A Comparison of Batch and Incremental Supervised Learning Algorithms

Leonardo Carbonara[1], and Alastair Borrowman[2]

[1] Database Marketing Team, British Telecom,
PP411.7, 120 Holborn, London EC1N 2TE, UK
leonardo.carbonara@bt.com
[2] Department of Computing Science, University of Aberdeen,
King's College, Aberdeen AB24 3UE
abj@csd.abdn.ac.uk

Abstract. This paper presents both a theoretical discussion and an experimental comparison of batch and incremental learning in an attempt to individuate some of the respective advantages and disadvantages of the two approaches when learning from frequently updated databases. The paper claims that incremental learning might be more suitable for this purpose, although a number of issues remain to be resolved.

1 Introduction

An important problem in KDD is deriving data models from frequently updated databases. Real-world databases, such as those held by credit card or telecommunications companies are constantly being enriched with new information, while old data is discarded. Although supervised learning algorithms have been used extensively to infer classification models from data, a number of issues still remain to be resolved to make them cope effectively with rapidly changing data. These algorithms can be divided into two distinct categories: incremental algorithms are able to build and refine a model in a step-by-step basis by incorporating new training cases into the model as they become available, whereas non-incremental algorithms work in batch mode. Incremental learning systems include Utgoff's (1989) ID5R and his more recent ITI (Utgoff, 1994), and Kalles & Morris's (1996) TDIDT. Among the most renown non-incremental learning algorithms are Michalski's (1980) AQ and Quinlan's (1993) C4.5.

Similarly to incremental learning systems, theory revision systems take as input an approximately correct model of a domain, usually expressed as a set of rules, and a set of training instances, and by means of a predefined set of refinement operators revise the rules to make them consistent with the training set. Although these systems cannot be considered pure learning algorithms as they assume that an initial model is provided, they share nevertheless with incremental learning algorithms the

ability to incorporate new data into an existing model. Hence, for the purpose of this paper they will be treated as incremental learning systems. Theory revision systems include Ginsberg's (1988) SEEK2, EITHER (Ourston & Mooney, 1991), PTR (Koppel, *et al.*, 1994), and Carbonara & Sleeman's (1996) STALKER.

The most appealing property of non-incremental learning algorithms is possibly their ability to achieve high classification accuracy, due to the fact that processing a batch of instances help them improve generalisation and avoid over-fitting. Incremental learning, on the other hand, is desirable because knowledge revision is typically much less expensive than knowledge creation.

This paper presents a comparison of batch and incremental learning in an attempt to individuate some of the advantages and disadvantages of these approaches when learning from frequently updated databases. The paper is organised as follows. Section 2 discusses some of the properties of the batch and incremental learning systems. Section 3 presents some experimental results comparing the performance of a batch learning system, C4.5, an incremental learning system, ITI, and a theory revision system, STALKER. Section 4 summarises the results and gives some conclusions.

2 Batch vs. Incremental Learning

It is commonly thought that batch algorithms achieve higher classification rates than incremental learners, as, in general, batch learners seem to be less prone to problems such as over-fitting. In fact, they can exploit their "global view" of the data to generate more robust classifiers. Incremental algorithms, on the other hand, follow a more 'myopic' approach as they attempt to incorporate each single instance into the model. However, in the context of rapidly changing domains there are three major problems that can be identified with the batch approach:

1. *Abrupt transition between successive models.* When inferring a classification model from data, the learning algorithm selects the attributes to be included in the model using some heuristic, e.g., Quinlan's *information gain* (Quinlan, 1986). This heuristic often depends on the distribution of the possible values of the attributes across the instances. Since, when new cases are added to the initial database, this distribution might change, the new model derived from the updated database can be based on different attributes from those used in the original model. Hence, there is no clear and explicit relationship between two successive models, and it becomes difficult to keep track of the evolution of the model over time. Some methods that are being investigated to overcome this problem include feature selection, discretization, and boosting.
2. *Inconsistency over data.* The fact that the subsequent models produced by a batch algorithm are unrelated also means that cases which were correctly classified by an earlier model may be mis-classified by a subsequent model, and vice versa. In other words, there is no assurance whatsoever that, moving from one model to another, any kind of consistency over the data will be maintained. As we shall see below, consistency is not always a desirable property. However,

there must be some way to monitor and control the inconsistencies introduced so that the user is informed of the reasons why inconsistency is achieved on a number of cases.

3. *Time inefficiency*: when dealing with very large collections of data which are constantly being updated, generating a new model from scratch every time new instances are received is a very inefficient way to revise the model. Sampling is a possible solution to this problem, although in the case of very skewed data interesting concepts which have little statistical support may not be identified.

Incremental learning inherently solves the first two problems identified above:

1. *Smooth transition between successive models.* Since the new model is not generated from scratch using the now-augmented set of instances, but is a revision of the initial model, there is a clear relationship between the two models. By analysing the changes implemented it is possible to know exactly how the initial model was modified to produce the revised model.
2. *Consistency over data.* The incremental approach also assures that the changes in the model are truly incremental, i.e., the classification performance of the revised model is unaltered on the cases contained in the initial data set.

Whether the incremental approach can also solve the third problem, i.e., the time inefficiency of the batch approach, needs to be investigated further. Batch algorithms such as C4.5 work in linear time in the number of cases processed. Incremental learners usually require more computational effort to incorporate each instance into an existing model. However, it is not necessary that the sum of the incremental costs be less than the batch cost, as every time the incremental algorithms only work on the updates. Hence, if the model needs to be updated frequently, the incremental approach should be less expensive.

It has already be pointed out that the second property possessed by incremental learning, i.e. the ability to make changes to the model which are consistent with the previously seen instances may not always be desirable. This is the case when the concepts of interest depend on some *hidden context* that changes over time. Changes in the hidden context can induce more or less radical changes in the target concepts, producing what is generally known as *concept drift* (Schlimmer & Granger, 1986). Widmer & Kubat (1996) showed that incremental learners can be adapted to successfully detect, and react to, concept drift. Although it is not the purpose of this paper to investigate learning in the presence of concept drift, this is another reason why incremental learning may prove superior to the batch approach in rapidly changing domains.

In this section some of the properties of incremental and batch learners have been presented and discussed. The main advantage of batch algorithms seems to be their superior classification power, while the incremental approach appears to be preferable when it is likely that new data will become available after the initial classifier has been built. Since this is a common occurrence in practical applications of machine learning, where the continual collection of data is the norm, it is evident that incremental learners able to achieve high classification accuracy would constitute an appealing alternative to batch learners. In the next section, the results of some ex-

periments are presented which attempt to understand the proportion of the gap, if any, between the classification power of current batch and incremental algorithms.

3. Some experiments comparing batch and incremental learning systems

In this section the results of some experiments comparing the classification accuracy of the batch learning algorithm C4.5, the incremental learning algorithm ITI, and the theory revision system STALKER will be presented. C4.5 and ITI are two state-of-the art decision tree induction algorithms. STALKER has been shown to perform at comparable levels with other well known theory revision systems on a number of benchmark domains (Carbonara & Sleeman, 1996). Moreover, it uses an incremental algorithm to refine each incorrectly solved instance, its approach therefore being closer to pure incremental learners than other batch theory revision systems.

C4.5 probably does not need any introduction as it is one of the most widely used algorithm for batch induction of decision trees. C4.5 produces an initial decision tree using the *information gain* metric to select the test at each decision node. The tree is then pruned to prevent overfitting. The final tree can also be converted into a collection of simplified rules.

ITI (Utgoff, 1994) produces models of the data in the form of binary decision trees, that is each test at a decision node can be answered *true* or *false*. ITI can be used both in batch and incremental mode.

Theory revision is the task of automatically refining a domain theory usually expressed as a set of rules to make it consistent with a given set of training instances. The theory revision system STALKER (Carbonara & Sleeman, 1996) generates a set of alternative refinements to correct each incorrectly solved training case. These alternative corrections are tested against all the previously seen instances to select the one that achieves the highest classification accuracy. The best refinement is implemented and the process repeated for the next training instance. Since testing each alternative correction is computationally expensive, STALKER overcomes this problem by converting rules and instances into a *Truth Maintenance System* (TMS), which is then used to efficiently test the refinements.

In the next subsection the method used to compare the three algorithms is described. The results of experiments with four domains from the UCI repository of machine learning databases are then presented.

3.1. The evaluation method

As already noticed, STALKER is not a pure incremental learning system as it needs to be given as input an initial set of rules to be refined. Hence, to compare its performance with the two other systems the following method has been used. An initial ruleset was produced with C4.5 using a subset of the training instances. This set of rules was then refined by STALKER using the remaining training instances. The

classification accuracy of the resulting ruleset was then compared to that of the ruleset produced by C4.5 using all the training instances. ITI was tested in a similar way, by generating an initial tree using the system in batch mode and then adding the remaining instances to the tree. However, the same results would have been obtained by generating the final tree from all the instances in batch mode, as ITI's tree transposition operator ensures that the incremental algorithm generates the same tree as it would have been produced with the batch algorithm. The ratio of batch and incremental instances was different in each subset to enable the investigation of the relationship between the accuracy of the final data model produced and the number of instances used to build the initial batch model. The ratio, or step, between the number of batch and incremental instances was decided upon based on the total number of training instances. A ratio was chosen which produced 4 or 5 subsets for each dataset considered. The results presented were averaged over ten independent trials.

The three systems were tested on four datasets from the UCI Repository of Machine Learning Databases (Merz & Murphy, 1996). A brief description of the domains and the results of the experiments follow.

Tables 1 to 4 show the classification accuracy results for the experiments carried out with the above domains. The figures reported are:

- **Initial KB** is the accuracy achieved by the ruleset produced by C4.5 with the 'batch' subset of training data which was used as the initial theory for the STALKER experiments;
- **STALKER** is the accuracy achieved by STALKER using the 'incremental' subset of training data to refine the Initial KB;
- **C4.5** is the classification performance of C4.5 on the full set of training data (this explains why the curve is a straight line);
- **ITI Batch** represents the accuracy of the initial tree generated by ITI from the 'batch' data;
- **ITI Incremental** is the accuracy of the final tree obtained by ITI by incorporating into the initial tree the remaining 'incremental' training instances. As ITI is always able to accommodate all the training instances, this is also a straight line.

Table 1. Classification accuracy for the Breast Cancer domain

Training Data					
Data Split	Initial Training Data	ITI Batch	STALKER	ITI Incremental	C4.5 - 600 cases
100/500	92.40	93.58	96.50	100	97.95
200/400	93.90	95.04	97.43	100	97.95
300/300	94.07	96.70	97.93	100	97.95
400/200	94.85	97.99	98.05	100	97.95
500/100	95.60	99.17	99.20	100	97.95
Test Data					
Data Split	Initial Test Data	ITI Batch	STALKER	ITI Incremental	C4.5 - 600 cases
100/500	93.23	93.23	94.34	95.05	95.96

200/400	94.34	93.43	95.95	95.05	95.96
300/300	94.24	94.04	95.65	95.05	95.96
400/200	94.54	93.03	95.85	95.05	95.96
500/100	94.54	93.83	95.85	95.05	95.96

Table 2. Classification accuracy for the Splice domain

Training Data					
Data Split	Initial Training Data	ITI Batch	STALKER	ITI Incremental	C4.5 - 900 cases
150/750	77.72	82.368	93.48	100	95.04
300/600	83.15	90.268	94.37	100	95.04
450/450	86.51	93.102	96.40	100	95.04
600/300	88.47	95.289	95.97	100	95.04
750/150	90.20	97.881	98.67	100	95.04
Test Data					
Data Split	Initial Test Data	ITI Batch	STALKER	ITI Incremental	C4.5 - 900 cases
150/750	76.70	78.60	86.60	91.30	92.80
300/600	85.30	87.20	91.00	91.30	92.80
450/450	88.70	87.20	92.10	91.30	92.80
600/300	91.80	88.20	94.20	91.30	92.80
750/150	92.10	90.50	93.50	91.30	92.80

Table 3. Classification accuracy for the Adult domain

Training Data					
Data Split	Initial Training Data	ITI Batch	STALKER	ITI Incremental	C4.5 - 1000 cases
200/800	79.33	74.77	87.03	100	87.39
400/600	80.77	79.43	92.00	100	87.39
600/400	81.65	83.13	93.55	100	87.39
800/200	82.40	86.20	94.60	100	87.39
Test Data					
Data Split	Initial Test Data	ITI Batch	STALKER	ITI Incremental	C4.5 - 1000 cases
200/800	79.32	79.62	81.80	81.44	83.22
400/600	80.80	80.62	80.88	81.44	83.22
600/400	81.36	81.30	82.20	81.44	83.22
800/200	82.12	81.38	82.24	81.44	83.22

Table 4. Classification accuracy for the Diabetes domain

Training Data					
Data Split	Initial Training Data	ITI Batch	STALKER	ITI Incremental	C4.5 - 600 cases
100/500	70.76	72.90	75.80	100	81.97
200/400	72.35	78.23	79.90	100	81.97

300/300	73.50	84.47	81.63	100	81.97
400/200	75.40	90.33	86.15	100	81.97
500/100	73.20	94.77	86.90	100	81.97
Test Data					
Data Split	Initial Test Data	ITI Batch	STALKER	ITI Incremental	C4.5 - 600 cases
100/500	66.90	66.90	69.52	67.98	72.44
200/400	69.23	67.56	72.26	67.98	72.44
300/300	71.01	69.35	72.14	67.98	72.44
400/200	73.33	67.62	73.15	67.98	72.44
500/100	71.37	67.86	73.45	67.98	72.44

As can be seen from the tables above the three systems seem to achieve comparable levels of accuracy on all the domains considered. Perhaps the only domain where there is a noticeable difference is the Diabetes domain, ITI performing significantly worse than C4.5 and STALKER.

STALKER is particularly sensitive to the accuracy of the initial KB which it is given to revise. This is somehow consistent with the *minor tweaking* assumption under which most theory revision systems work, i.e. the supposition that the initial KB is 'approximately' correct, and only needs minor tweaking rather than a major overhaul. This makes theory revision systems prefer 'minimal' refinements, i.e. corrections that are considered to be the least radical according to some metric (e.g., syntactical complexity). It is possible that this bias towards minimal revisions prevents such systems from making the more drastic changes that may be needed to refine very inaccurate theories. However, it must be noticed that, starting with an accurate KB, in two domains (Splice and Diabetes) STALKER outperforms the other two systems. In the Adult domain, on the other hand, a superior performance to C4.5 on the training data is not matched by a similar result on the test data, perhaps indicating over-fitting.

Some other possible cases of over-fitting can be seen in the test results for Diabetes experiments where for the 300/300 split and the 400/200 split, respectively, ITI's and C4.5's initial trees are more accurate than the corresponding final trees.

ITI has the desirable feature that trees built incrementally are always the same as the batch trees generated from the same data. This is achieved by means of its tree restructuring operators. However, this implies that extra work must be done to reshape the incrementally built trees which can affect the system's time performance. In fact, in the above experiments the time taken by ITI to incrementally revise the initial trees is consistently higher than the time taken by C4.5 to build the final batch trees. (Note that both C4.5 and ITI are implemented in C, and were tested on the same machine.) Another appealing characteristic of ITI's is that it always accommodates all training instance. This, however, can cause the trees generated by ITI to be exceedingly complex, which in turn can lead to over-fitting. In fact, ITI's test results are always slightly less accurate than those for the other two systems. To overcome this problem pruning could be used to eliminate subtrees that overfit the data. Utgoff (1994) explains that no pruning technique was used in ITI to retain the

property that the same tree will be found for the same set of instances, independent of the order in which they are presented. Nevertheless, he suggests that a pruning method could be incorporated if, instead of actually discarding unwanted subtrees, one could mark nodes as 'virtually pruned'. Thus subtrees could be marked in and out of existence without the expense of destroying or reconstructing anything. When one arrives at a virtually pruned decision node, one treats it as a leaf, returning the corresponding class.

We shall conclude this section with some notes on the time performance of the algorithms compared. A direct comparison of timings was not possible as the systems were developed with different programming languages (C4.5 and ITI are written in C, while STALKER is implemented in Common Lisp). Hence, we shall limit this discussion to the computational complexity of the three algorithms. C4.5 is linear in the number of training instances processed. In the incremental learner ITI, the incremental cost is, in general, proportional to the number of nodes in the decision tree. The tree generally grows to its approximate final size early in the training, with the rest of the training serving to improve the selection of the test at each node. Of concern is whether the incremental training cost continues to grow even after the size of the tree has more or less stabilised. Experimental results on a number of domains with both symbolic and numeric variables (Utgoff, 1994) seem to suggest that ITI's training cost appears to be effectively independent of the number of training instances seen. In theory, however, this is not the case when numeric variables are involved. For each numeric variable at each node, a sorted list of the values observed is maintained, which is used to select the best test for that node. Hence, more training instances means a greater cost to maintain each such list. In the theory revision system STALKER, training time is split between generation of alternative refinements and testing of revised KBs against previously processed training instances. Refinement generation is a potentially exponential process when dealing with long rule chains, although in most cases takes a negligible amount of time. Testing of refined KBs with the Truth Maintenance System is in theory cubic in the number of cases, although experiments in a number of domains seem to suggest that the process is in fact quadratic in the number of cases represented in the TMS (Carbonara, 1996). In conclusion, it seems that C4.5 is more efficient than the other two incremental algorithms. However, as pointed out in Section 2, when dealing with frequently updated very large databases the cost of incrementally updating an initial model should be less than building the model from scratch every time.

4. Conclusions

This paper presented a comparison of incremental and batch learning algorithms. In the literature, batch learning is often praised for its ability to achieve high classification accuracy, but it is considered to be inefficient when applied to frequently updated, large databases. Not only should incremental learning obviate this short-

coming, but, as shown by the results of an experimental comparison, can also achieve comparable levels of accuracy to batch learning. Moreover, incremental learners also possess some other interesting properties, such as the ability to monitor changes in the data model and to detect concept drift, which are particularly desirable when dealing with rapidly changing domains.

Acknowledgments

The work presented in this paper was supported by the Data Mining Research Project and was carried out at the BT Labs, Ipswich. The authors wish to thank Gavin Meggs for his useful comments on an earlier version of this paper.

References

Carbonara, L. and Sleeman, D. (1996). Improving the Efficiency of Knowledge Base Refinement. In *Proceedings of the 13th International Conference on Machine Learning* (pp. 78-86), Bari, Italy: Morgan Kauffman.

Clark, P. & Nibblet, T. (1989). The CN2 induction algorithm. *Machine Learning*, 3, 261-283.

Ginsberg, A. (1988). Automatic Refinement of Expert System Knowledge Bases. Morgan Kaufmann, San Mateo, California.

Kalles, D. & Morris, T. (1996). Efficient Incremental Induction of Decision Trees. *Machine Learning*, 24, 231-242.

Koppel, M., Feldman, R., & Segre, A.M. (1994). Bias-Driven Revision of Logical Domain Theories. *Journal of Artificial Intelligence Research*, 1, 159-208.

Michalski, R.S. & Chilauski, R.L. (1980). Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *Policy Analysis and Information Systems*, 4, 125-160.

Merz, C.J., & Murphy, P.M. (1996). UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

Ourston, D. and Mooney, R. (1991). Changing the Rules: A comprehensive approach to Theory Refinement. *In Proceedings of the 8th National Conference on Artificial Intelligence* (pp. 815-820), Cambridge, MA: MIT Press.

Quinlan, J.R. (1986). Induction of Decision Trees. *Machine Learning 1*, 81-106.

Quinlan, J.R. (1993). C4.5: Programs for machine learning. San Mateo, California: Morgan Kauffman.

Schlimmer, J.C. & Granger, R.H. (1986). Incremental Learning from Noisy Data. *Machine Learning*, 1, 317-354.

Utgoff, P.E. (1989). Incremental Induction of decision trees. *Machine Learning, 4*, 161-186.

Utgoff, P.E. (1994). An Improved Algorithm for Incremental Induction of Decision Trees. In *Proceedings of the 11th International Conference on Machine Learning*, New Brunswick, NJ, Morgan Kauffman.

Widmer, G. & Kubat, M. (1996). Learning in the Presence of concept Drift and Hidden Contexts. *Machine Learning*, 23, 69-101.