Recurring Hidden Contexts in Online Concept Learning

Stefan Mandl and **Bernd Ludwig** and **Sebastian Schmidt** and **Herbert Stoyan** [1]

Abstract. Learning systems in dynamic environments have to be able to process examples that occurred in different hidden contexts. We review several approaches to hidden context aware concept learning systems and question the appropriateness of the Calendar Apprentice domain as a real world benchmark for hidden context aware learning systems. By providing a simple lower error bound for batch learning systems in the presence of hidden contexts, we give reasons for the absence of such an agreed upon benchmark. Finally, we present a new hidden context aware concept learning algorithm, which is evaluated in a modified version of the synthetic Stagger domain.

1 Introduction

Online Learning, hence the acquisition and incorporation of new training examples over the course of time during system operation, is the mode of learning that comes closest to the learning style of human beings. It shall not be confused with the term ‘online update’, which means that examples are not stored for later reconsideration. If learning is considered a defining aspect of intelligent systems, then it is obviously online learning that is being meant. In disciplines like Electrical Engineering, *being intelligent* basically means that the system *adjusts itself during operation*. The alternative learning style, batch learning, occurs at the building phase of a system. Once deployed, it does not learn anymore. Thus, from an observer’s point of view, a system built by means of batch learning is no different from a system with behavior specified by hand completely. The term ’batch learning’ shall not be confused with ’batch update’, which means that a learning algorithm considers all training examples before updating the hypothesis. Actually, learning algorithms using ‘batch update’ can always be used in online learning situations by completely retraining the model, once a new example is presented to the system. Thus, batch and online learning characterize the learning behavior that is visible from an external point of view, while ‘batch update’ and ‘online update’ are concerned with internal aspects of a learning algorithm. In this paper, when talking about batch vs. online learning, we take the external point of view.

There are (among others) two problems related to online learning that have been identified by the Machine Learning community: learning in the presence of hidden context change, and adapting to drifting concepts. While the respective reasons for these two problems are different, concept drift and hidden context change cause similar problems and have been dealt with by the same basic strategy: only considering a subset of all available examples; where this subset very often is a selection of the most recent examples. Clearly, this simple approach is not optimal as fewer examples in general produce classifiers of lower quality. Therefore, there are several approaches that try to reuse old examples.

In this paper, we present a new online concept learning system that builds explicit models of hidden context and is able to notice when a previously identified hidden context recurs. Before classifying a given feature vector with respect to the concept, the system first selects an appropriate context for this classification. From the system’s point of view, a concept is a binary indicator function defined on a set of feature vectors. Our guiding vision is to create systems that will be able to reason about inductively inferred contexts in symbolic terms. A prerequisite for this long-term research goal is context awareness of online learners.

In the next section, we review previous work on learning in the presence of hidden context change. Section 3.1 describes a popular learning domain that has been claimed to be heavily influenced by hidden context. We suggest that even if this claim is true it may not be exploitable by current state of the art hidden context aware learning algorithms. In section 3.2, we give reasons for the absence of standard Machine Learning benchmarks that are suitable to assess the quality of hidden context aware learning algorithms and describe a modified version of the synthetic Stagger domain. In section 4, we describe a basic algorithm that after identifying hidden contexts utilizes some general statistical properties in order to select the current context and evaluate this algorithm in the modified Stagger domain. Section 5 concludes the paper and points at directions of future research.

2 Related Work

Several publications are concerned with learning in the presence of hidden context change and concept drift. The Stagger system is presented in [9]. This synthetic domain is used as a benchmark in various subsequent publications. In [12], Widmer describes a meta-level algorithm for online learning that determines ‘contextual features’, hence features that determine those (possibly) different features that are the most relevant (‘predictive’ in Widmer’s terms) for the classification task. In [3], Harries and Sammut present a form of ‘contextual clustering’ that finds stable concepts (actually stable concept descriptions) in a changing environment. Their approach is interesting for us, as identifying presumable hidden context changes is a necessary feature of any hidden context aware learning system. The problem of finding real world training data that is actually collected while changes in the hidden context occur is mentioned in [11]. Keeping a window of ‘current’ examples in memory is an often applied technique and is for example used in [13]. An ensemble algorithm that behaves as if implicitly adjusting the size of a window of last examples is presented in [10] and [4] shows how to directly adjust the window size when needed.

[1] reports results on the Calendar Apprentice domain, which is described in [7]. A nice overview of context in concept learning that

[1] University of Erlangen-Nuremberg, Germany, email: {mandl, bernd.ludwig, sebastian.schmidt, stoyan}@informatik.uni-erlangen.de

also mentions the Calendar Apprentice domain can be found in [6]. [5] describes an algorithm that besides considering a window of recent examples in order to adjust to concept drift also reconsideres previous examples. The evaluation is conducted in the Calendar Apprentice domain.

3 Online Learning Benchmarks

In order to evaluate empirical results of our algorithm, real world benchmarks are of highest interest. This section describes the Calendar Apprentice Domain, and investigates on why we do not think that CAP is suitable as a benchmark for hidden context aware online concept learning algorithms. The second part of this section explains why there is need for a real world benchmark specifically designed for learning situations with hidden context change. In lack of such a benchmark, we introduce a modified version of the Stagger domain that allows to change the feature distributions alongside the concept definitions in order to capture real world context changes more adequately.

3.1 Case Study: The Calendar Apprentice Domain

Mitchell et al.'s Calendar Apprentice Domain [7] (CAP) is one of the valuable sources for real life concept learning data. The task was to recommend the day, time, and place of a meeting with a certain person or group of persons by consulting previous decisions. The original approach was to collect training examples during the day and to improve the decision making system during the night by maintaining a list of rules that performed well in the past. New rules are generated by training a decision tree on the complete data set and transforming paths from the root to leafs into decision rules. The CAP authors were not satisfied with the system's performance and found that the performance decreased at semester boundaries. This observation has been confirmed in [3]. Figure 1 shows the performance diagram of the C4.5 [8] decision tree algorithm for the concept *req-start-time*. Examples are made available sequentially. and each time an example is added, the model rebuilt completely from scratch. The performance is determined by the percentage of correct classifications on ten subsequent examples that are not known yet. Please note that in order to make the diagrams more readable, we applied a rolling window of size five as a high frequency filter to smoothen the lines. The full information decision tree's performance is compared to a decision tree that is built from the last 30 examples (window approach).

The diagram shows — as expected — that there are regions of good and regions of bad performance. What strikes us about this diagram is the fact that both, the full information and the window restricted versions are affected by the performance declines. If these declines were due to hidden context changes, we would expect that the window based learner was able to recover more quickly than the full memory learner during these regions. There are two possible interpretations: (1) these are very irregular regions in the data set and no good decision trees can be built there, or (2) the number of training examples in these low performing regions is too low. In order to check the latter interpretation, we collected all the poorly (performance below 60%, denoted as CAP$_{<60\%}$ in table 1) classified examples into a new data set and trained those in isolation. Figure 1 shows the correctness of a full memory online learner and a traditional batch-style (using twofold cross-validation) learner on this subset, as well as as on the subset of well (performance greater or equal to 60%, CAP$_{\geq 60\%}$) classified instances, alongside the numbers on the complete data set.

	CAP$_{\text{complete}}$	CAP$_{\geq 60\%}$	CAP$_{<60\%}$
Batch C4.5	53.68	83.01	33.76
Online FM C4.5	43.79	68.43	25.85

Table 1. Performance of batch C4.5 (using twofold cross-validation) and online C4.5 with full memory on various subsets of the training data

These number suggest that the low online learner performance on the $CAP_{<60\%}$ subset is not due to previously seen examples but inherent in that subset. This interpretation does not invalidate previous claims that these regions are influenced by hidden context changes, it only suggests that the hidden context changes caused examples that do not generalize at all. Of course, by simply ignoring the examples in the low performing regions, an online learner could raise its overall performance, but if these low performance regions last several days, this is only little comfort to the user.

3.2 The Need for a Better Benchmark

As the previous section revealed, we are not happy with the Calendar Apprentice as a standard benchmark for hidden context aware online learning algorithms. We considered several other data sets from the UCI Repository [2] but found none suitable to test our algorithm for learning recurring hidden contexts. There are two explanations for this fact: (1) learning data with recurring hidden contexts is not interesting for practical applications or (2) nobody publishes results on such data sets. In order to support the latter explanation, in this section, we are going to provide a lower bound of the error rate of any batch learning algorithm in the presence of hidden contexts. First we have to elaborate on the source of hidden contexts in training examples.

By definition, a hidden context contains state of affairs that are not observable by the learning system but affect whether or not a given data item belongs to the concept. A typical setting that causes hidden contexts in training data is given when a human trainer 'labels' training examples presented to the system. In almost every case, the human trainer does not even look at the features in detail but will decide on the labels by considering the way the features are generated. For instance, when the system is to learn the concept *red-car*, the trainer will put red cars of various kinds in front of a camera. The example features are then created by applying image segmentation and other techniques. Hence, the system and the human observer actually have a very different view of the scene. Generally, the human observer also has background knowledge that helps in the labeling process. In the following, we focus on the classification error that is due to hidden contexts. Thus, providing a lower bound for the total classification error not taking noisy data, or errors caused by mislabeled examples into account.

Let X denote the attribute space of the training examples and X^* denote an hypothetical attribute space that contains enough attributes to describe the concept unambiguously. The set of attributes in X^* is a superset of the attributes in X. Then during the acquisition of a new training example, there occurs a projection from X^* into X. This projection may reduce different items in X^* to the same item in X, which do not necessarily have the same label. Let $p_p(x)$ denote the probability x is the projection of a positive example and $p_n(x) = 1 - p_p(x)$ denote the probability that x is the projection of a negative example. Let furthermore $p(x)$ denote the probability that x is presented to the system and $h(x)$ the hypothesis of x learned by a learning algorithm. Then the probability of misclassification $pm_h(x)$

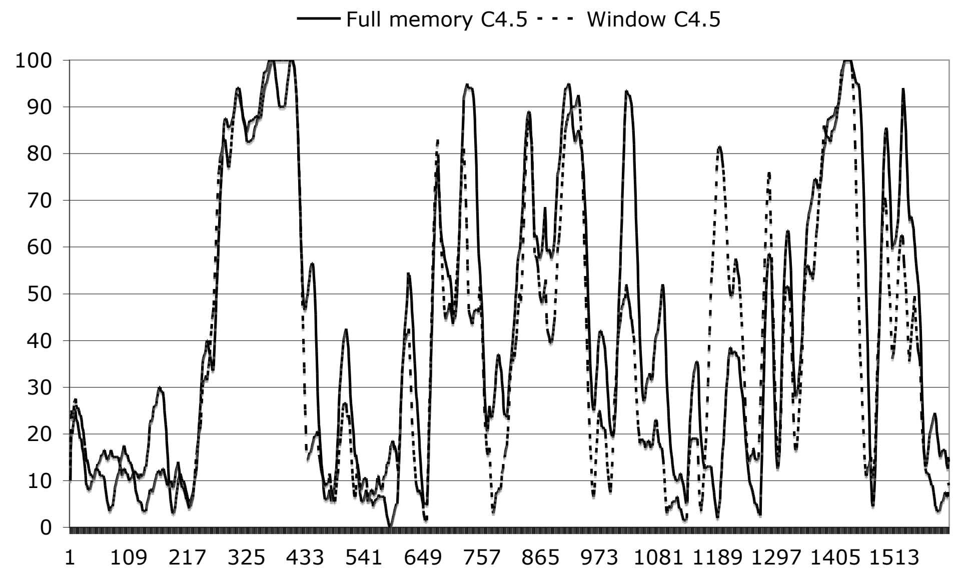

Figure 1. Online learning performance (percent correct on the next 10 examples) of the full memory C4.5 and a window restricted C4.5 with window size 60

is given by

$$pm_h(x) = \begin{cases} p_n(x) & \text{if } h(x) \text{ is positive} \\ p_p(x) & \text{if } h(x) \text{ is negative} \end{cases}$$

If it is possible to integrate over X, the total classification error of a hypothesis h is bound by

$$1 \geq Err(h) \geq \int_X pm_h(x) \cdot p(x) dx$$

As an illustrative example, consider X to contain a single discrete attribute *hour* with the values $\{10, 18, 23\}$ and X^* to contain an additional nominal attribute *place* with the values $\{$*ho(me), off(ice), out*$\}$. The following table describes the concept *late*:

hour	10	10	10	18	18	18	23	23	23
place	*ho*	*off*	*out*	*ho*	*off*	*out*	*ho*	*off*	*out*
late	*no*	*no*	*no*	*no*	*yes*	*no*	*yes*	*yes*	*no*

The probabilities for the observed hours are given in the next table:

hour	10	18	23
$p_p(x)$	0	$1/3$	$2/3$
$p_n(x)$	1	$2/3$	$1/3$

The best hypothesis h derivable on X is given by the following table:

x	$h(x)$
10	*no*
18	*no*
23	*yes*

Let's assume — for illustrative purposes — that $p(x)$ is uniformly distributed, hence $p(x) = 1/3$ for every $x \in X$. Then the error of h is

$$Err(h) = 0 \cdot \frac{1}{3} + \frac{1}{3} \cdot \frac{1}{3} + \frac{1}{3} \cdot \frac{1}{3} = \frac{2}{9} \approx 22\%$$

This simple example highlights that even for extremely simple learning problems, the presence of a hidden context leads to bad results that in a batch learning scenario cannot be improved.

Therefore — to continue our previous argument — it is not surprising that hardly any published results are influenced by hidden context problems [11]. Actually, the standard way to deal with hidden contexts is to remove them. If a learning algorithm performs poorly on a given data set and one suspects the reason to be the presence of hidden contexts, the best practice is to take more attributes into consideration, making the previously hidden context explicitly available. While this strategy is best for offline learning tasks, it is not realistic for online learning intelligent systems that are employed in ever new situations. Instead, they have to deal with the current state of affairs and make arrangements to cope with hidden context changes.

The previous considerations do not leave much hope that there exists data for batch learning algorithms that is suitable as benchmark for hidden context aware online learning systems.

Therefore, in order to evaluate our learning algorithm, we chose the second best method: testing on synthetic data. The domain chosen, Stagger, is the standard domain considered in papers on concept drift and learning in the presence of hidden context change. In Stagger, three nominal attributes are considered: *color* $\in \{$*red, green, blue*$\}$, *size* $\in \{$*small, medium, large*$\}$, and *shape* $\in \{$*circle, triangle, rectangle*$\}$. Concepts in Stagger are defined by proposition logical sentences like: *red*$\wedge$*small* or *medium*$\vee$*large*. When generating training examples, random attribute values are generated and then classified according to a given Stagger-Concept. To simulate concept drift or hidden context change, the used Stagger-Concept is changed while generating new examples. We build upon that practice but in addition to changing the concept definition, we also change the distribution by which examples are generated. This is done in order to reflect the idea that different contexts may put constraints on the probability that certain values show up.

4 A Hidden Context Learning Algorithm

In section 3.2, we provided a general bound for the error of batch learning algorithms in the presence of hidden context changes. Online learning systems can do better, because in online learning, the order in which the examples are generated depends on some process the observing system is undergoing in the real world. For example, time changes gradually, the change of place is constrained by physics and the public transportation system. The assumption made by our algorithm is that during this external process, the distribution of the observed features does change. For example, looking at ones clock may occur more often when being at office in late hours than when being out, hence if the observation (*hour=18*) is made frequently, this might be an indication for the unobserved attribute *place* to have the value *office*.

Given a list of online examples (feature vectors labeled with class values), the algorithm performs the following tasks:

1. Identification of positions in the sequence of examples where hidden context change occurs.
2. Extension of each example feature vector with an additional context attribute.
3. Training of a classifier for the class value given the extended examples. (Base level classifier)
4. Creation of a classifier for the context attribute. (Context classifier)

In the first step, we want to find points in time (actually position indexes) in the sequence of examples where changes in hidden context occur. Our strategy is simple: we are using two windows. One window contains the examples ranging from the last discovered context change (or the first example, if no hidden context change has been detected yet) to the current example, the other window has a fixed size, containing the 30 most recent examples. The idea is that if the classification error in the fixed size window is noticeably lower than in the other one, this indicates hidden context change. We found that a 20% better performance of fixed size window compared to the performance of the other window is a good value for the Stagger domain. Performance is determined by testing on the next ten subsequent examples. By this process, for a given sequence of examples, we compute a sequence of change points.

In the second step, our algorithm extends the example data by adding an extra nominal attribute to each feature vector. This attribute is meant to represent configurations of hidden parameters, hence hidden context. Its values are arbitrarily chosen: $\{c_0, c_1, \ldots, c_n\}$ with n being the number of change points discovered in step 1.

In the third step, we train a classifier for the concept on the examples that have the freshly introduced context parameter available.

In the fourth step, we train a classifier for the newly added attribute. In order to do so, features are computed on all the examples that are assigned the same hidden context value c_i during step 1. In our implementation, we use attribute-value histograms as features (see figure 2). The number of occurrences of each attribute-value combination for blocks of ten training examples from the group of examples having the same hidden context value is recorded and used to train a decision tree (C4.5) for classifying the hidden context attribute. Figure 3 shows a decision tree to classify the hidden context, which was built from example data generated in the modified Stagger domain.

At run time, when a new feature vector is exposed to the system, it has to do the following steps:

1. Maintain an 'instance buffer' containing the most recent examples (in our experiments, we used the ten most recent examples) in order to compute the current histogram.
2. Given the current histogram, use the context classifier to get the current value of the context attribute.
3. Extend the feature vector to be classified to contain the current context.
4. Use the base classifier in order to obtain the system's output for the class value.

One novelty of this approach is to explicitly represent hidden context by additional attribute(s). We think that this is a necessary foundation for systems that could actually reason about hidden contexts.

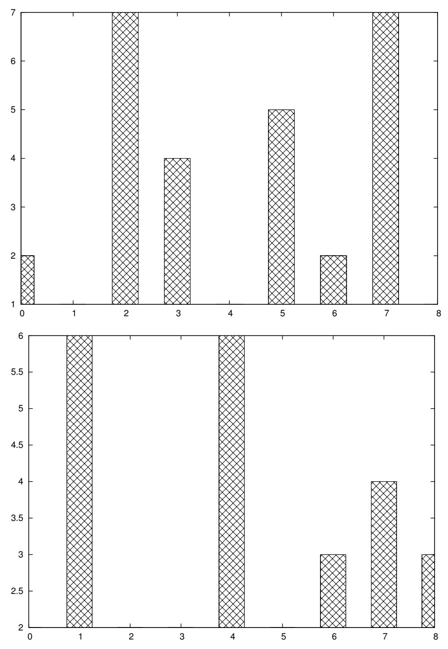

Figure 2. Characteristic feature histograms recorded in different contexts in the modified Stagger domain

We evaluated the algorithm on a modified Stagger domain containing three contexts. c_0 with the concept definition *color=red*$\wedge$*size=small*, c_1 with *color=green*$\vee$*shape=circle*, and c_2 with *size=medium*$\vee$*size=large*. In each of these contexts, we used different distributions for the attribute values. Like in batch learning, we provided a training set and a test set. This is similar to an online learning scenario like training spam mail classifiers, where after a training phase in which the examples are collected online, the system should operate without further user feedback.

The standard C4.5 implementation scores 76.59% correctness on the test set, the hidden context learning algorithm (also using C4.5 for both, base level classification and context classification) achieves 82.83% correctness on the test set.

This result shows that classifiers that are sensitive to hidden context change in examples collected in an online fashion potentially outperform other learners. In addition, it emphasizes our previous consideration that the important quality of online learning — as far as hidden context changes are concerned — is the fact that the outside world puts constraints on the order of the examples.

As our algorithm detects hidden contexts seen during training that reoccur at runtime, one drawback is that it does not detect hidden contexts that reoccur during training already.

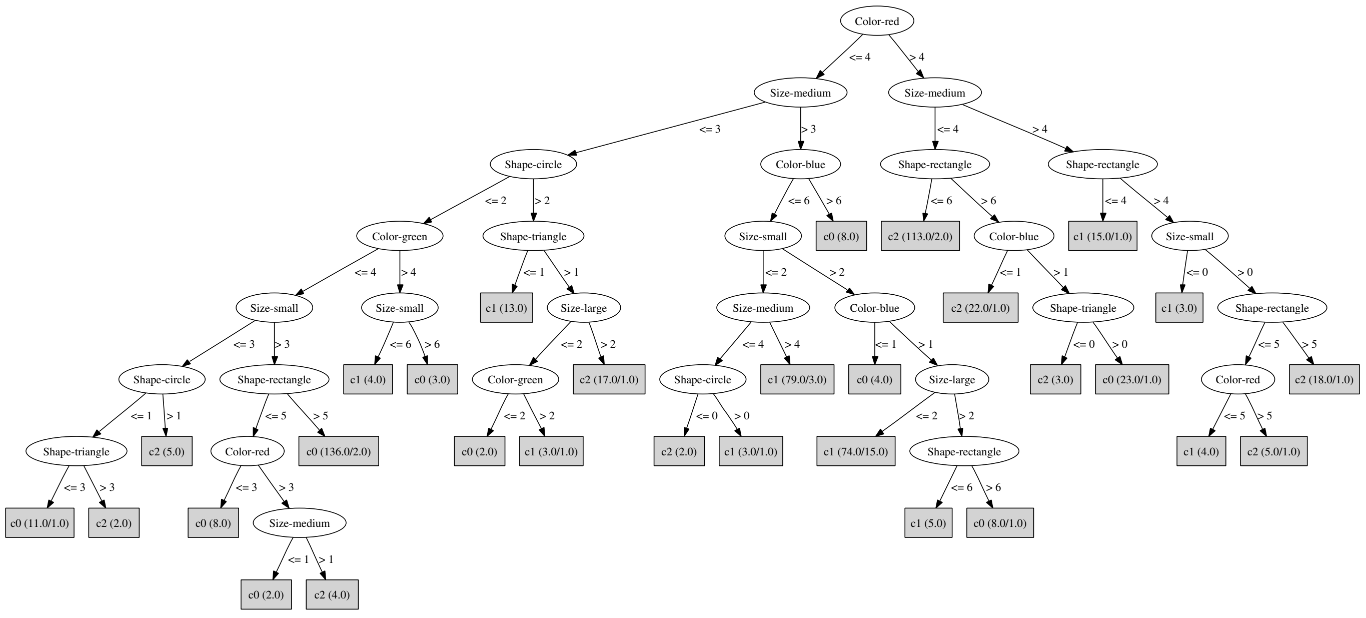

Figure 3. Decision tree that selects the context given a histogram

5 Conclusion

We argued that online learning systems have to be able to operate in the presence of hidden contexts. We claimed that the Calendar Apprentice domain is not suitable as a benchmark for such systems as, while influenced by hidden context changes, these do not seem to be exploitable. As our lower bound for the training error of batch-style learners suggests, there is little hope to find an appropriate benchmark data set in the standard repositories. In order to test our new hidden context learning algorithm, we modified the well known Stagger domain by allowing different contexts to show different feature distributions. The presented hidden context learning algorithm exploits these characteristics by using a meta-level learner that is trained on feature histograms. The evaluation in the modified Stagger domain proves the concept even in an online learning situation without feedback for every classification task. In the future, we plan to enhance the implemented algorithm by merging compatible contexts. In addition, we will try to establish a new standard benchmark for hidden context aware concept learning algorithms.

ACKNOWLEDGEMENTS

We would like to thank the referees for their comments which helped improve this paper. Günther Görz, Bernhard Schiemann, and Peter Reißprovided valuable hints.

REFERENCES

[1] Avrim Blum, ‘Empirical support for Winnow and weighted-majority based algorithms: results on a calendar scheduling domain’, in *Proc. 12th International Conference on Machine Learning*, pp. 64–72, San Francisco, CA., (1995). Morgan Kaufmann.

[2] C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.

[3] M. Harries, K. Horn, and C. Sammut, ‘Learning in time ordered domains with hidden changes in context.’, in *Papers from the AAAI 1998 Workshop on Predicting the Future: AI Approaches to Time-Series Problems*, pp. 29–33. AAAI, (1998).

[4] Ralf Klinkenberg and Thorsten Joachims, ‘Detecting concept drift with support vector machines’, in *ICML ’00: Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 487–494, San Francisco, CA, USA, (2000). Morgan Kaufmann Publishers Inc.

[5] I. Koychev, ‘Learning about user in the presence of hidden context’, in *In Proceedings of Machine Learning for User Modeling: UM-2001*, (2001).

[6] Stan Matwin and Miroslav Kubat, ‘The role of context in concept learning’, in *In Proceedings of the ICML-96, Workshop on Learning in Context-Sensitive Domains, Bari, Italy*, pp. 1–5, (1996).

[7] Tom M. Mitchell, Rich Caruana, Dayne Freitag, John McDermott, and David Zabowski, ‘Experience with a learning personal assistant’, *Communications of the ACM*, **37**(7), 80–91, (1994).

[8] J. Ross Quinlan, *C4.5: programs for machine learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[9] Jeffrey C. Schlimmer and Jr. Richard H. Granger, ‘Incremental learning from noisy data’, *Mach. Learn.*, **1**(3), 317–354, (1986).

[10] Kenneth Stanley. ‘learning concept drift with a committee of decision trees’, Department of Computer Sciences, The University of Texas at Austin, Technical Report AI-03-302, September 2003.

[11] Alexey Tsymbal. The problem of concept drift: definitions and related work,available at http://www.cs.tcd.ie.

[12] Gerhard Widmer, ‘Tracking context changes through meta-learning’, *Machine Learning*, **27**(3), 259–286, (1997).

[13] Gerhard Widmer and Miroslav Kubat, ‘Learning flexible concepts from streams of examples: Flora2’, in *ECAI ’92: Proceedings of the 10th European conference on Artificial intelligence*, pp. 463–467, New York, NY, USA, (1992). John Wiley & Sons, Inc.