	8/3/0	7/4/0	11/0/0	-

TABLE 7: Average classification performance of seven algorithms on eleven datasets with JRip.

Datasets	Average accuracy						
	Alpha-investing	Fast-OSFS	OSFS	SAOLA	OFS-A3M	OFS-Gapknn	RHOFS
Colon	67.7419	75.8065	69.3548	**83.8710**	69.3548	71.9355	79.0323
CNS	60.0000	**65.0000**	63.3333	58.3333	63.3333	64.1667	61.6667
Leuk	65.2778	93.0556	**95.8333**	94.4444	89.4444	86.8056	94.4444
Leuk_3c	84.7222	87.5000	91.6667	84.7222	87.2222	89.7222	**94.1666**
Leuk_4c	55.5556	86.1111	88.8889	84.7222	78.4722	78.8889	**90.2778**
MLL	80.5556	81.9444	76.3889	68.0556	85.2778	80.9723	**95.8333**
SRBCT	71.0843	74.6988	83.1325	75.9036	**87.1084**	86.8675	85.0602
Lung	79.8030	81.2808	81.2808	80.7882	88.6207	85.6158	**93.9409**
Ovarian	**98.8142**	98.4190	98.0237	97.6285	97.5099	96.7194	**98.8142**
Arcene	67.5000	72.5000	71.5000	71.5000	**74.7000**	73.2000	74.5000
Madelon	61.9500	62.1000	61.7500	61.7500	49.8000	49.8000	**66.7500**
Average	72.0913	79.8560	80.1048	78.3381	79.1676	78.6085	**84.9533**
Avg. Rank	(5.82)	(3.41)	(3.86)	(4.91)	(3.95)	(4.14)	(**1.91**)
W/L/T	10/0/1	10/1/0	9/2/0	9/1/1	8/3/0	9/2/0	-
Datasets	Average precision						
	Alpha-investing	Fast-OSFS	OSFS	SAOLA	OFS-A3M	OFS-Gapknn	RHOFS
Colon	64.4048	73.5772	66.1702	**83.7121**	66.1567	69.1569	77.5398
CNS	52.2095	**60.2273**	58.4131	53.7500	59.0501	55.8668	52.1609
Leuk	61.0000	92.7083	95.8333	93.8723	88.9256	86.2703	**96.0784**
Leuk_3c	86.2411	85.0838	88.8649	83.8624	86.7780	88.7339	**94.5881**
Leuk_4c	50.7895	87.1358	**93.5421**	85.9994	73.6629	71.8028	90.6583
MLL	80.3199	81.0458	77.2139	65.6854	85.4745	81.5546	**96.7742**
SRBCT	72.4459	75.3869	82.3990	73.2939	87.4735	**88.9116**	85.8327
Lung	67.0588	58.3874	66.9943	68.0133	82.8165	79.8544	**93.8823**
Ovarian	98.6025	98.2838	97.7484	97.4257	97.3839	96.5944	**98.7167**
Arcene	66.9727	72.0934	71.0953	71.0895	**74.4196**	72.9440	72.4377
Madelon	61.9976	62.1214	61.7613	61.7613	49.7986	49.7986	**66.7502**
Average	69.2766	76.9137	78.1851	76.2241	77.4491	76.4989	**84.1290**
Avg. Rank	(5.55)	(3.82)	(3.86)	(4.77)	(3.77)	(4.14)	(**2.09**)
W/L/T	10/1/0	10/1/0	9/2/0	9/2/0	8/3/0	8/3/0	-
Datasets	Average recall						
	Alpha-investing	Fast-OSFS	OSFS	SAOLA	OFS-A3M	OFS-Gapknn	RHOFS
Colon	63.7500	73.0682	62.9545	**80.3409**	65.0000	68.5341	75.5682
CNS	51.6484	**58.7912**	57.5092	53.6630	58.6081	57.6007	53.0952
Leuk	60.2979	91.8723	**94.9362**	93.8723	87.5149	84.7447	92.0000
Leuk_3c	73.9376	85.0838	**90.9981**	81.9610	83.9146	87.8916	90.5984
Leuk_4c	54.2450	80.4041	81.7199	79.2137	67.6190	66.5680	**83.5156**
MLL	80.8333	80.3968	75.2381	65.2381	84.2183	80.0119	**95.2778**
SRBCT	71.0331	70.2149	79.6294	70.6632	**87.1016**	85.4757	82.2522
Lung	64.3141	53.7369	59.1382	60.3834	80.1244	75.0138	**88.5658**
Ovarian	**98.8333**	98.2838	97.9752	97.4257	97.2127	96.2824	98.7129
Arcene	66.5990	72.0373	71.1445	70.7792	**74.6226**	73.0398	72.2687
Madelon	61.9500	62.1000	61.7500	61.7500	49.8000	49.8000	**66.7500**
Average	67.9492	75.0899	75.7267	74.1173	75.9760	74.9966	**81.6913**
Avg. Rank	(5.18)	(3.82)	(4.05)	(4.68)	(3.68)	(4.32)	(**2.27**)
W/L/T	10/1/0	10/1/0	8/3/0	8/3/0	8/3/0	8/3/0	-

is incapable of considering the remaining features even if they can significantly improve the performance. Compared with Fast-OSFS, RHOFS outperforms it on more than half of the datasets using three metrics and three classifiers. For example, the average accuracy, precision and recall of RHOFS have an increase of about 12% on *CNS* under Naive Bayes, 14% on *MLL* under JRip and 10% on *Arcene* under J48, respectively. Regarding to OSFS, the average precision of RHOFS is 15.7381%, 14.1492% greater than that of OSFS on *Colon* and *Leukemia_4c* with Naive Bayes. And the accuracy of RHOFS has an increase of 12.6601% and 19.4444% on *Lung* and *MLL* with JRip. Moreover, we observe from Table 9 that RHOFS is significantly better than Fast-OSFS under the JRip and J48 classifiers, and better than OSFS under the Naive Bayes classifier. The reason is that both Fast-OSFS and OSFS algorithms consider feature individually based on the relevance with the output class, while RHOFS evaluates the candidate feature with respect to the already selected features and the relevance with the output class. Furthermore, RHOFS selects most effective features by maximizing not only the discrimination ability in terms of the dependency criterion but also the class separability of boundary region.

TABLE 8: Average classification performance of seven algorithms on eleven datasets with J48.

Datasets	Average accuracy						
	Alpha-investing	Fast-OSFS	OSFS	SAOLA	OFS-A3M	OFS-Gapknn	RHOFS
Colon	67.7419	77.4194	62.9032	80.6452	80.6452	75.8065	**83.8710**
CNS	55.0000	61.6667	**73.3333**	65.0000	70.0000	64.1667	68.8333
Leuk	63.8889	90.2778	**93.0556**	91.6667	87.9167	85.8333	87.5000
Leuk_3c	84.7222	86.1111	**93.0556**	88.8889	86.2500	89.7222	90.8334
Leuk_4c	50.0000	86.1111	87.5000	80.5556	78.7500	78.1945	**89.8611**
MLL	88.8889	84.7222	86.1111	80.5556	86.9445	85.1389	**91.6667**
SRBCT	69.8795	79.5181	81.9277	85.5422	**89.6386**	89.0361	88.5542
Lung	82.2660	81.2808	79.8030	80.7882	87.1429	84.9754	**93.4483**
Ovarian	97.6285	98.4190	98.4190	97.6285	97.7866	96.8775	**98.5352**
Arcene	71.5000	71.0000	69.5000	72.5000	78.8000	75.6500	**81.5000**
Madelon	61.2500	61.4500	**61.8500**	**61.8500**	50.0000	50.0000	60.6000
Average	72.0696	79.8160	80.6780	80.5110	81.2613	79.5819	**85.0185**
Avg. Rank	(5.59)	(4.59)	(3.55)	(4.05)	(3.36)	(4.68)	(**2.18**)
W/L/T	10/1/0	9/2/0	7/4/0	9/2/0	8/3/0	10/1/0	-

Datasets	Average precision						
	Alpha-investing	Fast-OSFS	OSFS	SAOLA	OFS-A3M	OFS-Gapknn	RHOFS
Colon	66.4916	75.4386	58.5679	78.8636	79.1667	73.8833	**83.7121**
CNS	37.0192	59.8416	**73.8598**	61.2500	67.0330	60.7711	65.5128
Leuk	59.1036	88.7987	**92.7083**	90.3704	86.8576	84.5942	85.8766
Leuk_3c	82.5385	80.1451	**91.6667**	86.6306	82.6974	88.1649	90.6934
Leuk_4c	47.5245	82.3188	**85.2930**	77.5758	66.9445	66.5247	84.3608
MLL	88.7566	83.6131	85.6732	79.2063	86.9839	84.8239	**91.9141**
SRBCT	71.1087	78.2594	76.8961	82.6221	**89.0699**	88.4373	87.6728
Lung	64.7652	66.3670	62.9013	64.9240	74.4192	76.4313	**90.6972**
Ovarian	97.0520	98.0746	98.0746	97.2245	97.5452	96.7841	**98.3440**
Arcene	71.1225	70.8884	69.4419	72.1075	78.5687	75.3362	**82.1849**
Madelon	61.3398	61.4657	**61.8943**	**61.8943**	NaN	NaN	60.6021
Average	67.8929	76.8374	77.9070	77.5154	80.9286	79.5751	**83.7792**
Avg. Rank	(5.73)	(4.50)	(3.55)	(3.95)	(3.41)	(4.59)	(**2.27**)
W/L/T	10/1/0	9/2/0	6/5/0	9/2/0	7/4/0	9/2/0	-

Datasets	Average recall						
	Alpha-investing	Fast-OSFS	OSFS	SAOLA	OFS-A3M	OFS-Gapknn	RHOFS
Colon	67.8409	76.3636	57.9545	78.8636	77.8409	73.9886	**80.3409**
CNS	43.4066	60.6227	**76.1905**	60.9890	67.0330	60.2381	64.9267
Leuk	58.2979	90.6809	**91.8723**	91.7447	86.4383	85.1234	87.6170
Leuk_3c	80.5029	79.0097	**92.3314**	88.7875	83.1727	88.6850	88.2195
Leuk_4c	55.0856	85.9962	**86.1216**	84.4194	66.8625	65.1431	79.3228
MLL	89.1270	83.4524	84.8413	79.2063	86.6349	84.6548	**90.4762**
SRBCT	71.3626	79.0300	75.6951	82.1562	**89.7227**	87.0816	88.4037
Lung	62.2555	59.2230	63.1972	61.8606	72.5785	75.7580	**92.7722**
Ovarian	97.9073	98.5246	98.5246	97.6665	97.6696	96.4540	**98.6789**
Arcene	70.6575	71.1851	69.7240	72.1591	78.6364	75.2881	**81.7857**
Madelon	61.2500	61.4500	**61.8500**	**61.8500**	50.0000	50.0000	60.6000
Average	68.8813	76.8671	78.0275	78.1548	77.8718	76.5832	**83.0131**
Avg. Rank	(5.55)	(4.50)	(3.27)	(3.73)	(3.73)	(4.77)	(**2.45**)
W/L/T	10/1/0	8/3/0	6/5/0	7/4/0	9/2/0	10/1/0	-

TABLE 9: Wilcoxon signed-rank test of RHOFS against other algorithms on three classifiers.

Classifiers	Average accuracy					
	Alpha-investing	Fast-OSFS	OSFS	SAOLA	OFS-A3M	OFS-Gapknn
Naive Bayes	0.0020	0.0645	0.0186	0.0674	0.0273	0.0010
JRip	0.0020	0.0098	0.0146	0.0137	0.0244	0.0137
J48	0.0020	0.0098	0.1533	0.0322	0.0244	0.0020

Classifiers	Average precision					
	Alpha-investing	Fast-OSFS	OSFS	SAOLA	OFS-A3M	OFS-Gapknn
Naive Bayes	0.0010	0.2783	0.0420	0.2402	0.0674	0.0029
JRip	0.0020	0.0186	0.0674	0.0420	0.0322	0.0244
J48	0.0020	0.0137	0.3652	0.0186	0.0645	0.0039

Classifiers	Average recall					
	Alpha-investing	Fast-OSFS	OSFS	SAOLA	OFS-A3M	OFS-Gapknn
Naive Bayes	0.0020	0.0830	0.0420	0.0674	0.0674	0.0010
JRip	0.0020	0.0244	0.1016	0.0537	0.0537	0.0410
J48	0.0020	0.0537	0.3652	0.1748	0.0186	0.0020

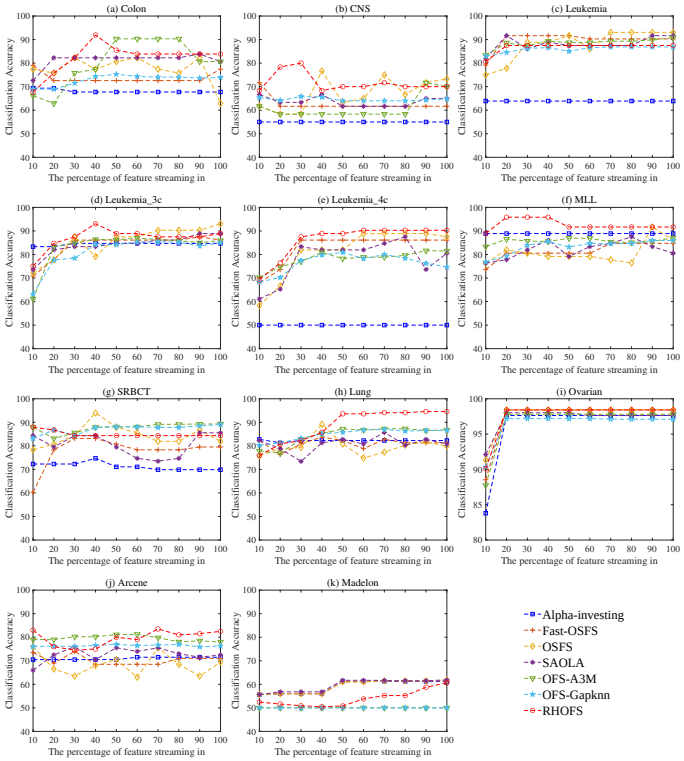

Fig. 9: Average classification accuracy of the selected feature subsets during features streaming with J48.

Although Fast-OSFS and OSFS achieve better compactness, they might be more inclined to ignore many important and relevant features and lead to worse prediction performance. Compared with SAOLA, RHOFS has higher classification accuracy, precision and recall than SAOLA on more than two-thirds of the datasets with Naive Bayes, JRip and J48 classifiers, respectively. Combining the results shown in Tables 5 and 9, we can find that RHOFS not only selects more compact feature subsets, but also is significantly better than SAOLA in classification performance in terms of accuracy and precision. Unlike RHOFS, which performs feature selection in the manner of subset evaluation, SAOLA employs the pairwise comparison technique and can only extract the relationship between the features in two feature layers, which lead to the classification performance of SAOLA is inferior to that of RHOFS, especially on datasets with a large number of features, such as the *MLL* and *Lung* datasets shown in Tables 7-8. Regarding to OFS-A3M, the proposed RHOFS algorithm has superior prediction performance on about two-thirds of the small datasets for three classifiers. And the sign 'NaN' in Table 8 means that the objects of

one class are all misclassified into the other class in *Madelon* using the selected features by OFS-A3M and OFS-Gapknn. Table 9 also shows that there is a significantly difference between RHOFS and OFS-A3M in terms of average accuracy on three classifiers, average precision on JRip, and average recall on J48. The main reason is that our RHOFS algorithm is capable of making full use of the class separability information contained in boundary regions and selecting more discriminative features. Compared with OFS-Gapknn, RHOFS has a slightly lower classification accuracy only for *CNS*, *SRBCT* with JRip and *SRBCT* with J48, but is superior than OFS-Gapknn on the other datasets for three classifiers. Moreover, the precision and recall of RHOFS are superior to that of OFS-Gapknn on at least ten, eight and nine datasets with Naive Bayes, JRip and J48. As shown in Table 9, RHOFS is significantly better than OFS-Gapknn in terms of three classification metrics on three classifiers. One reason for the superiority of RHOFS is that the hypercuboid approach evaluates feature through a supervised granulation process that makes full use of class information of objects, while the granulation of a universe in OFS-A3M and OFS-Gapknn only involve grouping similar objects with respect to the neighborhood distance between objects without utilizing class information. Another reason is that RHOFS is capable of capturing the class separability information contained in boundary regions, while OFS-A3M and OFS-Gapknn do not aim to exploit the useful implicit patterns derived from the boundary regions, and enforce only explicit patterns contained in the positive region on evaluating the quality of features. Figs. 8 and 9 further depict the average classification accuracy of the achieved feature subsets during the online selection using JRip and J48. An excellent online streaming feature selection algorithm should maintain accurate classification performance at each time instance. A common observation from all subgraphs in Figs. 8 and 9 is that our RHOFS algorithm has the highest classification accuracy at almost every time instance, and the average classification accuracy of RHOFS increases gradually by the arrival of features on most of the datasets, which implies that RHOFS can be considered to be an effective and dependable choice for online feature selection in streaming scenarios.

Table 10 further reports the average running time for each comparative algorithm on eleven datasets. It is obvious that the running times of Alpha-investing, Fast-OSFS, OSFS and SAOLA algorithms are shorter than that of OFS-A3M, OFS-Gapknn and RHOFS algorithms. This is mainly because the first four algorithms mostly perform statistical comparison methods with low time complexity to calculate the relevance between features. Moreover, Alpha-investing only considers each feature once, while ignoring the redundancy among selected features. Hence, Alpha-investing has the shortest running time, but achieves the worst performance in terms of compactness and classification accuracy. On the other hand, it can be seen that RHOFS is very competitive with OFS-A3M and OFS-Gapknn algorithms, especially on the Madelon dataset which has a relatively large number of objects. This is because the computation of dependency of the new arrival feature has the time complexity $O(n^2)$, while the time taken for this computation is $O(nc)$ in RHOFS. Thus, in summary, RHOFS could obtain the most highly discriminative features with a reasonably computational complexity, so as to favor a good trade-off between the accuracy and efficiency.

TABLE 10: Runtime comparisons of the seven algorithms on eleven datasets in seconds.

Datasets	Alpha-investing	Fast-OSFS	OSFS	SAOLA	OFS-A3M	OFS-Gapknn	RHOFS
Colon	**0.0362**	0.0729	0.1227	0.0861	0.4952	0.9946	1.7983
CNS	**0.1976**	0.2283	0.2325	0.2543	1.5500	3.1295	10.4072
Leuk	**0.2608**	0.3349	1.2660	0.5090	2.2078	4.5194	2.1072
Leuk_3c	**0.2306**	0.3288	0.9395	0.4646	2.3968	4.2829	2.1077
Leuk_4c	**0.2596**	0.3207	0.8437	0.5168	2.6549	4.2431	2.2360
MLL	**0.5296**	0.8553	7.7295	2.2832	4.0199	6.9363	3.1045
SRBCT	**0.0937**	0.1431	0.6670	0.1826	0.7478	0.7572	0.4393
Lung	**1.3857**	1.8376	44.2079	1.4328	22.4560	54.2826	11.1259
Ovarian	3.5089	**1.1920**	9.1047	1.6031	36.1604	43.5656	37.7735
Arcene	**0.6932**	0.7789	2.6087	0.9709	14.5246	39.1039	54.5104
Madelon	**0.0261**	0.0354	0.0473	0.0336	124.5529	83.6188	1.9189
Average	0.6565	**0.5571**	6.1609	0.7579	19.2515	22.3122	11.5935

6.3 Parallel Performance Analysis

In this section, we evaluate the time efficiency and scalability of the proposed distributed online feature selection algorithm (RHDOFS), so as to identify the improvement of computational performance in comparison with the original sequential RHOFS algorithm.

In order to verify the time efficiency of the RHDOFS algorithm, we conduct the experiments on a Spark cluster with six computing nodes and compared with the sequential RHOFS algorithm on four large-scale datasets. Fig. 10 depicts the running time of RHDOFS and RHOFS for streaming feature scenario, in which RHOFS is terminated at the 409th feature on *Isolet-2000*, the 1,867th feature on *Epsilon-10*, and the 710th feature on *Epsilon-20* because of the limited available memory of the local computer. As shown in Fig. 10, the running time of RHDOFS is much less than RHOFS on four large-scale datasets, and the superiority of RHDOFS becomes more and more obvious by the continuous arrival of features. For RHOFS, we can find that the time efficiency at each time a feature is arrived decreased gradually with streaming features. This is because the constantly selected features increases the computational and storage burden on the local computer. While RHDOFS is able to fully leverage the distributed computing and storage capacities of the Spark cluster, its running time at each time a feature is arrived is relatively stable during online selection, which is 0.96, 1.69, 0.96 and 1.85 seconds on average for datasets *Isolet-1000*, *Isolet-2000*, *Epsilon-10* and *Epsilon-20*, respectively. In addition, RHDOFS can reduce the running time of the sequential RHOFS algorithm by 81.7405% 93.3056% 93.5666% and 95.2354% on each of these four large-scale datasets.

The parallel performance of RHDOFS are evaluated in terms of speedup, scaleup and sizeup metrics. The speedup metric can be measured by increasing the number of computing nodes, while keeping the size of the dataset unchanged. Let $\mathbf{t}(\mathbf{1},\mathbf{1})$ be the running time to complete a single work unit with $\mathbf{1}$ computing node, $\mathbf{t}(\mathbf{1},\mathbf{N})$ be the running time to complete the same unit of work with $\mathbf{N}$ computing node, the speedup is given as: $\mathbf{t}(\mathbf{1},\mathbf{1})/\mathbf{t}(\mathbf{1},\mathbf{N}) * 100\%$. Hence, an ideal speedup of a parallel algorithm is linear,

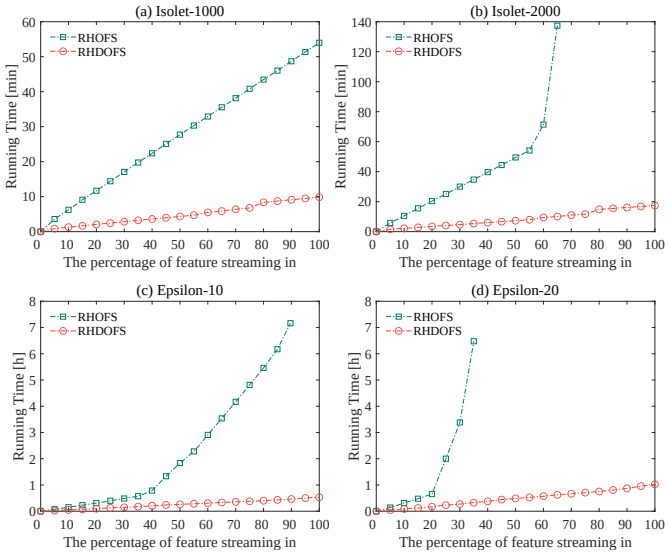

Fig. 10: Running time of RHDOFS during features streaming on four large-scale datasets (in second).

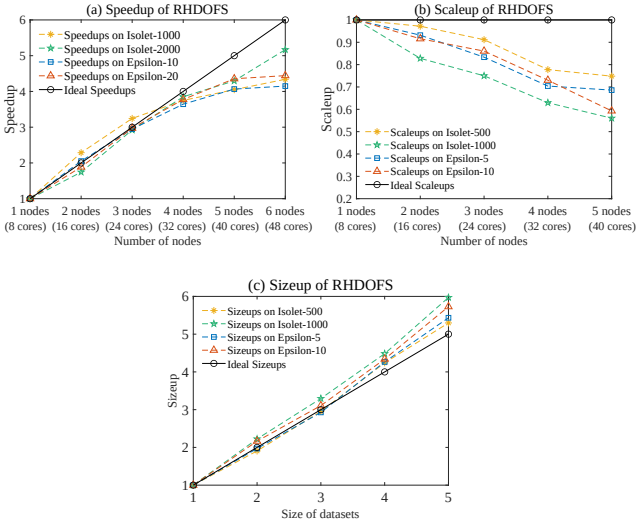

Fig. 11: Speedups, scaleups and sizeups of RHDOFS with different number of computing nodes (cores) and size of datasets.

the closer speedup is to N, the better parallel scalability the algorithm appears. Fig. 11(a) shows the speedup results of RHDOFS over the original sequential implementation with different number of nodes ranging from one to six, where the black line with circles indicates the ideal speedup.

It can be seen from Fig. 11(a) that our RHDOFS algorithm scaling well in number of computing nodes for Spark, and the speedup of RHDOFS increases in an approximated linear manner with the increase of the number of computing nodes. The increased time consuming for program configuration, data communication and task assignment lead to a decline in the growth rate of speedup. Specially, the speedup of RHDOFS is very close to the linear speedup when the number of nodes is less than or equal to 4 (32 cores). Although the growth rate of the speedups has slowed down when the number of nodes is greater than 4, the speedup value of RHDOFS on *Isolet-2000* with more than 15 million objects reaches 5.17, which illustrate that RHDOFS owns good speedup and is capable of handling very large datasets.

The scaleup metric is evaluated by increasing the number of computing nodes and the size of datasets proportionally. Let $\mathbf{t}(\mathbf{N},\mathbf{N})$ be the running time to complete $\mathbf{N}$ of the single work unit with $\mathbf{N}$ computing nodes, the scaleup is given as: $\mathbf{t}(\mathbf{1},\mathbf{1})/\mathbf{t}(\mathbf{N},\mathbf{N}) * 100\%$. In analyzing the scaleup of RHDOFS, the 500 and 1,000 times of *Isolet* dataset, the 5 and 10 times of *Epsilon* dataset are taken as the baseline reading of the Spark cluster. And the baseline reading of datasets and one node has been expanded from 1 time to 5 times simultaneously. The scaleup results of RHDOFS with the different sizes of datasets and clusters are depicted in Fig. 11(b), where the higher the scaleup, the better the scalability offers.

It can be observed from Fig. 11(b) that the scaleup decreases gradually by increasing the number of computing nodes and the size of dataset simultaneously. This because the cost of data transmission and communication in the network and other overhead of clusters become greater when the number of nodes and the size of datasets increased gradually. However, we can find that the scaleups of RHDOFS on *Isolet-2500* with about 19.5 million objects and *Epsilon-25* with 12.5 million objects are still around 0.7 even when the number of nodes is 5 (40 cores). And the scaleups are still greater than 0.55 on *Isolet-5000* and *Epsilon-50* datasets which have about 39 and 25 million objects, respectively. Hence, the results confirm that our RHDOFS algorithm exhibiting high scalability for different sizes of dataset.

The sizeup metric is measured by increasing the size of data while keeping the number of computing nodes unchanged. Let $\mathbf{t}(\mathbf{N},\mathbf{1})$ be the running time to complete $\mathbf{N}$ times of the single work unit using one parallel system with fixed computing nodes, the sizeup is given as: $\mathbf{t}(\mathbf{N},\mathbf{1})/\mathbf{t}(\mathbf{1},\mathbf{1}) * 100\%$. Similar to the setting of the scaleup experiment, the sizup is tested by taking the 500 and 1,000 times of *Isolet* dataset, the 5 and 10 times of *Epsilon* dataset as the baseline datasets. These datasets are expanded from 1 time to 5 times while the number of nodes are fixed to 5. The sizeup results of RHDOFS with the different sizes of datasets are depicted in Fig. 11(c), where the ideal sizeup of a parallel algorithm is linear, the closer sizeup is to N, the better parallel performance the algorithm owns.

It can be perceived from Fig. 11(c) that the sizeup results are still very close to the linear values, when the datasets are expanded to 4 times. And the larger the size of datasets, the farther the deviation from the linear sizups, which is understandable since more intermediate results need to be stored as the datasets grow, some will overflow from memory to disk, and increase the burden of the storage and reading of intermediate data. Despite all this, the maximum sizeup is 5.97 when *Isolet-1000* dataset is expanded to 5 times and have about 39 million objects. In a word, the results show that RHDOFS exhibits a high extensibility with the increase of the data size.

7 Conclusion

In this paper, we explored the issue of online feature selection in the distributed cloud computing context. Considering the useful information in boundary regions is ignored in the standard rough hypercuboid approach, a distance metric was employed to examine the class separability of objects that belong to the boundary region. A novel integrated feature evaluation criterion is then proposed to select more discriminative features by gathering discrimination knowledge from both positive and boundary regions. An effective online feature selection algorithm is developed for streaming feature scenarios by introducing the proposed evaluation criterion into the online relevance and redundancy analysis. Spark implementation that parallelizes the online selection of streaming features was further designed, which is able to efficiently exploit computing resources in the distributed clusters and is capable of processing large-scale datasets in a much more time-efficient manner. To show the effectiveness and superiority of the proposed novel online feature selection method, comparisons were conducted with many state-of-the-arts by using a series of benchmark real datasets in terms of the cardinality of feature subset, average predictive accuracy, running time as well as the classification accuracy during the features streaming. Several large-scale datasets were also used to evaluate the efficiency and parallel performance of the distributed version, and the experimental results show that our parallelization not only exhibits highly efficient, but also scales very well with the data size and cluster size. Future work could extend our algorithms to other online learning settings, e.g., online feature selection with streaming samples by considering online sliding window techniques. Besides, tuning performance of Spark to further improve the computational efficiency will be another topic for future research.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Nos. 62076171, 61573292, 61976182), the Key Program of National Natural Science Foundation of China (No. 61836006), and the Natural Science Foundation of Sichuan Province (2022NSFSC0898).

References

[1] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Comput. Surv.*, vol. 50, no. 6, pp. 94:1–94:45, 2018.

[2] H. Peng, F. Long, and C. H. Q. Ding, "Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1226–1238, 2005.

[3] J. C. Ang, A. Mirzal, H. Haron, and H. N. A. Hamed, "Supervised, unsupervised, and semi-supervised feature selection: A review on gene selection," *IEEE ACM Trans. Comput. Biol. Bioinform.*, vol. 13, no. 5, pp. 971–989, 2016.

[4] H. Chen, T. Li, C. Luo, S. Horng, and G. Wang, "A rough set-based method for updating decision rules on attribute values coarsening and refining," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 12, pp. 2886–2899, 2014.

[5] S. C. H. Hoi, J. Wang, P. Zhao, and R. Jin, "Online feature selection for mining big data," in *Proceedings of BigMine 2012*, 2012, pp. 93–100.

[6] H. Chen, T. Li, C. Luo, S. Horng, and G. Wang, "A decision-theoretic rough set approach for dynamic data mining," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 6, pp. 1958–1970, 2015.

[7] S. Perkins and J. Theiler, "Online feature selection using grafting," in *Proceedings of ICML 2003*, 2003, pp. 592–599.

[8] J. Zhou, D. P. Foster, R. A. Stine, and L. H. Ungar, "Streamwise feature selection," *J. Mach. Learn. Res.*, vol. 7, pp. 1861–1885, 2006.

[9] X. Wu, K. Yu, H. Wang, and W. Ding, "Online streaming feature selection," in *Proceedings of ICML 2010*, 2010, pp. 1159–1166.

[10] X. Wu, K. Yu, W. Ding, H. Wang, and X. Zhu, "Online feature selection with streaming features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 5, pp. 1178–1192, 2013.

[11] K. Yu, X. Wu, W. Ding, and J. Pei, "Scalable and accurate online feature selection for big data," *ACM Trans. Knowl. Discov. Data*, vol. 11, no. 2, pp. 16:1–16:39, 2016.

[12] P. Zhou, X. Hu, P. Li, and X. Wu, "Online streaming feature selection using adapted neighborhood rough set," *Inf. Sci.*, vol. 481, pp. 258–279, 2019.

[13] S. Li, K. Zhang, Y. Li, S. Wang, and S. Zhang, "Online streaming feature selection based on neighborhood rough set," *Appl. Soft Comput.*, vol. 113, pp. 273–287, 2021.

[14] J. Zhang, J. Wong, Y. Pan, and T. Li, "A parallel matrix-based method for computing approximations in incomplete information systems," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 2, pp. 326–339, 2015.

[15] J. Qian, P. Lv, X. Yue, C. Liu, and Z. Jing, "Hierarchical attribute reduction algorithms for big data using mapreduce," *Knowl. Based Syst.*, vol. 73, pp. 18 – 31, 2015.

[16] Q. Hu, L. Zhang, Y. Zhou, and W. Pedrycz, "Large-scale multimodality attribute reduction with multi-kernel fuzzy rough sets," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 1, pp. 226–238, 2018.

[17] H. Chen, T. Li, Y. Cai, C. Luo, and H. Fujita, "Parallel attribute reduction in dominance-based neighborhood rough set," *Inf. Sci.*, vol. 373, pp. 351 – 368, 2016.

[18] Z. C. Dagdia, C. Zarges, G. Beck, and M. Lebbah, "A scalable and effective rough set theory-based approach for big data pre-processing," *Knowl. Inf. Syst.*, vol. 62, no. 8, pp. 3321–3386, 2020.

[19] L. Kong, W. Qu, J. Yu, H. Zuo, G. Chen, F. Xiong, S. Pan, S. Lin, and M. Qiu, "Distributed feature selection for big data using fuzzy rough sets," *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 5, pp. 846–857, 2020.

[20] Z. C. Dagdia, C. Zarges, G. Beck, H. Azzag, and M. Lebbah, "A distributed rough set theory algorithm based on locality sensitive hashing for an efficient big data pre-processing," in *Proceedings of IEEE BigData 2018*, 2018, pp. 2597–2606.

[21] Y. Yao and J. Yang, "Granular rough sets and granular shadowed sets: Three-way approximations in pawlak approximation spaces," *Int. J. Approx. Reason.*, vol. 142, pp. 231–247, 2022.
[22] S. H. Nguyen and A. Skowron, "Quantization of real value attributes - rough set and boolean reasoning approach," in *Proceedings. of ICIS 1995*, 1995, pp. 34–37.
[23] R. Jensen and Q. Shen, "Fuzzy-rough sets assisted attribute selection," *IEEE Trans. Fuzzy Syst.*, vol. 15, no. 1, pp. 73–89, 2007.
[24] Q. Hu, D. Yu, J. Liu, and C. Wu, "Neighborhood rough set based heterogeneous feature subset selection," *Inf. Sci.*, vol. 178, no. 18, pp. 3577–3594, 2008.
[25] P. Maji, "A rough hypercuboid approach for feature selection in approximation spaces," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 16–29, 2014.
[26] Q. Shen and A. Chouchoulas, "A rough-fuzzy approach for generating classification rules," *Pattern Recognit.*, vol. 35, no. 11, pp. 2425–2438, 2002.
[27] S. Eskandari and M. M. Javidi, "Online streaming feature selection using rough sets," *Int. J. Approx. Reason.*, vol. 69, pp. 35–57, 2016.
[28] J. Liu, Y. Lin, Y. Li, W. Weng, and S. Wu, "Online multi-label streaming feature selection based on neighborhood rough set," *Pattern Recognit.*, vol. 84, pp. 273–287, 2018.
[29] P. Zhou, S. Zhao, Y. Yan, and X. Wu, "Online scalable streaming feature selection via dynamic decision," *ACM Trans. Knowl. Discov. Data*, vol. 16, no. 5, pp. 1–20, 2022.
[30] P. Zhou, P. Li, S. Zhao, and Y. Zhang, "Online early terminated streaming feature selection based on rough set theory," *Appl. Soft Comput.*, 2021. [Online]. Available: https://doi.org/10.1016/j.asoc.2021.107993
[31] P. Zhou, Y. Zhang, P. Li, and X. Wu, "General assembly framework for online streaming feature selection via rough set models," *Expert Syst. Appl.*, 2022. [Online]. Available: https://doi.org/10.1016/j.eswa.2022.117520
[32] X. Tang, Z. Huang, D. M. Eyers, S. Mills, and M. Guo, "Scalable multicore k-nn search via subspace clustering for filtering," *IEEE Trans. Parallel Distributed Syst.*, vol. 26, no. 12, pp. 3449–3460, 2015.
[33] J. Chen, K. Li, Z. Tang, K. Bilal, S. Yu, C. Weng, and K. Li, "A parallel random forest algorithm for big data in a spark cloud computing environment," *IEEE Trans. Parallel Distributed Syst.*, vol. 28, no. 4, pp. 919–933, 2017.
[34] C. Luo, S. Wang, T. Li, H. Chen, J. Lv, and Y. Zhang, "Spark rough hypercuboid approach for scalable feature selection," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 3, pp. 3130–3144, 2023.
[35] C. Luo, S. Wang, T. Li, H. Chen, J. Lv, and Y. Zhang, "Large-scale meta-heuristic feature selection based on BPSO assisted rough hypercuboid approach," *IEEE Trans. Neural Networks Learn. Syst.*, 2022. [Online]. Available: https://doi.org/10.1109/TNNLS.2022.3171614
[36] J. Wei, S. Wang, and X. Yuan, "Ensemble rough hypercuboid approach for classifying cancers," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 3, pp. 381–391, 2010.
[37] M. Hu, E. C. C. Tsang, Y. Guo, and W. Xu, "Fast and robust attribute reduction based on the separability in fuzzy decision systems," *IEEE Trans. Cybern.*, pp. 1–14, 2021.

Chuan Luo received the Ph.D. degree in Computer Science from Southwest Jiaotong University, Chengdu, China, in 2015. Currently, he is an Associate Professor with the College of Computer Science, Sichuan University, Chengdu, China. He was a Visiting Ph.D. Student with the University of Regina, Regina, SK, Canada, in 2014. In Feb. 2019, he was a Visiting Scholar with the Harvard University, Cambridge, MA, USA. He serves as an Editor of Human-Centric Intelligent Systems. His current research interests include granular computing, cloud computing, and incremental learning.

Sizhao Wang received the B.S. degree from the College of Computer Science, Sichuan University, Chengdu, China, in 2018. He is currently working toward the M.S. degree in the College of Computer Science, Sichuan University, Chengdu, China. His research interests include granular computing and cloud computing. He has published papers in several leading journals, such as IEEE Transactions on Knowledge and Data Engineering, etc.

Tianrui Li (SM'10) received the Ph.D. degree from the Southwest Jiaotong University, Chengdu, China, in 2002. He is currently a Professor with the School of Computing and Artificial Intelligence, Southwest Jiaotong University. He serves as Area Editor of International Journal of Computational Intelligence Systems, Editors of Knowledge-based Systems and Information Fusion, Associate Editor of ACM Transactions on Intelligent Systems and Technology. His research interests include big data, cloud computing, data mining, granular computing and rough sets.

Hongmei Chen received the Ph.D. degree in computer science from the Southwest Jiaotong University, Chengdu, China, in 2013. She is currently a Professor with the School of Computing and Artificial Intelligence, Southwest Jiaotong University. She is the author or coauthor of 2 books and more than 100 research papers in international journals and conferences. Her current research interests include data mining, pattern recognition, and rough sets.

Jiancheng Lv (M'09) received the Ph.D. degree in computer science and engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2006. He is currently a Professor with the College of Computer Science, Sichuan University, Chengdu. He has been an Associate Editor of the IEEE Transactions on Neural Networks and Learning Systems since 2015. His current research interests include neural networks, machine learning and big data.

Zhang Yi (M'08-SM'09-F'16) received the Ph.D. degree in mathematics from the Institute of Mathematics, Chinese Academy of Science, Beijing, China, in 1994. He is currently a Professor with the College of Computer Science, Sichuan University, Chengdu, China. He was an Associate Editor of the IEEE Transactions on Neural Networks and Learning Systems from 2009 to 2012. He has been an Associate Editor of the IEEE Transactions on Cybernetics since 2014. His current research interests include neural networks, machine learning and big data.