

Learning Flexible Concepts from Streams of Examples: *FLORA2*

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

Miroslav KUBAT, Computer Center, Brno Technical University,
Udolni 19, 60200 Brno, Czechoslovakia

Abstract

FLORA2 is a program for supervised learning of concepts that are subject to concept drift. The learning process is incremental in that the examples are processed one by one. A special feature of our program consists in keeping in memory a subset of examples – a window. In time, new examples are being added to the window while other ones are considered outdated and are forgotten. In order to track the concept drift, the system keeps in memory not only valid descriptions of the concepts as they are derived from the objects currently present in the window, but also ‘candidate descriptions’ that may turn into valid descriptions in the future.

1 Introduction

One of the key tasks of the Machine Learning discipline is to find powerful methods for abstracting concepts out of a set of objects. Basically, two subproblems of this task exist: supervised learning and unsupervised learning. The former assumes that a set of pre-classified examples (positive and negative) of some concept(s) are available. The latter subproblem can be defined as a conceptual clustering problem based on a set of non-classified objects, in a way similar to the statistical cluster analysis but with stress laid on the interpretation potential.

In this paper, we present the supervised learning algorithm *FLORA2*¹, capable of learning concepts that are not constant in time but, rather, flexible in the sense that their definition varies in time. The input to our program consists of a stream of pre-classified objects. For the moment, we will assume learning from simple objects described by nominal variables (symbolic attribute-value pairs). Since the real definition of flexible concepts is subject to time-dependent changes, also the classification of the learning instances will vary in time. Experiments modelling this can be arranged in a number of ways: in particular, the changes can be abrupt or gradual, periodic or ‘random’. In the experiments (section 5) we will analyze the system’s ability to adapt to abrupt (and radical) changes in the concept meaning.

¹The acronym FLORA stands for FLOating Rough Approximation; the term rough reflects the fact that the approach was motivated by Pawlak’s *Rough Set Theory* [12]

The paper is organized as follows: After a brief review of existing work in the field of learning flexible concepts, we will describe the basic idea of our algorithm and its evolution. Then, in sections 4 and 5, we describe in detail the system *FLORA2* and the ‘disciplines’ in which it compares favourably with its ancestors.

2 Related work

A detailed analysis of some aspects of the problem of context-dependent concepts is contained in Michalski’s paper [10] where a *two-tiered representation* is suggested for modelling concepts with flexible and modifiable boundaries. In this approach, it is assumed that concepts have some general tendency represented in the base concept representation (first tier) while flexible details that may be context-dependent are best represented by appropriate matching methods and knowledge-dependent rules of inference (second tier).

Another possibility is *exemplar-based* learning (see, e.g., [1]) where the concepts are represented by several typical examples to which new objects are matched. Context-sensitive and similarity-based matching procedures enable to separate the ‘core’ knowledge about the concept from the description of possible contextual modifications.

Among the systems from the area of incremental concept formation or unsupervised learning, *COBWEB* ([2]) and its derivatives, such as *CLASSIT* ([3]), deserve attention thanks to their two complementary operators *merge* and *split*. These enable the algorithm to reconfigure the current knowledge structure based on newly observed objects. As a matter of fact, the operator *merge* can be considered as a sort of forgetting. Similar recovery operators can be found also in other concept formation algorithms such as *UNIMEM* ([9]), *FAVORIT* ([4]), etc.

Most of the above approaches tacitly assume that the concepts in question have some constant central tendency and that it is only their boundaries that are moving with context. On the other hand, Schlimmer’s system *STAGGER* ([13, 14]) is able to cope with substantial changes in the entire concept meaning. In the process of incremental learning, it searches through the space of possible descriptions and refines its concept characterizations. Each description item is accompanied by two weighting measures, describing logical sufficiency and logical necessity of the item for the general description of the concept. These two measures guide the search in the sense that (1) weak description items are deleted from the description and (2) in the search for new possible descriptions, the strongest ones are chosen.

This paper builds on previous work of one of the authors on the system *FLORA* (e.g. [5, 6, 7, 8]). The basic idea of the system is that a ‘window’ is moved over the stream of training objects. Only the objects present in the window are kept in memory. Two processes are running simultaneously over the contents of the window: *learning* from new objects that have just been added to the window, and *forgetting* older objects together with the pieces of knowledge that have been derived from them. The basic philosophy is that if the meaning of the concept changes in time, the older pieces of knowledge that have not been confirmed for a considerably long time are not to be trusted. The section gives a

at1	at2	at3	C
1	1	1	+
0	1	1	-
1	0	1	+
0	1	0	-

Table 1: Examples submitted to the learning program.

brief account of the algorithm as it was realized in the original *FLORA* system.

3 Description sets and the ‘window’

Suppose we have at our disposal a set of positive and negative examples of some concept, and a description language. The task is to find a description of the concept. Note that the concept description found is determined not only by the description language, but also by the examples submitted to the learning program. This can be illustrated by the example from Table 1 where the examples are described by means of three boolean attributes. One possible (consistent) description of the concept C that can be derived from them is $des(C) = (at1 = 1) \wedge (at3 = 1)$. However, it is possible that the real description of the concept should be $des(C) = (at1 = 1)$. Also, $des(C) = (at3 = 1)$ would describe all the positive instances (but it covers also a negative one). If we neglect consistency for the moment (which may not be our top criterion if we deal with noisy instances or drifting concepts), we have three possible concept descriptions. If we add to the table another example, the set of possible descriptions is likely to change. A negative example can contradict those descriptions that are too general, while a positive example can indicate that a description is too specialized.

The choice of the examples is rarely ideal. If the description contains many multivalued variables, a human teacher may not be able to select the most representative examples. If the examples are obtained directly from the environment, they are not necessarily the most illustrative ones. Let us further complicate the task by assuming that the concept is flexible, i.e., that its definition can change in time. This happens in many realistic domains, and can also *appear* to happen in incremental learning if the description language is poor and fails to reflect some important (hidden) feature of the concept.

These two issues – (1) a concept description depends on the selection of examples, (2) hidden features cause the concept description to drift – motivated the development of the system *FLORA*. The algorithm was first published in [5], its application was reported e.g. in [7] and a more detailed analysis in [8].

The idea is to keep in memory all possible concept descriptions. Let us forget, for the moment, about combinatorial explosion which can be later avoided by suitable heuristics. We have a set *ADES* of descriptions that have been derived from the positive examples and have not been contradicted by any of the negative examples. Further, it is useful to maintain also a set *PDES* of ‘candidate descriptions’ that cover both positive and negative

Figure 1: Transitions among the description sets.

examples ($des(C) = (at3 = 1)$ from above would fall into this category), and a set *NDES* of descriptions that cover exclusively negative examples.

Now, we have said that we are learning incrementally from a stream of examples. Since we suppose the possibility of a concept drift, we have more trust in more recent examples than in the older ones. That is why *FLORA* takes care only of a subset of examples, called ‘window’. In the simplest case (see [5]) this means: each time when a new example arrives, we will add it to the window (*learn*) and delete from the window the oldest one (*forget*). The window thus moves along the stream of examples and the changes in its contents induce changes in the contents of *ADES*, *PDES*, and *NDES*. This is illustrated by Figure 1.

In the process of *learning* either the contents of the three description sets will not be affected; or a new description will be created and added to *ADES* (*NDES*) if the example was positive (negative); or an existing description can be transferred from *ADES* (*NDES*) to *PDES*, again depending on whether the example was positive or negative. Similarly, in the process of *forgetting* either the contents of the description sets will not be affected, or transitions according to Figure 1 will take place, or some descriptions will cease to exist. A theoretical analysis of these processes for a slightly more general case of simultaneous learning from m examples and forgetting p examples can be found in [6].

4 FLORA2

4.1 Motivation

The original *FLORA* algorithm worked well in the domains for which it was designed, but exhibited considerable shortcomings when we tried to apply it to larger, more complex domains. The new system *FLORA2* was designed to repair these problems and to arrive at a robust, efficient, and yet truly flexible algorithm. Specifically, the following improvements were made:

1. *FLORA2* uses a new generalization operator to integrate new, uncovered instances into the descriptions sets *ADES* and *PDES*;
2. The description sets *ADES*, *PDES*, and *NDES* are checked for subsumption;
(1) and (2) in combination prevent explosion of the sizes of the description sets; by virtue of (1) and (2), *FLORA2* can now learn fully incrementally, i.e., starting with empty description sets;
3. *FLORA2* can automatically determine an appropriate window size and flexibly change the window size during learning; this is achieved by means of heuristics.

4.2 Incremental update of description sets

Let us assume that the description sets $ADES$, $PDES$, and $NDES$ already exist (at the beginning, they might also be empty). Remember that $ADES$ is the set of all descriptions that are *consistent* (i.e., match only positive instances), $PDES$ is a set of candidate descriptions that, taken together, are *complete*, but not consistent (i.e., it matches all positive instances, but also some negative ones), and $NDES$ is a consistent description of the negative instances seen so far (i.e., it matches no positive instances). Specifically, the description sets are of the following form:

$$\begin{aligned} ADES &= \{ADEs_1/AP_1, ADEs_2/AP_2, \dots\} \\ PDES &= \{PDEs_1/PP_1/PN_1, PDEs_2/PP_2/PN_2, \dots\} \\ NDES &= \{NDEs_1/NN_1, NDEs_2/NN_2, \dots\} \end{aligned}$$

where the $ADEs_i$ ($PDEs_i$, $NDEs_i$) are *description items* (conjunctions of descriptors), and
 $\{A, P\}P_i$... number of positive examples matching description $\{A, P\}DEs_i$
 $\{P, N\}N_i$... number of negative examples matching description $\{P, N\}DEs_i$

Each description set can be interpreted as a DNF expression. The important thing is that the system keeps counts of the number of instances matched. These numbers concern only instances that are in the current window. They are used to decide when to transfer a description to a different description set or when to drop it altogether.

We want to keep these description sets in some sense *minimal*, in order to prevent combinatorial explosion. In *FLORA2*, this is achieved by exploiting the subsumption ordering of the description space: $ADES$ is checked to contain only the *most general* descriptions consistent with positive instances (that is, if two description items $ADEs_i$ and $ADEs_j$ both are consistent with positive instances, and $ADEs_i$ subsumes $ADEs_j$, only $ADEs_i$ is kept in $ADES$). Similarly, in $PDES$, only the most specific descriptions (that cover both positive and negative instances) are kept, and $NDES$ is again kept maximally general. These conditions are checked 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 Fig.1 in section 3):

If classification C is **positive**:

For all $ADEs_i$ in $ADES$:
 if $match(instance, ADEs_i)$ then $AP_i := AP_i + 1$;
For all $PDEs_i$ in $PDES$:
 if $match(instance, PDEs_i)$ then $PP_i := PP_i + 1$;

For all $NDes_i$ 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 include into $ADES$ all possible generalizations of the new positive instance with all $ADes_j$ present in $ADES$ such that the resulting expressions are (1) maximally general and (2) do 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;

If classification C is **negative**:

For all $ADes_i$ in $ADES$:

if $match(instance, ADes_i)$ then remove $ADes_i$ from $ADES$ and include it into $PDES$ as a triple $ADes_i/AP_i/1$ and check the updated $PDES$ for subsumptions;

For all $PDes_i$ in $PDES$:

if $match(instance, PDes_i)$ then $PN_i := PN_i + 1$;

For all $NDes_i$ in $NDES$:

if $match(instance, NDes_i)$ then $NN_i := NN_i + 1$;

If there is no $NDes_i$ in $NDES$ that matches the new instance, then include into $NDES$ all possible generalizations of the new negative instance with all $NDes_j$ present in $NDES$ such that the resulting expressions are (1) maximally general and (2) do not subsume any descriptions in $PDES$ or $ADES$ (this ensures consistency against positive instances); as an extreme case, the description of the instance itself may be added to $NDES$; then check $NDES$ for subsumptions;

Incremental forgetting:

Now assume that the system decides to ‘deliberately’ forget an old training instance with known classification C . This happens when an old instance is dropped from the current window. Then the description sets are updated as follows (again, see Fig.1 in section 3):

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

For all $ADes_i$ 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 $PDes_i$ 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:

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

In summary, the strength of the algorithm lies in the explicit representation of the three description sets $ADES$, $PDES$, and $NDES$. $ADES$ represents the core of the current concept hypotheses ($ADES$ is also used for classifying new instances); $PDES$ contains *candidate descriptions* that are slightly too general: they cover some negative instances, but might be relevant in the future (when old, contradictory instances are forgotten); $NDES$ characterizes the negative instances. Together, these three sets summarize the relevant information in the training instances. There is no need to re-examine all the instances at every learning step. Also, once the learning process is well on the way, there is little need for the construction of new descriptions. Most of the action is migration of descriptions between the three sets. All this contributes significantly to the efficiency of the algorithm.

Also, note that there is no specialization operator: 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.

4.3 Dynamic window adjustment

In the original *FLORA* system, the window size was fixed (this was a parameter that had to be set before learning). However, since the behaviour of the algorithm is very sensitive to the window size, and since the optimal window size is a function of the current learning situation, this was not a very satisfactory solution. Consider just briefly what can happen when the window is not appropriate: if it is too narrow, relevant instances will be forgotten too early, and characterizations will be dropped that would still be correct and necessary. On the other hand, if the window is too large, the system is too reluctant to follow a concept drift; it keeps noisy or outdated instances/hypotheses around too long. The perfect window size would be at most as long as the periods between perceptible shifts in the (hidden) definition of the concept; at the time of a shift it should be narrowed so that more of the old instances are forgotten.

The new system *FLORA2* automatically determines and adjusts its window size during learning. Obviously, this can only be done in a heuristic manner. The heuristics that *FLORA2* uses are quite simple (and produce surprisingly good results - see section 5). Basically, the idea is to shrink the window (and forget old instances) when a concept drift seems to occur, and to keep the window size fixed when the concept seems stable. Otherwise the window should gradually grow until a stable concept description can be formed. The heuristic that tries to guess whether a concept drift is occurring is based on the following

intuition: When the concept drifts, there will be a transition period in which it is difficult to form a concise consistent concept representation (in *ADES*). As a consequence, there will be a marked decrease in the coverage ratio $\# \text{ of instances covered by } ADES / \text{ size of } ADES$. Conversely, when the coverage ratio of *ADES* is high, the window is assumed to be wide enough and is kept fixed. The window adjustment algorithm, then, looks as follows:

Let N = number of (positive) instances covered by *ADES* and

S = size of *ADES* (in terms of number of literals)

Then:

If $N/S < 1.2$ (coverage low)

and *PDES* is not empty (there are alternative hypotheses)

then decrease window size by 10% and forget the oldest instances

else if $N/S > 5$ (coverage high)

then freeze the current window size

else grow window by 1 (accommodate new instance without forgetting the oldest one)

The parameter settings 1.2 and 5 were chosen rather arbitrarily; in fact, 5 as the threshold for high coverage was our first guess, and the 1.2 for low coverage was our second. There is, of course, ample room for improvement, but these parameters already produced very good performance.

5 Experimental results

We have performed extensive experiments to test the behaviour of the algorithm. For reasons of comparison, we also applied the system to the drifting concept used by Schlimmer & Granger, namely, (1) $size = small \wedge color = red$, (2) $color = green \vee shape = circular$ and (3) $size = (medium \vee large)$ (see [14]). Figure 2 displays the predictive performance of *FLORA2* on this drifting concept. The transition of the underlying concept definition from (1) to (2) was designed to happen after 40 training instances, from (2) to (3) after 80. Training instances were generated randomly according to the hidden concept, and after processing each instance, the predictive performance was tested on 40 test instances (also generated randomly). The set *ADES* was used to classify instances.

In terms of *accuracy*, our algorithm compares very favourably with *STAGGER*. *FLORA2* converges much faster on the correct hypothesis; this seems to be a consequence of the completely symbolic knowledge representation of *FLORA2* (as opposed to *STAGGER*'s probabilistic weights). Also, the total size of the description sets remains extremely moderate (Fig.3); this is due to the algorithm's generalization operator and to the subsumption tests.

It is also instructive to look at the development of the window size (Fig.4): a change in the definition of the underlying concept first leads to a more or less short *increase* in the size of the window (marked by asterisks in Fig.4), before the system reacts to the concept shift by narrowing the window and forgetting old, now irrelevant (or even irritating) instances. This behaviour has been observed consistently in our experiments. Colloquially, one might

Figure 2: Classification accuracy.

Figure 3: Total size of *ADES*, *PDES*, and *NDES*.

say that the system, when confronted with new instances that don't quite fit its current concept hypothesis, first tries to 'gather more information' by broadening the window. However, when more and more instances contradict its hypothesis, it 'changes its mind' and attributes this to a shift in the underlying concept definition. This behaviour is a direct consequence of the window adjustment heuristics.

This observation also gives us reason to believe that the algorithm will be able to cope with noisy instances. Noisy instances, when they appear during a relatively stable phase of the hypothesis, and as long as they don't appear in big bundles, will lead to a short increase in the window but won't be mistaken for indicators of a concept shift. It is only when new instances consistently contradict the system's hypothesis (and, as a consequence, the coverage of *ADES* drops dramatically) that the system will attribute this to a concept shift and will reduce its window. In fact, if noisy (= misclassified) instances do appear in bundles, there is no reason for the system to believe that this does *not* represent a concept shift.

Figure 4: Development of window size.

6 Discussion and directions for further research

In summary, the experimental data we have to date clearly indicate that *FLORA2* can flexibly adjust itself to concept drift while remaining fast and space-economical. When we compare *FLORA2* to its closest relative, *STAGGER*, we find that our approach is indeed a success, at least as far as tracking concept drift is concerned. Our method produces crisp concept descriptions in DNF, while *STAGGER* uses a distributed representation with numeric weights attached to description items. As the above experiment indicates, *FLORA2*'s purely symbolic representation leads to faster convergence. In addition, *FLORA2*'s descriptions are also easier to understand than the distributed representation of *STAGGER*. And finally, as a side effect, *NDES* provides a concise characterization of parts of the 'negative space', which allows certain new instances to be uniquely identified as negative (much like the general boundary of a version space [11]). This may be beneficial in some domains where it is critical to identify negative situations/instances as such or where it is important to be able to predict certain attributes of negative instances.

So far, we have studied mainly the algorithm's behaviour when it is confronted with concept drift. The related problem of dealing with noise (that is, of distinguishing between random noise and a beginning drift of the concept) has not been investigated in depth. Preliminary data shows that the system is rather robust in this respect. However, more systematic analysis and experiments are needed to establish this hypothesis. We are currently performing experiments in this direction. Other threads of research we intend to pursue are investigations concerning

- more sophisticated heuristics for dynamic window adjustment
- reactions to various kinds of concept drift (gradual, abrupt, shift in relevance of attributes, etc.)
- a more sophisticated (structural) description language
- the potential of *PDES* for probabilistic classification

Since the original algorithm has been successfully applied several times, it surely deserves further research attention. There are many practical domains where concept drift does occur and where time efficiency is an issue (e.g., load balancing in computer networks [7]), so we expect our algorithm to be of substantial practical use.

References

- [1] E.R.Bareiss, B.W.Porter, C.W.Craig: Protos: An Exemplar-Based Learning Apprentice. In: Y.Kodratoff, R.S.Michalski (eds.): *Machine Learning: An Artificial Intelligence Approach, Vol. III*, Morgan Kaufmann, 1990.
- [2] D.Fisher: Knowledge Acquisition Via Incremental Conceptual Clustering. *Machine Learning* 2 (1987), 139-172.
- [3] J.H.Gennari, P.Langley, D.Fisher: Models of Incremental Concept Formation. *Artificial Intelligence* 40 (1989), 11-61.
- [4] I.Krizakova, M.Kubat: *FAVORIT*: Dynamic Approach to Concept Formation. *Pattern Recognition Letters*, 1992 (in press).
- [5] M.Kubat: Floating Approximation in Time-Varying Knowledge Bases: *Pattern Recognition Letters* 10 (1989), 223-227.
- [6] M.Kubat: Conceptual Inductive Learning: The Case of Unreliable Teachers. *Artificial Intelligence* 52 (1991), 169-182.
- [7] M.Kubat: A Machine Learning Based Approach to Load Balancing in Computer Networks. *EUROCAST'91*, Krems, Austria, April 15-19, 1991 (also to appear in the *Cybernetics and Systems Journal*, 1992).
- [8] M.Kubat: Flexible Concept Learning in Real Time Systems. *Journal of Intelligent and Robotic Systems*, 1992 (in press).
- [9] M.Lebowitz: Experiments with Incremental Concept Formation: *UNIMEM*. *Machine Learning* 2 (1987), 103-138.
- [10] R.S.Michalski: Learning Flexible Concepts: Fundamental Ideas and a Method Based on Two-Tiered Representation. In Y.Kodratoff, R.S.Michalski (eds.): *Machine Learning: An Artificial Intelligence Approach, Vol. III*, Morgan Kaufmann, 1990.
- [11] T.M.Mitchell: Generalization as Search, *Artificial Intelligence* 18(2) (1982), 203-226.
- [12] Z.Pawlak: Rough Sets. *International Journal of Computer and Information Sciences* 11 (1982), 341-356.
- [13] J.C.Schlimmer, R.H.Granger: Beyond Incremental Processing: Tracking Concept Drift. *Proceedings of the AAAI'86 Conference*, Philadelphia, 1986, 502-507.
- [14] J.C.Schlimmer, R.H.Granger: Incremental Learning from Noisy Data. *Machine Learning* 1 (1986), 317-354.