Learning stable concepts in a changing world.

Conference Paper · January 1996

Source: DBLP

CITATIONS
5

READS
36

2 authors:

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

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

Learning stable concepts in a changing world

Michael Harries
Department of Artificial Intelligence
School of Computer Science and Engineering
University of NSW, Australia
mbh@cse.unsw.edu.au

Kim Horn
Predictive Strategies Unit
Australian Gilt Securities Limited
Australia
kim@ags.com.au

Abstract

Concept drift due to hidden changes in context complicates learning in many domains including financial prediction, medical diagnosis, and network performance. Existing machine learning approaches to this problem use an incremental learning, on-line paradigm. Batch, off-line learners tend to be ineffective in domains with hidden changes in context as they assume that the training set is homogeneous.

We present an off-line method for identifying hidden context. This method uses an existing batch learner to identify likely context boundaries then performs a form of clustering called *contextual* clustering. The resulting data sets can then be used to produce context specific, locally stable concepts. The method is evaluated in a simple domain with hidden changes in context.

1 INTRODUCTION

Prediction in real world domains is complicated by potentially unstable underlying phenomena. In the financial domain, for example, market behaviour can change dramatically with changes in contract prices, interest rates, inflation rates, budget announcements, and political and world events. This *concept drift* is due to changes in context, and is often directly reflected by one or more attributes. When changes in context are not reflected by any attribute they can be said to be hidden. Hidden changes in context cause problems for any predictive approach that assumes concept stability.

The most common approach to learning in domains with hidden changes in context is to generalise from a window that moves over recently past instances and use the learnt concepts for prediction only in the immediate future. Such an approach was initially suggested in (Kubat, 1989). Examples of this approach include (Widmer and Kubat, 1996), (Kubat and Widmer, 1995), (Kilander and Jansson, 1993), and an adaptation in (Salganicoff, 1993). In domains of this nature, quick adaption to new contexts is the primary requirement for learning systems. The Flora family of algorithms (Widmer and Kubat, 1996) directly address fast adaptation to new, previously unseen, contexts by dynamically adjusting the window size in response to changes in accuracy and concept complexity.

There are a large number of domains in which context can be expected to repeat. In these domains, it would be valuable to store existing knowledge about past contexts for re-use. Flora 3 (Widmer and Kubat, 1993) addresses domains in which contexts recur, and does so as a secondary consideration to the adaptive learner. This is achieved by storing and retrieving concepts that appear stable as the learner traverses the series.

An alternative to the windowing approach is to examine the data *en masse* and attempt to directly identify concepts associated with stable, hidden contexts.

In this paper we present Splice, a meta-learning system that implements such an approach. The Splice algorithm has three stages:

1. Use an existing batch machine learner to identify possible changes in context as partitions on the time attribute.

2. Refine the identified changes in context by clustering the intervals according to apparent similarity

of context. We call this *contextual* clustering.

3. Use the clustered, context specific data sets to learn stable concepts. These concepts are referred to as *local* concepts and together form a description of the whole domain, known as a *global* concept.

Splice is evaluated on a simple domain drawn from the literature using C4.5 (Quinlan, 1993) as the underlying batch learner. We use this domain and learning system only to demonstrate that the method is viable. The approach is applicable to more complex domains and other learning paradigms such as ILP.

1.1 REPRESENTING TIME

In the domains we are examining in this paper time plays an important role. Time allows us to differentiate between different contexts when the changes in context are hidden. Windowing, and other existing approaches to these domains, do not explicitly represent time as an attribute. The reason for this exclusion is that using time as an attribute causes problems for prediction. For example, an attempt to build a system to predict political parties from past data, may induce the following decision tree.

```
Country = America
    Year < 1992:  Government = Republican
    Year >= 1992:
        Public Wants Change = yes: Government = Democrat
        Public Wants Change = no:  Government = Unknown
```

This decision tree is of limited use in predicting future governments as all future values of year must be greater than 1992. Hence, if such a tree were used directly for prediction, only the post 1992 fragment of the tree would be used.[1]

Such a decision tree is, however, useful for understanding the past and implies that time is important in delineating two contexts, pre 1992 and post 1992. The post 1992 section of the tree contains a local, or context specific, concept in which the importance of 'Public Wants Change' is evident.

[1]Such a tree could be used directly for prediction in an iterative sense. This approach has the advantage that no window size need be set or adjusted as the tree incorporates segmentation on time. It is also, unfortunately, susceptible to very small contexts being incorrectly identified at the most recent interval of time due to random serial correlation (unpublished work).

1.2 IDENTIFYING CONTEXT CHANGE

Despite the problems introduced by using time in predictive problems, the decision tree in the example above does contains useful information about changes in context. The ability of decision trees to capture context is associated with the fact that decision tree algorithms use a form of context-sensitive feature selection (CSFS) (Domingos, in press). A number of machine learning algorithms can be regarded as using CSFS including decision tree algorithms (Quinlan, 1993), rule induction algorithms (Clark and Niblett, 1989) and ILP systems (Quinlan, 1990). All of these systems produce concepts containing information about context.

The problem faced in domains with hidden changes in context is that the context is not explicitly available. Context is, however, often reflected as contiguous intervals of some attribute, in this case it is time. When time has this property, we call it the *environmental* attribute.[2]

Given an *environmental* attribute, we can utilise a CSFS machine learning algorithm to gain information on hidden changes in context.[3] In order to represent hidden changes in context, as partitions on the environmental attribute, we need to further restrict the machine learning algorithm to one that is able to provide splits on that attribute.

The likelihood of such a system correctly identifying partitions on the environmental attribute will be a function of, at least, context duration, the number of different contexts, the complexity of each local concept, and noise.

The partitions initially identified can be expected to contain errors of two types.

- Noise or serial correlation errors. These would take the form of random splits on time.
- Errors due to repetition of tests on time in different parts of the concept. These would take the form of a group of values clustered around the

[2]The *environmental* attribute could be any ordinal attribute over which instances of a hidden context are liable to be contiguous. There is also no restriction, in principle, to one dimension. Some alternatives to time as environmental attributes are dimensions of space, and space-time combinations.

[3]Windowing approaches also assume that hidden context is liable to be contiguous in time. This assumption is made implicitly and is necessary as a window is useful only if its instances are primarily from a single context.

actual point where the context changed.

The initial partitioning of time into stable intervals can be refined by contextual clustering. This process combines similar intervals of the dataset, where the similarity of two intervals is based upon the degree to which a partial model is accurate on both intervals.

1.3 INCREMENTAL 'VS' BATCH APPROACHES

Machine learning techniques can be broadly categorised as either batch or incremental. Batch systems learn by examining a large collection of instances en masse and forming a single concept. Incremental systems evolve and change a classification scheme as new observations are processed (Schlimmer and Granger, 1986). The window based systems, described above, could all be described as incremental in approach.

In many situations, there is no constraint to learn incrementally. For example, many organisations maintain large data bases of historical data that are prime targets for data mining. These data bases may hold instances that belong to a number of contexts but do not have this context explicitly recorded. Many of these data bases may incorporate time as an essential attribute, for example, financial records and stock market price data. Interest in mining datasets of this nature suggest the need for systems that can learn global concepts and are sensitive to a changing and hidden context.

The likely advantages of employing this method on domains with hidden changes in context are:

- A global concept can be learnt that partitions the data based on context rather than on an arbitrary window of most recent cases.
- After producing a set of local concepts and associated contexts, it should be possible to identify repeating contexts and to reason about expected context duration and sequence.
- Local concepts can be verified by experts.
- Some form of cross-validation over time can be applied for local concept verification.
- Local concepts can used for prediction.

2 SPLICE

Splice uses time as an environmental attribute to learn a set of local concepts. Local concepts identified by Splice can be used both for understanding the target domain and for prediction.

Splice is a meta-level algorithm that incorporates an existing batch learner. In this study we employed C4.5 (Quinlan, 1993), without any modifications, as the underlying learning system. The underlying learner for Splice could, in principle, be replaced by any other CSFS machine learner that can provide splits on time. As we expect the underlying learner to deal with noise, Splice does not have (or need) a mechanism to deal with noise directly.

The Splice algorithm is detailed in Figure 1. It consists of three stages:

1. Partition Dataset
2. Perform Contextual Clustering
3. Learn Local Concepts

We examine each of these in turn.

2.1 Partition Dataset

Splice first uses the underlying batch learner to build an initial concept from the whole data set. As Splice uses C4.5 the initial concept is a decision tree. By learning a concept description of the whole domain including time, we can expect to identify the significant tests on time. Each test on time provides a split, identifying a possible change in context. The splits on time are extracted from the initial concept and used to define both *intervals* of the dataset and *partial* concepts.

2.2 Perform Contextual Clustering

In this stage, we attempt to cluster the *intervals* identified above.

Splice determines the accuracy of each partial concept by classifying the individuals in each *interval*. This process is simplified by shifting each *interval* into the relevant time values for each partial concept. This allows us to re-use the initial concept. We may need, for instance, to classify an *interval* with time values from i to j using a candidate concept defined by a partition of time from k to l in the initial concept. This is achieved by replacing the time value for each instance of the *interval* with the time k.[4] The *interval* is then classified

[4]The value used in replacing time values in the *interval* could be any in the partition defining the partial concept, in this case it could be any value from k to l.

Input required:

- Data set with an environmental attribute
- Threshold accuracy parameter θ with a possible range of 0 to 100

Algorithm:

- Stage 1: Partition Dataset
  - Use batch learner to classify the initial data set and produce an initial concept.
  - Extract tests on the time identified in the initial concept.
  - Tests on time are used to partition the dataset into *intervals*.
  - Tests on time also used to partition the initial concept into *partial* concepts. Partial concepts are fragments of the the initial concept associated with a particular interval of time.
- Stage 2: Perform Contextual Clustering
  - Evaluate the accuracy of each *partial* concept on each *interval* of data.
  - Rate each *partial* concept by coverage of the data set. Coverage is the total number of examples in *intervals* classified by the *partial* concept at better than the threshold accuracy θ.
  - Create an ordered set X of partial concepts.
  - While X is not empty:
    - Select best partial concept from X.
    - Create a new cluster from covered *intervals*.
    - For all *intervals* used in the cluster, remove the associated *partial* concept from X.
- Stage 3: Learn Local Concepts
  - Apply the batch learner to each contextual cluster in order to learn a new local concept. Context is delineated in time by the boundaries of the cluster.

The Splice output consists of all local concepts produced.

Figure 1: The Splice Algorithm

using the initial concept. This substitution is repeated for each combination of *partial* concept and *interval*. The error rate is recorded in a Local Accuracy Matrix.

A *partial* concept is considered to cover an *interval* of the data set if the error rate (as a percentage) when classifying that *interval* is less than the accuracy threshold parameter θ. The default setting for θ is 10%. We expect that adjusting this parameter will be useful in different domains. Each *partial* concept is rated in terms of data set coverage. This is the number of instances in all the *intervals* of the data set that it covers. An ordered set X of *partial* concepts is created.

The actual clustering procedure proceeds as follows. The best *partial* concept is selected from the set X. All the *intervals* that it covers are used to form a new cluster. The *partial* concepts associated with the *intervals* used are removed from the set X. This step is then repeated with the next best candidate concept until set X is empty.

The contextual clustering step could conceivably be replaced by an algorithm that directly post-processed the initial C4.5 tree. Preliminary investigations into such an algorithm suggest that it could be viable in an environment with simple concepts and low noise, but is likely to fail with more complex and noisy environments. The Splice contextual clustering algorithm was designed to identify and combine similar intervals of the data set under a broad range of conditions including those of substantial noise and complex concepts.

2.3 Learn Local Concepts

The underlying learner, C4.5, is used to learn a *local* concept from each contextual cluster from the previous stage. It is important to note that at this stage the environmental attribute, time, is not included in the attribute set.

Splice is able to exploit recurring contexts by building larger combined data sets. Using larger combined data sets can improve the quality of the local concepts produced by C4.5.

3 EXPERIMENTAL RESULTS

The following experiments were primarily designed to demonstrate the viability of the Splice approach. The first experiment applies Splice to a single dataset with recurring contexts. It serves to both confirm that Splice will work on domains with recurrent con-

cepts and as an illustration of the execution of Splice. The second experiment compares the classification accuracy of Splice and C4.5 in a domain with hidden changes in context. The third experiment investigates the effect of noise and training duration on Splice's ability to correctly report target local concepts.

3.1 DATA SETS

The data sets used in the following experiments are based on those used in Stagger (Schlimmer and Granger, 1986) and subsequently used by (Widmer and Kubat, 1996). While our approach and underlying philosophy are substantially different, this allows some comparison of results.

The domain chosen is artificial and a program was used to generate the data. This program allows us to control recurrence of contexts and other factors such as noise[5] and duration. The domain has four attributes, time, size, colour and shape. Time is treated as a continuous attribute. Size has three possible values: small, medium and large. Colour has three possible values: red, green and blue. Shape also has three possible values: circular, triangular, and square.

The program randomly generates a series of examples from the above attribute space. Each example is given a unique time stamp and classified according to the currently active target concept. In this study, we employed three simple target concepts:

1. (size = small) $\wedge$ (colour = red)
2. (colour = green) $\vee$ (shape = circular)
3. (size = medium) $\vee$ (size = large)

Artificial contexts were defined by changing the currently active target concepts at predetermined intervals. In this study, the intervals were over a count of cases generated by the program.

3.2 EXPERIMENT 1 - Applying Splice to Temporal data

This experiment applies Splice to a dataset with recurring concepts. It serves both to confirm that Splice is able to extract target local concepts from a dataset with recurring concepts and as an illustration of the execution of Splice[6].

```
Size = small:
|   Time > 188 : no (22.0/1.3)
|   Time <= 188 :
|   |   Colour = blue: no (19.0/2.5)
|   |   Colour = red:
|   |   |   Time <= 52 : yes (9.0/1.3)
|   |   |   Time > 52 :
|   |   |   |   Time <= 126 : no (8.0/1.3)
|   |   |   |   Time > 126 : yes (4.0/2.2)
|   |   Colour = green:
|   |   |   Time > 159 : yes (6.0/1.2)
|   |   |   Time <= 159 :
|   |   |   |   Shape = square: no (3.0/1.1)
|   |   |   |   Shape in {circular,triangular}:
|   |   |   |   |   Time > 91 : no (7.0/1.3)
|   |   |   |   |   Time <= 91 :
|   |   |   |   |   |   Time <= 38 : no (5.0/1.2)
|   |   |   |   |   |   Time > 38 : yes (4.0/1.2)
Size in {medium,large}:
|   Time <= 39 : no (21.0/1.3)
|   Time > 39 :
|   |   Time > 202 : yes (20.0/1.3)
|   |   Time <= 202 :
|   |   |   Time <= 119 :
|   |   |   |   Colour = green: yes (26.0/1.3)
|   |   |   |   Colour in {red,blue}:
|   |   |   |   |   Time > 77 : yes (19.0/1.3)
|   |   |   |   |   Time <= 77 :
|   |   |   |   |   |   Shape in {square,triangular}: no (10.0/1.3)
|   |   |   |   |   |   Shape = circular: yes (3.0/1.1)
|   |   |   Time > 119 :
|   |   |   |   Time <= 158 : no (27.0/1.4)
|   |   |   |   Time > 158 :
|   |   |   |   |   Colour = green: yes (11.0/1.3)
|   |   |   |   |   Colour in {red,blue}:
|   |   |   |   |   |   Shape in {square,triangular}: no (13.0/1.3)
|   |   |   |   |   |   Shape = circular: yes (3.0/1.1)
```

Figure 2: C4.5 decision tree (using time)

The training set consists of concepts (1) for 40 instances, (2) for 40 instances, (3) for 40 instances, and repeating (1) for 40 instances, (2) for 40 instances, and (3) for 40 instances. No noise was applied to the data set.

3.2.1 Results

The sequence followed by Splice in learning local concepts from the data set described above is illustrated in Figure 2, Table 1 and Figure 3. Figure 2 contains the initial concept built by C4.5 over the entire data set. Table 1 shows the Local Accuracy Matrix built in stage 2 of the algorithm. Figure 3 shows the local concepts learnt from this particular data set.

The initial concept decision tree in Figure 2, as generated by C4.5, is not succinct nor is it easy to interpret. If applied directly to the prediction of future instances,

[5] In the following experiments, n% noise implies that the class was randomly selected with a probability of n%. This method for generating noise was chosen to be consistent with (Widmer and Kubat, 1996).

[6] In all experiments reported Splice was run with the threshold accuracy parameter θ set to a default of 10%. The underlying learner, C4.5, was run with default pruning parameters and with subsetting.

Table 1: Local Accuracy Matrix

Data set	Range	Accuracy achieved by each partial concept: % error											
0	0-38	0	13	44	65	88	75	21	0	31	47	52	75
1	39-39	0	0	0	100	100	100	100	0	0	0	100	100
2	40-52	70	54	8	8	39	54	70	70	24	8	24	54
3	53-77	68	64	12	0	24	28	56	68	16	12	4	28
4	78-91	86	79	58	43	0	8	72	86	65	72	50	8
5	92-119	83	90	43	33	8	0	72	83	36	43	25	0
6	120-126	0	15	58	58	100	86	0	0	43	58	43	86
7	127-158	0	13	41	47	85	72	7	0	29	41	35	72
8	159-159	0	0	0	0	0	0	0	0	0	0	0	0
9	160-188	59	45	11	11	38	52	59	59	25	4	25	52
10	189-202	58	58	29	0	36	36	29	58	29	29	0	36
11	203-239	71	84	55	38	14	0	55	71	41	57	25	0

```
Current best partial concept is 0 with 80 data items covered
        context 0    0-38
        context 1    39-39
        context 6    120-126
        context 7    127-158
        context 8    159-159
Local concept learnt:
        Colour in {green,blue}: no (52.0)
        Colour = red:
        |   Size = small: yes (11.0)
        |   Size in {medium,large}: no (17.0)
**** equals target concept 1: (size = small) & (colour = red)


Current best partial concept is 5 with 80 data items covered
        context 4    78-91
        context 5    92-119
        context 8    159-159
        context 11    203-239
Local concept learnt:
        Size = small: no (31.0/1.0)
        Size in {medium,large}: yes (49.0)
**** equals target concept 3: (size = medium) | (size = large)


Current best partial concept is 3 with 53 data items covered
        context 2    40-52
        context 3    53-77
        context 8    159-159
        context 10    189-202
Local concept learnt:
        Colour = green: yes (22.0)
        Colour in {red,blue}:
        |   Shape in {square,triangular}: no (24.0)
        |   Shape = circular:
        |   |   Size = small: no (3.0/1.0)
        |   |   Size in {medium,large}: yes (4.0)
**** close to target concept 2 but not correct


Current best partial concept is 9 with 44 data items covered
        context 1    39-39
        context 2    40-52
        context 8    159-159
        context 9    160-188
Local concept learnt:
        Colour = green: yes (20.0)
        Colour in {red,blue}:
        |   Shape in {square,triangular}: no (17.0/1.0)
        |   Shape = circular: yes (7.0/1.0)
**** equals target concept 2: (colour = green) | (shape = circular)
```

Figure 3: Annotated extract from splice log

only a small fragment of the tree would be used.

The decision tree contains a substantial amount of information on past contexts. Splits identified on time in the initial concept are used to define both partial concepts and *intervals* of the data set. Each partial concept is evaluated on all *intervals* of the data set. The results of this evaluation form the Local Accuracy Matrix (LAM), Table 1. It should be noted that some of the ranges shown in the LAM are due to spurious splits on time in the initial concept.

The LAM is used as input for the contextual clustering operation. As shown in Figure 3, the contextual clusters are then used to produce local concepts. For the first two local concepts generated, Splice was able to successfully identify and combine all non-contiguous intervals of the same context. The results for this run were that all target concepts were successfully identified and that one additional incorrect local concept was identified.

The twin tasks for Splice are to correctly identify target concepts and minimise the number of incorrectly identified concepts. In unpublished work on the same task, Splice correctly identified concept one 97 times out of 100 trials, concept two 96 times out of 100 trials, and concept three 100 times out of 100 trials. Splice also identified, on average, 0.8 incorrect or spurious concepts per trial. We anticipate that a substantial proportion of these spurious concepts could be identified by restricting contextual clustering to a minimum required coverage and a maximum permitted fragmentation. This has not yet been investigated. In the case presented, neither of these strategies would have been effective.

Although spurious local concepts are not desirable, their effect is minimal in prediction tasks. When local concepts are used for prediction, a secondary system must be used to select the currently active local concept for use in prediction of the next item[7]. The task of selecting the correct local concept from a set of local concepts valid in different contexts is not altered by that set containing a concept that is not valid in any context[8]. Hence, if the secondary system is competent in selecting the current concept with no spurious concepts, it should be able to avoid selecting spurious concepts in most cases.

This experiment demonstrates that Splice can be used to improve our understanding of the domain and to generate succinct and accurate local concepts.

3.3 EXPERIMENT 2 - Direct Comparison with C4.5

This experiment compares the accuracy of Splice to C4.5 when trained on a data set containing hidden context shifts. After training, the resulting concepts are fixed and used for prediction on a similar data set. C4.5 is employed to provide a baseline performance for this task and was trained without the attribute time. This comparison is not altogether fair on C4.5, as it was not designed for use in domains with hidden changes in context.

The training set consists of concepts (1) for 50 instances, (2) for 50 instances, and (3) for 50 instances. The test set consists of concepts (1) for 50 instances, (2) for 50 instances, (3) for 50 instances, and repeated (1) for 50 instances, (2) for 50 instances, and (3) for 50 instances.

To apply the local concepts identified by Splice for prediction purposes, it was necessary to devise a method for selecting relevant local concepts. This is not a trivial problem, hence, for the purposes of this experiment we chose a simple method. The classification accuracy of each local concept over the last five examples was recorded. The most accurate concept was used in predicting the class of the next example. Any ties in accuracy were solved by randomly selecting between local concepts. The first case was classified by randomly selecting a local concept. Future work will investigate alternate local concept selection methods.

The results, presented below, show the average classification success rates at several levels of noise for both Splice and C4.5 over 100 randomly generated training and test sets. For this experiment, noise was only generated in the training set.

[7] See experiment 2.

[8] With the exception of possible effects due to the number of concepts from which to select.

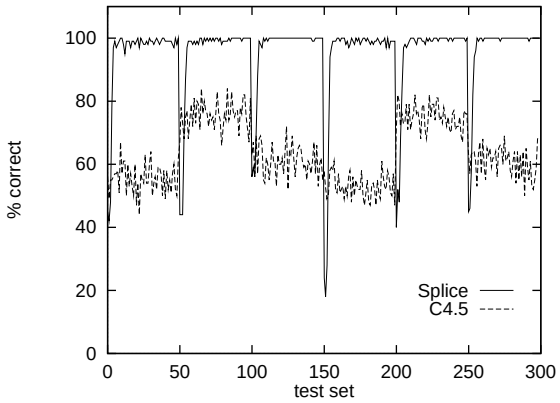

Figure 4: C4.5 comparison, trained with no noise

3.3.1 Results

Figure 4 shows that Splice successfully identified the local concepts from the training set and that the correct local concept can be successfully selected for prediction purposes in better than 95% of cases. The extreme dips in accuracy when contexts change are an effect of the local concept selection method. C4.5 seems to do relatively well on concept 2 with an accuracy of approximately 70%. C4.5 on concepts 1 and 3 correctly classifies between 50% and 60% of cases.

Figures 5,6 and 7 show a gradual decline in the classification accuracy achieved by Splice as noise increases. At 30% noise, the worst result achieved by Splice is an 85% classification accuracy on concept 2. C4.5 on the other hand is still classifying with approximately the same accuracy as it achieved in figure 4. C4.5 predictive stability over a range of noise of between 0 and 30% is testament to its stability in adverse situations.

3.4 EXPERIMENT 3 - The Effects of Noise and Duration

This experiment investigates the performance of Splice under varying levels of noise and duration. Splice was first trained on a randomly generated training set containing examples of each of the Stagger concepts. The set of local concepts learnt by Splice were then assessed for correctness against each target concept. The results show the proportion of correct local concept identifications achieved, and the average number of in-

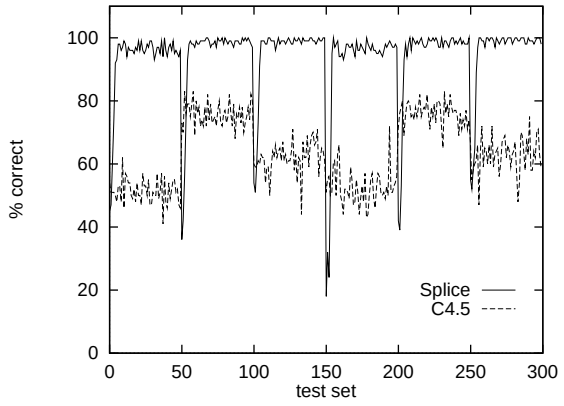

Figure 5: C4.5 comparison, trained with 10% noise

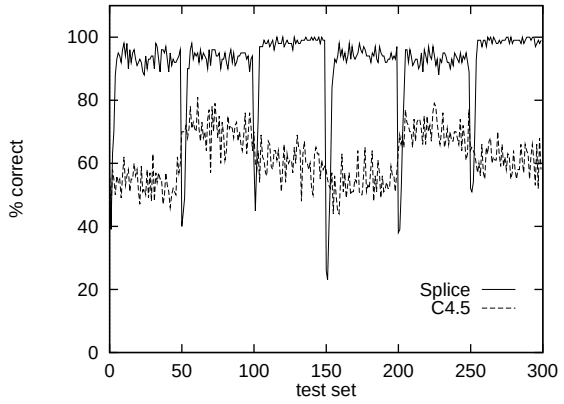

Figure 6: C4.5 comparison, trained with 20% noise

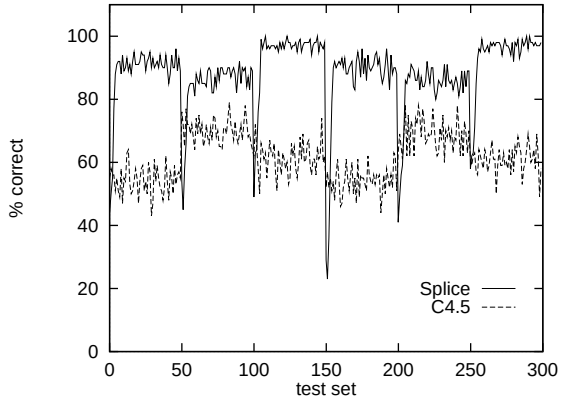

Figure 7: C4.5 comparison, trained with 30% noise

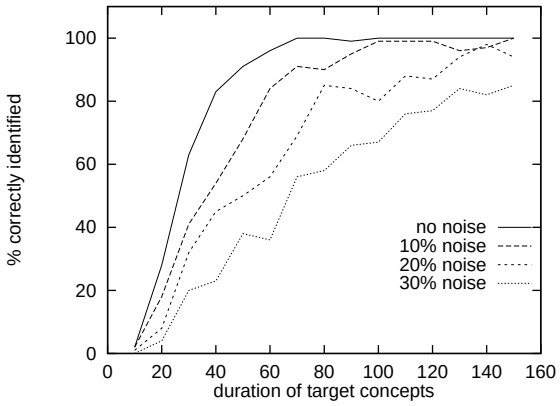

Figure 8: Identification of concept 1

correct local concepts identified.

Training sets were generated using a range of concept duration and noise. Concept duration corresponds to the number of instances for which a concept is active. Duration ranges from 10 instances to 150 instances. Noise ranges from 0% to 30%. Each training set consists of concept (1) for D instances, concept (2) for D instances, and concept (3) for D instances, for some duration D.

Each result reported is based on 100 randomly generated training sets.

3.4.1 Results

Figures 8, 9 and 10 show the success levels of Splice in correctly identifying each target concept under varying levels of both concept duration and noise. In this domain, Splice is well behaved, with a graceful degradation of performance as noise levels increase. Concept duration reduces the negative effect of noise. Figure 11 shows the number of incorrect concepts learnt by Splice for different levels of noise and training concept duration. In this too, Splice is well behaved, showing both graceful degradation of performance with increased noise, and well bounded numbers of incorrect concepts learnt.

4 DISCUSSION

Experiment 1 demonstrated that Splice can reconstruct the original concepts used in the generation of a dataset with recurring contexts.

In experiment 2, Splice was shown to perform well on a task similar to that approached by both Flora (Widmer and Kubat, 1996) and Stagger (Schlimmer

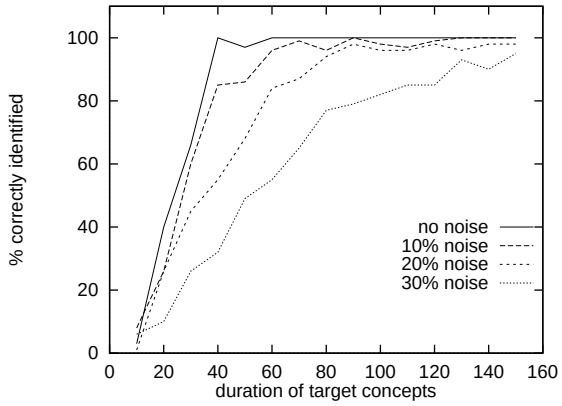

Figure 9: Identification of concept 2

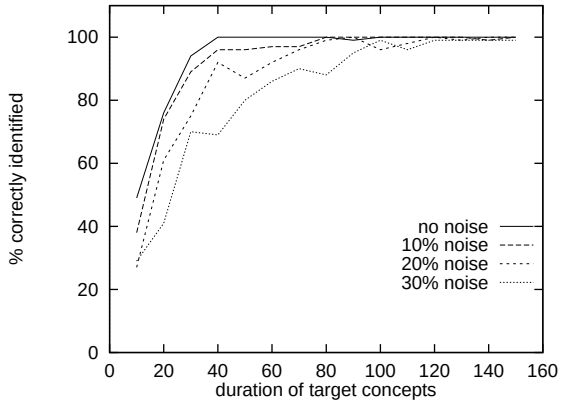

Figure 10: Identification of concept 3

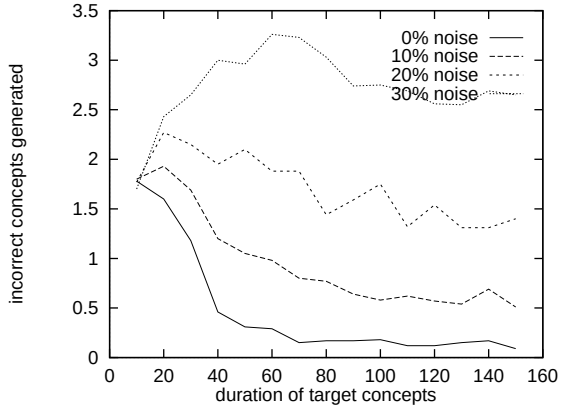

Figure 11: Incorrect concepts identified

and Granger, 1986). The combination of Splice with a simple strategy for selection of the current local concept was demonstrated to be effective on a prediction task. As the selection mechanism assumes that at least one of the local concepts will be correct, Splice almost immediately moves to its maximum accuracy on each new local concept. In published results on a similar domain the Flora family (Widmer and Kubat, 1996) (in particular Flora 3, the learner designed to exploit recurring context) appear to reach much the same level of accuracy as Splice. As expected for an on-line learning approach, however, they require some time to fully reflect changes in context.

This comparison is problematic for a number of reasons. Splice has the advantage of first seeing a training set containing 50 instances of each context before beginning to classify and of being able to assume that all possible contexts had at least been seen. The iterative learners have the advantage of continuous feedback with an unconstrained updating of concepts. Splice does have feedback, but can only select from amongst its current local concepts in adaptation to the feedback. Once Splice misses a local concept, there is no second chance. If Splice can be shown to be effective on more complex domains, it may be possible and desirable to use some combination of Splice with an adaptive learner.

Experiment 3 demonstrates that on this domain, Splice is well behaved with changing levels of noise and duration of target concept. Splice was able to take advantage of additional concept duration in order to minimise the effect of noise. At all noise levels, the number of spurious concepts identified by Splice fell within reasonable bounds. This is particularly promising, given that no heuristics were added to disqualify poor candidate local concepts.

The reason that Splice has been able to perform well is that it is able to identify stable concepts under a range of concept duration and noise. We conjecture that its ability to deal with noise is enhanced by the ability to combine similar intervals of the data set. It can use all the data set at one time and benefit from the accuracy achieved by the underlying learner. The potential for combining similar contexts and hence building more accurate concepts is substantial.

We see the main application for Splice, not in direct prediction, but rather as the first stage in identifying and reasoning about context in domains when context is hidden.

4.1 APPLICATION OF SPLICE TO PREDICTION TASKS

The above experiments demonstrate that Splice can successfully learn local concepts from hidden contexts in a simple domain. A number of issues arise in the possible application of Splice to more complex domains and real world problems. Two of these are re-training and identification of the relevant local concept.

In a real world predictive application, we anticipate that new training could be triggered by poor predictive performance or by recognising that new cases fall in previously unseen areas of attribute space (Harries and Horn, 1995). Re-training could either employ all past cases or employ a window overlapping the previous training set. If the second of these options were used, any new local concepts thus identified by re-training could be added to the set of previously identified local concepts. This scheme avoids the need to maintain a full history of seen examples.

In Experiment 2, a very simple heuristic was used to select the relevant local concept employed for prediction. More complex domains will require more complex methods to identify the relevant local concept. One approach may be to reason about duration and likely sequences of the identified contexts as suggested in (Widmer, 1996).

Some on-line learners can use the proceeding concept to seed the development of new concepts. Splice is unable to share structure across contexts. It is possible that this limitation will reduce the effectiveness of Splice in situations with slow drift. On the other hand, this characteristic is an advantage in situations with sudden drift. This is also a problem in situations where concepts are disjunctive and drift occurs only in one part of the concept.

Another exciting prospect is to use Splice with an underlying relational learner.

5 CONCLUSION

A new meta-learning algorithm, Splice, enables an underlying machine learning system to learn local concepts from domains with hidden changes in context. We demonstrated that the Splice approach is viable in at least a simple domain and is robust in the presence of noise in the domain presented.

The results indicate that Splice works well on the sample domain. It is essential that scaling issues be investigated in future work. We expect to apply additional heuristics to rate the viability of local concepts and to minimise time complexity.

We also aim to investigate methods for utilising local concepts in predictive applications, and methods for reasoning about the local concepts and contexts identified.

The meta-learning approach will allow us to investigate the application of Splice to problems that require a more complex inductive paradigm. Our only requirement is that we are able to extract information about context change in order to partition the data set by context. Initially, we will investigate relational learners such as FOIL. Other ILP systems and more complex inductive systems will then be considered.

6 ACKNOWLEDGEMENTS

Michael Harries is supported by an Australian Postgraduate Award (Industrial).

References

Clark, P. and Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, 3:261–283.

Domingos, P. (in press). Context-sensitive feature selection for lazy learners. *to appear in Artificial Intelligence Review*.

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

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

Kubat, M. (1989). Floating approximation in time-varying knowledge bases. *Pattern Recognition Letters*, 10:223–227.

Kubat, M. and Widmer, G. (1995). Adapting to drift in continuous domains. In *Proceedings of the 8th European Conference on Machine Learning*, pages 307–310, Berlin. Springer.

Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning*, 5:239–266.

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

Salganicoff, M. (1993). Density adaptive learning and forgetting. In *Machine Learning Proceedings of the Tenth International Conference*, pages 276–283, 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. (1996). Recognition and exploitation of contextual clues via incremental meta-learning. Technical Report oefai-96-01, Austrian Research Institute for Artificial Intelligence.

Widmer, G. and Kubat, M. (1993). Effective learning in dynamic environments by explicit concept tracking. In Brazdil, P. B., editor, *European Conference on Machine Learning*, pages 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.