Machine Learning, 23, 69–101 (1996)

Learning in the Presence of Concept Drift and Hidden Contexts

GERHARD WIDMER gerhard@ai.univie.ac.at
Department of Medical Cybernetics and AI, University of Vienna, and Austrian Research Institute for Artificial Intelligence, Schottengasse 3, A-1010 Vienna, Austria

MIROSLAV KUBAT mkubat@csi.uottawa.ca
Department of Computer Science, University of Ottawa, 150 Louis Pasteur, Ottawa, Ontario K1N 6N5, Canada

Editor: Douglas Fisher

Abstract. On-line learning in domains where the target concept depends on some hidden context poses serious problems. A changing context can induce changes in the target concepts, producing what is known as concept drift. We describe a family of learning algorithms that flexibly react to concept drift and can take advantage of situations where contexts reappear. The general approach underlying all these algorithms consists of (1) keeping only a window of currently trusted examples and hypotheses; (2) storing concept descriptions and re-using them when a previous context reappears; and (3) controlling both of these functions by a heuristic that constantly monitors the system's behavior. The paper reports on experiments that test the systems' performance under various conditions such as different levels of noise and different extent and rate of concept drift.

Keywords: Incremental concept learning, on-line learning, context dependence, concept drift, forgetting

1. Introduction

The work presented here relates to *incremental* or *on-line* concept learning, which has recently received considerable attention among theoreticians (e.g., Angluin, 1988; Maass, 1991; Helmbold, Littlestone, & Long, 1992) as well as practitioners (e.g., Schlimmer & Granger, 1986; Langley, Gennari, & Iba, 1987; Kilander & Jansson, 1993; Kubat, 1989, 1993; Salganicoff, 1993a; Widmer & Kubat, 1993). The principal task is to learn a concept incrementally by processing labeled training examples one at a time. From another point of view, the problem may be seen as minimizing the total number of erroneous classifications in a feedback system: a stream of objects are classified, one by one, as positive or negative instances of a concept, and immediately afterwards the correct answer is received. The learner uses the current state of the knowledge base to predict the class of each incoming example. A discrepancy between the prediction and the real class value will usually trigger modifications to the knowledge base.

A difficult problem in such a learning scenario is that the concepts of interest may depend on some *hidden context*. Mild weather means different things in Siberia and in Central Africa; Beatles fans had a different idea of a fashionable haircut than the

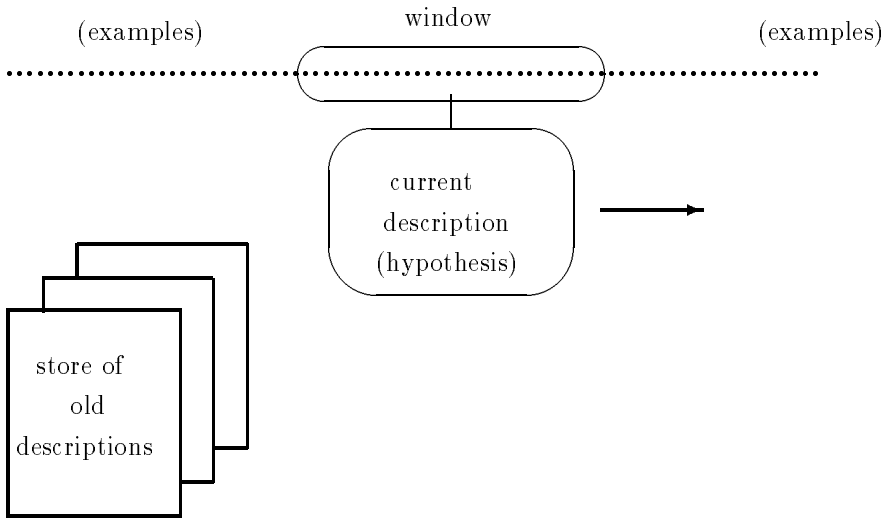

Figure 1. Current and old concept descriptions and the window moving along the stream of examples.

Depeche-Mode generation. Or consider weather prediction rules, which may vary radically depending on the season. Changes in the hidden context can induce more or less radical changes in the target concepts, producing what is generally known as *concept drift* in the literature (e.g., Schlimmer & Granger, 1986).

An example of real-world concept drift is described by Kubat (1992), where a system was presented that learned to control the load redistribution in computer clusters: overloaded units should send part of their load to underloaded units in order to improve the overall response time. The 'real' rules describing "overloadedness" would depend on the workload profile as defined by the frequency of disk operations, CPU and memory requirements, and the like. However, the only variables visible to the system were the lengths of various CPU and disk queues. Thus, the workload structure was the hidden context that determined the interpretation of the visible variables. Note that this context varies in time and that similar contexts can reappear.

Effective learning in environments with hidden contexts and concept drift requires a learning algorithm that can detect context changes without being explicitly informed about them, can quickly recover from a context change and adjust its hypotheses to a new context, and can make use of previous experience in situations where old contexts and corresponding concepts reappear.

One possible approach is sketched in Figure 1. As the context is known to vary in time, the learner trusts only the latest examples — this set is referred to as the *window*. Examples are added to the window as they arrive, and the oldest examples are deleted from it. Both of these actions (addition and deletion) trigger modifications to the current concept hypothesis to keep it consistent with the examples in

the window. In the simplest case, the window will be of fixed size, and the oldest example will be dropped whenever a new one comes in.

To extend the basic model, assume that the learner maintains a store of concept descriptions or hypotheses pertaining to previously encountered contexts. This is indicated by the boxes in the lower left part of Figure 1. When the learner suspects a context change, it will examine the potential of the previously stored descriptions to provide better classifications. Based on the result, the system may either replace the current concept description with the best of the stored descriptions, or start developing an entirely new description.

Generally, a learning algorithm embodying these ideas needs (1) operators that modify the concept description in reaction to changes in the contents of the window; (2) the ability to decide when and how many old examples should be deleted from the window ('forgotten'); and (3) a strategy that maintains the store of current and old descriptions and assesses the relative merits of concept hypotheses. Clearly, items (2) and (3) will require the system to make some guesses as to when a context change is occurring.

The basic approach to learning and forgetting will be elaborated in the following section, where we specify a simple algorithm for maintaining a consistent concept hypothesis. Section 3 looks at computational learning theory for some hints concerning the main parameter of this algorithm (the window size). The following sections then describe three extensions of the basic method and their realization in experimental systems: the algorithm *FLORA2* (Section 4) possesses the ability to dynamically adjust the window to an appropriate size during learning; *FLORA3* (Section 5) stores concepts for later use and reassesses their utility when a context change is perceived; and *FLORA4* (Section 6) is designed to be particularly robust with respect to noise in the input data. Experiments with all three algorithms under varying conditions are presented in Section 7. Finally, Section 8 relates the *FLORA* approach to other research in machine learning.

2. Learning and Forgetting: The General *FLORA* Framework

Currently, our algorithms are restricted to the relatively simple representation language based on attribute-value logic without negation. Throughout the paper we will often use the notion of a *description item*, which is a conjunction of attribute-value pairs. We will say that a description item *matches* an example if it is true for it. For instance, `(color = white)` $\wedge$ `(temperature = low)` matches 'snow' and `(shape = cube)` does not match the Globe. Formally, a description item matches the description of an object if all its literals (attribute-value pairs) occur in the object's description.

In the *FLORA* framework, a *concept description* or *hypothesis* is represented in the form of three description sets: the set *ADES* (*A*ccepted *DES*criptors) contains description items matching only positive examples. Like the other two sets, *ADES* can be interpreted as a disjunctive normal form (DNF) formula. The set *NDES* (*N*egative *DES*criptors) similarly summarizes the negative examples; and *PDES*

(*P*otential *DES*criptors) contains description items that are too general, matching positive examples, but also some negative ones.[1] The set *ADES*, representing the current (positive) concept hypothesis, is used to classify new incoming examples. *NDES* summarizes the negative examples and is used to prevent over-generalization of *ADES* (see below), while *PDES* acts as a reservoir of hypotheses that are currently too general but might become relevant in the future.

Assume the following structure:

$$\begin{aligned} ADES &= \{ADes_1/AP_1, ADes_2/AP_2, \ldots\} \\ PDES &= \{PDes_1/PP_1/PN_1, \ldots\} \\ NDES &= \{NDes_1/NN_1, \ldots\} \end{aligned}$$

where $ADes_i$, $PDes_i$, and $NDes_i$ are conjunctive description items; AP_i and PP_i are counters that specify how many positive examples in the current window match the individual description items in *ADES* and *PDES* respectively; similarly, PN_i and NN_i count how many negative examples match the respective description items. The counters are updated with any addition to or deletion from the window and are used to decide when to move an item from one set to another or when to drop it altogether from the hypotheses. In any case, items are retained only if they cover at least one (positive or negative) example in the current window.

The description items in *ADES* and *NDES* are generated by incremental generalization in response to positive and negative instances. When new instances are added to or deleted from the window, some items will be moved from one set to another. In particular, the set *PDES* of 'potential hypotheses' contains items that were once in *ADES* or *NDES*, but are contradicted by some examples. They are kept in *PDES* in the hope that they may become relevant again when old instances are dropped from the window. More precisely, modifications to the window can affect the contents of the description sets in the following ways:

- Adding a *positive* example to the window may cause a new description item to be included in *ADES*, or some existing items to be either 'confirmed' or generalized to accommodate the new instance, and/or existing items to be transferred from *NDES* to *PDES*.
- Similarly, adding a *negative* example to the window may cause a new description item to be included in *NDES*, or some existing items to be 'reinforced' or generalized, or existing items to be transferred from *ADES* to *PDES*.
- *Forgetting* an example (dropping it from the window) can cause existing description items to be 'weakened' (i.e., the corresponding counters are decremented), or even deleted from the current description set (if the counter drops to zero), or moved from *PDES* to *ADES* (if the example was the only negative instance covered) or to *NDES* (if the example was the only positive one).

Figure 2 helps visualize what is going on during the learning process. The arrows indicate possible migrations of hypotheses between description sets after the arrival

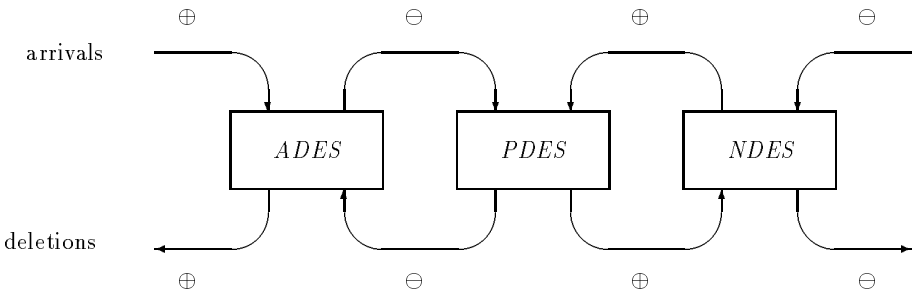

Figure 2. Transitions among the description sets

or deletion of a positive ($\oplus$) or negative ($\ominus$) instance, respectively. The extent of these transitions for the case of n arrivals and m deletions is quantified by Kubat (1991).

The complete basic *FLORA* algorithm for maintaining the hypotheses when a positive example is processed is sketched in Table 1 (if the example is negative, the algorithms work analogously — just substitute *NDES* for *ADES*). Note that there are two procedures, one for the case when a new example is added and one for the case when the oldest example is deleted from the window.

The actions taken when removing an example from the window are rather straightforward. In the case when a new (positive) example arrives, the respective counters in the set of potential descriptors, *PDES*, are incremented and description items in *NDES* matching the instance are moved to *PDES*. For *ADES* (accepted descriptors), there are three possible cases: if the example matches some items, the counters are simply incremented; otherwise, if some item can be generalized to accommodate the instance without becoming too general (i.e., subsuming some item in *PDES* or *NDES*), generalization is performed. Otherwise, the description of the instance is added as a new description to *ADES*. The only generalization operator used is the *dropping condition rule* (Michalski, 1983), which drops attribute-value pairs from a conjunctive description item.

Generally, the three description sets are kept non-redundant and consistent by checking for subsumption within and between the sets. In this way, for instance, over-generalization of *ADES* is avoided by checking it against *PDES* and *NDES* whenever one of the description items is generalized.

Note also that there is no *specialization operator* in this framework: if a new positive (negative) instance cannot be incorporated consistently into any of the generalizations, its full description is added to *ADES* (*NDES*); the instance acts as a specific 'seed' which may be generalized later. Overly general descriptions are discarded when old examples are forgotten.

Function *learn-from(I)* for a positive instance I

functions:

- *match(I, $XDes_i$)* ... determines whether the $XDes_i$ is true for instance I
- *delete(X,XDES)* ... deletes X from *XDES*
- *include(X, XDES)* ... puts X into *XDES* if not subsumed by an existing item
- *generalize(I,XDES,YDES,ZDES)* ... performs minimal generalization of some item X in *XDES* to cover instance I (if possible without subsuming some item in the other sets *YDES, ZDES*); tries to select X that requires least amount of generalization; returns true upon success;

algorithm:

```
MATCH := false;
for i := 1 to |ADES |
    if match(I, ADes_i) then
            begin   AP_i := AP_i + 1;
                    MATCH := true
            end;
if not MATCH then G := generalize(I,ADES,PDES,NDES);
if not MATCH and not G then include(I/1,ADES);
for i := 1 to |PDES |
    if match(I, PDes_i) then PP_i := PP_i + 1;
for i := 1 to |NDES |
    if match(I, NDes_i) then
            begin   delete(NDes_i, NDES);
                    include(NDes_i/1/NN_i,PDES)
            end;
```

Function *forget(I)* for a positive instance I

algorithm:

```
for i := 1 to |ADES |
    begin   if match(I,ADes_i) then AP_i := AP_i - 1;
            if AP_i = 0 then delete(ADes_i, ADES);
    end;
for i := 1 to |PDES |
    begin   if match(I,PDes_i) then PP_i := PP_i - 1;
            if PP_i = 0 then
                    begin       delete(PDes_i,PDES);
                                include(PDes_i/PN_i,NDES)
                    end
    end;
```

Table 1. The basic *FLORA* algorithm: Functions *learn_from(X)* and *forget(X)*

The general approach presented here assumes that only the latest examples are relevant and should be kept in the window, and that only description items consistent with the examples in the window are retained. While each new example is automatically included in the window, the question of how many examples should be deleted is more difficult. In the following section, we briefly review some theoretical results from computational learning theory as they relate to this question. Then we present a heuristic solution to the problem.

3. Theoretical results concerning learning under drift

The notion of concept drift has received some attention in the literature on computational learning theory in recent years. For instance, Helmbold and Long (1991, 1994) and Kuh, Petsche, and Rivest (1991, 1992) have investigated various conditions under which effective drift tracking is possible. They start from the observation that drift tracking is strictly impossible if there are no restrictions on the type of concept changes allowed. (As an extreme example, consider a sequence of concepts that randomly alternates between the constant function 1 and the constant function 0 after every example). They then go on to study various restrictions on the severity (*extent*) or the frequency (*rate*) of concept changes.

In particular, Helmbold and Long (1994) assume a permanent (possibly with each example) but very slight drift. In the following statement of results, the c_i's are positive constants, ϵ is the maximum allowed probability of misclassifying the next incoming example, n is the number of available attributes, and d is the Vapnik-Chervonenkis dimension (see, e.g., Blumer et al., 1989) of the target class. The *extent of concept drift* Δ is measured in terms of the relative error of two successive concepts, i.e., the probability that they will disagree on a randomly drawn example. Their main results are:

- an algorithm that tolerates drift of extent up to $\Delta \leq c_1\epsilon^2/(d \ln(1/\epsilon))$;
- a randomized version of this algorithm that is potentially more efficient computationally but tolerates drift of lower extent ($\Delta \leq c_2\epsilon^2/(d^2 \ln(1/\epsilon))$); and
- upper bounds on the tolerable amount of drift for two particular concept classes (halfspaces and axis-aligned rectangles) — no algorithm can track concept drift greater than $c_3\epsilon^2/n$ if the prediction error is to stay below ϵ.

Helmbold and Long also show that it is sufficient for a learner to consider only a fixed number of previous examples (i.e., a fixed *window* sliding over the input stream). Their analysis leads to rough estimates as to the window size needed for effective tracking; in the case of the first of the above algorithms, for instance, they show that a window size $m = (c_0 d/\epsilon)\log(1/\epsilon)$ (together with the above restriction on the allowable amount of drift) is sufficient to guarantee trackability.

"Computationally efficient", in their framework, means that updating the hypothesis and classifying the next incoming example should be feasible in polynomial time. In effect, their algorithm recomputes the current hypothesis from the

entire window after every new instance ("batch learning"). This is a reasonable assumption in complexity theory, but it may not be what we desire for a practical application. What we are looking for is a truly *incremental* algorithm that only looks at the new example to modify its current hypothesis.

Kuh et al. (1991) introduce the notion of *PAC-tracking* as a straightforward extension of Valiant's (1984) PAC framework. Again, their general results relate to the batch-learning scenario, where a hypothesis is recomputed from the entire window after each instance. Their approach is somewhat orthogonal to the work of Helmbold and Long: rather than restricting the *extent* of drift, they set out to determine the maximum *rate* of drift (i.e., how frequently a concept is allowed to change) that is tolerable by a learner.

Their main general result is a lower bound on the allowed drift rate: $\lambda < \frac{\delta}{2} m(\epsilon, \delta/2)$ (where a drift rate λ means that on average, a concept is stable for at least $1/\lambda$ time steps or instances). Again, this result assumes a minimum (fixed) window size of $w(\epsilon, \delta) = m(\epsilon, \delta/2)$ where $m(\epsilon, \delta)$ is derived from the general bound on the number of training examples that guarantee PAC-learning (Blumer et al., 1989): $m(\epsilon, \delta) = max(\frac{4}{\epsilon} \log \frac{2}{\delta}, \frac{8d}{\epsilon} \log \frac{13}{\epsilon})$. The similarity to the sample size derived by Helmbold and Long is evident.

Attempts to characterize fully incremental learning under concept drift (Kuh et al., 1992) have led to bounds on the expected mistake rate of drift trackers for some very specific concept classes (e.g., half-planes and intersections thereof). No general results for arbitrary concept classes are known to us.

For many practical applications with real-world data, truly incremental learning is important, and so are reasonably sized windows. On the other hand, we cannot always demand or expect an arbitrarily small error. With respect to the large window size prescribed by theoretical analysis, Kuh et al. (1991) conclude that "[a]n algorithm that removes inconsistent examples more intelligently, e.g., by using conflicts between examples or information about allowable changes, will be able to track concept sequence spaces that change more rapidly." That is exactly what we attempt to do in the *FLORA* family of algorithms. In the tradition of "AI-type" learning, we will take a *heuristic* approach to dynamically adjusting the window size based on strategies for explicitly detecting context changes. We will assume that the *rate* of context changes is rather low (i.e., that there are phases of stability between periods of change), but on the other hand we will allow concept changes of arbitrarily large *extent* (successive versions of the target concept may be very different). Of course, under these circumstances one cannot expect the prediction error ϵ always to be bounded; but our heuristic approach aims at enabling the learner to recover very quickly from low predictive accuracy after a context change.

4. Flexible Windowing: *FLORA2*

The first realization of the *FLORA* framework that we discuss is the algorithm *FLORA2* (Widmer & Kubat, 1992), which attempts to dynamically adapt the size of its window during the learning process.

4.1. Description of *FLORA2*

Let us start with an intuitive look at the effects of an inappropriate window: basically, a narrow window will not accommodate a sufficient number of examples for a stable concept description; a wide window, on the other hand, will slow down the learner's reaction to concept drift, particularly if the change in the concept is quite dramatic. Obviously, the ideal setting depends on the extent of the concept drift and on the momentary state of learning. Both of these variables can only be determined dynamically, during learning. Moreover, the occurrence of a concept change can only be guessed at. A good heuristic for dynamic window adjustment should shrink the window (and forget old instances) when a concept drift seems to occur, and keep the window size fixed when the concept seems stable. Otherwise the window should gradually grow until a stable concept description can be formed.

In trying to guess when a concept change occurs, *FLORA2* uses two heuristic indicators: the system's predictive performance *Acc* (monitored over a fixed number of past classifications) and syntactic properties of the evolving hypotheses. The basic assumption is that serious drops in *Acc* or an explosion of the number of description items in *ADES* may signal a possible concept drift. As both of these indicators depend heavily on characteristics of the learning task, three parameters are used to customize the heuristic to the application domain:

lc (= threshold for low coverage of $ADES$);
hc (= threshold for high coverage of $ADES$); and
p (= threshold for acceptable predictive accuracy).

By the coverage of a description set we mean the ratio of the number of instances covered by items in the set and the size (in terms of the total number of literals over all descriptions) of the set. This definition of coverage trades off the number of examples covered and the "cost" (syntactic complexity) of description items.

Table 2 shows the Window Adjustment Heuristic (*WAH*) that is used in *FLORA2*. The *WAH* decreases the window size by 20% if a concept drift is suspected. In contrast, the window size is decreased by 1 (after the addition of a new example, the two oldest examples are deleted) if the hypothesis seems to be extremely stable; this is to avoid retention of unnecessarily large numbers of examples. If the current hypothesis seems sufficiently stable, the window size is simply left unchanged. If none of these conditions is satisfied, the program assumes that more information is needed and it does not forget the oldest example, thus incrementally increasing the window size by 1.

The particular parameter settings currently used in *FLORA2* are $lc = 1.2$, $hc = 4.0$ and $p = 70\%$. The parameters were set after some preliminary experiments with the '*STAGGER* concepts' (see below) and were left unchanged in all the other experiments to be reported.[2]

Given that the heuristic is (necessarily) very syntactically oriented and is thus very sensitive to the description language used, it seems hopeless (or at least difficult) to make it completely free of parameters.

denotations:

N ... number of positive instances covered by $ADES$
S ... size of $ADES$ in terms of number of literals
Acc ... current predictive accuracy (monitored over recent classification attempts)
$\mid W \mid$ ... window size
parameters lc, hc and p are user-defined

algorithm:

if $(N/S < lc)$ or $((Acc < p)$ and $decreasing(Acc)$	/* *drift suspected* */
then $L := 0.2 * \mid W \mid$	/* *reduce window by 20%* */
else if $(N/S > 2 * hc)$ and $(Acc > p)$	/* *extremely stable* */
then $L := 2$	/* *reduce window by 1* */
else if $(N/S > hc)$ and $(Acc > p)$	/* *stable enough* */
then $L := 1$	/* *keep window fixed* */
else $L := 0$.	/* *grow window by 1* */

Table 2. Window Adjustment Heuristic (WAH): $how\_many\_to\_forget(lc, hc, p)$

4.2. A simple experiment: The *STAGGER* concepts

For a quick comparison with one of the first concept drift trackers, *STAGGER* (Schlimmer & Granger, 1986), *FLORA2* was tested on the same artificial learning problem as used by Schlimmer and Granger. The instance space of a simple blocks world is defined by the three attributes $size \in \{small, medium, large\}$, $color \in \{red, green, blue\}$, and $shape \in \{square, circular, triangular\}$. There is a sequence of three target concepts (1) $size = small \wedge color = red$, (2) $color = green \vee shape = circular$ and (3) $size = (medium \vee large)$. 120 training instances are generated randomly, labeled according to the hidden concept, and after processing each instance, the predictive accuracy is tested on an independent test set of 100 instances, also generated randomly. The underlying concept is made to change after every 40 training examples.[3] The results are averaged over 10 runs. Figure 3 shows *FLORA2*'s predictive accuracy on the test set after processing each training example. The dotted vertical lines indicate where the underlying concept changes.

It can be seen that the dramatic concept shifts lead to a sharp decrease of the predictive accuracy, but *FLORA2* adjusts very quickly and soon approaches the 100% mark again. Figure 4 indicates that this is due to the workings of the *Window Adjustment Heuristic*. Figure 4 plots the development of the window size in a typical (single) run. The *WAH* behaves as expected: a change in the definition of the underlying concept first leads to a short *increase* in the size of the window, before the system reacts to the concept shift by narrowing the window and forgetting old, now irrelevant or contradictory instances.

A comparison of these curves with the respective figures in (Schlimmer & Granger, 1986) suggests that *FLORA2* is comparable to *STAGGER* in terms of convergence and re-adjustment speed on this basic task. In a second experiment, Schlimmer

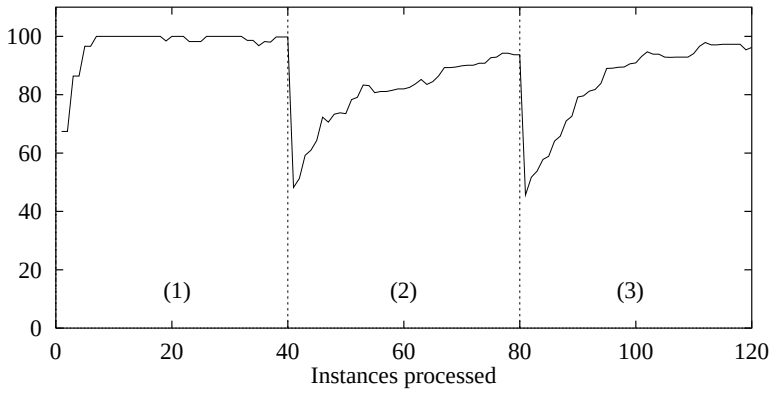

Figure 3. Adjusting to drift: predictive accuracy.

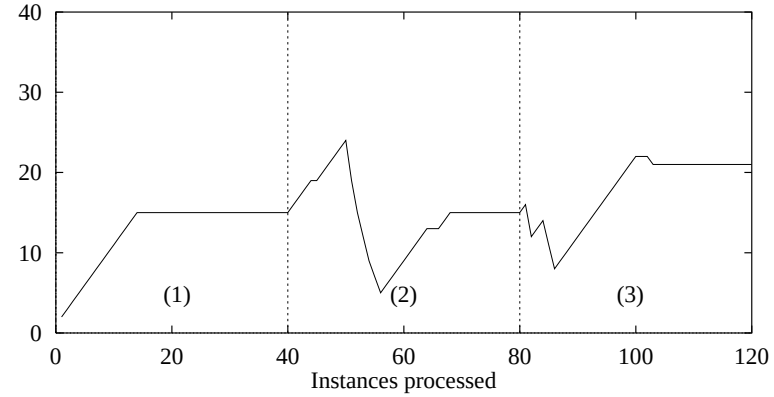

Figure 4. Adjusting to drift: dynamic window size.

and Granger showed that *STAGGER* is sensitive to over-training. The longer it has been trained on a stable concept, the slower it will be in 'letting go' and changing its hypothesis when the underlying context changes. When the same experiment (making each of the three concept periods 150 instances long) is performed with *FLORA2*, there is no such effect, which is, of course, due to the fact that *FLORA2* stops the window growth once its hypotheses are stable. We will return to this point in Section 8.

In any case, *FLORA2*'s explicit drift detection mechanism (the *WAH*) has additional advantages, as will be seen in the following section, where we introduce an algorithm for explicit context handling.

5. Dealing with Recurring Contexts: *FLORA3*

There are many natural domains where there is a finite number of hidden contexts that may reappear, either cyclically or in an unordered fashion. For instance, there are four seasons that follow each other in a cyclic order and cause regular changes in many natural phenomena. Biological and economic systems tend to go through cycles of development with recurring patterns.

In domains where contexts and associated concept versions reappear, it would be a waste of effort to relearn an old concept from scratch for each recurrence. Instead, concepts or hypotheses should be saved so that they can be reexamined at some later time, when there are indications of a context change. The effect should be faster convergence if the concept (or a similar one) has already occurred. This section introduces an extension of *FLORA2* that includes a mechanism for context storage and recall. The mechanism is tightly coupled with the window adjustment algorithm.

5.1. Description of *FLORA3*

The top level of *FLORA3* is similar to *FLORA2*. The system first tries to classify the new incoming example, updates its on-line classification accuracy, then learns from the instance by incorporating it into the window and updating the description sets, and, after calling the *WAH* to decide whether and how to adjust the window size, 'forgets' the appropriate number of old instances. However, after each learning cycle, *FLORA3* inspects the current state of learning in order to decide whether it should reconsider concept descriptions that were useful in some old context.

The idea is that when a context change seems to occur, the system should consult its store of old concept descriptions to see whether some old concept might better describe the examples currently in the window. Conversely, when a stable concept hypothesis has been reached, it might be worthwhile to store the current hypothesis for later reuse. It is the *Window Adjustment Heuristic* (*WAH*), as embodied in the function *how_many_to_forget* in Table 2, that tries to determine the relevant conditions (the occurrence of a context change and the stability of the learning situation). So in *FLORA3*, storage and reexamination of old hypotheses are tightly linked to changes in the window size.

The corresponding function *choose_context*, which is called at the end of each learning loop, is sketched in Table 3. When the current hypothesis is *stable* according to the *WAH*, the system saves it for later reuse, unless there is already a stored concept with the same set of *ADES* descriptions. On the other hand, if there is reason to believe that a context change is taking place (i.e., when the *WAH* enforces a *narrowing of the window*), the system examines its store of old concept descriptions in an attempt to find one that fits the current situation. If one is found that seems more appropriate than the current hypothesis, it is reinstalled as the new hypothesis.

denotations:
    Stable ... boolean variable; *true* if the current hypothesis is stable
        according to the *WAH*;
    Drift_suspected ... boolean variable; *true* if the *WAH* suspects a
        concept drift and has narrowed the window;
functions:
    . *store_current* ... store current description sets
    . *find_best_candidate* ... find best matching old context
    . *regeneralize_old_description* ... regeneralize according to current window
    . *replace_if_applicable* ... reinstall old hypothesis, if better than current

algorithm:
if *Stable*
    then *store_current*
else if *Drift_suspected* then
    begin *Best* := *find_best_candidate*;
        G := *regeneralize_old_description(Best)*;
        replace_if_applicable(G)
    end.

Table 3. Function *choose_context*

Note that when a concept description pertaining to an old context is retrieved, it will usually not agree 100% with the examples in the current window. Therefore, all examples in the current window must be regeneralized. The counters associated with the items of the retrieved hypothesis are set to zero, and then the regular *FLORA* learning algorithm (Table 1) is invoked for each example in the window. All description items that have counters equal to zero after re-generalization are removed as irrelevant. The algorithm for reassessing old concepts proceeds in three steps (see Table 3):

1. *Find the best candidate* among the stored concepts: an old hypothesis becomes a *candidate* if it is consistent with the current example. All the candidates are evaluated with respect to the ratio of the numbers of positive and negative instances they match (from the current window);

2. *Update the best candidate* w.r.t. the current data by setting all the counters in the description sets to 0 and then reprocessing all the examples in the window;

3. *Compare the updated best candidate C_b to the current concept description C*: use some 'measure of fit' to decide whether C_b is better than C; if so, replace C with C_b. In the current version of *FLORA3*, the measure of fit is simply the relative *complexity* of the description: a concept description is considered better if its *ADES* set is more concise. (Remember that by construction, the *ADES* sets of both C and C_b cover all the positive and no negative instances from the window).

The algorithm tries to maintain efficiency by limiting the number of expensive reprocessing episodes. Old concepts are not reconsidered after every new training

instance; they are only retrieved when the window adjustment heuristic suspects a concept drift. In addition, the expensive part of reconsidering an old concept — the regeneralization of all the instances in the window — is done only for one of them – the best candidate. The best candidate is determined through a simple heuristic measure, which, of course, can sometimes lead to an inappropriate candidate being chosen. Thus, efficiency is achieved at the possible expense of quality.

It seems worth emphasizing that the role of the retrieved old concept/hypothesis is to act as a *model* or *bias* for regeneralizing the current examples. The retrieved candidate provides a list of generalizations that were useful in the past and that might, at least in part, also be useful in the new context. That reflects the insight that when an old context returns, the target concepts will tend to be similar, but not necessarily identical to how they appeared in the old context.

5.2. An Experiment with recurring contexts

To measure the effectiveness of the context tracking (store/recall) mechanism, we created a situation of recurring contexts by repeating the three *STAGGER* concepts three times, in the cyclic order 1-2-3-1-2-3-1-2-3. Training and test instances were generated according to the same procedure as above. Again, results are averages over 10 runs.

Figure 5 compares *FLORA3* (solid line) to *FLORA2* (dashed line) on this task. Storing and reusing old concepts leads to a noticeable improvement over the simpler system when contexts actually reappear: starting from the fourth period (the first reoccurrence of context 1) the solid line shows faster readjustment to higher accuracy levels in four out of six cases (the differences in the last two periods are too small to be significant). An interesting phenomenon appears in the third period of the plot — the first occurrence of context 3. Here, *FLORA2* did better than *FLORA3*. That may seem odd at first sight, as there is no context recurrence at this point, so ideally, both systems should behave the same. But the concept retrieval and adaptation algorithm is driven by heuristics and can sometimes lead the system to reinstall an old concept erroneously.[4] The context tracking mechanism thus adds another degree of freedom (and source of potential errors) to the learning process. However, when old contexts actually do reappear, the advantages of the context tracking approach begin to outweigh the disadvantages, as can be seen from the following phases in the experiment.

Similar results were achieved in experiments with a more complex world (see Widmer & Kubat, 1993). However, there it also turned out that very slow concept drift and especially noise in the training data destabilized *FLORA3*'s (and equally *FLORA2*'s) performance more than seemed strictly necessary. That observation led to the development of *FLORA4*.

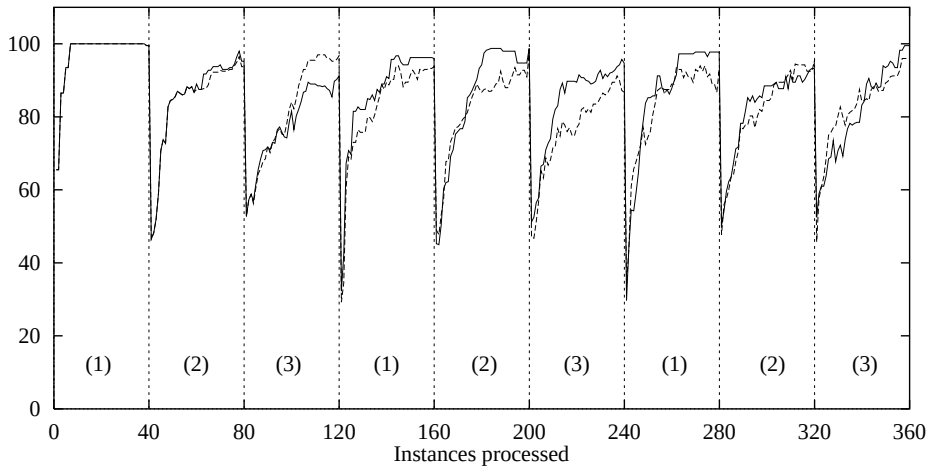

Figure 5. Learning in domain with recurring contexts: FLORA3 (solid) vs. FLORA2 (dashed).

6. Drift vs. Noise: *FLORA4*

Generally, it is very difficult in incremental learning to distinguish between 'real' concept drift and slight irregularities that are due to *noise* in the training data. Methods designed to react quickly to the first signs of concept drift may be misled into overreacting to noise. This results in unstable behavior and low predictive accuracy. On the other hand, an incremental learner that is designed primarily to be highly robust in the face of noise runs the risk of not recognizing real changes in the target concepts and may adjust to changing conditions very slowly, or only when the concepts change radically. An ideal learner should combine stability and robustness with flexible and effective context-tracking capabilities. On the face of it, the two requirements seem diametrically opposed. Nonetheless, we can at least try to achieve a compromise between them.

A simple analysis of *FLORA2* and *FLORA3* reveals that their brittleness in the face of noise is a result of the strict *consistency condition* that is used to decide which generalizations to keep in *ADES*. As hypotheses in *ADES* (and *NDES*) must be strictly consistent with the examples (e.g., an expression in *ADES* must not cover any negative instances), one negative example is sufficient to invalidate a description item and cause it to be moved from *ADES* to *PDES*, even if it covers a large number of positive examples. That can lead to somewhat unstable behavior even in noise-free domains, especially when a concept change is taking place, but it is particularly problematic when the input data are noisy, i.e., when some of the training examples may be mislabeled.

6.1. Description of *FLORA4*

To counter this problem, *FLORA4* drops the strict consistency condition and replaces it with a 'softer' notion of reliability or predictive power of generalizations. The idea is to continuously monitor the predictive accuracy of each generalization in the description sets and to statistically evaluate the confidence of these accuracy estimates: *FLORA4*, like its predecessors, uses its current hypothesis (*ADES*) to classify each incoming example before learning from it. Now the central idea is to keep a classification record for each individual description item (conjunction) and to construct statistical *confidence intervals* around these measures. Decisions as to when to move an item from one set to another or when to drop it altogether are based on the relation between these confidence intervals and the observed class frequencies: a hypothesis is kept in *ADES* as long as its predictive accuracy is higher (with high confidence) than the observed frequency of the class it predicts.

More precisely, let μ = required confidence level (parameter); assume that each description item is associated with two numbers, α_l and α_u, that represent the lower and upper endpoints, respectively, of the statistical confidence interval (with confidence μ) around the item's classification accuracy, computed over the instances in the current window. Let γ_l and γ_u be the lower and upper endpoints, respectively, of the confidence interval (with confidence μ) around the relative frequency of the positive class observed so far (i.e., the percentage of processed training instances that are positive examples of the target concept).[5]

FLORA4 then uses the following criteria to maintain its description sets (compare this to Table 1):

- a description item X is kept in *ADES* if the lower endpoint of its accuracy confidence interval is greater than the class frequency interval's upper endpoint ($\alpha_l > \gamma_u$); similarly, any X in *PDES* that satisfies this condition is moved to *ADES* — we say that X is (temporarily) *accepted* as a predictor;
- a description item X in *ADES* whose accuracy interval overlaps with the class frequency interval ($\alpha_u > \gamma_l$) is moved to *PDES* — X is a *mediocre* predictor (it does not do significantly better than guessing); expressions in *PDES* are not used for classification;
- a description item X is dropped completely if the upper endpoint of its accuracy interval is lower than the class frequency interval's lower endpoint ($\alpha_u < \gamma_l$) — X is *rejected*;
- description items in *NDES* are kept as long as they are acceptable predictors of negative instances ($\alpha_l > \gamma_u$, computed over the negative examples in the window). In contrast to *FLORA2* and *FLORA3*, there is no migration of generalizations between *NDES* and *PDES*. Unacceptable hypotheses in *NDES* are simply dropped.

The general approach to deciding which hypotheses to trust has been adopted from the instance-based learning method *IB3* (Aha et al., 1991), which uses similar

measures to distinguish between reliable and unreliable predictors (*exemplars* in *IB3*). The terms *accepted*, *mediocre*, and *rejected* are used here to highlight this similarity. In all experiments with *FLORA4*, a confidence level of $\mu = 80\%$ was used.

The main effect of the strategy is that generalizations in *ADES* and *NDES* may be permitted to cover some negative or positive instances, respectively, and still to remain in *ADES* or *NDES* if their overall predictive accuracy warrants it. *PDES* is a reservoir of alternative generalizations that are recognized as unreliable at the moment, either because they cover too many negative examples, or because the absolute number of instances they cover is still too small (and thus the confidence intervals are large). The rest of the *FLORA3* strategy, including the generalization operator and the context reuse mechanism, remains unchanged. After every learning step the window adjustment heuristic is invoked and may decide to grow or shrink the window. Predictive accuracy of hypotheses is always computed with respect to the current window. In this way, *FLORA4* combines the advantages of the windowing approach with a less brittle strategy for maintaining generalizations.

6.2. Two preliminary experiments

The following two experiments briefly compare *FLORA4* to its predecessors and to its 'cousin' *IB3*, from which its statistical hypothesis evaluation strategy was adopted. They are again based on the *STAGGER* concepts. More thorough experiments are described in Section 7.

6.2.1. Basic drift tracking

Figure 6 compares *FLORA4* to *FLORA2* and *FLORA3* on the basic noise-free drift tracking task. The characteristic effect that can be seen in this plot (at least in the second period) and that appears even more clearly in the experiments in the next section, is that *FLORA4* is initially a bit slower in reacting to the change in the target concept, but then soon picks up and eventually regains high accuracy faster than both *FLORA2* and *FLORA3*.

The explanation is to be found in *FLORA4*'s statistical confidence measure. *FLORA4* reacts more reluctantly initially because several contradicting examples are necessary to invalidate a hitherto stable hypothesis in *ADES*, while *FLORA2* and *FLORA3* will drop a description item as soon as the first contradicting instance appears. The same observation also explains why *FLORA4* later reaches high accuracy faster: a consequence of *FLORA2*'s strict consistency condition is that one old negative instance (pertaining to the outdated context) erroneously still in the window may prevent a good generalization from being included in *ADES*. *FLORA4*, with its 'softer' consistency condition, is less disturbed by remnants of the old context still in the window and thus readjusts faster to the new context.

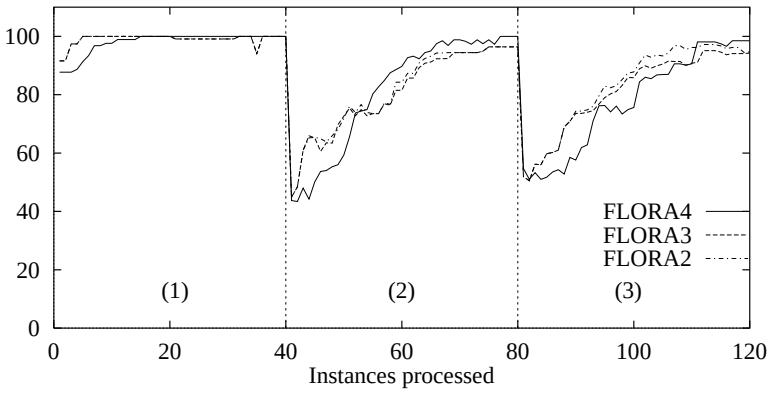

Figure 6. FLORA4 vs. *FLORA2* and *FLORA3* on *STAGGER* concepts.

6.2.2. FLORA4 *vs.* IB3

Figure 7 compares *FLORA4* to the publicly available version of *IB3* (Aha et al., 1991). *IB3* was modified so that it used different test sets according to the current context (which changed after every 40 instances). The improvement of *FLORA4* over *IB3* is evident.

Generally, our experience from various experiments with *IB3* is that *IB3* requires significantly more examples to converge to a high level of predictive accuracy, and that it is slower in recovering from changes in the target concept. The first effect is due to the general instance-based learning method. The latter difference is clearly attributable to the combination in *FLORA4* of *IB3*'s statistical confidence measures with a highly reactive window-based forgetting strategy, which permits the system to get rid of outdated information much faster. As a side note, one could also point out that a symbolic generalizer like *FLORA4* has certain advantages over an instance-based learner in terms of the comprehensibility of the results of learning.

7. Systematic Experiments

The following experiments study the behavior of the *FLORA* systems in some more detail, by systematically varying different aspects of the learning task. Again, we use artificial domains, as they make it easy to control the learning situation.

In particular, we study the following dimensions: (1) the level of *classification noise* in the training data — that is of particular interest to *FLORA4*, which we claimed is more robust in the face of noise than its predecessors; (2) the *speed* of (a gradual) drift, which is the time it takes for a new context to completely take over; this seems to us a more natural dimension for practical scenarios than the notion

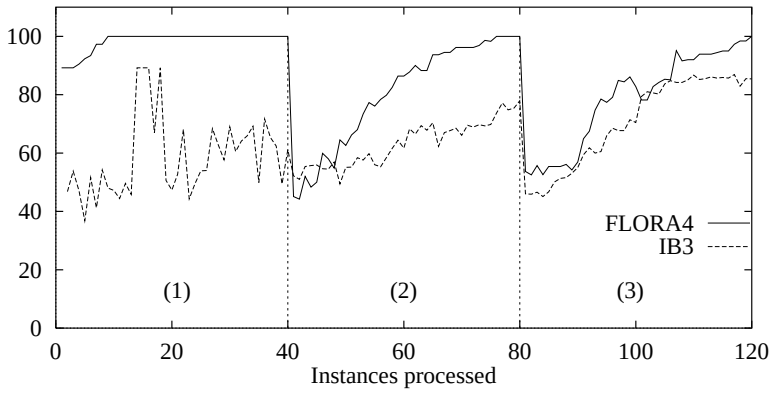

Figure 7. FLORA4 vs. *IB3.*

of drift *rate*, which is one of the two main parameters in theoretical investigations (see Section 3); (3) the *extent* of drift, i.e., the degree of dissimilarity of successive concepts, quantified in terms of the relative error between the concepts; and (4) the effect of *additional, irrelevant attributes* on the effectiveness of the learning process.

As none of the following experiments involves recurring contexts, *FLORA3*'s concept store/recall mechanism was disabled in *FLORA4*. That is, *FLORA4* resembles *FLORA2*, with the exception of the statistical hypothesis maintenance criterion. This was done to help separate the effects of concept reuse (as performed by *FLORA3*) and accuracy-based hypothesis maintenance. All the results in the following sections are averages over 10 runs.

7.1. Varying the amount of noise

The conjecture motivating the first experiment is that *FLORA4* should have significant advantages in noisy environments due to its combined strategy: the statistical confidence measures provide a certain robustness against noise, especially in relatively stable situations, and the window adjustment heuristic should recognize persistent misclassifications as indicators of a concept change and should lead to effective adjustment by shrinking the window in such situations. The targets in this experiment are again the three *STAGGER* concepts, but now the training data are corrupted with various levels of classification noise.

Figures 8 through 10 compare the performance of the three *FLORA* algorithms at noise levels in the training data of 10%, 20%, and 40%, respectively. (In this article, η% class noise means that with probability $\eta/100$, the class label of an instance will be assigned randomly. Thus, completely random data will be generated when $\eta = 100$.)

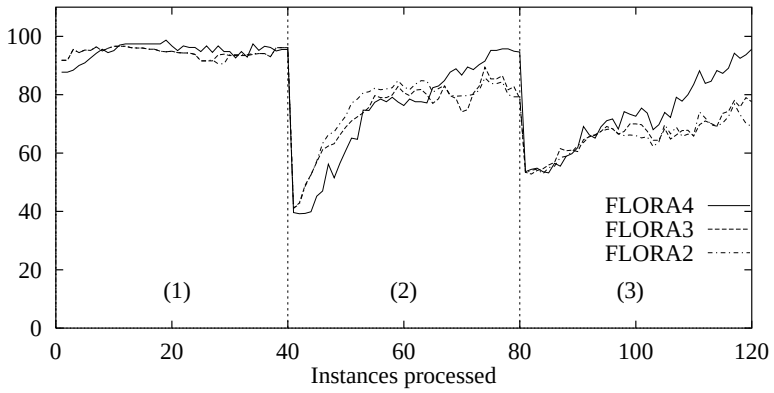

Figure 8. FLORA2, FLORA3, FLORA4 at 10% noise.

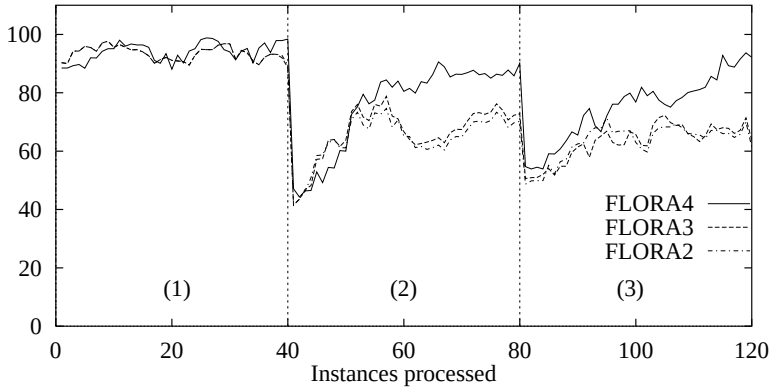

Figure 9. FLORA2, FLORA3, FLORA4 at 20% noise.

Again, we see that *FLORA4* is usually a bit slower in its initial reaction to the concept change, but then soon outperforms *FLORA2* and *FLORA3*. The difference is markedly greater than in the noise-free case. *FLORA2* and *FLORA3* have obvious problems, while *FLORA4*'s accuracy quickly rises to a mark that corresponds roughly to the given level of classification noise (remember that N% noise means $N/2$% misclassified instances on average in a two-class learning task).

A comparison of the average window sizes in the experiment with 40% noise (Figure 11) confirms our expectations: in the *FLORA4* curve, the characteristic shape (growing the window to a reasonable size during phases of concept stability, shrinking it in reaction to a perceived context change) is still clearly recognizable, whereas the behavior of the other two systems is less predictable. They apparently

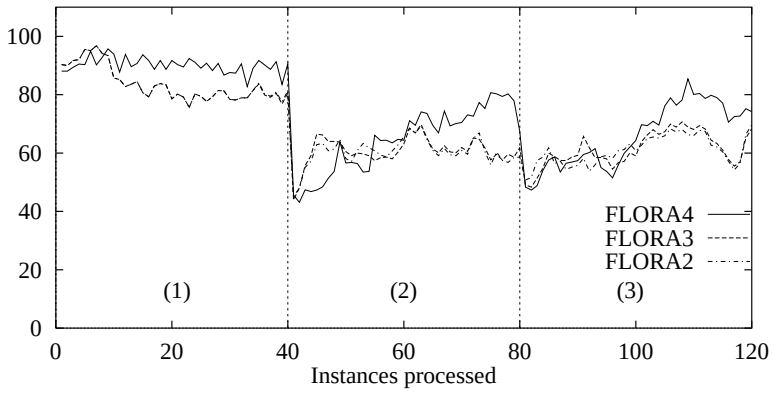

Figure 10. *FLORA2*, *FLORA3*, *FLORA4* at 40% noise.

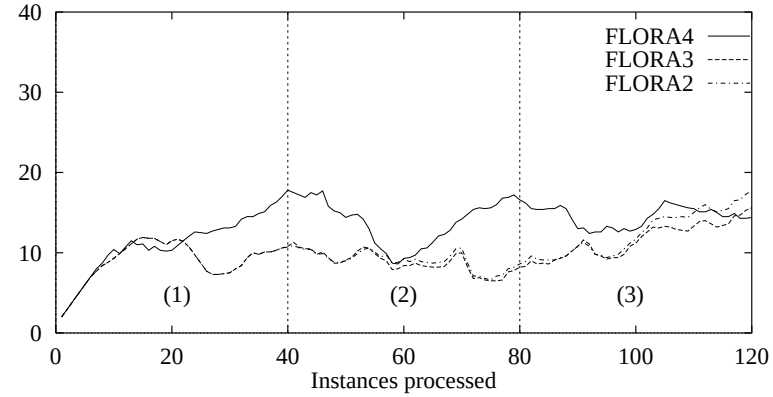

Figure 11. Average window sizes at 40% noise.

misinterpret noisy examples as indicators of concept drift, as evidenced by the constant growing and shrinking of the window, which in many cases prevents them from reaching a window size that is sufficient for stable concept identification. In *FLORA4* this kind of erratic behavior is largely prevented by the robustness of the generalizer's statistical criteria.

As a side note, we also notice that there is almost no difference between *FLORA2* and *FLORA3*. One might expect *FLORA3* to exhibit even less stable behavior than *FLORA2*, as erroneous reactions to perceived drift would lead it to constantly reexamine and sometimes reinstall previously stored hypotheses. However, the fact is that in this noisy environment, *FLORA3* hardly ever reaches a situation that it

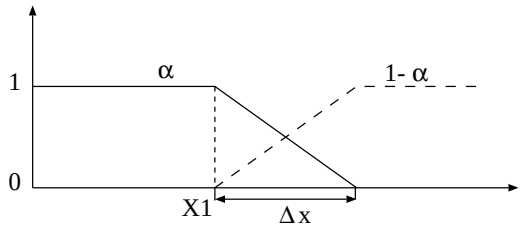

Figure 12. The function α.

considers stable enough to store its current hypothesis for possible future use, so there simply are no old concepts that could be reinstalled by mistake.

Schlimmer and Granger (1986) have noted that *STAGGER* distinguishes between random (examples of both classes affected) and systematic noise (only positive or only negative instances corrupted). In our experiments, we could not detect a similar tendency in *FLORA4*. That seems reasonable, as there are no components in our model comparable to *STAGGER*'s *LS* and *LN* measures, which are sensitive to one-sided variations. We conjecture that *STAGGER* may be more robust than *FLORA4* in situations with extremely high, but systematic noise.

7.2. Varying the speed of drift

The following experiment is concerned with what might be called the *speed* of concept drift. Sometimes concepts will change only gradually, creating a period of uncertainty between stable states. The new concept only gradually takes over, and some examples may still be classified according to the old concept. An example is the behavior of a device beginning to malfunction — it first fails (classifies in a new way) only sometimes, until the new failure mode becomes dominant.

Speed of drift can modelled by a function α (see Figure 12) that represents the degree of dominance of the old concept A over the new concept B or, in other words, the probability that the current concept still belongs to the old context. $\alpha = 1$ means that A is fully in effect, $\alpha = 0$ means that B has completely taken over. The x axis in Figure 12 represents the number of examples processed so far; assuming that instances arrive at constant intervals then this can also be regarded as a dimension of time. $X1$ is the point where the concept begins to drift. The slope of the function α can then be characterized by Δx, the number of training instances until α reaches zero. Between $X1$ and $X1+\Delta x$, $\alpha * 100\%$ of the examples are still classified according to A, and B labels $(1-\alpha) * 100\%$ of the cases.

This situation was modelled in a simple artificial domain. In a universe spanned by six boolean attributes $\{a_1 \ldots a_6\}$, we defined a sequence of two (rather different) target concepts $A \Leftrightarrow a_1 \wedge a_2$ and $B \Leftrightarrow (a_3 \wedge a_4) \vee (a_5 \wedge a_6)$, where A would gradually change into B. In order to ensure that the test set also reflected the

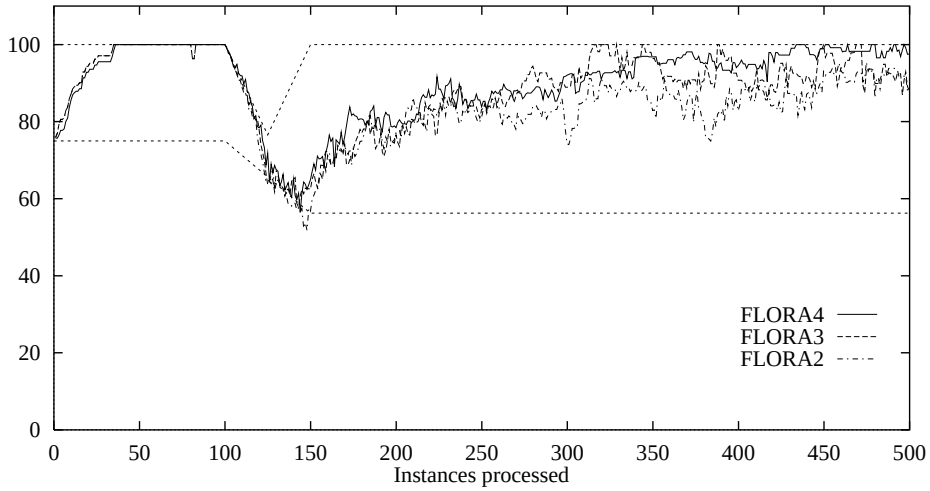

Figure 13. Performance for $\Delta x = 50$.

changing environment, the same set of 200 testing examples was used throughout each run, but the instances in the set were relabeled after each training example was processed; the relabeling was probabilistic in the same way as the labeling of training examples (i.e., it was also determined by the function α). The drift rates compared were $\Delta x = 50$ (moderately fast drift), $\Delta x = 100$, and $\Delta x = 200$ (very slow drift). $X1$ (the point where the concept begins to drift from A to B) was at 100 instances. As in all other experiments in this paper, the following results are averages over 10 runs.

Figures 13 through 15 compare our three learners in each of the three different drift situations. In addition, for purposes of orientation, the dotted horizontal lines in the figures indicate simple hypothetical upper and lower bounds on predictive accuracy in this task. The upper line plots the maximum accuracy that could be achieved if a learner had perfect information (i.e., if it knew both *when* the concept starts to change, and *what* the target concepts are). In effect, the upper bound indicates the effect of the *noise* created by the slow drift. The lower bound is simply the accuracy of guessing the majority class (which is always $\ominus$ in our case).

The most important finding is that the qualitative behavior of the learners seems to be quite robust. As expected, the shape of the valley of decreased accuracy depends on the slope of the drift function α. All three algorithms usually start to recover and readjust before the concept change is complete (i.e., while there is still noise in the data). *FLORA4* seems to do best overall, at least for $\Delta x = 50$ and $\Delta x = 100$. No significant difference can be found in the case with the longest period of uncertainty ($\Delta x = 200$), which seems to confuse all three learners alike.

The robustness of the Window Adjustment Heuristic is also documented in Figure 16, which plots *FLORA4*'s window sizes (again averaged over the 10 runs) in the

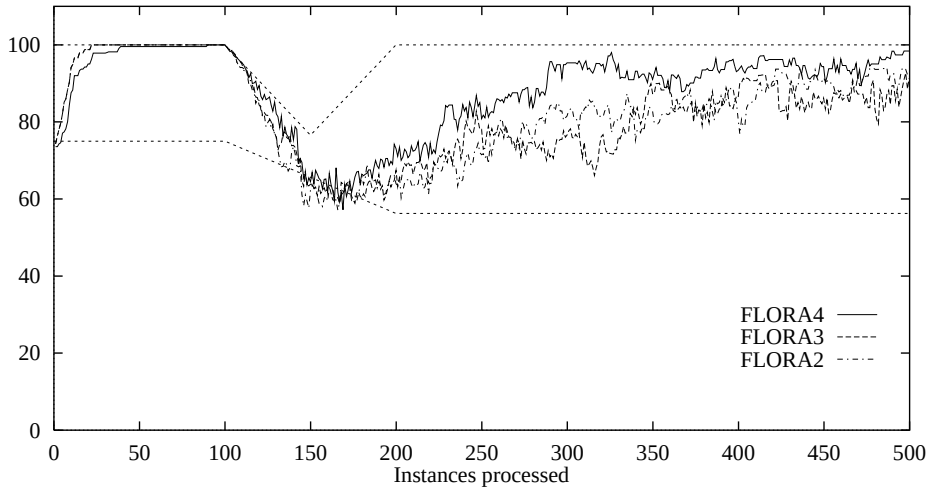

Figure 14. Performance for $\Delta x = 100$.

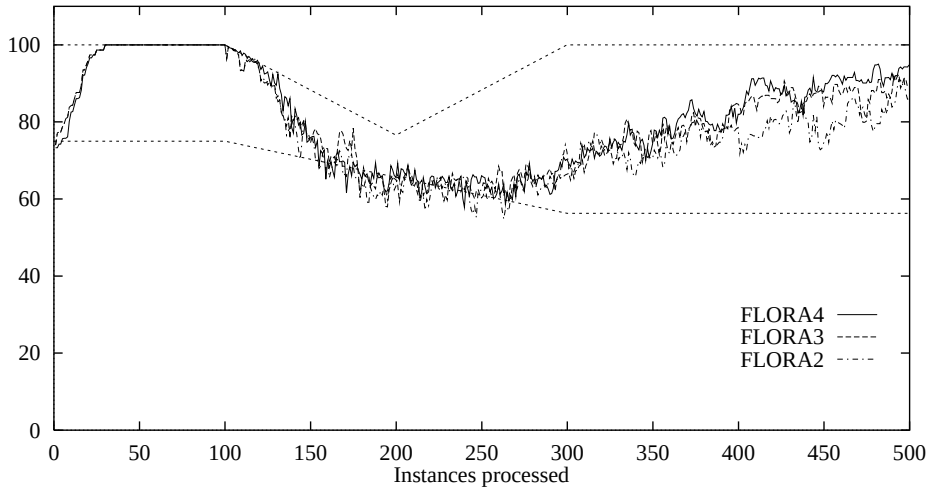

Figure 15. Performance for $\Delta x = 200$.

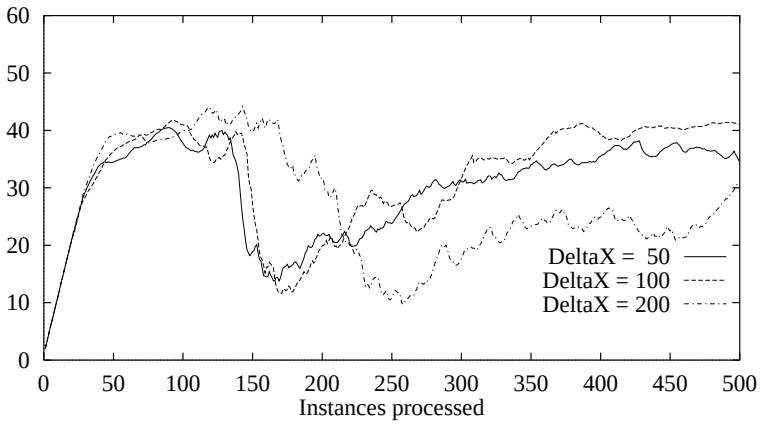

Figure 16. Window size for different speed of drift (FLORA4).

three experiments. The characteristic shape (narrowing of window at beginning of drift, then increase until stable concept is learned) is clearly recognizable in all three curves. The more sudden the drift, the easier it is to detect, and the steeper is the window narrowing curve. Note also the stable range of the window size over the entire experiment: in none of the three conditions did the window grow to an unreasonable size, nor did it collapse except in situations of concept drift or noise.

7.3. Varying the extent of drift

Another important dimension is the *extent* of drift, i.e., the dissimilarity between two successive concepts A and B. Computational learning theory quantifies drift extent as the relative error between the two concepts, which is the probability that B will misclassify a randomly drawn example that is labeled according to A (and vice versa). We can view this as the probability of drawing an example from the symmetric difference, $A \oplus B$, of the two concepts. Theoretical results like those in Section 3 suggest that the smaller the extent, the easier it should be for a learner to track the drift.

Again, we used our artificial domain defined by six boolean attributes $\{a_1 \ldots a_6\}$ to test this conjecture. We devised a 'starting concept' A and four different 'successor' concepts B_i with linearly increasing degrees of dissimilarity to A. The concepts are defined as follows:

$$
\begin{aligned}
A &\Leftrightarrow a_1 \wedge \overline{a}_2, \\
B1 &\Leftrightarrow [a_1 \wedge \overline{a}_2 \wedge (a_3 \vee a_4)] \vee [\overline{a}_1 \wedge a_2 \wedge a_3 \wedge a_4], \\
B2 &\Leftrightarrow [a_1 \wedge \overline{a}_2 \wedge a_3] \vee [\overline{a}_1 \wedge a_2 \wedge a_3], \\
B3 &\Leftrightarrow [a_1 \wedge \overline{a}_2 \wedge a_3 \wedge a_4] \vee [\overline{a}_1 \wedge a_2 \wedge (a_3 \vee a_4)], \\
B4 &\Leftrightarrow \overline{a}_1 \wedge a_2.
\end{aligned}
$$

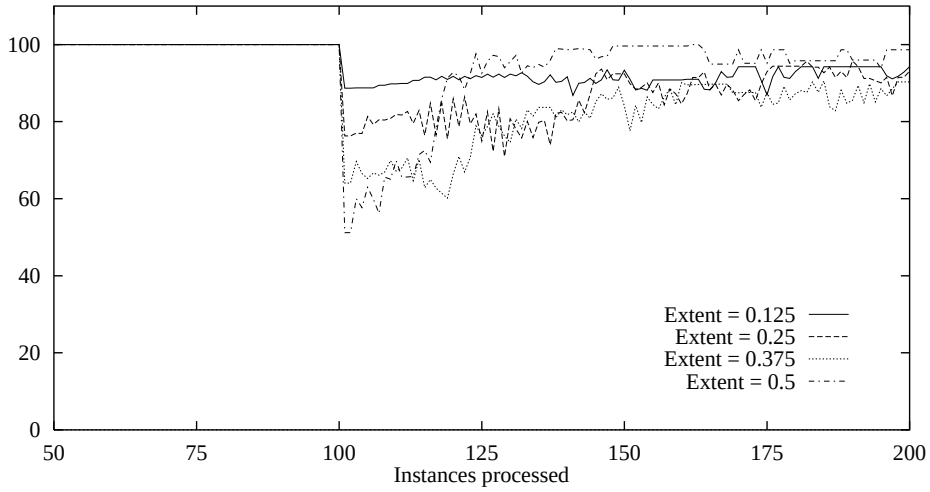

Figure 17. FLORA4 at drift of varying extent.

Assuming a uniform probability distribution over the instance space, the extents of difference are as follows:[6] $ext(A, B1) = 8/64 = 0.125$; $ext(A, B2) = 16/64 = 0.25$; $ext(A, B3) = 24/64 = 0.375$; and $ext(A, B4) = 32/64 = 0.5$.

As the previous experiments have shown, *FLORA4* is the most stable learner overall. We ran *FLORA4* on the four concept sequences $A \rightarrow B1$, $A \rightarrow B2$, $A \rightarrow B3$, and $A \rightarrow B4$. The results (accuracy measured on test set of 200 instances, plots are averages over 10 runs) are shown in Figure 17.

At first sight, the results seem rather surprising. To be sure, the drops in accuracy after the context change are as expected: the smaller the extent of change, the smaller the drop. That is a trivial consequence of applying the old concept A to a new test set that reflects the instance distribution of the new concept B_i and its overlap with A. But contrary to what theory seems to tell us — namely, that the smaller the extent, the quicker the learner's recovery — we see that in fact the concept sequence with the largest extent (0.5) led to the fastest readjustment!

Closer examination of the results and learning protocols reveals the reasons for the apparent paradox: learning theory is (of course) right, but it is not applicable to our scenario. The theoretical models of Section 3 assume a window of fixed (large) size, while *FLORA4* reduces its window very effectively once it comes to believe in a concept change; this is effective enough to deal with drift of any extent.

For the *FLORA* systems, the effectiveness of adjusting to drift depends on (1) whether the concept drift is perceived at all — which may in fact be *easier* when the difference between concepts is larger — and (2) what the new concept looks like. It is the *complexity* of the concept descriptions that causes problems for the *FLORA* systems, in particular through the *Window Adjustment Heuristic*. Whether the window grows to an appropriate size depends on the syntactic complexity of the

concept's description and the absolute and relative frequency with which incoming examples confirm a particular conjunct of the hypothesis. That is also a function of the *sparsity* of the concept. In our experiment, the concept with the largest drift extent ($B4$) was also the simplest one, syntactically speaking, and that is why it was learned much more easily than the others. We can easily get better results for this experiment by modifying those parameters of the *WAH* that relate to the complexity of a hypothesis vis-a-vis the number of examples covered (*lc* and *hc* — see Table 2), but that is not the point of this exercise.

Our analysis points to a fundamental problem with all the *FLORA* systems: the *Window Adjustment Heuristic* — and, by implication, the entire hypothesis maintenance algorithm — is rather sensitive to the form and complexity of the target concept. For practical applications, preliminary experiments with the aim of finding good parameter settings for the *WAH* will be essential. From a theoretical point of view, the situation is not very satisfactory, but we have not been able to devise a general solution so far.

7.4. Adding irrelevant attributes

For a test of whether more irrelevant attributes would have a detrimental effect on *FLORA4*'s performance, we repeated the above experiment with examples that possessed four additional boolean attributes with randomly assigned values. That gives a total of 10 attributes and increases the size of the instance space from $2^6 = 64$ to $2^{10} = 1024$ and the number of possible hypotheses (all of which *FLORA4* can theoretically represent) from $2^{2^6} \approx 10^{18}$ to $2^{2^{10}} \approx 10^{300}$. The target concepts were the same as above.

The experimental result, as given in Figure 18, does not show any grave effects. The simplest concept ($B4$) is still easily learned. The more complex concepts show some slight deterioration in comparison to Figure 17, which was to be expected, but no inordinate degradation occurs. This is a consequence of the explicit symbolic generalization (projection into a lower-dimensional attribute space) in *FLORA4*; the performance of an instance-based learner like *IB3* would degrade much more dramatically, as has been shown in various empirical and theoretical studies (see, e.g., Langley & Iba, 1993). Again, we suspect that the more complex the target concepts, the stronger will be the detrimental effect of irrelevant features.

8. Related Work

Although the notion of context drift is rarely discussed explicitly in the machine learning literature, several well-known learning techniques can be ascribed a certain plasticity in the face of changes. For instance, the momentum function in the delta rule used in *neural networks* (see, e.g., Hecht-Nielsen, 1990) essentially realizes a form of memory decay; recent experience can be made to have a stronger influence on the network's internal weight configuration than very old examples. In principle,

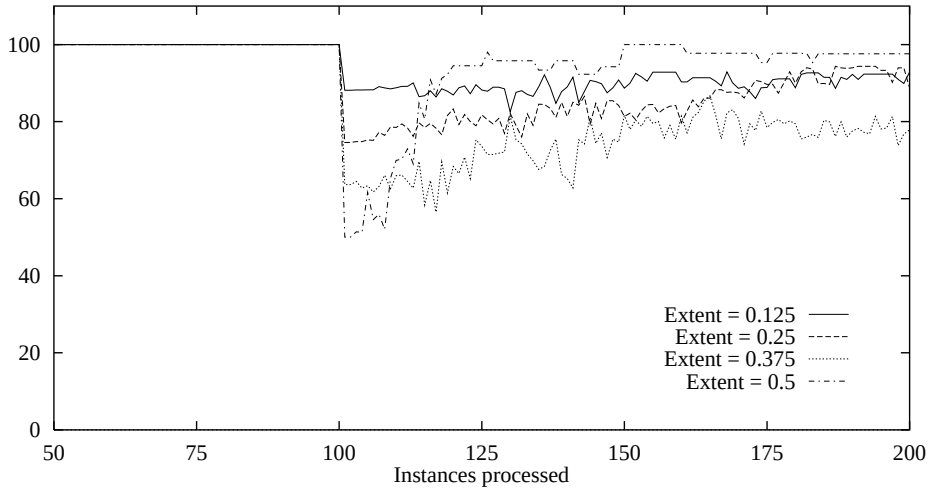

Figure 18. FLORA4 at drift of varying extent (10 attributes).

neural networks can adjust to changing contexts. For instance, Adaptive Resonance Theory (Grossberg, 1987) represents a significant step in this direction. However, even though this architecture explicitly facilitates incremental learning, it is rather reluctant to dismiss outdated information.

Simple *Instance-Based Learning* algorithms like *IB1* (Aha et al., 1991) can be viewed as incremental on-line learners that first classify each new example by some nearest-neighbor method and then store it as a new exemplar. The basic *IB1* algorithm cannot adjust to drift, since all exemplars will remain in memory, even if the context changes.

The more sophisticated variant *IB3* (Aha et al., 1991) possesses a mechanism similar to *FLORA4*'s for deciding which of the exemplars are 'trustworthy' predictors, which of them should be discarded as possibly noisy or outdated, and which are as yet undecided. The decision about the quality of an exemplar is based on its success in the tentative classification of the newly arriving examples. In the process, the individual exemplars can move between the three categories in a way similar to the description items in *FLORA*. This property gives the algorithm a strong capability to track concept drift. However, as our experiments have confirmed, there is a certain amount of inertia in the statistical criterion used to assess the quality of exemplars. *IB3* is well-suited to situations of slow drift, but it is somewhat reluctant to adjust quickly to radical changes. Also, instance-based algorithms are known to be sensitive to attribute relevance: irrelevant attributes have a detrimental effect on predictive accuracy, though some approaches to improve on this have recently been suggested (Salzberg, 1991; Cost & Salzberg, 1993).

In recent years, some authors have begun to explicitly address the problem of concept drift and context dependence. Probably the first system to attack the

problem of drift was *STAGGER* (Schlimmer & Granger, 1986), which learns symbolic characterizations from classified examples. The main adjustment mechanism in *STAGGER* is again of a statistical nature: for each description item, *STAGGER* maintains statistics of logical sufficiency (LS) and necessity (LN) of the item for the target concept, and these determine which description items will be used in further generalization, and which ones will be dropped. *STAGGER* adjusts to changes quite effectively.

As briefly noted in Section 4.2, *STAGGER* exhibits a strong sensitivity to overtraining: the longer it has been trained on a particular target concept, the slower it is in adapting to changes and tracking a concept drift. Schlimmer and Granger regard that as an asset — it mirrors empirical results from the psychology of learning. The *FLORA* systems show no such behavior, because their windows do not grow linearly with the number of examples processed. The window is kept at a more or less fixed size once the learner's hypotheses are stable. Psychological plausibility may be lost, but the guarantee of quick adjustment to changes irrespective of the learning history may be an advantage in certain practical applications.

In contrast to *FLORA3*, *STAGGER* does not possess the ability to recognize recurring contexts and take advantage of that in periodic or otherwise regular environments. On the other hand, it can use already learned concepts in the characterization of other, more abstract concepts. This capability of *constructive induction* (Michalski, 1983) is not implemented in *FLORA3*, although the contexts recognized by *FLORA3* can be viewed as constructed higher-level attributes that might be used explicitly to characterize different situations.

The idea of introducing a *forgetting* operator to improve learning was discussed in (Markovitch & Scott, 1988), in the context of a system that learned macro operators for search. Their experiments had nothing to do with concept drift, but were motivated by the so-called *utility problem* in Explanation-Based Learning (Mitchell et al., 1986), which is the problem that learned macro-operators or schemata, even if correct, are not always helpful, but may actually decrease the performance of the system. Similar observations were made by Tambe and Newell (1988) and Minton (1988). The general conclusion is that forgetting can be beneficial even in stable domains.

Forgetting as a means of adjusting to concept drift was used in the original *FLORA* system described in (Kubat, 1989), which was also applied to a practical problem (Kubat, 1992). Forgetting was controlled by a window of fixed size, which was sufficient for the particular application, but turned out to be ineffective in dealing with various types of concept drift. The window adjustment heuristic introduced in this paper significantly increased the system's flexibility and power.

An alternative to a time window as a means of controlling forgetting is *ageing* of knowledge. This method was used in the concept formation system *FAVORIT* (Krizakova & Kubat, 1992), which performs conceptual clustering in a way similar to Lebowitz' *UNIMEM* (Lebowitz, 1987). In *FAVORIT*, each exemplar is assigned a weight which slowly decays with time. If the same exemplar reappears, the weight is incremented. Exemplars whose weight drops below some threshold are forgotten.

Another recent concept formation system using a forgetting operator is *COBBIT* (Kilander & Jansson, 1993), which adapted *FLORA*'s windowing philosophy to unsupervised clustering scenarios.

Both ageing and window-based forgetting refer to the temporal order of the incoming training examples, i.e., to *time*. For numeric domains, an alternative approach named *density-adaptive forgetting* has been proposed by Salganicoff (1993a). The idea is not to rely solely on the age of exemplars. Rather, exemplars are forgotten only if there is subsequent information in their vicinity in attribute space to supersede them. In this way, the algorithm is more robust to drifting sampling distributions during incremental learning. The integration of some variant of this approach into the *FLORA* systems might further add to the stability of their behavior.

Finally, Turney (1993) discusses the problem of *context-dependence* from a different angle. In his scenario, the testing examples may come from a different context than the training examples. He presents various techniques for normalization of learning results and for adapting learned concepts for prediction in new contexts. Though this problem is somewhat different from ours, some of the techniques might well be transferable to the *FLORA* setting.

9. Conclusion

To recapitulate briefly, the article has presented a family of algorithms for on-line learning in domains with context-dependent concepts and concept drift. The main techniques constituting the basic method are (1) representation of hypotheses in the form of three description sets that summarize both the positive and the negative information; (2) a forgetting operator, controlled by a time window over the input stream; and (3) a method for the dynamic control of forgetting through flexible adjustment of the window during learning. The central idea is that forgetting should permit faster recovery after a context change by getting rid of outdated and contradictory information.

Experiments with an initial implementation of the basic method in the program *FLORA2* showed that the method behaves essentially as expected. In particular, the window adjustment heuristic proved to be rather robust under a variety of types of concept drift.

FLORA3 extends *FLORA2* with a mechanism for storing concepts in stable situations and recalling them in similar contexts. In environments with a relatively small number of contexts, this capability speeds up the process of re-learning concepts by biasing the learner towards generalizations that have proven useful in the past. Again, the window adjustment heuristic plays an important role in this process as an indicator of context changes.

Finally, *FLORA4* uses a refined strategy based on the monitored predictive performance of individual description items to deal with the problem of noisy data. *FLORA4*'s robustness derives from the fact that it integrates two different learning strategies. The statistical criteria used to distinguish between reliable and unreli-

able generalizations make it robust against noise, and the 'forgetting' of outdated information, controlled by reactive window adjustment, enables it to quickly adapt to new contexts. In terms of the framework of Salganicoff (1993b), *FLORA4* can be characterized as integrating "performance-error weighted forgetting" and "time-weighted forgetting".

There are numerous possibilities for further improvement. As noted before, a central problem is that the window adjustment heuristic (*WAH*) is dependent on parameters. Although the parameter settings we chose early on turned out to yield rather robust behavior in most of our artificial domains, this is not satisfactory. One possible solution might be to adapt a technique presented in (Moore, 1992), which estimates task-specific forgetting parameters (for instance-based learning) via cross-validation. Another possibility is to perform some kind of *beam search* in the space of parameter settings by having several versions of, say, *FLORA4* run in parallel and tune their parameters during learning.

The algorithm's flexibility could be further increased by combining the dynamic windowing approach with more selective forgetting mechanisms like those described in (Salganicoff, 1993b). That is, decisions as to which instances (and generalizations) to discard would be based not only on the items' age, but also on other characteristics like the relative proximity of observations and observed distributions.

In some domains, there may be contextual attributes or combinations of attributes that are characteristic of the current context and whose change signals a context change. As a simple example, as long as we are in one particular country, say, Austria, all or almost all of the cars we see will have Austrian license plates. As soon as we move to a different country, this feature will change in a systematic manner. An interesting idea might be to try to explicitly learn to recognize such clues, for example by keeping track of attributes that have a constant value over all instances in the current window. [7]

Another interesting extension would be the integration of a notion of *expectation*. In many domains, the order in which contexts can occur is not random, but highly constrained. The four seasons usually follow each other in a cycle, and countries border only on a limited number of neighbors. A learner should be able to develop expectations as to which context(s) will most likely become relevant next. This will require an explicit representation of contexts, an extension that would open the door to a number of other interesting possibilities.

Finally, the representation language can be extended. The introduction of numeric attributes, though a relatively simple step, will be important for possible applications in control or monitoring tasks. An extension of the approach to (some subset of) first-order logic will certainly be more difficult; incremental generalization and subsumption checking are not trivial problems. Nonetheless, the general ideas of dynamic forgetting and context tracking might be of interest for relational learners.

Acknowledgments

The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry for Science, Research, and Arts. The authors gratefully acknowledge the extremely helpful and constructive criticism by the editor, Douglas Fisher, and several anonymous reviewers.

Notes

1. The name *FLORA* is an acronym for FLOating Rough Approximation, which indicates that the original framework as conceived in (Kubat, 1989) was inspired by Rough Set Theory (Pawlak, 1982). *ADES* was a lower approximation of the concept; the union $ADES \cup PDES$ formed its upper approximation. This interpretation is no longer valid in the current implementation of the *FLORA* systems. The sets are maintained for practical reasons, to summarize information from the training examples.
2. One might ask why p (the threshold for acceptable predictive accuracy) should not be much higher (or even 100% if one knows that the data are noise-free). Closer investigation reveals that that would seriously destabilize *FLORA2*'s behavior. Especially during phases of concept drift, too high a threshold prevents the system from ever growing the window to a sufficient size. The value $p = 70\%$ is, of course, purely heuristic.
3. In the terminology of Section 3, the *extent* of drift is 0.59 (16/27) between concepts (1) and (2) and 0.48 (13/27) between concepts (2) and (3).
4. In fact, the system does not know how many hidden contexts there are. In the experiment reported here, the number of contexts that *FLORA3* stored was never exactly 3, as we would expect, knowing the target concept. In most cases, it was between 4 and 7.
5. The formula used to compute confidence intervals is the same as in (Aha et al., 1991) and (Kaelbling, 1994, p.283).
6. The total size of the instance space (the number of distinct examples describable with 6 boolean attributes) is $2^6 = 64$; the 'size' (the number of instances belonging to the concept) of each of the concepts is $|A| = |B1| = |B2| = |B3| = |B4| = 16$; set intersections and differences are $|A \cap B1| = 12$, $|A \oplus B1| = 8$, $|A \cap B2| = 8$, $|A \oplus B2| = 16$, $|A \cap B3| = 4$, $|A \oplus B3| = 24$, $|A \cap B4| = 0$, $|A \oplus B4| = 32$.
7. This idea was suggested by one of the anonymous reviewers of this paper.

References

Aha, D., Kibler, D., & Albert, M.K (1991). Instance-Based Learning Algorithms. *Machine Learning, 6(1)*, 37–66.

Angluin, D. (1988). Queries and Concept Learning. *Machine Learning, 2(4)*, 319–342.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. (1989). Learnability and the Vapnik-Chervonenkis Dimension. *Journal of the ACM, 36(4)*, 929–965.

Cost, S., & Salzberg, S. (1993). A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. *Machine Learning, 10(1)*, 57–78.

Grossberg, S. (1987). Competitive Learning: From Interactive Activation to Adaptive Resonance. *Cognitive Science, 11*, 23–63.

Hecht-Nielsen, R. (1990). *Neurocomputing*. Reading, MA: Addison-Wesley.

Helmbold, D.P., Littlestone, N., & Long, P.M. (1992). Apple-Tasting and Nearly One-Sided Learning. *Proceedings of the 33rd Annual Symposium on the Foundations of Computer Science* (pp. 493–502). IEEE Computer Science Press.

Helmbold, D.P., & Long, P.M. (1991). Tracking Drifting Concepts Using Random Examples. *Proceedings of the Fourth Annual Workshop on Computational Learning Theory* (pp. 13–23). San Mateo, CA: Morgan Kaufmann.

Helmbold, D.P., & Long, P.M. (1994). Tracking Drifting Concepts by Minimizing Disagreements. *Machine Learning, 14(1)*, 27–45.

Kaelbling, L.P. (1994). Associative Reinforcement Learning: Functions in k-DNF. *Machine Learning, 15(3)*, 279–298.

Kilander, F., & Jansson, C.G. (1993). COBBIT - A Control Procedure for COBWEB in the Presence of Concept Drift. *Proceedings of the Sixth European Conference on Machine Learning* (pp. 244–261). Berlin: Springer Verlag.

Krizakova, I., & Kubat, M. (1992). FAVORIT: Concept Formation with Ageing of Knowledge *Pattern Recognition Letters, 13*, 19–25.

Kubat, M. (1989). Floating Approximation in Time-Varying Knowledge Bases. *Pattern Recognition Letters, 10*, 223–227.

Kubat, M. (1991). Conceptual Inductive Learning: The Case of Unreliable Teachers. *Artificial Intelligence, 52*, 169–182.

Kubat, M. (1992). A Machine Learning Based Approach to Load Balancing in Computer Networks. *Cybernetics and Systems, 23*, 389–400.

Kubat, M. (1993). Flexible Concept Learning in Real-Time Systems. *Journal of Intelligent and Robotic Systems, 8*, 155–171.

Kuh, A., Petsche, T., & Rivest, R.L. (1991). Learning Time-varying Concepts. *Advances in Neural Information Processing Systems, 3* (pp. 183–189). San Mateo, CA: Morgan Kaufmann.

Kuh, A., Petsche, T., & Rivest, R.L. (1992). Incrementally Learning Time-varying Half-planes. *Advances in Neural Information Processing Systems, 4* (pp. 920–927). San Mateo, CA: Morgan Kaufmann.

Langley, P., Gennari, J.H., & Iba W. (1987). Hill-Climbing Theories of Learning. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 312–323). Los Altos, CA: Morgan Kaufmann.

Langley, P., & Iba, W. (1993). Average-case Analysis of a Nearest-Neighbor Algorithm. *Proceedings of the Thirteenth International Conference on Artificial Intelligence* (pp. 889–894). San Mateo, CA: Morgan Kaufmann.

Lebowitz, M. (1987). Experiments with Incremental Concept Formation: UNIMEM. *Machine Learning, 2(2)*, 103–138.

Maass, W. (1991). On-line Learning with an Oblivious Environment and the Power of Randomization. *Proceedings of the Fourth Annual Workshop on Computational Learning Theory* (pp. 167–175). San Mateo, CA: Morgan Kaufmann.

Markovitch, S., & Scott, P.D. (1988). The Role of Forgetting in Learning. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 450–465). San Mateo, CA: Morgan Kaufmann.

Michalski, R.S. (1983). A Theory and Methodology of Inductive Learning. In R.S. Michalski, J.G. Carbonell and T.M. Mitchell (eds.), *Machine Learning: An Artificial Intelligence Approach*, (Vol. 1). San Mateo, CA: Morgan Kaufmann.

Minton, S. (1988). Quantitative Results Concerning the Utility of Explanation-Based Learning. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 564–569). San Mateo, CA: Morgan Kaufmann.

Mitchell, T.M., Keller, R.M., & Kedar-Cabelli, S.T. (1986). Explanation-Based Generalization: A Unifying View. *Machine Learning, 1(1)*, 47–80.

Moore, A.W. (1992). Fast, Robust Adaptive Control by Learning only Forward Models. *Advances in Neural Information Processing Systems, 4* (pp. 571–578). San Mateo, CA: Morgan Kaufmann.

Pawlak, Z. (1982). Rough Sets. *International Journal of Computer and Information Sciences, 11*, 341–356.

Salganicoff, M. (1993a). Density-Adaptive Learning and Forgetting. *Proceedings of the 10th International Conference on Machine Learning* (pp. 276–283). San Mateo, CA: Morgan Kaufmann.

Salganicoff, M. (1993b). *Explicit Forgetting Algorithms for Memory-Based Learning* (Technical Report MS-CIS-93-80). Philadelphia, PA: University of Pennsylvania, Department of Computer and Information Science.

Salzberg, S. (1991). A Nearest Hyperrectangle Learning Method. *Machine Learning, 6(3)*, 251–276.

Schlimmer, J.C., & Granger, R.H. (1986). Incremental Learning from Noisy Data. *Machine Learning, 1 (3)*, 317–354.

Tambe, M., & Newell, A. (1988). Some Chunks are Expensive. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 451–458). San Mateo, CA: Morgan Kaufmann.

Turney, P.D. (1993). Robust Classification with Context-Sensitive Features. *Proceedings of the 6th Intl. Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems* (pp. 268–276). Edinburgh, Scotland.

Valiant, L. (1984). A Theory of the Learnable. *Communications of the ACM, 27(11)*, 1134–1142.

Widmer, G. (1994). Combining Robustness and Flexibility in Learning Drifting Concepts. *Proceedings of the 11th European Conference on Artificial Intelligence* (pp. 468–472). Chichester: Wiley & Sons.

Widmer, G., & Kubat, M. (1992). Learning Flexible Concepts from Streams of Examples: *FLORA2*. *Proceedings of the 10th European Conference on Artificial Intelligence* (pp. 463–467). Chichester: Wiley & Sons.

Widmer, G., & Kubat, M. (1993). Effective Learning in Dynamic Environments by Explicit Context Tracking. *Proceedings of the Sixth European Conference on Machine Learning* (pp. 227–243). Berlin: Springer Verlag.