Machine Learning, 27, 259–286 (1997)

Tracking Context Changes through Meta-Learning

GERHARD WIDMER gerhard@ai.univie.ac.at

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

Editors: Ryszard S. Michalski and Janusz Wnek

Abstract. The article deals with the problem of learning incrementally ('on-line') in domains where the target concepts are context-dependent, so that changes in context can produce more or less radical changes in the associated concepts. In particular, we concentrate on a class of learning tasks where the domain provides explicit *clues* as to the current context (e.g., attributes with characteristic values). A general two-level learning model is presented that effectively adjusts to changing contexts by trying to detect (via 'meta-learning') contextual clues and using this information to focus the learning process. Context learning and detection occur *during* regular on-line learning, without separate training phases for context recognition. Two operational systems based on this model are presented that differ in the underlying learning algorithm and in the way they use contextual information: METAL(B) combines meta-learning with a Bayesian classifier, while METAL(IB) is based on an instance-based learning algorithm. Experiments with synthetic domains as well as a number of 'real-world' problems show that the algorithms are robust in a variety of dimensions, and that meta-learning can produce substantial increases in accuracy over simple object-level learning in situations with changing contexts.

Keywords: Meta-learning, on-line learning, context dependence, concept drift, transfer

1. Motivation

The fact that concepts in the real world are not eternally fixed entities or structures, but can have a different appearance or definition or meaning in different contexts has only gradually been recognized as a relevant problem in concept learning. Michalski (1987) was one of the first to formulate it; he suggested a specialized *two-tiered representation* formalism to represent different aspects of context-dependent concepts (see also Bergadano et al., 1992). Recently, context dependence has been recognized as a problem in a number of practical machine learning projects (e.g., Katz et al., 1990; Turney, 1993; Turney & Halasz, 1993; Watrous, 1993; Watrous & Towell, 1995; see also Kubat & Widmer, 1996). There, various techniques for context handling were developed. Most of these methods either assume that contextual attributes are explicitly identified by the user, or require separate pre-training phases on special data sets that are cleanly separated according to context.

We are studying the effects of context dependence and changing contexts in the framework of *incremental* (or *on-line*) learning, and we are interested in learners that can adapt to different contexts without explicit help from a teacher. The scenario is as follows: assume that a learner is learning on-line from a stream of incoming (labeled) examples. Assume further that the concepts of interest depend on some (maybe hidden) *context*, and that changes in this context can induce corresponding changes in the target concepts. As a simple example, consider weather prediction rules, which may vary drastically with the change of seasons. The visible effects of such changes are increased prediction error rates.

The development of on-line learners that can cope with concept drift and changing contexts has been the subject of recent work on the FLORA family of algorithms (Kubat, 1989; Widmer & Kubat, 1996). The basic strategy in the FLORA algorithms is to continually monitor the success of on-line prediction and to make educated guesses at the occurrence of context changes and corresponding concept changes. There is no explicit representation of contexts.

But maybe one can do better. In some domains, the data may in fact contain explicit *clues* that would allow one to identify the current context, if one knew what these clues are. Technically, such clues would be attributes or combinations of attributes whose values are characteristic of the current context; more or less systematic changes in their values might then indicate a context change.

As a simple example, consider the license plates attached to vehicles in a particular country. An agent crossing the border between, say, Austria and Germany might notice that all of a sudden the license plates look different, in a systematic way, and that might lead it to suspect that it is now in a different environment where some of the rules it had learned before may not be valid any more. Many other examples of such context-defining factors come to mind (e.g., climate or season in weather prediction, environmental conditions like exterior temperature in technical process control tasks, lighting conditions or background color in automatic vision, characteristics of particular rooms in indoor robotic tasks, or speaker nationality and sex in speech processing). In the following, we will refer to such context-defining attributes as *contextual clues*.

In this article, we describe a general two-level learning model, and its realization in two specific systems named $\mathrm{METAL(B)}$ and $\mathrm{METAL(IB)}$, that can *learn* to *detect* such contextual clues, and can react accordingly when a change in context is suspected. The model consists of a *base level* learner that performs the regular on-line learning and classification task, and a *meta-learner* that tries to identify attributes and features that might provide contextual clues. Context learning and detection occur *during* regular on-line learning, without separate training phases for context recognition. Perceived context changes are used to focus the on-line learner specifically on information relevant to the current context. The result is faster adaptation to changed concept definitions, and generally an increase in predictive accuracy in dynamically changing domains. In the following presentation, we will first assume only nominal (discrete) attributes, but extensions of the model to numeric domains are quite straightforward and have in fact been implemented (see Section 7.3).

The structure of this article is as follows: Section 2 briefly introduces the basic notions underlying Bayesian classifiers, as these will play an important role throughout the paper. Section 3 then develops the central definitions on which our methods rest, and shows how these definitions can be turned into operational procedures. Section 4 presents the general meta-level learning architecture and our first realization of the architecture in a system named $\mathrm{METAL(B)}$. Various aspects of $\mathrm{METAL(B)}$ and meta-learning in general are studied in a set of systematic experiments in Section 5. Section 6 describes $\mathrm{METAL(IB)}$, an alternative realization of the meta-learning model. The two systems are tested and compared on both synthetic data and a number of more realistic problems in Section 7. Related work is discussed in Section 8, and Section 9 summarizes the main points of the paper.

2. Preliminaries: Bayesian Classifiers

For the moment, let us assume that our basic incremental induction algorithm is a *simple* (or *naive*) *Bayesian classifier* (as is indeed the case in the first of the systems to be presented below, METAL(B)). That will make it easier to explain the meta-level learning strategy, which also has a distinct Bayesian flavor.

A *simple Bayesian classifier* is a probabilistic classification scheme that uses Bayes' theorem to determine the probability that an instance belongs to a particular class, given the instance's description. In the following, we assume that examples are described in terms of (discrete) *attributes* a_i; we will use the term *feature* for a specific attribute-value combination, notated as $a_i : v_{ij}$. Examples are assumed to belong to mutually exclusive classes c_i. Bayes' theorem defines the posterior probability that some new instance I belongs to class c_i as

$$p(c_i|I) = \frac{p(c_i)p(I|c_i)}{p(I)}$$

where $p(c_i)$ is the prior probability of class c_i and $p(I|c_i)$ is the probability of an instance like I, given class c_i. Assuming that I is a conjunction of attribute values v_j, the above formula can be rewritten as

$$p(c_i|\bigwedge v_j) = \frac{p(c_i)p(\bigwedge v_j|c_i)}{\sum_k p(\bigwedge v_j|c_k)p(c_k)}$$

To make this formula operational, one usually assumes that the attributes are independent, so that $p(\bigwedge v_j|c_k)$ can be computed as the product of the $p(v_j|c_k)$.

Incremental induction of a Bayesian classifier is straightforward. One maintains a number of counters, from which the prior and conditional probabilities can be estimated: a count N of the total number of instances encountered so far, a table C_i that keeps track of the relative frequency of class c_i observed so far; a table AV_{ij} that records the number of examples with attribute value $a_i = v_{ij}$, and a table AVC_{ijk} that records the number of examples with $a_i = v_{ij}$ belonging to class c_k. Learning then simply consists in updating these counters after processing each instance. The algorithm is simple and naturally incremental. Its major weakness is that we must assume independence of the attributes, which severely limits the class of learnable concepts (but see Section 5.5 below).

In the following, we take this to be our basic incremental learning algorithm, with one important modification: our learner maintains a *window* of fixed length. As new examples are added to the window, the oldest ones are deleted from it if the window size is exceeded. This is to ameliorate the problem that very old instances pertaining to an outdated context may prevent the learner from effectively adjusting to new hypotheses. The window size is a user-settable parameter, but it remains fixed during the entire learning process. The tables C_i, AV_{ij}, and AVC_{ijk} are always updated with respect to the examples in the current window.

3. Meta-Learning: Learning to Recognize Contextual Clues

When the underlying target concept drifts or changes due to a changed context, the Bayesian classifier (indeed, any induction algorithm that bases its hypotheses on the contents of the window) will eventually adjust to the new concept, if the new context is stable for a sufficiently long period. The smaller the window, the faster the adjustment, as old, contradictory examples will be forgotten more quickly. However, in domains that provide explicit context clues, one would expect more: the learner should learn to recognize such clues and react in some appropriate way when they signal a potential context change. To operationalize this goal, we first need to define the central notions.

3.1. Definitions

What are contextual clues? Turney (1993) was one of the first to explicitly acknowledge the problem of context in learning and to try to give a formal definition of contextual and context-sensitive features. Eventually, however, he relied on the user to identify contextual features beforehand. His particular approach was motivated by batch learning problems where the *testing examples* (i.e., those to which the learned concepts would eventually be applied) might be governed by a different context than the training examples from which the concepts were learned. The contextual features were then used for different kinds of normalization at prediction time. In contrast, we present a method to automatically *detect* contextual attributes in an on-line learning setting and to utilize this information *during learning*.

Our operational definition of contextual attributes, i.e., attributes that provide contextual clues, is based on the notion of *predictive features*. Intuitively speaking, an attribute is predictive if there is a certain correlation between the distribution of its values and the observed class distribution. This is formalized in the following two definitions:

Definition 1 *(Predictive features).* A feature (attribute-value combination) $a_i : v_{ij}$ is *predictive* if $p(c_k \mid a_i = v_{ij})$ is significantly different from $p(c_k)$ for some class c_k.

Definition 2 *(Predictive attributes).* An attribute a_i is *predictive* if one of its values v_{ij} (i.e., some feature $a_i : v_{ij}$) is predictive.

The most obvious kind of contextual clue one could imagine is that one or more attributes would have constant values during a certain context (regardless of an instance's class). Think, for instance, of the color of the walls in a particular room. That cannot be expected in general. We will rely on a more abstract and powerful notion: a feature is considered a contextual clue if there is a strong correlation between its temporal distribution of values and the times when certain other attributes are predictive. Intuitively, a contextual attribute is one that could be used to predict which attributes are predictive at any point in time.[1] This notion is formalized in definitions 3 and 4:

Definition 3 *(Contextual features)*. A feature $a_i : v_{ij}$ is *contextual* if it is predictive of predictive features, i.e., if $p(a_k : v_{kl}\ is\ predictive \mid a_i = v_{ij})$ is significantly different from $p(a_k : v_{kl}\ is\ predictive)$ for some feature $a_k : v_{kl}$.

Definition 4 *(Contextual attributes)*. An attribute a_i is *contextual* if one of its values v_{ij} is contextual.

We thus have a two-level definition of contextual attributes, with both levels of definition being of the same type. Definition 2 (*predictive attributes*) is identical to Turney's (1993) notion of *primary feature*. Definition 4 (*contextual attributes*) is more specific and operational than Turney's definition of *contextual features* (which essentially defines an attribute as contextual if it is not in itself predictive, but would lead to less predictive accuracy if omitted)[2]. We now specify procedures to identify potentially predictive and contextual features and attributes during the incremental learning process.

3.2. Identifying contextual features through meta-learning

Assume the base-level Bayesian classifier is learning on-line from a stream of incoming examples. After each learning step, we use a *statistical χ^2 test of independence* to determine which features are currently *predictive*:

Criterion 1 *(Predictive features)*: A feature $a_i : v_{ij}$ is recognized as *predictive* if the distribution of classes in examples with $a_i = v_{ij}$ is significantly different (as determined by a χ^2 test with a given significance level) from the unconditioned distribution of classes within the current window.

Predictive features are computed relative to the *current window* because predictivity is a temporary quality that may change with time and context. The information needed for the χ^2 test is readily available in the tables C_i and AVC_{ijk} that are maintained by the base-level learner.

Contextual features are also determined by a χ^2 test, on a higher level. To this end, we define 'meta-classes' $\hat{c}_{ij}$: an instance I is in class $\hat{c}_{ij}$ if feature $a_i : v_{ij}$ is recognized as predictive at the time of classification of I. Analogously to above, tables are maintained for these meta-classes: the table $\hat{C}_{ij}$ counts the number of examples in meta-class $\hat{c}_{ij}$, $\hat{AV}_{ij}$ counts the number of examples with attribute value $a_i = v_{ij}$, seen since the very beginning, and $\hat{AVC}_{ijkl}$ counts the number of examples with $a_i = v_{ij}$ in meta-class $\hat{c}_{kl}$. In other words, $\hat{AVC}_{ijkl}$ keeps track of the number of co-occurrences of $a_i : v_{ij}$ combinations in examples and the predictiveness of certain other features $a_k : v_{kl}$. These three tables are maintained with respect to the entire learning history (not the current window), as changes of context and the emergence of skewed predictivity distributions are long-term processes. Table 1 summarizes the various tables that need to be maintained. There are then two conceivable operational criteria by which one could detect contextual features and attributes:

Criterion 2a *(Contextual features)*: A feature $a_i : v_{ij}$ is recognized as *contextual* if the distribution of meta-classes $\hat{c}_{kl}$ in examples with $a_i = v_{ij}$ is significantly different (as

Table 1. Tables maintained for identifying predictive and contextual attributes.

Table	Counts occurrences/rel.frequency of	Computed over	Used at
C_i	# examples in class c_i	current window	base-level
AV_{ij}	# examples with $a_i = v_{ij}$	current window	base-level
AVC_{ijk}	# examples with $a_i = v_{ij}$ in class c_k	current window	base-level
$\hat{C}_{ij}$	# examples in meta-class $\hat{c}_{ij}$	entire history	meta-level
$\hat{AV}_{ij}$	# examples with $a_i = v_{ij}$	entire history	meta-level
$\hat{AVC}_{ijkl}$	# examples with $a_i = v_{ij}$ in meta-class $\hat{c}_{kl}$	entire history	meta-level

determined by a χ^2 test with a given significance level) from the unconditioned distribution of the $\hat{c}_{kl}$, observed over the entire learning history.

Criterion 2b *(Contextual features)*: A feature $a_i : v_{ij}$ is recognized as *contextual* if, for *some* feature $a_k : v_{kl}$, the distribution of meta-class $\hat{c}_{kl}$ versus $\overline{\hat{c}_{kl}}$ in examples with $a_i = v_{ij}$ is significantly different (as determined by a χ^2 test with a given significance level) from the unconditioned distribution of $\hat{c}_{kl}$ versus $\overline{\hat{c}_{kl}}$, observed over the entire learning history.

Criterion 2a pays attention to global distribution changes between the predictivity of different features, while criterion 2b is basically a direct translation of definition 3 above: $a_i : v_{ij}$ is contextual if its values correlate with the predictivity of *some* other feature $a_k : v_{kl}$. After some experimentation with both approaches, we have settled for criterion 2b (though criterion 2a yields very similar results in most cases).

Recognizing contextual features and attributes via this two-stage process constitutes an act of *meta-learning*: the base-level learning process is monitored, and the temporal coincidence of predictivity of certain features and the presence of other features in training instances leads to the identification of attributes that could provide contextual clues. The contextual features are taken as a description or *identifier* of the current context. In the following, we present two learning systems with different underlying learning algorithms that take advantage of the information provided by meta-learning.

4. A Bayesian Classifier with Meta-Learning: METAL(B)

Our first algorithm is called METAL(B) (METALearning with underlying Bayes classifier) and uses a simple Bayesian classifier as its underlying ('object-level') incremental learning algorithm. It was first presented in (Widmer, 1996).

In METAL(B), the contextual attributes identified by meta-learning are used to *focus* the base-level Bayesian classifier on relevant examples when making predictions: whenever a new instance comes in, the set of attributes that are currently contextual (if any) is established, and the Bayesian classifier is then made to use for prediction only those examples from the window that have the same values for the contextual attributes as the new instance to be classified. In other words, the base-level classifier uses only those instances as a basis for prediction that seem to belong to the *same context* as the incoming example. If no

Procedure METAL(B)(W, α):
/* *Parameters:* W (fixed window size), α (significance level for χ^2 test). */
initialize tables C_i, AV_{ij}, AVC_{ijk}, $\hat{C}_{ij}$, $\hat{AV}_{ij}$, $\hat{AVC}_{ijkl}$;
for each new incoming instance I do
begin
CAs := current_context_attributes($\hat{C}_{ij}$, $\hat{AV}_{ij}$, $\hat{AVC}_{ijkl}$, α);
Vs := values of attributes *CAs* in current instance I;
RIs := examples in current window with values *Vs* for attributes *CAs*;
Class := class predicted for I by naive Bayesian classifier from examples *RIs*;
add I to current window and drop oldest instance from window;
update tables C_i, AV_{ij}, AVC_{ijk} for base-level Bayesian learning;
PFs := currently_predictive_features(C_i, AV_{ij}, AVC_{ijk}, α);
update tables $\hat{C}_{ij}$, $\hat{AV}_{ij}$, $\hat{AVC}_{ijkl}$ for meta-level learning, given *PFs*
end;

Figure 1. The two-level algorithm of METAL(B).

attributes are currently recognized as contextual, the entire window is used for Bayesian classification. After classification, the true class of the new instance is read, and the learning tables for both base and meta-level are updated. Figure 1 summarizes the complete two-level algorithm of METAL(B).

A consequence of this selective strategy is that base-level prediction becomes more expensive: the Bayesian classifier can no longer use the available tables C_i, AV_{ij}, and AVC_{ijk}, as these summarize all examples from the window. The relative frequencies required for the application of Bayes' rule must now be computed dynamically from the set of currently relevant instances (which are a subset of the window). On the other hand, this is no more expensive than the lookup in an instance-based learning algorithm (Aha et al., 1991), and the cost could be reduced by using an appropriate indexing scheme.

Note that METAL(B) depends on two *parameters*. Unless otherwise noted, all experiments reported below were performed with α=0.01 (99% confidence that the observed difference between conditioned and unconditioned distributions is not due to chance) and window size $W = 100$.

5. Preliminary Experiments with METAL(B)

Before moving on to an alternative realization of our general meta-learning model, let us first have a look at a set of experiments with a synthetic domain that were designed to establish the basic feasibility of context detection and to test various aspects of METAL(B)'s behavior. They provide a number of insights into the dynamics of meta-learning, most of which also carry over to our second learner, METAL(IB), to be described in the next section.

Experiments with real data as well as a comparison with METAL(IB) will be presented in Section 7.

5.1. Basic context identification

The first experiment demonstrates the basic effects of the context detection mechanism. METAL(B) was applied to a sequence of simple concepts that were first introduced by Schlimmer and Granger (1986) to test their concept drift tracking algorithm STAGGER. The concepts were later on also studied extensively in experiments with the FLORA algorithms (Widmer & Kubat, 1996). In a simple blocks world defined by three nominal attributes $size \in \{small, medium, large\}$, $color \in \{red, green, blue\}$, and $shape \in \{square, circular, triangular\}$, we define a sequence of three target concepts (1) $size = small \wedge color = red$, (2) $color = green \vee shape = circular$ and (3) $size = (medium \vee large)$. The (hidden) target concept will switch from one of these definitions to the next at certain points, creating situations of extreme concept drift. In addition, we introduce a fourth attribute $ctxt \in \{c1, c2, c3\}$, which is used to create perfect contextual information: whenever concept (1) is in effect, all examples (positive and negative) are made to have value $ctxt = c1$, etc.

Figure 2 plots the on-line prediction accuracy of METAL(B) vs. the simple Bayesian classifier on the concept sequence 1-2-3-1-2-3. Sequences of 600 examples each were generated randomly (according to a uniform distribution) and labeled according to the currently ruling concept; after every 100 instances the context plus underlying concept was made to change. On-line accuracy was computed as a running average over the last 20 predictions. The figure plots the averages over ten (paired) runs.

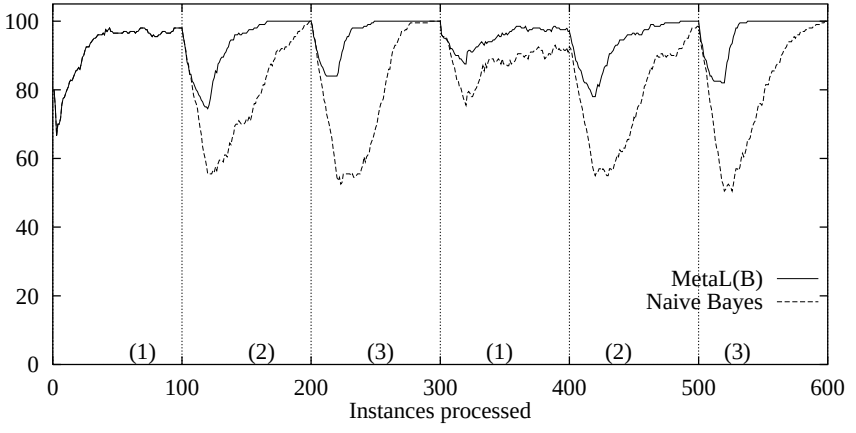

Figure 2. Effect of context identification in STAGGER concepts.

The curves show convincingly that METAL(B) does make effective use of the contextual information contained in the data. We witness both a less dramatic drop in accuracy

at points of context change, and significantly faster re-convergence to high accuracy levels. Obviously, METAL(B) quickly identifies the contextual attribute $ctxt$ (soon after the context has first switched from 1 to 2) and from then on concentrates only on examples pertaining to the new context, whereas the naive Bayes algorithm gives equal consideration to all examples in the current window, including those that still pertain to the old context. The fact that both algorithms fail to reach an accuracy of 100% in context (1) is due to the sparsity of the concept (only 11% of the examples are positive). The improvement produced by meta-learning, however, is evident.

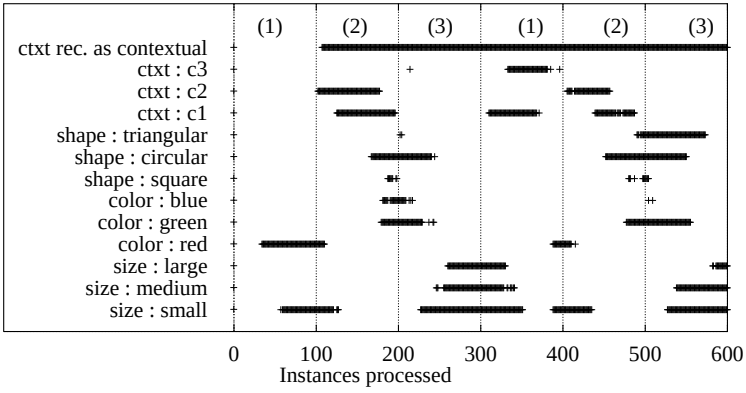

Figure 3. Predictive features and contextual attribute in STAGGER experiment.

To understand more precisely what is going on, it is instructive to look at the details of which features are identified as predictive and contextual, and when. Figure 3 shows, for each feature, at what times it was considered predictive by METAL(B) in a single run. The topmost bar indicates when the attribute $ctxt$ was recognized as contextual. No other attribute was ever considered contextual in the entire run.

The identification of predictive attributes works basically as expected: of the 'base-level' attributes (the ones used in our hidden concept definitions), the two most relevant $color$ and $size$ features are recognized as predictive about midway during context (1), the $color$ and $shape$ features during context (2), and $size$ during context (3).[3] One can observe a certain inertia in this recognition process. It takes a while for predictive features to establish themselves, and they are considered predictive way into the following context. This is simply an effect of the fixed window (of 100 instances, in this case) from which predictivity is computed. However, that has little effect on the efficiency of the usage of context information: it may take a while for contextual attributes to be first recognized, but once they are established, they immediately lead to selective behavior when the context changes.

Interestingly, the attribute $ctxt$ is also recognized as predictive, and that creates a surprising effect: $ctxt$ is recognized as contextual because of itself becoming suddenly predictive! That $ctxt$ is identified as predictive in context (2) is easily explained by the observation

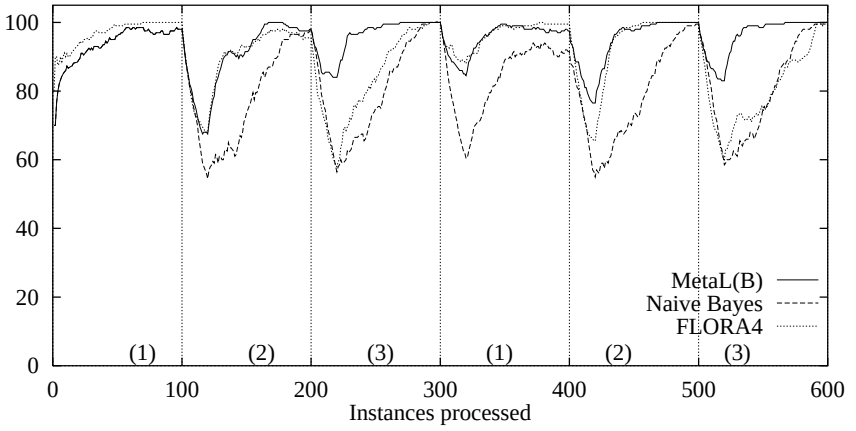

Figure 4. METAL(B) vs. FLORA4.

that the class distribution is very different in the first two contexts: positive examples in context (1) are much rarer than positive examples in context (2), because concept (1) ($size = small \wedge color = red$) is very specific — only 11% of the possible $size$-$color$-$shape$ combinations are instances of concept (1). So the feature $ctxt = c2$ in the examples during context (2) is accompanied by a class distribution that is very different from when $ctxt = c1$, which makes $ctxt$ a predictive attribute (see definition 2 in Section 3.1). At the same time, the predictiveness of $ctxt$ is accompanied by constant values of $ctxt = c2$, whereas $ctxt$ was not predictive while $ctxt$ was $c1$ in context (1), which makes $ctxt$ also a contextual attribute (definitions 3 and 4). The status of $ctxt$ as a contextual attribute is later corroborated by the changing temporal distribution of the other predictive features.

This extreme change in class distribution made it very easy to establish $ctxt$ as contextual. In order to remove this effect in the other experiments, in all the subsequently reported tests with synthetic domains we made sure that the absolute distribution of classes would be the same in each context (50% positive, 50% negative).

5.2. Comparison with FLORA4

Let us now take a quick look at how METAL(B) compares to another recent drift tracking system. FLORA4 (Widmer, 1994) is the most recent member of the FLORA family of algorithms (Widmer & Kubat, 1996). FLORA4 is based on an incremental, symbolic learning algorithm. Like METAL(B), it maintains a time window; the window size is not kept fixed, however, but is continually adjusted by a sophisticated heuristic that monitors the learning process and tries to recognize periods of concept drift. On the other hand, FLORA4 cannot take advantage of contextual clues in the data, it only monitors the learner's success in prediction.

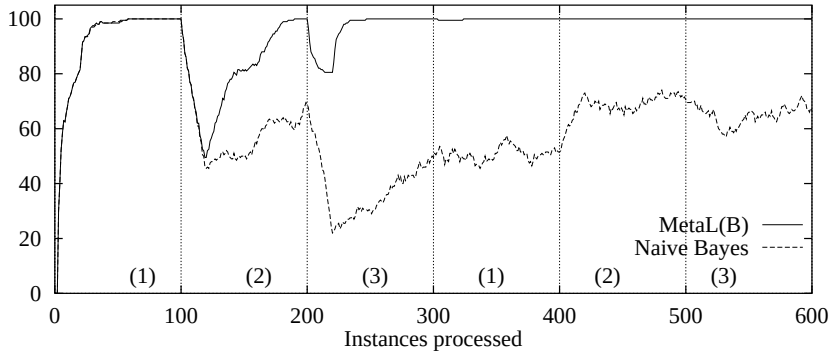

Figure 5. METAL(B) on STAGGER concepts — window size 300.

Figure 4 shows the accuracy curves for the simple Bayesian classifier (with fixed window), METAL(B), and FLORA4, on the same task as above (except that the classes are now equally distributed). This simple experiment does not support general conclusions, but it does allow us to make a number of preliminary observations; results of a more systematic comparison that support these observations are given in Tables 2 and 3 in Section 7.1. As expected, the basic behavior of the algorithms is similar. FLORA4's ability to recognize concept drift and adjust its window allows it to adapt to changes significantly faster than a simple (Bayesian) learner with fixed window. However, it does not 'see' the contextual clue $ctxt$. METAL(B), after having identified $ctxt$ as a context attribute in context (2), clearly outperforms FLORA4. Conversely, this result shows that if there were no contextual clues in the data, FLORA4 would be the algorithm of choice. Combining FLORA4's drift detection ability with METAL(B)'s meta-learning might thus be a promising enterprise.

One must keep in mind that FLORA4's object-level generalization algorithm is very different from METAL(B)'s; it is difficult to separate out the contributions of the underlying learning algorithm and of meta-learning. Comparisons such as this are only meant to show that METAL(B) is competitive with other drift tracking algorithms. The main point of this paper is that simple learning can be improved by meta-learning, and that is what all the following experiments will attempt to establish.

5.3. The effect of the system parameters

METAL(B) depends on two parameters: the significance level α used in the χ^2 tests at two levels, and the fixed *window size*. A level of $\alpha = 0.01$ has produced good results in all our experiments. Less strict significance levels make the meta-learner less discriminating: it becomes more easily possible for features to be accidentally 'recognized' as predictive for short periods, which causes frequent changes in the distribution of predictive features. The effect is instability. On the other hand, tighter significance levels (e.g., $\alpha = 0.001$) have left our results virtually unchanged.

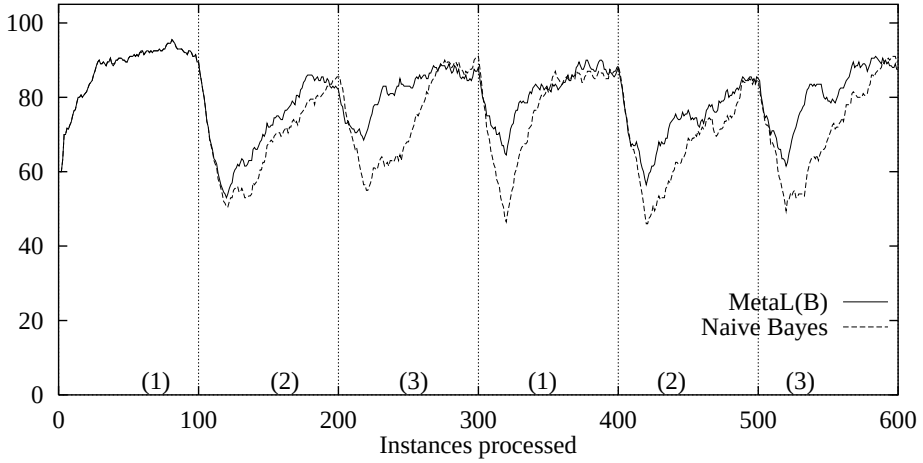

Figure 6. STAGGER concepts with 20% attribute noise.

As for the *window size*, the smaller the window, the faster the base-level learner will adapt to changes, and thus the smaller the advantage afforded by meta-learning. Too narrow a window is detrimental, as the Bayesian classifier bases its predictions on too few data points. Too large a window, on the other hand, reduces the base-level learner's ability to adjust to changes and, if it permanently contains conflicting instances from different contexts, may prevent the learner from ever reaching high predictive accuracy. Figure 5 plots the results on the STAGGER task when the window size is set to 300. $\mathrm{METAL(B)}$ detects the contextual attribute somewhere during context (2) and uses its values to always select the correct examples for prediction. After the first 300 examples, the window always contains instances from all three contexts, and $\mathrm{METAL(B)}$ can perfectly predict from then on. For the same reason, the simple Bayesian classifier fails completely.

However, too large a window may also cause problems for $\mathrm{METAL(B)}$: in a large set of partially conflicting instances, it may become impossible to find predictive features according to our definition, and then meta-learning will never find contextual attributes, even if they are very clear. Generally, then, the window should be large enough for stable concept identification, but no larger than necessary.

5.4. Irrelevant attributes and noise

Bayesian learners can be expected to be quite robust in the face of irrelevant attributes and noise (see, e.g., Langley et al., 1992). Experiments with $\mathrm{METAL(B)}$ have confirmed this, both for object-level and meta-level learning. As an illustrative example, Figure 6 plots the performance on the STAGGER task when the examples were subjected to 20% attribute noise. Here, a noise level of $\eta\%$ means that for each attribute in a training example, its

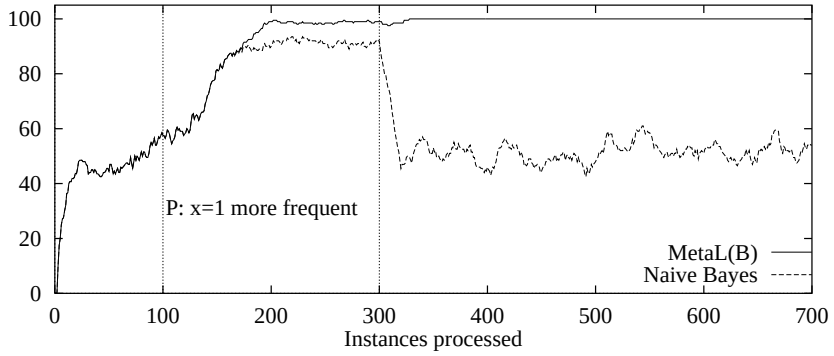

Figure 7. "Quasi-contextual" learning: XOR with 5 irrelevant attributes.

true value is replaced by a random value from its domain with probability $\eta/100$. Both the 'base-level' attributes $color$, $size$, and $shape$ and the context attribute $ctxt$ were equally subjected to noise. It is evident that meta-learning still leads to improved adjustment to changes, though such high noise levels will eventually produce effects of instability, as evidenced by the fact that in some cases the simple Bayesian learner achieves slightly higher accuracy. Tables 2 and 3 in Section 7 below give a summary of the results for various noise levels and numbers of irrelevant attributes and a comparison with our second learner, $\mathrm{METAL(IB)}$.

5.5. "Quasi-contextual learning"

An interesting side-effect of meta-learning is that it may in certain cases help the Bayesian learner to overcome its inherent representational limitations, incurred by the attribute independence assumption. As an example, consider the XOR function: in the concept $x = 1 \oplus y = 1$, neither of the two attributes x and y in isolation contributes directly to the classification. A Bayesian classifier will always linger around the base-level accuracy of 50%, given a set of uniformly distributed examples. The same holds for $\mathrm{METAL(B)}$, as long as the examples are presented in random order. However, if for some time examples appear in a skewed distribution, meta-learning may exploit this by learning to regard one of the attributes as contextual and the other as predictive. This two-level view of the XOR function would then allow the system to perfectly classify from that point on: if context is $x = 1$, then $y = 0$ implies XOR, and vice versa.

Figure 7 shows the learners on sequences of XOR examples with five additional irrelevant attributes, where during a certain period *P* (between the 100th and the 300th instance), examples — both positive and negative — with $x = 1$ were more frequent than those with $x = 0$ (90% vs. 10%). Before example 100 and after example 300, instances were presented in a uniform distribution. The results are again averages over ten runs.

The effect is very clear: METAL(B) does indeed single out x as the contextual attribute at some point during period *P*, which allows it to reach an accuracy level of (almost) 100% quickly, and also to hold on to this level during the following random period. The simple Bayesian classifier improves its performance between examples 100 and 300 (it can never reach 100%), but quickly drops to 50% as the instances are again uniformly distributed.[4]

This result highlights the close relation between context-sensitive learning and learning disjunctive concepts. Context-specific versions of a concept can be viewed as disjuncts in a large 'universal' concept definition expressed in disjunctive normal form (DNF), with the context attributes acting as 'keys' that identify which disjunct is currently active.[5] Conversely, if during the presentation of a disjunctive concept there are periods of skewed example distributions, meta-learning could learn to regard different disjuncts as representatives of different contexts and could thus improve the discriminating power of an underlying (e.g., Bayesian) learning algorithm that has fundamental problems with disjunctive concepts.

5.6. Abrupt vs. gradual vs. no context changes

Other experiments we have performed confirm that METAL(B) is rather insensitive to the *speed* of a context change, i.e., whether a new context takes over abruptly or only gradually. Unlike systems like FLORA4 (Widmer, 1994), METAL(B) has no problems with slow drift — it is the overall distributions of the predictivity of features that determine METAL(B)'s behavior, not necessarily dramatic short-term changes. That is important, because many realistic applications are presumably characterized by more or less gradual (and noisy) context changes.

A related point is whether the meta-learning mechanism has a detrimental effect when there is *no* concept drift at all. Intuitively, one would expect that not to be the case: in a stationary environment with stable target concepts, the set of predictive features will remain constant, so no attributes will ever appear to be contextual. There are, however, two types of situations that might mislead METAL(B) into erroneously 'detecting' context clues: a skewed ordering of training instances, and the presence of a large number of *irrelevant attributes*. The former case is a natural consequence of the relation between contexts and disjunctive concepts that was demonstrated and discussed in the previous section. The latter case presents a problem: large numbers of irrelevant (random) attributes increase the probability of *chance correlations* which might start the whole meta-learning machinery and lead to instabilities in the learning process. That has been confirmed experimentally with various stationary concepts in the STAGGER domain; the addition of 50 irrelevant attributes led to performance decreases between two and ten percentage points. Using stricter values (e.g., 0.001) for the significance level α alleviates the problem significantly.

5.7. Complex and imperfect contextual clues

In real life, contexts may be defined by more complex combinations of features, and contextual attributes may be changing. Preliminary experiments suggest that METAL(B)

has no problems with *conjunctively* defined contexts. Contexts defined by *disjunctions* of features create problems, especially when the contextual attributes are not the same in different contexts. The main problem is that contextual information is used conjunctively in METAL(B)'s focusing strategy, which makes the base-level learner rely on very few examples in some cases. Generally, of course, the meta-learner suffers from the same limitation as the base-level Bayesian classifier: it must assume independence between *contextual attributes*.

Another potential complication is that contexts may not always be characterized by perfectly constant values. Experiments with noise in the context attributes suggest that METAL(B) is robust to that kind of imperfection, but more refined experiments may be needed to study other types of imperfect clues (e.g., changing value distributions).

6. An Alternative Realization of the Model: METAL(IB)

METAL(B) is but one possible incarnation of a very general framework. It seemed natural and elegant to use a Bayesian classifier for object-level learning, because metalearning as defined here also has a distinct Bayesian flavor. However, it should be possible in principle to use any other incremental induction algorithm for base-level learning. This section presents such an alternative realization of the general model: METAL(IB) (META-Learning with underlying Instance-Based classifier) realizes the same meta-learning strategy as METAL(B), but uses a simple *instance-based* learning algorithm (Aha et al., 1991) as base-level learner. New incoming examples are classified by a straightforward 1-NN (nearest-neighbor) method, using Euclidean distance as the (dis)similarity measure. Again, METAL(IB) maintains a *window* of fixed size, and only examples in the current window are used in the nearest-neighbor search.

An instance-based learner allows us to use context information in a more flexible way, e.g., for feature weighting, which is not directly possible in METAL(B)'s Bayesian classifier. Also, instance-based learners can in principle approximate any target function, while Bayesian classifiers are severely limited. In the following, we present four variants of METAL(IB) that differ in the way they use the information about suspected contextual clues:

1. **Exemplar Selection —** METAL(IB)–ES**:**
   Here, contextual information is used in the same way as in METAL(B), namely, to *select relevant exemplars* during classification: only those instances from the window are used for prediction that have the same values for the current context attributes (i.e., presumably belong to the same context) as the current instance (the one to be classified).

2. **Exemplar Weighting —** METAL(IB)–EW**:**
   In this variant, contextual clues are used for exemplar *weighting* rather than strict selection. When classifying by nearest neighbor, each instance (*exemplar*) in the current window is assigned a weight W, and the measured similarity between new instance and exemplar is multiplied by W. The idea is that exemplars that are more likely to belong to the same context as the new instance should have more influence in classification.

Maximum weight should be given to those exemplars that perfectly match the current context, i.e., that share the values for all current context attributes with the current instance, and generally the weight should decrease correspondingly as this match grows weaker. METAL(IB)–EW uses the following formula to compute exemplar weights:

$$W(E) = 1/(1 + D_{|CAs|}(E, CI))$$

where $D_{|CAs|}$ is the distance between the exemplar E and the current instance CI, measured only with respect to the context attributes CAs.

3. **Feature Weighting —** METAL(IB)–FW**:**
This approach is somewhat orthogonal to exemplar weighting. Instead of weighting the examples in the window, METAL(IB)–FW uses a weighted similarity measure that assigns different importance to individual attributes during nearest neighbor classification. Features or attributes believed to be *predictive in the current context* should receive correspondingly higher weights. To this end, the meta-level algorithm must be augmented with a component that *predicts* which attributes are or should be *predictive*, given the current context. How can attribute predictivity be predicted? Remember that the tables updated in the meta-learning process (in particular $\hat{C}_{ij}$ and $AV\hat{C}_{ijkl}$) summarize information about the correlation between the occurrence of certain attribute values and the predictivity of others. That is exactly the kind of information needed by a *Bayesian classifier*. METAL(IB)–FW thus uses Bayes' rule on the tables $\hat{C}_{ij}$ and $AV\hat{C}_{ijkl}$ to predict, for each attribute, its probability of being predictive, given the current context, i.e., given the incoming instance's values for the currently recognized context attributes. The weight for each attribute is then computed as

$$W(A) = 1 + P_{pred}(A)$$

where $P_{pred}(A)$ is the probability which the system assigns to the belief that A is a predictive attribute relative to the current instance.

4. **Combined Exemplar/Feature Weighting —** METAL(IB)–COM**:**
This variant performs both exemplar and feature weighting.

Our expectation was that weighting approaches would generally be less brittle than strict exemplar selection, and that *feature weighting* in particular would produce a new, interesting effect: as the feature weights are derived, via Bayes' rule, from METAL(IB)'s meta-level tables, which in turn are a kind of *memory of past contexts*, this feature weighting strategy should enable the system to readjust more quickly to contexts that have already occurred before. METAL(IB)–COM, finally, should combine the advantages of exemplar weighting (fast reaction to perceived context change by disregarding exemplars in the window that pertain to an outdated context) and feature weighting (fast adjustment to contexts that have been encountered before).

Figure 8 gives a first impression of the effects. It compares the simple instance-based learner (with fixed window of size 100) with its various meta-learning variants on the basic

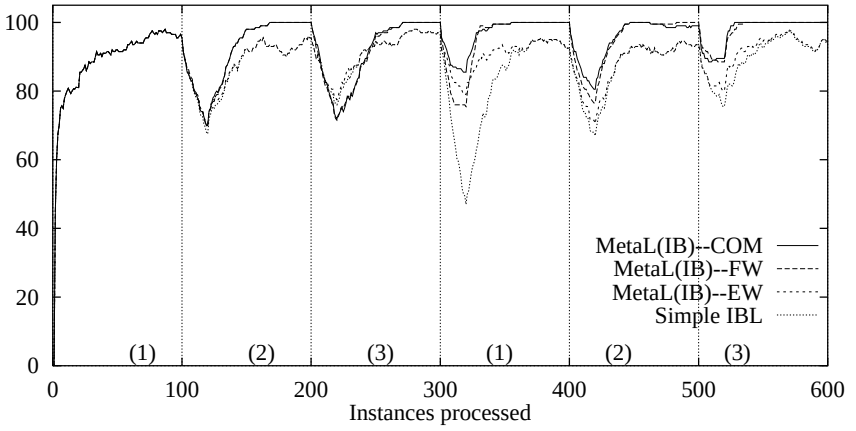

Figure 8. Context identification in METAL(IB).

STAGGER task (compare this to Figure 2 in Section 5.1). The curve of METAL(IB)–ES is not included in this plot; exemplar selection and exemplar weighting performed nearly identically in all our experiments. As before, the improvement brought about by meta-learning is evident. Among the meta-learning approaches, feature weighting performs markedly better than exemplar weighting (and equally exemplar selection), and the two feature weighting variants (METAL(IB)–FW and METAL(IB)–COM) do indeed seem to adjust to contexts faster when they reappear (see the second appearance of contexts (1), (2), and (3)). However, more focused experiments will be needed to establish this latter effect beyond doubt.

7. Summary of Experimental Results

In the following, we present some experimental results obtained with METAL(B) and METAL(IB). The following section summarizes comparative results on our artificial STAGGER domain. Subsequent sections report on a number of applications of the learners to more or less 'real-world' data where contextual effects may play a role, but where the exact characteristics of the problem are not always known.

7.1. METAL(B) vs. METAL(IB) on synthetic data

Tables 2 and 3 offer a systematic comparison of METAL(B) and METAL(IB) under various conditions in the STAGGER domain (please see Section 5 above for a description of the experimental setup). For comparison, we also give the corresponding figures for the drift tracking algorithm FLORA4. Each table entry represents the *total on-line predictive*

Table 2. Results on simple STAGGER problem.

	No irrelevant attrs Acc.% (σ)	10 irrelevant attrs Acc.% (σ)
Naive Bayes:	82.75 (2.02)	77.51 (2.12)
METAL(B):	95.75 (0.49)	85.90 (3.07)
Simple IBL:	88.10 (0.69)	68.68 (2.34)
METAL(IB)–ES:	90.05 (0.91)	74.35 (3.48)
METAL(IB)–EW:	90.05 (0.91)	74.38 (3.42)
METAL(IB)–FW:	93.75 (0.73)	73.70 (2.57)
METAL(IB)–COM:	94.23 (0.69)	78.03 (4.09)
FLORA4:	91.03 (2.19)	79.20 (5.89)

Table 3. Results on noisy STAGGER problem (with 5 irrelevant attributes).

	0% noise Acc.% (σ)	10% noise Acc.% (σ)	20% noise Acc.% (σ)	40% noise Acc.% (σ)
Naive Bayes:	79.70 (1.20)	75.73 (1.96)	73.78 (1.41)	66.80 (1.93)
METAL(B):	88.40 (2.41)	80.95 (1.79)	75.60 (2.48)	66.00 (2.22)
Simple IBL:	72.40 (1.81)	68.03 (2.17)	63.78 (2.02)	60.05 (1.30)
METAL(IB)–ES:	78.03 (2.53)	72.10 (2.96)	66.00 (2.45)	61.11 (1.46)
METAL(IB)–EW:	77.95 (2.45)	72.12 (2.88)	65.90 (2.41)	61.11 (1.50)
METAL(IB)–FW:	80.47 (1.76)	73.23 (2.57)	67.80 (2.00)	61.37 (2.34)
METAL(IB)–COM:	83.22 (2.12)	75.57 (2.87)	68.58 (2.41)	62.62 (2.22)
FLORA4:	82.80 (5.33)	72.40 (4.22)	68.30 (2.64)	61.68 (1.16)

accuracy achieved by a learner over 6000 training instances (i.e., average and standard deviation over ten runs on sequences of 600 training instances each).

Table 2 gives the results for the noise-free situation, both for the basic task and with ten additional irrelevant (random) attributes added to the data. Generally, the figures clearly show the benefits produced by meta-learning, and among the various METAL(IB) strategies, they establish the combined exemplar weighting/feature weighting strategy METAL(IB)–COM as the most effective (at least for this task).

We note that simple instance-based learning is markedly better than simple Bayesian classification in the task with no irrelevant attributes. We may attribute this to the inherent representational limitations of Bayesian classifiers; METAL(B)'s result shows that this handicap is compensated by meta-learning. In contrast, the instance-based algorithms are significantly inferior in the situation with ten irrelevant attributes, which confirms one of the fundamental (and well-known) shortcomings of instance-based approaches.

A similar picture emerges when the data are subjected to *attribute noise*. Table 3 lists overall accuracies in the STAGGER domain (with 5 additional irrelevant attributes) for different noise levels. Again, meta-learning brings considerable improvement, but the amount of improvement decreases with higher noise levels. The combined strategy METAL(IB)–COM

again turns out to be the best METAL(IB) variant, but the Bayesian system METAL(B) clearly outperforms the instance-based learners.

FLORA4, which detects concept drift by performance monitoring but cannot utilize contextual clues, performs clearly better than simple base-level learning, but not quite as good as the best meta-learners (except in the situation with ten irrelevant attributes, where it outperforms the METAL(IB) learners, which suffer from the well-known *curse of dimensionality* problem (see, e.g., Friedman, 1994)). Also, FLORA4 seems to be more sensitive to noise than the Bayesian learners. Note also the higher variance in FLORA4's results; the system seems to be less stable than the meta-learners, which is due to the highly reactive, but also somewhat 'nervous' window adjustment heuristic.

7.2. Real data: Chord prediction

The systems were also tested on a number of more complex 'real-world' problems. The first problem comes from the domain of tonal music and consists in learning to predict (on-line) what chord should accompany the next note in a given melody. Specifically, the task is to correctly predict one of three classes: *tonic harmony* (e.g., the note is to be played with a C major chord, if the piece is in the key of C major), *dominant* (i.e., a G major chord in the key of C), or *other*. In terms of a real scenario, imagine a guitar player who is trying to accompany a singer in real time on pieces she does not know and tries to get at least the two most important chord types (tonic and dominant) right.

The data used for the experiment were the melodies of Franz Schubert's *German Mass*, a collection of eight songs of varying length (between 42 and 113 notes). There are 553 melody notes in total. The distribution of classes is 290 (52%) *tonic*, 179 (32%) *dominant*, and 84 (15%) *other*.

The individual notes were described by 11 discrete attributes: the *mode* of the current piece (major or minor), the *meter* (e.g., 4/4, 3/4, or 6/8), the current *tactus* (i.e., whether the major metrical level — the level at which one would tap one's foot in rhythm — is the level of quarter or eighth notes), the current *local key* (to describe modulations within a piece), and various attributes that describe the current note itself and its predecessor: *scale degree* (a tonality-independent abstraction of the note name), *duration*, and *metrical strength* of the current note, *duration* of the note's predecessor, the *interval* and its *direction* between the previous and the current note, and the *harmony* that accompanied the previous note.

We conjectured that more global properties like mode, meter, tactus, and local key might have a context-defining effect in certain cases, i.e., that the rules determining the harmonic role of a note might be slightly different in some of these contexts. However, we don't know this in detail, and the contextual effects, if any, might be weak and difficult to discover.

What makes the problem even harder is that the given attributes are highly *inadequate*: there are numerous cases of notes with the same description but different classification. Harmonic choices are by no means unique, and the specific decision also depends on aspects of larger context (musical form, harmonic rhythm, etc.) that are not captured by our local representation. It is thus clearly impossible to achieve a predictive accuracy of anything close to 100%.

Table 4. Results of Schubert experiment.

Learning algorithm	Mean acc. (%)	Std. dev.	# runs better	min.better	max.better
Naive Bayes:	68.75	1.07	0	—	—
METAL(B):	74.62	1.44	50	2.57	9.22
Simple IBL:	76.14	0.87	0	—	—
METAL(IB)–COM:	79.58	0.89	50	1.99	5.25

To reduce the effect of the specific ordering of the songs, the algorithms were run on 50 random permutations of the eight songs. The window size was set to 300 to enable the learners to reuse information learned from previous songs (the average song length is 69). Table 4 shows the results in terms of total classification accuracy. In both cases, the advantage of meta-learning over the respective simple base-level learner is significant at the 0.05 level, according to a two-tailed t-test. Also, the meta-learners scored better than the simple classifiers in *all* of the 50 runs, with a maximum advantage of 9.2 percentage points in the Bayesian and 5.25 percentage points in the instance-based case. Both simple IBL and METAL(IB) perform markedly better than their Bayesian counterparts. The representational flexibility of the underlying IBL learner seems to be an advantage in this domain.

The attributes most often singled out as contextual were meter and tactus, less frequently mode, and very rarely local key (which was surprising to us, but probably indicates that the periods of modulation are too short and unstable to be contextually significant). Interestingly, note duration also was sometimes considered contextual: it does not help in directly predicting the harmony, but it is useful as a 'secondary' decision criterion. In other words, there is some dependency between this attribute and some more predictive ones, and meta-learning resolves the dependency by treating note duration as a contextual attribute.

As an illustrative example, Figure 9 plots the on-line prediction accuracy of METAL(B) vs. the simple Bayesian classifier in a single run (on the standard song ordering as given in the original mass). The dashed vertical lines indicate the boundaries between different songs. Note, however, that context changes may also occur in the middle of songs — e.g., when a modulation changes the local key. Indeed, inspection of the learning traces indicates that there is quite a number of rather small contexts, some of which recur quite frequently.

This observation points to an alternative way of interpreting meta-learning: instead of a focusing or selection strategy, it could also be viewed as a process of *transfer* of (learned) knowledge *across contexts*. That interpretation leads one to ask the following question: could it be that the eight pieces are so different that there cannot be much useful transfer from one piece to the next, i.e., that one would achieve better results by learning from each piece *separately*, simply starting from scratch with every new piece? And indeed, it turns out that running, e.g., a simple Bayesian classifier on each piece separately yields a total accuracy over the eight songs of 69.54%, which is slightly *higher* (though not at a statistically significant level) than the 68.75% achieved by simple Bayesian learning with a fixed window over the whole sequence! METAL(B), on the other hand, achieves markedly higher accuracy. An intriguing, though at this point somewhat speculative, explanation of

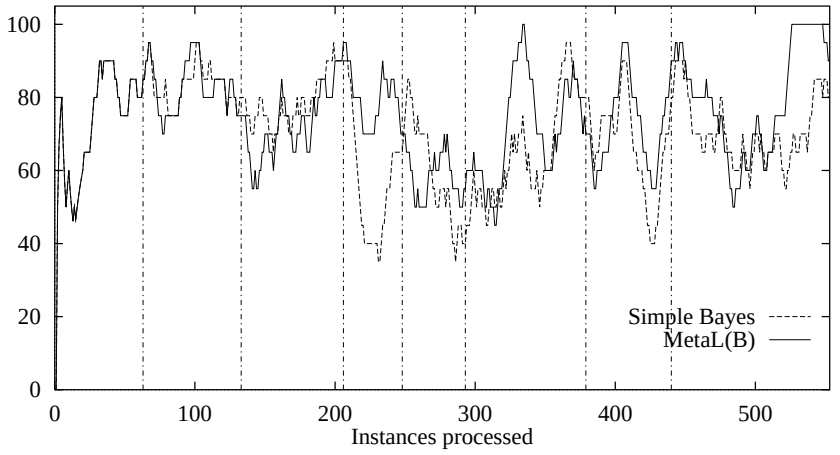

Figure 9. Accuracy in chord prediction task (METAL(B)).

this effect is that indiscriminate transfer can indeed be harmful, but that our meta-learners perform what might be called *selective cross-contextual transfer* — only those pieces of information are carried over that appear relevant to the current context. This observation suggests an interesting direction for future investigations.

7.3. Real data: Vowel recognition

Another real-world domain with obvious contextual effects is speech recognition. A speaker's sex, nationality, or age may have a strong influence on the relevance of various features. For an experiment in this domain, we used P. Turney's version of the 'vowels' data set from the UCI repository (Merz & Murphy, 1996). The problem is to recognize a vowel spoken by some arbitrary speaker. The training instances are vowels spoken by different persons. There are eleven classes (different vowels), and the instances are described by ten continuous features (derived from spectral data). In addition, there is a symbolic attribute specifying the speaker's sex (male or female). Speaker identity (the person's name), which is also given in the original data, was not used as an attribute.

Each of the eleven vowels is spoken six times by each speaker. There are 15 different speakers, eight male, seven female. The data come in groups: all the 66 examples pertaining to one particular speaker appear in a contiguous group, and the sequence of speakers in terms of sex is 4 male – 4 female – 4 male – 3 female.

As the data contain numeric attributes, our algorithms had to be extended to handle also numbers. For the instance-based learner underlying METAL(IB) this is trivial. The solution adopted for the Bayesian classifier in METAL(B) is quite simple (for more sophisticated approaches see John & Langley, 1995). We assume that numeric features follow a normal distribution; instead of maintaining counts of attribute-value and attribute-value-

class combinations, the Bayesian classifier keeps track of the mean values and standard deviations, from which the required probabilities can be estimated. At the meta-level, finally, we have settled for a rather simplistic preliminary solution: numeric attributes are discretized into N (currently 4) intervals of equal length; these are then treated as symbolic values at the meta-level (i.e., when it comes to determining which features are predictive or contextual).

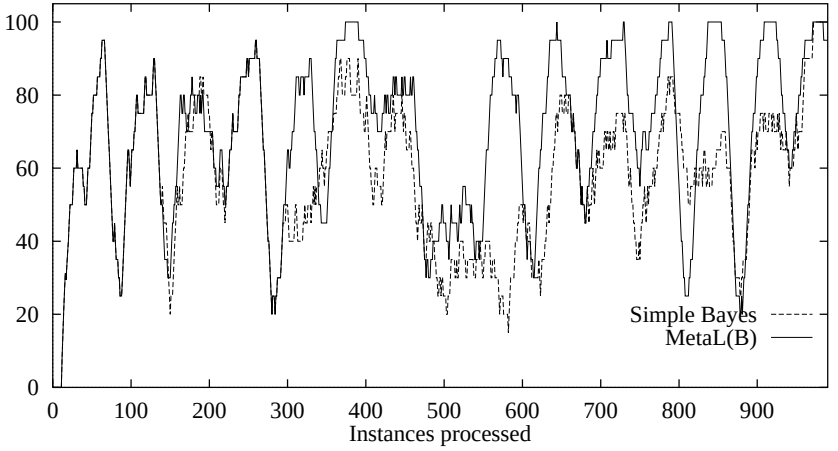

Figure 10. Accuracy in vowel classification (METAL(B)).

Figure 10 shows METAL(B)'s performance in this task (with a window size of 300). Note the clear peaks corresponding to the 15 speakers. There seem to be considerable differences between individual speakers. Simple Bayes and METAL(B) perform basically identically for the first four speakers (which are all male), but soon after the gender has changed for the first time (with speaker 5), METAL(B) recognizes sex as a contextual attribute and from then on adapts to new speakers much more effectively than the naive Bayesian classifier.

METAL(IB)'s result is rather surprising (Figure 11): there is no difference between simple IBL and meta-learning. Simple IBL is near perfect in this domain. All the speakers seem to be extremely consistent in their way of producing vowels, and sufficiently different from each other so that a local classification method like IBL can achieve maximum accuracy. On the other hand, the differences between individual speakers, even of the same sex, seems to preclude any useful transfer between them.

7.4. Real data: Calendar scheduling

As a final example of a real-world domain where context tracking may be beneficial, consider the calendar scheduling problem described in (Mitchell et al., 1994), where the task is to predict certain aspects (location, duration, start time, and day of week) of meetings

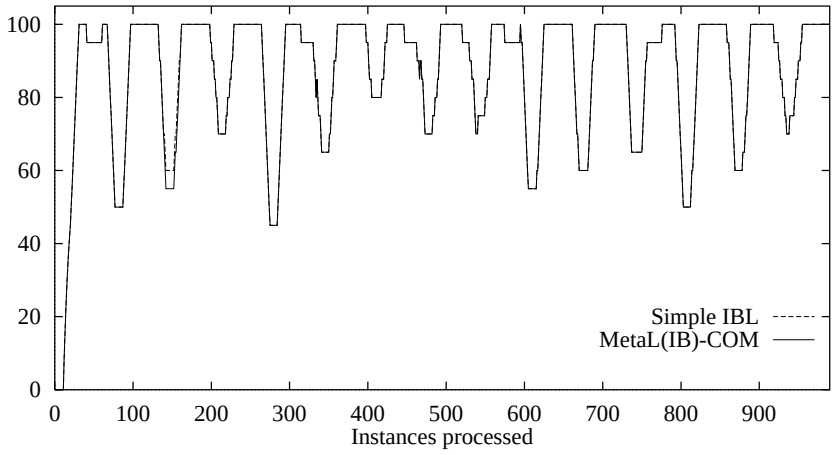

Figure 11. Accuracy in vowel classification (METAL(IB)).

in an academic environment, based on information about the events (e.g., information about the attendees) and about the recent scheduling history. It seems plausible to assume that scheduling patterns and preferences may depend on the season and may change at semester boundaries in the academic year.

As a very preliminary first result, Figure 12 shows METAL(B)'s performance in predicting the duration of each meeting on the sequence of 1685 entries from one user's (Tom Mitchell's) calendar.[6] The calendar spans a period of approximately 23 months, and in addition to the given attributes, we added a very grossly (i.e., without knowledge about the specific situation at CMU) derived attribute to each instance, namely, the *semester* or *academic term*: events in the months January through May were assigned to the *spring* term, June through August to *summer*, September through December to *fall*.

The figure shows no really spectacular effects. The two most obvious points where the two curves differ are around $x = 400$ — this is soon after the first context change from summer to fall — and around $x = 800$, which is soon after a change from fall to spring. In particular, between instances 780 and 860 METAL(B) avoids a valley of decreased accuracy that the Bayesian classifier has to go through. An analysis of the learning protocol shows that this is indeed due to the attribute *term* being picked out as a contextual clue. METAL(IB) produces a similar effect, but it generally performs more poorly in this task.

This preliminary result is quite inconclusive; if anything, it does suggest that some contextual influences are at work here, and that they can be detected. More experimental work with different target classes and maybe more refined context information may lead to more substantial results.

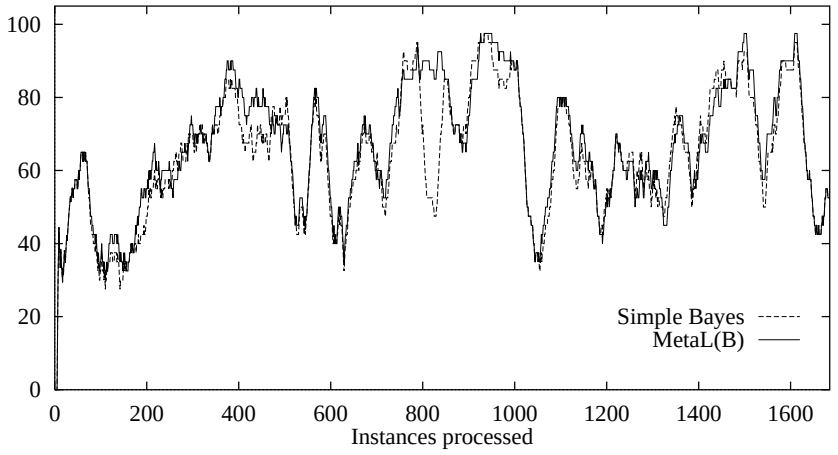

Figure 12. Predicting the duration of meetings (METAL(B)).

8. Related Work

A number of methods and algorithms for adapting classifiers to different contexts have recently been developed, many of them motivated by the emergence of context effects in practical applications. For instance, Watrous and Towell (1995) describe a neural network for ECG monitoring that is augmented with an explicit 'patient model'. The model consists of three pre-defined parameters and is used to adjust the neural classifier to the individual characteristics of a particular patient by modulating the weights on the inter-layer connections. The model can be trained on individual patients. A similar approach was taken to adapt a classifier to different speakers in speech recognition (Watrous, 1993).

Earlier, Katz et al. (1990) had described a two-stage neural network classifier, where a higher-level network learned to switch between a set of n base-level classifiers. The application domain was the recognition of targets on infrared and daytime television images. Different contexts were characterized by features such as lighting conditions and maximum image contrast. Again, these contextual attributes were explicitly defined beforehand. Examples from different contexts had to be presented in separate batches.

Turney (1993) discusses classification problems where the test examples (those that will be processed using the learned classifier) are governed by a different context than the training set from which the classifier was learned. He discusses several *normalization strategies* that use information about contextual and context-sensitive features to transform the examples to be classified. The methods assume that the contextual and context-sensitive features are known *a priori*. The methods are demonstrated in a number of practical applications, among them, the diagnosis of aircraft engines (Turney & Halasz, 1993).

In the area of learning control rules, interesting work on detecting context changes has been reported in (Ramsey & Grefenstette, 1993). In their system SAMUEL, a learner monitors a set of pre-defined context indicator variables; a change in these is interpreted

as signaling the beginning of a new context, and the underlying concept learner (a genetic algorithm) is restarted on the set of new observations (more precisely, it generates new synthetic observations by running an internal world simulation model).

All these approaches assume that contextually relevant attributes are known, or that the learner is in some way explicitly trained on different contexts. Previous attempts at automatically recognizing context changes without information about context clues have relied primarily on monitoring the performance element (e.g., Cobb & Grefenstette, 1993; Widmer & Kubat, 1996): sharp drops in predictive accuracy or some other measure of performance were interpreted as signs of a potential context change. The novelty of the methods presented here is that contextual features are detected automatically and dynamically, during the regular (on-line) learning process, and that they are then used immediately to focus the classifier on relevant information.

On the other hand, the SAMUEL system of Ramsey & Grefenstette (1993) includes an interesting strategy called *case-based initialization of genetic algorithms*. When the genetic algorithm is restarted on a new set of observations, the initial population of candidate control rules is selected on the basis of previous experience: the system looks at previous (stored) contexts and select the best rules from the n most similar previous contexts to seed the genetic algorithm. That is a form of direct *transfer* of knowledge between similar contexts. A similar strategy was implemented in the context tracker FLORA3 (Widmer & Kubat, 1993), which stored and re-used classification rules presumably pertaining to different contexts. Some such ability would be desirable also in the $\mathrm{METAL(B)}$ and $\mathrm{METAL(IB)}$ systems. Also, there has recently been some work on adapting learned concept descriptions to new contexts (e.g., Kubat, 1996) that might lead the way to interesting extensions of our algorithms. These are topics of current research.

With respect to on-line learning and the dynamic tracking of changes, the first to address the problem of *concept drift* were Schlimmer and Granger (1986). Their system STAGGER learns by updating statistical (Bayesian) measures of logical sufficiency and necessity of a set of description items in a distributed concept representation, and by simultaneously and incrementally searching the space of description items that can be constructed by conjunction and disjunction of individual features.

The FLORA algorithms (Widmer & Kubat, 1996) use a time window and a dynamic windowing strategy in an attempt to react even faster to concept drift. The window size und thus the rate of *forgetting* is controlled and dynamically adjusted by a heuristic that monitors the learning process. Similar time-based forgetting operators were also put forward for unsupervised learning situations (Kilander & Jansson, 1993).

More sophisticated and selective forgetting strategies are conceivable. For instance, Salganicoff (1993) introduced the notion of *density-adaptive forgetting*. The idea is not to rely solely on the age of exemplars. Rather, examples are forgotten only if there is subsequent information in their vicinity in attribute space to supersede them. That prevents useful information from being discarded simply because it has not been refreshed quickly enough. The $\mathrm{METAL(B)}$ and $\mathrm{METAL(IB)}$ learners in their current form need to maintain a time-based window, because the calculation of predictive and contextual attributes crucially depends on tracking changes over time. Still, some more selective forgetting strategy akin

to density-adaptive forgetting might be an interesting option. Again, this is a topic for future research.

9. Conclusion

To summarize, the main contribution of this article is to have shown that it is indeed possible for an incremental learner to autonomously detect, during on-line object-level learning, contextual clues in the data if such exist. The key is an operational definition of predictive and, based on those, contextual features. Identification and subsequent use of contextual information is an act of *meta-learning*. There are various ways in which such context information can be used. Two different realizations of the meta-learning model have been presented: METAL(B) relies on a Bayesian classifier as the underlying incremental learner and uses context information to select those known instances as a basis for prediction that seem to belong to the same context as the new instance. METAL(IB) is based on an instance-based algorithm and uses contextual information for exemplar and feature weighting. The feasibility of context recognition and its benefits have been shown in a number of experiments.

The question as to which of the two systems — METAL(B) or METAL(IB) — is better cannot be answered in this form. As our experimental evidence suggests, some domains lend themselves more to an instance-based approach, while in others — especially in domains with high noise rates and many irrelevant attributes — the Bayesian approach may prove more effective. Moreover, alternative realizations of the general learning model with other, more robust or powerful base-level learners are conceivable. For instance, the simple nearest neighbor classifier used in METAL(IB) could be replaced by more sophisticated instance-based methods that are also better at tolerating noise. A number of other conceivable extensions and refinements of the algorithms have been identified in the previous section.

Generally, we regard the work presented here as a small first step into what might become a rich field of research. The identification of contextual features is a first step towards *naming*, and thus being able to *reason about*, contexts. That is the level where we expect the full power of meta-learning to become apparent. Reasoning and learning about contexts could be used in a variety of ways and for a number of purposes, e.g., constructive induction, the recognition of (and faster readjustment to) previously encountered contexts, the emergence of expectations and predictions of the next context, etc.

There is also an interesting connection between our learning model and the notions of *transfer* and *life-long learning*, as recently proposed by Thrun and Mitchell (1995). As noted above in the context of the Schubert experiment, our learners can be interpreted as performing *cross-contextual transfer*, and they certainly are 'lifelong' learners. At first sight, the two models might appear to be orthogonal (one performs transfer across learning tasks, the other across contexts within a single task), but there are interesting parallels, and further research might lead to the formulation of a more general model that integrates both aspects of transfer.

Acknowledgments

The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry for Science, Transport, and the Arts. The author gratefully acknowledges the extremely helpful and constructive criticism by two anonymous reviewers.

Notes

1. Indeed, contextual attributes will be used for this purpose in the algorithm $\mathrm{METAL(IB)}$.
2. Note also how our definitions generalize the intuitive notion of a contextual clue as something that is *constant* during one context: an attribute that takes a constant characteristic value during each context (and a different one for each context) will be easily identified as a contextual clue by our definitions, provided that there are indeed changes in the target concept that manifest themselves in the form of changing attribute predictivity. However, our definitions do not require perfect constancy; all they require is a sufficiently strong correlation, over time, between observed predictivity of certain features and observed occurrence of others.
3. Of course, *all* values of the respective attributes (in principle), not just those appearing in the positive concept definition; the other values are predictive of negative instances.
4. It may seem surprising that $\mathrm{METAL(B)}$ fails to achieve a perfect 100% during period *P*, but then performs perfectly afterwards. Close examination reveals that this is due to the limited window size of 100 and the highly unbalanced instance distribution during *P*. Instances with $x = 0$ are so rare that the prediction of the focused base-level learner is based on very few cases whenever $x = 0$ in an incoming example, which allows coincidental features to mislead the Bayesian classifier. It is only later, when the examples are again uniformly distributed, that predictive performance becomes perfect.
5. Of course, this analogy is only approximate, implying as it does that each context-specific concept version would be purely conjunctive. Still, the analogy may be useful in giving us an alternative view of the problem of learning DNF, especially in on-line settings.
6. The data have been made available by Tom Mitchell and co-workers at http://www.cs.cmu.edu/ afs/cs/project/theo-5/www/cap-data.html. Special thanks to Dayne Freitag for help with data and attributes.

References

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

Bergadano, F., Matwin, S., Michalski, R.S., & Zhang, J. (1992). Learning two-tiered descriptions of flexible contexts: The POSEIDON system. *Machine Learning, 8(1)*, 5–43.

Cobb, H.G., & Grefenstette, J.J. (1993). Genetic algorithms for tracking changing environments. *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp. 523–530). San Mateo, CA: Morgan Kaufmann.

Friedman, J.H. (1994). *Flexible metric nearest neighbor classification.* Unpublished manuscript, available by anonymous FTP from playfair.stanford.edu/pub/friedman.

John, G.H., & Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann.

Katz, A.J., Gately, M.T, & Collins, D.R. (1990). Robust classifiers without robust features. *Neural Computation, 2(4)*, 472–479.

Kilander, F., & Jansson, C.G. (1993). COBBIT - A control procedure for COBWEB in the presence of concept drift. *Proceedings of the Sixth European Conference on Machine Learning* (pp. 244–261). Berlin: Springer Verlag.

Kubat, M. (1989). Floating approximation in time-varying knowledge bases. *Pattern Recognition Letters, 10*, 223–227.

Kubat, M. (1996). Second tier for decision trees. *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 293–301). San Francisco, CA: Morgan Kaufmann.

Kubat, M., & Widmer, G. (Eds.) (1996). *Learning in context-sensitive domains (Workshop Notes)*. 13th International Conference on Machine Learning, Bari, Italy.

Langley, P., Iba, W., & Thompson, K. (1992). An analysis of Bayesian classifiers. *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 223–228). Menlo Park, CA: AAAI Press.

Merz, C.J., & Murphy, P.M. (1996) UCI repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science. [http://www.ics.uci.edu/∼mlearn/MLRepository.html].

Michalski, R.S. (1987). How to learn imprecise concepts: A method employing a two-tiered knowledge representation for learning. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 50–58). Los Altos, CA: Morgan Kaufmann.

Mitchell, T.M., Caruana, R., Freitag, D., McDermott, J., & Zabowski, D. (1994). Experiences with a learning personal assistant. *Communications of the ACM, 37(7)*, 81–91.

Ramsey, C.L., & Grefenstette, J.J. (1993). Case-based initialization of genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp. 84–91). San Mateo, CA: Morgan Kaufmann.

Salganicoff, M. (1993). Density-adaptive learning and forgetting. *Proceedings of the Tenth International Conference on Machine Learning* (pp. 276–283). San Mateo, CA: Morgan Kaufmann.

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

Thrun, S., & Mitchell, T.M. (1995). Learning one more thing. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (pp. 1217–1223). San Mateo, CA: Morgan Kaufmann.

Turney, P.D. (1993). Exploiting context when learning to classify. *Proceedings of the Sixth European Conference on Machine Learning* (pp. 402–407). Berlin: Springer Verlag.

Turney, P.D., & Halasz, M. (1993). Contextual normalization applied to aircraft gas turbine engine diagnosis. *Journal of Applied Intelligence, 3*, 109–129.

Watrous, R.L. (1993). Speaker normalization and adaptation using second-order connectionist networks. *IEEE Transactions on Neural Networks, 4(1)*, 21–30.

Watrous, R.L., & Towell, G. (1995). A patient-adaptive neural network ECG patient monitoring algorithm. *Proceedings Computers in Cardiology 1995*, Vienna, Austria.

Widmer, G., & Kubat, M. (1993). Effective learning in dynamic environments by explicit context tracking. *Proceedings of the Sixth European Conference on Machine Learning* (pp. 227–243). Berlin: Springer Verlag.

Widmer, G. (1994). Combining robustness and flexibility in learning drifting concepts. *Proceedings of the Eleventh European Conference on Artificial Intelligence* (pp. 468–472). Chichester, UK: Wiley.

Widmer, G. (1996). Recognition and exploitation of contextual clues via incremental meta-learning. *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 525–533). San Francisco, CA: Morgan Kaufmann.

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning, 23(1)*, 69–101.

Received June 28, 1996
Accepted October 3, 1996
Final Manuscript November 7, 1996