A Buffering Strategy to Avoid Ordering Effects in Clustering

Luis Talavera[1] and Josep Roure[2]

[1] Universitat Politècnica de Catalunya
Departament de Llenguatges i Sistemes Informàtics
Campus Nord, Mòdul C6, Jordi Girona 1-3
08034 Barcelona, Catalonia, Spain
talavera@lsi.upc.es
[2] Escola Universitària Politècnica de Mataró
Departament d'Informàtica de Gestió
Avda. Puig i Cadafalch 101-111
08303 Mataró, Catalonia, Spain
roure@eupmt.upc.es

Abstract. It is widely reported in the literature that incremental clustering systems suffer from instance ordering effects and that under some orderings, extremely poor clusterings may be obtained. In this paper we present a new strategy aimed to mitigate these effects, the Not-Yet strategy which has a general and open formulation and it is not coupled to any particular system. Results suggest that the strategy improves the clustering quality and also that performance is limited by its limited foresight. We also show that, when combined with other strategies, the Not-Yet strategy may help the system to get high quality clusterings.

1 Introduction

Ideally, intelligent agents should possess the ability of adapting their behavior to the environment over time through learning. Thus, learning methods should be able of updating a knowledge base in a continual basis as new experience is gained. Particularly, if an agent performing a *clustering* task [5] should be able of using its learned knowledge to carry out some performance task at any stage of learning, the conceptual scheme should evolve as every new instance is observed without simultaneously processing previous instances. This sort of clustering is often referred to as *incremental clustering*. As noted by Langley [8], there can be several interpretations of incremental learning. In the remainder of this paper, we will assume that a clustering method is incremental if inputs one instance at a time, does not reprocess previous instances and maintains only one conceptual structure in memory.

Incremental clustering, as defined above, has to rely on some sort of hill climbing strategy which triggers small modifications of the knowledge base as new instances are observed. This way of incorporating single instances into the cluster structure makes incremental systems to be sensitive to instance order, as widely reported in the clustering literature [4, 6, 7, 8, 9].

Let I be an instance and P be a partition
Let E be the expected utility/confidence of adding I to P
Let α be the Not-Yet threshold value
if $E \geq \alpha$ **then** add(P,I)
  else add_NY_buffer(I)
endif

Table 1. Not-Yet control strategy

We say that incremental clustering algorithms exhibit ordering effects when they may yield different cluster structures when the same instances are presented in different orders. In some cases, they even can produce very poor quality clusterings. The problem lies in that a hill climbing strategy may narrow too much the search through the clustering space in a manner that initial observations may lead to a clustering scheme which does not reflect the real structure in the domain. In the worst case, the system might never be able of reaching a good clustering despite of gaining new experience.

2 The Not-Yet strategy

Since our goal is to (at least partially) solve the instance ordering problem while maintaining the incremental nature of clustering systems, we propose a solution to be applied during the clustering process. We refer to it as the *Not-Yet* strategy and it has a general and open definition. The strategy states that the incorporation of instances will be deferred if they are in either one of the following two cases, a) we do not expect the utility of the resulting clustering after incorporating the instance to be improved, and b) we do not have confidence enough about how the instance should be included in the existing clustering. The Not-Yet strategy assumes the existence of a buffer which stores instances that have not been incorporated into the clustering and some criterion to decide the moment in which the buffer is cleared.

In Table 1 an algorithmic formulation of the Not-Yet control strategy is shown. We introduce a parameter, α, which constraints the amount of utility or confidence required for an instance in order to be incorporated into a clustering. If we assume E to be always positive, when α is 0, the Not-Yet control strategy simply reduces to the original clustering algorithm, which becomes a particular case of a more general strategy. Although a system using the proposed strategy should perform better, it must be noted that the Not-Yet has a limited foresight, so it may perform roughly the same in the worst case.

Complexity when using our strategy will vary from system to system depending on the cost of effectively incorporating an instance and computing the expected utility or confidence of adding the instance. However, most clustering systems use some quality function to decide the best choice when an instance

Let I be an instance and P be a partition
Let $M1$,$M2$ be the best and second best CQF
Let α be the Not-Yet threshold value
if (1 - $M2/M1$) $\geq \alpha$ **then** add(P,I)
  else add_NY_buffer(I)
endif

Table 2. Implementation of the Not-Yet control strategy for the experiments.

is observed, so it is likely that this function is a good candidate to measure the amount of utility/confidence. If this is the case, complexity is augmented by a constant factor. This factor is dependent on the times every instance is considered for incorporation into the cluster structure.

3 Experiments

In order to empirically evaluate the Not-Yet strategy we conducted several experiments using four well-known datasets of the UCI repository. Since the clustering task is an unsupervised learning task, we have treated labels just as another attribute. In the experiments we assume a general model of hierarchical incremental clustering using two basic operators, one for creating a new class and another to incorporate an instance to an existing class. A concept hierarchy grows incrementally as new instances are observed after applying one of these operators according to the value of some *cluster quality function* (CQF). This is a typical model of incremental clustering using a hill climbing strategy which estimates the goodness of applying the available operators and chooses the best option, without reconsider any decision made. Particularly, this model corresponds to the one used in the COBWEB system [3]. The measure of *category utility* used in this system is also used in the experiments as the CQF. We used a COBWEB-like clustering strategy because it is simple, well-known and it has been applied (or augmented) in several learning systems [1, 7].

As stated before, we embed the basic control procedure into another one implementing the Not-Yet strategy. The buffer is cleared at the end of the clustering process and, therefore, an unlimited size is assumed. The strategy does not incorporate an instance to a cluster structure if there is not evidence enough to decide between the available operators. As shown in Table 2, for each instance, a ratio between the second best CQF and the best one is computed. We consider that an operator does not yield a significant better clustering than others if the confidence is below the α threshold, which is in the [0,1] range. In addition, instances in the buffer were randomized before reprocessing.

Experiments were performed on both random and worst case orderings. Table 3 shows the results obtained with both orderings using several values of the α parameter for the Not-Yet strategy. The zero value for this parameter corre-

		Basic				Augmented			
		CQF		Buffered inst.		CQF		Buffered inst.	
	α	Rand.	Worst	Rand.	Worst	Rand.	Worst	Rand.	Worst
soyb. small	0.0	1.49 (0.14)	0.91 (0.11)	0.00	0.00	1.62 (0.01)	1.12 (0.19)	0.00	0.00
	0.05	1.52 (0.12)	0.99 (0.13)	0.03	0.46	1.61 (0.03)	1.42 (0.17)	0.21	0.37
	0.10	1.49 (0.15)	1.02 (0.17)	0.11	0.80	1.60 (0.05)	1.52 (0.10)	0.52	0.80
	0.15	1.48 (0.15)	1.04 (0.12)	0.40	0.88	1.60 (0.04)	1.58 (0.05)	0.69	0.89
	0.20	1.46 (0.12)	1.10 (0.11)	0.81	0.90	1.61 (0.04)	1.58 (0.06)	0.83	0.90
	0.25	1.48 (0.13)	1.12 (0.11)	0.93	0.91	1.61 (0.03)	1.59 (0.05)	0.93	0.93
soyb. large	0.0	1.00 (0.10)	0.63 (0.14)	0.00	0.00	1.17 (0.02)	0.95 (0.14)	0.00	0.00
	0.05	1.03 (0.09)	0.65 (0.14)	0.02	0.06	1.16 (0.03)	1.06 (0.11)	0.22	0.40
	0.10	1.05 (0.10)	0.72 (0.10)	0.09	0.81	1.16 (0.03)	1.13 (0.07)	0.52	0.74
	0.15	1.07 (0.10)	0.73 (0.08)	0.31	0.96	1.16 (0.03)	1.16 (0.03)	0.82	0.96
	0.20	1.05 (0.11)	0.76 (0.08)	0.75	0.98	1.17 (0.02)	1.16 (0.03)	0.90	0.98
	0.25	1.01 (0.11)	0.77 (0.09)	0.92	0.99	1.16 (0.03)	1.16 (0.03)	0.96	0.99
house	0.0	1.29 (0.28)	0.85 (0.19)	0.00	0.00	1.61 (0.00)	1.43 (0.12)	0.00	0.00
	0.05	1.33 (0.26)	0.82 (0.17)	0.00	0.01	1.61 (0.00)	1.50 (0.11)	0.01	0.30
	0.10	1.35 (0.25)	0.84 (0.15)	0.01	0.09	1.60 (0.05)	1.53 (0.10)	0.06	0.48
	0.15	1.34 (0.25)	0.83 (0.16)	0.08	0.72	1.60 (0.04)	1.58 (0.07)	0.17	0.80
	0.20	1.37 (0.25)	0.83 (0.13)	0.29	0.98	1.61 (0.01)	1.60 (0.01)	0.38	0.93
	0.25	1.35 (0.27)	0.85 (0.13)	0.64	0.98	1.61 (0.00)	1.60 (0.01)	0.66	0.99
zoo	0.0	1.05 (0.12)	0.67 (0.17)	0.00	0.00	1.17 (0.03)	0.95 (0.14)	0.00	0.00
	0.05	1.06 (0.12)	0.67 (0.14)	0.02	0.10	1.17 (0.03)	1.05 (0.10)	0.13	0.39
	0.10	1.05 (0.13)	0.73 (0.11)	0.06	0.54	1.17 (0.03)	1.10 (0.08)	0.30	0.67
	0.15	1.03 (0.15)	0.77 (0.12)	0.19	0.87	1.16 (0.03)	1.15 (0.05)	0.53	0.87
	0.20	1.00 (0.17)	0.76 (0.08)	0.45	0.92	1.17 (0.03)	1.16 (0.03)	0.70	0.93
	0.25	1.03 (0.17)	0.78 (0.09)	0.78	0.93	1.16 (0.03)	1.16 (0.03)	0.83	0.94

Table 3. Clustering results. Averages and standard deviations over 50 trials

sponds to the original algorithm without buffering any instance. Although we are using a hierarchical clustering method, the CQF is given only for the top level, which is expected to score highest [4]

Results demonstrate that instance ordering has a critical effect in cluster quality. The quality of discovered clusterings consistently drops in a 35-40% when bad orderings are used, being far from the optimal values found in the literature [4]. The Not-Yet strategy modestly improves results in the random case, but note that the good performance of the original clustering procedure on random orderings lets little room for improvement. With bad orderings, the strategy improves the poor scores obtained with the basic algorithm up to a 20%. However, results are still far from the ones obtained with random orderings. Table 3 shows that the number of buffered instances increases as the α value does, and also that this increment is faster with bad orderings. This demonstrates the ability of the Not-Yet strategy for detecting bad instance orders.

In our second experiment, we augmented the basic clustering procedure by adding the *merge* and *split* operators used in COBWEB. Briefly, the merge

operator modifies a hierarchy by combining two existing clusters while the split operator breaks existing clusters into smaller ones. Split and merge operators provide a sort of backtracking to the clustering system. However, due to the fact that they are only triggered by new observations, their impact is still limited.

Again, there is almost no chance to improve the results obtained with random orderings. However, our strategy allows the system to reach high quality clusterings under bad instance orderings. These results suggest that combination of several strategies may yield better results than their isolate application.

In both experiments, the most important improvements are obtained at the expense of maintaining a big Not-Yet buffer, i.e., with high α values. It may appear counterintuitive with the idea of incremental learning to maintain a buffer of about the 90% of the instances in the dataset. A solution could be to limit the Not-Yet buffer in a way that it would be cleared several times during learning. It is not clear how this limitation would affect performance and further experiments should be made.

4 Related work

Several works have approached the ordering problem in incremental clustering, although this research has mainly benefited from two particular approaches. Lebowitz first introduced the idea of *deferred commitment* within the framework of his UNIMEM conceptual clustering system [9]. Our proposal extends Lebowitz's work by decoupling the buffering strategy from any particular system. Also, we have introduced the α parameter, that allows to see the original algorithm as a particular case of the new control strategy. We think that this formulation should help in applying the strategy to any existing algorithm without any major changes.

The second related work is the application of this strategy to the LINNEO$^+$ clustering system [2, 10]. This work contains the basic ideas proposed here, but again the application is tuned for an specific system and the problems studied are deeply related to a particular clustering strategy.

Also, we have to mention relevant Fisher's work on iterative optimization of clusterings [4]. This work explores several methods for iteratively improving clustering quality, showing that among these methods some exhibit an optimum performance. But recall that these methods operate by reprocessing the whole dataset and violate the constraints stated for incremental clustering.

5 Conclusions and future work

A new buffering strategy has been proposed to deal with ordering effects. We think that the formulation of the strategy is open in the sense that it is not coupled with any particular evaluation function or algorithm. Since the strategy is applied during the clustering process, its impact over the entire conceptual structure is limited as the experiments have shown. Nevertheless, in the worst

case ordering, the Not-Yet strategy consistently improved the quality of obtained clustering, specially when combined with additional operators. It is difficult to assess the quality of this improvement beyond the simple quantitative analysis in terms of the CQF. For some applications it can suppose an important improvement in terms of understandability or performance while for others it may be imperceptible. These benefits are obtained at the expense of maintaining a large Not-Yet buffer. Since an incremental system has to be able of using the acquired knowledge for some performance task at any learning stage, we have to assume that the system has also to be able of quickly incorporate the buffered instances before actuating. If this was not possible, the system should carry out the performance task with the partial knowledge structure learned so far. Future work will study the Not-Yet performance limiting the buffer size. In practice, buffer size will be limited by the amount of instances that the system can manage in a reasonable amount of time before entering in 'performance mode' and this time will be dependent on the particular application.

We plan to extend this work by considering a number of different, probably more complex, buffering strategies. Extensions studying the order of instances in the buffer, the criterion to reprocess instances or the number of times instances may be reprocessed appear to be promising topics for further research.

Acknowledgments. We thank the anonymous referees for their insightful comments which helped to improve the quality and understandability of the paper.

References

1. J. R. Anderson and M. Matessa. Explorations of an incremental, bayesian algorithm for categorization. *Machine Learning*, (9):275–308, 1992.
2. J. Béjar. *Adquisición automática de conocimiento en dominios poco estructurados.* PhD thesis, Facultat d'Informàtica de Barcelona, UPC, 1995.
3. D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, (2):139–172, 1987.
4. D. H. Fisher. Optimization and simplification of hierarchical clusterings. *Journal of Artificial Intelligence Research*, (4):147–180, 1995.
5. D. H. Fisher and P. Langley. Conceptual clustering and its relation to numerical taxonomy. In W. A. Gale, editor, *Artificial Intelligence and Statistics.* Addison-Wesley, Reading,MA, 1986.
6. D. H. Fisher, L. Xu, and N. Zard. Ordering effects in clustering. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 163–168, 1992.
7. J. H. Gennari, P. Langley, and D. Fisher. Models of incremental concept formation. *Artificial Intelligence*, (40):11–61, 1989.
8. P. Langley. Order effects in incremental learning. In P. Reimann and H. Spada, editors, *Learning in humans and machines: Towards an Interdisciplinary Learning Science.* Pergamon, 1995.
9. M. Lebowitz. Deferred commitment in unimem: waiting to learn. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 80–86, 1988.
10. J. Roure. Study of methods and heuristics to improve the fuzzy classifications of LINNEO$^+$. Master's thesis, Facultat d'Informática de Barcelona, UPC, 1994.