Extracting Hidden Context*

Michael Harries Claude Sammut
Department of Artificial Intelligence
School of Computer Science and Engineering
The University of New South Wales, Sydney 2052, Australia.
E-mail: {mbh,claude}@cse.unsw.edu.au

Kim Horn
RMB Australia Limited,
Level 5 Underwood House,
37-47 Pitt Street, Sydney 2000, Australia.
E-mail: kim@rmb.com.au

UNSW-CSE-TR-9708 — November 1997

THE UNIVERSITY OF
NEW SOUTH WALES

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

*Submitted to the Machine Learning Journal for the special issue on Context-Sensitive Learning (1998).

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. An off-line, meta-learning approach for the identification of hidden context is presented. The new 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 the extraction of context reflected in time and spacial attributes. Several algorithms for the approach are presented and evaluated. A successful application of the approach to a complex control task is also presented.

1 Introduction

Prediction in real world domains is complicated by potentially unstable underlying phenomena. Financial market behaviour, for instance, can change dramatically with changes in contract prices, interest rates, inflation rates, budget announcements, and political and world events. Thus, concept definitions that may have been learned in one context become invalid in a new context. This *concept drift* can be 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.

Machine learning approaches can be broadly categorised as either batch or incremental. Batch systems learn off-line by examining a large collection of instances *en masse* and form a single concept. Incremental systems evolve and change a concept definition as new observations are processed [25].

The most common approach to learning in domains with hidden changes in context has been to use an incremental learning approach in which the importance of older items is progressively decayed. A popular implementation of this, originally presented in [11], is to use a window of recent instances from which concept updates are derived. Other examples of this approach include [30], [13], [10], and [22]. Swift adaption to changes in context can be achieved by dynamically varying the window size in response to changes in accuracy and concept complexity [30].

There are many domains in which the context can be expected not only to change but for earlier contexts to hold again at some time in the future. That is, contexts can repeat in domains such as financial prediction, dynamic control, and under represented data mining tasks. In these domains, prediction accuracy can be improved by storing knowledge about past contexts for re-use. FLORA3 [29] addresses domains in which contexts recur by storing and retrieving concepts that appear stable as the learner traverses the series of input data.

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 incorporate time as an essential attribute, for example, financial records and stock market price data. Interest in mining datasets of this nature suggests the need for systems that can learn global concepts and are sensitive to changing and hidden contexts. Systems such as FLORA3 also show that an off-line recognition of stable concepts would be useful for on-line prediction. The use of batch (off-line) learning in domains with hidden changes in context has not been extensively explored.

An alternative to on-line learning for domains with hidden changes in

context is to examine the data *en masse* in an attempt to directly identify concepts associated with stable, hidden contexts. Some potential benefits of such an approach are listed below.

- Context specific (known as *local*) concepts could be used as part of a multiple model on-line predictive system.
- Local concepts could be verified by experts, or used to improve domain understanding.
- The identified hidden contexts could be reasoned about, allowing a global concept to be augmented with expectations of hidden context duration, order, and stability.
- The identified hidden contexts could provide target characteristics for selecting additional attributes from the outside world as part of an iterative data mining process.

This article presents *Splice*, a meta-learning system that implements a context sensitive batch learning approach. Splice is designed to identify intervals with stable hidden context, and to induce and refine local concepts associated with these hidden contexts.

We proceed by describing the use of existing machine learners for detecting changes in context. The first implementation of Splice is then presented and evaluated. Some shortcomings of the method are discussed and a new method, Splice-2, designed to improve upon these shortcomings, is presented and evaluated. An application of Splice-2 to a complex task [23] is also presented.

1.1 Identifying Context Change

In many domains with hidden changes in context, time can be used to differentiate hidden contexts. Existing machine learning approaches to these domains do not explicitly represent time as they assume that current context can be captured by focusing on recent examples. The implication is that hidden context will be reflected in contiguous intervals of time. For example, an attempt to build a system to predict changes in the stock market could produce the following decision tree.

```
Year > 1992
   Year < 1995
      Attribute A = true: Market Rising
      Attribute A = false: Market Falling
   Year >= 1995
      Attribute B = true: Market Rising
      Attribute B = false: Market Falling
```

This tree contains embedded knowledge about two intervals of time: in one of these, 1992 to 1994, attribute A is predictive; in the other, 1995 onward, attribute B is predictive. As time (in this case, Year) is a monotonically increasing attribute, future classification using this decision tree will only use attribute B. If this domain can be expected to have recurring hidden context, information about the prior interval of time could be valuable.

The decision tree in the example above contains information about changes in context. We define context as:

> Context is any attribute whose values tend to be stable over contiguous intervals of another attribute known as the environmental attribute.

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) [4]. A number of machine learning algorithms can be regarded as using CSFS including decision tree algorithms [19], rule induction algorithms [2] and ILP systems [18]. All of these systems produce concepts containing local information about context.

When contiguous intervals of time reflect a hidden attribute or context, we call time the *environmental* attribute. The environmental attribute is not restricted to time alone as it 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.

Given an environmental attribute, we can utilise a CSFS machine learning algorithm to gain information on likely hidden changes in context. The accuracy of the change points found will be dependent upon at least hidden context duration, the number of different contexts, the complexity of each local concept, and noise.

The CSFS identified context change points can be expected to contain errors of the following types:

- Noise or serial correlation errors, taking the form of additional incorrect change points.
- Errors due to the repetition of tests on time in different parts of the concept. These would take the form of a group of values clustered around the hidden changes in context.
- Errors of omission, changes in context missed altogether.

The initial set of identified context changes can be refined by contextual clustering. Contextual clustering 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.

2 Splice-1

Splice-1 [8] is designed to recognise stable context and extract local concepts from domains with hidden changes in context. Splice-1 can use any ordinal attribute as the environmental attribute, in order to preserve clarity in the following discussion we have substituted *Time* for the broader term *environmental attribute*.

Splice-1 is a meta-level algorithm that incorporates an existing batch learner. In this study the underlying learner is the decision tree learner, C4.5 [19], with no modifications. The underlying learner for Splice-1 could, in principle, be replaced by any other CSFS machine learner able to provide splits on time. As we expect the underlying learner to deal with noise, Splice-1 does not have (or need) a mechanism to deal with noise directly.

The Splice-1 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.

Partition Dataset.

Splice-1 first uses the underlying CSFS learner to build an initial concept from the whole data set. As Splice-1 uses C4.5 the initial concept is a decision tree. By learning a concept description of the whole domain including time, we can identify splits in time that were important for concept description. Each of these splits is interpreted as a possible change in context. Each split on time is extracted from the initial concept and used to define both *intervals* of the dataset and the associated fragments of the initial concept, termed *partial* concepts. Each partial concept consists of the rules embedded in the leaves of the original decision tree that would act upon examples in the same interval of the environmental attribute.

Perform Contextual Clustering.

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

Splice-1 determines the accuracy of each partial concept on the examples in each interval[1]. The error rate for each combination of partial concept and interval 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

[1]The initial concept is reused by shifting the data series associated with each interval into the relevant time values for each partial concept.

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-1 output consists of all local concepts produced.

Figure 1: The Splice-1 Algorithm

the accuracy threshold parameter θ. The default setting for θ is 10%. 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 clustering procedure operates as follows. The partial concept with the highest coverage is selected from the set X. All the intervals that it covers are used to form a new cluster. The partial concepts associated with these intervals are removed from the set X. This step is then repeated with the next best candidate concept until set X is empty.

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-1 is able to exploit recurring contexts for improved local concept quality by building larger combined data sets.

3 Splice-1 Performance

This section first provides an introduction to the artificial domain upon which Splice has been evaluated. It continues with a walk through of the Splice-1 algorithm on a problem drawn from the sample domain, then describes investigations into Splice-1 performance in a prediction task and Splice-1 accuracy in local concept identification.

3.1 STAGGER Data Set

The data sets used in the following experiments are based on those used in evaluating STAGGER [25] and subsequently used by [30]. 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[2] 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.

[2]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 [30].

Table 1: Local Accuracy Matrix: Partial concept error on individual *intervals* (%).

Interval	Range	Partial concepts											
		0	1	2	3	4	5	6	7	8	9	10	11
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

The program randomly generates a series of examples from the above attribute space. Each example is given a unique time stamp and a boolean classification based upon one of three target concepts. The target concepts are:

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

Artificial contexts were created by fixing the target concepts to one of the above STAGGER concepts for preset intervals of the data series.

3.2 Splice-1 Walk through

Splice-1 is applied to a simple dataset with recurring concepts. The training set consists of STAGGER 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.

Partition Dataset.

The initial concept decision tree in Figure 2, as generated by C4.5, is not succinct nor is it easy to interpret. It does, however, contain a substantial amount of information on past contexts and changes in context. Splits identified on time in the initial concept are used to define both partial concepts and intervals of the data set.

```
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)

```
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

Contextual Clustering.

Each partial concept is evaluated on all intervals of the data set. Table 1, the Local Accuracy Matrix (LAM), shows the error rate achieved by each partial concept upon instances drawn from each interval. For instance, partial concept 2 had an error rate of 44% on the data set drawn from interval 0 and an error rate of 0% on the data set drawn from interval 1. Some of the intervals shown in the LAM have a very brief duration, shown in the range column, and are the result of spurious splits on time in the initial concept. The LAM is used as input for the clustering operation[3].

Local Concepts.

The contextual clusters are then used to produce local concepts, shown in Figure 3. For the first two local concepts generated, Splice-1 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.

Splice-1 must both correctly identify target concepts and minimise the number of incorrectly identified concepts. On the same task, Splice-1 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-1 also identified, on average, 0.8 incorrect or spurious concepts per trial.

The identification of local concepts alone does not provide much utility. A more pressing question is the application of local concepts in a prediction task.

3.3 Prediction: Splice-1 'vs' C4.5

Splice-1 local concepts can be effectively used for on-line prediction in a domain with hidden changes in context. This experiment compares the accuracy of Splice-1 to C4.5 when trained on a data set containing changes in a hidden context. After training, the resulting concepts were used for prediction on a similar data set. C4.5 is provides 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 consisted of concepts (1) for 50 instances, (2) for 50 instances, and (3) for 50 instances. The test set consisted 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.

[3]In all experiments reported Splice-1 was run with the threshold accuracy parameter θ set to a default of 10%. Unless otherwise noted, the underlying learner, C4.5, was run with default pruning parameters and with sub-setting.

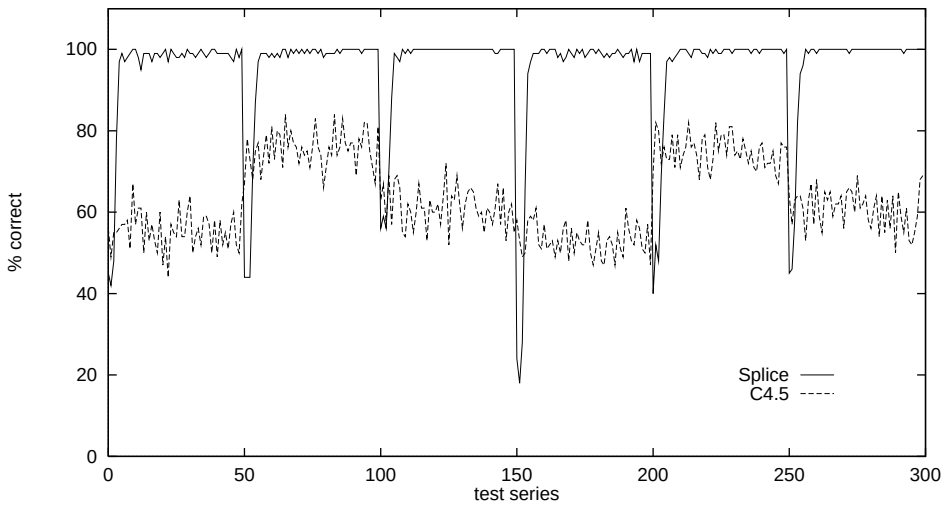

Figure 4: Splice-1, C4.5 comparison, trained with no noise

To apply the local concepts identified by Splice-1 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 arbitrarily 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 a randomly selected local concept.

Results

Figures 4 and 5 show the average classification success rates at several levels of noise for both Splice-1 and C4.5 over 100 randomly generated training and test sets. Noise was generated only in the training set.

Figure 4 shows that Splice-1 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 performs 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.

As noise increases, the performance of Splice-1 gradually declines. Figure 5 shows that at 30% noise, the worst result achieved by Splice-1 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.

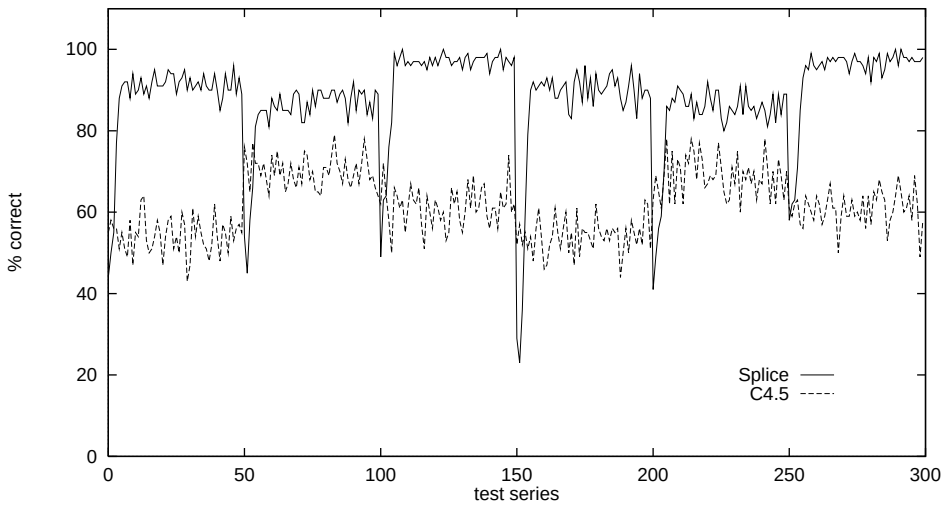

Figure 5: Splice-1, C4.5 comparison, trained with 30% noise

This task is similar to the on-line learning task tackled by both FLORA [30] and STAGGER [25]. The combination of Splice-1 with a simple strategy for selection of the current local concept is effective on a simple context sensitive prediction task. As the selection mechanism assumes that at least one of the local concepts will be correct, Splice-1 almost immediately moves to its maximum accuracy on each new local concept. On a similar domain the FLORA family [30] (in particular FLORA3, the learner designed to exploit recurring context) appear to reach much the same level of accuracy as Splice-1, although as an on-line learning method, FLORA requires some time to fully reflect changes in context.

This comparison is problematic for a number of reasons. Splice-1 has the advantage of first seeing a training set containing 50 instances of each context before beginning to classify and of being correct in the assumption 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-1 does have feedback, but is constrained to its current local concepts in adaptation to the feedback. When Splice-1 has not learnt a local concept, there is no second chance. For more complex, real world domains, it could be beneficial to use a combination of Splice-1 and an adaptive, on-line learner.

This experiment looked at prediction given a single concept duration. How well does Splice-1 perform on different levels of noise and concept duration?

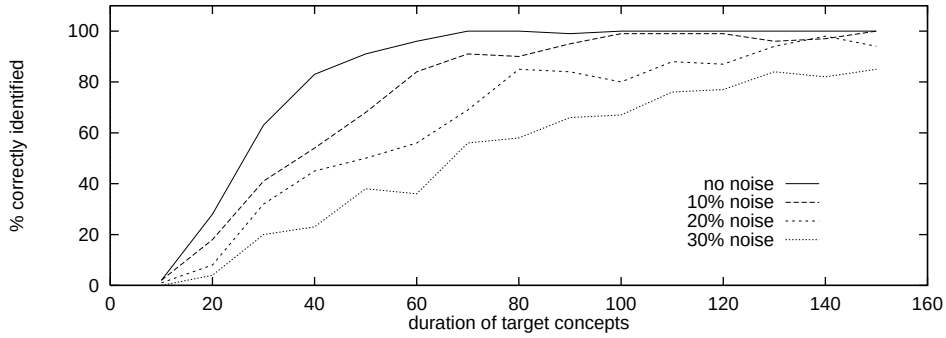

Figure 6: Splice-1 identification of concept 1

3.4 The Effects of Noise and Duration on Splice-1

Splice-1 is able to correctly induce the three STAGGER concepts over a range of noise and context duration conditions. Splice-1 exploits longer concept durations to offset the negative effects of additional noise. Splice-1 was trained on a randomly generated training set containing examples of each of the STAGGER concepts. The set of local concepts learnt by Splice-1 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 incorrect 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 combination of noise level and concept duration was repeated 100 times.

Results

Figures 6, 7 and 8 show the accuracy of Splice-1 in correctly identifying each target concept under varying levels of both concept duration and noise. In this domain, Splice-1 is well behaved, with a graceful degradation of performance as noise levels increase. Concept duration reduces the negative effect of noise. Figure 9 shows the number of incorrect concepts learnt by Splice-1 for different levels of noise and training concept duration. In this too, Splice-1 is well behaved, showing both graceful degradation of performance with increased noise, and well bounded numbers of incorrect concepts learnt.

These results show that on this domain, Splice-1 is well behaved with changing levels of noise and duration of target concept. Splice-1 was able to take advantage of additional concept duration in order to minimise the

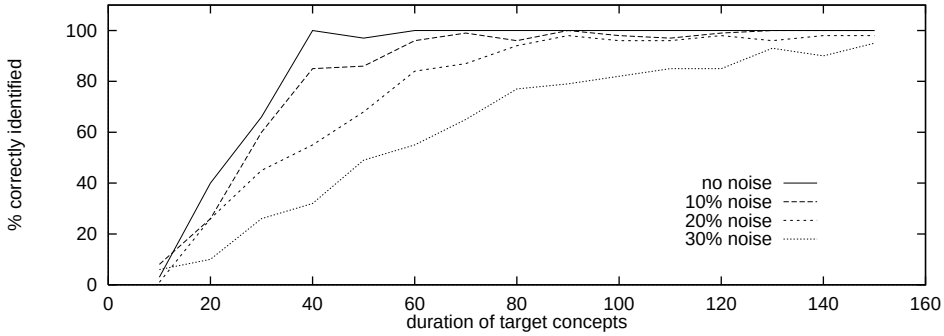

Figure 7: Splice-1 identification of concept 2

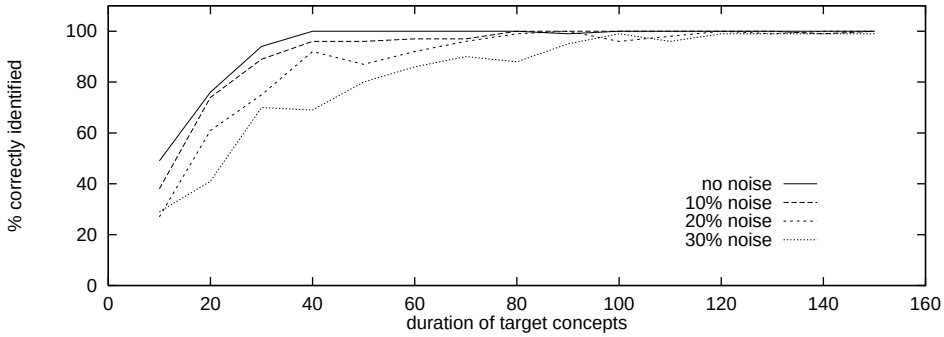

Figure 8: Splice-1 identification of concept 3

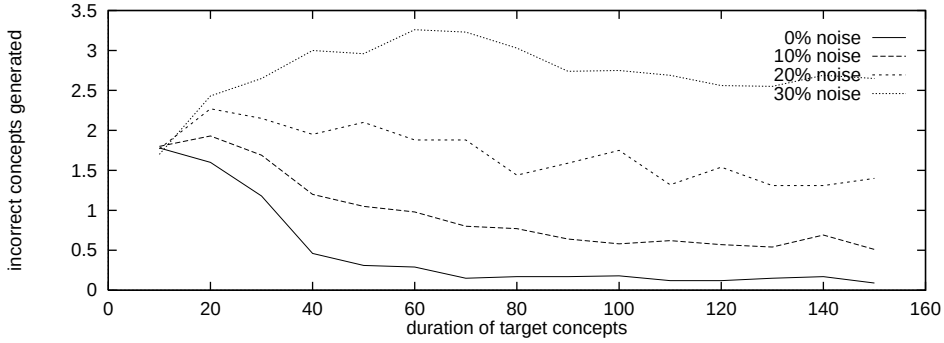

Figure 9: Splice-1 incorrect concepts identified

effect of noise. At all noise levels, the number of spurious concepts identified fell within reasonable bounds.

While the results for Splice-1 are promising, the quality of context recognition by Splice-1 is largely dependent upon the accuracy of partitions found by the CSFS partitioning. In other words, Splice-1 is restricted to domains upon which the CSFS partitioning is effective. An alternate algorithm, Splice-2, is designed for domains with high levels of noise and context repetition.

4 Splice-2

Splice-1 is dependent for initial partitioning upon the ability of the underlying CSFS algorithm to correctly detect changes in context. The difficulty of inducing the correct classifier has been related to the number of regions [19] or peaks [20] that must be described. Another measure of domain difficulty is the average information entropy in the concept over all relevant attributes [20]. Drawing upon these definitions for domain difficulty, we anticipate that the CSFS algorithm will be more likely to miss context changes as the following domain characteristics increase:

- Context repetition.
- Irrelevant attributes.
- Noise.

Splice-2 is designed to minimise the impact of poor initial partitioning.

The Splice-2 algorithm is detailed in Figure 10. Splice-2 begins by partitioning the data set using either the CSFS algorithm, a random split or domain knowledge. We denote the version of Splice-2 using random partitioning as Splice-2R. Splice-2R has no restriction that the underlying machine learning algorithm use CSFS. These partitions form the initial contextual clusters (CCs). C4.5 is then used on each CC, to create the initial interim concepts.

The next stage, contextual clustering, clusters individual items on the basis of interim concept accuracy on a fixed size window surrounding each original data item. (This replaces Splice-1 contextual clustering, which clusters only intervals created in the partitioning stage.) New CCs are created by clustering initial dataset items according to similarity of context, as represented by the interim concepts. The new CCs are then used as training sets for the creation of new interim concepts.

The contextual clustering stage can be iterated for further contextual context refinement. The final set of interim concepts form the output *local concepts*. The boundaries between the final seed sets form context boundaries. We now examine contextual clustering in more detail.

Splice-2 algorithm: *(Input: Environmental attribute ordered data set.)*

- Stage 1: Partition Dataset
  - Partition the dataset over the environmental attribute using either:
    - C4.5 as per Splice-1.
    - A pre-set number of random splits.
    - Prior domain knowledge.
  - The identified partitions form the initial contextual clusters (CCs) and are in turn used to create the initial interim concepts.
- Stage 2: Contextual Clustering
  - Each combination of interim concept and item in the original data set is allocated a score based upon the total accuracy of that concept on items in a fixed size window over the environmental attribute surrounding the item.
  - Cluster the original data set items that share maximum scores with the same interim concept. These clusters form the new set of CCs.
  - Create a new set of interim concepts from the new CCs.
  - Stage 2 is repeated until the interim concepts do not change or until a fixed number of iterations are completed.
- Stage 3: Create Local Concepts
  - The final set of interim concepts form the output local concepts.

Figure 10: The Splice-2 Algorithm

4.1 Contextual Clustering

This stage clusters on similarity of context where context is represented by each current interim concept. This similarity of context is based on a shared interim concept classification accuracy on a window surrounding each data item. We define a simple measure W_{ij} to represent the similarity of context between an interim concept j and example i:

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

where:

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

New CCs are built by allocating each example i to a contextual cluster with other examples whose maximum weight W_{ij} is associated with the same interim concept j. A new set of interim concepts is then created by applying C4.5 to the new CCs. Contextual clustering is halted after a given number of iterations.

5 Splice-2 Performance

This section shows two experiments with Splice-2. The first is a direct comparison with Splice-1 on the recognition of STAGGER concepts in a domain with a range of noise and duration and a fixed number of hidden changes in context. The second compares the concept recognition abilities of Splice-1, Splice-2, and Splice-2R on a range of noise and context change levels with a fixed duration.

5.1 Concept Recognition: Splice-1 'vs' Splice-2

Splice-2 is shown to be superior to Splice-1 on a series of STAGGER concept recognition task with many changes in context.

The basic STAGGER concept recognition problem presented in Section 3.4 was altered by repeating the training set five times giving a total of 14 changes in context. Both versions of Splice were trained on datasets with a range of noise and context duration. The sets of local concepts learnt were assessed against each target concept. The results show the average number of correct local concept identifications achieved, and the average number of incorrect local concepts identified.

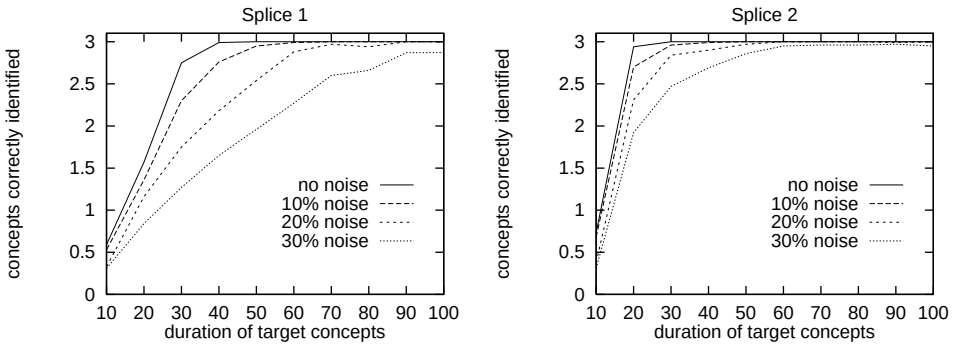

Figure 11: Number of concepts correctly recognised by Splice-1 and Splice-2

Training sets were generated using a range of context duration and noise. Context duration corresponds to the number of instances for which a concept is active. Each training set consists of random examples classified according to the following pattern of contexts: 5 repetitions of the following structure (concept (1) for D instances; concept (2) for D instances; and concept (3) for D instances). Where the duration D ranges from 10 instances to 100 instances and noise ranges from 0% to 30%. Results are an average of 100 repetitions of each combination of noise and duration.

Results

Figure 11 shows the average number of STAGGER concepts correctly recognised by Splice-1 and Splice-2[4]. for each context duration and noise level. Both versions of Splice converge upon the recognition of all concepts at higher context durations. Splice-2 converges upon the recognition of all three local concepts more quickly for all levels of noise.

Figure 12 shows the number of incorrect local concepts induced by Splice-1 and Splice-2. These charts show the number of incorrect concepts induced for each context duration and noise level. For all levels of noise, Splice-1 induces more incorrect concepts than Splice-2. The number of incorrect concepts is similar only for 0% noise and high context duration.

On this task, both versions of Splice respond well to a range of training set noise and context duration. Both versions exploit increased context durations to minimise the effect of noise. Splice-2 consistently induces more correct and less incorrect local concepts than Splice-1. These results suggests that the Splice-2 clustering mechanism is better able to overcome the effects of frequent context changes and high levels of noise. In the next experiment we further investigate this result by fixing context duration and

[4]In all experiments reported, Splice-2 was run with three contextual clustering iterations. Unless otherwise noted, the underlying learner, C4.5 was run with default parameters and with sub-setting.

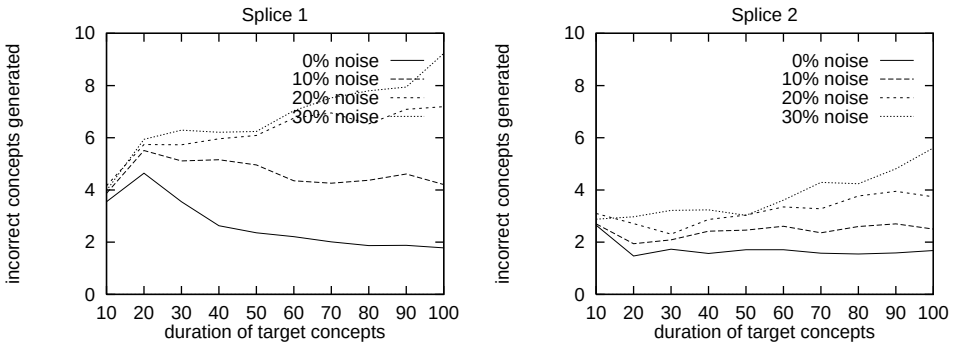

Figure 12: Incorrect concepts found by both Splice-1 and Splice-2

testing different levels of context change.

5.2 Splice-1, Splice-2, and Random Seeding

This experiment shows that high levels of context repetition can lead to a reduction in the recognition accuracy of Splice-1. Splice-2 is shown to improve with high levels of context repetition. Splice-2 with random partitioning is also shown to be effective on these domains.

The previous experiment looked at the effects of different context durations on local concept recognition by both Splice-1 and Splice-2. While this demonstrated that Splice-2 has a superior accuracy on a fixed number of context changes, it provided little insight into the effect of different levels of context change. This experiment investigates the effects of different levels of context repetition and noise. We first compare the concept recognition accuracy of Splice-1 with that of Splice-2. The Splice-2 results are subsequently compared with the accuracy achieved by Splice-2R (Splice-2 with random partitioning).

Each algorithm was evaluated on its ability to correctly induce the STAGGER local concepts from a training set containing hidden changes in context. The training set consisted of randomly generated STAGGER instances classified according to the following pattern of contexts: R repetitions of the structure (concept (1) for 50 instances; concept (2) for 50 instances; concept (3) for 50 instances) where R varies from one to five. The effects of noise were also evaluated with a range of noise from 0% to 40%. The results shown are an average based upon 100 iterations of each combination of R and noise. Splice-2R used 10 random partitions.

Results

Figure 13 shows the number of STAGGER concepts correctly induced by both Splice-1 and Splice-2 for each combination of context repetition and noise.

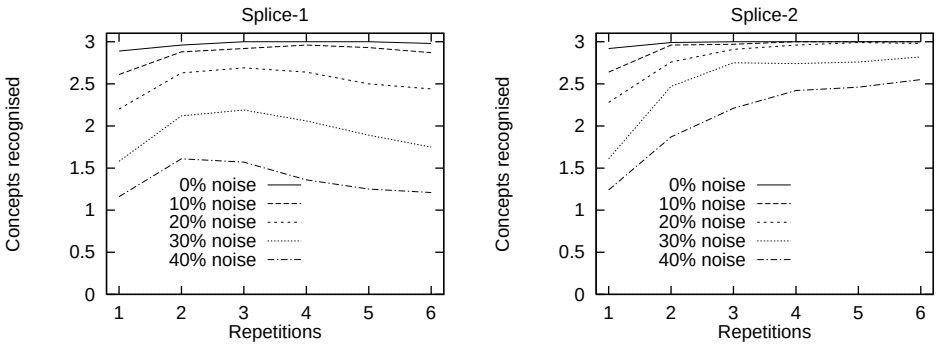

Figure 13: Splice-1 and Splice-2 concept recognition

The primary feature of the Splice-1 chart is an initial rise and subsequent decline in concept recognition as the number of repetitions increases. The only exception from this is at 0% noise, for which Splice-1 does not decline in accuracy. The Splice-2 chart shows increasing concept recognition (up to a maximum of three concepts) with increases in repetition.

Splice-1 and Splice-2 achieve similar levels of concept recognition for a single iteration of the contexts (repetition = 1). Splice-2 recognises more concepts for all levels of repetition greater than one. The exception is at 0% noise, for which both versions recognise all three concepts for repetition levels of three and more.

Splice-1 is not competitive on more than one context repetition. In fact, greater numbers of context change have a negative effect on the accuracy achieved. This is due to the failure of the partitioning method on domains with many changes in context. Splice-1 is still interesting, as it does substantially less work than Splice-2, and can be effective on domains with relatively few context changes. We anticipate that a stronger partitioning method would make Splice-1 more resilient to frequent changes in context.

The Splice-2 algorithm, on the other hand, improves concept recognition as context repetition increases. Splice-2 is not effected by poor initial partitioning as it re-builds context boundaries at each iteration of contextual clustering. Hence, a poor initial partition has a minimal effect and Splice-2 is able to take advantage of increases in context examples.

Figure 14 shows the number of correct STAGGER concepts induced by Splice-2R with 10 random partitions. This chart shows a rise in recognition accuracy as repetitions increase (up to the maximum of 3 concepts recognised) for all noise levels. The number of concepts recognised is similar to those in Figure 13 for Splice-2.

The similarity of results for Splice-2 and Splice-2R shows that, for this domain, CSFS partitioning provides no benefit over the use of random partitioning for Splice-2. On more complex domains the bias provided by the initial partitioning can effect Splice-2 accuracy. The results attained by

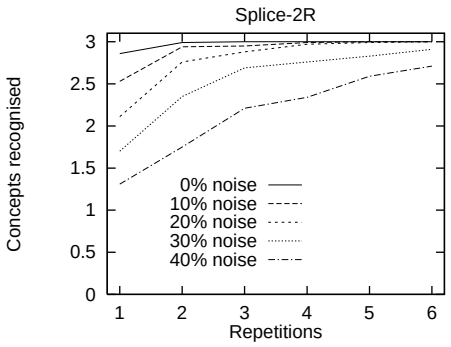

Figure 14: Splice-2R concept recognition

Splice-2R provide another indication of the power of the Splice-2 clustering method.

6 Learning to Fly

To test the Splice-2 methodology, we wished to apply it to a substantially more complex domain than than the artificial data described above. We had available, data collected from flight simulation experiments used in behavioural cloning [23]. 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. In the following sections we briefly describe the problem domain and the application of Splice-2.

6.1 Domain

The "Learning to Fly" experiments [23] 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 that requires a high degree of skill to operate successfully and yet is well understood. The experimental setup was to collect data from several human subjects flying a predetermined flight plan. These data would then be input to an induction program, C4.5.

The flight plan provided to the human subjects was:

1. Take off and fly to an altitude of 2,000 feet.
2. Level out and fly to a distance of 32,000 feet from the starting point
3. Turn right to a compass heading of approximately 330 degrees.

4. At a North/South distance of 42,000 feet, turn left to head back towards the runway. [...] The turn is considered complete when the azimuth is between 140 degrees and 180 degrees.

5. Line up on the runway.

6. Descend to the runway, keeping in line.

7. Land on the runway.

The log includes 15 attributes showing position and motion and 4 control attributes. The position and motion attributes were: on_ground, g_limit, wing_stall, twist, elevation, azimuth, roll_speed, elevation_speed, azimuth_speed, airspeed, climbspeed, E/W distance, altitude, N/S distance, fuel. The control attributes were: rollers, elevator, thrust and flaps. (The rudder was not used as its implementation was unrealistic.) Decision trees induced from the logged data were tested by compiling the trees into the autopilot code of the simulator and then "flying" the simulator.

In the original experiments, three subjects flew the above flight plan 30 times each. In all, a data set of about 90,000 records was produced. Originally, it was thought that the combined data could be submitted to the learning program. This proved too complex a task for the learning systems that were available. The problems were largely due to mixing data from different contexts.

The first, and most critical type of context, was the pilot. Different pilots have different flying styles, so their responses to the same situation could differ. Hence, the flights were separated according to pilot. Furthermore, the actions of a given pilot differ according to the stage of the flight. That is, the pilot adopts different strategies depending on whether he or she is turning the aircraft, climbing, landing, etc. To succeed, an induction program would have to be able to distinguish these different cases. The classification learning programs available, could not do this, so further manual separation of the data into flight stages was required. Since the pilots were given intermediate flight goals, the division into stages was not too onerous. Not all divisions were immediately obvious. In the initial division, for example, lining up and descending were not separated into two different stages. However, without this separation, the decision trees generated by C4.5 would miss the runway. It was not until the "line-up" stage was introduced that a successful "behavioural clone" could be produced.

Until now, the stages used in behavioural cloning could only be found through human intervention which often included quite a lot of trial-and-error experimentation. The work described below suggests that flight stages can be treated as different contexts and that the Splice-2 approach can automate the separate of flight data into appropriate contexts for learning.

6.2 Flying with Splice-2

This domain introduces an additional difficulty. Since there are four control actions available to the pilot, previous behavioural cloning experiments built decisions trees for each of the four actions in each of the seven stages, resulting in 28 decision trees that are switched in depending on the current stage.

When Splice-2 is applied to the four learning tasks, *viz*, building a controller for elevators, another for rollers, for thrust and flaps, there is no guarantee that exactly the same context divisions will be found. This causes problems when two or more actions must be coordinated. For example, to turn the aircraft, rollers and elevators must be used together. If the contexts for these two actions do not coincide then a new roller action, say, may be commenced, but the corresponding elevator action may not start at the same time, thus causing a lack of coordination and a failure to execute the correct manoeuvre.

This problem was avoided by combining rollers and elevators into a single attribute, corresponding to the stick position. Since the rollers can take one of 15 discrete values and elevators can take one of 11 discrete values, the combined attribute has 165 possible values. Of these, 97 are represented.

Switching between the local concepts during the flight is not trivial as no (immediate) feedback on classification accuracy is available. With the previous experiments reported in this paper, a voting mechanism was used to select the current context and local concept. For this domain, we chose to use a decision tree for context selection. All examples from each final local concept were labelled with the relevant local concept. A decision tree was induced using the same attributes as provided for learning the individual local concepts and subsequently used to select the current context for each instant of the flight.

We also found that the original Splice-2 W_{ij} formula using only classification accuracy did not perform well when class frequencies were wildly different. We found that well represented classes were dominating the contextual clustering process leading to clusters with similar classification over well represented classes and dissimilar classification over poorly represented classes. This was problematic as successful flights depend upon the correct classification of rare classes. The problem was reduced by altering the weighting formula W_{ij} to give an equal importance to accuracy on all classes in a given window while ignoring the relative representations of the different classes. The $W_{ij}\prime$ formula is:

$$W_{ij}\prime = \sum_{c=1}^{C} \frac{\sum_{m=i-w/2}^{i+w/2}(c_m = c).Correct_{jm}}{\sum_{m=i-w/2}^{i+w/2}(c_m = c)} \tag{2}$$

where:

C is the number of classes
c_m is the class number of example m

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

Splice-2 was also augmented to recognise domain discontinuities such as the end of one flight and the beginning of the other by altering $W_{ij}\prime$ such that no predictions from a flight other than the flight of example i are incorporated in any $W_{ij}\prime$.

Results

We were able to successfully fly the first four stages of the flight by training on data extracted from only these stages for 30 flights. It should be noted that even with the changes in the domain (combining rollers and elevator) C4.5 is unable to make the first turn without the explicit division of the domain into stages.

Figure 15 shows three flights:

- The successful Splice-2 flight on stages 1 to 4.
- The best C4.5 flight.
- A sample complete flight.

The settings used in Splice-2 were:

- A window size of 50 instances.
- C4.5 post pruning turned off (-c 100).
- Three iterations of the clustering stage.
- Initial partitioning was set to four equal divisions of the first flight.

At present, the addition of further stages of the flight causes catastrophic interference between the first two stages and the last 3 stages. Splice-2 is, as yet, unable to completely distinguish these parts of the flight. However, the use of Splice-2 in synthesising controllers for stages 1 - 4 is the first time that any automated procedure has been successful for identifying contexts in this very complex domain.

The use of a decision tree to select the current context was reasonably effective but inelegant. The "learning to fly" problem combines a complex, well understood domain with a poorly represented task. This combination would make a good test bed for extending context selection methods by reasoning about context.

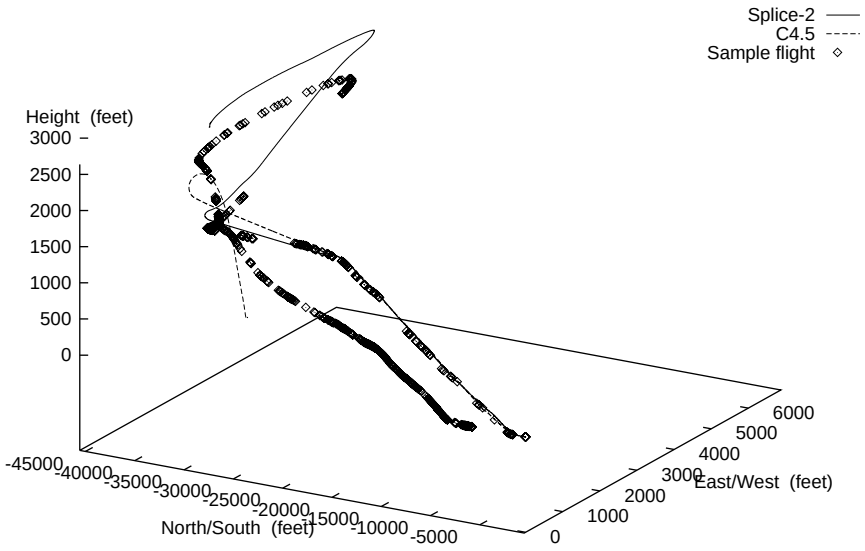

Figure 15: Flight comparison

7 Related Work

Splice is most closely related to the FLORA [30] family of on-line learners designed to adapt to hidden changes in context by drawing changes to the concept from a window of recent instances. Rapid adaption to changes in context is assured by altering the window size in response to changes in prediction accuracy and concept complexity. One version, FLORA3, [29] was adapted for domains with recurring hidden context by storing stable concepts for possible re-use when a context change is suspected. In order to guard against a concept drift, genuinely new concepts, and mistaken selection of a prior stable concept, the resurrected stable concept is updated to match examples in the current window. Splice extends the concept of storing stable concepts as an adjunct to on-line learning to be the primary focus of an off-line batch learning approach.

Most on-line learning methods that deal with concept drift decay the importance of older instances. STAGGER [24, 25], for instance, was probably the first machine learning system dealing with concept drift. As the system moves forward over the data series, a search frontier of conjunctive and disjunctive features is altered according to changes in statistics measuring logical sufficiency (LS) and logical necessity (LN). The goal is to converge upon a succinct concept description. The system allows for concept drift by backtracking on the search frontier when the statistics for a current charac-

terisation fall beneath a threshold. The failed characterisation is removed from the search frontier and replaced by alternate characterisations. Discarding characterisations that are no longer effective provides a decay in the importance of older information.

Many other on-line learners use an explicit window heuristic similar to that used in FLORA [11] [12] [13] [10]. Batch learners can also be used for on-line learning on domains with concept drift by repeatedly learning from a window of recent instances [7] [3]. The window update mechanism need not use a *first in, first out*, organisation, [22] discards older examples only when a new item appears in a similar region of attribute space.

On-line learners for domains with hidden changes in context assume that context will tend to be contiguous over time. Splice broadens this assumption by allowing context to be contiguous over any environmental attribute: usually time or space. Splice would need only minor changes to deal with an environmental attribute with several dimensions.

There is very little in the literature on dealing with hidden changes in context using a batch approach. This is probably because most batch learning methods assume that all available information will be directly represented in the attributes provided, hence, hidden changes in context will be treated as noise. We suspect that in domains where an unseen context is important for classification accuracy, reported work tends to focus upon successful representations (including an explicit representation of the previously hidden context). Such a process is rarely made explicit. One paper that does explicitly refer to an augmentation of the domain representation with a previously hidden context is [23] in which the task of learning to fly could not be achieved until the domain representation was augmented by splitting the learning task into sub-tasks.

There has been substantial work on dealing with known changes in context. One approach to a known context is to divide the domain into a series of different learning tasks, induce different classifiers for each task, then switch between these classifiers according to the current context. This method has been applied to the learning to fly domain [23] and to target recognition [9]. The application of local concepts for prediction and classification in this article used a similar model switching approach.

The transfer of knowledge learnt in one context to a new, previously unseen, context is similar to an on-line adaption to a hidden change of context. Knowledge embedded in a decision tree can be transfered to a new context [14] by applying a two tiered structure. The fixed decision tree is used as the first tier. The second tier, providing soft matching and weights for each leaf of the decision tree, is trained on the second context. This is similar to the two tiered structure originally proposed [15] for dealing with *flexible* contexts. Knowledge from an existing network can be used to significantly increase the speed of learning in a new context [16, 17] by using weights from the existing network to initialise the new neural network.

Methods for the transfer of knowledge from one context to another could be used to adapt Splice local concepts on-line in a manner analogous to that used by FLORA3.

Context has thus far been defined as any attribute whose values tend to be stable over contiguous intervals of the environmental attribute. Context can also be interpreted as the effect of one (contextual) attribute on the interpretation of another (context-sensitive) attribute. Classifier accuracy can be improved when context-sensitive attributes are present for both instance based learning [1] and multivariate regression by the methods of contextual normalisation, contextual expansion and contextual weighting [26, 27, 28]. Instance based learning could be used as the Splice-2R underlying learner, any hidden contexts thereby recognised by Splice-2R could then be utilised with these techniques for on-line prediction.

A somewhat different on-line method designed to detect and exploit contextual attributes is MetaL(B) [31]. In this case, contextual attributes are considered to be predictive of the relevance of other attributes. MetaL(B) works by using the detected contextual attributes to trigger changes to the set of features presented to the classifier. While this approach and context definition is quite different to that used by Splice, the overall philosophy is similar. Widmer concludes by stating that

> "... the identification of contextual features is a first step towards naming, and thus being able to reason about, contexts."

One long term goal for the Splice approach is precisely this.

To summarise, Splice begins to build a bridge between on-line methods for dealing with hidden changes in context and batch methods for dealing with known change in context. Splice applies the on-line assumption that contexts are liable to be contiguous over an environmental attribute to the broader problem of detecting and extracting hidden context and the associated concepts.

8 Conclusion

This article has presented a new off-line paradigm for recognising 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. Some domains with hidden context are data mining problems, financial market prediction, and behavioural cloning. Most previous work with hidden changes in context has used an on-line learning approach.

The new approach, Splice, uses off-line, batch, meta-learning to extract hidden context and induce the associated *local* concepts. It incorporates existing machine learning systems (in this article, C4.5 [19]). Two implementations of Splice were presented. The evaluation of the Splice approach

included an on-line prediction task, a series of hidden context recognition tasks, and a complex control task.

Splice-1 uses a context sensitive feature selection (CSFS) algorithm to divide a data series by likely changes of context. A process called contextual clustering is then applied to these intervals to group intervals that appear to be from the same context. This process uses the semantics of concepts induced from each interval as a measure of the similarity of context. The resulting clusters are used to create context specific concepts and to specify context boundaries.

Splice-2 differs from Splice-1 primarily in the method used for contextual clustering. Splice-2 clusters on the basis of individual members of the data series. Hence, context boundaries are not restricted to the boundaries found in the partitioning stage and context boundaries can be refined. Splice-2 is much more robust to the quality of the initial partitioning.

Splice-2 successfully detected and dealt with hidden context in a complex control task. "Learning to Fly" is a behavioral cloning domain based upon learning an autopilot given a series of sample flights with a fixed flight plan. Previous work on this domain required the user to specify stages of the flight. Splice-2 was able to successfully fly a substantial fragment of the initial flight plan without these stages (or contexts) being specified. This is the first time that any automated procedure has been successful for identifying context in this very complex domain.

A number of improvements could be made to the Splice algorithms presented. The partitioning method used was shown to be problematic for Splice-1 at high levels of noise and hidden changes in context. While the use of an existing CSFS machine learning system to provide partitioning is elegant, a better solution may be to implement a specialised method designed to deal with additional complexity on environmental attributes. One approach to this is to augment a decision tree algorithm to allow many splits [5] on selected attributes.

Neither Splice-1 or Splice-2 provide a direct comparison of the relative advantage of dividing the domain into one set of contexts over another. One comparison method that could be used is the minimum description length (MDL) [21] heuristic. The MDL principle is that the best theory for a given concept will minimise the amount of information that need be sent from a sender to a receiver so that the receiver can correctly classify items in a shared dataset. In this case, the information to be sent must contain any local concepts, a context switching method and a list of exceptions. At the very least, this would allow a direct comparison of a given context-sensitive global concept (using local concepts and context switching) with a context-insensitive global concept. Further, a contextual clustering method could use an MDL heuristic to guide a search through the possible context divisions.

The approaches used here for selecting the current context were an on-

line voting method for domains with immediate feedback and a decision tree for a domain without immediate feedback. More sophisticated approaches would use a model of the hidden context. Such a model could use knowledge about the expected context duration, order and stability. It could also incorporate other existing attributes and domain feedback. The decision tree used for context switching in the learning to fly task is a primitive implementation of such a model using only existing attributes to select the context.

An exciting possibility is to use the characteristics of contexts identified by Splice to guide a search of the external world for an attribute with similar characteristics. Any such attributes could then be incorporated with the current attribute set allowing a bootstrapping of the domain representation. This could be used within the Knowledge Discovery in Databases (KDD) approach [6] which includes the notion that analysts can reiterate the data selection and learning (data mining) tasks. Perhaps too, this method could provide a way for an automated agent to select potentially useful attributes from the outside world, with which to extend its existing domain knowledge.

9 Acknowledgements

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

References

[1] D.W. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

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

[3] Scott Clearwater, Tze-Pin Cheng, and Haym Hirsh. Incremental batch learning. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 366–370. Morgan Kaufmann Publishers, 1989.

[4] Pedro Domingos. Context-sensitive feature selection for lazy learners. *Artificial Intelligence Review*, 11:227–253, 1997. Special issue on lazy learning, edited by David Aha.

[5] Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *13th International Joint Conference on Artificial Intelligence*, pages 1022–1027, California, 1993. Morgan Kaufmann.

[6] Usama M. Fayyad, Gregory Piatsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery: An overview. In Usama M. Fayyad, Gregory Piatsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. MIT Press, 1996.

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

[8] M. Harries and K. Horn. Learning stable concepts in domains with hidden changes in context. In M. Kubat and G Widmer, editors, *Learning in context-sensitive domains (Workshop Notes)*. 13th International Conference on Machine Learning, Bari, Italy, 1996.

[9] A.J. Katz, M.T. Gately, and Collins D.R. Robust classifiers without robust features. *Neural Computation*, 2:472–479, 1990.

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

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

[12] M. Kubat. A machine learning based approach to load balancing in computer networks. *Cybernetics and Systems Journal*, 1992.

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

[14] Miroslav Kubat. Second tier for decision trees. In *Machine Learning: Proceedings of the 13th International Conference*, pages 293–301, California, 1996. Morgan Kaufmann.

[15] R.S. Michalski. Learning flexible concepts: Fundimental ideas and a method based on two-tiered representation. In Y. Kodratoff and R.S. Michalski, editors, *Machine Learning: An Artificial Intelligence Approach, Vol. III*. Morgan Kaufmann, 1990.

[16] L. Y. Pratt. Discriminability-based transfer between neural networks. In S. J. Hanson, C. L. Giles, and J. D. Cowan, editors, *Advances in Neural Information Processing Systems 5*, pages 204–211. Morgan Kaufmann, 1993.

[17] Lorien Y. Pratt and Candace A. Kamm. Direct transfer of learned information amoung neural networks. In *Proceedings 9th National Conference on Artificial Intelligence (AAAI-91)*, 1991.

[18] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

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

[20] Larry Rendell and Harish Ragavan. Improving the design of induction methods by analyzing algorithm functionality and data-based concept complexity. In *13th International Joint Conference on Artificial Intelligence*, pages 952–958, California, 1993. Morgan Kaufmann.

[21] J. Rissanen. A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11(2):416–431, 1983.

[22] M. Salganicoff. Density adaptive learning and forgetting. In *Machine Learning: Proceedings of the Tenth International Conference*, pages 276–283, San Mateo, California, 1993. Morgan Kaufmann Publishers.

[23] Claude Sammut, Scott Hurst, Dana Kedzier, and Donald Michie. Learning to fly. In *Machine Learning: Proceedings of the Ninth International Conference*, pages 385–393, San Mateo, California, 1992. Morgan Kaufmann Publishers.

[24] J. C. Schlimmer and R. I. Granger, Jr. Beyond incremental processing: Tracking concept drift. In *Proceedings AAAI-86*, pages 502–507, Los Altos, California, 1986. Morgan Kaufmann Publishers, Inc.

[25] Jeffory Schlimmer and Richard Granger, Jr. Incremental learning from noisy data. *Machine Learning*, 1(3):317–354, 1986.

[26] P. D. Turney. Exploiting context when learning to classify. In Pavel B. Brazdil, editor, *European Conference on Machine Learning*, pages 402–407, Berlin, 1993. Springer-Verlag.

[27] P. D. Turney. Robust classification with context sensitive features. In *Paper presented at the Industrial and Engineering Applications of Artificial Intel ligence and Expert Systems*, 1993.

[28] Peter Turney and Michael Halasz. Contextual normalization applied to aircraft gas turbine engine diagnosis. *Journal of Applied Intelligence*, 3:109–129, 1993.

[29] G. Widmer and M. Kubat. Effective learning in dynamic environments by explicit concept tracking. In Pavel B. Brazdil, editor, *European Conference on Machine Learning*, pages 227–243, Berlin, 1993. Springer-Verlag.

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

[31] Gerhard Widmer. Recognition and exploitation of contextual clues via incremental meta-learning. In Lorenza Saitta, editor, *Machine Learning: Proceedings of the Thirteenth International Workshop*, pages 525–533, San Francisco, 1996. Morgan Kaufmann.