An Integrated Approach to Mining Data Streams

Robert Munro
University of London
rmunro@soas.ac.uk

Sanjay Chawla
University of Sydney
chawla@it.usyd.edu.au

Abstract

The biggest shortcoming of efficient Bayesian learning is the attribute independence assumption. Current state-of-the-art learners weaken this to capture pairwise correlations with quadratic efficiency.

This paper details a recent advance in mining correlated data through the integration of techniques from the fields of outlier detection, clustering and supervised learning. The algorithm described here is able to learn and exploit correlations between a potentially unbounded number of attributes in one pass over the data, with strictly bounded memory and processing costs.

1 Introduction

It is typical to find attribute correlations (attribute dependencies) in most data. As streaming architectures often represent the feedback of multiple sensors recording the same objects/phenomena, strong correlations in the information provided by these sensors is common.

Naive Bayes is inherently a streaming algorithm, but gets its name from its inability to capture correlations. In the exploration of ways to relax Naive Bayes' attribute independence assumption, current state-of-art learners capture pairwise correlations with quadratic salability [23].

Decisions trees are the most popular type of stream mining algorithm [7, 11, 14, 8, 13], and while the rules they express are correlations between attributes, this is only a weak learning of correlations, as the knowledge of rules isn't generalised and is limited by the depth of the tree at a given node. In a streaming environment, the learner should not build the same model as in a batch processing environment, as the learner cannot assume that unseen data is identically and independently distributed. The distributions of values may shift, and within the values the significance of attribute correlations may also shift. Therefore, it is desirable for a learner seeking to capture attribute correlations in a streaming environment to produce a very general and adaptive model of any correlations [3].

1.1 Problem Definition

The problem can be formalised as follows:

1. **Given**:

   A high dimensional streaming data set.

2. **Find**:

   A representation of that set that can be used in the accurate classification of unlabeled streaming data.

3. **Constraints**:

   Unknown correlations: the attributes may be highly correlated, but these correlations are not known in advance.

   Concept drift: the attribute values may shift with respect to the classes. The correlations (dependencies) may shift with respect to the classes, even when the relative frequency of attribute values remains constant.

   Streaming architecture: the correlations need to be discovered and modelled in one pass of the data. The number of instances that may be stored in memory is bounded.

1.2 Contributions

This paper makes significant contributions to the fields of data stream mining and Bayesian learning:

1. An integration of clustering, outlier detection and supervised learning is possible such that it produces a powerful, scalable algorithm suitable for mining data streams.

2. Capturing and exploiting attribute correlations between a (theoretically) unbounded number of attributes is possible in Bayesian learning such that it guarantees linear salability.

1.3 Outline

Section 2 describes related work. Section 3 defines how clustering may be used to learn attribute correlations. Section 4 describes the architecture of the algorithm used here. Section 5 gives the results of testing, for learning attribute correlations and adapting to changes in correlations. Section 6 discusses the implications of the results for stream mining algorithms. Section 7 gives conclusions and discusses possible future directions.

2 Related Work

2.1 Data Stream Mining

The most well-known data stream mining algorithm is probably the Very Fast Decision Tree (VFDT) [7], which was updated to an adaptive model in [11]. This algorithm keeps multiple alternate subtrees learned on a window of data in order to dynamically update the model to adapt to concept shift. This has lead to a large volume of recent work describing the use of decision trees in data stream mining [14, 8, 13]. Most of the problems with these model are those generally associated with decision trees: they can be more prone to over-fitting than other types of learners and they are unstable, that is, small changes in the data may lead to large changes in the representation, which is a significant issue in an adaptive learning environment.

There has been some research into the use of clustering algorithms for incremental and data stream learning [10, 20, 16, 1]. As with most clustering algorithms, they don't seek to capture or exploit attribute correlations.

In [21] a successful ensemble technique was proposed, dividing the stream into sequential chunks and using classifiers including decision trees and Naive Bayes. As stated, Naive Bayes is inherently a streaming algorithm, and its extension to adaptive abilities in a streaming environment through ensemble techniques has also recently been shown to be successful in [15].

In [24] an incremental unsupervised mixture modeller was implemented, showing the success of outlier detection in streaming tasks such as network intrusion detection.

2.2 Bayesian Learning and Clustering

Extensions of Naive Bayes to learners that capture attribute correlations include Lazy Bayesian Rules (LBR) [25], Averaged One-Dependence Estimators (AODE) [23], and a number of Bayesian Networks [4, 6].

Both LBR and AODE learn linearly with respect to the number of instances, but LBR has an expensive $O(N^3)$ cost with respect to the number of attributes for classification, and AODE has $O(N^2)$ cost with respect to the number of attributes for both learning and classification. As such, their successful extension to a streaming environment would be very dependent on the number of attributes. They are further limited in that they may only find a correlation between an attribute and at most one other, but otherwise have been shown to perform at the level of a boosted decision tree over a number of data sets [23].

Bayesian Networks (Bayesnets) are another popular method for capturing attribute correlations. They perform particularly well when the correlated attributes are defined by the user, but can also successfully learn the correlations. The results here are compared to two well-known Bayesian Network algorithms described in [4] and [6]. One extension of Bayesnets to a streaming environment is described in [5].

The algorithm described here was developed from a batch supervised mixture modeller [18].

3 Building clusters as correlations

3.1 Proofs and Lemmas

Lemma 1: Clustering subsets of categorical data can capture attribute correlations.

Proof Sketch: Let $A = (A_1, A_2, \ldots, A_m)$ be a set of categorical attributes of arity $k_1, \ldots, k_m$ respectively. Let X be a relation defined on A, and let t be some probability threshold. Then potential correlations between the attributes in A can be discovered and exploited by clustering the elements of X.

The instances of relations X can be expressed as binary vectors in a $k = k_1 + k_2 + \ldots + k_m$ dimensional 0-1 vector space. If X is clustered using a Euclidean distance norm $|| \; ||$ and if x and y are instances of X, then:

$$||x - y||^2 = ||x||^2 + ||y||^2 - 2|x.y|$$

where $x.y$ is the dot-product of x and y in the k dimensional space.

This has previously been called the *clustering correlation*, and (as above) it is usually calculated as a correlation between two instances [17].

Letting y represent a cluster instead of an instance, it will have probabilistic values for all attribute values, and can therefore be treated as a 'fuzzy' instance. Therefore, the strength of the correlation X captured by the cluster y will be the aggregate distance of the instances in y, or for each instance again simply $x.y$.

If we only allow y to contain instances where $x.y$ is greater than the threshold t, then the value of t will govern the minimum strength of correlations captured in y.

Thus, the cluster is a representation of the correlation X, with its strength bounded by t.

Lemma 2: Attribute correlations may be learned and represented as the sum of disjunctive clusters.

Proof Sketch: There may be multiple sets of correlations corresponding to a single concept. The simplest example of this is the well-known XOR problem, where there are four sets of correlations representing two target classes.

From above, the instances below the probability threshold t (the outliers) are not discarded, as they may be early indicators of a new concept.

Outliers are cached and later examined to determine if a new concept exists. If so, a new cluster representing this new concept can be formed. As the new cluster contains instances that were outliers of the first cluster, they represent two disjunctive sets of correlations.

Lemma 3: Supervised learning may be used to optimise the strength of the correlations learned.

Proof Sketch: From above, the value of t will affect both the strength of the correlations mined, and potentially the number of clusters found. A small t will form a small number of weakly correlated clusters, while a high value for t will produce a large number of clusters representing highly correlated concepts.

The incremental supervised accuracy can be used to calculate an optimal value for t. When there are a larger number of incremental errors than outliers, the learner is over-generalising, therefore t is too small and needs to be raised. When there are a smaller number of errors than outliers, the learner is potentially overfitting, and so t needs to be reduced.

Incremental error will be a better measure of the learner's performance than a non-streaming algorithm's training set error, as the incremental error is calculated *before* that instance contributes to the data model.

Thus, by adapting t accordingly, it should converge on a value optimised for modelling correlations in a supervised task.

Additionally, it is possible that the optimal value for t will change as the data stream evolves, and so there is also the advantage of adapting the model to different strengths of correlations at different times.

3.2 Correlations and conditional probability

Correlations may exist in a data set that are not important with respect to a supervised classification - a repeated attribute is a good example. While Lemma 3 will converge on an appropriate level of correlation, these are the unsupervised correlations in that they are not explicitly calculated in terms of the conditional probability.

AODE and LBR do calculate correlations with respect to the conditional probability, which is why they only capture pairwise correlations at greater expense. In [22] they are described in terms of Bayesian networks, but as the correlations in the algorithm described here are captured by clustering, not explicit modelling, it does not lend itself to being described in terms of a Bayesian network.

4 Architecture

The architecture of our learner is based the Lemmas given in Section 3.

4.1 Components

There are two components to the architecture:

1. A set of clusters: for each target class, containing the frequencies of each attribute value within that cluster (but not the instances themselves).

2. An outlier cache: containing instances

There are four important variables:

1. An outlier threshold. (t from Section 3) This is the probability below which an instance is treated as an outlier

2. A cache size threshold. The (static) maximum size of the outlier cache, that is, the number of outliers seen before the cache is flushed.

3. An error count. An incremental count of the number of training instances that had a greater probability of being members of a class other than their own (calculated *before* the instance contributed to the data model).

4. An outlier count. An incremental count of the number of outliers seen.

Each target class is represented by one or more clusters. No assumptions are made about the number of clusters required to represent each target class. The metric for defining cluster membership derives from both Bayes theorem and current clustering techniques.

Given an instance d, with attribute values $d_{a_1} \dots d_{a_m}$ for attributes $a_1 \dots a_m$, and a cluster c with size $S(c)$. Assume attribute a_j has l distinct values $a_{j_1} \dots a_{j_l}$, and the count of attribute value a_{j_k} in c is $freq(c, a_{j_k})$. The probability of d being a member of c is given by:

$$\frac{1}{m} \sum_{i=1}^{m} \frac{freq(c, d_{a_i}) + 1}{S(c) + 1} \tag{1}$$

The probability of an instance belonging to a cluster is the percentage of attribute values in the cluster that are equal to the attribute values of that instance. In Bayesian learning, for each attribute value this is simply known as the probability of d_a given c_a.

4.2 Treatment of instances and outliers

When a training instance arrives, it goes to either a cluster or the outlier cache, determined by whether its maximum probability of cluster membership is below the outlier threshold t.

If that instance is above the threshold, it is assigned to the cluster for which it had the highest probability, and the cluster counts, $S(c)$ and $freq()$, are updated accordingly, with the instance itself removed from memory.

There are three reasons that an instance may appear to be an outlier in data stream mining:

1. It is a true outlier - either noise or the product of some, possibly rare, process not able to be modelled by the learner.

2. It is evidence of a new concept (this is the motivation for creating clusters from outliers).

3. It is part of an existing cluster, but labelled an outlier due to concept shift.

Outliers are not removed from memory, but added to the outlier cache.

4.3 Creation of clusters

All clusters are created within the outlier cache. When the outlier cache has reached the cache size threshold it is ‘flushed’ to create new clusters where possible.

When the cache is flushed, all instances are initially attempted to be added to the existing clusters, as an outlier may have been early evidence of concept drift for a cluster.

Then, all remaining outliers are initially assigned to a single new cluster. All the outliers that are below the outlier threshold for that cluster are then returned to the outlier cache and the new cluster is added to the set of clusters (with the instances themselves removed from the memory). If there are still outliers remaining in the outlier cache, this process is repeated.

In the research described here, the outlier cache size threshold was set at 1% of the total data set size. In a real streaming environment there is, of course, no known size of the data set, and so an appropriate limit will need to be chosen.

4.4 Outlier threshold

An assumption is made here: the number of outliers discovered should be approximately equal to the number of incremental errors.

This follows the dynamic adjustment of the outlier threshold t described in Lemma 3.

In the research described here, the adjustment to the threshold was always by 10% of the existing threshold. This seemed to be a good adaptive change for the data tested here (brief testing with other rates made little difference), but making the adaptation rate dynamic would be an additional improvement. For example, the use of momentum would be especially suited to streaming environments where shifts and changes in concepts are inherently bursty.

4.5 Discounting and concept drift

Correlation shift and dynamically adjusting the outlier threshold have already been discussed.

In general, most of the additional discounting and/or windowing methods that have been applied to previous clustering or Naive Bayes data stream learners can easily be applied to this architecture.

An appealing aspect of the distributed nature of this learner is that it is able to capture concept drift and discount at different rates for different parts of the model, meaning that the learner is not restricted assuming that one data-window will fit all concepts.

In the current model, the removal of clusters is implemented by recording the average and standard deviation of the gap between instances being assigned to each cluster. Whenever the gap for a given cluster has reached greater than 10 standard deviations (from Chebyshev’s theorem), that cluster is removed.

4.6 Processing Costs

The cost can be broken down into the following components: cost for per instance n, cost per attribute a and cost per cluster c.

The cost for all operations is linear. When an instance arrives, the worst case cost is when its probability in all clusters needs to be calculated: $O(nca)$.

When iterating through clusters, the highest probability for a cluster of the same class, and the highest probability for a cluster of a different class are searched for. If at some point a cluster with confidence p has been found, then if half the attributes of some other cluster have aggregated

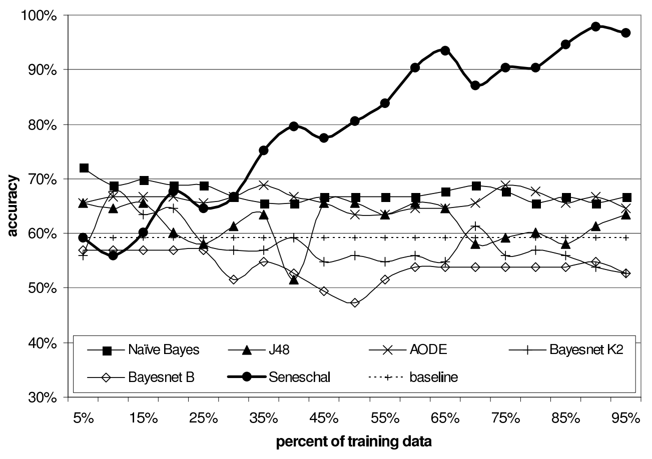

Figure 1. Accuracy on tic-tac-toe

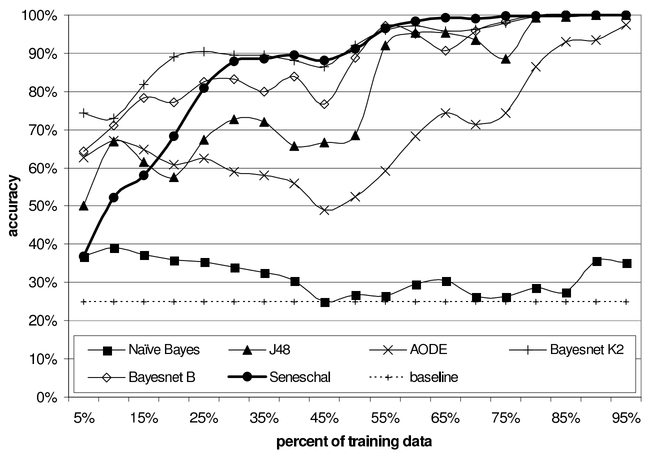

Figure 2. Accuracy on COR4

to less than $p/2$, the remaining attributes do not need to be calculated for that cluster. If the attribute values for all clusters are homogenous, then all but the correct cluster can be abandoned after measuring the first attribute, hence the best case is $O(n2c)$. Therefore, the more highly correlated the data is, the more efficiently the algorithm will perform.

An additional cost can be incurred in the flushing of the outlier cache. As the cache is iterated through once for the formation of each new cluster, each instance in the outlier cache may be additionally compared to all new clusters formed when the cache is flushed. This is still a cost of c, but two additional iterations of the cache may occur: one to confirm no new clusters can be formed, and then one to assign all remaining instances to a cluster.

4.7 Memory Costs

The memory cost for the set of clusters is simply ca multiplied by the arity of each attribute a.

The maximum memory cost for the outlier cache with a cache size threshold of T is Ta.

The only unbounded variable with respect to memory is the number of clusters c. It is difficult to imagine many situations where more memory wasn't available for more clusters, but c could simply be capped in such cases.

5 Testing

Testing was undertaken to compare this algorithm's performance to other streaming and non-streaming Bayesian learners, and to analyse its ability to adapt to shifts in correlations in a streaming environment.

5.1 Accuracy

The testing of learning algorithms on most UCI data sets is not appropriate for testing the mining of data streams [15]. While most UCI set do contain correlated data [23], it is difficult to distinguish whether given reductions in error are the results of correlated or non-correlated attributes. The exception to this that *is* well suited for benchmarking correlations in a streaming environment is the tic-tac-toe set.

As stated, in a streaming environment, the learner cannot assume that the same distributions will be evident in the unseen data, and so not only is the data unseen, but within that data the significance of attribute correlations may also shift. As such, a learner seeking to capture attribute correlations in a streaming environment needs to produce as general a model of attribute correlation as possible. The tic-tac-toe set concept displays this in both correlation and shift. The correlations *are* the classes, as the data represents every possible finishing position of a tic-tac-toe game, with the classes indicating whether or not ‘x’ won the game, that is, whether a correlation exists between 8 possible sets of 3 values across 9 attributes. The data was generated systematically, and the split between training and testing data is weighted in frequency more towards some possible sets of wins then others. This is why batch processing algorithms such as AODE perform much better when a stratified cross-validation is performed across the entire set (74% vs 68%) [23]. The algorithm described here performs approximately the same in both cases, classifying without error once about 90% of the data has been seen (ties were simply recorded as incorrect).

Figure 1 gives the accuracy and learning rate of a number of the Bayesian learners described in Section 2. Of these, only Naive Bayes and Seneschal are streaming learners. *Bayesnet B* is the Bayesian network described in [4],

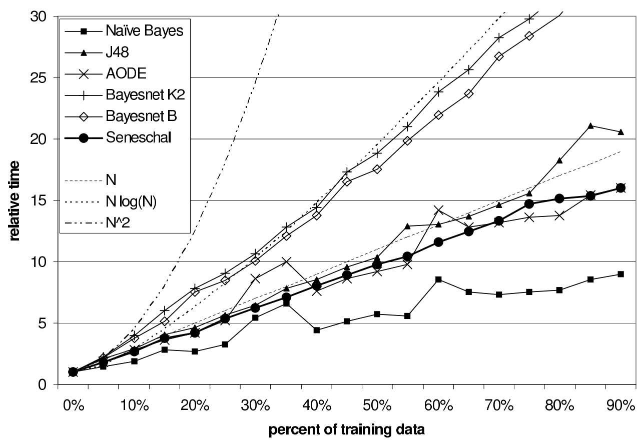

Figure 3. Relative training times on COR4

and *Bayesnet K2* is the Bayesian network described in [6].

Initial experiments with LBR were abandoned when it was found to perform too slowly (its accuracy was almost identical to that of AODE's). Experiments with supervised versions of the COBWEB, EM and K-Nearest-Neighbour (KNN) algorithms showed that they were unable to learn the concept better than Naive Bayes, with KNN performing with the greatest error and least efficiency, so these are also omitted from the reported results.

The additional algorithm tested is J48, an implementation of the C4.5 algorithm [19], as decision trees are currently the most commonly used learner in data stream mining. While J48 is not a streaming algorithm, it is a good indication of the ability of decision trees to learn highly correlated data.

As Figure 1 shows, all the algorithms other than *Seneschal* failed to produce a good model of the data. Interestingly, the generalization of the Naive Bayes learner proved one of the most robust, in that it failed to learn the concept well, but was the only learner that didn't overfit the data. Both Bayesnets overfit the data, classifying below the baseline, as did *Seneschal* initially.

The classic failure of learners obeying attribute independence assumption is the XOR problem, which is the simplest form of a correlation between two attributes in supervised learning. A synthetic set was created that extended the XOR concept to correlations between four attributes, called COR4.

16 binary attributes were created in total, with 4 classes, A,B,C & D. Class A was represented by attributes 1,2,3 & 4 having the value 1 *or* attributes 5,6,7 & 8 having the value 1 *or* attributes 9,10,11 & 12 having the value 1 *or* attributes 13,14,15 & 16 having the value 1. Class B was represented by attributes 16,1,2 & 3 having the value 1 *or* attributes 4,5,6 & 7 having the value 1 etc.

For each class and correlation, the instance was repeated with the 12 non-correlated attributes forming every possible combination of 1's and 0's that did not result in any additional series of four 1's. This resulted in 12,368 instances.

As with the tic-tac-toe data, the non-correlated attribute values were added systemically (counting up in binary numbers), so that at any single point, a split of the data into training and test sets would not result in identically distributed data sets with respect to the correlations in the data describing the class distributions.

As Figure 2 shows, this concept was able to be learned by all but the Naive Bayes classifier, but AODE, J48 and to a lesser extent Bayesnet B overfit the data when between 25% and 50% of the data was seen. This was as due to their over-fitting the 'correlations' they discovered in the noise.

Brief experimentation with the cache size threshold showed that the results were fairly robust. The tic-tac-toe data set tended to 100% accuracy with a threshold as low as 4, and the COR4 set with a threshold as low as 20. Predictably, this led to faster training times (with the same scalability), but slower convergence. Interestingly, at some very high thresholds, the accuracy was *reduced* for the tic-tac-toe set, indicating that a either a larger threshold was too slow to capture some concepts or the method for forming clusters within the outlier cache was inappropriate. It would be interesting to see if the cache size threshold could itself be made a dynamic measure, but this is left as future work.

5.2 Training Time

Figure 3 gives the relative learning times of the algorithms. These times were divided by the time taken at 5% of the data to normalize differences in implementation platforms etc. The decision tree *was* able to learn this simpler concept, although the tree was still quite large at 100% accuracy (about 800 nodes) indicated by the sharp jump in time when 80% of the data was seen. It is assumed that Naive Bayes's reported *sub*-linear performance is simple a product of inaccurate measures of time for the smaller sets. As expected, changes in the learning of Naive Bayes where echoed in AODE (hidden in this data is AODE's quadratic cost with respect to the number of attributes). Both AODE and Seneschal performed with linear cost.

Figure 3 also shows the algorithm described here to be one of the most stable with respect to training set sizes.

5.3 Adaptation

Two tests for adaptation were undertaken. One where the concept shifts were systematic and one where the concept shifts were random. For both, the relative frequency of all raw attribute values remained constant.

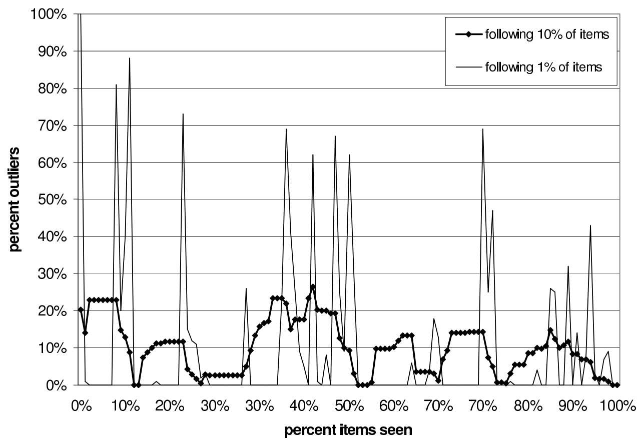

Figure 4. Percent of outliers at 1% and 10% intervals for training on COR4, ordered for systematic concept drift

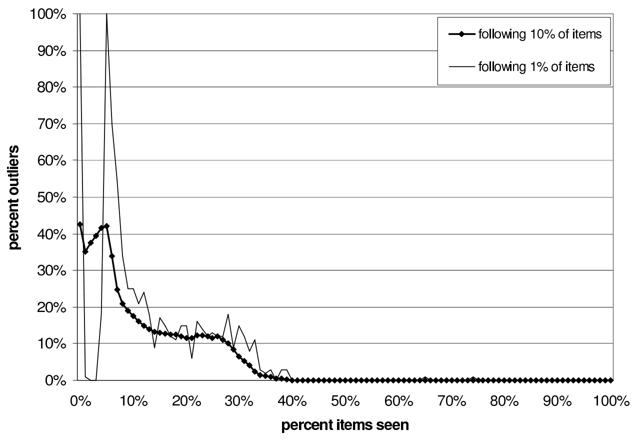

Figure 5. Percent of outliers at 1% and 10% intervals for training on COR4, with random ordering

Figures 4 shows the adaptive behaviour of the algorithm on a modified version of COR4. The set was re-ordered so that all A's and B's arrived before the C's and D's, with the classes C and D then re-labeled as A and B respectively. This allows us to analyse the behaviour of the learner in learning both concept drift, and abrupt concept shift. The Figure shows the percentage of outliers over subsets of 1% and 10% of the data. The sharp peaks of the 1% trend show points at which the cache was successfully flushed to create a new cluster. The large number of outliers at around 50% of the data show the concept shift to the former C's and D's.

Figure 5 shows the adaptive behaviour of the same set as in Figure 4, but with the order of training instances randomised. The learner converged much quicker and there were fewer outliers in total.

6 Discussion

Historically, decision trees have typically performed much better on highly correlated data then Bayesian learners due to the attribute independence assumption, and so it might be surprising to some that a Bayesian learner with this assumption relaxed can significantly outperform a decision tree in the mining of correlations, especially with only one pass of the data.

The reason for this is that a decision tree doesn't generalise its learning of correlations. Like a Bayesian learner or clustering algorithm, a decision tree learns *distributions* by inference, but it learns *correlations* only as they are expressed as the rules of divisions. As such, a decision tree learns correlations as interpolations, not inferences and is not able to generalise its knowledge of correlations. To learn the tic-tac-toe set, for example, it would need to create more nodes than there are training set instances and so the learning of this and similar highly correlated sets is impossible (while decision graphs do address this through their ability to learn disjunctive concepts [12], they are typically inefficient and are currently untested in streaming environments).

To increase the number of correlations that it captures, a decision tree must split on as many attributes as possible, therefore reducing its ability to generalise the distributions. It is this trade-off that caused the learning of J48 to be so erratic in the testing here, and is one of the reasons it is known as an unstable learner.

7 Conclusions

This paper has described the learning of attribute correlations in data streams, and described the first data stream mining algorithm able to learn highly correlated data.

We have proved that learning correlations between a potentially unbounded number of attributes is possible such that it guarantees linear scalability with respect to the number of instances, attributes and classes, which is a significant advance in the scalable learning of complicated concepts.

The success is derived from the architecture and processing strategy, which is an integration of techniques from outlier detection, clustering and supervised learning: outlier detection is used to bound the correlation strength, detect new concepts and create clusters; clusters represent correlations in a supervised learning task; and the supervised incremental error is used to dynamically adjust the outlier threshold.

7.1 Future Work

Outside of machine learning algorithms, there have been several recent papers proposing solutions for computing correlated aggregate counts suitable for estimating query costs in large databases [9, 2], and this may become a more important area for the mining of highly correlated streams. If the outlier threshold was made static, the current algorithm could be adapted to unsupervised learning for this purpose.

A simple probability measure was the core metric used here. Variations of this, and the inclusion of numeric attributes, would be appropriate in different environments.

This approach is ideally suited for massive parallelism and it would be interesting to see how much more efficient this is and how learning from multiple instances simultaneously may affect the development and accuracy of the learner.

References

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *The International Conference on Very Large Data Bases (VLDB)*, 2003.

[2] R. Ananthakrishna, A. Das, J. E. Gehrke, F. Korn, S. Muthukrishnan, and D. Srivastava. Efficient approximation of correlated sums on data streams. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):569–572, 2003.

[3] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16. ACM Press, 2002.

[4] W. L. Buntine. Theory refinement of Bayesian networks. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 52–60, 1991.

[5] R. Chen, K. Sivakumar, and H. Kargupta. An approach to online bayesian learning from multiple data streams. In *The Proceedings of Workshop on Mobile and Distributed Data Mining (PKDD)*, 2001.

[6] G. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[7] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 71–80, 2000.

[8] J. Gama, R. Rocha, and P. Medas. Accurate decision trees for mining high-speed data streams. In *The 9th International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2003.

[9] J. E. Gehrke, F. Korn, and D. Srivastava. On computing correlated aggregates over continual data streams. In *Proceedings of the 2001 ACM Sigmod International Conference on Management of Data*, 2001.

[10] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *IEEE Symposium on Foundations of Computer Science*, pages 359–366, 2000.

[11] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 97–106, San Francisco, CA, 2001. ACM Press.

[12] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag New York, Inc., 2001.

[13] R. Jin and G. Agrawal. Efficient decision tree construction on streaming data. In *The 9th International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2003.

[14] H. Kargupta and B.-H. Park. Mining decision trees from data streams in a mobile environment. In *The First IEEE International Conference on Data Mining (ICDM2001)*, pages 281–288, 2001.

[15] J. Kolter and M. Maloof. Dynamic weighted majority: A new ensemble method for tracking concept drift. In *Proceedings of the Third IEEE International Conference on Data Mining*, pages 123–130. IEEE Press, 2003.

[16] F. Korn, S. Muthukrishnan, and D. Srivastava. Reverse nearest neighbor aggregates over data streams. In *The International Conference on Very Large Data Bases (VLDB)*, 2002.

[17] M. A. Merzbacher and W. W. Chu. Pattern-based clustering for database attribute values. In *Proceedings AAAI Workshop on Knowledge Discovery in Databases*, 1993.

[18] R. Munro. Seneschal: classification and analysis in supervised mixture-modelling. In *Proceedings of the Third International Conference on Hybrid Intelligent Systems (HIS'03)*. IOS Press, 2003.

[19] J. Quinlan. C4.5 – programs for machine learning. *The Morgan Kaufmann series in machine learning*, 1993.

[20] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.

[21] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235. ACM Press, 2003.

[22] G. Webb and Z. W. J. Boughton. Averaged One-Dependence Estimators: Preliminary results. In *Proceedings of the Australasian Data Mining Workshop*, 2002.

[23] G. Webb and Z. W. J. Boughton. Not so naive bayes: Averaged one-dependence estimators. *Machine Learning*, 2004 (in press).

[24] K. Yamanishi, J.-I. Takeuchi, G. Williams, and P. Milne. Online unsupervised outlier detection using finite mixtures with discounting learning algorithms. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 320–324. ACM Press, 2000.

[25] Z. Zheng and G. I. Webb. Lazy learning of bayesian rules. *Machine Learning*, 41(1):53–84, 2000.