Electronic Preprint for Journal of Information Processing Vol.21 No.2

Regular Paper

Numerosity Reduction for Resource Constrained Learning

Khamisi Kalegele[1,a) Hideyuki Takahashi[1,2,b) Johan Sveholm[2,c) Kazuto Sasai[1,2,d) Gen Kitagata[1,2,e) Tetsuo Kinoshita[1,2,f)

Received: June 28, 2012, Accepted: January 11, 2013

Abstract: When coupling data mining (DM) and learning agents, one of the crucial challenges is the need for the Knowledge Extraction (KE) process to be lightweight enough so that even resource (e.g., memory, CPU etc.) constrained agents are able to extract knowledge. We propose the Stratified Ordered Selection (SOS) method for achieving lightweight KE using dynamic numerosity reduction of training examples. SOS allows for agents to retrieve different-sized training subsets based on available resources. The method employs ranking-based subset selection using a novel Level Order (LO) ranking scheme. We show representativeness of subsets selected using the proposed method, its noise tolerance nature and ability to preserve KE performance over different reduction levels. When compared to subset selection methods of the same category, the proposed method offers the best trade-off between cost, reduction and the ability to preserve performance.

Keywords: data reduction, agent learning, machine learning, instance ranking

1. Introduction

Data Mining (DM)-integrated agent frameworks are appropriate platforms for the realization of symbiotic integration [5] of DM and intelligent agents. The frameworks enable intelligent agents to use DM techniques to extract knowledge from training examples. In a way, the integration [5] supplements agent's deductive logic with tasks like learning a classifier and association rules. This integration, which helps to overcome the breakdown-with-complexity problem of agent's deductive reasoning logic, faces two crucial challenges. The first is the need for the knowledge extraction (KE) process to be lightweight enough so that even resource (e.g., memory, CPU etc.) constrained agents are able to extract knowledge. The second is a well designed middleware to bridge the gap between reasoning logics of DM and agents. The scope of this paper covers the first challenge. There are two approaches for achieving a lightweight KE; the use of smaller training data (TD) sets or algorithm improvement. The use of smaller examples subsets is facilitated by five categories of data reduction approaches [15]; dimensionality reduction, feature selection, numerosity reduction, data aggregation and discretization. Whichever of the approaches used, studies have shown that reducing TD sets beyond a certain level have negative effects on KE [21]. In this paper we propose a numerosity reduction method to realize the objective of achieving smaller TD subsets with either zero or minimum effects on KE. The method uses a novel Level Order (LO) ranking scheme to rank examples within classes and then stratified selections are made on these classes to form smaller subsets. By ranking examples, the method provides room for class balancing which is necessary for highly imbalanced dataset. This method's preliminary versions appear in [16], [17], [18]. In these versions, the method was not evaluated extensively. For example, only C4.5 algorithm was used as the KE algorithm. Also the critical steps of the method were not explained explicitly and sufficiently. Moreover, the representativeness of the selected smaller subsets was also not shown.

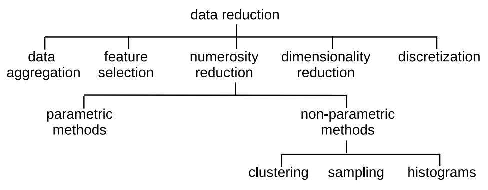


Fig. 1 Simplified taxonomy of data reduction approaches.

The five data reduction approaches are shown in **Fig. 1**. We have chosen to focus on the numerosity reduction approach because methods based on other approaches are too closely related to specific data for a generic approach [15]. Among the numerosity reduction methods, sampling-based ones are the most likely appropriate candidates for lightweight KE in mining-based agent learning due to two reasons. First, they are non-parametric and therefore incur no extra cost of (re)creation of data estimation models. Second, amongst non-parametric methods, sampling cost increases linearly with sample size while costs in other methods increase exponentially. The proposed numerosity reduction method is meant to make various machine learning and DM tasks either possible or performance-improved for resource constrained agents. These tasks include classification tasks and rule mining

[1] Graduate School of Information Sciences, Tohoku University, Sendai, Miyagi 980–8579, Japan
[2] Research Institute of Electrical Communication, Tohoku University, Sendai, Miyagi 980–8577, Japan
[a)] kalegs@k.riec.tohoku.ac.jp
[b)] hideyuki@riec.tohoku.ac.jp
[c)] jsveholm@riec.tohoku.ac.jp
[d)] kazuto@riec.tohoku.ac.jp
[e)] minatsu@riec.tohoku.ac.jp
[f)] kino@riec.tohoku.ac.jp

tasks, to mention a few. In these tasks, an agent learns models from TD examples using some learning algorithm. TD sets are sets of data in a features-label format ($\mathbf{x}_k, y_k$), where features $x_1, x_2, \ldots, x_n$ describe some past experience labeled y. The individual instances of these experiences $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ are refered to as examples and their respective labels $y_1, y_2, \ldots, y_k$ as classes. It is this process of learning knowledge models from TD which we are referring to as KE and the respective learning algorithms as KE algorithms. Environments which involve resource constrained agents and the tasks, include wireless sensor networks and smart homes and devices.

Section 2 of this paper covers related works and the position of the proposed method. In Section 3, we summarize how mining-based agent learning have been approached in existing multi-agent frameworks. Our proposed method is presented in Section 4 and evaluated in Section 5.

2. Data Reduction and Subset Selection

Sampling-based reduction approaches are characterized by the following properties [4], [12].

(1) Selection method: A method for selecting examples when forming a TD subset, e.g., random selection.

(2) Selection type: This tells whether a method seeks to retain boundary examples in a TD set (condensation type) or central examples (edition type) or both (hybrid).

(3) Fitness function or test: A criteria which the selected examples are supposed to meet.

(4) Search direction: Incremental or decremental subset formation or representatives identification.

(5) Stratification: When examples are independently selected from individual strata or classes of examples, then the selection is said to be stratified.

(6) Other properties: Related to the used KE algorithm, e.g., distance function.

Subset selection approaches are presented, discussed and compared using these properties. This paper's related works have two viewpoints. One is subset selection (or sampling) which is discussed in this section together with benchmarking choices (for evaluation) using the above properties. Another one is DM-integrated agent framework approaches discussed in Section 3.

2.1 Sampling-based Selection

Among the three non-parametric categories in Fig. 1, sampling has the advantage that the cost of obtaining a subset sample is proportional to the subset size as opposed to the TD set size in other methods. Other non-parametric methods require at least one complete pass through the TD set in every selection [15]. Therefore, for the same subset size, selection complexity has a linear relationship with TD subsets' dimensions in sampling-based methods whereas in other methods (e.g., histograms) complexities increase exponentially with TD dimensions.

Table 1 Sampling-based methods and characteristics.

Characteristic	Cluster	SRS	RMHC	SOS
Selection Method	random	random	random	ordered
Selection Type	-	-	-	hybrid
Fitness Test	-	-	accuracy	-
Stratification	yes	yes	no	yes
Search Direction	increm.	increm.	increm.	increm.
Other properties	n/a	n/a	Manhattan	Minkowski

Table 1 shows three classical sampling-based methods for selecting subsets (Cluster, SRS and RMHC) plus the proposed method, Stratified Ordered Selection (SOS). The three methods are incremental and random-based. One crucial shortfall of random-based sampling methods is that they are not very reliable. There is no guarantee that they will always give performance enhancing (or preserving) subsets when applied on the same TD sets. For instance, SRS (Stratified simple Random Sample) [15] involves random selection of examples from each of the mutually disjoint strata (classes) of a TD set. While SRS ensures a representative subset like its derivatives SRSWOR (Simple Random Sample Without Replacement) and SRSWR (Simple Random Sample With Replacement), it neither can regenerate the same result nor does it explicitly attempt to retain decisive examples of a class (e.g., boundary and central). Cluster (cluster sample) is very similar to SRS. The only difference is that in cluster sample if there are C mutually disjoint strata, the examples selection can result into s clusters, where $s < C$. This is not allowed in SRS where the number of strata should always be preserved.

Methods like RMHC (Random Mutation Hill Climbing) [4], [9], [28] improve reliability by using a fitness test. In RMHC, although randomly selected, examples are only kept if they enhance or preserve accuracy. The use of fitness test during subset selection imposes extra computational cost which in mainstream learning is not a big problem because selection is only done once for a TD set (mainly for noise removal). However, in resource constrained environments where learning agents have differing resources, selection is done many times with varying subset sizes.

The proposed method, SOS, for numerosity reduction in a resource constrained environment, has two main potentials over others. It improves reliability of KE by selecting examples in an orderly manner. Another potential is that examples are ranked before selection. In terms of the above properties and in addition to using ordered selection, the proposed method does stratified selection, it is incremental and of hybrid selection type. We have summarized how the proposed method is characterized by calling it the Stratified Ordered Selection (SOS) method and we will use the name henceforward.

SOS addresses the above-described shortfalls of other sampling methods. It uses examples ranking and ordered selection in order to improve reliability. However, the method does not employ fitness tests at selection time in order to keep the cost low. It is both a condensation and an edition type method so that the generated subsets enhance or preserve KE performance even better. To ensure representativeness of the formed subsets, the method also employs a stratification approach. SOS also provides room for on the fly class balancing. Class imbalance is a well known problem, in learning, with negative effects on KE [13], [27].

2.2 Benchmarking

Evaluation works (Refs. [12], [26]) appearing in literature reveal that RMHC and other two non sampling-based methods, AkNN (All *k*-NN) [31] and DROP3 (Decremental Reduction Optimization Procedure 3) [32], offer excellent balance between size reduction and accuracy. Both, AkNN and DROP3, build on top of ENN (Edited Nearest Neighbor rule) [31] which edits out noisy instances from datasets and leaves smoother decision boundary. AkNN extends ENN by employing a loop of *k*s and DROP3 employs a noise filtering pass similar to ENN.

Due to the expensiveness of non sampling-based methods, we have chosen (from these methods) RMHC as our candidate for benchmarking. It is the method which is the most similar to SOS and the least expensive one (among the three). Additionally, we also use SRS because of its inexpensiveness, its simplicity and its ability to generate representative subsets. Moreover, SRS, like SOS, provides room for on-the-fly class balancing. Nevertheless, for further clarity, we also conduct experiments to compare the proposed SOS with AkNN and ENN.

3. Mining-based Agent Learning

In an agent-based application, agents are either created with some initial intelligence or sent out to learn by themselves through interaction with their peers. Both, initial intelligence and agent's self-acquired intelligence can be extracted or mined from federated information repositories containing various experiences of agents [29]. Since these federated information repositories contain agents' life time experiences, they might grow tremendously in size and make it difficult for automated knowledge extraction by agents, e.g. in agent-based home automation systems [14]. Although DM-integrated agent frameworks and platforms to realize this extraction exist (e.g., ABLE [3] and Agent Academy [22]), provisionings to deal with data size in presence of noise based on agent's available resources are not available. Unfortunately smaller (agent-affordable) data size is crucial in achieving lightweight KE. In ABLE, the idea of lightweight KE is only highlighted without further details. In Agent Academy, the data miner provides few TD sets reduction functionalities which are insufficient as far as lightweight KE is concerned. Since tasks like example (instance) ranking and subset selection are not in database management system's agenda, achieving lightweight KE through the use of smaller subsets has entirely been left to manual intervention. Existing frameworks rely on SQL's abilities to manipulate database pages. These abilities, unfortunately, are only limited to the following.

(1) Simple filters: i.e. numerosity reduction
- e.g., WHERE *colA* = *value1*

(2) Simple reducers: i.e. dimensionality reduction
- e.g., *colA* + *colB* AS *colNew*

(3) Simple aggregators: i.e. data aggregation
- e.g., GROUP BY *colA*

(4) Selectors: i.e. feature selection
- e.g., SELECT *colA*, *colB*, ...

SOS advocates WHERE's role in these frameworks by making dynamic numerosity reduction possible, only that unlike WHERE, SOS offers much smarter, non-blind and performance-preserving reduction.

4. Stratified Ordered Selection

Stratified Ordered Selection (SOS) aims at enabling the retrieval of TD subsets of varying sizes based on available resources. SOS-selected subsets are representative and performance preserving. Two main concepts behind SOS are example ranking and ordered sampling. As highlighted in Table 1, ordered sampling (selection) based on example ranking differentiates the proposed SOS method from the others which either employ exhaustive fitness test (everytime an example is selected during sampling) or conduct random sampling. The concepts, with an extra functionality of class balancing as previously mentioned in Section 2.1, are together integrated into a retrieval intermediary shown in **Fig. 2**.

The retrieval manager implements sampling (or selection) functions as well as facilitating communication with the outside world (i.e., intelligent agents). During subset retrieval process, as will be described in Section 4.1, the retrieval manager selects subsets based on ordered sampling using the ranking function provided by the ranking module of the retrieval intermediary. Subset selection by ordered sampling is described in Section 4.2. Ranking function is based on a novel scheme called Level Order (LO) ranking scheme, described in Section 4.3.

4.1 Subset Retrieval Process

The retrieval intermediary implements a four-step subset retrieval process. In this paper's context, the four steps are basically negotiations about subset specifications between an agent (lightweight KE) and the retrieval manager. They are as follows:

1. Request: An agent sends its desired size and class balance to the manager.
2. Verification: The manager verifies the desired size and class balance. The manager checks whether the desired size is achievable and the amount of sampling (needed to achieve the desired size) is reasonable and (re)adjusts the specifications accordingly. The desired size is limited in such a way that no oversampling is allowed on the larger, majority class. Reasonable sampling is limited to undersampling of the majority class and oversampling of the minority to a 50:50 balance. Finally, the new or verified subset specifications are sent to an agent.

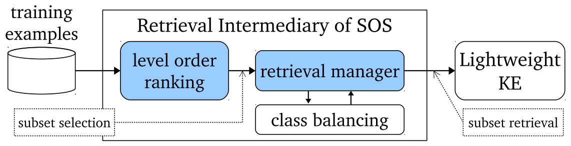


Fig. 2 Retrieval intermediary.

3. Retrieval: The manager selects and retrieves the desired subset using SOS as will be described in Section 4.2.
4. Acknowledgement: An agent acknowledges if it is satisfied with the new or verified specifications. Otherwise it goes back to step 1 above.

In step 2, verification is done using TD set information like size, class proportions and pre-defined sampling limits, explained further in the next subsection.

4.2 Subset Selection

Examples which make a subset are incrementally selected from the individual strata or classes of examples, contained in the original TD set, to meet the desired number of class quota which is necessary to achieve the desired class balance. Examples within classes or strata are pre-ranked using the LO ranking scheme (described in Section 4.3) and therefore the selection task is brought down to either ordered undersampling or synthetic oversampling of the individual classes or strata from the view point of the original TD set. Ordered undersampling basically means selecting the desired number of highly ranked examples. Synthetic oversampling is achieved by creation of synthetic examples using the SMOTE technique [6]. The process of creation of synthetic examples based on SMOTE implementation involves the following four steps.

1. An example is chosen at random from within the class to be oversampled.
2. 5 nearest examples to the chosen one (in step 1) are identified. 5 is the value used in the current implementation [6].
3. Desired number of points are selected at random along the line segments joining the randomly chosen example and any two of its nearest neighbors.
4. New examples are synthetically created at each of the selected points in step 3.

In Ref. [6], results showed that the SMOTE approach can improve the accuracy of classifiers for the minority class. The combination of SMOTE and undersampling was found to perform even better than plain undersampling. In the proposed SOS, subsets are dynamically selected by ordered (Section 4.3) undersampling of TD sets. During selection, on-the-fly class balancing is possible. This is achieved by increasing the minority class of the unbalanced subsets through synthetic oversampling using SMOTE. To avoid overfitting [20] and concept drift [11] problems, oversampling is limited to the minority class of the original TD sets.

4.3 Level Order ranking

Level Order (LO) ranking is an intra-class ranking scheme that seeks to identify representatives which broaden class representation by retaining both central and boundary examples. Representatives are selected in levels determined by recursive partitioning of the distance space of examples. In each level, except *level zero* which reflects the initial space, partitions' representatives are identified as follows.

Suppose N class examples in their distance space where I_x is the x^{th} example at a distance d_x from the centroid of the class (The centroid can be computed using simple k-means with number of classes set to one). We refer to this as *level zero* of representation where the prime central example is the closest-to the centroid ($I_{\min} : d_{\min} = \text{minimum}\{d_1, \ldots, d_N\}$) and the prime boundary example is the furthest-from the central example ($I_{\max} : d_{\max} = \text{maximum}\{d_1, \ldots, d_N\}$). This scheme considers $I_{\min}$ and $I_{\max}$ as *level zero* representatives, as shown in **Fig. 3**. In the higher levels, we want to identify the most representative example for each partition. We first considered the measure of central tendency, median, expressed by the expression below.

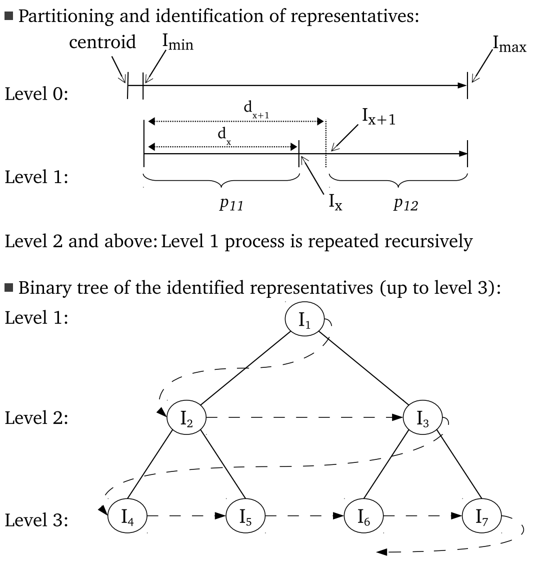

■ Partitioning and identification of representatives:
centroid
Imin
Imax
Level 0:
dx+1
Ix+1
dx
Level 1:
p11
Ix
p12
Level 2 and above: Level 1 process is repeated recursively
■ Binary tree of the identified representatives (up to level 3):
Level 1:
I1
Level 2:
I2
I3
Level 3:
I4
I5
I6
I7

Fig. 3 Level Order ranking.

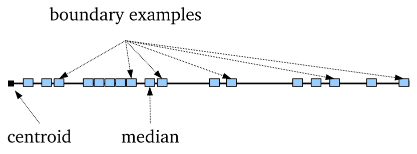


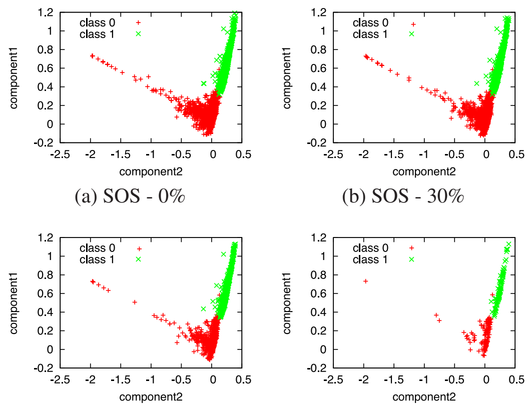
Fig. 4 Median example in a distance space.

$$\exists I_x : d_x = d_{\frac{N}{2}} \ \forall N \in even,\ d_x = d_{\frac{N+1}{2}} \ \forall N \in odd$$

However, since SOS seeks to retain boundary examples as well, and the median is not necessarily at the boundary of a cluster of examples, the median does not suffice. Consider the examples in the distance space in **Fig. 4** with the centroid at distance zero. The median is not at the boundary but an example next to it is.

$$E(I_x) = [d_{x+1} - d_x] \times \frac{\text{minimum(size of } p_{i1}\text{, size of } p_{i2})}{\text{maximum(size of } p_{i1}\text{, size of } p_{i2})} \quad (1)$$

Decision boundaries are affected by not only where examples (instances) of one class lie, but where those of other classes lie as well [32]. Because of this and the fact that it takes a large number of boundary examples to completely define a boundary, it is important that selected representatives keep decision boundary in the correct vicinity. We achieve this in two ways. One way is to ensure that partitioning does not take place through high density areas by targeting examples I_x for which $d_{x+1} - d_x$ is the largest. Second way is by targeting examples which are at the median of the partitions so that the general vicinity within the class is maintained. These are examples which divide partitions as pro-



(a) SOS - 0% (b) SOS - 30% (c) SOS - 50% (d) SOS - 90%

Fig. 5 Representativeness per reduction percentage.

portionally as possible.

We combine these two ways into a measure $E(I_x)$, Eq. (1), and use it to select representative examples during partitioning. In each partition, a representative is an example $I_{x=i}$ which maximizes $E(I_{x=i})$ and subsequently divides the distance space into partitions p_{i1} and p_{i2} as shown in Fig. 3.

To sum up, representatives are established as follows:

Level one representative: An example I_1 which maximizes the measure shown in Eq. (1) is identified as a representative which consequently divides the space into p_{11} and p_{12}.

Level two representatives and *above*: The *Level one* process is repeated on partitions p_{11} and p_{12} separately to obtain I_2 and I_3 as *level two* representatives which consequently divide their respective partitions into $\{p_{21}$ and $p_{22}\}$ and $\{p_{31}$ and $p_{32}\}$ respectively. This is then repeated recursively until all examples are seen.

With the exception of *level zero*, every time a representative is identified, it is made into a binary tree using its respective distance. The root node of the binary tree, shown in Fig. 3, is the *level one* representative. Finally, example rankings are determined according to level order traversal of the tree. The complete ranking order, shown below, is obtained by first taking *level zero* representatives and then level-order traverse the tree as shown by the dashed line in Fig. 3.

$$I_{min},\ I_{max},\ I_1,\ I_2,\ I_3,\ I_4,\ \cdots$$

We have adopted a binary tree approach for two reasons. The first reason comes from our partitioning approach in which we divide partitions into two. This partitioning follows a binary tree pattern. The second reason is the flexibility which binary tree brings during implementation. For instance, a binary tree which is implemented as a queue is less expensive than one which is implemented using functions.

As demonstrated in **Fig. 5**, the LO ranking enables retrieval intermediary to hold key boundary examples longer during subset formation (or during reduction from the view point of the original TD set). The plots in Fig. 5 show two principal components of experimental machine learning spam data subsets (DS4 in Section 5) obtained after the indicated percentage reduction of the original set using SOS. Notice examples at minimum and maximum principal component values being maintained even after 90% reduction.

The LO ranking uses the Minkowski distance (Eq. (2)). Our choice is due to the fact that two of the commonly used distance functions (Manhattan and Euclidean) are contained in it. Therefore we get more flexibility during experimentation. Also, the Minkowski assigns more weight onto features on which data objects differ most [8]. This makes identification of boundary examples easier.

$$D(\mathbf{x},\mathbf{y}) = \left[\sum_{i=1}^{n} |x_i - y_i|^p\right]^{\frac{1}{p}}, \tag{2}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ and

p *is* the order of the Minkowski metric.

The ranking style of the LO ranking scheme resembles Minimum Enclosing Ball (MEB) [2], [19], [23], [30]. However the core concepts and computational problems which each addresses, are significantly different. Most parametric MEB algorithms and their implementations compute the smallest enclosing balls of point sets in Euclidean spaces [2], [23]. As highlighted in Ref. [23], application of MEB in other spaces is still an open issue. Although, theoretically, MEB can be used to dynamically reduce numerosity of data for a generic use, it is bound to be costful and we have not come across an implementation which does so. Applications of MEB have so far been in areas like gap tolerant classifiers, clustering and tuning Support Vector Machine [19]. Non-parametric LO ranking is designed to work with any distance space and focuses on identification of representatives of level-based partitions in order to form subsets dynamically.

5. Experiments and Evaluation

Three experiments were conducted as follows.

5.1 Experiment 1

5.1.1 Overview

The objectives of this experiment are:

- To investigate the capabilities of SOS in terms of formation of subsets which preserve performance.
- To compare the proposed SOS with bechmarking methods (SRS and RMHC).

To achieve the above objectives, we used the proposed SOS and the two benchmarking methods to form different sized subsets from sets of well known datasets (in machine learning community). The formed subsets were used to learn classification models (using the WEKA environment [33]) which were then compared to baseline models, to check by how much they preserved performance, and among them to check which are best models. A numerosity reduction method (SOS or SRS or RMHC) which leads to the best models is considered more performance-preserving than the others. Overall, a method is considered better than others if it offers the best trade-off between performance preservation, numerosity reduction and cost.

Table 2 TD sets.

Name	ID	Size	Balance	No. of Features
Page-blocks	**DS1**	5472	10:90	10 (4/6/0)
Pima	**DS2**	768	35:65	8 (8/0/0)
Segment	**DS3**	2308	14:86	19 (19/0/0)
Spam	**DS4**	4597	39:61	57 (57/0/0)
Yeast	**DS5**	1484	11:89	8 (8/0/0)

In Section 5.1.2, we explain algorithms used to learn classification models and performance metrics which were used for evaluation when investigating the above objectives. In Section 5.1.3, we provide a description of the used datasets. In Section 5.1.4, we explain the employed methodology. In this Section, we describe the agent-based experimental setup. We also explain how the training sets were formed and provide the parameters which were used. We present the results and analysis in Section 5.1.5.

5.1.2 Algorithms and Evaluation Metrics

The machine learning models were built using three algorithms: C4.5 [25], Naive Bayes (NB) and Support Vector Machine (SVM) [7]. C4.5 is one of the most popular algorithms based on the concept of information gain. NB is a probabilistic classifier based on Bayes' theorem. SVM is a non-probabilistic binary linear classifier which is one of the best performing algorithms. We use Sequential Minimal Optimization (SMO) implementation of SVM by Platt [24]. Since we used C4.5, NB and SVM algorithms, classification accuracy is the most important metric for KE performance, provided that classes are not highly imbalanced. This is also argued by Ref. [26]. Nonetheless, we also investigated the F-Measure of the classifiers in order to compare them in case the classes are highly imbalanced. F-Measure is a single numerical metric commonly used to compare model performances [15], [33]. Equation (3) shows the formula for F-Measure where β is a positive real number which is used to put more emphasis on one of either recall or precision. We put equal emphasis on both recall and precision by setting $\beta = 1$. Moreover, we recorded the times which each method took to form the subsets.

$$F - Measure = \frac{(1+\beta) \times precision \times recall}{\beta^2 \times precision + recall} \tag{3}$$

These metrics were used in investigating the following:

- Baseline performance values on the original datasets.
- Optimal value of the order of the Minkowski metric (p in Eq. (2)) for use in SOS method.
- Effects of SOS on knowledge extraction (KE) in order to understand its capabilities in terms of formation of subsets and performance preservation.
- Comparison between SOS, SRS and RMHC in order to find out the method which offers the best trade-off between performance preservation, numerosity reduction and cost.

5.1.3 Datasets

Experiments and simulations were conducted on five well known machine learning datasets, shown in **Table 2**. Different datasets were used so that our evaluations are not influenced by any particular characteristics of datasets. Page-blocks (DS1) is a dataset of document page layouts. Pima (DS2) is a dataset of diabetes data. Segment (DS3) is a dataset of image segmentation data. Spam (DS4) is a dataset of spam mail filtering data. Yeast (DS5) is a dataset of cellular localization sites of proteins. These are all two-class problem sets from the repository of the University of California, Irvine [10]. There are only two classes of examples in the sets (the minority and the majority class). The minority-majority classes ratio is called class balance (indicated as "Balance" in the table). In the last column of the table, the numbers of features in the datasets are listed. In brackets, after the number of features, the numbers of real-valued, integer-valued and nominal features are included in respective order.

For investigation of dimensionality effects, datasets in Table 7 were used. The datasets have up to about 300 dimensionality.

5.1.4 Methodology and Parameters

Three agents were setup using the ABLE framework [3]. A Retrieval Intermediary Agent (RIA) implements functions of the retrieval intermediary (SOS) and the two benchmarking methods, SRS and RMHC. RIA has access to a DB which stores training and testing partitions of the TD sets. A Learning Agent (LA) simulates different levels of resource constraints by demanding specific TD sets requirements (size and class balance). Lastly, an Evaluation Agent (EA) provides evaluation services to LA. For convenience, only three agents were used. RIA comes naturally from Fig. 2. Separation into EA and LA enables LA to be dedicated to learning for better performance monitoring.

During simulation, first, RIA is initialized. Initialization involves preparation of training and testing partitions and calculations of LO rankings. Second, LA simulates the different levels of resource constraints. At each level, the following steps are followed for each set of the TD partitions.

1. LA requests TD subsets of the desired size and class balance from RIA based on simulated available resources.
2. RIA prepares training subsets using SOS, SRS or RMHC and in turn sends them to LA. For each subset sent, the corresponding testing subset is sent to EA.
3. For each subset, LA performs KE using KE algorithms under the WEKA environment and sends the learnt models to EA for testing and evaluation.
4. EA evaluates the learnt model. The rationale of evaluating learnt models outside LA is that for same pairs of models and testing data subsets, results do not depend on where evaluation is conducted. For convenience we evaluate models using EA which is dedicated to the task.

The partitions were obtained using a 10-fold cross validation procedure. A TD set is randomly divided into 10 disjoint sets of same size. In turn, each of the disjoint sets becomes a testing partition while a union of the rest of the disjoint sets becomes a training partition on which SOS, RMHC and SRS are applied to obtain the subsets. From the training partitions, 20 subsets (including 0% reduction) were formed by recursively reducing the partition by 5% of the original size. When a balance (other than

Table 3 Simulation parameters and environments.

Parameter/Environment	Values
KE algorithm	C4.5 (J48)
	Naive Bayes (NB)
	Support Vector Machines (SVM)
Training partitions	0 to 95% of the original TD sets
Model Validation	10-fold cross validation
Order of Minkowski metric	1, 2, 3, 5 and 7
Class balance	original balance and 50:50
OS	Ubuntu x86 64
Programming	JAVA 1.6
Learning	WEKA 3.7.1
Agent framework	ABLE 2.3.0

Table 4 Baseline performances on original TD sets.

	Avg. Accuracy			Avg. F-Measure		
TD set	C4.5	NB	SVM	C4.5	NB	SVM
DS1	97.13	89.74	94.13	0.971	0.897	0.932
DS2	73.70	75.45	77.47	0.735	0.750	0.764
DS3	99.31	84.30	99.65	0.993	0.862	0.996
DS4	93.24	79.22	90.85	0.932	0.794	0.907
DS5	76.01	75.00	74.37	0.754	0.743	0.685
Avg	87.87	80.74	87.29	0.877	0.809	0.857

Table 5 Average accuracy values on TD sets for p = 1, 2, 3, 5 and 7.

TD set	p=1	p=2	p=3	p=5	p=7
DS1	97.03	**97.15**	96.87	96.63	96.96
DS2	79.31	76.90	77.72	**79.66**	78.26
DS3	99.27	98.79	99.09	**99.28**	99.14
DS4	93.20	93.30	93.64	**93.65**	93.56
DS5	78.04	78.24	78.67	**79.13**	78.77

the original balance) is desired, we use the SMOTE technique to oversample the minority class. The paramaters, environments and algorithms used are all listed in **Table 3**.

5.1.5 Results and Analysis

Baseline performance:

For comparison purposes, we first calculated baseline performance values of the KE algorithms on the original TD sets. These are shown in **Table 4**. For baseline performance, models were learnt from the original unaltered sets. Note that this is different from 0% reduction. 0% reduction by a reduction method returns a maximum subset that the method can produce, which is not necessarily the same as the original unaltered set because some noise and redundances are removed.

Optimal value of the order of the Minkowski metric, p:

Table 5 summarizes our findings when C4.5 models were learnt from subsets formed by SOS using various values of the order of the Minkowski. $p = 5$ was marginally found to be an optimal value for p. Similar observations were made with NB and SVM. This value was therefore used for the rest of our simulations.

Effects of SOS on KE performance:

The performance of SOS on datasets for the three algorithms can be seen in subfigures (a), (b) and (c) of **Fig. 6**, **Fig. 7**, **Fig. 8**, **Fig. 9** and **Fig. 10**. Evidently, SOS maintains accuracy values close to the baseline values. The accuracy values are almost the same as the baseline values at least up to close to 50% reduction in most cases with few cases going up to 90%. Occasionally (e.g., Fig. 6 (b)), SOS enhances performance beyond the baseline. This

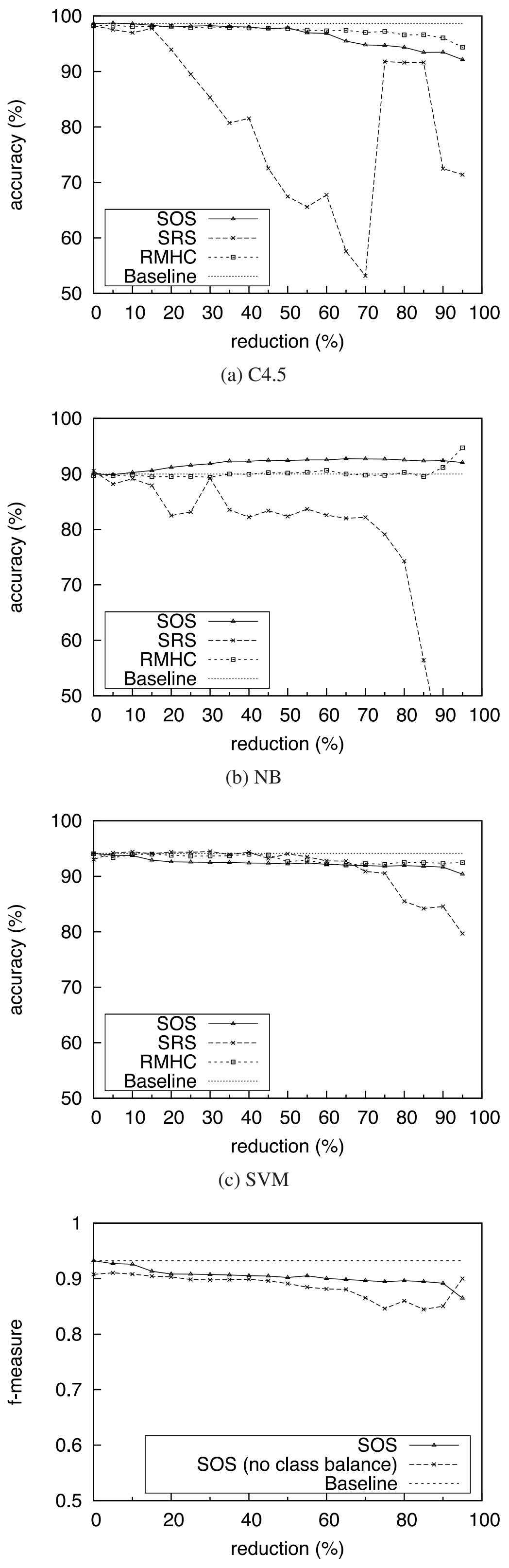
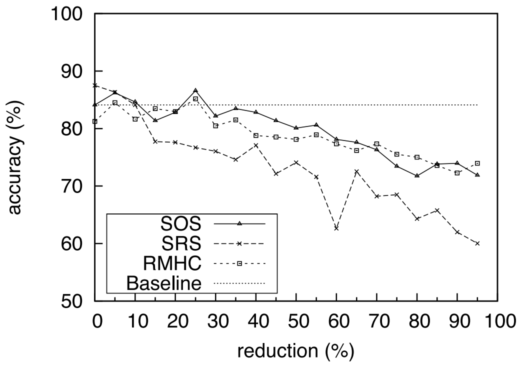
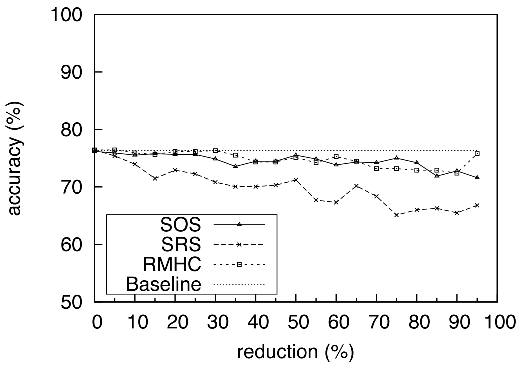


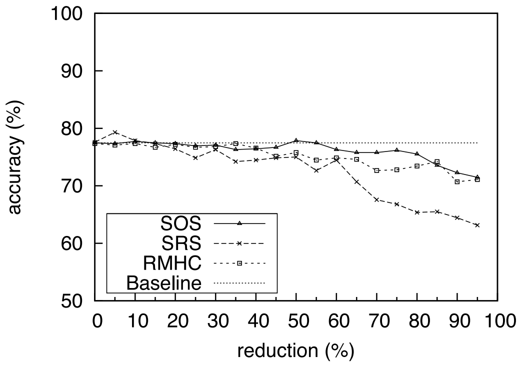
Fig. 6 Results on DS1.



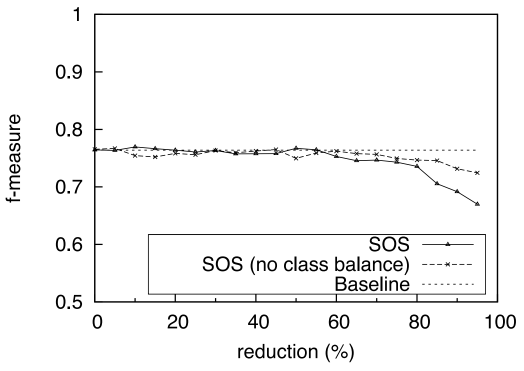
(a) C4.5



(b) NB

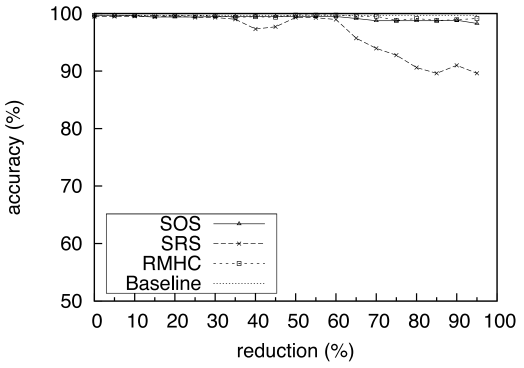


(c) SVM

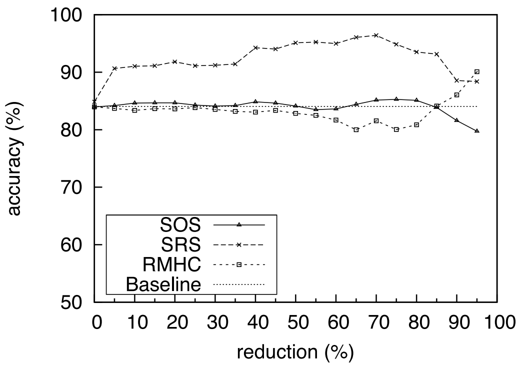


(d) Effects of Class Balancing

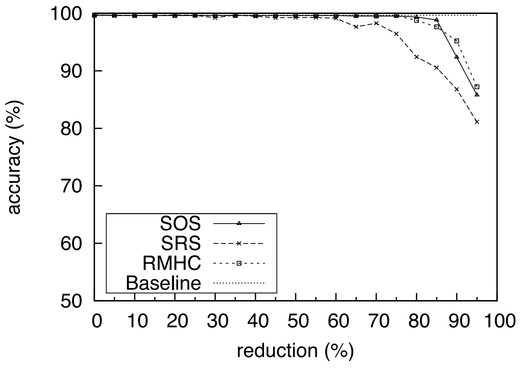
Fig. 7 Results on DS2.



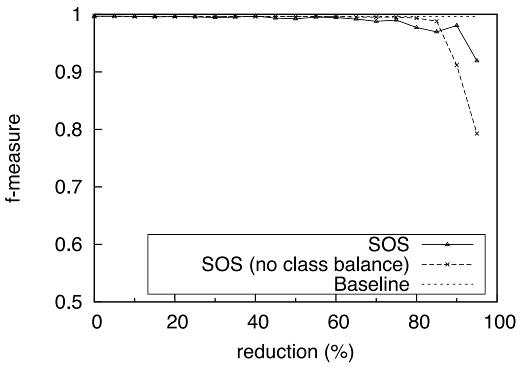
(a) C4.5



(b) NB



(c) SVM

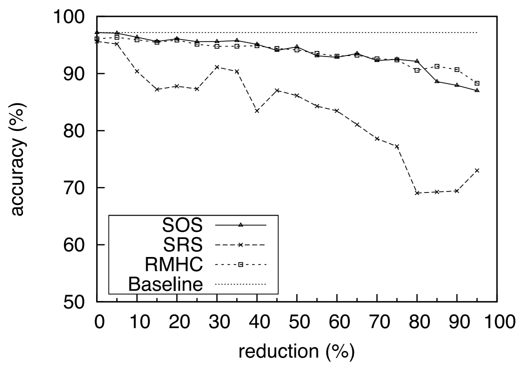


(d) Effects of Class Balancing

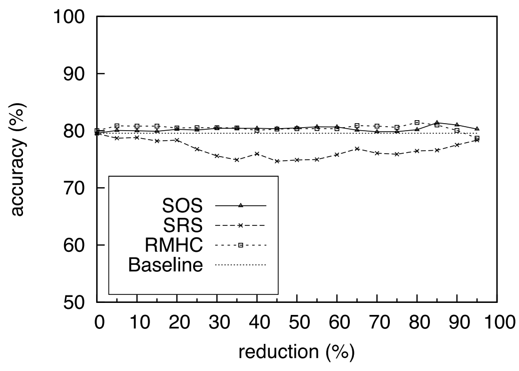
Fig. 8 Results on DS3.

is attributed to the fact that in addition to scaling down of datasets, SOS acts like a systematic randomization method as well. The observation of utmost importance is that SOS is able to scale down all datasets without compromising KE performance (accuracy). Class balancing does not seem to bring a great deal of improvement for the datasets we have investigated. But it does lead to better classifiers for highly imbalanced datasets (DS1, DS3 and DS5) as can be seen in Fig. 6 (d), Fig. 8 (d) and Fig. 10 (d).

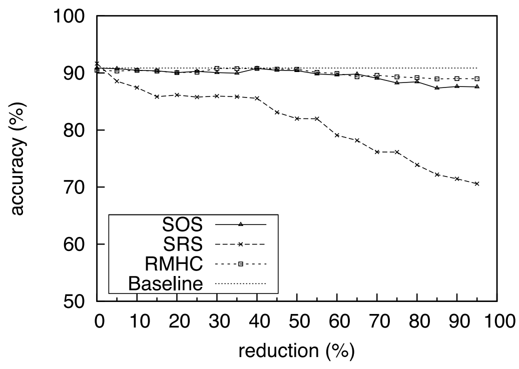
Comparison between SOS, SRS and RMHC:
Subfigures (a), (b) and (c) in Fig. 6, Fig. 7, Fig. 8, Fig. 9 and Fig. 10 also, comparatively, show accuracy levels of TD subsets



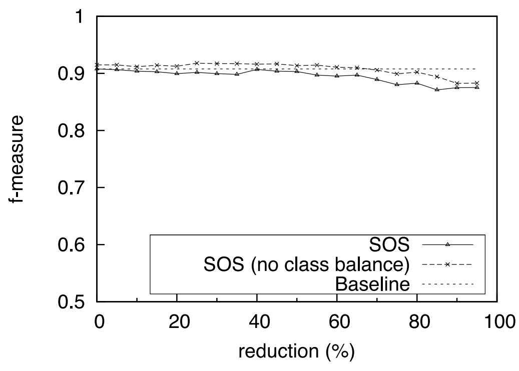
(a) C4.5



(b) NB

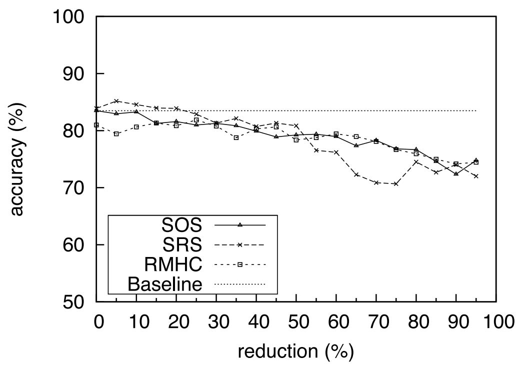


(c) SVM

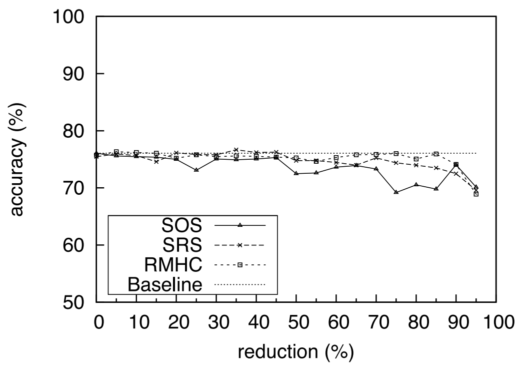


(d) Effects of Class Balancing

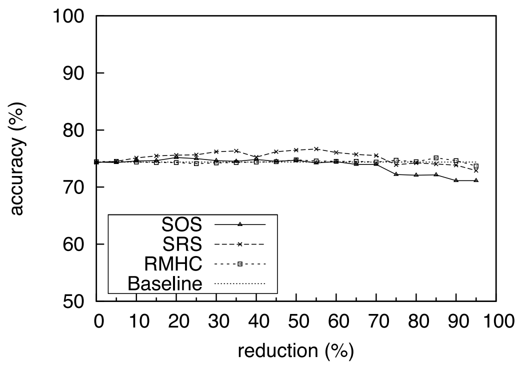
Fig. 9 Results on DS4.



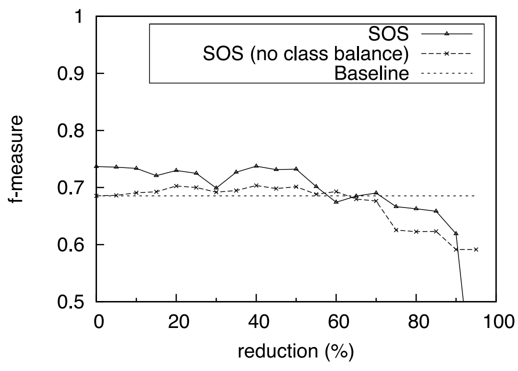
(a) C4.5



(b) NB



(c) SVM



(d) Effects of Class Balancing

Fig. 10 Results on DS5.

formed by using SOS, SRS and RMHC for C4.5, SVM and NB algorithms. In general all algorithms greatly preserve performance closer to the baseline. In three datasets (DS1, DS2 and DS4), SOS and RMHC outperform SRS while SOS slighty edges RMHC. Mostly, SOS slightly edges RMHC at lower reduction levels while slightly lags behind at higher reduction levels. These observations reflect the noise reduction ability of RMHC and noise tolerance ability of SOS. By imposing fitness test during selection, RMHC manages to get rid of many noisy examples, the effect of which is more noticeable on smaller subsets. This also implies that since SOS is doing better than RMHC at lower reduction levels, without getting rid of noisy examples, it tore-

Table 6 Comparison between SOS, SRS and RMHC (C4.5).

Metric	DS1	DS2	DS3	DS4	DS5	Avg.
SOS-Selected Subsets						
Accuracy	96.62	**78.84**	**99.37**	**92.27**	74.75	88.37
F-Measure	**0.952**	**0.791**	**0.994**	**0.923**	**0.746**	**0.881**
Time(ms)	158.22	41.97	49.77	229.15	40.41	103.90
SRS-Selected Subsets						
Accuracy	82.33	73.01	90.27	78.96	75.19	79.95
F-Measure	0.852	0.734	0.913	0.783	0.741	0.805
Time(ms)	**140.24**	**16.74**	**45.78**	**129.97**	**28.45**	**72.24**
RMHC-Selected Subsets						
Accuracy	**97.00**	76.57	99.18	92.14	**77.38**	**88.45**
F-Measure	0.918	0.734	0.983	0.918	0.701	0.851
Time(ms)	35660	2300	10560	147570	4120	20042

Table 7 TD sets for dimensionality test.

Name	ID	Size	Balance	Dimensionality
Page-blocks	**DS1**	5472	10:90	10
Twonorm	**DS6**	1729	49:51	20
Satimage	**DS7**	1720	44:56	36
coil	**DS8**	1767	6:94	85
Yeast	**DS5**	1739	11:89	103
Scene	**DS9**	1732	19:81	294

lates noise better than RMHC. The unreliable nature of the SRS method is evident in Fig. 6, Fig. 7 and Fig. 9. Accuracy values of the SOS and RMHC are much steadier than those of SRS in these figures. SRS happens to perform better than the others on one TD subset (Fig. 8) by the NB algorithm. This is because DS3 has many features and is highly imbalanced. Overall, SOS and RMHC are the best performers although SOS marginally edges RMHC as can be observed by F-Measure values shown in **Table 6** for the C4.5 algorithm. Results on the other algorithms follow the same pattern. In this table, the best performance values are shown in bold. Table 6 also shows how expensive RMHC is compared to the other two methods. RMHC is hundreds of times more expensive than the others. Therefore, from these observations, the best trade-off between cost, reduction ability and performance preservation is achieved when using SOS.

5.2 Experiment 2

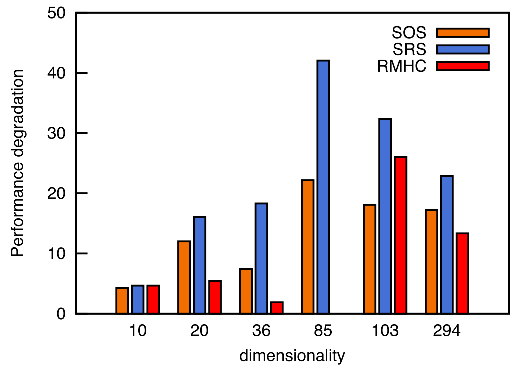
5.2.1 Overview

In this experiment, we investigate the effects of dimensionality and how the proposed method performs when used against multimodal dataset. Three performance indicators are used: average drop in accuracy after reduction, drop in F-Measure and average CPU time taken to form subsets. For testing dimensionality, datasets of six different dimensionality levels from UCI [10] and KEEL [1] are used. The datasets are shown in **Table 7**. For multimodal test, we used a subset of 'Opportunity Challenge' dataset [34] which comprised of two activities Locomotion and Gestures. **Table 8** describes this dataset.

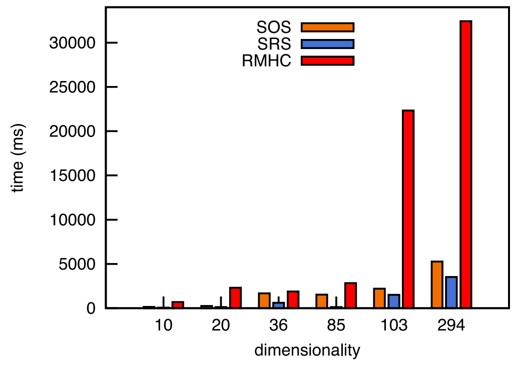
The datasets are prepared in a similar way as described in Experiment 1 and similar methodology is used. For multimodal dataset, we use 'Locomotion' modes to reduce numerosity and then test for classification performance against both 'Locomotion' and 'Gestures' modes. Performance test is done using C4.5 and SVM (SMO implementation).

Table 8 'Opportunity Challenge' Multimodal dataset.

Dimensionality	Size	Activity	Modes (Classes)
46	4501	Locomotion	Stand, Walk
		Gestures	Open_Door,Close_Door
			Close_Door,Open_Fridge
			Close_Fridge,Open_Washer
			Close_Washer,Open_Drawer
			Clean_Table,Drink_Cup



(a) Performance Degradation



(b) Subset Formation Time

Fig. 11 Effects of Dimensionality.

5.2.2 Results and Analysis

Effects of dimensionality:

Figure 11 (a) shows accuracy drops for SOS, SRS and RMHC approaches when tested against datasets in Table 7. Higher dimensional datasets led to more performance degradation for all methods. Figure 11 (b) shows how much time, on average, each method took to form the subsets. The results show that it takes longer with higher dimensionality for all methods.

From these results, a number of observations can be made. In Fig. 11 (a), although performance degraded, some high dimension datasets led to less degradation than their predecessors (e.g., dataset with dimensionality 294 against 103 and 85). This is because performance depends not only on dimensionality but also on other factors like balance, types of features and other data specific characteristics. As was the case in Experiment 1, although RMHC offers the least degradation in accuracy values, it consumes much more CPU time than other methods. On the other

Table 9 Multimodal dataset - C4.5.

Mode Type	Approach	Accuracy	F-Measure	Time (ms)
Locomotion	Benchmark	89.16	0.891	n/a
	SOS	88.16	0.885	7,285
	SRS	85.42	0.858	**6,220**
	RMHC	**89.36**	**0.892**	19,909
Gestures	Benchmark	93.67	0.937	n/a
	SOS	88.48	0.885	n/a
	SRS	86.86	0.868	n/a
	RMHC	**90.78**	**0.908**	n/a

Table 10 Multimodal dataset - SVM.

Mode Type	Approach	Accuracy	F-Measure	Time (ms)
Locomotion	Benchmark	83.20	0.823	n/a
	SOS	80.08	**0.809**	**4,172**
	SRS	79.69	0.806	4,775
	RMHC	**81.35**	0.793	14,218
Gestures	Benchmark	92.73	0.927	n/a
	SOS	89.99	0.899	n/a
	SRS	90.06	0.900	n/a
	RMHC	**90.37**	**0.903**	n/a

hand, SOS and SRS consume much less CPU time. However, contrary to SRS, performance degradation by SOS is only slightly higher than RMHC.

Performance on multimodal classes:

Tables 9 and **10** show results for SOS, SRS and RMHC approaches when tested using C4.5 and SVM respectively. In these tables, the bolded values indicate the best performing approach under each of the three performance metrics. We also show benchmark values for each tesing algorithm. Since we used 'Locomotion' modes to form the subsets (as stated above), subset formation time is non-applicable (n/a) for 'Gestures' modes.

From these results, four main observations can be made. First, all approaches have significantly preserved performance (close to bechmarking values). Second, subsets by RMHC lead to the best classification performances, in terms of accuracy and F-Measure. Third, RMHC is too expensive compared to SOS and SRS which take more or less similar times in forming the subsets. The fourth observation, which is the key observation, is that although SOS comes out second best, it offers the best balance between cost and performance. Its performance is only slightly below RMHC and takes only slightly more time than SRS in forming the subsets.

5.2.3 Conclusion

We conclude from our observations that the proposed SOS scales well with dimensionality (at least up to about 300). This means that the proposed SOS can also be used to reduce numerosity of high dimensional data while significantly preserving performance. Also the proposed SOS can be used with multimodal datasets.

5.3 Experiment 3

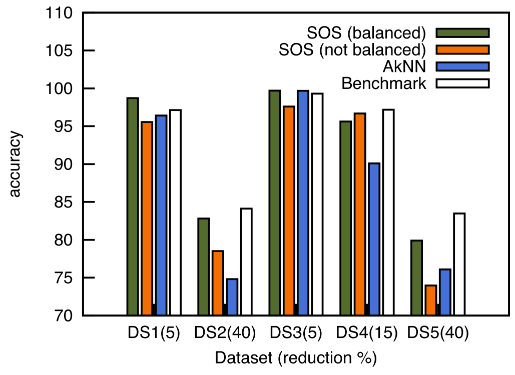
5.3.1 Overview

In this experiment, we compare the proposed SOS against parametric approaches (ENN and AkNN). The datasets and methodology of Experiment 1 are used. However, since ENN and AkNN can only give one subset (the smallest), we only use the proposed SOS to generate subsets which are similar in size with those offered by ENN and AkNN.

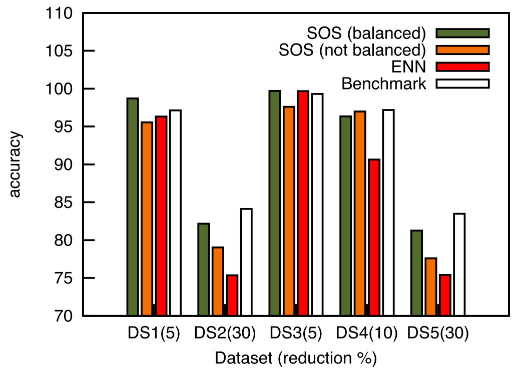
5.3.2 Results and Analysis

Figure 12 (a) and (b) summarize experimental results which compare the proposed SOS with AkNN and ENN respectively. In these figures, shown next to datasets IDs (DS1 to DS5) are the reduction percentages which were achieved by either AkNN or ENN and subsequently used to generate equivalent subsets using the proposed SOS.

It can be seen from the figures that the three methods (SOS, AkNN and ENN) offer close performances. For datasets DS1, DS2, DS4 and DS5, SOS (with balancing) outperforms both AkNN and ENN. This implies that a statistical advantage goes to SOS. In Experiment 1, it can also be seen that while SOS is shown to be able to reduce the datasets to around 95%, AkNN and ENN fails to reduce the datasets beyond 40% (at least for the datasets we have tested). In fact, for some datasets (DS1 and DS3), the reduction is even as low as 5%. This is because AkNN and ENN are more of noise reduction methods than serious reduction methods. This observation is also made in Ref. [31] where AkNN and ENN are seen to serve more as noise filters. These results do not give a general comparison between these methods, but compare trade-offs between reduction and performance preservation. Also, the observations assert the claim we made in Section 5.2.2 that SOS is noise tolerant. This is verified by the fact that it leads to even better performances than methods which were specifically designed to remove noise.



(a) Comparison with AkNN



(b) Comparison with ENN

Fig. 12 Effects of dimensionality.

5.3.3 Conclusion

From these experimental results, SOS with balancing is the better choice for reducing datasets in resource constrained environments while giving a good trade-off between reduction and performance. We conclude that the proposed SOS leads to performances more or less similar to parametric approaches.

6. Conclusion

In order to achieve lightweight KE from TD sets by the use of smaller data subsets, a method to scale down the sets with the best trade-off between cost, reduction ability and performance preservation was proposed in this paper. The method (SOS) is sampling-based and uses a novel ranking scheme called LO ranking to pre-rank examples before sampling or selection. Sampling-based methods are particularly good candidates when KE is taking place in a resource constrained environment because of their inexpensiveness. We have shown how SOS evaluates against baseline performances and how it compares against two classical sampling-based methods (SRS and RMHC). Our evaluation results show that SOS offers the best trade-off when compared to other methods. Moreover, SOS was shown to offer steadier performance, enhance performance in some cases and tolerate noise at lower reduction levels. We have also shown that the proposed SOS scales well with dimensionality and can be used with multimodal datasets.

Some areas still need improvements and further evaluations. These include the following. The first area is in experimentation on real agent-based applications running in a resource constrained environment, like automated smart homes and devices or sensor devices. The second is further research on a different approach to select representatives of the class partitions. The third is in researching on other possible applications of the proposed method.

References

[1] Alcalá-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., García, S., Sánchez, L. and Herrera, F.: KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework, *Journal of Multiple-Valued Logic and Soft Computing*, Vol.17, pp.255–287 (2011).

[2] Bdoiu, M. and Clarkson, K.L.: Optimal core-sets for balls, Comput. Geom, *Theory Application* (2008).

[3] Bigus, J.P.: The agent building and learning environment, *Proc. 4th International Conference on Autonomous Agents*, New York, USA, pp.108–109 (2000).

[4] Cano, J.R., Herrera, F. and Lozano, M.: Using evolutionary algorithms as instance selection for data reduction in KDD: An experimental study, *IEEE Trans. Evolutionary Computation*, Vol.7, No.6, pp.561–575 (2003).

[5] Cao, L., Gorodetsky, V. and Mitkas, P.A.: Agent Mining: The Synergy of Agents and Data Mining, *IEEE Trans. Intelligent Systems*, Vol.24, No.3, pp.64–72 (2009).

[6] Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P.: SMOTE: Synthetic Minority Oversampling Technique, *Journal of Artificial Intelligence Research*, Vol.16, pp.321–357 (2002).

[7] Cortes, C. and Vapnik, V.: Support-Vector Networks, *Machine Learning*, pp.273–297 (1995).

[8] Cunningham, P.: A Taxonomy of Similarity Mechanisms for Case-Based Reasoning, *IEEE Trans. Knowledge and Data Engineering*, Vol.21, No.11, pp.1532–1543 (2009).

[9] Farmer, M.E., Bapna, S. and Jain, A.K.: Large Scale Feature Selection Using Modified Random Mutation Hill Climbing, *Proc. 17th International Conf. on Pattern Recognition*, Vol.2, pp.287–290 (2004).

[10] Frank, A. and Asuncion, A.: UCI Machine Learning Repository, available from ⟨http://archive.ics.uci.edu/ml⟩, Irvine, CA: University of California, School of Information and Computer Science (2010).

[11] Gao, J., Ding, B., Fan, W., Han, J. and Yu, P.S.: Classifying Data Streams with Skewed Class Distributions and Concept Drifts, *IEEE Trans. Internet Computing*, Vol.12, No.6, pp.37–49 (2008).

[12] Garcia, S., Derrac, J., Cano, J.R. and Herrera, F.: Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.99 (2011) (PrePrint).

[13] Gustavo, E.A., Batista, P.A., Prati, R.C. and Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data, *ACM SIGKDD Explorations Newsletter*, Vol.6, No.1, pp.20–29 (2004).

[14] Hagras, H., Callaghan, V., Colley, M., Clarke, G., Pounds-Cornish, A. and Duman, H.: Creating an ambient-intelligence environment using embedded agents, *IEEE Intelligent Systems*, Vol.19, pp.12–20 (2004).

[15] Han, J. and Kamber, M.: *Data mining: Concepts and Techniques*, Second Edition, Morgan Kaufmann (2006).

[16] Kalegele, K., Sveholm, J., Takahashi, H., Sasai, K., Kitagata, G. and Kinoshita, T.: On-demand Data Numerosity Reduction for Learning Artifacts, *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, pp.151–159 (2012).

[17] Kalegele, K., Sveholm, J., Takahashi, H., Sasai, K., Kitagata, G. and Kinoshita, T.: Dynamic Numerosity Reduction for Mining-Based Agent Learning, *Proc. 19th Workshop on Multimedia Communications and Distributed Processing*, pp.51–56 (Oct. 2011).

[18] Kalegele, K., Sveholm, J., Takahashi, H., Sasai, K., Kitagata, G. and Kinoshita, T.: On-demand numerosity reduction for object learning, *Proc. Workshop on Internet of Things and Service Platforms*, Vol.6, pp.1–8 (Dec. 2011).

[19] Kumar, P., Mitchell, J.S.B. and Yildirim, E.A.: Approximate minimum enclosing balls in high dimensions using core-sets, *J. Exp. Algorithmics*, Vol.8, Article 1.1 (2003).

[20] Liu, H., Dougherty, E.R., Dy, J.G., Torkkola, K., Tuv, E., Peng, H., Ding, C., Long, F., Berens, M., Parsons, L., Zhao, Z., Yu, L. and Forman, G.: Evolving feature selection, *IEEE Trans. Intelligent Systems*, Vol.20, No.6, pp.64–76 (2005).

[21] Makita, Y., de Hoon, M. and Danchin, A.: Hon-yaku: A biology-driven Bayesian methodology for identifying translation initiation sites in prokaryotes, *Journal of BMC Bioinformatics*, Vol.8, No.1, p.47 (2007).

[22] Mitkas, P., Symeonidis, A., Kechagias, D., Athanasiadis, I.N., Laleci, G., Kurt, G., Kabak, Y., Acar, A. and Dogac, A.: An agent framework for dynamic agent retraining: Agent Academy, *Proc. 12th Annual Conference on eBusiness and eWork*, pp.16–18 (2002).

[23] Nielsen, F. and Nock, R.: Approximating smallest enclosing balls with applications to machine learning, *International Journal on Computational Geometry and Applications*, Vol.19 (2009).

[24] Platt, J.: Fast Training of Support Vector Machines using Sequential Minimal Optimization, *Advances in Kernel Methods - Support Vector Learning*, MIT Press (1998).

[25] Quinlan, J.R.: Improved Use of Continuous Attributes in C4.5, *Journal of Artificial Intelligence Research*, Vol.4, pp.77–90 (1996).

[26] Segata, N., Blanzieri, E., Delany, S.J. and Cunningham, P.: Noise reduction for instance-based learning with a local maximal margin approach, *Journal of Intelligent Information Systems*, Vol.35, pp.301–331 (2010).

[27] Seiffert, C., Khoshgoftaar, T.M., Van Hulse, J. and Napolitano, A.: RUSBoost: A Hybrid Approach to Alleviating Class Imbalance, *IEEE Trans. Systems, Man, and Cybernetics*, Vol.40, No.1, pp.1–4 (2010).

[28] Skalak, D.B.: Prototype and feature selection by sampling and random mutation hill climbing algorithms, *Proc. 11th International Conf. on Machine Learning*, pp.293–301 (1994).

[29] Symeonidis, A.L., Mitkas, P.A. and Kechagias, D.D.: Mining patterns and rules for improving agent intelligence through an integrated multi-agent platform, *Proc. 6th IASTED International Conference, Artificial Intelligence and Soft Computing* (July 2002).

[30] Tsang, I.W., Kwok, J.T. and Cheung, P.M.: Core Vector Machines: Fast SVM training on very large data sets, *Journal of Machine Learning Research* (*JMLR*), Vol.6, pp.363–392 (2005).

[31] Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data, *IEEE Trans. System, Man and Cyber-netics*, Vol.2, pp.408–421 (1972).

[32] Wilson, D.R. and Martinez, T.R.: Reduction Techniques for Instance-Based Learning Algorithms, *Journal of Machine Learning*, Vol.38, No.3, pp.257–286 (2000).

[33] Witten, I.H. and Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques*, Second Edition, Morgan Kaufmann (2005).

[34] available from ⟨http://www.opportunity-project.eu/⟩.

Khamisi Kalegele is a Ph.D. student at the Graduate School of Information Sciences, Tohoku University. He received his M.E. degree from Ehime University in 2010. He is currently researching on Knowledge Discovery and Data Mining in network and systems. His other areas of interest include intelligent network management and information extraction. He is a student member of IEEE and IPSJ.

Hideyuki Takahashi is an assistant professor of Research Institute of Electrical Communication of Tohoku University, Japan. He received his doctoral degree in Information Sciences from Tohoku University in 2008. His research interests include ubiquitous computing, green computing and agent-based computing. He is a member of IPSJ and IEICE.

Johan Sveholm was born in Sollentuna, Sweden. He received his M.E. in electrical engineering from The Royal Institute of Technology, Stockholm, Sweden in 2004 and his Dr.Eng. from Tohoku University, Sendai, Japan in 2008. Since 2011 he is employed as a fellow researcher of the Research Institute of Electrical Communication of Tohoku University. His current research interest includes Agile software development, artificial neural networks and intelligent agents.

Kazuto Sasai is an assistant professor of Research Institute of Electrical Communication, Tohoku University, Japan. He received his Dr.Sci. in Earth and Planetary Science from Kobe University. His research interests include network management, network analysis, complex systems and agent-based systems. Dr. Sasai is a member of IEICE, IPSJ, JSAI, and BSJ.

Gen Kitagata is an associate professor of Research Institute of Electrical Communication of Tohoku University, Japan. He received a doctoral degree from Graduate School of Information Sciences, Tohoku University in 2002. His research interests include agent-based computing, network middleware design, and symbiotic computing. He is a member of IEICE and IPSJ.

Tetsuo Kinoshita is a professor of Research Institute of Electrical Communication of Tohoku University. He received his B.E. degree in electronic engineering from Ibaraki University, Japan, in 1977, and M.E. and Dr.Eng. degrees in information engineering from Tohoku University, Japan, in 1979 and 1993, respectively. His research interests include agent engineering, knowledge engineering, knowledge-based and agent-based systems. He received the IPSJ Research Award, the IPSJ Best Paper Award and the IEICE Achievement Award in 1989, 1997 and 2001, respectively. Dr. Kinoshita is a member of IEEE, ACM, AAAI, IEICE, IPSJ, and JSAI.