Learning in the Presence of Concept Drift and Hidden Contexts

Gerhard WIDMER
Department of Medical Cybernetics and Artificial Intelligence, University of Vienna
and
Austrian Research Institute for Artificial Intelligence,
Schottengasse 3, A-1010 Vienna, Austria

Miroslav KUBAT
Ludwig-Boltzmann Institute of Medical Informatics and Neuroinformatics
and
Department of Medical Informatics, Institute of Biomedical Engineering
Graz University of Technology
Brockmanngasse 41, A-8010 Graz, Austria

Abstract

On-line learning in domains where the target concept depends on some hidden context poses serious problems. Context shifts 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 if a previous context re-appears; 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 levels noise and different extent and speed of concept drift.

Key words. Incremental concept learning, on-line learning, context dependence, concept drift, forgetting

1 Introduction

The work presented here relates to the global model of *incremental* or *on-line* concept learning, which has recently received considerable attention among theoreticians (see Angluin, 1988; Maass, 1991; Helmbold, Littlestone, and Long, 1992; and the references therein) as well as practitioners (Schlimmer and Granger, 1986; Langley et al., 1987; Kubat, 1989, 1993; Salganicoff, 1993a; Widmer and Kubat, 1993; and many others). The principal task is to learn a concept incrementally by processing labelled training examples one at a time. From another point of view, the problem may also 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 something else in Siberia and in Central Africa; Beatles fans had a different idea of fashionable hair-cut than the 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 and Granger, 1986).

An example of real-world concept drift is described in (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—their set is referred

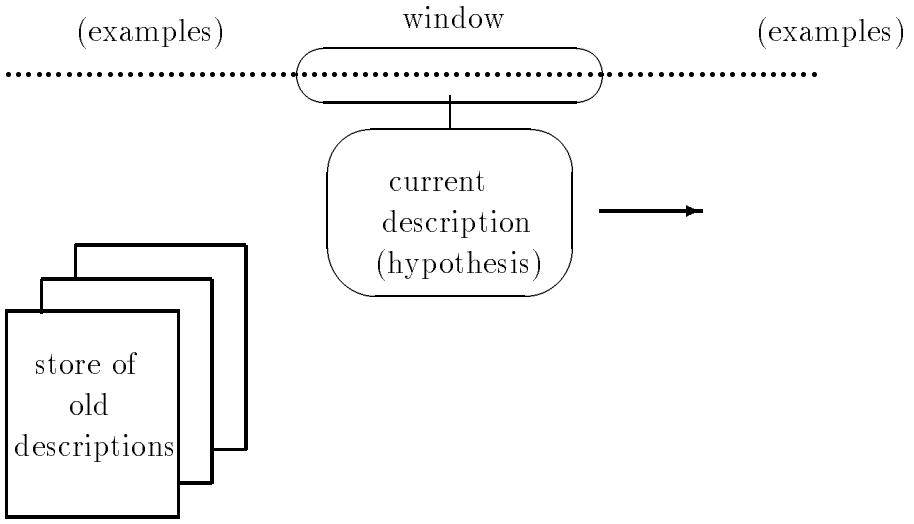

Figure 1: Current and old concept descriptions and the window moving along the stream of examples.

to as the *window*. Newly arrived examples are added to the window and the oldest ones tend to be deleted from it. Both of these actions (addition and deletion) trigger due 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. In the terminology of (Salganicoff, 1993b), this would be called "time-based forgetting".

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 the picture. When the learner comes to suspect 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 one.

Generally, a learning algorithm embodying these ideas needs (1) operators for the modification of 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 for maintaining such a store of current and old descriptions and the ability to assess 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 con-

sistent 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 against noise in the input data. Experiments with all three algorithms will be described. Finally, in section 7 we discuss related research in Machine Learning and some relevant results in Cognitive Science.

2 Learning and Forgetting: The General *FLORA* Framework

At the moment, 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 description.

In the *FLORA* framework, a *concept description* or *hypothesis* is represented in the form of three description sets: the set *ADES* contains description items matching only positive examples; it can be regarded as a DNF formula representing the current (positive) concept hypothesis. The set *NDES* similarly summarizes the negative examples; and *PDES* contains description items that are slightly too general, matching positive, but also some negative examples.[1]

Assume the following structure:

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

[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.

where $ADes_i$, $PDes_i$, and $NDes_i$ are 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 hypothesis. Only such items are retained that cover at least one example in the current window.

The description items in *ADES* and *PDES* are generated by incremental generalization in response to positive and negative instances. When new instances are added to or deleted from the current 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. *PDES* acts as a reservoir of potentially useful hypotheses. 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 or deletion of a positive ($\oplus$) or negative ($\ominus$) instance, respectively. (The extent of these transitions for the case n arrivals and m deletions is quantified in Kubat, 1991.)

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

The actions taken when removing an example from the window are rather

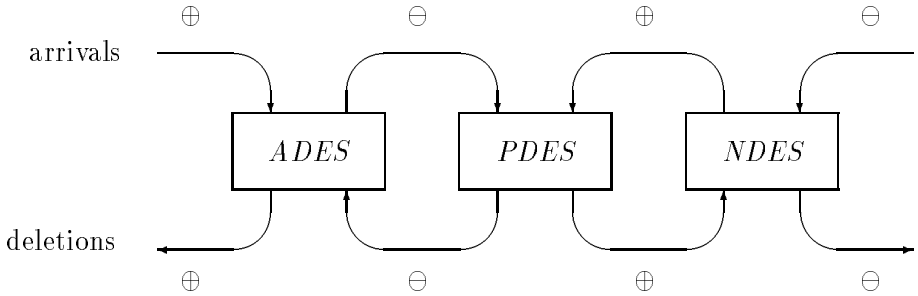

Figure 2: Transitions among the description sets

straightforward. In the case of a new example coming in, the respective counters in *PDES* are incremented and description items in *NDES* matching the instance are moved to *PDES*. For *ADES*, there are three possible cases: if the example matches some items, the counters are simply incremented; otherwise, if some item can be generalized so as to accommodate the instance without becoming too general (i.e., subsuming some item in *PDES* or *NDES*), this is carried out, otherwise the description of the instance is added to *ADES* as a whole.

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 or negative) instance cannot be incorporated consistently into any of the generalizations, its full description is added to *ADES* (*NDES*); it acts as a kind of specific 'seed' which will be generalized later on. Overly general descriptions are discarded when old examples are forgotten.

The general approach presented here assumes that only the latest examples are relevant and should be kept in the window, and that only those description items matter that are consistent with the examples in the window. While each newly arriving example is automatically included in the window, the question of how many examples should actually be deleted is more difficult. In the following section, we briefly review some theoretical results from computational learning theory as far as they pertain to this question. The following section then describes *FLORA2*, our first incarnation of the *FLORA* framework which explicitly reasons about the appropriate window size during learning.

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

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 in $XDES$ to cover instance I (if possible without subsuming some item in the other sets *YDES, ZDES*); 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 generalize(I,ADES,PDES,NDES)
    else 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;
```

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 et al. (1991, 1992) have explicitly 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. Their main results are:

- a general algorithm that tolerates concept drift of extent up to $\Delta \leq c_1\epsilon^2/(d\ln(1/\epsilon))$;
- a randomized version of this algorithm which 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), which essentially say that no algorithm can track concept drift greater than $c_3\epsilon^2/n$ (where n is the dimension of the example space) if the prediction error is to stay below ϵ.

Here, the c_i's are positive constants, ϵ is the maximum allowed probability of misclassifying the next incoming example, and d is the Vapnik-Chervonenkis dimension 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).

In addition, 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, the algorithm they describe recomputes the current hypothesis from the entire window after every new instance ("batch learning"). This is a reasonable assumption in complexity theory, but is not 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 which in this case turns out to be $w(\epsilon, \delta) = m(\epsilon, \delta/2)$ where $m(\epsilon, \delta)$ is derived from the general bound on the number of training examples that guarantee PAC-learning (Blumer et al., 1989): $m(\epsilon, \delta) = max(\frac{4}{\epsilon} \log \frac{2}{\delta}, \frac{8d}{\epsilon} \log \frac{13}{\epsilon})$ (where d is the VC-dimension of the class of target concepts). 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 at this time.

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 the 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." This 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*

4.1 Description of *FLORA2*

The first realization of the *FLORA* framework that we will discuss is the algorithm *FLORA2* (Widmer and Kubat, 1992), which attempts to dynamically adapt the size of its window during the learning process. 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, in turn, 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 drift 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 drift occurs, *FLORA2* uses two heuristic indicators: the system's predictive performance *Acc* (monitored over a fixed number of past classification attempts) and syntactic properties of the evolving hypotheses. The basic assumption is that sharp drops in *Acc* or a sudden explosion of the number of description items in *ADES* may signal a possible concept drift. As both of these indicators depend heavily on the learning task characteristics, 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 here the ratio of the number of instances covered by items in the set and the size (in terms of the number of literals) of the set.

Table 2 shows the Window Adjustment Heuristic (*WAH*) that is currently used in *FLORA2*. To put it in words, the *WAH* decreases the window size by

Table 2. Function L :=*how_many_to_forget*(lc, hc, p)

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* | W |                            /* 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 */
```

20% if a concept drift is suspected; the window size is decreased by 1 (after the addition of the new example, delete the two oldest examples) if the hypothesis seems to be extremely stable; this is to avoid keeping in memory unnecessarily large numbers of examples. If the current hypothesis seems stable enough, the window size is simply left unchanged. And if none of these conditions is satisfied, the program assumes that more information is needed and does not forget the oldest example, thus incrementally increasing the window size.

The particular parameter settings currently used in *FLORA2* are $lc = 1.2$, $hc = 4.0$ and $p = 70\%$. Given that the heuristic is (necessarily) very syntactically oriented and is thus very sensitive to the description language used, it seems hopeless to try to make it completely general and free of parameters.

4.2 Experiments

4.2.1 Adjusting to concept drift

To permit comparison with one of the first concept drift trackers, *STAGGER* (Schlimmer and Granger, 1986), *FLORA2* was tested on the same artificial learning problem as used by Schlimmer and Granger. There is a sequence of three target concepts (1) $size = small \wedge color = red$, (2) $color = green \vee shape = circular$ and (3) $size = (medium \vee large)$ in a simple blocks world. Training instances are generated randomly according to the hidden concept, and after processing each instance, the predictive accuracy is tested on an

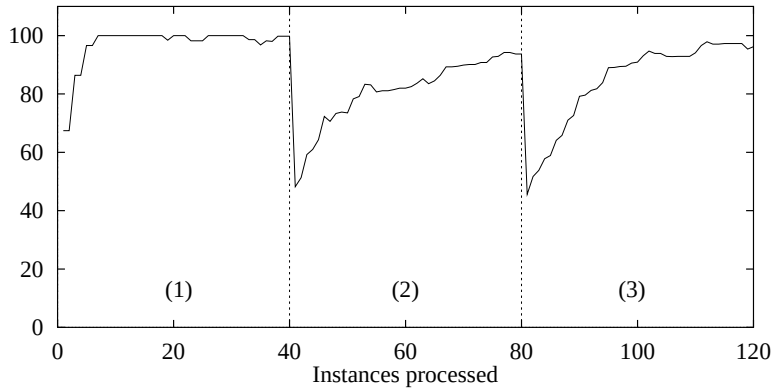

Figure 3: Adjusting to drift: predictive accuracy.

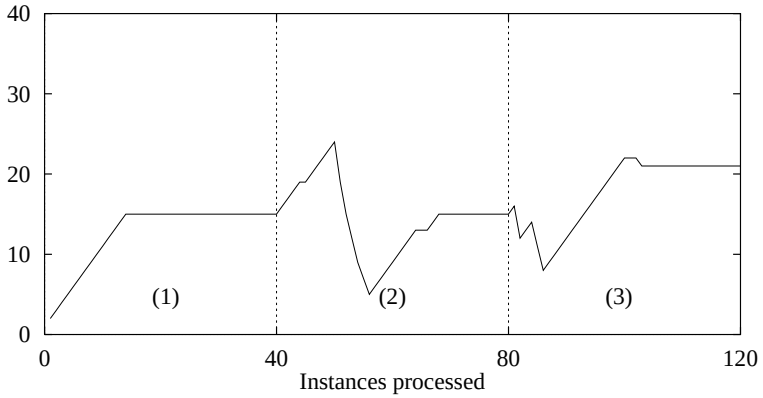

Figure 4: Adjusting to drift: dynamic window size.

independent test set of 100 instances, also generated randomly. The underlying concept is made to change after every 40 training examples. The results are averaged over 10 runs. Figure 3 shows *FLORA2*'s predictive accuracy on the test set after every training step. 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. That this is due to the workings of the *Window Adjustment Heuristic* can be seen from figure 4, which 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 more or less 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

disturbing instances.

A comparison of these curves with the respective figures in (Schlimmer and Granger, 1986) suggests that *FLORA2* compares well with *STAGGER* in terms of convergence and re-adjustment speed. However, *FLORA2*'s explicit drift detection mechanism (the *WAH*) has additional advantages, as will be seen in section 5, where we introduce an algorithm for explicit context handling.

4.2.2 Slow drift vs. radical shifts

There are many different kinds of concept drift in the real world. Characteristic attribute values may change, the value domains of attributes may evolve over time, attributes once important may become meaningless, new ones may emerge; differences between successive concepts may be drastic or only affect some small facet; some concepts change only gradually, creating phases of ambiguity and uncertainty between periods of stability, other concepts may change overnight.

Here we show some simple experiments testing *FLORA2* along two of these dimensions. The following two expectations were to be verified: (1) the smaller the difference between two successive target concepts, the faster *FLORA2* adjusts to the change; and (2) abrupt concept changes are easier to deal with than slow drift associated with periods of uncertainty.[2] Both expectations are tested in artificial domains with binary attributes.

Figure 5 shows *FLORA2*'s performance (again averaged over 10 runs) on a set of three problems with more or less drastic differences between successive concepts. In a universe defined by six Boolean attributes $\{a_1 \ldots a_6\}$, we defined concepts $A \Leftrightarrow a_1 \vee a_2 \vee a_3$ and $B \Leftrightarrow a_4 \vee a_5 \vee a_6$ and two concepts 'in between': $A' \Leftrightarrow a_1 \vee a_2 \vee a_6$ and $B' \Leftrightarrow a_1 \vee a_5 \vee a_6$. Obviously, A' is more similar to A than to B, B' is more similar to B, and A and B are most dissimilar. The figure plots *FLORA2*'s predictive accuracy in learning from three different series of concepts: A followed by B (followed by another period of B) (solid line); A followed by B' followed by B (dashed line); and A followed by A' followed by B (dotted line). The graphs show quite clearly that the rate of recovery from a concept change is directly related to the extent of the change.

The second experiment is concerned with the speed of concept drift. Sometimes concepts will change only gradually, creating a period of uncertainty between stable concept states. The new concept only gradually takes over, and some examples may still be classified according to the old concept. A typical example is a beginning malfunction of some measuring device that first

[2] This effect is a consequence of *FLORA2*'s forgetting strategy and is contrary to the assumptions of Helmbold and Long (1994), who assume steady, but almost imperceptible drift.

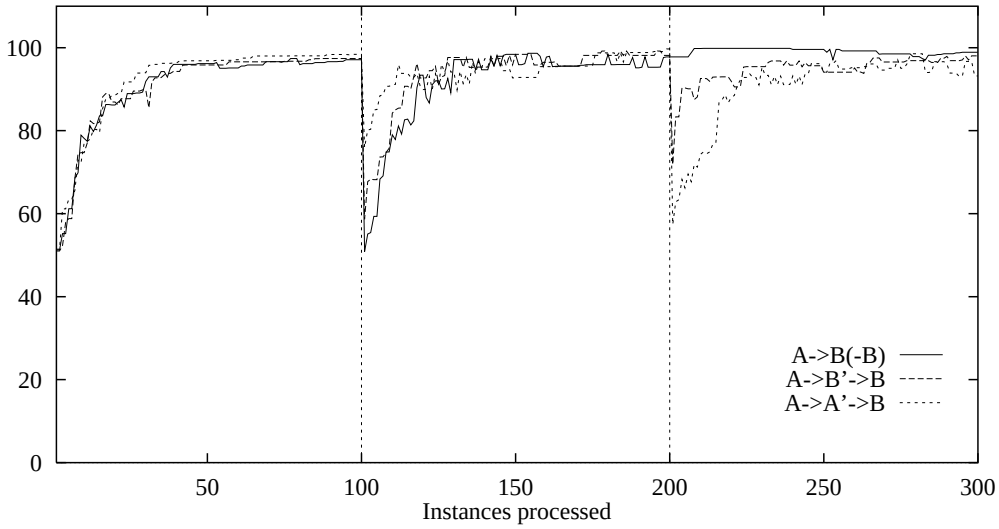

Figure 5: More or less radical shifts.

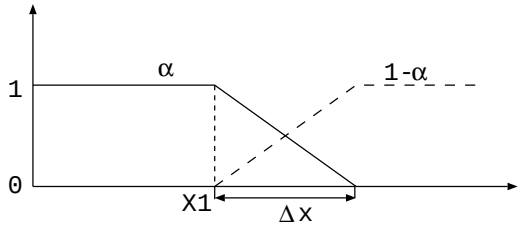

Figure 6: The function α.

fails (classifies in a new way) only sometimes, until the new mode becomes dominant. In effect, the period between stable states creates varying levels of noise. This can modelled by a function α (see figure 6) 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 6 should represent time; if we assume that instances arrive at constant intervals, time can be equated with the number of examples processed so far. $X1$ 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 is already in effect in $(1 - \alpha) * 100\%$ of the cases.

Figure 7 compares *FLORA2*'s performance in situations of varying drift rates. The target concepts used were $A \Leftrightarrow a_1 \wedge a_2$ and $B \Leftrightarrow (a_3 \wedge a_4) \vee (a_5 \wedge a_6)$.

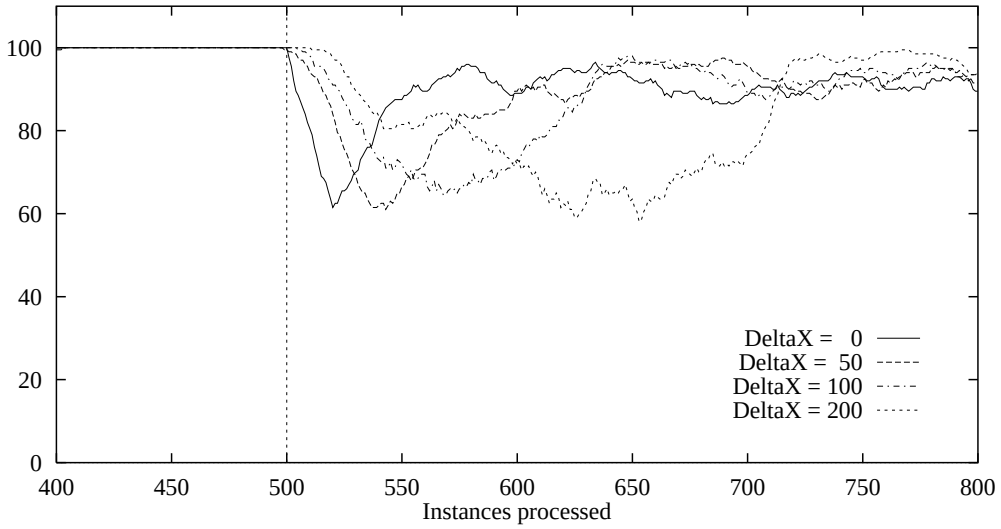

Figure 7: Slow vs. fast drift.

The drift rates compared were $\Delta x = 0$ (sudden shift), $\Delta x = 50$, $\Delta x = 100$, and $\Delta x = 200$ (very slow drift). $X1$ (the point where the concept begins to drift from A to B) was at 500 instances.

In this situation of continuously varying concept dominance, it does not make sense to test predictive accuracy on an objective, independent test set. Instead, figure 7 plots *FLORA2*'s *current accuracy*, i.e., its success at predicting the class of the incoming training examples before learning from them (averaged over the last 20 predictions).

The most important finding is that the qualitative behavior of *FLORA2* seems to be quite robust. As expected, the shape of the valley of decreased accuracy depends on the slope of the drift function α. However, *FLORA2* has usually regained high accuracy shortly after the concept change is complete.

The robustness of the Window Adjustment Heuristic is also documented in figure 8, which plots the window sizes (averaged over the 10 runs) in the same experiment. The characteristic shape (narrowing of window at beginning of drift, then increase until stable concept is learned) is clearly recognizable in all four curves. The more sudden the drift, the easier is it 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 four conditions did the window grow to an unreasonable size, nor did it collapse except in situations of concept drift or noise.

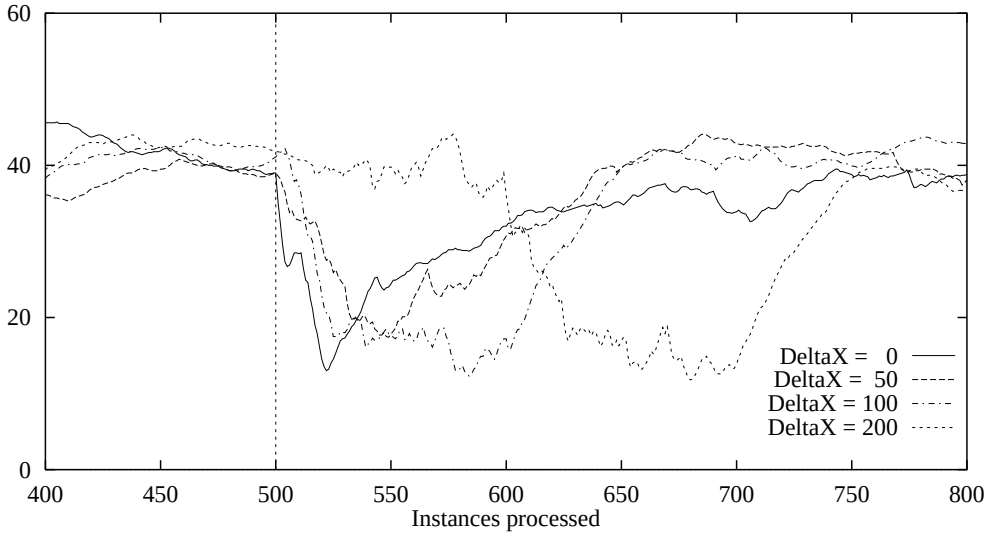

Figure 8: Window size for slow vs. fast drift.

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. The specific environment where a robot is expected to work might consist of several rooms, each with its own characteristics. A country has only a limited number of neighbors, with more or less different sets of rules relevant to many types of tasks.

In domains where contexts and associated concept versions reappear, it would be a waste of effort to re-learn an old concept from scratch every time. Instead, concepts or hypotheses should be saved so that they can be re-examined 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*

Table 3 sketches the top-level structure of *FLORA3*. In principle, it is similar to *FLORA2* — the system first tries to classify the new incoming example, updates its classification record, then learns from the instance by incorporating it into the window and updating the description sets, and, after calling the

Table 3. The general framework of *FLORA3*

input: stream of examples
parameters lc, hc and p to control the window size

functions:

- *classify(I)* ... uses the contents of *ADES* to classify instance I;
- *learn_from(I)* ... adds I to the window and updates the description sets;
- *how_many_to_forget(lc,hc,p)* ... uses the *WAH* to determine the window size;
- *forget_the_oldest(L)* ... deletes the oldest L examples from the window and updates the description sets;
- *choose_context* ... uses information from the *WAH* and recent classifications to decide whether to supersede the current context with an older one.

algorithm:

```
for i := 1 to no_of_examples
  begin   classify(example(i));
          learn_from(example(i));
          L := how_many_to_forget(lc,hc,p);
          forget_the_oldest(L);
          choose_context
  end.
```

WAH to decide whether and how to adjust the window size, 'forgets' the appropriate number of old instances. However, after each such learning cycle, *FLORA3* inspects the current state of learning in order to decide whether it should reconsider some old context (that is, a concept description that was 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, when some similar context as the current one reappears. It is the *Window Adjustment Heuristic* (*WAH*) as embodied in the function *how_many_to_forget* in Table 2 that tries to determine precisely these circumstances (the occurrence of a context change and the stability of the learning situation). So in *FLORA3*, storage and re-examination 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 4. When the current concept is *stable* according to the *WAH*, the system saves the current concept hypothesis for later reuse, unless there is already a stored concept with the same set of *ADES*

Table 4. Function *choose_context*

denotations:

Stable ... boolean variable; *true* if the current hypothesis is stable according to the *WAH*; *false* otherwise;

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.
```

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 re-installed as the new hypothesis.

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 re-generalized. 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 every example in the window. All description items that have counters equal to zero after re-generalization are removed as incorrect or irrelevant.

The algorithm for re-assessing old concepts proceeds in three steps (see Table 4):

1. *Find the best candidate* among the stored concepts: an old concept 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 that 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 re-processing 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 the 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 re-processing episodes. Old concepts are not reconsidered after every new training instance; they are only retrieved when the window adjustment heuristic suspects that a concept drift is taking place. And second, the expensive part of reconsidering an old concept—the re-generalization 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, the number of positive and negative matches. This is a very weak measure, of course, and can sometimes lead to an inappropriate candidate being chosen. Thus, efficiency is achieved at the possible expense of quality.

It seems worth pointing out once more exactly what the role of the retrieved old concept/hypothesis is in this process: it is not simply retrieved and used as the current concept hypothesis. Instead, it is used as a *model* or *bias* in the process of re-generalizing the current examples. It simply 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. This 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. [3]

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 familiar *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 averaged over 10 runs.

Figure 9 compares *FLORA3* (solid line) to *FLORA2* (dashed line) on this task. The curves show quite clearly that storing and re-using old concepts leads

[3] Fashion certainly is a prime example of this phenomenon.

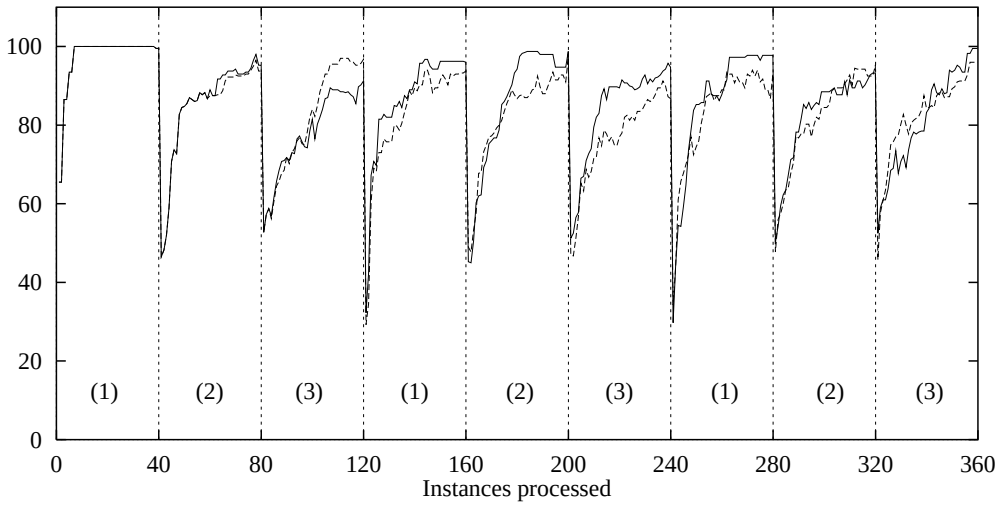

Figure 9: Learning in domain with recurring contexts.

to a considerable improvement over the simpler system when contexts actually reappear: starting from the fourth period (the first re-occurrence of context 1) the solid line shows faster readjustment and convergence in almost all cases. An interesting phenomenon appears in the third period of the plot—the first occurrence of context 3. Here, *FLORA2* did better than *FLORA3*. This may seem odd at first sight, as there is no context recurrence at this point, so ideally, both systems should behave the same. However, 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 and Kubat, 1993). However, there it also turned out that very slow concept drift and especially noise in the training data destabilized *FLORA3*'s performance more than seemed strictly necessary. The following section discusses the problems with noise in more detail and describes an extension of the *FLORA* strategy that tries to rectify that problem.

[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.

6 Robustness Against Noise: *FLORA4*

Generally, it is very difficult in strictly 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 over-reacting to noise, erroneously interpreting it as concept drift. This leads to unstable behaviour and low predictive accuracy in noisy environments. On the other hand, an incremental learner that is designed primarily to be highly robust against noise runs the risk of not recognizing real changes in the target concepts and may thus adjust to changing conditions very slowly, or only when the concepts change radically. An ideal learner should combine stability and robustness against noise with flexible and effective context tracking capabilities. On the face of it, these two requirements seem diametrically opposed. But we can at least try to achieve a compromise between them.

A simple analysis of *FLORA3* reveals that the algorithm's 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 hypothesis and cause it to be moved from *ADES* to *PDES*, even if this hypothesis covers a large number of positive examples. This can lead to somewhat unstable behavior even in noise-free domains, especially when a concept drift is taking place, but it is particularly problematic when the input data are noisy, i.e., when some of the training examples may be mislabelled.

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 to classify each incoming example before learning from it. Now the central idea in *FLORA4* is to keep a classification record for each individual description item (conjunction) and to construct statistical *confidence intervals* with a given confidence level around these measures. Decisions as to when to move an item from one set to another or when to drop it altogether are now 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; and 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). Confidence intervals are computed as in (Aha et al., 1991).

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; 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.

This general approach to deciding which hypotheses to trust has been adopted from the instance-based learning method *IB3* (Aha et al, 1991), which also uses statistical confidence 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 our experiments with *FLORA4*, we used a confidence level $\mu = 80\%$.

The main effect of this 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

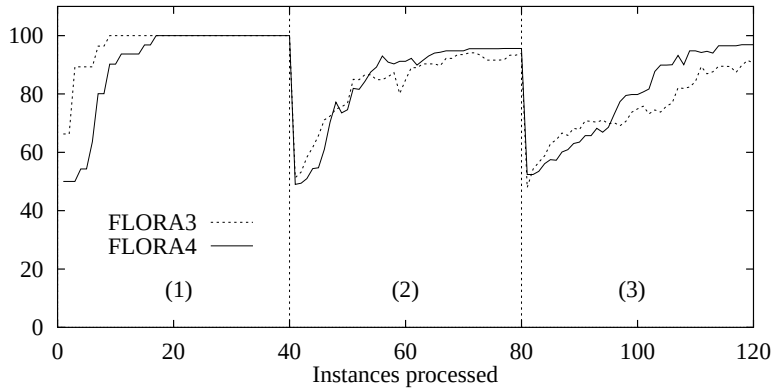

Figure 10: *FLORA3* vs. *FLORA4* on sequence of three concepts.

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, remains unchanged. After every learning step the window adjustment heuristic is invoked and may decide to grow the window or shrink it. 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 relevant generalizations.

6.2 Experiments

This section describes two sets of experiments that test *FLORA4* vis-a-vis *FLORA3* (and *IB3*) both in noise-free and noisy concept drift scenarios. Again, the artificial *STAGGER* domain with a sequence of three rather different concepts was used, and the results are averaged over 10 runs.

6.2.1 Adjusting to concept drift: *FLORA4* vs. *FLORA3* vs. *IB3*

Figure 10 compares *FLORA4* to *FLORA3* on the basic noise-free drift tracking task. The characteristic difference between the two systems that is immediately obvious from this result, and that appeared very clearly in all experiments, 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 *FLORA3*, and usually with a smoother curve. The explanation is to be found in *FLORA4*'s statistical confidence measure. *FLORA4* reacts more reluctantly initially because several contradicting examples are necessary to

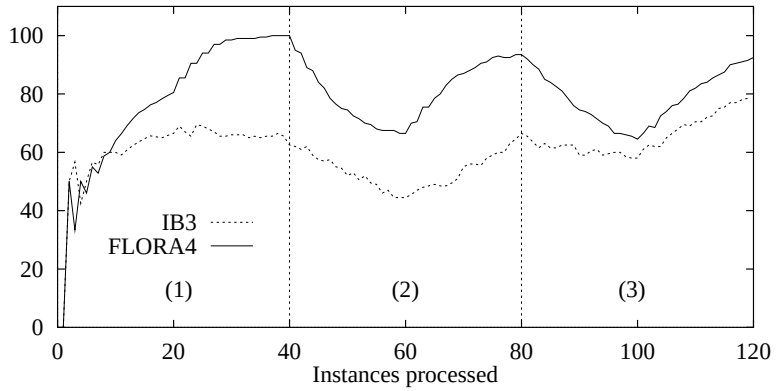

Figure 11: *FLORA4* vs. *IB3*: cumulative on-line accuracy.

invalidate a hitherto stable hypothesis in *ADES*, while *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 than its predecessor: a consequence of *FLORA3*'s strict consistency conditions 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 11 compares *FLORA4* to the publicly available version of *IB3* (Aha et al., 1991), from which our statistical method was adopted. As the method of testing employed in the previous experiment (using separate test sets) would have required substantial changes to that program, figure 11 plots the accuracy of the systems in classifying the incoming training examples, averaged over the past 20 predictions. The improvement of *FLORA4* over *IB3* is evident.

Generally, our experience from various experiments with *IB3* is that *IB3* seems to require 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.

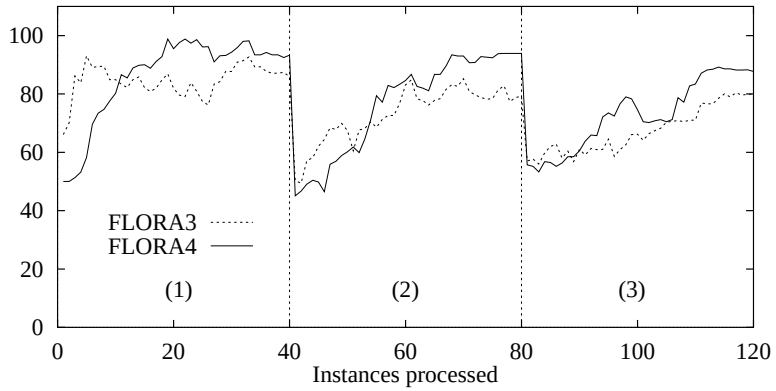

Figure 12: Performance of *FLORA3* and *FLORA4* at 20% noise.

6.2.2 Distinguishing between noise and concept drift

FLORA4's strengths should come out even more clearly when the training data are noisy. Both noise and concept drift make themselves known to the learner in the form of prediction errors. Here we expect that *FLORA4*'s combined strategy will come to bear. The statistical confidence measures provide a certain robustness against noise, especially in relatively stable situations, and the window adjustment heuristic should recognize persistent misclassifications that indicate a concept change, and should lead to effective adjustment by shrinking the window in such situations.

In the following experiment, the same target concepts were used, but the training data were corrupted with various levels of classification noise. *FLORA4* was compared to *FLORA3* throughout and turned out to be consistently and significantly superior. For instance, figure 12 compares the performance of the two systems at a noise level of 20% in the training data.[5] Here again, we see the characteristic difference: *FLORA4* is a bit slower in its initial reaction to the concept change, but then soon outperforms *FLORA3*. However, the difference between the two curves is markedly greater than in the noise-free case. *FLORA3* has obvious problems, while *FLORA4*'s accuracy quickly rises to a mark that corresponds roughly to the given level of noise (remember that 20% noise means 10% misclassified instances on average in a two-class learning task).

A comparison of the average window sizes in this experiment (figure 13) illustrates the workings of the window adjustment heuristic and also the effect

[5] 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 $\eta = 100$.

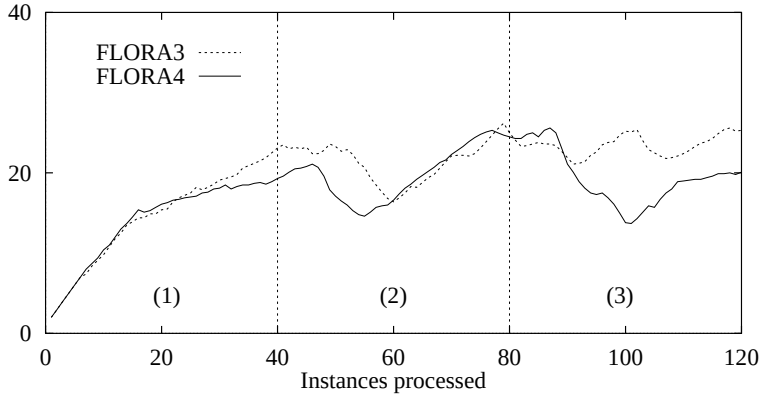

Figure 13: Average window size at 20% noise level.

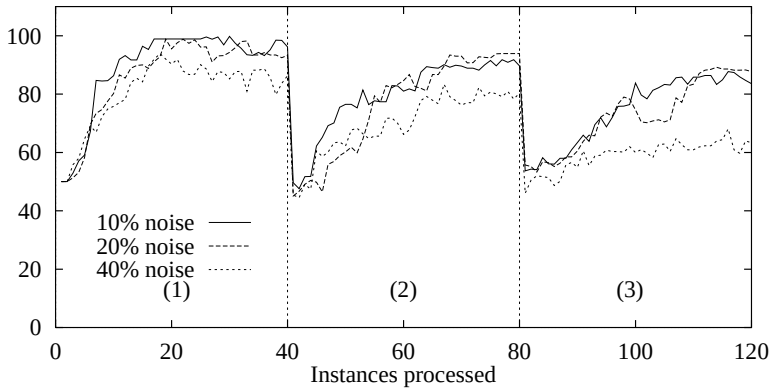

Figure 14: Performance of *FLORA4* at various noise levels.

of the statistical strategy of the generalizer. The expected characteristic shape of the curve comes out clearer in the *FLORA4* case. As the *WAH* reacts to factors (e.g., the number and complexity of accepted description items in *ADES*) that are also affected by the generalizer's strategy, there is a *synergy* between the two components: the generalizer's robustness against noise prevents the *WAH* from erroneously growing or shrinking the window. This effect is clearly visible in the third phase in figure 13, where *FLORA3* grows and shrinks the window in the middle of a phase of concept stability, obviously confused by noisy examples.

Figure 14 shows *FLORA4*'s performance at various noise levels (10, 20, and 40%). The qualitative shape of the performance curves remains unchanged. The rapid drop in accuracy after a concept change is followed by relatively fast re-convergence toward a quasi-optimal prediction accuracy. In no case does

the performance really collapse. (The closest it comes to collapsing is with the third (disjunctive) concept in the 40% noise situation, where *FLORA4*'s convergence is rather slow, but still recognizable. This part of the concept seems to be inherently more difficult to adjust to than the other two, as evidenced by *FLORA3*'s and *FLORA4*'s slower convergence even in the noise-free case (figure 10)).

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*. This 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 Related Work

The notions of concept drift and hidden contexts have a number of ramifications, both from a machine learning and a more general point of view. In the following section, we relate the *FLORA* approach to a number of approaches and methods from the field of machine learning, especially those that realize some form of forgetting. Then, some pertinent results from cognitive psychology and their relevance to the *FLORA* philosophy are briefly discussed.

7.1 Contexts and concept drift in machine learning

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 in the concept definition. 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 experiences. In principle, neural networks can adjust to changing contexts. For instance, Adaptive Resonance Theory (Carpenter and Grossberg 1988) 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* (sometimes called *memory-based learning*) algorithms like *IB1* (Aha et al., 1991) can be viewed as incremental on-line learners that first classify each newly arrived 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. As in *FLORA4*, 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 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 the predictive accuracy (though some approaches to improve on this have recently been suggested—e.g., Salzberg, 1991; Cost and 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 and 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. In contrast to *FLORA3*, it does not possess the ability to recognize recurring contexts and take advantage of this 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 and Scott, 1988), in the context of a system that learns macro operators in a search task. 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), that 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. This has also been

noted by (Tambe and Newell, 1988) and (Minton, 1988), among others. 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 and 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 and 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). Here, 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. A combination of some variant of this approach with flexible windowing strategies might lead to even more reactive systems. This is one of our current lines of research.

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.

7.2 Relevant research in cognitive science

There is quite some work on conceptualization and categorization in the field of cognitive science that touches on some aspects of the *FLORA* learning scenario. For instance, Fodor et al. (1980) argue that real-world concepts, used as words by humans, cannot be defined generally and independently from their *context*. In a recent study, Johnson-Laird (1987) distinguishes three different

entailments of a context: (1) in the case of a concept with several meanings, the context enables the selection of the particular meaning; (2) if the concept has only one meaning, the context facilitates its specification; and (3) if the concept has several aspects, the context stresses some particular aspect at the cost of other aspects.

These points do not directly translate into the *FLORA3* world, as this system does not explicitly represent *contexts*, though it does make conscious decisions about when to switch from one to another context. However, it should be easy to augment the system so that it constructs explicit context identifiers, which would allow it to explicitly reason about contexts and their influence on concept meanings or descriptions. This is a promising topic for future research.

Bartlett (1932) was the first who suggested that, in human learning, the acquisition of new knowledge is facilitated by *schemata* stored in long-term memory. Schemata are typical structures acquired by an abstraction process. The concept descriptions stored by *FLORA3* and used as a model in the re-generalization of the examples in a new context can be interpreted as such schemata. Of course, the approach reported in this paper is a heavy simplification of the general notion. Abstraction operators and a more expressive representation language would move *FLORA3* closer to this schema theory of learning.

The difference between Short-Term Memory (STM) and Long-Term Memory (LTM) has been investigated in cognitive science for a long time, and is analyzed at great length in (Klimesch, 1988). LTM takes from STM only those pieces of information that are considered relevant, interesting, etc. In our case, the window represents STM and the stored contexts represent LTM. In modeling the phenomenon of forgetting, our work builds on the idea of *memory decay*: any piece of information that enters memory tends to slowly fade out. The only way how to prevent it from being completely forgotten is by revision or a strong emotional bias. In psychology, the principle of memory decay was discovered and studied by Brown (1958), though, in principle, it relates to the very early work by Ebbinghaus (1885); for a more recent analysis see (Reitman, 1974).

Another important psychological finding relevant to our study is that forgotten pieces of information are learned faster than new ones, as reported by Nelson (1978). Even though forgotten, they leave a trace in the form of tendencies and general structures. The reader will see that this aspect is also reflected in rudimentary form in *FLORA3*.

8 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 maybe disturbing information.

Various 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 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 unreliable 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".

Generally, it is interesting to see how two conceptualizations of learning—the three description sets in *FLORA4* and Aha's categories of accepted, mediocre, and rejected predictors—that were developed out of quite different motivations (the basic motivation and interpretation framework for the description sets in the original *FLORA* approach was Pawlak's (1982) *Rough Set Theory*) finally converge on a common interpretation. We may take this as another indication that one fruitful strategy for achieving powerful learning algorithms is to actively search for promising methods in machine learning research and try to combine or integrate them in a more general framework.

There are numerous possibilities for further improvement. A general problem with the current system is that the window adjustment heuristic (*WAH*) is

dependent on parameters. Although the parameter settings we chose early on (and never changed throughout all the experiments) turned out to yield rather robust behavior in 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 *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, observed distributions, etc.

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 which would open the door to a number of other interesting possibilities (see section 7 above).

Also, the representation language should 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 no trivial problems. However, the general ideas of dynamic forgetting and context tracking might well be of interest also to relational learners.

Acknowledgments

The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry for Science and Research.

References

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

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

Bartlett, F.C. (1932). *Remembering.* Cambridge, U.K.: Cambridge University Press.

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

Brown, J. (1958). Some Tests of the Decay Theory of Immediate Memory. *Quarterly Journal of Experimental Psychology 10*, pp.12–21.

Carpenter, G.A. and Grossberg, S. (1988). The ART of Adaptive Pattern Recognition by a Self-Organising Neural Network. *IEEE Computer*, pp.77–88.

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

Ebbinghaus, H. (1885). Über das Gedächtnis. Reprinted in: Veith of Eckhardt, B. and Potter, M.C. (1985). Clauses and the Semantic Representation of Words. *Memory and Cognition 13*, pp.371–376.

Fodor, J.A., Garrett, M.F., Walker, E.C.T., and Parkes, C.H. (1980). Against Definitions. *Cognition 8*, pp.263–367.

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

Helmbold, D.P., Littlestone, N., and Long, P.M. (1992). Apple-Tasting and Nearly One-Sided Learning. *Proceedings of the 33rd Annual Symposium on the Foundations of Computer Science*, Pittsburg, PA.

Helmbold, D.P. and Long, P.M. (1991). Tracking Drifting Concepts Using Random Examples. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory (COLT-91)*, Santa Cruz, CA, pp. 13–23.

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

Johnson-Laird, P.N. (1987). The Mental Representation of the Meaning of Words. *Cognition 25*, pp.189–212.

Kilander, F. and Jansson, C.G. (1993). COBBIT - A Control Procedure for COBWEB in the Presence of Concept Drift. In *Proceedings of the European Conference on Machine Learning (ECML-93)*, Vienna, Austria.

Klimesch, W. (1988). *Struktur und Aktivierung des Gedächtnisses. Das Vernetzungsmodell: Grundlagen und Elemente einer übergreifenden Theorie.* Bern, Switzerland: Verlag Hans Huber.

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

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

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

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

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

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

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

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

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

Maass, W. (1991). On-line Learning with an Oblivious Environment and the Power of Randomization. *Proceedings of the 1991 Workshop on computational Learning Theory*

Markovitch, S. and Scott, P.D. (1988). The Role of Forgetting in Learning. *Proceedings of the 5th International Conference on Machine Learning*, Ann Arbor, MI, pp.450–465.

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. I.* San Mateo, CA: Morgan Kaufmann.

Minton, S. (1988). Quantitative Results Concerning the Utility of Explanation-Based Learning. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, Morgan Kaufmann, San Mateo, CA, pp.564–569.

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

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

Nelson, T.O. (1978). Detecting Small Amounts of Information in Memory: Savings for Nonrecognized Items. *Journal of Experimental Psychology, Human Learning and Memory 4*, pp.453–468.

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

Reitman, J.S. (1971). Without Surreptitious Rehearsal, Information in Short-Term Memory Decays. *Journal of Verbal Learning and Verbal Behavior 13*, pp.365–377.

Salganicoff, M. (1993a). Density-Adaptive Learning and Forgetting. *Proceedings of the 10th International Conference on Machine Learning*, Amherst, MA.

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

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

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

Tambe, M. and Newell, A. (1988). Some Chunks are Expensive. In *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, MI.

Turney, P.D. (1993). Robust Classification with Context-Sensitive Features. In *Proceedings of the Sixth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-93)*, Edinburgh, Scotland, pp.268–276.

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

Widmer, G. (1994). Combining Robustness and Flexibility in Learning Drifting Concepts. In *Proceedings of the 11th European Conference on Artificial Intelligence, ECAI94*, Amsterdam. Chichester: Wiley & Sons.

Widmer, G. and Kubat, M. (1992). Learning Flexible Concepts from Streams of Examples: *FLORA2*. In *Proceedings of the European Conference on Artifical Intelligence (ECAI-92)*, Vienna, Austria.

Widmer, G. and Kubat, M. (1993). Effective Learning in Dynamic Environments by Explicit Context Tracking. In *Proceedings of the European Conference on Machine Learning (ECML-93)*, Vienna. Berlin: Springer Verlag.