Learning in Time Ordered Domains with Hidden Changes in Context

Michael Harries
Citrix Systems
91 PUBLICATIONS **1,130** CITATIONS
SEE PROFILE

Kim Horn
Infomedia
8 PUBLICATIONS **275** CITATIONS
SEE PROFILE

Claude Sammut
UNSW Sydney
284 PUBLICATIONS **6,355** CITATIONS
SEE PROFILE

Learning in Time Ordered Domains with Hidden Changes in Context

Michael Harries
Department of Artificial Intelligence,
University of NSW, Sydney 2052, Australia.
mbh@cse.unsw.edu.au

Kim Horn
Predictive Strategies Unit,
RMB Australia, Lvl 5, 37-49 Pitt St,
Sydney 2000, Australia
kim@rmb.com.au

Claude Sammut
Department of Artificial Intelligence,
University of NSW, Sydney 2052, Australia.
claude@cse.unsw.edu.au

Abstract

Concept drift due to hidden changes in context complicates learning in many real world domains including financial prediction, medical diagnosis, and communication network performance. Machine learning systems addressing this problem generally use an incremental learning, on-line paradigm.

An off-line, meta-learning approach to the identification of hidden context is presented. This approach uses an existing batch learner and the process of *contextual clustering* to identify stable hidden contexts, and the associated, context specific, locally stable concepts. The approach is broadly applicable to a range of domains and learning methods. We describe several evaluation domains and report current progress on these domains.

Introduction

Real world machine learning problems can be complicated by changes in important properties of the domain that are hidden from view. For example, in finance, a successful stock buying strategy can change dramatically in response to interest rate changes, world events, or with the season. As a result, concepts learnt at one time can subsequently become inaccurate, or worse, concepts learnt from a data-set combining several such changes can be quite inaccurate. *Hidden changes in context* cause problems for any machine learning approach that assumes concept stability.

In many domains, hidden contexts can be expected to recur. These domains include: financial prediction, dynamic control and other commercial data mining applications. Recurring context can be due to cyclic phenomena, such as seasons of the year or may be associated with irregular phenomena, such as inflation rates or market mood.

In this paper, we present recent work on SPLICE, an off-line meta-learning system for context-sensitive learning. SPLICE is designed to identify stable concepts during supervised learning in domains with hidden changes in context.

Motivation

This study had its genesis in an earlier data mining project with an Australian merchant bank, RMB Australia. The project goal was to find rules predictive of financial market movement, for subsequent use in an automated trading system.

Our early work on this domain (Harries & Horn 1995), took the approach of converting the domain from an explicit time series problem to a predicate learning problem. Each instance in the original data-set was represented by a target classification, and a series of attributes deemed significant by domain experts. We applied C4.5 (Quinlan 1993) to this data.

The learning task was complicated by changes in the target concepts over time, known as concept drift, caused by changes in the financial market. In common with many other researchers (such as (Widmer & Kubat 1996)) we dealt with concept drift by re-training on a window of recent instances for prediction in the near future. We also incorporated a mechanism for detecting when the current concept was outdated by associating acceptable attribute ranges with each leaf of the decision tree.

Concept drift is often associated with changes in a context hidden from the machine learning system. In finance, the hidden context might be the current interest rate or even the market mood. For a given value of the hidden context a particular concept might be

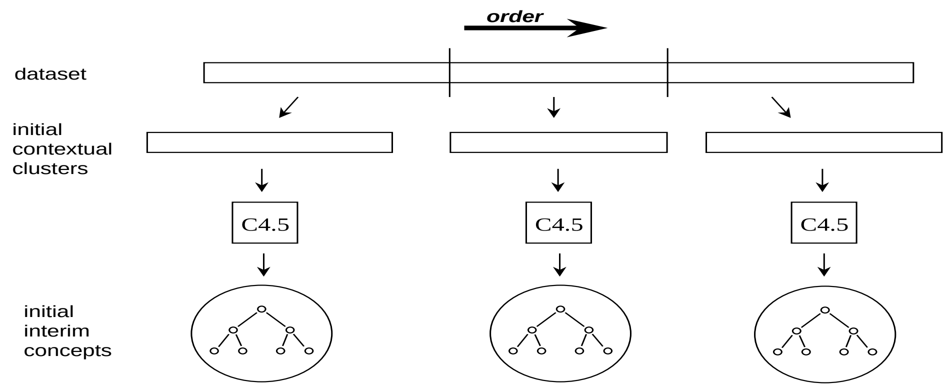

Figure 1: SPLICE-2: stage 1

expected to be predictive. For instance, some technical trading rules (trading methods based only on the pattern made by past prices) have been observed to be effective only with particular market moods, such as bullish, bearish, or jittery. Hidden changes of context can also effect many other real world domains, including the data mining of commercial data-bases, medical diagnosis, and robotics.

We chose to investigate the detection and extraction of concepts that are stable over time (i.e. between drifts), with the aim of re-using the stable concepts when similar contexts recur. The approach was partially inspired by Widmer and Kubat (1993; 1996) who augmented an on-line learning system with a mechanism to record and re-use apparently stable concepts. In domains where a large amount of historical data exists, batch learning can be used in favor of on-line learning.

SPLICE is a batch meta-learning system designed to extract recurring hidden context in classification domains. Past work on the Splice method up to and including SPLICE-2 has been published in (Harries & Horn 1998; Harries, Sammut, & Horn in press).

This paper presents an overview of the current state of the Splice project. We present the SPLICE-2 algorithm, and an improvement on the algorithm, SPLICE-2.1. We also review progress on a number of evaluation domains.

SPLICE-2

SPLICE's input is a sequence of training examples, each consisting of a feature vector and a known classification. The data are ordered over time and may contain hidden changes of context. From these data, SPLICE attempts to learn a set of stable concepts, each associated with a different hidden context. Since contexts can recur, disjoint intervals of the data-set can be associated with the same concept.

SPLICE uses a process called *contextual clustering* to group instances that appear to share a similarity of context. This similarity of context is measured by the degree to which intervals of the data-set are well classified by the same concept.

SPLICE is implemented as a meta-learning algorithm, hence, concepts are not induced directly, but by application of an existing batch learner. In this study, we use Quinlan's C4.5 (Quinlan 1993), but the SPLICE methodology could be implemented using other machine learning systems. C4.5 is used without modification. Furthermore, since noise is dealt with by C4.5, Splice contains no explicit noise handling mechanism. Unusual levels of noise are dealt with by altering the C4.5 pruning parameters. The SPLICE-1 algorithm is detailed in (Harries & Horn 1998). Here, we focus upon SPLICE-2, and a modification, SPLICE-2.1.

The SPLICE-2 algorithm has two stages, detailed briefly below. For a more complete description of SPLICE-2, see (Harries, Sammut, & Horn in press).

Stage 1: Partition data-set SPLICE-2 begins by guessing an initial partitioning of the data and subsequent stages refine the initial guess.

Several methods may be used for the initial guess:

- *Random partitioning.* Randomly divide the data-set into a fixed number of partitions.
- *Prior Domain Knowledge.* In some domains, prior knowledge is available about likely stable concepts.

Once the data-set has been partitioned, each interval of the data-set is stored as an initial *contextual cluster*. C4.5 is applied to each of these clusters to produce a decision tree known as an *interim concept*. Figure 1 shows a schematic of this stage.

Stage 2: Contextual clustering Stage 2 iteratively refines the contextual clusters generated in stage 1. With each iteration, a new set of contextual clusters is created in an attempt to better identify stable concepts in the data-set. Table 1 provides an overview of this stage.

This stage proceeds by testing each interim concept for classification accuracy against all items in the original data-set. A score is then computed for each pair of interim concept and item number. This score is based upon the number of correct classifications achieved in a

Table 1: SPLICE-2 overview.

```
Stage 1:
     create the initial contextual clusters and interim concepts
Stage 2:
     repeat
     -   Assess interim concepts on the original data-set
     -   Discard contextual clusters
     -   Create new contextual clusters - based upon similarity to interim concepts
     -   Discard interim concepts
     -   Create new interim concepts from the new contextual clusters
     until - fixed number of iterations completed, or clusters stop changing.
```

window surrounding the item. The window is designed to capture the notion that a context is likely to be stable for some period of time.

On-line learning systems apply this notion implicitly by using a window of recent instances. Many of these systems use a fixed window size, generally chosen to be the size of the minimum context duration expected. The window in SPLICE has the same function as in an on-line method, namely, to identify a single context.

The context of an item can be represented by a concept that most correctly classifies all items within the window surrounding that item. We define W_{ij} to be the score for concept j when applied to a window centered on example i.

$$W_{ij} = \sum_{m=i-w/2}^{i+w/2} Correct_{jm} \qquad (1)$$

where:

$$Correct_{jm} = \begin{cases} 1 & \text{if interim concept } j \text{ correctly} \\ & \text{classifies example } m \\ 0 & \text{if interim concept } j \\ & \text{misclassifies example } m \end{cases}$$

$$w = \text{the window size}$$

The current contextual clusters, from stage one or a previous iteration of stage two, are discarded. New contextual clusters are then created for each interim concept j. Once all scores are computed, each item, i, is allocated to a contextual cluster associated with the interim concept, j, that maximizes W_{ij} over all interim concepts. These interim concepts are then discarded. C4.5 is applied to each new contextual cluster to learn a new set of interim concepts.

The contextual clustering process iterates until either a fixed number of repetitions is completed or until the interim concepts do not change from one iteration to the next. The last iteration of this stage provides a set of stable concepts. The final contextual clusters give the intervals of the data-set for which different contexts are active.

SPLICE-2 Limitations

SPLICE-2 performs well on a number of domains but in domains with high levels of noise it often converges on contextual clusters that are very similar to the initial clusters. The effect is apparently due to over-fitting and can be reduced by increasing C4.5 pruning levels. The use of C4.5 pruning in this way is not considered to be a long term solution as SPLICE is designed to be independent of the underlying machine learning system used.

SPLICE-2.1

SPLICE-2.1 was designed to overcome over-fitting in SPLICE-2. It contains two improvements, both aimed at reducing the likelihood of poor convergence due to over-fitting. The first is an additional method for initial partitioning, in which a number of equally sized sets can be drawn randomly from the initial data-set. This avoids bias in the initial contextual clusters.

The second SPLICE-2.1 improvement is designed to reduce over-fitting in the clustering stage. When testing an interim concept, IC_i, SPLICE-2 does not distinguish between items that were in the prior contextual cluster, CC_i, used for training the interim concept IC_i, and items that were not. This means that IC_i is likely to get the maximum accuracy possible on most items in CC_i. As a result, there is a tendency for SPLICE-2 concepts to become fixed which limits the contextual cluster refinement that can occur. SPLICE-2.1 improves the process of testing each interim concept for classification accuracy against items in the associated contextual cluster.

SPLICE-2.1 divides each contextual cluster into ten equally sized hold-out sets. In turn, each is held out of the training set for a new interim concept, that interim concept is then assessed for classification accuracy on items in the hold-out set. The new classification accuracy figures are then used for calculating the W_{ij} scores as per SPLICE-2.

Application Areas

Here we introduce some evaluation domains. The first two have been tackled with SPLICE-2, and the third with both SPLICE-2 and SPLICE-2.1.

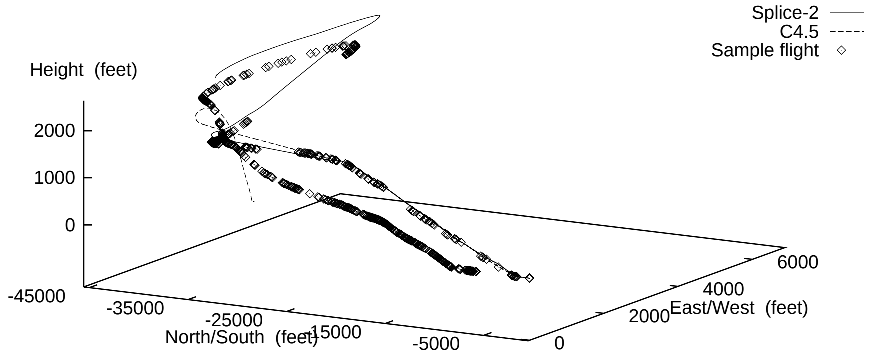

Figure 2: Flight comparison

Stagger

The Stagger domain is an artificial test domain designed to test on-line methods for concept drift (Schlimmer & Granger 1986). Splice has been applied and extensively tested on this domain (Harries & Horn 1998; Harries, Sammut, & Horn in press). For on-line prediction, Splice is competitive with on-line methods once given historical data containing the Stagger concepts. Splice is also very effective under a broad range of noise and other conditions at recognizing the Stagger concepts.

Learning to Fly

The "Learning to Fly" domain is a behavioral cloning task, in which the goal is to use a machine learning system to clone the control actions of a pilot flying a flight simulator through a fixed, seven stage, flight plan. This domain was initially investigated by Sammut *et al.* (1992). Previous work on this domain found it necessary to explicitly divide the domain into a series of individual learning tasks or stages. SPLICE-2 was able to induce an effective pilot for a substantial proportion of the original flight plan with no explicitly provided stages (Harries, Sammut, & Horn in press). In the following sections we briefly describe the problem domain and the application of SPLICE-2.

The "Learning to Fly" experiments (Sammut *et al.* 1992) were intended to demonstrate that it is possible to build controllers for complex dynamic systems by recording the actions of a skilled operator in response to the current state of the system. A flight simulator was chosen as the dynamic system because it was a complex system requiring a high degree of skill to operate successfully and yet is well understood.

We had 30 flight logs from a single pilot. The logs contained a record for every control action made by the pilot. Each log contained upward of 500 items with a total of 18781 training examples over all. Each example had 15 attributes showing position and motion, and 4 control attributes. The values for each of the control attributes are used as target classes for the induction of separate decision trees for each control attribute. These decision trees are then tested by compiling the trees into the autopilot code of the simulator and then "flying" the simulator.

Splice was used to learn stable concepts from the combined data logs. These were then combined into a control procedure by associating each example with a context (detected by Splice), a single decision tree was then induced to predict the current context (and thus the correct stable concept to apply).

This is a very complex task. At best, C4.5 alone was able to complete only two stages of the flight plan. Previous research on this domain has been successful only after providing a manual partitioning of the task. Even then, they still had to put in a great deal of work to make the simulator fly.

We were able to use SPLICE-2 to successfully fly four of the seven stages of the flight without dividing the task explicitly into stages. Figure 2 shows three flights: the successful SPLICE-2 flight on stages 1 to 4; the best C4.5 flight; and a sample complete flight.

Although we have not yet been able to fly all seven stages of the flight plan, successfully flying four stages is much more than had previously been achieved on this domain without manually partitioning the task.

The use of SPLICE-2 in synthesizing controllers for stages 1 - 4 is the first time that any automated procedure has been successful for identifying contexts in this very complex domain.

Calendar Scheduling

Another machine learning domain with hidden context is the calendar scheduling problem, described by Mitchell *et al.* (1994). In this domain, personal calendar software is augmented with a machine learning system designed to intelligently assist users by making scheduling predictions (time, location, day of week, and

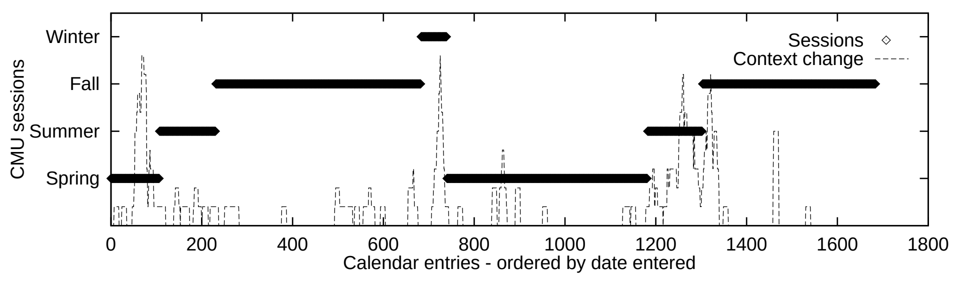

Figure 3: CMU Calendar.

duration) about new meetings. Each prediction is made on the basis of past scheduling decisions and some information (such as attendance) about the new meeting.

We evaluate both SPLICE-2 and SPLICE-2.1 contextual clustering on an aspect of the calendar scheduling problem, the prediction of meeting duration, for a single user, Tom Mitchell. The data-set[1] runs from 3 March 1992 to 16 December 1993 and has 1685 entries, each entry has 14 attributes and a classification. These data were previously used by (Mitchell *et al.* 1994; Widmer 1997).

This preliminary result shows the context boundaries found by SPLICE-2.1 on this data-set. SPLICE-2.1 was run ten times with four randomly drawn initial clusters, a window size of 100 items, and for 20 iterations. The window size was chosen to make the recognition of large scale contexts, such as semesters, more likely.

Repeated runs of SPLICE-2 on this domain tend not to agree on context boundaries. In fact, the boundaries detected are often very similar to those of the original partitions, suggesting that SPLICE-2 is unable to adapt contextual contexts in this domain. We have not included a figure to illustrate this.

Figure 3 shows CMU semesters, based upon the dates in the 97/98 calendar. The dashed line in this graph shows the proportion of Splice runs finding a context boundary within five items of each calendar entry. There are four prominent spikes in this line at entries 69, 727, 1263, and 1324. The second and fourth of these correspond to the beginning of teaching semesters. This chart shows that SPLICE-2.1 finds consistent context boundaries. It also suggests that a strong hidden context affects this domain.

SPLICE-2.1 has been shown to identify consistent context changes in this domain. Further experimentation is required to confirm that the context changes identified are useful for prediction.

[1]These data have been made available by Tom Mitchell and colleagues at http://www.cs.cmu.edu/afs/cs/project/theo-5/www/cap-data.html. Thanks to Gerhard Widmer for his copy of this data in C4.5 format.

Conclusion

This article has presented a new off-line paradigm for recognizing and dealing with hidden changes in context. Hidden changes in context can occur in any domain where the prediction task is poorly understood or where context is difficult to isolate as an attribute.

References

Harries, M., and Horn, K. 1995. Detecting concept drift in financial time series prediction using symbolic machine learning. In Yao, X., ed., *Eighth Australian Joint Conference on Artificial Intelligence*, 91–98. Singapore: World Scientific Publishing.

Harries, M., and Horn, K. 1998. Learning stable concepts in a changing world. In *Learning and Reasoning with Complex Representations*, number 1359 in LNAI. Springer-Verlag.

Harries, M.; Sammut, C.; and Horn, K. in press. Extracting hidden context. *Machine Learning.*

Mitchell, T.; Caruana, R.; Freitag, D.; McDermott, J.; and Zabowski, D. 1994. Experiences with a learning personal assistant. *Communications of the ACM* 37(7):81–91.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning.* San Mateo, California: Morgan Kaufmann Publishers Inc.

Sammut, C.; Hurst, S.; Kedzier, D.; and Michie, D. 1992. Learning to fly. In *Machine Learning: Proceedings of the Ninth International Conference*, 385–393. San Mateo, California: Morgan Kaufmann Publishers.

Schlimmer, J., and Granger, Jr., R. 1986. Incremental learning from noisy data. *Machine Learning* 1(3):317–354.

Widmer, G., and Kubat, M. 1993. Effective learning in dynamic environments by explicit concept tracking. In Brazdil, P. B., ed., *European Conference on Machine Learning*, 227–243. Berlin: Springer-Verlag.

Widmer, G., and Kubat, M. 1996. Learning in the presence of concept drift and hidden contexts. *Machine Learning* 23:69–101.

Widmer, G. 1997. Tracking context changes through meta-learning. *Machine Learning* 27:259–286.