Recognition and Exploitation of Contextual Clues via Incremental Meta-Learning (Extended Version)

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

Abstract

Daily experience shows that in the real world, the meaning of many concepts heavily depends on some implicit context, and changes in that context can cause more or less radical changes in the concepts. Incremental concept learning in such domains requires the ability to recognize and adapt to such changes.

This paper presents a solution for incremental learning tasks where the domain provides explicit *clues* as to the current context (e.g., attributes with characteristic values). We present a general two-level learning model, and its realization in a system named METAL(B), that can *learn* to *detect* certain types of contextual clues, and can react accordingly when a context change 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 identifies potential contextual clues. Context learning and detection occur *during* regular on-line learning, without separate training phases for context recognition. Experiments with synthetic domains as well as a 'real-world' problem show that METAL(B) is robust in a variety of dimensions and produces substantial improvement over simple object-level learning in situations with changing contexts.

The meta-learning framework is very general, and a number of instantiations and extensions of the model are conceivable. Some of these are briefly discussed.

Key words: on-line learning, concept drift, context dependence, transfer

1 Motivation

This article is concerned with incremental supervised concept learning, also known as *on-line* learning (e.g., Littlestone, 1988). A learning algorithm is incrementally processing a stream of examples coming in. Each instance is associated with a given class label. The learner first attempts to classify the instance, then learns the correct class label, and updates some internal concept hypothesis, if appropriate. Many real-world problems (e.g., discrete control tasks) can be cast in such a framework.

A practical problem that has been more and more acknowledged in recent years is that in real-world tasks, the concepts of interest may not be entirely stable, in other words, they may *change* in time. An on-line learning algorithm should be capable of recognizing such changes and adjust its hypotheses and behavior accordingly. This notion of *concept drift* was first discussed by Schlimmer and Granger (1986), in the context of their drift-tracking algorithm STAGGER.

Another way of viewing the phenomenon of concept drift is to assume that the concepts of interest depend on some (maybe hidden) *context*, and that changes in this context induce corresponding changes in the target concepts. Recent work on drift and context tracking in the FLORA family of algorithms (Widmer and Kubat, in press) was motivated by this view. 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 *contextual clues* come to mind (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 in-door robotic tasks, speaker nationality and sex in speech processing, etc.). 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 a system named METAL(B), 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. At the moment, both object-level and meta-level learning are restricted to nominal (discrete) attributes, but extensions of the model to numeric domains seem rather straightforward.

The following section briefly describes the specific learning algorithm we use for object-level learning. Section 3 then motivates and describes in detail the meta-learning strategy and its realization. The algorithm's behavior was tested along various dimensions; experiments with several synthetic domains and one 'real-world' problem are presented and discussed in section 4. Section 5 discusses other work in machine learning that is related to our approach. We believe that the work described in this article represents a first step into a very interesting and promising field of research, and section 6 points to a number of interesting directions that should be pursued.

2 The base level: learning a simple Bayesian classifier

Let us take a very simple learning algorithm for the basic object-level learning task. 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. A very readable introduction to simple Bayesian classifiers can be found, e.g., in (Langley, 1993). 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 the 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 compute the probability of a conjunction of values, one usually assumes that the attributes are independent, so that $p(\bigwedge v_j|c_k)$ can be computed as

$$p(\bigwedge v_j|c_k) = \prod_j p(v_j|c_k)$$

A new instance is assigned to the class with the highest posterior probability. Incremental induction of a Bayesian classifier is straightforward. One only needs to maintain a number of counters, from which the required 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, naturally incremental, and robust in the face of noise. Its major weakness is that we must assume that the given attributes are *independent* in their influence on the class value, which severely limits the class of learnable concepts. Concepts like XOR are inherently unlearnable. (But see section 4.6 below.)

In the following, we will use this simple algorithm for incremental, on-line learning from a stream of incoming examples. As we are interested in dynamic environments where changes in context may lead to changes of the target concepts, the learner will be equipped with a *window* of fixed length: as new examples are added to the window, the oldest ones will be 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 three tables C_i, AV_{ij}, and AVC_{ijk} are always updated with respect to the examples in the current window.

3 Contextual features and meta-learning: MetaL(B)

When the underlying target concept drifts or changes due to a changed context, the Bayes induction algorithm will eventually adjust to the new concept, if the new context is stable for a sufficiently long period. The smaller the fixed window, the faster the adjustment, as old, contradictory examples will be forgotten more quickly. However, in domains that provide explicit clues as to the current context, one would expect more: the learner should learn to recognize such clues and should react in some appropriate way when changes in these clues signal a potential context change. This section shows how such behavior can be achieved, and presents an algorithm called MetaL(B) (MetaLearning with underlying Bayes classifier) that learns at two levels at a time: a base-level Bayesian algorithm learns to classify incoming examples, and a meta-level learner tries to explicitly identify potential contextual clues, which are used to make the base-level classifier more selective with respect to the training instances used for prediction.

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 explicitly identify contextual features beforehand. His particular approach was motivated by batch learning problems where the *testing examples* (i.e., those on 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. This cannot be expected in general. We will rely on a more abstract and powerful notion: a feature is considered a contextual clue if it does not directly determine or influence the class of an object, but 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. 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).

In the following, we 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 our 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 (see section 2 above).

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 table $\hat{AVC}_{ijkl}$ 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 features $a_k : v_{kl}$. These three tables are maintained with respect

Table 1: Tables maintained for base-level and meta-level learning.

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

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 have to be maintained by our algorithm. 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 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).

Obviously, since the definition of contextual attributes comprises two levels, the success of this process hinges on the reliable recognition of predictive features. Our experience indicates that very strict significance levels for the χ^2 tests yield the best results. More on that in section 4.2 below.

3.3 Using contextual features to focus on relevant examples

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. They are now 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 base-level learner 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

Table 2: The complete two-level algorithm of MetaL(B)

Parameters: W (fixed window size), α (significance level for χ^2 test).

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 selected 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;

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 attributes are currently recognized as contextual, the entire window is used for Bayesian classification of the new instance. After classification, the true class of the new instance is read, and the learning tables for both base and meta-level are updated. Table 2 summarizes the complete two-level algorithm of MetaL(B).

A consequence of this selective strategy is that base-level class 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) would be, and the fixed window size at least puts an upper bound on the number of examples that must be examined at each point in the learning process. In the learning experiments performed so far, we have not encountered any practical problems in this respect.

Note that the algorithm MetaL(B) depends on two *parameters*. In all the experiments reported below we used a significance level of α=0.01 (99% confidence that the observed difference between conditioned and unconditioned distributions is not due to chance) and a fixed window size of 100.

4 Experiments

In the following, various aspects of MetaL(B)'s capabilities and behavior are tested on a number of synthetic domains which allow us to systematically experiment with different problem characteristics. Section 4.7 then evaluates the meta-learning approach on a hard 'real-world' problem where contextual effects might play a role, but the exact characteristics of the problem are not known.

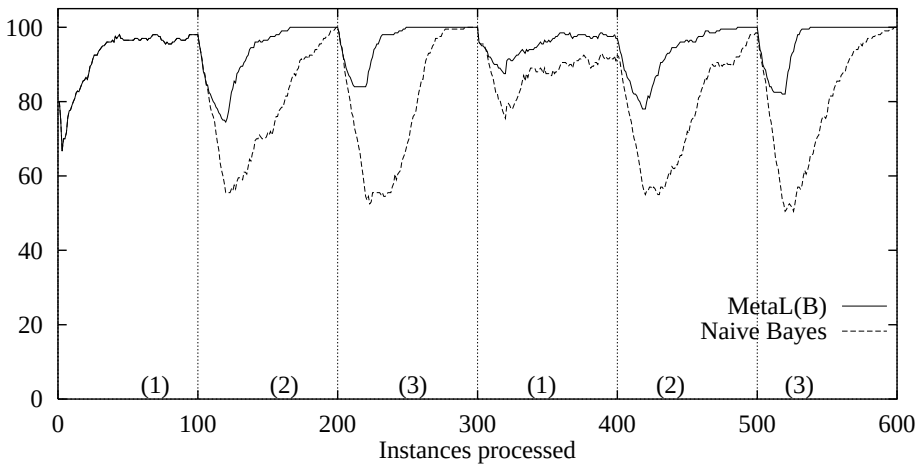

Figure 1: Effect of context identification in STAGGER concepts.

4.1 Basic context identification

To demonstrate the basic effects of the context detection mechanism, METAL(B) was first applied to a sequence of simple concepts 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 and Kubat, in press). 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 1 plots the on-line predictive 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 and labeled according to the currently ruling concept, and after every 100 instances the context plus underlying concept was made to change. On-line predictive accuracy was computed as a running average over the last 20 predictions. The figure plots the averages over 10 (paired) runs. Parameter settings were $\alpha = 0.01$ and $window\ size = 100$.

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 the still pertain to the old context. The fact that both algorithms do not always reach an accuracy of 100% within the span of 100 instances is due to a certain inertia of the underlying Bayesian learner and, in the case of concept (1), the sparsity of the concept. The improvement produced by meta-learning, however, is evident.

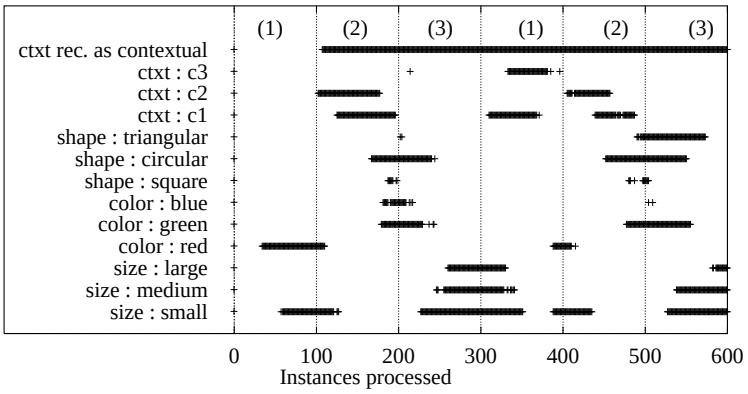

Figure 2: Predictive features and contextual attribute in STAGGER experiment.

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 2 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).[1] 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 the context attributes are established, they immediatly 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 that the class distribution (positive vs. negative instances) 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.

[1] 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.

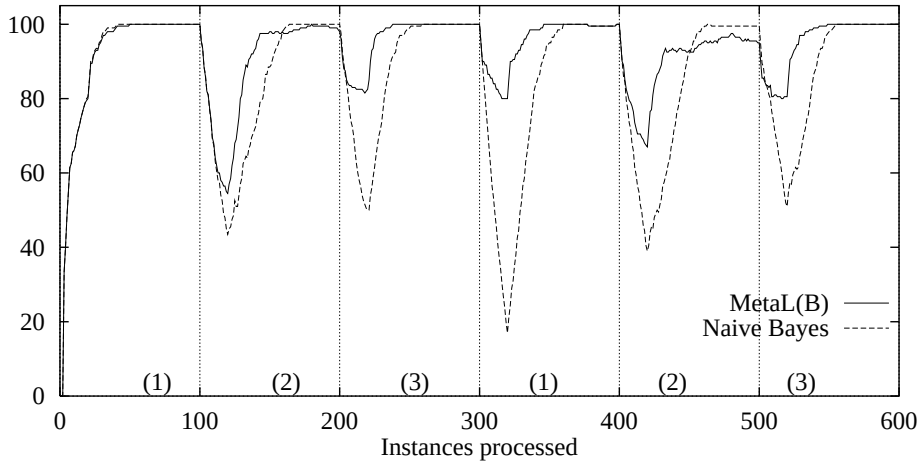

Figure 3: STAGGER concepts — window size 50.

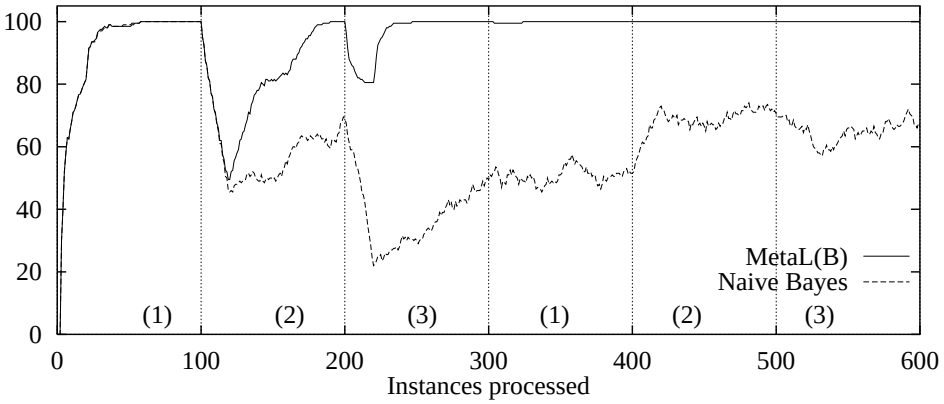

Figure 4: STAGGER concepts — window size 300.

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).

4.2 The effect of the system parameters

METAL(B) depends on two system parameters: the significance level α used in the χ^2 tests at two levels, and the fixed *window size*. A significance level of $\alpha = 0.01$ has produced good results in all our experiments so far. Less strict significance levels generally tend to 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 that certain other features are erroneously labeled contextual, and the learning process becomes unstable. On the other hand, tighter significance levels (e.g., $\alpha = 0.001$) have left our results virtually unchanged. Ultimately, the optimal setting of α will depend on the characteristics of the given application domain. This

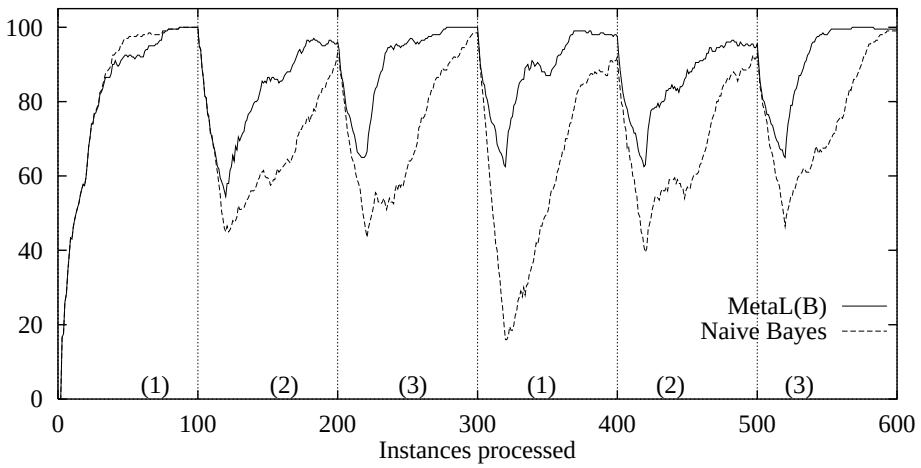

Figure 5: STAGGER concepts + 10 irrelevant attributes.

issue will require more systematic investigation in the future.

As for the *window size*, the smaller the window, the faster the base-level Bayesian learner will adapt to changes, and thus the smaller the difference between meta-learning and simple base-level learning. This is borne our in Figure 3, where the two algorithms were trained on the STAGGER concepts with a fixed window size of 50. The advantage of meta-learning is still visible. For both learners, a window that is too narrow will adversely affect predictive accuracy, as the Bayesian classifier will base its predictions on too few data points.

On the other hand, too large a window reduces the base-level learner's ability to adjust to changes and, if it permanently contains conflicting instances from different contexts, may prevent the classifier from ever reaching high predictive accuracy. Figure 4 plots the results on the STAGGER task, when the window size is set to 300. The simple Bayesian classifier fails completely, while METAL(B) detects the contextual clue somewhere during context (2) and uses it to always select the correct examples for prediction. After the first 300 examples, the window always contains instances of all three contexts, and METAL(B) perfectly masters context changes from then on. However, too large a window may also cause problems for 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.

4.3 Irrelevant attributes and noise

The presence of irrelevant attributes in the data has a detrimental effect on many inductive learners (most notably, instance-based ones). Bayesian learners, on the other hand, are generally quite robust against irrelevant attributes. One would thus expect METAL(B) not to be easily fooled by the presence of random features in the data. This should also hold for the meta-level learning algorithm, which also has a Bayesian flavor. Figure 5 confirms our expectations. Here, METAL(B) and the naive Bayesian classifier are compared on the basic task (see section 4.1), where the training examples were extended with 10 irrelevant attributes with randomly assigned values (from domains of 3 possible values each). Both learners still

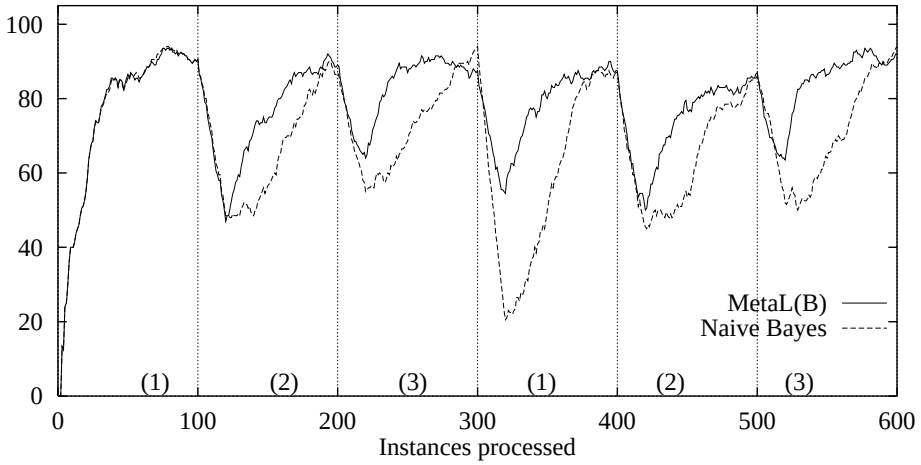

Figure 6: STAGGER concepts — 5 irrelevant attributes and 10% attribute noise.

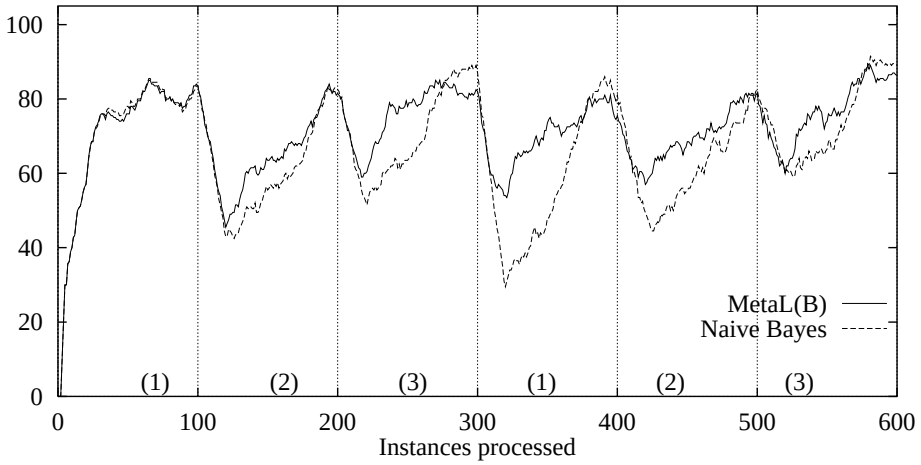

Figure 7: STAGGER concepts — 5 irrelevant attributes and 20% attribute noise.

perform well, and the context-detection capability is more or less unaffected, as evidenced by the marked advantage of METAL(B).

It is also well known that Bayesian learners are inherently noise-tolerant. To check how noise affects the context-detection facility, experiments with different levels of *attribute noise* were run (where a noise level of η% means that for each attribute in a training example, its 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. In addition, 5 irrelevant attributes with random values were added to the example descriptions. Figures 6 and 7 show the results of experiments with noise levels $\eta = 10\%$ and $\eta = 20\%$, respectively. Meta-learning still leads to improved adjustment to changes, but high noise levels may produce effects of instability, as can be seen from Figure 7. Features that are erroneously regarded as contextual, even if only for a short time, lead the Bayesian classifier to rely on too few examples for classification decisions.

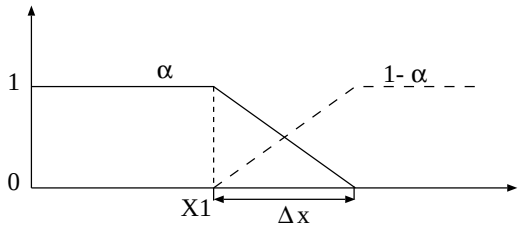

Figure 8: The function α.

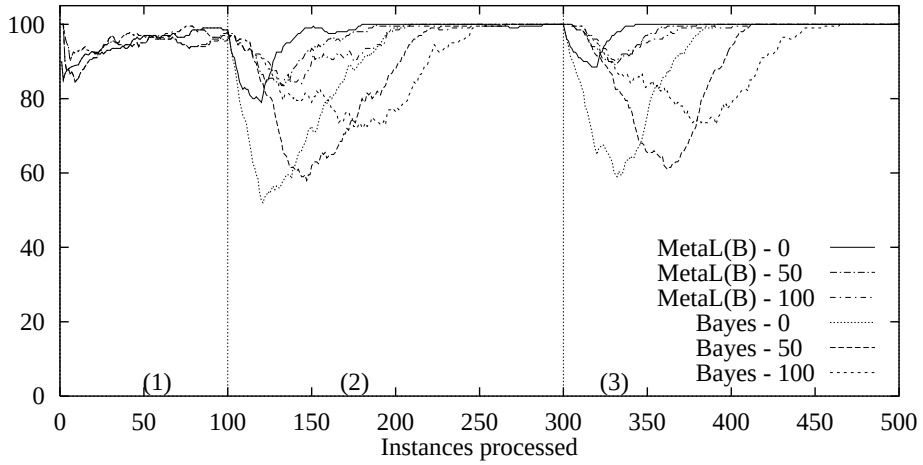

Figure 9: Abrupt vs. gradual context change.

4.4 Gradual vs. abrupt context changes

The following experiment is concerned with what might be called the *speed* of context changes, that is, whether contexts and the associated concepts change abruptly or gradually. As in (Widmer and Kubat, in press), we use a linear function α (see Figure 8) to model the speed of change. The value of α at any point in time represents the degree of dominance of the old context A over the new context B or, in other words, the probability that the current instance still pertains to the old context. $\alpha = 1$ means that A is fully in effect, $\alpha = 0$ means that B has completely taken over. $X1$ is the point where the context begins to change. The slope of the function α can be characterized by Δx, the number of training instances until α reaches zero. Between $X1$ and $X1 + \Delta x$, $\alpha * 100\%$ of the examples still belong to context A, $(1 - \alpha) * 100\%$ to B.

Figure 9 compares METAL(B) and the simple Bayesian classifier on different rates of change. Target concepts are our three familiar STAGGER concepts with additional context attribute *ctxt*; context (1) always starts to change into (2) after 100 instances, (2) starts to change into (3) after 300 instances (see the dotted vertical lines); the change rates compared are $\Delta x = 0$ (abrupt change), $\Delta x = 50$, and $\Delta x = 100$ (slow drift). The results are again averages over 10 runs.

The results are as expected. Naturally, the slower the change, the longer the valley of decreased accuracy. But it is evident that the effectivity of meta-learning is more or less unaffected by the speed of change. In particular, unlike systems like FLORA4 (Widmer,

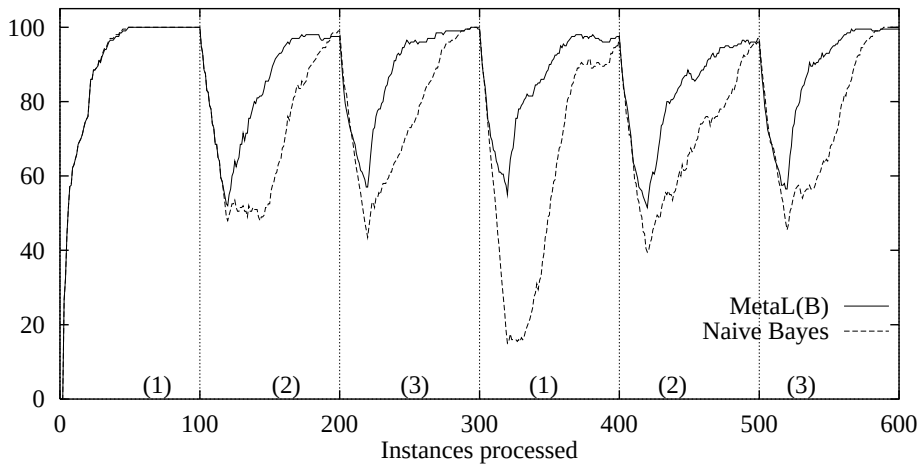

Figure 10: STAGGER concepts with changing conjunctive contextual clues.

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.

4.5 Complex and imperfect contextual clues

In real life, contexts may be defined by more complex combinations of features. As an additional complication, the identity of the contextual attributes themselves may be changing. Generally, of course, the meta-learner suffers from the same fundamental limitation as the base-level Bayesian classifier: it must assume independence between *contextual attributes*.

Preliminary experiments suggest that MetaL(B) has no problems with *conjunctively* defined contexts; *disjunctive* contexts may cause problems.

For some simple experiments, we use again the STAGGER concepts, but replace the attribute $ctxt$ with three contextually relevant attributes $a \in \{a_1, a_2, a_3\}$, $b \in \{b_1, b_2, b_3\}$, and $c \in \{c_1, c_2, c_3\}$. We define a series of three contexts as follows: context (1) (associated with the first STAGGER concept $size = small \wedge color = red$) is characterized by $a = a_1 \wedge b = b_1$, context (2) by $b = b_2 \wedge c = c_2$, and context (3) by $a = a_3 \wedge c = c_3$. Figure 10 compares the two learners on this task. Clearly, MetaL(B)'s performance is not affected by these more complex contextual clues.

The story changes considerably when contexts are characterized by *disjunctive* combinations of features. Consider the three contexts $a = a_1 \wedge b = b_1$, $b = b_2 \vee c = c_1$, and $a = a_2 \vee a = a_3$, again associated with the three STAGGER concepts.[2] As Figure 11 shows, MetaL(B) outperforms the simple Bayesian classifier in context (1), which is defined by a conjunction of features, but is clearly inferior in the other two contexts.

The explanation of this negative effect involves two factors. One problem is that contextual information is used *conjunctively* in MetaL(B)'s focusing strategy — only those examples from the window are used for prediction that share *all* context attribute values with the new instance —, which makes the base-level learner rely on too few examples in some cases.

[2]In effect, these context definitions are equivalent in form and complexity to the 'base-level' concepts to be learned.

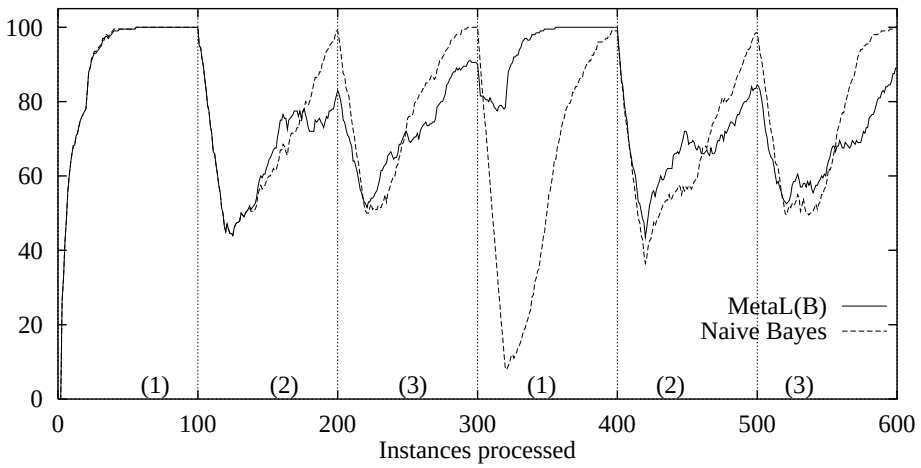

Figure 11: STAGGER concepts with partially disjunctive contextual clues.

Replacing METAL(B)'s instance selection strategy with a less brittle method (e.g., feature weighting — see section 6 below) might solve this problem.

A second problem is that changes in the identity of contextual attributes are generally difficult to track. In our example, attribute a is contextually relevant in contexts (1) and (3), but not in context (2). This is more or less impossible to detect, given the data and criteria to which METAL(B) has access. METAL(B) considers a to be contextual during the entire learning episode. But then, this may not be a serious problem in practical applications; it is hard to think of realistic situations where the contextual attributes change so radically.

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

4.6 "Quasi-contextual learning"

An interesting side-effect of the meta-learning strategy is that it may in certain cases help the Bayesian learner to overcome its inherent representational limitations. Remember that a naive Bayesian classifier must assume the independence of attributes, which severly limits the class of concepts it can represent.

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 of the examples. A Bayesian classifier will always linger around the base-level accuracy of 50%, given a set of uniformly distributed examples of XOR.

The same holds for METAL(B), as long as the examples are presented in random order. However, if during some period of the learning episode 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 of the concept. 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.

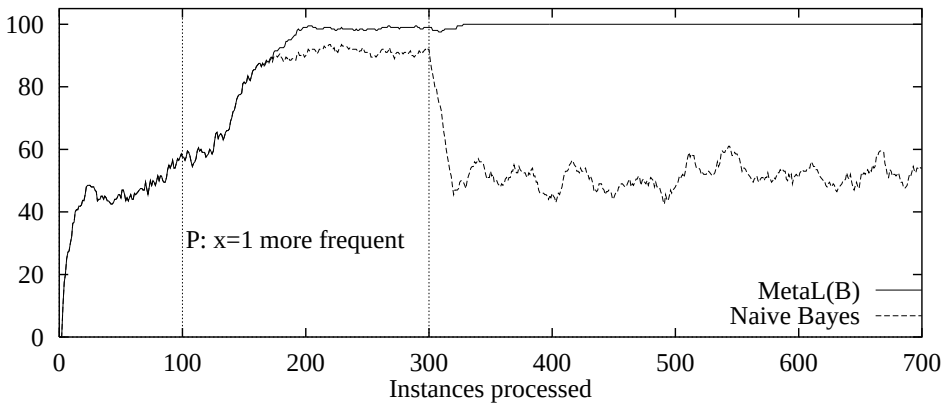

Figure 12: "Quasi-contextual" learning: XOR with 5 irrelevant attributes.

Figure 12 demonstrates this effect. Here, METAL(B) and the naive Bayesian classifier were trained 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 10 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.

It may seem surprising that 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.

In summary, this result indicates that meta-learning may be useful also in settings that are not normally thought of as characterized by changing contexts. If there are periods of skewed example distributions, for whatever reason, meta-learning can improve the discriminating power of an underlying (e.g., Bayesian) learning algorithm that has problems with disjunctive concepts.

4.7 Learning from real data

METAL(B) was also tested on a difficult 'real-world' problem with unknown characteristics. The problem comes from the domain of tonal music and consists in learning to predict (online) 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

Learning algorithm	Mean acc. (%)	Std. dev.	# runs better	max.better
Naive Bayes:	68.75	1.07	0	—
MetaL(B):	74.62	1.44	50	9.22

Table 3: Results of Schubert experiment.

accompany a singer in real time on pieces he does not know and at least tries to get 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 8 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 of 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%.

In order to reduce the effect of the specific ordering of the songs, the algorithms were run on 50 random permutations of the 8 songs. The window size was set to 300, so that the window would contain notes from more than one song. The results, in terms of the total number of examples classified correctly, were as follows: the naive Bayes learner achieved an average accuracy (over the 50 runs) of 68.75%, with a standard deviation of 1.07. The accuracy of MetaL(B) on the same data was 74.62%, standard deviation 1.44. The difference of 5.87 percentage points is significant at the 0.0001 level, according to a two-sided t-test. MetaL(B) scored better than the simple Bayesian classifier in *all* of the 50 runs, with a maximum advantage over the base-level learner of 9.2 percentage points. Table 3 summarizes the results.

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, also note duration 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 METAL(B) resolves the dependency by treating note duration as a contextual attribute.

As a kind of afterthought, we propose an alternative view of contextual meta-learning as realized in METAL(B). Instead of a focusing or selection strategy, it could also be interpreted as a process of *transfer* of (learned) information or knowledge *across contexts*. That perspective leads one to ask the following question: could it be that the 8 pieces are so different that there cannot be much useful transfer from one piece to the next, in other words, that one would achieve better overall results by learning from each piece *separately*, simply discarding the learned information when a new piece starts? And indeed, it turns out that the base-level learner, when run on each piece separately, reaches a total accuracy over the 8 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. The conclusion is that indiscriminate transfer can indeed be harmful, but METAL(B) performs what might be called *selective cross-contextual transfer* — only those pieces of information are carried over that appear relevant to the current context.

5 Relation to other work

The idea that real-world concepts are not necessarily perfectly stable and well-defined has only gradually become a topic for machine learning research. Michalski (1987) was one of the first to discuss the problem of *context dependence*. He proposed a *two-tiered* knowledge representation scheme for learning and describing context-dependent concepts, where the first tier (the 'base concept representation') would capture the more general, typical aspects of a concept, and the second tier (the 'inferential concept interpretation') would contain relevant knowledge to dynamically determine the actual meaning of the concept in a particular context. Algorithms for learning both tiers were implemented in the system POSEIDON (Bergadano et al., 1992).

Making adjustments (before or after base-level learning) to account for context effects has been the focus of some recent application-oriented research. 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. It 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. The network thus has an explicit context model that can be adjusted to the current context in a separate training phase. 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 classifier. 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 the problem of context from a different angle. He is concerned with 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 intent is to reduce the context-sensitivity of certain features. 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 and Halasz, 1993).

All these approaches assume that contextually relevant attributes are known, and that the learner is in some way explicitly trained on different contexts. The novelty of our method 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.

With respect to on-line learning and the dynamic tracking of changes, the first to address the problem of *concept drift*, i.e., that concepts may change over time, 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 family of algorithms (Widmer and Kubat, in press) introduces the idea of explicitly *forgetting* old instances and of dynamically controlling the rate of forgetting during learning. Forgetting is controlled by a window over the stream of incoming examples, and the window size is dynamically adjusted by a heuristic that monitors the learning process. Similar ideas were also put forward for unsupervised learning situations (Kilander and Jansson, 1993). In addition, FLORA3 (Widmer and Kubat, 1993) introduces a mechanism for storing and re-using learned concept descriptions in environments with changing contexts; the goal is faster re-adaptation to concepts that had already appeared in the past. In situations of abrupt context change, the FLORA systems may be a bit more reactive than METAL(B), but they are also more brittle; heuristic window adjustment is based on a number of parameters whose optimal setting may be a delicate matter. METAL(B) was designed for a different class of learning situations, where *explicit context clues* are present. The identification of these is METAL(B)'s key to success. Explicit context identification and representation also opens the door to a number of other interesting possibilities (see below).

As suggested above, there is also a certain relation between our meta-learning scenario and the notion of *transfer* of knowledge between learning tasks, as mentioned – though in rather different settings — by, e.g., Pratt et al. (1991), Ourston and Mooney (1991), Caruana (1993), and Thrun and Mitchell (1995). While these authors study the effect of *cross-category transfer*, METAL(B) can be interpreted as performing *cross-context transfer*. The effect can be most clearly seen in the Schubert experiment above.

In terms of the dynamic selection of predictors, there is some relation between METAL(B) and instance-based learners like IB3 (Aha et al., 1991). IB3 determines predictive *exemplars* by monitoring each stored instance's individual predictive accuracy. Only those exemplars are used for prediction that have established a stable classification record. METAL(B), on the other hand, uses identified contextual attributes to select predictive exemplars. The two mechanisms are thus quite different, both in intent and in effect.

Finally, the capability of "quasi-contextual learning" discussed in section 4.6 also places METAL(B) in the vicinity of systems that try to increase the representational power of Bayesian classifiers (e.g., Kononenko, 1991; Langley, 1993).

6 Discussion and directions for further research

The main contribution of this paper 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 contextually relevant feature is an act of *meta-learning*. MetaL(B), our specific incarnation of this meta-learning architecture, employs a simple Bayesian classifier as the object-level learning algorithm and uses contextual attributes, once identified, to focus the object-level learner on examples that appear pertinent to the current context. The algorithm has been shown to be powerful and quite robust in a number of test domains.

A general advantage of the MetaL(B) algorithm, at least as compared to systems like FLORA3 or FLORA4 (Widmer and Kubat, 1993; Widmer, 1994), is that it depends on only two parameters: the significance level α used in the χ^2 tests, and the fixed window size. Experiments so far indicate that the algorithm is not exceedingly sensitive to the specific settings of these. On the other hand, despite the apparent simplicity of the definitions and the MetaL(B) algorithm, there are some subtle interactions between (the detection of) predictive and contextual features, as could be seen in section 4.1. These need to be further studied.

MetaL(B) is currently limited to domains described by symbolic, discrete attributes. A generalization to numeric features should not prove too difficult. Instead of maintaining counts of attribute-value and attribute-value-class combinations, the Bayesian classifier could simply assume that numeric features follow a normal distribution and keep track of the mean values and standard deviations. At the meta-level, the χ^2 tests would have to be replaced by an appropriate test of independence for continuous distributions.

The meta-learning framework proposed is very general. The choice of a Bayesian classifier for object-level learning was rather arbitrary, motivated mainly by considerations of elegance. The meta-level, i.e., the criteria used to identify predictive and contextual features, is of a Bayesian nature, so employing a Bayesian learner for on-line classification seemed natural. The power of meta-learning, however, does not depend on the underlying generalizer, and we expect instantiations or variants of this general model for other base-level learners.[3]

One highly interesting extension suggests itself immediately. It is a trivial matter to augment the meta-level algorithm with the ability to *predict* which attributes are or should be predictive for each incoming example, given the instance's values for the currently recognized context attributes. The tables updated in the meta-learning process ($\hat{C}_{ij}$, $\hat{AV}_{ij}$, and $\hat{AVC}_{ijkl}$) contain all the necessary statistics for the prediction of predictive attributes via Bayes' rule. As these predictions are not categorical, but associated with probabilities, they could be trivially used for *feature weighting* in object-level learning. Features believed to be predictive relative to the current context would receive correspondingly higher weights, which might lead to less brittle behavior than strict exemplar selection in noisy or otherwise unstable situations. Feature weighting is not directly possible in MetaL(B), because the underlying Bayesian classifier combines feature information in a multiplicative manner. But in other types of learners (e.g., instance-based ones) this would be a simple possibility.

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*,

[3]Hence the explicit (B) in the name MetaL(B); it would be replaced appropriately in other instantiations.

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 a very interesting connection between our learning model and the notion of *life-long learning*, as recently proposed by Thrun and Mitchell (1995). Our learner can be interpreted as performing *cross-contextual transfer*, and it certainly is a 'lifelong' learner. 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 and powerful model that integrates both aspects of transfer.

Acknowledgments

I would like to thank Miroslav Kubat for continuing cooperation on the issue of context-dependent concepts. Financial support for the Austrian Research Institute for Artificial Intelligence is provided by the Austrian Federal Ministry for Science and Research.

References

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

Bergadano, F., Matwin, S., Michalski, R.S., and Zhang, J. (1992). Learning Two-tiered Descriptions of Flexible Contexts: The POSEIDON System. *Machine Learning 8(1)*, pp.5–43.

Caruana, R.A. (1993). Multitask Learning: A Knowledge-based Source of Inductive Bias. In *Proceedings of the 10th International Conference on Machine Learning (ML-93)*, Amherst, MA. San Mateo, CA: Morgan Kaufmann.

Katz, A.J., Gately, M.T, and Collins, D.R. (1990). Robust Classifiers Without Robust Features. *Neural Computation* 2(4), pp.472–479.

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

Kononenko, I. (1991). Semi-naive Bayesian classifier. In *Proceedings of the 5th European Working Session on Learning (ECML-91)*, Porto. London: Pitman.

Langley, P. (1993). Induction of Recursive Bayesian Classifiers. In *Proceedings of the European Conference on Machine Learning (ECML-93)*, Vienna, Austria.

Littlestone, N. (1988). Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm. *Machine Learning* 2(4), pp.285–318.

Michalski, R.S. (1987). How to Learn Imprecise Concepts: A Method Employing a Two-tiered Knowledge Representation for Learning. In *Proceedings of the 4th International Workshop on Machine Learning*, Irvine, CA. Los Altos, CA: Morgan Kaufmann.

Ourston, D. and Mooney, R.J. (1991). Improving Shared Rules in Multiple Category Domain Theories. In *Proceedings of the 8th International Workshop on Machine Learning (ML-91)*, Evanston, Ill. San Mateo, CA: Morgan Kaufmann.

Pratt, L.Y., Mostow, J., and Kamm, C.A. (1991). Direct Transfer of Learned Information Among Neural Networks. In *Proceedings of the 9th National Conference on Artificial Intelligence (AAAI-91)*, Anaheim, CA.

Schlimmer, J.C. and Granger, R.H. (1986). Beyond Incremental Processing: Tracking Concept Drift. In *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI-86)*, Philadelphia, PA.

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

Thrun, S. and Mitchell, T.M. (1995). Learning One More Thing. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montréal. San Mateo, CA: Morgan Kaufmann.

Turney, P.D. (1993). Robust Classification with Context-Sensitive Features. In *Proceedings of the 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-93)*, Edinburgh, Scotland.

Turney, P.D. and Halasz, M. (1993). Contextual Normalization Applied to Aircraft Gas Turbine Engine Diagnosis. *Journal of Applied Intelligence* 3, pp.109–129.

Watrous, R.L. (1993). Speaker Normalization and Adaptation Using Second-order Connectionist Networks. *IEEE Transactions on Neural Networks* 4(1), pp.21–30.

Watrous, R.L. and Towell, G. (1995). A Patient-adaptive Neural Network ECG Patient Monitoring Algorithm. In *Proceedings Computers in Cardiology 1995*, Vienna, Austria.

Widmer, G. and Kubat, M. (1992). Learning Flexible Concepts from Streams of Examples: *FLORA2*. In *Proceedings of the 10th European Conference on Artifical Intelligence (ECAI-92)*, Vienna. Chichester, UK: Wiley.

Widmer, G. and Kubat, M. (1993). Effective Learning in Dynamic Environments by Explicit Context Tracking. In *Proceedings of the 6th European Conference on Machine Learning (ECML-93)*, Vienna. Berlin: Springer Verlag.

Widmer, G. (1994). Combining Robustness and Flexibility in Learning Drifting Concepts. In *Proceedings of the 11th European Conference on Artificial Intelligence (ECAI-94)*, Amsterdam. Chichester, UK: Wiley.

Widmer, G. and Kubat, M. (in press). Learning in the Presence of Concept Drift and Hidden Contexts. *Machine Learning* (in press).