

# On-demand Numerosity Reduction for Object Learning

Khamisi Kalegele  
GSIS, Tohoku University  
2-1-1 Katahira, Aoba-Ku  
Sendai 980-8577, Japan  
[kalegs@k.riec.tohoku.ac.jp](mailto:kalegs@k.riec.tohoku.ac.jp)

Kazuto Sasai  
GSIS/RIEC, Tohoku University  
2-1-1 Katahira, Aoba-Ku  
Sendai 980-8577, Japan  
[kazuto@fir.riec.tohoku.ac.jp](mailto:kazuto@fir.riec.tohoku.ac.jp)

Johan Sveholm  
RIEC, Tohoku University  
2-1-1 Katahira, Aoba-Ku  
Sendai 980-8577, Japan  
[jsveholm@riec.tohoku.ac.jp](mailto:jsveholm@riec.tohoku.ac.jp)

Gen Kitagata  
GSIS/RIEC, Tohoku University  
2-1-1 Katahira, Aoba-Ku  
Sendai 980-8577, Japan  
[minatsu@fir.riec.tohoku.ac.jp](mailto:minatsu@fir.riec.tohoku.ac.jp)

Hideyuki Takahashi  
GSIS/RIEC, Tohoku University  
2-1-1 Katahira, Aoba-Ku  
Sendai 980-8577, Japan  
[hideyuki@riec.tohoku.ac.jp](mailto:hideyuki@riec.tohoku.ac.jp)

Tetsuo Kinoshita  
GSIS/RIEC, Tohoku University  
2-1-1 Katahira, Aoba-Ku  
Sendai 980-8577, Japan  
[kino@riec.tohoku.ac.jp](mailto:kino@riec.tohoku.ac.jp)

## ABSTRACT

In Internet of Things, softwares shall enable their host objects (everyday-objects) to monitor other objects, take actions, and notify humans while using some form of reasoning. The ever changing nature of real life environment necessitates the need for these objects to be able to generalize various inputs inductively in order to play their roles more effectively. These objects shall learn from stored training examples using some generalization algorithm. In this paper, we investigate training sets requirements for object learning and propose a Stratified Ordered Selection (SOS) method as a means to scale down training sets. SOS uses a new instance ranking scheme called LO ranking. Everyday-objects use SOS to select training subsets based on their capacity (e.g. memory, CPU). LO ranking has been designed to broaden class representation, achieve significant reduction while offering same or near same analytical results and to facilitate faster on-demand subset selection and retrieval for resource constrained objects. We show how SOS outperforms other methods using well known machine learning datasets.

## Categories and Subject Descriptors

I.2.6 [ARTIFICIAL INTELLIGENCE]: Learning

## General Terms

Algorithms, Performance

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ACM IoTSP 2011, December 6, 2011, Tokyo, Japan.

IoTSP'11 December 6, 2011, Tokyo Japan

Copyright 2011 ACM 978-1-4503-1043-7/11/0012 ...\$10.00.

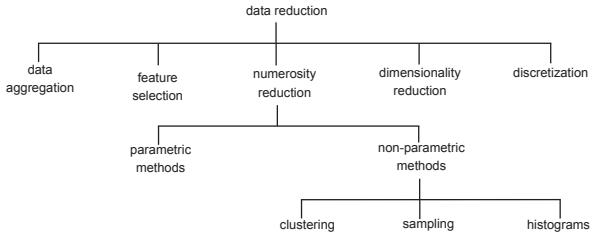
## Keywords

Learning, Ranking, Sampling, Data Reduction

## 1. INTRODUCTION

The new promising so called Internet of Things (IoT) concept is not a wheel under invention but rather existing technologies being federated. Various federation proposals which cover low level issues like architectural foundations, protocol suite designs and service layers are now being put forward. In few years time (around 2020 as per “Global Trends 2025” by SRI Consulting Business Intelligence), we shall see advancements of IoT-based systems in areas like ubiquitous positioning and physical-world web. Prospects will then be even more federation of existing successful paradigms and methodologies into solutions for objects in order to make them more smarter and capable. Solutions for IoT-based systems will include approaches like fusion of agent technologies and advanced sensors, adaptation of machine learning (ML) approaches into object learning (OL) etc. We use OL to refer to object-embedded agent learning. In this paper, we focus on how OL in IoT-based systems can be adopted from ML approaches.

OL differs from learning in superior IT systems primarily because of the fact that objects have scarce resources. For whichever of the existing multitude of learning methods to be adopted, its respective resource-sensitive issues must be looked upon. In this paper, we focus on inductive learning (generalization) in which there are two resource-sensitive issues: generalization algorithms and training data. Computational complexity of a generalization algorithm might further constrain an already resource-constrained object. Also the impact of each of the specifications of training sets on generalization must be checked upon. The specifica-



**Figure 1: Taxonomy of reduction approaches.**

tions include numerosity, dimensionality, features types etc. Numerosity is the scope of this paper. We chose numerosity because it is the one specification for which a generic approach for its optimization towards an improved learning performance can be sought. Other specifications are too dependent of specific data.

While reinforcement learning (e.g. Q-learning, ISL[5]) is a performance improvement necessity to object-embedded agents when interacting with the world, inductive learning is very useful for learning past experiences and fixed behaviors and rules (usually pre-programmed), e.g. learning temperatures at which water pipes freeze.

## 2. NUMEROSITY OF TRAINING DATA

In generalization, the more numerous training sets are the better [6]. So, it is best to have as many training examples as possible though a tradeoff exists between their numbers (numerosity), accuracy and generalization cost. This tradeoff is dealt with while observing an acceptable level of generalization performance. To maintain an acceptable level of generalization performance when dealing with this tradeoff, two types of approaches have been used.

1. Approaches which seek to reduce numerosity in a way that preserves or enhances generalization performance when cost (e.g. memory, CPU, bandwidth) is not an issue. They are referred to as numerosity reduction methods and are part of general data reduction methods shown in Figure 1.
2. Those which seek to either develop new generalization algorithm or improve existing ones in order to reduce cost while maintaining performance levels irrespective of numerosity of the datasets.

### 2.1 Numerosity Reduction in ML

Numerosity reduction in ML, have been done for two major reasons. One is for the sole purpose of saving storage and reducing cost (e.g. CPU). Another is for noise reduction as part of data pre-processing. Either ways, studies [7] have revealed that reducing numerosity beyond a certain data specific level leads to a degraded generalization performance.

Referred to as instance selection [1], instance ranking [3], storage reduction [11], noise reduction [8], editing,

or simply sampling, numerosity reduction have been approached in a number of ways. They include;

1. Smaller random sample (SRS) [6]:  $s$  of the  $N$  examples from training dataset,  $D$  ( $s < N$ ) are randomly drawn. A special technique is called stratified sample whereby if there exist mutually disjoint parts in  $D$  called strata, a stratified sample of  $D$  is generated by obtaining an SRS at each stratum.
2. NN-based methods: (e.g. RENN and AkNN [10]). The idea is to consider  $k$  nearest neighbors ( $k$ -NN) and either remove an example which can correctly be classified by the majority of its neighbors (decremental) or take an example which correctly classifies most of its neighbors as the representative of the  $k$  examples (incremental).
3. Ordered Removal: These take special care of the order in which examples (instances) are removed. Prominent example is DROP procedures [11]. These procedures seek to remove an example which would degrade leave-one-out cross-validation (a statistical cross-validation approach by which each of the examples contained in a dataset is used in turn as a testing example while the rest become training examples) accuracy. They work as follows; suppose  $S$  is the desired subset and  $T$  is the original set. An employed rule is that an example  $P$  is removed from  $S$  if at least as many of its associates in  $S$  would be classified correctly without  $P$ .
4. Random Mutation Hill Climbing (RMHC) [9]: This is a classical sampling-based method which improves reliability by using an accuracy fitness test. Although randomly selected, examples are only kept if they enhance accuracy.

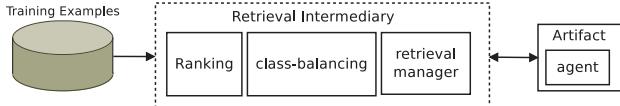
The various methods are compared in terms of their size reduction ability, noise tolerance, generalization performance improvement and computational complexity. What have mattered the most though have been size reduction and generalization performance. Computational complexity has always been considered less important because reduction process is a one-time process.

### 2.2 Numerosity Reduction for OL

Considering the aforesaid fact that objects in IoT-based systems have scarce resources, resource sensitivity is of utmost importance when devising or choosing a numerosity reduction approach for OL. As mentioned in the previous subsection, computational complexity is relatively less a concern in ML. This makes most of the approaches unlikely fit for OL.

### 2.3 Research Objectives

Just like in ML, reducing numerosity for OL beyond some data dependent level leads to a degraded gener-



**Figure 2: Proposed subset retrieval approach.**

alization performance. So while observing all other requirements (listed below), numerosity of training datasets in IoT-based systems must be reduced in a way that either maintains or enhances generalization performance. A numerosity reduction method for OL must therefore adhere to the following requirements.

- Significant reduction
- Acceptable level of generalization accuracy
- Low computational cost to allow for faster on-demand reduction in order to cope with the objects' individualism and dynamism.

The objectives of this research is to devise a training data subset selection method fit for use in OL, i.e. a method which is simple, inexpensive and observes the abovesaid requirements. In the next section, we present a proposed solution. We describe SOS subset selection method and its associated LO ranking in details in this section. The proposed method is evaluated in section 4, and section 5 concludes this paper.

### 3. THE PROPOSED METHOD

Our proposed method allows objects to retrieve different sized data subsets based on available resources. Figure 2 shows our approach to data subset selection for OL. An agent-based intermediary implements a ranking-based subset selection method (SOS). It facilitates subsets selections and retrievals. To compensate for losses in generalization performance associated with the use of smaller subsets, the retrieval intermediary additionally employs class balancing.

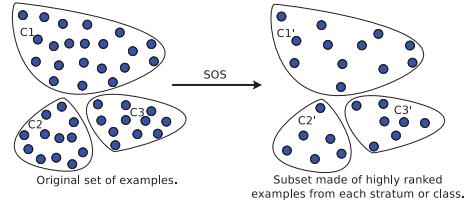
We describe the proposal according to the three functions implemented in the retrieval intermediary.

1. Retrieval Manager: Handles negotiations (about subsets sizes and class balances) with objects.
2. Ranking: Ranks examples within each stratum or class of examples for subset selection and class balancing.
3. Class Balancing: Undersamples or oversamples a class in order to achieve the desired subset requirements.

We use the term “representatives” to refer to “selected training examples”.

#### 3.1 Subset Selection and Retrieval

Subset selection and retrieval are functions of the retrieval manager. Selection is done using SOS method.



**Figure 3: Stratified Ordered Selection.**

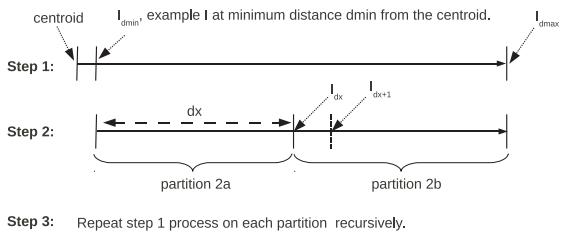
We call this selection “stratified” just like in SRS because examples are selected from different classes independently and proportionally as per the desired class balance. Our assumption is that training data will be composed of disjoint strata (classes). We also call this selection method “ordered” just like in DROP [11] because we ought to select examples from the individual classes in a special order as opposed to random in SRS. For this reason, examples are ranked using a new ranking scheme called, LO ranking, before selection. Figure 3 gives more insight into SOS. Below are the four-steps involved during subset retrieval.

1. An object initiates the process by sending its requirements (desired size and class balance).
2. The manager verifies the requirements. It checks whether the desired subset specifications are achievable, i.e. whether the amount of sampling needed to achieve the class balance ratio is reasonable and re-adjust the specifications whenever necessary. Finally, it sends the new or the verified subset specifications to an object.
3. An object acknowledges if it is satisfied with the proposed specifications otherwise it goes back to step 1 and send new requirements.
4. The manager selects and retrieves a subset according to the desired specifications using SOS.

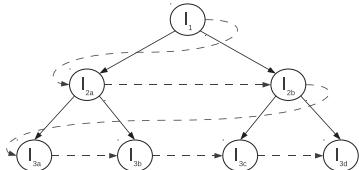
In step 2 above, the manager uses user-defined sampling limits and information about training datasets to perform verification task. Sampling limits are discussed in subsection 3.3. In step 4, the retrieval process involves oversampling and undersampling. Undersampling or simply selection is achieved by selecting highly LO-ranked examples and oversampling is achieved by creating new synthetic examples using SMOTE technique [2]. Subsection 3.3 provides more details.

#### 3.2 Level Order Ranking

LO ranking is an intra-class ranking scheme. It seeks to identify representatives which broaden class representation. A hybrid selection type is used incrementally as opposed to condensation (methods which seek to retain border points) only or edition (methods which seek to retain central points) only. We argue that the prime central example of any class is the closest-to centroid and the prime border example is the furthest-from the



(a) Representation levels



(b) Ranking binary tree.

Figure 4: Level Order ranking

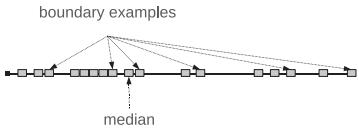


Figure 5: Example distance space.

central example. This scheme therefore considers the closest-to class centroid example as the highest ranked in the class and the furthest-from the highest ranked example as the second highest ranked example.

The rest of the ranking order is established as shown in Figure 4. First, representatives from each class portion within the distance space of a class are identified. Then using their respective distances, they are made into a binary tree, ranked in a level order manner. The levels in the tree correspond to the different class portions and sizes. It is from the way by which examples are retrieved from the tree that we named this method LO ranking. We explain the details of LO ranking next.

1. *Distance space of class examples:* Let  $I_{dx}$  be an example at a distance  $dx$  from the centroid in a distance space (step 1 in Figure 4(a)). There exists two examples  $I_{dmin}$  and  $I_{dmax}$  such that,  $dmin = \min\{dx\}$  and  $dmax = \max\{dx\}$  for  $1 \leq x \leq N$ . where  $N$  is the total number of examples.
2. *Level one representative:* Using the measure of central tendency on distance, an ideal representative would be the median ( $I_{dmedian}$ ) described by the following expression.

$$\exists I_{dx} : x = \frac{N}{2} \quad \forall N \in even, \quad x = \frac{N+1}{2} \quad \forall N \in odd$$

Where  $2a$  and  $2b$  are partitions of the distance space when partitioned at a representative as shown in step 2 of Figure 4(a). However, since we are interested in boundary examples too, the median example is not necessarily at the boundary of a cluster of examples. Consider examples in a distance space in Figure 5. The median here is not at the boundary but an example next to it is. We therefore settle for an example which will give  $\frac{\min(\#2a, \#2b)}{\max(\#2a, \#2b)}$  as close as possible to 1 and large  $d(x+1) - dx$  for a representative of level 1.

So, a representative of level 1 is an example  $I_{dx}$  which maximizes the measure shown in Equation 1.

$$E(I_{dx}) = [d(x+1) - dx] * \frac{\min(\#2a, \#2b)}{\max(\#2a, \#2b)} \quad (1)$$

We call this representative  $I_1$  and its distance  $d_1$  (root of a binary tree, T in Figure 4(b)).

3. *Representatives of subsequent levels:* Representatives of subsequent levels are obtained by repeating step 2 on partitions  $2a$  and  $2b$  separately to get  $I_{2a}$  and  $I_{2b}$  which then partition their respective spaces into  $\{3a$  and  $3b\}$  and  $\{3c$  and  $3d\}$  respectively. Distances,  $d_{2a}$  and  $d_{2b}$ , then make entries into T. This is then done recursively until no more boundary examples exist or until the desired number of representatives is reached.

4. *Ranking:* From highest to lowest rank, examples are then ordered as

$$I_{dmin}, I_{dmax}, I_1, I_{2a}, I_{2b}, I_{3a}, \dots$$

whereby from  $I_1$ , examples are retrieved from T in a level order fashion as shown by the dotted lines in Figure 4(b).

LO ranking uses Minkowski distance (2). Our choice is because two of the commonly used distance spaces ( $p = 1$  for Manhattan distance and  $p = 2$  for Euclidean distance) are also included in Minkowski space and also the fact that at higher values of  $p$  (order of Minkowski metric), Minkowski distance assigns more weight onto features on which data-objects differ the most. This makes it perform better when dealing with similarities.

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

$$\mathbf{x} = (x_1, x_2, \dots, x_n), \quad \mathbf{y} = (y_1, y_2, \dots, y_n)$$

### 3.3 Sampling

According to SOS, subsets are formed by sampling each of the classes in a training dataset. During sampling, we allow for class balancing to compensate for any losses in generalization performance associated with size

**Table 1: Sampling limits.**

case	min sampling	maj sampling
I	$-(\beta N - s_m)$	$-(N(1 - \beta) - s_m)$
II	$-(\beta N - s_m)$	none
III	$+(s_m - \beta N)$	none

reduction and class imbalance. Class imbalance and size reduction are well known ML problems which can lead to performance degradation. As pointed out already in subsection 3.1, oversampling is achieved by adding new synthetic examples into a class to be oversampled in order to achieve the desired class balance. Synthetic examples are created using SMOTE technique [2] whose steps are described below.

**Step 1.** An example is chosen at random from within the class to be oversampled.

**Step 2.** Five nearest neighbors are selected.

**Step 3.** A desired number of points are selected at random along the line segments joining the randomly chosen example and any two of its neighbors.

**Step 4.** New examples at each of the selected points in step 3 are inserted.

Assuming a two-class problem, training dataset contains minority (min) examples and majority (maj) examples. Class balance is the ratio of minority examples to majority examples. We define training dataset specification to have two attributes; size and class balance.

$$D = (\mathbf{x}_i, y_i) : i = 1, \dots, N, \mathbf{x}_i \in \mathbb{R}^p, y_i \in \{\text{maj}, \text{min}\}$$

and denote as  $D\{N, \beta\}$  where  $\beta = \text{class balance}$ .

Suppose a subset  $s\{n, \alpha\} \in D\{N, \beta\} : n < N$  is needed and let  $s_m = \alpha n$  be the desired number of minority examples. There are three possible cases as shown below with their respective achievable specifications (specs).

#### Case I:

$$s_m \leq \beta N \text{ and } n - s_m \leq N(1 - \beta) \rightarrow \text{specs} = s\{n, \alpha\}$$

#### Case II:

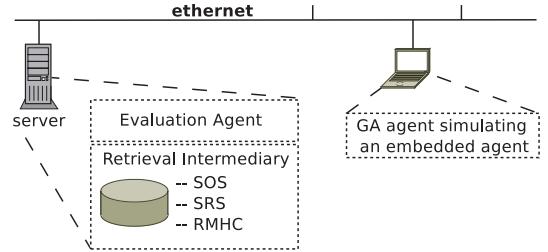
$$s_m \leq \beta N \text{ and } n - s_m > N(1 - \beta) \rightarrow$$

$$\text{specs} = s\{s_m + N(1 - \beta), \frac{s_m}{s_m + N(1 - \beta)}\}$$

#### Case III:

$$s_m > \beta N \text{ and } n - s_m \leq N(1 - \beta) \rightarrow \text{specs} = s\{n, \alpha\}$$

For each case, the sampling limiting amounts are shown in Table 1. The negative sampling amounts indicate undersampling while the positive sampling amounts indicate oversampling. Sampling is restricted to undersampling and oversampling of the minority class and undersampling of the majority class only.



**Figure 6: Simulation setup.**

## 4. EVALUATION

Our plan is a two-phase evaluation. In phase I, we investigate performance of SOS in terms of size reduction ability (while preserving generalization performance) and how it compares with other methods. Phase II covers feasibility of inductive learning and SOS in OL. We consider all objects' environments including connectivity, computational power and memory in this phase. In this paper, we present phase I in which we evaluated SOS on a number of well known machine learning datasets (Table 2) and investigated the following.

- (1) *Performance of SOS.* Generalization performance on different sized SOS-selected subsets and comparison with baseline values.
- (2) *Comparison with other methods.* Generalization performances and execution times of SOS and benchmarking methods.

The datasets were obtained from repositories of the Center of Machine Learning and Intelligent Systems of the University of California and KEEL (Knowledge Extraction based on Evolutionary Learning). It has been revealed in some published evaluation works [4, 8] that DROP gives the best mix of generalization accuracy and storage (therefore numerosity) reduction. AkNN was also found to offer good accuracy and significant size reduction. However we choose to compare with SRS and RMHC because first they are the most similar to SOS as they are all sampling-based methods. Second, we choose SRS for its inexpensiveness and yet ability to offer significant reduction with minimum performance degradation. And lastly, because RMHC is one of the classical sampling-based numerosity reduction methods.

### 4.1 Methodology, Metrics and Parameters

Generalizations were done on subsets selected using SOS, RMHC and SRS from partitions of the original datasets. For each dataset, 10 partitions were used. The partitions were obtained using 10-fold cross validation procedure. A dataset is randomly divided into 10 disjoint sets of same sizes. 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 sub-

**Table 2: Training datasets.**

Name	Size	Balance	#Features
Page-blocks	5472	10%:90%	10
Pima	768	35%:65%	8
Spam	4597	39%:61%	57
Segment0	2308	14%:86%	19
Yeast0	1484	11%:89%	8

Page-blocks: Blocks of document page layout.

Pima: Pima Indians diabetes data.

Spam: Spam filtering data.

Segment0: Image segmentation data .

Yeast0: Cellular localization sites of proteins.

**Table 3: Parameters and environments.**

Item	Details
Generalization algorithm	C4.5
Validation Method	10-fold cross validation
Learning Library	WEKA
Agent Framework	ABLE
Training Partitions	0 to 95% of original
Class Balance	Original and 50:50 balance
Distance Function	Minkowski with p=5
OS	Ubuntu x86_64

sets for generalization using a generalization algorithm. Everything was done in an agent-based computing environment as shown in Figure 6. Other parameters and simulation environments are shown in Table 3. For each pair of training and testing partitions, generalization was done twice and average generalization performance metric values were recorded. Using C4.5 as a generalization algorithm, makes classification accuracy the most important metric for generalization performance as far as numerosity reduction is concerned. This is also argued by Segata et. al [8]. Nonetheless, we also investigated Receiver Operating Characteristic (ROC) and F-Measure of the resulting C4.5 classifiers. The ROC curve is created by plotting true positive rate against false positive rate. Area under the ROC curve (AUC) and F-Measure (Equation 3) are single numerical metrics commonly used to compare model performances [6].

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

## 4.2 Results and Analysis

*Optimal value of the order of Minkowski metric, p.* Table 4 shows average generalization accuracy values for different values. Marginally,  $p = 5$  offers the best performance for all datasets but one. We therefore used

**Table 4: Optimal order of Minkowski metric.**

dataset	p=1	p=2	p=3	p=5	p=7
Page-blocks	95.20	94.99	95.71	<b>96.62</b>	94.70
Pima	76.78	77.94	75.37	<b>78.84</b>	77.77
Spam	92.02	91.82	92.14	<b>92.27</b>	<b>92.27</b>
Segment0	98.40	98.98	98.37	<b>99.37</b>	98.85
Yeast0	<b>75.34</b>	74.62	75.04	74.75	75.08

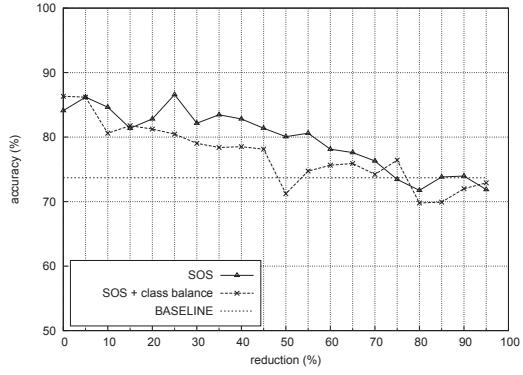
**Table 5: Baseline performance metrics.**

Dataset	accuracy	F-Measure	AUC
Page-blocks	97.13	0.971	0.942
Pima	73.70	0.735	0.761
Spam	93.24	0.932	0.940
Segment0	99.31	0.993	0.988
Yeast0	76.01	0.754	0.743

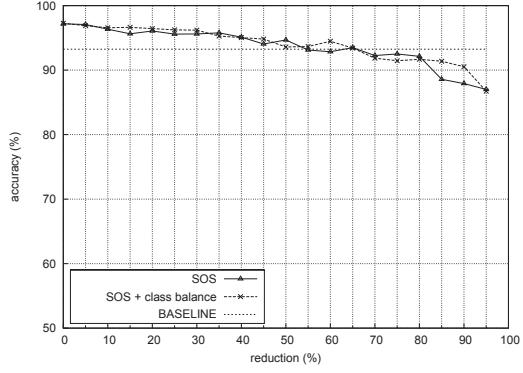
5 as the value of the order of Minkowski metric.

*Performance of SOS.* For comparison purposes, baseline performance metrics on the original datasets are shown in Table 5. Figure 7 shows performance of SOS (with and without class balancing) in comparison to the baseline values for four of the five datasets. Comparisons with other selection methods are shown in Figure 8. In Figure 7, as expected, performances drop with reduced numerosity. However, SOS-selected subsets are shown to yield performances higher than or closer to baseline values at initial stages. After much reduction performances drop slightly below baseline values. Class balancing does not seem to bring much improvement from these accuracy plots. This is because class balancing is primarily meant to improve the quality of the model in classifying minority class examples. This is depicted to be the case by F-Measure(s) of the minority classes, e.g. for page-blocks dataset as shown in Figure 9. It is shown in this figure that after further reduction, it reaches a stage where balanced SOS-selected subsets yield better generalization models.

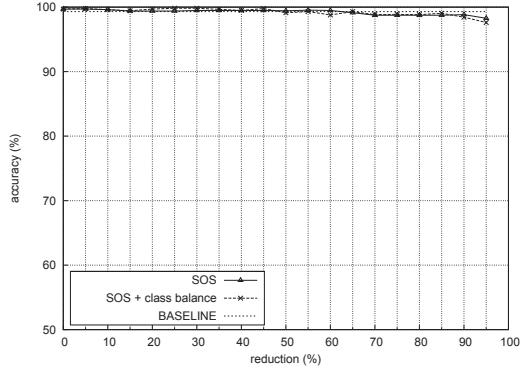
*Comparison with RMHC and SRS.* In Figure 8, the unreliability nature of SRS is revealed, e.g. in Figure 8(b) and Figure 8(d). Performances by RMHC-selected subsets seem to start below that of SOS but finish higher after further reduction. This can be seen in Figure 8(a), Figure 8(b) and Figure 8(d). This observation between RMHC and SOS indicates the noise reduction ability of RMHC at later reduction stages and noise tolerance ability of SOS and early reduction stages. Table 6 summarizes average metric values and execution times for all datasets. In the table, the best performing values are bolded. The cost of fitness function in RMHC is evident as it was observed to be the most expensive selection method.



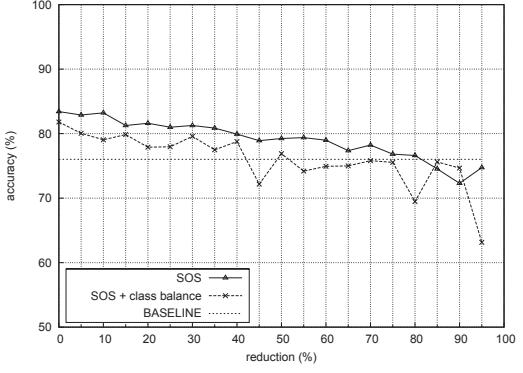
(a) Pima



(b) Spam

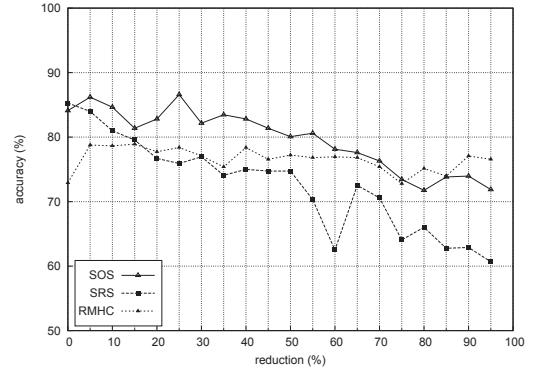


(c) Segment

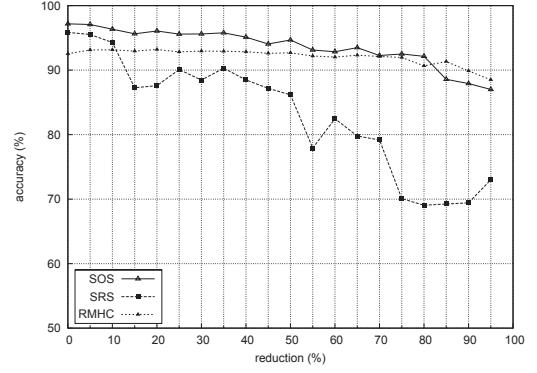


(d) Yeast

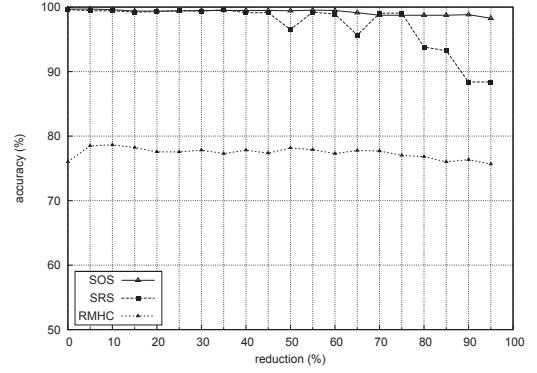
**Figure 7: Performance of SOS.**



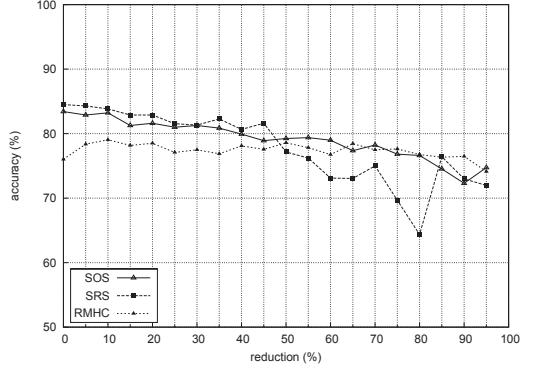
(a) Pima



(b) Spam



(c) Segment

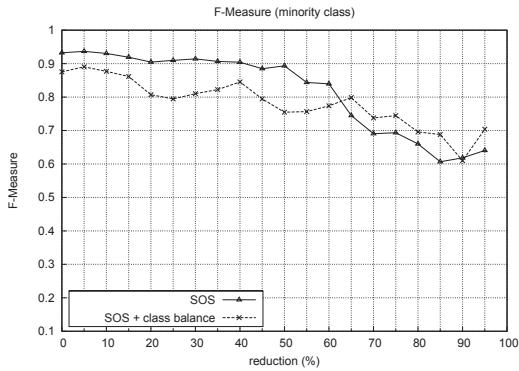


(d) Yeast

**Figure 8: Comparison with SRS and RMHC.**

**Table 6: Other performance metrics.**

SOS					
Metric(avg)	Page-blocks	Pima	Spam	Segment0	Yeast0
Accuracy	96.62	<b>78.84</b>	<b>92.27</b>	<b>99.37</b>	74.75
F-Measure	<b>0.952</b>	<b>0.791</b>	<b>0.923</b>	<b>0.994</b>	<b>0.746</b>
AUC	0.914	<b>0.822</b>	0.924	<b>0.989</b>	0.730
Time(ms)	158.22	41.97	229.15	49.77	40.41
SRS					
Accuracy	82.33	73.01	78.96	90.27	75.19
F-Measure	0.852	0.734	0.783	0.913	0.741
AUC	0.823	0.746	0.760	0.891	0.672
Time(ms)	<b>140.24</b>	<b>16.74</b>	<b>129.97</b>	<b>45.78</b>	<b>28.45</b>
RMHC					
Accuracy	<b>97.00</b>	76.57	92.14	99.18	<b>77.38</b>
F-Measure	0.918	0.734	0.918	0.983	0.701
AUC	<b>0.936</b>	0.750	<b>0.931</b>	0.985	<b>0.732</b>
Time(s)	35.66	2.30	147.57	10.56	4.12



**Figure 9: Effects of class balancing.**

## 5. CONCLUSION AND FUTURE WORK

This paper has presented a ranking-based method of selecting training subsets for inductive learning for resource constrained objects. We have shown, using some well known machine learning datasets, that SOS can achieve significant reduction while maintaining same or near same learning performance. At lower reduction percentages, SOS-selected subsets yield better generalization performance than baseline performance. SOS can therefore also be used as a generalization performance enhancement method. SOS has also been shown to outperform other sampling-based, non-parametric methods in terms of a better mix of performance preservation, noise tolerance and computational complexity.

Presented in this paper is evaluation phase I where we focused only on the performance of SOS with regard to numerosity reduction ability. In evaluation phase II, we will look into the real aspects of objects' environments (computing and networking resources), studying feasibility of inductive learning in OL, and usefulness of SOS.

## 6. REFERENCES

- [1] J. Cano, F. Herrera, and M. Lozano. Using evolutionary algorithms as instance selection for data reduction in kdd: an experimental study. *Evolutionary Computation, IEEE Transactions on*, 7(6):561 – 575, dec. 2003.
- [2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.
- [3] S. Clémençon and N. Vayatis. Ranking the best instances. *J. Mach. Learn. Res.*, 8:2671–2699, December 2007.
- [4] S. Garcia, J. Derrac, J. Cano, and F. Herrera. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(PrePrints), 2011.
- [5] H. Hagras, V. Callaghan, M. Colley, G. Clarke, A. Pounds-Cornish, and H. Duman. Creating an ambient-intelligence environment using embedded agents. *IEEE Intelligent Systems*, 19:12–20, November 2004.
- [6] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.
- [7] Y. Makita, M. de Hoon, and A. Danchin. Hon-yaku: a biology-driven bayesian methodology for identifying translation initiation sites in prokaryotes. *BMC Bioinformatics*, 8(1):47, 2007.
- [8] N. Segata, E. Blanzieri, S. J. Delany, and P. Cunningham. Noise reduction for instance-based learning with a local maximal margin approach. *J. Intell. Inf. Syst.*, 35:301–331, October 2010.
- [9] D. B. Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 293–301. Morgan Kaufmann, 1994.
- [10] D. Wilson. Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *IEEE Transactions on Systems, Man, and Cybernetics*, 2-3:408–421, 1972.
- [11] D. R. Wilson and T. R. Martinez. Reduction techniques for instance-based learning algorithms. *Mach. Learn.*, 38:257–286, March 2000.