Dynamic Numerosity Reduction for Mining-Based Agent Learning

Khamisi Kalegele,[†1] Johan Sveholm,[†2]
Hideyuki Takahashi,[†1,†2] Kazuto Sasai,[†1,†2]
Gen Kitagata[†1,†2] and Tetsuo Kinoshita[†1,†2]

When coupling data mining and intelligent agents, one of the crucial challenges is the need for the knowledge extraction process to be lightweight enough so that even resource (e.g. memory, CPU etc.) constrained agents are able to extract knowledge. In this work we propose a method of achieving lightweight knowledge extraction using dynamic numerosity reduction to allow for agents to retrieve training data subsets of different sizes based on their available resources. To compensate for the possible loss of data integrity due to training data size reduction, a novel ranking method, Level Order (LO) ranking, is proposed for selection of data representatives.

1. Introduction

The symbiosis[1)] of Data Mining (DM) and Agent Technology (AT), in order to overcome the breakdown-with-complexity problem of AT'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 logics of DM and AT. In this paper, we deal with the first challenge. There are two approaches of achieving a Lightweight KE (LKE); the use of smaller training data (TD) sets or algorithm improvement. Reducing numerosity of TD is one way to reduce size of sets in order to meet the available capacity of a resource contrained agent. However, reducing the sets beyond a certain level is known to have negative effects on performance (e.g. KE accuracy). We propose a numerosity reduction based method to deal with this problem with no or minimum performance loss.

†1 Graduate School of Information Sciences, Tohoku University
†2 Research Institute of Electrical Communication, Tohoku University

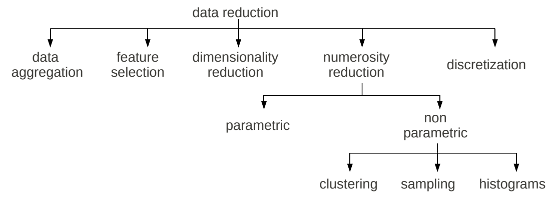


Fig. 1 Taxonomy of data reduction approaches

TD sets reduction methods (known by names like noise or data reduction, subset selection etc.) are in five categories[2)], shown in Fig. 1. In this paper we explore numerosity reduction based on sampling. Methods other than numerosity reduction are too related to data structure for a generic reduction approach. Sampling is preferred for LKE during mining-based agent learning [⋆1] over other numerosity reduction methods for two reasons. First, it is non parametric and therefore there is no extra cost of creation and recreation of data estimation models. Second, among non-parametric methods, sampling cost increases linearly with sample size while reduction or selection cost in other methods increases exponentially.

Sampling-based numerosity reduction methods, like other reduction methods, are characterized in terms of the following proporties.

- Selection method: A way of selecting examples, e.g. random.
- Selection type: Condensation type if the selection method seeks to retain border examples and edition type if it seeks to retain central examples.
- Fitness function or test: A criteria which the selected examples are to meet.
- Search direction: Whether a subset selection is incremental or decremental.
- Stratification: Independent selection from each stratum contained in TD set.
- Others: Related to the used KE algorithm, e.g. distance function.

The proposed sampling-based TD numerosity reduction method for mining-based agent learning has two main advantages over others. It improves relia-

⋆1 A learning process whereby an agent learns a knowledge model from training examples.

bility of the extracted knowledge by using ordered selection method as opposed to random selection. Our ordered selection method, called Stratified Ordered Selection (SOS), is based on ranking of examples using a novel ranking scheme called Level Order (LO) ranking. SOS allows for on-demand selection of subsets of desired sizes which enhance or preserve KE performance. Additionally, SOS provides room for class balancing during subset retrieval. Class balancing removes class imbalances. Class imbalance is a well known machine learning problem whereby training examples belonging to one of the classes contained in a TD set are significantly many compared to another class.

In the next section, we compare and contrast sampling-based numerosity reduction methods. We also present an overview of how mining-based agent learning have been approached in existing multi-agent frameworks. In section 3 we describe our proposed method in detail. Section 4 presents evaluation of the proposed method and finally section 5 concludes the paper.

2. Related Work

There are two viewpoints; subset selection and DM-integrated agent approach.

2.1 Sampling-based subset selection

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

Table 1 shows four (three existing methods and SOS) sampling-based methods of subset selection. All are incremental random sampling methods. One crucial shortfall of random-based sampling methods is that they are not very reliable. There is no guarantee that they will always give a performance enhancing (or preserving) subset when applied on the same TD set. For instance, SRS (Stratified simple Random Sample)[2] involves random selection of examples from each of the mutually disjoint strata (e.g. a classes) of a TD set. While SRS ensures a representative subset like its other peers (SRSWOR-Simple random sample without replacement, SRSWR-Simple random sample with replacement), it can neither regenerate same result nor does it explicitly attempt to retain decisive (e.g. boundary, central) examples of a class. Cluster (cluster sample) is very similar to SRS. The only difference is that in cluster sample, if there are C mutually disjoint strata, selection of examples can result into s clusters, where $s < C$.

Table 1 Sampling methods and characteristics

Approach	Fitness Test	Selection Method	Selection Type	Stratification	Search Direction	Distance Function
cluster	-	random	-	yes	incremental	n/a
SRS	-	random	-	yes	incremental	n/a
RMHC	accuracy	random	-	no	incremental	manhattan
SOS	-	ordered	hybrid	yes	incremental	minkowski

Methods like RMHC (Random Mutation Hill Climbing) improves reliability by using a fitness test. In RMHC, although randomly selected, examples are only kept if they enhance or preserve accuracy. The use of a fitness test during subset selection imposes extra computational cost which in mainstream DM is not a big problem because selection is only done once for a TD set. However, in resource constrained environments where agents have differing resources, subset selection is done many times with varying subset sizes. We therefore find RHMC less appropriate for mining-based learning by a resource constrained agent.

The SOS method addresses these shortfalls which are experienced by other sampling methods. It uses an ordered (pre-defined order) selection method in order to improve reliability. The method does not employ a fitness test at selection time in order to keep the cost low. It is both a condensation and an edition method so that the generated subsets enhance or preserve KE performance even better. To ensure a representative subset like in SRS, the method also employs a stratification approach, hence the name Stratified Ordered Selection (SOS).

2.2 DM-integrated agent approaches

In most existing DM-integrated agent frameworks, a provisioning for on-demand TD subsets selection is not given. In ABLE[3], the idea of LKE is only highlighted without further details. In Agent Academy[4], Data Miner provides few insufficient TD sets reduction functionalities, as far as LKE is concerned.

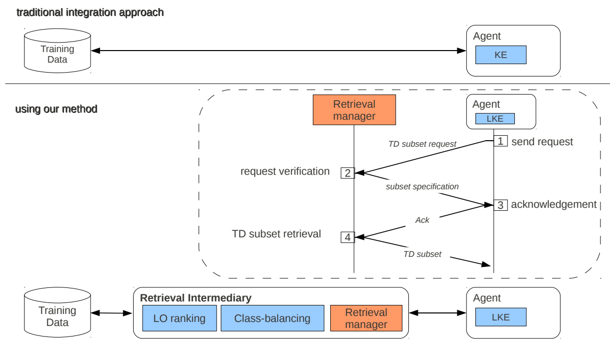


Fig. 2 Retrieval intermediary and TD subset retrieval

2.3 Benchmarking

In general, previously published evaluation works[5),6)] reveal that RMHC, AkNN (All k Nearest Neighbor) and DROP3 (Decremental Reduction Optimization Procedure 3) offer excellent balance between size reduction and accuracy. Among these best performing methods, RMHC is the only sampling-based method and therefore the only possible benchmarking candidate. However as discussed in subsection 2.1, RMHC is less appropriate for LKE. We therefore retreat to SRS as our benchmarking method. SRS is well known for its inexpensiveness and ability to generate representative subsets. Moreover, contrary to all other methods but inline with the proposed method, SRS has the ability of generating multiple subsets of varying sizes as well as room for class balancing.

3. The Proposed Method

Our proposed method aims at allowing agents to retrieve TD subsets of varying sizes based on available resources. Fig. 2 shows an approach to mining-based agent learning in which a retrieval intermediary implements the proposed SOS to allow dynamic and on-demand retrieval of subsets. Pre-ordering of examples is achieved by using Level Order (LO) ranking.

3.1 TD Subset Retrieval

Fig. 2 shows four-step negotiation between an agent and the retrieval manager.

1. Send request: An agent sends its desired size and class balance.
2. Request verification: The manager verifies the desired size and class balance. It checks whether the desired size is achievable, whether the amount of sampling to achieve it is reasonable and re-adjusts the specifications accordingly. Finally, the new or verified subset specifications are sent to an agent.
3. Acknowledgement: An agent acknowledges if it is satisfied with the proposed specifications. Otherwise it goes back to step 1.
4. Subset retrieval: The manager retrieves a subset using SOS selection.

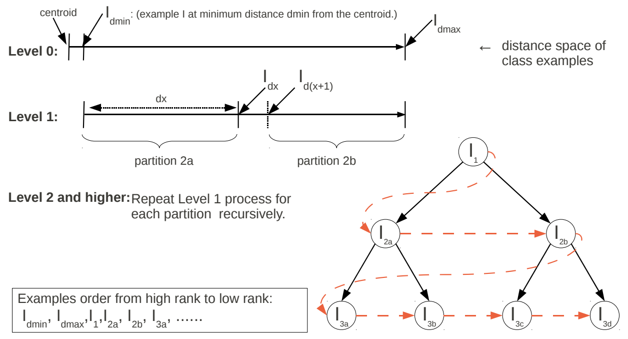
3.2 TD Subset Selection

The SOS method is stratified because the desired examples are selected independently from individual classes. It is ordered because examples are selected in a pre-defined order based on LO ranking. During SOS, class balancing is allowed to compensate for performance losses associated with size reduction and class imbalances. The desired class balancing is achieved by either oversampling the minority class using synthetic examples or undersampling the majority class. A minority class is a class with fewer examples and a majority class is a class with many examples. These two sampling procedures are achieved as follows.

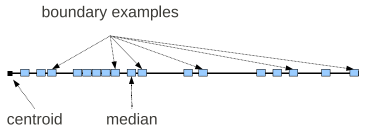
- *Oversampling*: Oversampling is achieved by creating new synthetic examples using the SMOTE[7)] technique. The process involves four steps.
  1. An example is chosen at random from within the class to be oversampled.
  2. Its five nearest neighbors are selected.
  3. A desired number of points are selected at random along the line segments joining the randomly selected example and any two of its neighbors.
  4. New examples at each of the selected points in step 3 are then created.
- *Undersampling*: Undersampling is achieved by selecting only the desired number of highly LO-ranked examples from the class to be undersampled.

3.3 LO Ranking scheme

In our description we adopt the term “representatives” to refer to “selected examples”. Fig. 3(a) describes the LO ranking scheme, an intra-class ranking scheme which seeks to identify representatives which broaden class representation by retaining both central and boundary examples. Representatives are selected in levels. Suppose N class examples in their distance space where I_{dx} is an example at a distance dx from the centroid. We refer to this as *level zero* of representation where the prime central example is the closest-to centroid (I_{dmin} : $dmin = min\{d1, ..., dN\}$) and the prime border example is the furthest-from


centroid
I dmin: (example I at minimum distance dmin from the centroid.)
I dmax
Level 0:
← distance space of class examples
I dx
I d(x+1)
dx
Level 1:
partition 2a
partition 2b
Level 2 and higher: Repeat Level 1 process for each partition recursively.
I 1
I 2a
I 2b
I 3a
I 3b
I 3c
I 3d
Examples order from high rank to low rank:
I dmin, I dmax, I 1, I 2a, I 2b, I 3a, ...

(a) Ranking steps



(b) Examples in a distance space

Fig. 3 Level Order ranking

the central example (I_{dmax} : $dmax = max\{d1, ..., dN\}$). This scheme considers I_{dmin} and I_{dmax} as *level zero* representatives, as shown in Fig. 3(a). Other representatives are established as follows.

- *Level one* representative: Using measure of central tendency on distance, an ideal representative would have been the median example, expressed below.

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

  However, since SOS seeks to retain boundary examples also, the median is not always necessarily at the boundary. Consider examples in a distance space in 3(b). The median is not at the boundary but an example next to it is. Therefore, we select, as *level one* representative, an example $I_1 = I_{dx}$ which maximizes the measure shown in Eq. 1 and partitions the distance space into 2a and 2b. In Eq. 1, $\frac{min(\#2a,\#2b)}{max(\#2a,\#2b)}$ ensures that a representative is closer to the median and $d(x+1) - dx$ ensures that a representative is at the boundary.

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

- *Level two and higher* representatives: The *Level one* process is repeated on partitions 2a and 2b separately to obtain I_{2a} and I_{2b} as *level two* representatives which subsequently divide their respective partitions into {3a and 3b} and {3c and 3d} respectively. This is then repeated recursively until the desired number of representatives is reached.

Level one and *higher* representatives are then made into a binary tree using their respective distances with *level one* representative as a root node as shown in Fig. 3(a). Except for *level zero* representatives, examples selection order (i.e. ranking) is determined in accordance to level order traversal. The order in Fig. 3(a), is indicated by dotted arrows ($I_{dmin}, I_{dmax}, I_1, I_{2a}, I_{2b}, I_{3a}, ...$).

LO ranking uses Minkowski distance (Eq. 2) space. Where p is the order of the Minkowski metric. Our choice is due to the fact that two of the commonly used distance functions ($p = 1$ for Manhattan distance and $p = 2$ for Euclidean distance) are contained in Minkowski space. Hence more flexibility.

$$D(\mathbf{x}, \mathbf{y}) = \left[\sum_{i=1}^{n} |x_i - y_i|^p\right]^{\frac{1}{p}} : \mathbf{x} = (x_1, x_2, ..., x_n), \;\; \mathbf{y} = (y_1, y_2, ..., y_n). \qquad (2)$$

4. Experiments and Evaluation

We evaluated SOS on five well known machine learning TD sets, shown in Table 2. They are all two-class problem sets from datasets repository of the University of California, Irvine. There are only two classes in the sets (a minority and a majority class). The minority-majority classes ratio is called class balance or just balance. Numbers of features in the datasets are shown in the last column of the table. Enclosed in brackets, after the number of features, are numbers of real-valued features, integer-valued features and nominal features, respectively. We used these data sets to investigate the following.

I *Optimal value of p, order of the Minkowski metric*: Five different values of p (1,2,3,5,7) were compared for average performance on all TD sets.

II *Effects of SOS on KE performance*: Baseline performances on original TD

Table 2 Training Data (TD) sets

Name	Description	Size	Balance	#Features
Page-blocks	Blocks of document page layout.	5472	10%:90%	10 (4/6/0)
Pima	Pima Indians diabetes data.	768	35%:65%	8 (8/0/0)
Spam	Spam filtering data.	4597	39%:61%	57 (57/0/0)
Segment	Image segmentation data	2308	14%:86%	19 (19/0/0)
Yeast	Cellular localization sites of proteins.	1484	11%:89%	8 (8/0/0)

Table 3 Simulation parameters and environments

Parameter or Environment Description	Values
KE algorithm	C4.5
Subsets sizes (% of original)	0,5,...,95
Model Validation	10-fold cross validation
Environments	Programming=JAVA, Learning=WEKA, Agent =ABLE

sets were first computed. Then for each set, subsets were generated on-demand, used for agent training and the resulting models evaluated.

III *Comparison between SOS and benchmarking choice (SRS)*: We repeated the procedure in II using SRS and compared the results.

4.1 Parameters, Enviroments and Performance Metrics

Table 3 shows parameters used in our simulations. For a C4.5 classifier, classification accuracy is the most important metric for KE performance. This is also argued by Segata et. al[5]. Nonetheless, we also investigated Area under ROC curve (AUC) and F-Measure of the resulting models. The ROC curve is created by plotting true positive rate against false positive rate. AUC and F-Measure are single numerical metrics commonly used to compare model performances[2].

Preparations of subsets involved random division of TD sets into 10 disjoint sets of the 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 or SRS was applied.

4.2 Results and Analysis

I *Optimal value of p*: Table 4 summarizes our findings. 5 was marginally found to be an optimal value for p. It was used for the rest of our simulations.

Table 4 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
Page-blocks	97.03	**97.15**	96.87	96.63	96.96
Pima	79.31	76.90	77.72	**79.66**	78.26
Spam	93.20	93.30	93.64	**93.65**	93.56
Segment	99.27	98.79	99.09	**99.28**	99.14
Yeast	78.04	78.24	78.67	**79.13**	78.77

Table 5 Performance metrics on original TD sets

Dataset	avg. accuracy (%)	avg. F-Measure	avg. AUC
Page-blocks	97.13	0.971	0.942
Pima	73.70	0.735	0.761
Spam	93.24	0.932	0.940
Segment	99.31	0.993	0.988
Yeast	76.01	0.754	0.743

Table 6 F-Measure, AUC and execution times

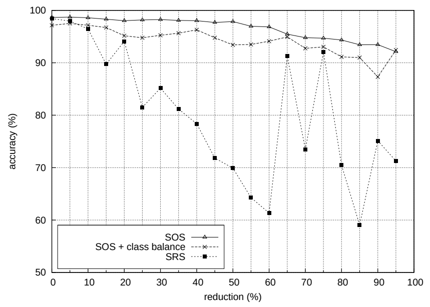
	SOS (average values)			SRS (average values)		
Dataset	F-Measure	AUC	time	F-Measure	AUC	time
Page-blocks	**0.952**	**0.914**	158.22	0.852	0.823	140.24
Pima	**0.791**	**0.822**	41.97	0.734	0.746	16.74
Spam	**0.923**	**0.924**	229.15	0.783	0.760	129.97
Segment	**0.994**	**0.989**	49.77	0.913	0.891	45.78
Yeast	**0.746**	**0.730**	40.41	0.741	0.672	28.45

II *Effects of SOS on KE performance*: Table 5 shows baseline performance of the KE algorithm on the original TD sets. Fig. 4 shows accuracy levels of extracted models for all TD sets. Due to space limitation, we only show the effects of class balancing on one TD set (page-blocks) in Fig. 4(f).

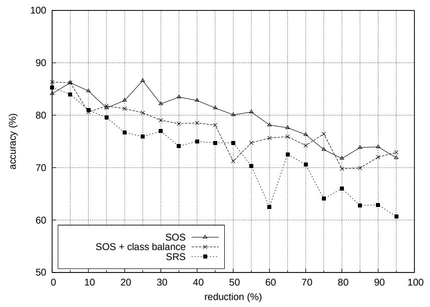
III *Comparison between SOS and benchmarking choice (SRS)*: Fig. 4 comparatively shows accuracy levels for both SOS and SRS generated subsets. Other metrics (F-Measure, AUC and execution time) are summarized in Table 6.

From the evaluation results, the following can be observed.

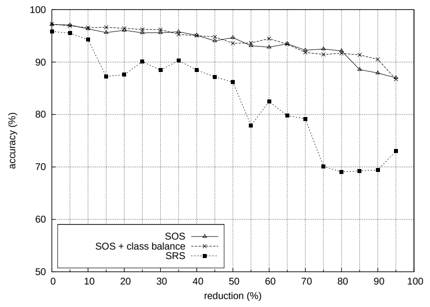
- SOS leads to better than baseline performance even with 0% reduction.
- Class balancing helps minority class especially when the subsets get smaller (Fig. 4(f), from about 60% reduction).



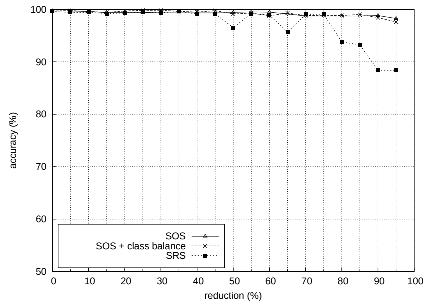
(a) Page-blocks



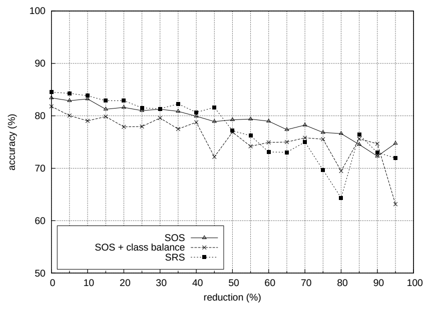
(b) Pima



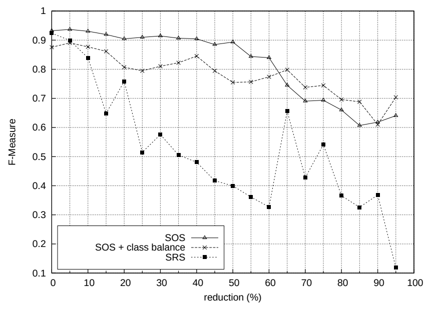
(c) Spam



(d) Segment



(e) Yeast



(f) Effects of class balancing

Fig. 4 Accuracy and F-Measure

- In general, SOS outperforms SRS except in terms of execution time (Table 6). Longer execution times for SOS are attributed to traversal of the binary tree used in LO ranking to store examples order.
- SOS preserves (Fig. 4) and enhances (Fig. 4(b), 4(c), 4(e)) performance.

5. Conclusion

We have presented our sampling-based method (SOS) of reducing numerosity of TD sets for mining-based agent learning. We have shown performance enhancement ability of SOS and its superiority against a benchmarking method. SOS leads to better performance even with 0% reduction. SOS can therefore be used not only as a subset selection method for resource constrained agent learning, but also as a performance enhancement method even in mainstream learning.

Further empirical analysis of performance of SOS on other datasets of differing nature and composition is ongoing as well as review of other relevant numerosity reduction methods.

References

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

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

3) Bigus, J. P.: The agent building and learning environment, *Proceedings of the fourth international conference on Autonomous agents*, New York, NY, USA, pp. 108–109 (2000).

4) Mitkas, P., Symeonidis, A., Kechagias, D., Athanasiadis, I. N., Laleci, G., Kurt, G., Kabak, Y., Acar, A., Dogac, A.: An agent framework for dynamic agent retraining: Agent Academy, *In eBusiness and eWork, 12th annual conference and exhibition*, 16-18 October (2002).

5) 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).

6) Garcia, S., Derrac, J., Cano, J. R. and Herrera, F.: Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2010.

7) 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).