Real-time Learning when Concepts Shift

Jeffrey Coble **Diane J. Cook**

The University of Texas at Arlington
Department of Computer Science and Engineering
Box #19015, Arlington TX, 76019
{coble,cook}@cse.uta.edu

Abstract

We are interested in real-time learning problems where the underlying stochastic process, which generates the target concept, changes over time. We want our learner to detect when a change has occurred, thus realizing that the learned concept no longer fits the observed data. Our initial approach to this problem has been to analyze offline methods for addressing concept shifts and to apply them to real-time problems. This work involves the application of the Minimum Description Length principle to detecting real-time concept shifts.

Introduction

If enough consistent data can be obtained, standard machine learning algorithms can be applied to most learning problems in an offline, batch learning approach [5, 6, 7]. Our interest is in an online, sequential learning process, where the probabilistic functions that are generating the attributes and classifications of the world, change over time.

In the classic supervised learning problem [5, 9], it is generally stated that some stochastic function F(x) is generating an attribute vector x, based on a fixed probability distribution. The attribute vector, x, represents the salient features of the world. In a batch learning mode, the learner is given pre-classified training examples based on a conditional distribution function F(y|x), where the values that y can assume are the classification values that would be associated with an instantiation of the attribute vector x. This function is unknown to the learner. Consequently, it is the job of the learner to attempt to approximate this function, F(y|x), as best it can by observing the training values and applying a learning algorithm.

The online, sequential learning process presents several added difficulties. For example, since the learner is receiving it's training data sequentially; it will need to repeatedly apply the learning algorithm until it is satisfied that it has converged to a good model of the world. This means that it has to store all of the previously encountered training vectors in some usable form. Due to the enormity of the datasets, some summarization technique must be used that does not sacrifice valuable information [3]. This summarization technique must also be chosen so that it does not inappropriately bias the next learning iteration toward a previously learned model.

Our ultimate goal is to address uncertainty in real-time distributed computing problems with agent-based solutions. At the core of these agent-based solutions would be a real-time learning component that can detect and address shifts in the underlying target concept.

Our initial approach to this problem has been to analyze offline approaches that address concept shifts and to apply them to real-time problems. Our first effort involves the application of the Minimum Description Length principle to detecting concept shifts.

Sliding Window

One method used to address concept drift is the sliding window [4, 10] approach (figure 1). With this approach, the learner only considers the most recent n observed examples when learning the concept. There are a number of parameters the learner must consider, such as the size of the window and whether the window size is adaptive or stable. The size of the window is of importance in that a small window would compromise the confidence in the learned concept. Learning a concept on a small data sample can easily lead to overfitting the learned concept to small aberrations in the data or simply overlooking entire elements of the target concept. Conversely, large window sizes lead to resource issues in real time systems and may also leave the learned concept overly vulnerable to drift in the target concept. If the window size is large enough so that the range of enclosed attribute vectors drifts significantly, then the learned concept will perform poorly in a predictive capacity.

Lastly, it seems that the sliding window approach is much better suited to concept drift, rather than concept shift. Concept drift occurs when the target concept changes gradually, whereas concept shift occurs when the target concept changes instantly. Instantaneous change in the target concept would leave the learned concept open to error until the entire window was drawn from the new target concept.

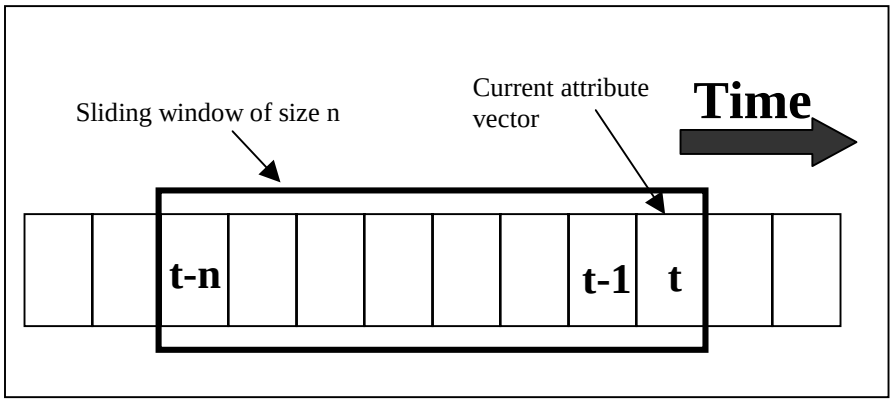

Figure 1. Addressing real-time concept drift/shift with a sliding window approach. The learner only considers the most recent n attribute vectors.

Problem Domain

The goal of our research is to create highly adaptable, real-time distributed computer systems. The benefits of this work could be realized through improved network data transmission speeds, greater autonomous control and coordination in satellite constellations and robot applications, and greater reliability in wireless communication, just to name a few.

Even under controlled conditions, distributed computing issues, such as maintaining global state and determining causality among a series of events, are fraught with difficulty. There are known algorithms for addressing these problems, however they are designed to work in environments where communication reliability is guaranteed and processors never fail. These algorithms quickly fail when a distributed computing application is introduced into a volatile environment, in which communication links may be unstable, the number of nodes may change, or adversaries may introduce errors into the communication channel.

Our research addresses the need for autonomous adaptability in real-time environments by introducing specialized intelligent agent technology into the distributed computing arena. Using this technology, we are providing a method for coping with uncertainty in order to address real-time adaptability issues.

One example of an application in this area is dynamic network routing. Real-time learning agents could be deployed at network routers so that they may learn to adapt routing patterns on-the-fly as traffic patterns change. For instance, traffic patterns may change with the time of day or due to an anomalous event, such as an outage in one part of the network. The growth of Internet use also causes traffic patterns to change in ways that have yet to be foreseen. Adaptive agents could reduce network down-time, provide greater throughput, and improved customer service.

Our initial approach has been to use a very popular problem, known as the 'the bandit problem' [1]. The bandit problem refers to the notion of a slot machine, which is a stochastic process with an unknown chance for payoff.

The bandit problem has many variations but for the purposes of this work, let it be defined as follows:

- Let there be two or more arms, where each arm is a stochastic process that can be in state 0 or 1.
- State 0 produces a corresponding reward of 0. State 1 produces a corresponding reward of 1.
- When an agent selects an arm, it receives the reward produced by the arm in its current state.
- Selecting an arm causes it to transition to a new state with some fixed probability. There is no correlation between the probability distribution governing each of the two arms
- The selection of one arm does not affect the state of the other arm.
- The agent may make N total selections, with the objective of maximizing its total reward. This is typically called an N-horizon problem.

- Each arm starts in a random state, selected according to its probability distribution.

This problem provides a simple testing ground for our machine learning research and is the subject of this initial work.

Bandit Simulator

For the purpose of experimentation, we have developed a k-armed bandit simulator. The simulator allows the user to add any number of arms where each arm can be configured with the following parameters:

Odds: The arm's actual odds of paying off. The arm is a Bernoulli Process, in that it generates independent results of the form "win or lose". The odds parameter is the true likelihood that the arm will produce a "win".

Payoff: This parameter is simply the value (number of wins)/(number of plays). In other words, it is the percentage of times this arm has produced a win. It should converge to the same value as the odds parameter over time.

Belief: This parameter is the agent's belief about the likelihood of the arm paying off. This value is calculated by the agent in response to spending a pull on the arm. The agent uses the observed result to update its belief about the arm's odds. If a single arm is played enough times, this value should converge to the value of the odds parameter for the arm. Currently, the agent computes a Beta density function for an arm after it is pulled. This function is used to formulate a belief as to the payoff odds for each arm. The agent assumes a uniform prior belief on the payoff odds of each arm. The agent always selects the arm for which it has the highest belief about payoff.

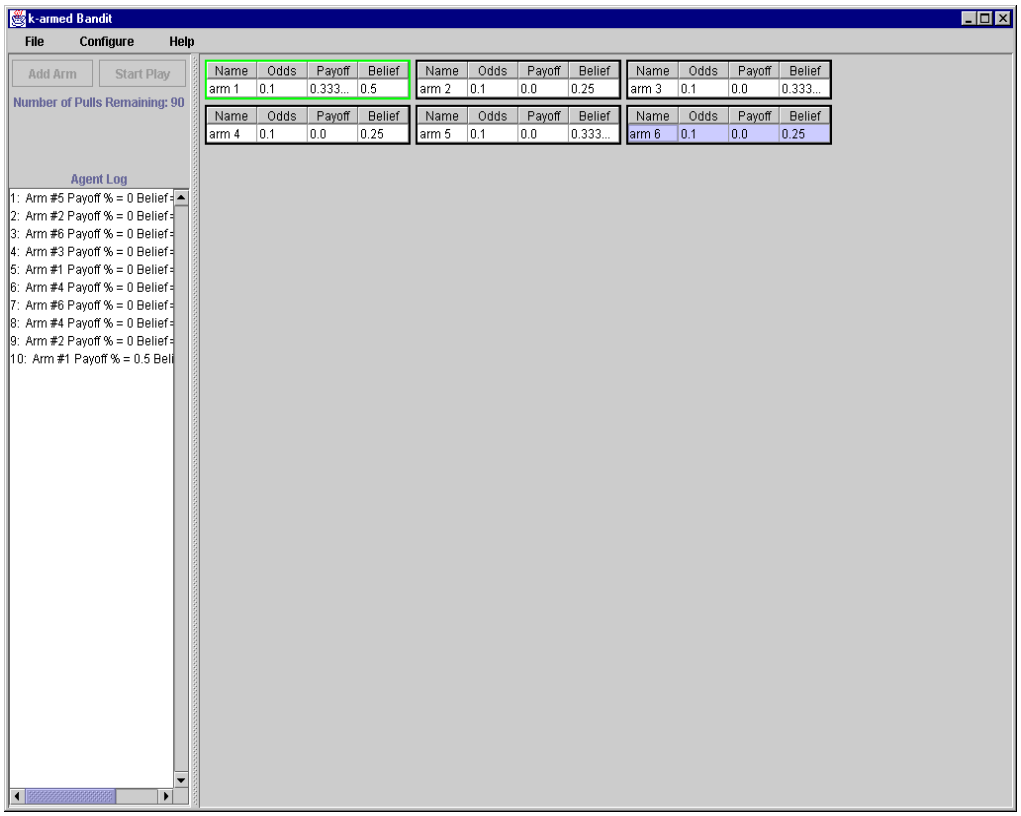

Figure 2. k-armed bandit simulator.

For the purposes of this work, we also want to generate a concept shift. This is realized by changing the "Odds" parameter to a new value during play. An arm can be configured with the following parameters, designed to facilitate concept shifts:

Initial Payoff: The starting "Odds" value for the arm.

Amount of change: By how much the "Odds" will change at each shift.

Direction of change: Values of "increment" or "decrement" can be selected. This value indicates whether to add or subtract the "Amount of change" to/from the "Initial Payoff" value.

# of examples per segment: The number of examples that will be included in each stable target concept. A segment refers to a section of stable target concept values, in this case simply a stable "Odds" value.
# of segments: The number of concept shifts.

MDL Approach

In [2], the authors present an approach for detecting the boundaries between segments in data mining problems. A segment represents data that was generated by a consistent process or system. For example, a coin with a 50-50 bias will roughly generate an equal number of <heads> and <tails> outcomes. When the bias changes, the data will obviously reflect this change and we consequently call this a segment shift. The author's original motivation for this algorithm was to detect surprising patterns in data mining problems.

The algorithm presented in [2] is based on the Minimum Description Length (MDL) principle [8]. In this section we describe the basic MDL algorithm for detecting concept shifts.

Each data item is represented as a tuple of boolean variable values. An example of a four variable item set might be <true, false, false, true>, where each boolean value represents the presence or absence of the particular variable. Each data tuple, containing k variables, can be thought of as being generated by a 2^k-sided coin or k two-sided coins. We will discuss the algorithm with respect to a tuple size of one variable. In other words, a single biased coin where the outcome is either <heads> or <tails>.

The underlying premise of the algorithm is that the segmentation and coin parameters that minimize C(M) + C($\vec{x}$ | M) can be computed in O(T^2), where T represents the number of tuples, $\vec{x}$ is the data set, and M is the model. C(M) represents the cost of encoding the model while C($\vec{x}$ |M) represents the cost of encoding the data, given the model. By finding the parameters that minimize the cost, the MDL algorithm finds the optimal segmentation. The costs are computed for each possible segmentation. We paraphrase the steps in the algorithm from [2] as follows:

1. Construct a directed acyclic graph (DAG) with T+1 nodes, and directed edges (i, j), for all $0 \le i < j \le T$. Each edge is weighted with the cost, c(i,j), representing the model and data encoding costs for the data items between between i (exclusive) and j (inclusive).
2. Find Model Parameters, $p_1(i, j)$ and $p_0(i,j)$, where $p_1(i, j)$ is the probability of <heads> between data items i and j and $p_0(i,j)$ is the probability of tails. These values are calculated from the data as follows: $p_1(i,j) = h_1(i, j)/t(i, j)$, where $h_1(i,j)$ is the number of <heads> between data items i and j and t(i,j) is the total number of data items between i and j. $p_0(i,j) = 1- p_1(i,j)$.
3. Find Data Encoding Cost for segment (i,j). Applying Shannon's theorem with the parameters above yields:
4. C($\vec{x}$ | M) = $-\sum p_x \log p_x$, where x = 0, 1.
5. Find Parameter Encoding Cost. One of the values $p_0(i,j)$ or $p_1(i,j)$ must be transmitted. Since the maximum likelihood estimate for $p_1(i,j)$ can only assume t(i,j)+1 values, the parameter encoding cost can be estimated as log(t(i,j)) bits.
6. Find Segmentation Costs. The boundaries of each segment must also be encoded. For k coins, this value is estimated as klogT.
7. Find Shortest Path. The shortest path through the graph is calculated in O(T^2) time, where each edge in the shortest path is a segment boundary.

This approach can be extended to tuples of size k by modifying the way in which the edge weights are calculated. The most substantial modification stems from the fact that with an increased number of variables, the number of possible models grows. The model encoding costs must be calculated for each model in order to choose the best. The authors present a simple heuristic for circumventing this problem. They simply reuse the best model from the itemset of k-1 and only consider generalizations of this model.

To evaluate this learning method in a real-time setting, we have reduced the bandit problem to one arm, in order to constrain the problem to that of simply learning one shifting target concept in real-time, rather than the arduous task of selecting the best of many learned concepts. Our experimentation has yielded some promising results but has also shown the need for substantial further work.

Results

As expected, the MDL solution degrades with respect to the size of the shift and the number of examples in each segment. As the shift in the target concept becomes less than 0.3 (30%), the segmentation boundaries become increasingly less accurate. Furthermore, as the number of examples in the new segment decrease below 50, the segmentation boundaries again become less accurate. With shifts larger than 0.3 and segment sizes approaching 100 examples, the MDL approach finds the segmentation boundaries with nearly perfect accuracy.

The primary issue with the MDL approach is its demand on computational resources. The search for the least costly segmentation is accomplished by constructing a graph with T nodes and O(T^2) edges, computing costs for each edge, and finding the shortest path through the graph with the weighted edges. Obviously the performance of this solution degrades rapidly as the dataset grows.

Conclusions and Future Work

We are encouraged by the accuracy of the MDL approach for detecting concept shifts. We believe that this solution has many areas in which optimization and summarization techniques can be applied to reduce the computational overhead involved, making this a workable solution to real-time learning problems.

One solution might be to apply summarization techniques to the graph so that the number of edges could be significantly reduced. Since the costs of many edges are simply composites of collections of edges, this seems like a very viable approach. Another alternative might be to apply the sliding window approach, used in concept drift, to this concept shift problem. If we limit our 'relearning' process to a window size that will yield acceptable accuracy while keeping computational costs low, the MDL approach may work well in real time.

Clearly the one-armed bandit test is very simple and our research must be expanded to substantially complex concepts. These issues will be addressed in our ongoing work in this area.

Reference

1. Berry, D. A., and Fristedt, B., *Bandit Problems: Sequential Allocation of Experiments*, Chapman and Hall, London, UK, 1985.
2. Chakrabarti, Soumen, Sarawagi, Sunita, Dom, Byron: *Mining Surprising Patterns Using Temporal Description Length*. Proceedings of 24th International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA. Morgan Kaufmann 1998: 606-617
3. Friedman, Nir and Goldszmidt, Moises, *Sequential Update of Bayesian Network Structure*, In the Proceedings of the Thirteenth Conf. on Uncertainty in Artificial Intelligence (UAI 97).
4. Klinkenberg, Ralf and Renz, Ingrid, *Adaptive Information Filtering: Learning Drifting Concepts.* Beiträge zum Treffen der GI-Fachgruppe 1.1.3 Maschinelles Lernen (FGML-98), Wysotzki, F. and Geibel, P. and Schädler, K. (ed.), Forschungsberichte des Fachbereichs Informatik, 98/11, Fachbereich Informatik, TU Berlin, 1998.
5. Mitchell, Tom. *Machine Learning*. McGraw Hill, 1997.
6. Quinlan, J. Ross, *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, USA, 1993.
7. Russell, Stuart and Norvig, Peter, *Artificial Intelligence: A Modern Approach*, Englewood Cliffs, NJ: Prentice Hall Inc, 1995.
8. Shannon, C. E. and Weaver, W., *The Mathematical Theory of Communication*. Urband: University of Illinois Press, 1949.
9. Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, NY, USA, 1995.
10. Widmer, G. and Kubat, M., *Learning in The Presence of Concepts Drift and Hidden Contexts,* Machine Learning, 323, 69-101 (1996).