Not-Yet: a local strategy to avoid ordering effects in clustering

Luis Talavera
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

Josep Roure
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 general strategy aimed to mitigate these efects, the Not-Yet strategy, which has a general and open formulation and it is not coupled to any particular system. In addition, we propose a classification of strategies to avoid ordering effects which clarifies the benefits and disadvantages we can expect from the proposal made in the paper as well from existing ones. A particular implementation of the Not-Yet strategy is used to conduct several experiments. Results suggest that the strategy can improve the clustering quality and also that performance is limited by its local nature. We also show that, when combined with other local strategies, the Not-Yet strategy may help the system to get high quality clusterings. The observed benefits and limitations suggest future work under the proposed framework.

Keywords: ordering effects, incremental clustering.

1 Introduction

Ideally, intelligent agents should posses 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 sustained, continual basis as new experience is gained. When the learning task is a *clustering* task, the learner's goal is to discover a conceptual structure underlying a set of given observations (Fisher & Langley, 1986). If an agent performing a clustering task should be able of use its learned knowledge to carry out some performance task at any stage of learning, the conceptual scheme must 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 (1995), 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.

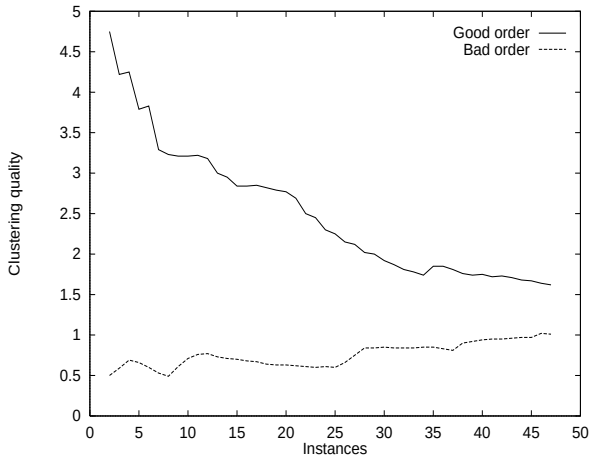

Figure 1: Cluster quality for different instance orders

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 (Fisher, 1995; Fisher, Xu & Zard, 1992; Gennari, Langley & Fisher, 1989; Langley, 1995; Lebowitz, 1988). 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. Figure 1 shows an example of this behavior. The graph shows the evolution of the value of a clustering quality function with every new observed instance for two instance orders. When instance ordering is good, the graph reveals that high quality clusters are initially constructed because instances covers very differents clusters underlying the domain which are easy to discriminate by the system. Later, the system clustering gratefully converge to the quality global maximum. This maximum is below the initial obtained scores since additional instances may present a more uncertain cluster membership so that initial confidence is reduced. On the other hand, a bad order may lead the system into a completely different learning path far away from the optimum clustering. As shown in the graph, in the worst case, the system might never be able of reaching a good clustering despite of gaining new experience.

2 Avoiding ordering effects

Research in incremental clustering has approached the ordering effects problem by using several strategies. In this section, we will give a classification of strategies to avoid ordering effects with regard to two different dimensions, namely, the stage in the clustering process in which they are applied and the scope. Our aim is to clarify the potential benefits that a given strategy can provide as well as its limitations and, also, to provide a general context to place in our own research.

If we divide the strategies according to the stage in the clustering process in which they are applied, we can distinguish among three application points: before clustering, during clustering or after clustering. Methods which are applied *before clustering* can only be used when all or a great amount of data is known before hand. The idea is to arrange the instance order in such a way that favors the system search process to reach the best classification. It is seen that when dissimilar objects are consecutively presented, the resulting classification is much better than when similar objects are presented together (Fisher, Xu & Zard, 1992; Fisher, 1995). This occurs because,

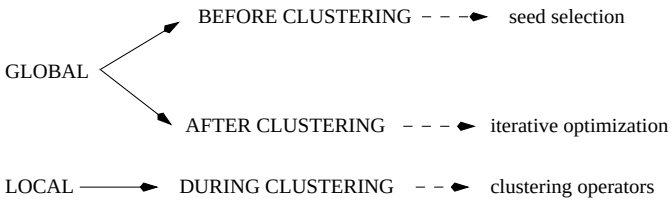

Figure 2: Classification of strategies to avoid order effects and some examples

in the former case, initial observations are from different areas of the observation description space leading initial clusters to reflect these areas, while in the later, a skewed cluster structure may evolve. Thereafter, the clustering system may not be able to recover when further instances from other parts of the description space are observed. An typical example of this procedure are *seed selection* methods which select 'seed' observations from data, growing clusters around them (Michalski & Stepp, 1983; Béjar, 1995).

When constructing a cluster structure in an incremental fashion, only two basic operators are needed, one to *create* a new cluster given an instance and another to *incorporate* an instance to an existing cluster. Using these two operators, in theory, any clustering structure could be built. However, once an object is consolidated into the structure, the clustering system cannot move it using the two basic operators, therefore the system cannot recover from previously taken bad decisions when further experience is gained. The clustering system may be provided with additional operators to be applied *during clustering* in order to be able of recovering from bad instance orders. These operators can be viewed as providing some sort of backtracking capabilities to the system without having a memory of previous knowledge structures. A classical example of clustering operators are the *merge* and *split* operators of the COBWEB system (Fisher, 1987).

Finally, we can tackle with ordering effects using *after clustering* strategies. These strategies take an initial partition and try to improve it by iteratively making changes to the structure until some stopping condition holds. The aim of the iterative algorithms is to reach the maxima of some objective function that evaluates the cluster structure quality. These methods not only need the dataset to be known in advance, but they also rely on a continuous reprocessing of the clustering structure.

From another point of view, but related to the previously used one, we can distinguish strategies according to the scope of their application and effects. For a *global strategy* we mean a method which uses information about the whole domain and, therefore, needs to know a great amount of data before hand, possibly including an extensive reprocessing of instances. In contrast, a *local strategy* acts upon a small piece of knowledge assuming that small, local changes will contribute to improve global clustering quality and usually will be triggered only by new observations. Figure 2 shows the classification of strategies discussed so far. Clearly, global strategies are expected to give significantly better results than local ones since we cannot guarantee local changes to have a sufficiently strong effect upon the global knowledge structure. However, global strategies may be undesirable under the incremental learning assumption because they extensively reprocess the instances in the dataset.

3 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. As noted before, this is a local strategy and so implies a trade-off between the degree of clustering quality improvement and the preservation of the incremental properties of a system.

Our solution is based on a simple and intuitive idea. We refer to it as the *Not-Yet* strategy and it has a general, informal and open definition. The strategy states that we will defer incorporation of instances which are in either of the following two cases, a) we do not expect the utility of the

Let I an instance
Let P 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

resulting clustering after incorporating the instance to be improved, and b) we do not are confident enough about how the instance must 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. We will not specify neither this criterion nor specific metrics for measuring utility or confidence since our aim is to propose a general enough framework to fit into several incremental approaches. In the experiments, however, we will propose an example of how the Not-Yet strategy can be effectively implemented.

In Table 1 an algorithmic formulation of the Not-Yet control strategy is shown. We introduce a new term α, which is the Not-Yet threshold, to allow different degrees of utility/confidence be used in specific implementations. The α value can be seen as a parameter to the Not-Yet control strategy which constraints the amount of utility or confidence required for an instance in order to be incorporated into a clustering. If we assume the E value to be always positive, when α is 0, the Not-Yet control strategy simply reduces to the original clustering algorithm. As the α value increases, the control strategy strongly constraints the incorporation of new instances to the cluster scheme.

Figure 3 shows the expected behavior of a clustering system using the Not-Yet strategy. This is the same graph shown in Fig. 1 including two extreme cases of instance ordering. We have added a new curve reflecting the evolution of the clustering quality when using the Not-Yet strategy with a bad ordering. We have noted before that good orderings promote high quality clusterings to be created in the early stages of the process so that the system may slowly converge to the global optimum. Bad orderings have the opposed effect. Low quality clusterings are created at the beggining which strongly condition the rest of the process. In the graph we show an optimal behavior of a system using the Not-Yet strategy with a bad order. At the beginning, the system behaves in the same manner that the original algorithm, but as it buffers some instances –around the fifth instance– the cluster quality increases. When quality reaches its maximum, it converges with the good ordering curve. The graph clearly shows how our strategy tries to reach the same 'good learning path' starting from a different learning point. Note, that the horizontal axis measures the number of instances effectively incorporated into the cluster structure. So, when using the Not-Yet strategy, a quality maximum is reached around the 25th instance but, at this point, other instances may have been observed and buffered as well. In fact, in the particular case of the graph shown, only around 25 instances passed the Not-Yet 'filter' the first time they were observed, the rest of them being buffered. When incorporating buffered instances a decrease of the clustering quality is observed but this behavior is similar to that of the system with a good ordering.

The discussed graph shows an optimistic behavior of the Not-Yet strategy in the sense that the system is able to reach almost the same clustering quality with bad and good orderings. However, since we are using a local strategy, we can expect any level of improvement between the good and bad orderings curves. In general, the Not-Yet strategy should perform better (either with

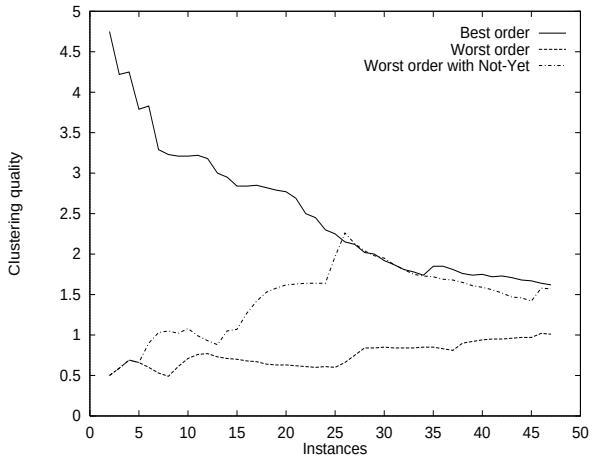

Figure 3: An example of the Not-Yet strategy effect on clustering quality

modest or important improvements) than the original strategies or perform roughly the same in the worst case. On the other hand, 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 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, in the worst case, every instance would be considered for incorporation twice, the first time it is observed, and when emptying the Not-Yet buffer. So, the algorithmic complexity of the initial system remains the same since we are just adding a constant factor.

4 Experiments

In order to empirically evaluate the Not-Yet strategy applied to clustering algorithms, we conducted several experiments using four well-known datasets of the UCI repository and described in Table 2. 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 the two basic operators mentioned earlier, 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, not reconsidering any decision made so far. Particularly, this model corresponds to the one used in the COBWEB system (Fisher, 1987). The measure of *category utility* used in this system is also used in the experiments as the CQF. We have chosen a COBWEB-like clustering strategy because it is simple, well-known and it has been applied (or even augmented) in many learning systems (Anderson & Matessa, 1992; Gennari, Langley & Fisher, 1989; Kilander & Jansson, 1993). Table 2 gives also the estimated optimal values for the CQF obtained with good orderings. This value gives an approximate upper bound on the performance of the system on every dataset.

The Not-Yet strategy is implemented in this model as follows. Following the general scheme in Fig. 1, we embed the basic control procedure into another procedure which implements the Not-Yet strategy assuming that we will not incorporate an instance to a cluster structure if we do not have evidence enough to decide between the available operators. In terms of the CQF this means that the quality of the clustering obtained by using the best operator is not significantly better than the quality of the clustering that would be obtained by applying another one. To decide when some operator yields significantly better clusterings, the Not-Yet threshold is used.

Dataset	Instances	Attributes	Optimal CQF
Small soybean	47	36	1.62
Large soybean	307	36	1.17
Voting records	435	17	1.61
Zoo	101	17	1.18

Table 2: Description of the four datasets used in the experiments

Let I an instance
Let P a partition
Let $M1$,$M2$ the best and second best CQF
resulting from applying the available operators
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 3: Implementation of the Not-Yet control loop in the clustering method used in the experiments.

For each new instance, a ratio between the second best CQF and the best one is computed. When the difference between the quality of the best operator ($M1$) and the quality of the second best one ($M2$) is high, the ratio approaches to 0. Note that when this ratio yields 1, none of the operators appear to be better than the other which is exactly the case we want to deal with. In order to let the ratio show the degree of confidence in the best choice we use the expression $1 - M2/M2$ which increases at the same time as the difference between the best choices does. As shown in Table 3, we relax this constraint and consider that an operator does not yield a significant better clustering than others if the confidence is below the Not-Yet threshold, which is in the [0,1] range. Clearly, when the threshold is 0, we have the original algorithm since the confidence is always greater or equal than this value and instances will be always incorporated to the clustering.

4.1 Experiment 1: clustering with basic operators

In our first experiment, we used the general clustering procedure outlined before using only the two basic operators for creating new clusters or incorporating instances to existing ones. Experiments were performed on random orderings as well worst case orderings. These later orderings were generated by randomly selecting an instance, and iteratively selecting as the next instance the most similar to the previous one. Similarity is measured by a simple metric which counts the number of shared attribute values between instances. Table 4 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 corresponds to the original algorithm without buffering any instance. Since we are using a hierarchical clustering method, the CQF is given only for the top level, which is expected to score highest.

The results demonstrate that the basic clustering procedure performs relatively well on random orderings scoring a CQF between the 80-90% of the optimal one. However, 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 shown in Table 2.

The Not-Yet strategy modestly improves results in the random case giving approximately similar results for all the α values. As noted earlier, the Not-Yet is a local strategy and therefore, has a limited application foresight. Note also that the good performance of the original clustering

		CQF		Buffered instances	
	α	Random	Worst	Random	Worst
soybean small	0.0	1.49 (0.14)	0.91 (0.11)	0.00	0.00
	0.05	1.52 (0.12)	0.99 (0.13)	0.03	0.46
	0.10	1.49 (0.15)	1.02 (0.17)	0.11	0.80
	0.15	1.48 (0.15)	1.04 (0.12)	0.40	0.88
	0.20	1.46 (0.12)	1.10 (0.11)	0.81	0.90
	0.25	1.48 (0.13)	1.12 (0.11)	0.93	0.91
soybean large	0.0	1.00 (0.10)	0.63 (0.14)	0.00	0.00
	0.05	1.03 (0.09)	0.65 (0.14)	0.02	0.06
	0.10	1.05 (0.10)	0.72 (0.10)	0.09	0.81
	0.15	1.07 (0.10)	0.73 (0.08)	0.31	0.96
	0.20	1.05 (0.11)	0.76 (0.08)	0.75	0.98
	0.25	1.01 (0.11)	0.77 (0.09)	0.92	0.99
voting records	0.0	1.29 (0.28)	0.85 (0.19)	0.00	0.00
	0.05	1.33 (0.26)	0.82 (0.17)	0.00	0.01
	0.10	1.35 (0.25)	0.84 (0.15)	0.01	0.09
	0.15	1.34 (0.25)	0.83 (0.16)	0.08	0.72
	0.20	1.37 (0.25)	0.83 (0.13)	0.29	0.98
	0.25	1.35 (0.27)	0.85 (0.13)	0.64	0.98
zoo	0.0	1.05 (0.12)	0.67 (0.17)	0.00	0.00
	0.05	1.06 (0.12)	0.67 (0.14)	0.02	0.10
	0.10	1.05 (0.13)	0.73 (0.11)	0.06	0.54
	0.15	1.03 (0.15)	0.77 (0.12)	0.19	0.87
	0.20	1.00 (0.17)	0.76 (0.08)	0.45	0.92
	0.25	1.03 (0.17)	0.78 (0.09)	0.78	0.93

Table 4: Clustering results with basic operators. Averages and standard deviations over 50 trials

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%, the *voting records* dataset being the exception. However, results are still far from the optimal ones. Table 4 shows that, obviously, the number of buffered instances increases with the α value, but also that this increment is faster with bad orderings. These results demonstrate the ability of the Not-Yet strategy for detecting bad instance orders. It is worth to notice that in these situations, the number of buffered instances is very high, showing the difficulties from recovering from initial bad orders and confirming the importance of instance ordering in incremental clustering. Only a few initial instances may completely determine the learning path followed by the system in the remainder of the learning process.

4.2 Experiment 2: clustering with merge/split operators

In our second experiment, we augmented the basic clustering procedure previously used by adding two operators. These operators are the *merge* and *split* operators used by Fisher 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 has still local effects in the cluster structure. Results were obtained using again random and bad orderings and several α values and are shown in Table 5.

The augmented clustering method shows itself to be very robust in random orderings, approximating the optimal CQF value for all datasets. Again, worst case orderings lead the system to decrease the quality of discovered clusters, showing the limitations of applying operators during clustering in mitigating ordering effects. As in the previous experiment, the Not-Yet strategy has

		CQF		Buffered instances	
	α	Random	Worst	Random	Worst
soybean small	0.0	1.62 (0.01)	1.12 (0.19)	0.00	0.00
	0.05	1.61 (0.03)	1.42 (0.17)	0.21	0.37
	0.10	1.60 (0.05)	1.52 (0.10)	0.52	0.80
	0.15	1.60 (0.04)	1.58 (0.05)	0.69	0.89
	0.20	1.61 (0.04)	1.58 (0.06)	0.83	0.90
	0.25	1.61 (0.03)	1.59 (0.05)	0.93	0.93
soybean large	0.0	1.17 (0.02)	0.95 (0.14)	0.00	0.00
	0.05	1.16 (0.03)	1.06 (0.11)	0.22	0.40
	0.10	1.16 (0.03)	1.13 (0.07)	0.52	0.74
	0.15	1.16 (0.03)	1.16 (0.03)	0.82	0.96
	0.20	1.17 (0.02)	1.16 (0.03)	0.90	0.98
	0.25	1.16 (0.03)	1.16 (0.03)	0.96	0.99
voting records	0.0	1.61 (0.00)	1.43 (0.12)	0.00	0.00
	0.05	1.61 (0.00)	1.50 (0.11)	0.01	0.30
	0.10	1.60 (0.05)	1.53 (0.10)	0.06	0.48
	0.15	1.60 (0.04)	1.58 (0.07)	0.17	0.80
	0.20	1.61 (0.01)	1.60 (0.01)	0.38	0.93
	0.25	1.61 (0.00)	1.60 (0.01)	0.66	0.99
zoo	0.0	1.17 (0.03)	0.95 (0.14)	0.00	0.00
	0.05	1.17 (0.03)	1.05 (0.10)	0.13	0.39
	0.10	1.17 (0.03)	1.10 (0.08)	0.30	0.67
	0.15	1.16 (0.03)	1.15 (0.05)	0.53	0.87
	0.20	1.17 (0.03)	1.16 (0.03)	0.70	0.93
	0.25	1.16 (0.03)	1.16 (0.03)	0.83	0.94

Table 5: Clustering results with merge/split operators. Averages and standard deviations over 50 trials

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

As it was the case with the previous experiment, the most important improvements are obtained at the expense of maintaining a big Not-Yet buffer. 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 possible solution would 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 will affect performance and further experiments should be made.

Finally, we have to point out that both experiments have shown that the clustering quality tends to improve as we use higher α values. This result could be obviously inferred from the Not-Yet formulation and reflects that, presumably, no optimal value exists. However, we have noted that as the α value, so does the number of buffered instances. This may suggest that the optimal α value for a particular application would be a trade-off between these two aspects.

5 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 (Lebowitz, 1988). Our proposal is similar in spirit to that of Lebowitz, but we have clarified the terms and decoupled the formulation from any specific clustering system.

The second related work (from which the Not-Yet name is borrowed) is the application of this strategy to the LINNEO$^+$ clustering system (Roure, 1994). 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.

Although devoted to global methods, we have to mention relevant Fisher's work on iterative optimization of clusterings (Fisher, 1995). 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 reprocessing the whole dataset and violate the constraints stated for incremental clustering. This sort of strategies are useful from the viewpoint of a data analysis task in which the entire dataset is avalaible before hand so that we are not constrained by strict incremental constraints.

6 Conclusions and future work

We have presented a classification of strategies to avoid ordering effects in clustering with regard to two related dimensions, namely, the point of the clustering process in which they are applied and the scope of the strategy. This classification aims to clarify the benefits and disadvantages which we can expect from the application of existing or newly proposed strategies.

A new local strategy, the Not-Yet strategy, has been proposed to deal with ordering effects. We think that the formulation of the strategy is simple and open in the sense that it is not coupled with any particular evaluation function or algorithm. As a local strategy, this strategy has a limited impact over the entire conceptual structure as the experiments have shown. In the worst case ordering, the Not-Yet strategy consistently improved the quality of obtained clustering. It is difficult to asess 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 other it may be imperceptible. On the other hand, when coupled with another local strategy such as the merge/split operators, the Not-Yet strategy allows the clustering system to reach an optimum quality clustering even with worst orderings. However, 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 is not possible, the system should carry out the performance task with the partial knowledge structure learned so far. Future work should 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 dependant of the particular application.

It is unclear how the proposed procedure scales to large datasets such as those typically referred to in *data mining* tasks (Fayyad, Piatetsky-Shapiro & Smyth, 1996). However, we think that the Not-Yet strategy may be an inexpensive and effective way of avoiding ordering effects since it is unlikely that a whole large dataset would present a worst ordering case. Rather, it probably will have bad ordered subsets, so that a large enough Not-Yet buffer will be able to deal with the problem. Note that the size which we have considered large for the buffer –around 400 instances– may be simply a small part of a very large dataset of thousands of instances. We plan to explore these issues in future work.

In this paper, we are concerned with incremental clustering, but we think that the proposed strategy may be useful under incremental supervised learning settings as well. Future work may help to asess the benefits of the strategy in a wider range of learning systems.

References

Anderson, J. R. and Matessa, M. (1992). Explorations of an incremental, bayesian algorithm for categorization. *Machine Learning*, (9):275–308.

Béjar, J. (1995). *Adquisición automática de conocimiento en dominios pco estructurados*. PhD thesis, Facultat d'Informàtica de Barcelona, UPC.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). Knowledge discovery and data mining: towards a unifying framework. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD96*, Portland, OR. AAAI Press.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, (2):139–172.

Fisher, D. H. (1995). Optimization and simplification of hierarchical clusterings. *Journal of Artificial Intelligence Research*, (4):147–180.

Fisher, D. H. and Langley, P. (1986). Conceptual clustering and its relation to numerical taxonomy. In Gale, W. A., editor, *Artificial Intelligence and Statistics*. Addison-Wesley, Reading,MA.

Fisher, D. H., Xu, L., and Zard, N. (1992). Ordering effects in clustering. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 163–168.

Gennari, J. H., Langley, P., and Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence*, (40):11–61.

Kilander, F. and Jansson, C. G. (1993). Cobbit - a control procedure for Cobweb in the presence of concept drift. In Bradzil, P. B., editor, *Proceedings of the European Conference on Machine Learning*, pages 244–261. Springer Verlag.

Langley, P. (1995). Order effects in incremental learning. In Reimann, P. and Spada, H., editors, *Learning in humans and machines: Towards an Interdisciplinary Learning Science*. Pergamon.

Lebowitz, M. (1988). Deferred commitment in unimem: waiting to learn. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 80–86.

Michalski, R. S. and Stepp, R. E. (1983). Learning from observation: Conceptual clustering. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial intelligence approach*, pages 331–363. Morgan Kauffmann, San Mateo, CA.

Roure, J. (1994). Study of methods and heuristics to improve the fuzzy classifications of LINNEO$^+$. Master's thesis, Facultat d'Informática de Barcelona, UPC.