Effective Learning in Dynamic Environments by Explicit Context Tracking

Gerhard Widmer[1] and Miroslav Kubat[2]

[1] Dept. of Medical Cybernetics and Artificial Intelligence, University of Vienna, and Austrian Research Institute for Artificial Intelligence,
Schottengasse 3, A-1010 Vienna, Austria
e-mail: gerhard@ai.univie.ac.at
[2] Institute of Biomedical Engineering, Dept. for Medical Informatics,
Graz University of Technology
Brockmanngasse 41, A-8010 Graz, Austria
e-mail: mirek@fbmtds04.tu-graz.ac.at

Abstract. Daily experience shows that in the real world, the meaning of many concepts heavily depends on some implicit context, and changes in that context can cause radical changes in the concepts. This paper introduces a method for incremental concept learning in dynamic environments where the target concepts may be context-dependent and may change drastically over time. The method has been implemented in a system called *FLORA3*. *FLORA3* is very flexible in adapting to changes in the target concepts and tracking concept drift. Moreover, by explicitly storing old hypotheses and re-using them to bias learning in new contexts, it possesses the ability to utilize experience from previous learning. This greatly increases the system's effectiveness in environments where contexts can reoccur periodically. The paper describes the various algorithms that constitute the method and reports on several experiments that demonstrate the flexibility of *FLORA3* in dynamic environments.

1 Introduction

One of the basic tasks of Machine Learning is to provide methods for deriving descriptions of abstract concepts from their positive and negative examples. So far, many powerful algorithms have been suggested for various types of data, background knowledge, description languages, and some special 'complications' such as noise or incompleteness.

However, relatively little attention has been devoted to the influence of varying contexts. Daily experience shows that in the real world, the meaning of many concepts can heavily depend on some given context, such as season, weather, geographic coordinates, or simply the personality of the teacher. 'Ideal family' or 'affordable transportation' have different interpretations in poor countries and in the North, the meaning of 'nice weather' varies with season, and 'appropriate dress' depends on time of day, event, age, weather, and sex, among other things. So time-dependent changes in the context can induce changes in the meaning or definition of the concepts to be learned. Such changes in concept meaning are sometimes called *concept drift* (especially when they are gradual).

To discover concept drift, the learner needs feedback from its classification attempts, to update the internal concept description whenever the prediction accuracy decreases. This was the experimental setting of the system *FLORA*, which was first published in Kubat (1989), with a theoretical analysis in Kubat (1991). The system, though very simple, was successfuly applied in an expert-system-driven control mechanism for load re-distribution in computer networks (Kubat, 1992).

Frankly spoken, the original program *FLORA* was not very sophisticated from the machine learning (ML) point of view because it did not contain such common ML mechanisms as explicit generalization operators or search heuristics. These came later, in the frame of *FLORA2* (Widmer & Kubat, 1992) where some kind of intelligence was implemented (generalization, check for subsumption, and flexible reaction to the speed of drift).

Still, even this later version lacked an important attribute of intelligent behavior: the ability to use experience from previous learning. Whenever an old context reoccured, the system just blindly tried to re-learn it, waiving any previous experience. The consequence was that even if the same context re-appeared a thousand times, the system always needed, on average, the same number of examples to modify the concept description. This shortcoming motivated another upgrade of the system, *FLORA3*, which is able to adapt to concept drift while utilizing past experience and deals with recurring contexts much more effectively.

The next section discusses, in more detail, the issues of hidden contexts and concept drift, and the relevance of this problem in real-world applications. Then, the system *FLORA2* is described. Section 4 is dedicated to the main contribution of this paper, the algorithm for context tracking, which differentiates *FLORA3* from her predecessors. Section 5 reports on experimental results demonstrating the utility of the idea.

2 Dynamic environments, hidden contexts, and concept drift

When speaking about concept drift, one might distinguish two different types of drift (though they are not always clearly separable): *Real* concept drift reflects real changes in the world and can be exemplified by the changes in fashion—'fancy skirt' or 'modern music'—or language—the semantic variation of such words as left-wing policy, conservatism, or liberalism.

Virtual concept drift, on the other hand, does not occur in reality but, rather, in the computer model reflecting this reality. In a practical setting, this kind of effect can emerge when the representation language is poor and fails to identify all relevant features, or when the order of training examples for learning is skewed, so that different types of instances are not evenly distributed over the training sequence.

Many potential sources of virtual concept drift can be identified. Most typically, the teacher is to blame, having only a particular context in mind and considering only the related pieces of information to be relevant; or the teacher's

knowledge is limited. Also, the teacher may have good knowledge but some of the features may depend on values that cannot be measured, or the measurements are too expensive.

Sometimes, the agent learns by experimentation and simply does not come across all reasonable examples. For illustration, consider an autonomous agent or robot moving through a foreign world and trying to induce rules to survive (see the experiments reported in Section 5). In a complex world where not all relevant features are explicit, there is no choice but to put up with variables and predicates that can be acquired by the robot's devices. Their number is of course limited. Obviously, slightly different laws are valid in different parts of the world. If you want to grow a palm tree, you will surely apply different techniques in Africa and on the Arctic Circle.

Another aspect of the problem is that the agent does not a priori know how many contexts exist in the world, how to discern them, what their ordering is, and what impact they excercise on the concept drift. Sometimes, the drift consists in changed values of some variables, sometimes also the relevance of individual variables or predicates can dramatically change. Moreover, the transition is usually only gradual with rather fuzzy boundaries between two different concept interpretations. All this must be taken into account when building a flexible learning system.

The core idea underlying the philosophy of *FLORA* is that more recent pieces of information should receive higher levels of trust, in the belief that the older the examples, the higher the danger that they relate to an outdated context (the agent has meanwhile moved from the Arctic Circle to the Equator). The system always maintains a set of current (positive and negative) examples that represent its current world and that should be correctly described by the current concept hypothesis. The set of these examples is called window (*FLORA*, sitting in a coach, observes through it the world passing by). One by one, new examples are encountered and used to update the internal knowledge structures; at the same time, however, older examples are distrusted and deleted from the window. This, too, causes changes to the concept description. In this way, the current context is approximated—the system trusts only those examples that are currently inside the window. That enables *FLORA* to recognize a concept drift and adjust itself to it.

The latest descendant of the family, *FLORA3*, possesses the ability to store encountered contexts for future use, and to re-examine them whenever it discovers (or suspects) drift. Evidently, this makes sense only if the same (or similar) contexts reappear in the future, which is certainly the case in many realistic applications where the number of possible contexts is *finite*. For instance, there are four seasons that follow one by one 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. Even in fashion we can see that some phenomena reappear periodically, among them short skirts, preferred dress fabrics, or hair style. The same goes for contexts in political life—autocracy versus oligarchy versus democracy, lesser or greater

influence of the church, and the like. Each of them implies different heuristics for defining law, guidelines for everyday life, and morale (these reappear, too).

3 The basic *FLORA* framework: learning and forgetting

In this section we briefly review the basic learning mechanism in the *FLORA* framework, as it was already realized in *FLORA2* (Widmer & Kubat, 1992). The following section will then describe the more advanced features of *FLORA3*.

The principle of the *FLORA* algorithm is shown in Figure 1. The rectangle 'knowledge' stands for the current concept description, the rectangle 'window' contains the currently trusted examples. Each time a new example arrives, it is added to the window; from time to time, the oldest or, alternatively, least relevant example is deleted from the window. Both events necessitate updates to the concept description.

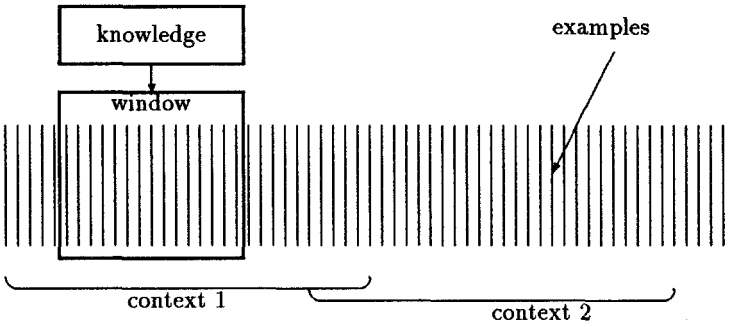

Fig. 1. The window of the system FLORA moving across the stream of examples.

The concept description is represented by three description sets, *ADES*, *PDES*, and *NDES*. The description sets are collections of *description items*—conjunctions of attribute-value pairs. Thus a description set can be interpreted as a *DNF* expression. *ADES* is a set of 'accepted' description items (*DI*s) covering only positive examples in the window (not necessarily all of them) and no negative examples; *PDES* is a set of 'potential' *DI*s, covering both positive and negative examples; *NDES* is a set of 'negative' *DI*s, covering only negative examples. Any *DI* for which we cannot find at least one example in the window is deleted. (Widmer & Kubat, 1992) gives an intuitive motivation for these three sets.

Obviously, each example that is being added to or deleted from the window may be described by a number of *DI*s. This entails the following consequences:

Adding a *positive* example to the window can cause new description items to be included in *ADES*, or some existing items to be 'confirmed,' or existing items to be transferred from *NDES* to *PDES*.

Adding a *negative* example to the window can cause new description items to be included in *NDES*, or some existing items to be 'reinforced,' or existing items to be transferred from *ADES* to *PDES*.

Forgetting an example can cause existing description items to be 'weakened,' or even deleted from the current description set, or moved from *PDES* to *ADES* (if the example was negative) or to *NDES* (if the example was positive).

Figure 2 summarizes these updates. The arrows indicate possible migrations of description items between sets after learning (L) or forgetting (F) from a positive (+) or negative (−) instance, respectively.

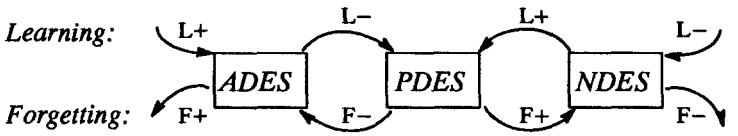

Fig. 2. Transitions among the description sets.

To operationalize this learning schema, let us recapitulate the learning algorithm of *FLORA2* as it was presented in Widmer and Kubat (1992). Assume that the three description sets already exist (at the beginning they might also be empty) and that they are encoded in the following form:

$$\begin{aligned} ADES &= \{ADes_1/AP_1, ADes_2/AP_2, \ldots\} \\ PDES &= \{PDes_1/PP_1/PN_1, \ldots\} \\ NDES &= \{NDes_1/NN_1, \ldots\} \end{aligned} \tag{1}$$

where $ADes_i$ ($PDes_i$, $NDes_i$) are description items; AP_i and PP_i represent the number of positive examples matching the respective *DI*s; and PN_i and NN_i represent the number of negative examples matching the respective *DI*s. The counters AP_i, PP_i, PN_i, and NN_i help to decide whether to move the respective item to another description set, or, if it is equal to zero, whether to drop it altogether.

In order to prevent combinatorial explosion, the sizes of these description sets must somehow be restricted. In *FLORA2*, the set *ADES* is not a set of all possible description items. It is constructed by stepwise careful generalization (see below) and, in effect, represents one non-redundant DNF formula that expresses the current concept hypothesis. The same holds for *NDES*. *Redundancy* is eliminated by checking for subsumption *within* description sets: *ADES* is kept maximally general (that is, if some description item $ADes_i$ subsumes some $ADes_j$, only $ADes_i$ is kept in *ADES*). In *PDES*, only the most specific descriptions are kept, and *NDES* is again maintained maximally general. *Inconsistency* is avoided by checking for subsumption *between* description sets. In this way, for instance,

over-generalization of $ADES$ is avoided by checking it against $PDES$ and $NDES$. These conditions are tested whenever one of the description sets is modified. The algorithms for incremental learning and forgetting then proceed as follows:

Incremental learning:

Assume that the system is presented with a new training instance, with given classification $C \in \{positive, negative\}$. Then the description sets are updated as follows (see also Figure 2):

If classification C is **positive**:

For all $ADes_i/AP_i$ in $ADES$:
    if $match(instance, ADes_i)$ then $AP_i := AP_i + 1$;
For all $PDes_i/PP_i/PN_i$ in $PDES$:
    if $match(instance, PDes_i)$ then $PP_i := PP_i + 1$;
For all $NDes_i/NN_i$ in $NDES$:
    if $match(instance, NDes_i)$ then remove $NDes_i$ from $NDES$ and include it into $PDES$ as a triple $NDes_i/1/NN_i$ and check the updated $PDES$ for subsumptions;
If there is no $ADes_i$ in $ADES$ that matches the new instance, then find a generalization of one of the $ADes_i \in ADES$ such that (1) the generalization covers the new instance; (2) the required degree of generalization is *minimal* and (3) the generalization does not subsume any descriptions in $PDES$ or $NDES$ (this ensures consistency against negative instances); as an extreme case, the description of the instance itself may be added to $ADES$; then check $ADES$ for subsumptions (remove redundant descriptions);

If classification C is **negative**, the algorithm works analogously (just exchange $ADES$ and $NDES$ in the above algorithm).

Incremental forgetting:

When an old instance is dropped from the current window and 'forgotten,' the description sets are updated as follows (again, see Figure 2):

If the instance was a **positive** one:

For all $ADes_i/AP_i$ in $ADES$:
    if $match(instance, ADes_i)$ then $AP_i := AP_i - 1$;
    if $AP_i = 0$ then remove $ADes_i$ from $ADES$;
For all $PDes_i/PP_i/PN_i$ in $PDES$:
    if $match(instance, PDes_i)$ then $PP_i := PP_i - 1$;
    if $PP_i = 0$ then remove $PDes_i$ from $PDES$ and include it into $NDES$ as a pair $PDes_i/PN_i$ and check the updated $NDES$ for subsumptions;

If the instance was a **negative** one, the algorithm works analogously (just exchange *ADES* and *NDES* in the above algorithm).

This algorithm provides the basis for learning in dynamic environments. However, more is needed to achieve really flexible and effective learning behaviour in domains with substantial concept drift.

4 *FLORA3*: Explicit context tracking

The ability to *forget*, as described in the previous section, provides the fundamental basis for the system's ability to adapt to concepts that change over time. Eventually, old instances will drop out of the window and be forgotten. However, this will work well only in domains where changes in the concepts are almost imperceptibly slow. When dramatic or sudden concept shifts occur, the fixed window size prevents the system from reacting flexibly. Ideally, when the system notices (or suspects) a beginning concept drift, it should shrink the window in order to discard old instances that now contradict the new concept. *FLORA3* includes a heuristic to automatically adjust the size of its window during learning. This is the subject of the next section.

Also, in environments where contexts can re-occur (periodically or randomly), the system would have to re-learn concepts that it had already learned at some earlier time. In such environments it would be advantageous to keep old, outdated concepts around for possible use in the future. In *FLORA3*, a mechanism for doing this has been developed. It is tightly coupled to the window adjustment algorithm and will be described in the section after next.

4.1 Automatic adjustment of window size

The behaviour of a *FLORA*-type system depends crucially on the size of the window. Too narrow a window will cause relevant instances and information to be forgotten too early; and when the window is too wide, the system will be very reluctant to follow a concept drift: it will hang on to noisy or outdated instances and hypotheses too long. The optimal window size, then, is a function of the current learning situation and should not be fixed beforehand. Rather, the learning system should be intelligent enough to automatically adjust its window size to the current demands. These demands are, of course, not clearly definable; the adjustment decisions can be made on a heuristic basis only.

We have experimented with many different heuristics for automatic window adjustment. The latest version takes into account both the complexity of the current hypothesis (vis-a-vis the number of instances covered) and the current estimated predictive accuracy of the hypothesis, which is constantly monitored by the system. The heuristic depends on three parameters which must be set by the user: lc (= threshold for low coverage of *ADES*); hc (= threshold for high coverage of *ADES*); and p (= threshold for acceptable predictive accuracy). Given these three parameters, the *window adjustment heuristic (WAH)* is defined as follows:

Let N = number of (positive) instances covered by *ADES* and
S = size of *ADES* (in terms of number of literals)
Then:
If $N/S < lc$ (coverage of *ADES* is low)
or the current predictive accuracy is bad ($< p$ and falling)
and if *PDES* is not empty (there are alternative hypotheses)
then decrease window size by 20% and forget the oldest instances
else if $N/S > 2 * hc$ (coverage extremely high)
and the current predictive accuracy is good ($> p$)
then reduce the window size by 1
else if $N/S > hc$ (coverage high)
and the current predictive accuracy is good ($> p$)
then freeze the current window size (i.e., forget one example each time a new one is added)
else grow window by 1 (accommodate new instance without forgetting the oldest one)

where the *predictive accuracy* is an incrementally computed and updated measure of how well the current hypothesis fits the data: before learning from a new training instance, *FLORA3* first tries to classify the instance on the basis of its current hypothesis; the *predictive accuracy* measure is the ratio, over the last 20 instances, of correctly classified instances.

In more colloquial terms, the window adjustment heuristic operationalizes the idea that the window should *shrink* (and old, now possibly outdated examples should be forgotten) when a concept drift seems to occur, and should be kept *fixed* when the concept seems stable enough. (When the concept is extremely stable, the window is even reduced stepwise by 1, in order to avoid keeping in memory unnecessarily large numbers of instances.) Otherwise the window should gradually *grow* until a stable concept description can be formed. The occurrence of a concept drift can only be guessed at by the learner, and the two heuristic indicators of such a drift used by *FLORA3* are (1) the complexity of the descriptions in the set *ADES* (where the intuition is that during the time of occurrence of a concept drift, it will be difficult to find a concise concept description that is consistent with all the examples), and (2) drops in the predictive accuracy, which is constantly monitored.

The ideal parameter settings will vary from one domain to the next. In all our experiments, we used $lc = 1.2$, $hc = 4$ 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 find a completely general heuristic that would not depend on any parameters. (Widmer & Kubat, 1992) discusses in more detail the effects of this heuristic on the learning process.

4.2 Storage and re-use of old contexts

As already noted, there are many natural domains where there is a finite number of hidden contexts that may reappear, either cyclically or in an unordered

fashion. In such domains, it would be a waste of effort to re-learn an old concept from scratch when it reappears. Instead, concepts or hypotheses should be saved so that they can be re-examined at some later time, when there are indications that they might be relevant again. The effect should be faster convergence if the concept (or some similar one) has already occurred. *FLORA3* includes a mechanism for doing just this. In designing this mechanism, several questions had to be answered:

1) Which parts of a hypothesis should be stored?
2) Which hypotheses are worth saving?
3) When should old hypotheses/concepts be reconsidered?
4) How can the adequacy or 'degree of fit' of an old concept be measured in a new situation?
5) How is an old hypothesis/concept to be used in a new situation?

The answer to question 1) is quite simple: the concept description/hypothesis is a triple {*ADES*, *PDES*, *NDES*} and is saved as such, because these sets summarize the current state of affairs. The match counts associated with the description items in the sets are not stored, because they will not be meaningful in some new situation.

In designing a solution to question 5), we note that when an old concept is retrieved at some later point in time, it finds itself in a new context: it will not agree with all the training instances in the current window; some items in the concept's description sets will be too specific, and others will be inconsistent with the data. Thus, it is not enough just to recompute the counts for the various description items. Instead, all the instances in the current window must be re-generalized. The retrieved concept is used as a *model* in this re-generalization process: the counts associated with all the items in the description sets are set to zero, and then the regular *FLORA* learning algorithm is invoked for every training instance in the current window. Instances that fit items already in the concept's description sets will confirm these items, and generally, those partial generalizations in the old concept that are in accordance with the new data will be used in the re-generalization process. Others will not be confirmed by the new instances and thus their counts will remain at zero. After re-generalizing all instances in the window, all those description items that still have counts of zero are removed as incorrect or irrelevant from the updated concept.

As for questions 2) - 4) — which hypotheses deserve to be saved, and when; when should old concepts be reconsidered; how is the appropriateness of an old concept to a new situation measured — the criteria that can be used to make these decisions can only be of a heuristic nature. Intuitively, only stable hypotheses/concepts should be saved, and the system should reconsider some old concepts whenever it perceives some substantial concept drift. It is the *window adjustment heuristic (WAH)* that tries to determine precisely these circumstances. So in *FLORA3*, storage and re-examination of old hypotheses are tightly linked to changes in the window size.

The complete algorithm for handling old contexts works as follows:

- When the current concept is *stable* (according to the WAH - see section 4.1): save the current hypothesis (unless there is already a stored concept with the same set of *ADES* descriptions).
- When *FLORA3* suspects a *concept drift*, i.e., when the WAH enforces a *narrowing of the window*: reconsider old, saved concepts and compare them to the current hypothesis. This is done in three steps:
  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). The best candidate according to this measure is chosen.
  2) *Update the best candidate* w.r.t. the current data: the retrieved concept description is updated by setting all the counts in the description sets to 0 and then re-processing all the instances in the window according to this hypothesis (see the above discussion).
  3) *Compare the updated best candidate to the current hypothesis*: use some 'measure of fit' to decide whether the updated candidate (= old concept) is better than the current hypothesis; if so, replace the current hypothesis with the updated old concept. In the current version of *FLORA3*, the measure of fit is simply the relative *complexity* of the description, as it is used also in the window adjustment heuristic: a hypothesis is considered better if its *ADES* set is more concise. (Remember that *ADES* covers all positive and no negative instances, so the number of instances covered is the same for the *ADES* sets in both the current hypothesis and the updated best candidate.)

As one possible class of application domains for the *FLORA* systems is flexible control in real time systems (cf. Kubat, 1992), *efficiency* of the learning algorithm is an important criterion. The above algorithm tries to maintain efficiency by limiting the number of expensive re-processing episodes. First, 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. Which old concept is the best candidate is determined through a simple heuristic measure, the number of positive and negative matches (see above). 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 old concepts/hypotheses is in this process: at the time of a suspected concept shift, an old concept is used to *bias* the re-generalization of the examples in the window. It is not just retrieved and used as the current concept hypothesis. Instead, the old concept is used as a *model* for the re-generalization of the instances: 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 hidden context reappears, the target concepts will tend to be *similar*, but not necessarily *identical* to how they appeared in the old context. [3]

5 Experimental results

In (Widmer & Kubat, 1992) it was shown that *FLORA2* (i.e., the basic learning algorithm plus automatic window adjustment) compares very favourably with systems like *STAGGER* (Schlimmer & Granger, 1986), which was also designed to deal with problems of concept drift. Here, we will concentrate on demonstrating that in domains with recurring hidden contexts, learning can still be considerably improved by explicitly storing and re-using old concepts.

We have done extensive experiments with *FLORA3* in various artificial domains, where we had full control over the rate and strength of concept drift. [4] In each case, we contrasted two versions of *FLORA3*, one with and one without the algorithm for re-examining old concepts (the latter one will be called *FLORA2* here, as it corresponds essentially to the system described in (Widmer & Kubat, 1992)).

For reasons of comparability, the first set of experiments used the same kind of data and concepts that were originally introduced in (Schlimmer & Granger, 1986) and then also used in (Widmer & Kubat, 1992), namely, a sequence of three (rather different) 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 were generated randomly according to the hidden concept, and after processing each instance, the predictive accuracy was tested on 40 test instances (also generated randomly, according to the same underlying concept). The underlying concept was made to change after every 40 training instance, in the cyclic order 1-2-3-1-2-3-1-2-3. Thus, we created a situation of recurring concepts. This experiment was repeated several times. In the following plots, the solid line represents the results achieved by *FLORA3*, and the dashed line gives the results for *FLORA2*. Figure 3 displays the results of two typical individual runs, and Figure 4 shows the averaged results of 10 runs. The dotted vertical lines indicate where the underlying concept changes.

In interpreting these results, we first note that both systems do recover very quickly (in most cases) from changes in the underlying hidden concept. This is due to the basic learning and forgetting operators and to the window adjustment heuristic. This was discussed already in our previous publication on *FLORA2*.

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

[4] We also did experiments with 'real world' data, namely, the well-known *lymphography* data from Ljubljana. Ivan Bratko (personal communication) had suggested that there might be some perceptible concept drift in this data set. However, when analyzing *FLORA3*'s behaviour on these data, we could not discern any noticeable drift, and in comparative learning experiments, *FLORA3*'s performance on these data lagged behind that of a non-incremental learner like *CN2* (Clark & Niblett, 1989), so we concluded that the lymphography data were not appropriate for studying issues of concept drift.

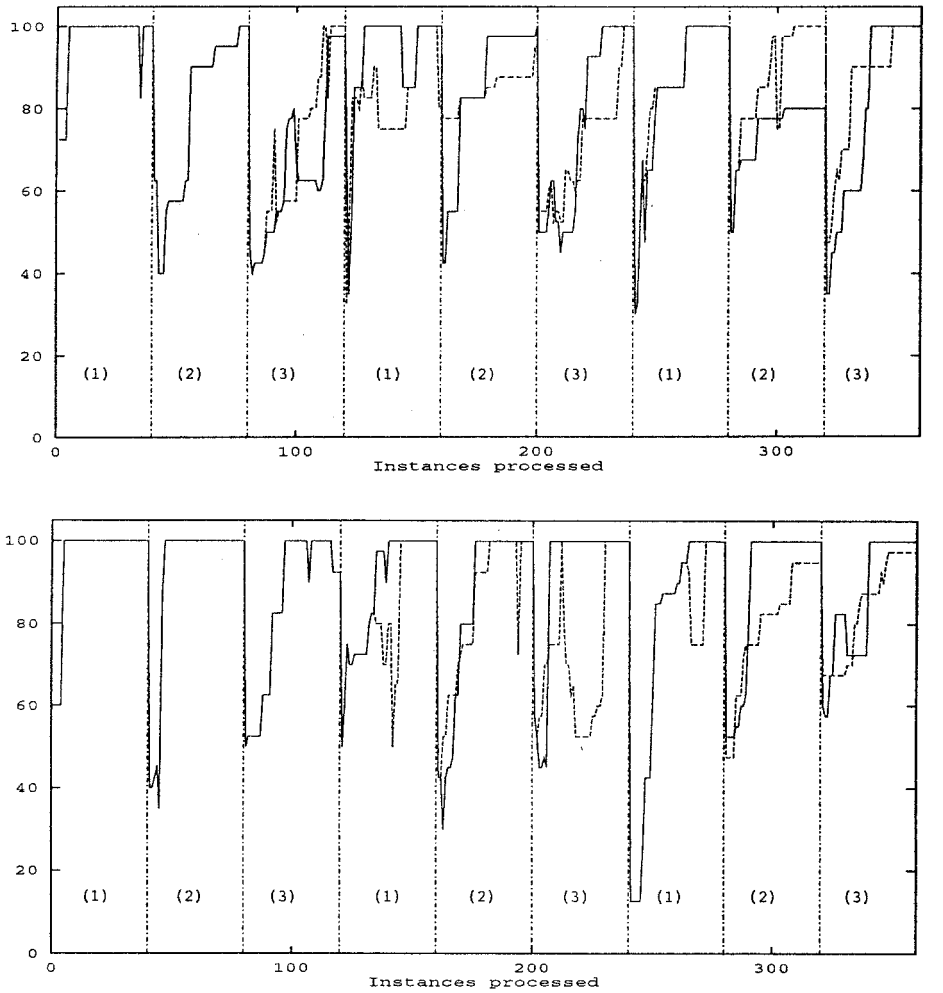

Fig. 3. Predictive accuracy in two individual runs (experiment 1).

What is more interesting here is that in situations where a concept re-occurs after a few cycles, there is a marked superiority of *FLORA3* over *FLORA2* in re-adjusting to this old concept. This can be seen most clearly from the two single-run plots, where *FLORA3* returns to the 100% mark much faster than does *FLORA2*. This strongly confirms the utility and importance of the context tracking mechanism in domains with recurring contexts.

The superiority of *FLORA3* can also be seen in the averaged plot in Figure 4, albeit less clearly. In the particular experiment summarized in this figure, it happened that in 2 out of the 10 random runs, *FLORA3* performed worse than *FLORA2* on the third concept – in fact, very much worse (which caused the average to be pushed below the curve for *FLORA2*). In both cases, the

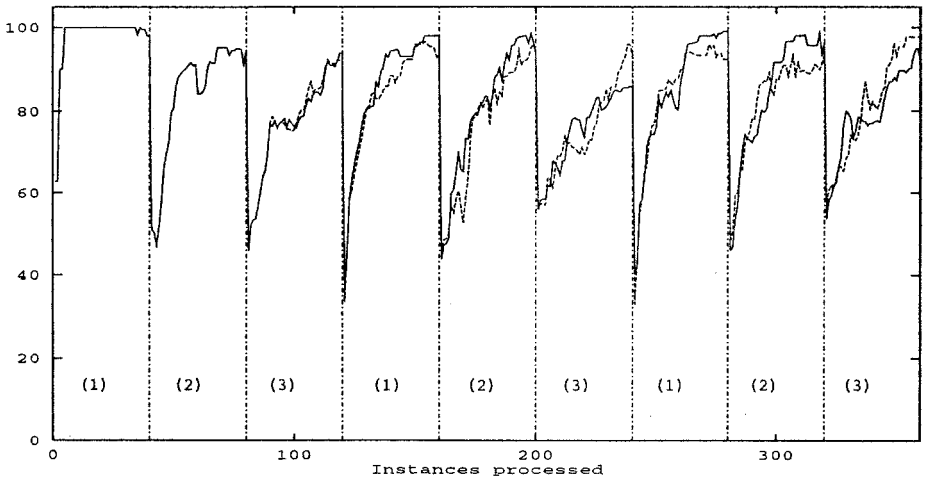

Fig. 4. Predictive accuracy, averaged over 10 runs (experiment 1).

reason was that *FLORA3* stored some overly general hypotheses during the first occurrence of concept 3, which caused it to go astray in subsequent occurrences of this same concept. As the context-tracking mechanisms of *FLORA3* are very heuristic in nature, irregularities like this are unavoidable.

Note that at the beginning (during the first occurrence of each concept, i.e, when no context recurrence happens), *FLORA3*, with its explicit reconsideration of saved hypotheses, actually seems to perform a bit worse than the simpler *FLORA2* (see the first plot in Figure 3). This has happened quite frequently in our experiments. There is a simple explanation for this. The availability of stored contexts may in fact sometimes lead the learning process astray: due to the heuristic nature of the context retrieval decisions, some context may erroneously be selected because it seems to be better than the current hypothesis. So the context tracking mechanism adds another degree of freedom - or source of potential errors, if you will - to the learning algorithm. 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.

In a second set of experiments, we started from a fictitious scenario of an autonomous agent by the name of *FLORA3* exploring some unknown territory (planet), searching for food and trying to predict where food might be found. On this planet, food can be found in containers distributed throughout the country, and these containers are characterized by many attributes (such as their shape, color, size, weight, material, whether they hang from trees or bushes, ...). Some containers do contain food, others don't. Now this particular planet is divided into several kingdoms, and the rules determining whether a container holds food

are different in every kingdom.[5] [6] Again, training data for this set of experiments were generated randomly – in this case, not only the descriptions of the training instances, but also *FLORA*'s path through the various kingdoms.

This learning problem is more difficult not only because it deals with a larger description space, but also because we introduced two sources of *noise* into this world: the borders of the kingdoms were designed to be imprecise, that is, in the vicinity of the border between two kingdoms, the instance generator assumed that with a certain probability, concepts (food containers) from both kingdoms could occur. And the second source of noise was the fact that *FLORA3*'s path (as a sequence of moves in arbitrary direction) was also generated randomly, so there was, in most cases, no clear transition from one context to another; rather, *FLORA3* would sometimes wander back and forth between two kingdoms before finally venturing more deeply into one of them.

As an example of *FLORA3*'s performance in this domain, Figure 5 shows the result of one random run. The planet in this experiment had 6 different kingdoms arranged in a circle, and *FLORA3* wandered through this circle twice. Note that in this figure there are no vertical lines to indicate the precise points where the underlying context changes, because the two types of noise mentioned above make it difficult to determine precisely when *FLORA3* is in a new context. (However, the reader is encouraged to examine the plot and try to guess where *FLORA3* is deeply in a particular kingdom.)

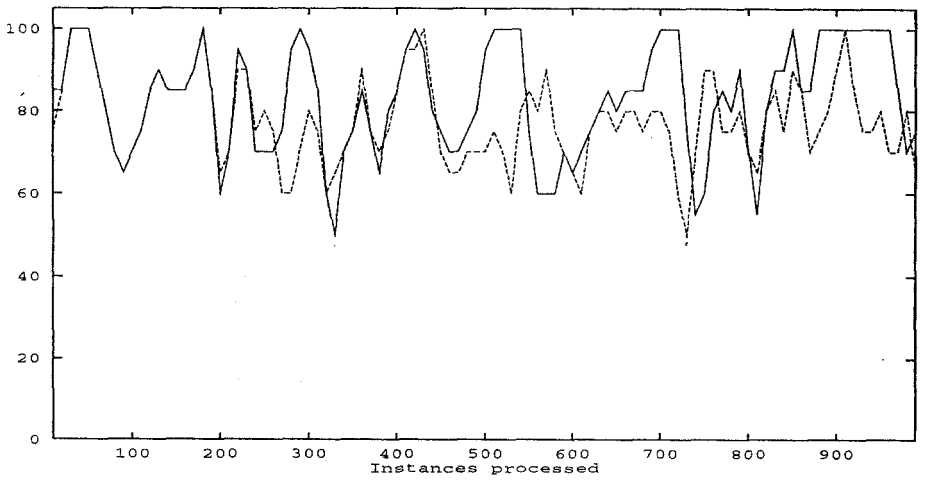

Fig. 5. Predictive accuracy on strange planet (experiment 2).

[5] Miroslav Kubat has a more elaborate story around this scenario, but space does not permit us to repeat it here.

[6] Readers less inclined towards fairytales might simply imagine a robot in a complex building with different types of rooms, where each room presents the robot with different operating conditions.

Basically, the results in this set of experiments confirmed our expectations. The systems' behaviour is again characterized by quite flexible adjustment to new contexts, with *FLORA3* markedly better in situations where old contexts reoccur. Due to the noise in the training data and the larger hypothesis space, convergence was, of course, slower than in the first type of experiments, and not always perfect, but our preliminary experience is that *FLORA3* seems to handle limited amounts of noise quite well.

6 Discussion and related work

To recapitulate briefly, the two basic ideas that contributed to the success of *FLORA3* are (1) recognizing and 'forgetting' old, harmful knowledge, and (2) explicitly storing old concepts and re-using them when a context transition seems to take place.

The idea of using forgetting to improve learning may seem counter-intuitive at first sight, but it has been suggested in the literature by a number of authors. Indeed, a kind of forgetting was implemented already in (Samuel, 1959) with the objective of avoiding the danger of storing prohibitively many pieces of experience in the memory. Samuel used *refreshing* (reinforcement) when a description item was utilized. The algorithm is very simple: After regular intervals, the item's age is incremented by 1. If the age exceeds a prespecified threshold, the item is deleted—forgotten. Reinforcement, in turn, consists in decrementing the age. Also Fikes, Hart, and Nilsson (1972) suggested that some sort of forgetting should be considered to prevent an unmanageably large store of experience. For some other considerations on forgetting, see Markovitch and Scott (1988), Torgo and Kubat (1991), or Kubat and Krizakova (1992).

In the above references, forgetting was understood mainly as a measure to prevent uncontrolled growth of the occupied memory, with the subsequent problem of computational tractability. Another motivation can be selection or filtering of the most useful knowledge to get rid of noise—this is the rationale behind pruning mechanisms in decision trees (Niblett, 1987), which also represent a kind of forgetting.

To a limited extent, the idea of tracking concept drift in the incremental learning paradigm has been studied in the context of unsupervised learning, especially in concept formation. In the systems *COBWEB* (Fisher, 1987) and *CLASSIT* (Gennari et al., 1989), the dynamic modification of the concept description is facilitated by two complementary operators *merge* and *split*. These perform generalization of two existing concepts into one (merge) and specialization of a concept into two subconcepts (split). Similar recovery operators are available also in *UNIMEM* (Lebowitz, 1987). But none of these systems can actually discard old, harmful information, and none of them explicitly stores previous concept descriptions. However, recent work on modified versions of *COBWEB* (Kilander & Jansson, 1993) is very much related to our approach and seems to yield very promising results.

The closest relative to our program is perhaps *STAGGER* (Schlimmer & Granger, 1986), because that system was designed explicitly to deal with concept drift. We showed already in (Widmer & Kubat, 1992) that *FLORA2* compared very favourably with *STAGGER* in terms of adjustment to concept drift. *FLORA3*'s added capability to use experience from past learning in new contexts leads to even more effective learning in environments with recurring contexts.

Schlimmer and Granger mention as one of *STAGGER*'s assets that it is sensitive to the amount of previous training, that is, the longer it has been trained on some concept, the more deeply ingrained will the concept be, and the more hesitant will *STAGGER* be to abandon it and adjust to a new context. This seems to mirror some results from the psychology of learning.

FLORA3 does not exhibit this type of behaviour: once a concept is deemed stable, *FLORA3* freezes the window size, which prevents the concept from becoming too deeply ingrained. This allows the system to quickly follow radical concept shifts, no matter how stable the previous hypothesis was thought to be. We regard this as an advantage of our approach. *FLORA3* is not meant to be a psychologically valid model of learning. Our interests are in practical applications, such as robotics and control in dynamic environments with limited information. There *flexibility* seems to be of prime importance. The system should be quick in adjusting to changes in the world. A related requirement is that the learning algorithm be *efficient*. And that is clearly the case. The basic learning and forgetting operators are very simple, and also the method for reassessing old concepts has been designed so as to keep the computational overhead small (see section 4.2).

One of the goals of our future work is a better formalization of the method and a more thorough theoretical analysis of its convergence properties. For simplicity reasons, we have so far relied on a very simple representation language. Once we have a better theoretical framework, we hope to be able to extend the system so that it can deal also with more complicated description languages, e.g., some subset of first order logic as it is used in systems like *FOIL* (Quinlan, 1990) or *LINUS* (Lavrač et al, 1991).

Acknowledgments

Thanks to Bernhard Pfahringer for very helpful comments on a first draft of this paper. We also wish to thank Ivan Bratko for supplying the lymphography data. Support for the Austrian Research Institute for Artificial Intelligence is provided by the Austrian Federal Ministry for Science and Research. The second author was partially supported by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung* under grant no. M003-MED.

References

Clark, P. and Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning Journal 3(4)* (1989), 261–283.

Fikes, R.E., Hart, P.E., and Nilsson, N. (1972). Learning and Executing Generalized Robot Plans. *Artificial Intelligence 3*, 251–288.

Fisher, D. (1987). Knowledge Acquisition via Incremental Conceptual Clustering. *Machine Learning 2*, 139–172.

Gennari, J., Langley, P., and Fisher, D. (1989). Models of Incremental Concept Formation. *Artificial Intelligence 40*, 11–61.

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

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. and Krizakova, I. (1992). Forgetting and Ageing of Knowledge in Concept Formation. *Applied Artificial Intelligence 6*, pp. 193–204.

Lavrač, N., Džeroski, S. and Grobelnik, M. (1991). Learning Nonrecursive Definitions of Relations with Linus. *Proceedings of the 5th European Working Session on Learning (EWSL-91)*, Porto.

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

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

Niblett, T. (1987). Constructing Decision Trees in Noisy Domains. In Bratko, I.–Lavrač, N. (eds.) *Progress in Machine Learning*. Sigma Press, Wilmslow.

Quinlan, J.R. (1990). Learning Logical Definitions from Relations. *Machine Learning 5(3)*, 239–266.

Samuel, A.L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal 3*, No.3.

Schlimmer, J.C. and Granger, R.H. (1986). Beyond Incremental Processing: Tracking Concept Drift. *Proceedings of the AAAI'86 Conference*, Philadelphia, 502–507.

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

Torgo, L. and Kubat, M. (1991). Knowledge Integration and Forgetting. *Proceedings of the Czechoslovak Conference on Artificial Intelligence*, Prague, Czechoslovakia, June 25–27.

Widmer, G. and Kubat, M. (1992). Learning Flexible Concepts from Streams of Examples: *FLORA2. Proceedings of the 10th European Conference on Artificial Intelligence (ECAI-92)*, Vienna, 363–367.