

Learning Concept Drift with a Committee of Decision Trees

Kenneth O. Stanley (kstanley@cs.utexas.edu)
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712 USA

Abstract

Concept drift occurs when a target concept changes over time. I present a new method for learning shifting target concepts during concept drift. The method, called Concept Drift Committee (CDC), uses a weighted committee of hypotheses that votes on the current classification. When a committee member's voting record drops below a minimal threshold, the member is forced to retire. A new committee member then takes the open place on the committee. The algorithm is compared to a leading algorithm on a number of concept drift problems. The results show that using a committee to track drift has several advantages over more customary window-based approaches.

Contents

1	Introduction	3
2	Concept Drift and the CDC Algorithm	4
2.1	Definition of Concept Drift	4
2.2	The CDC Algorithm	4
3	Experimental Evaluation	7
3.1	Experimental Methodology	8
3.1.1	Instantaneous Concept Change	8
3.1.2	Moderate and Slow Concept Drift	9
3.2	Results	10
3.2.1	Instantaneous Concept Change	10
3.2.2	Moderate Concept Drift ($\Delta x = 100$)	11
3.2.3	Slow Concept Drift ($\Delta x = 200$)	13
3.3	Discussion	14
4	Related Work	15
5	Future Work	16
6	Conclusion	16

1 Introduction

When a classifier for a static concept is learned, it can be used to classify future instances indefinitely. However, if the concept can change, the problem of classification becomes more difficult. Learning must continue as long as instances arrive so that the changing concept can be tracked. The presence of a changing target concept is known as *concept drift*.

Concept drift frequently occurs in the real world. People’s preferences for products change. The factors that determine a successful stock change with the economy. When factory conditions change, the process for validating a product changes as well. Many times the cause of change is hidden, leaving the change to be inferred from the classifications themselves. Algorithms that track concept drift must be able to identify a change in the target concept without direct knowledge of the underlying shift in distribution.

Presently, the majority of research into concept drift has been theoretical in nature. Theoretical treatments of the problem generally make simplifying assumptions about the kinds of drift that can occur in order to establish bounds. For example, Helmbold and Long (1994) establish bounds on the *extent* of drift that can be tolerated assuming a permanent and very slight drift, where *extent* is defined as the probability that two successive concepts will disagree on a random example. Bartlett et al. (1996) establish necessary bounds on drift rate and sample complexity for an algorithm to be able to learn the *structure* of a repeating sequence of concept changes. In other words, they show what is necessary in order to learn a sequence of functions determining changing distributions. Other theoretical results establish bounds given assumptions such as known linearity (Freund and Mansour 1997) or slow drift (Barve and Long 1996).

Research into specific algorithms has proceeded to a lesser extent. However, several effective methods exist. For example, Klinkenberg and Thorsten (2000) developed a method for detecting concept drift using support vector machines. Widmer and Kubat (1996) use sets of disjunctive normal form formulae to characterize the current hypothesis. Both of these methods use a *window* to track drift. The idea is to have a window of recent examples that ideally reflect the distribution of the current examples. The algorithms adjust window size as the target concept changes.

In this paper, I present an alternative to using window size to track concept drift. The method presented is called *Concept Drift Committee* (CDC). While changing window size requires heuristics to decide when the window size should change, CDC requires no such heuristics. Instead, a committee of decision trees is maintained, each with a vote weighted according to their recent record. When a committee member’s performance drops too low, it is replaced by a completely new member. Each committee member maintains a hypothesis based on every example seen in its lifetime. Thus, there is no explicit window. However, there is an *implicit* window because when the concept changes, many committee members are forced to retire, and new members begin learning from only the latest examples.

The windowless committee is like a very exclusive clique controlling an advertising agency. The committee tries to stay on top of current trends. The youngest members tend to be valuable during changing times, but the older members are most reliable during times of stability. However, the clique is exclusive because it does not tolerate older members who become mired in their ways. When older members start showing signs of age, they are quickly ejected and replaced with more trend-aware youngsters.

This paper will demonstrate that CDC performs as well as Widmer and Kubat’s window-based method on some problems and better on others. We will examine both sudden and gradual concept drift. Although more work is necessary in realistic domains, these early results establish the promise of using a committee to track concept drift. The main conclusion is that it is not necessary to explicitly detect concept changes and adjust a window in order to successfully predict the target concept. In fact, heuristically adjusting a window can be a disadvantage.

2 Concept Drift and the CDC Algorithm

This section will formally define concept drift and then describe the committee-based algorithm for handling the problem.

2.1 Definition of Concept Drift

A concept is a DNF formula defined over a finite set of binary features. Thus a concept can be something like “big and smart” or “short or smart.” The instance space is defined as all the possible conjuncts of feature values. An instance is either representative of the target concept or not. Thus, the classification of an instance is a boolean value.

Concept drift involves a changing target concept. Consider two target concepts, A and B . A sequence of instances i_1 to i_n are presented in order to the concept drift algorithm. Before some instance i_x , the target concept A is stable and does not change. After some number of instances Δx beyond i_x , the concept is once again stable, this time at concept B . Between instance i_x and $i_{x+\Delta x}$ the concept is *drifting* between targets A and B according to some distribution.

If $\Delta x = 1$ then the concept shifts instantaneously between A and B . We will see how CDC handles instantaneous shifts in the experimental work (Section 3). When $\Delta x > 1$, the concept is changing over a number of instances. We can model the changing concept using the function α , which represents the dominance of concept A over concept B at a specific instance. Thus, before i_x , $\alpha = 1$, and after $i_{x+\Delta x}$, $\alpha = 0$. While the concept is drifting, α is between 0 and 1. The probability that a given instance is in concept A is given by $p(A) = \alpha$. The probability that the same instance is in concept B is thus $p(B) = 1 - \alpha$.

If the current instance i_c appears during a period of drift, we can model gradual concept drift between i_x and $i_{x+\Delta x}$ by setting $\alpha = \frac{c-x}{\Delta x}$. Thus the probability of an instance being in concept A declines linearly as the probability of an instance being in concept B increases until A is completely replaced by B . The shorter the period of drift, Δx , the faster the drift rate. We will see CDC’s performance on $\Delta x = 100$, a moderate drift, and $\Delta x = 200$, a slow drift (Section 3).

The instantaneous and gradual drift problems are both interesting in their own right. A concept drift algorithm may handle sudden changes quite well but have trouble with gradual change. Therefore, it is informative to examine both types of change. In the real world, underlying shifts in concepts can occur in both sudden and gradual ways. For example, someone graduating from college might suddenly have completely different monetary concerns, whereas a slowly wearing piece of factory equipment might cause a gradual change in the quality of output parts. Widmer and Kubat (1996) examine both types of drift in their research, and I will compare my results directly with theirs.

We now turn to the implementation details of the CDC algorithm, which will be used to handle the kinds of drift discussed.

2.2 The CDC Algorithm

The Concept Drift Committee algorithm is motivated by the use of voting committees in methods such as bagging and boosting (Bauer and Kohavi 1999). If a committee is useful for deciding on a classification in a fixed unknown distribution, it should be useful for a drifting distribution as well.

The algorithm works as follows. A committee C is composed of a maximum of n hypotheses, h_1 through h_n . Each hypothesis is a decision tree. On an arbitrary instance i_a , each committee member is the decision tree that is derived from training on every instance it has ever seen. Let the sequence of instances

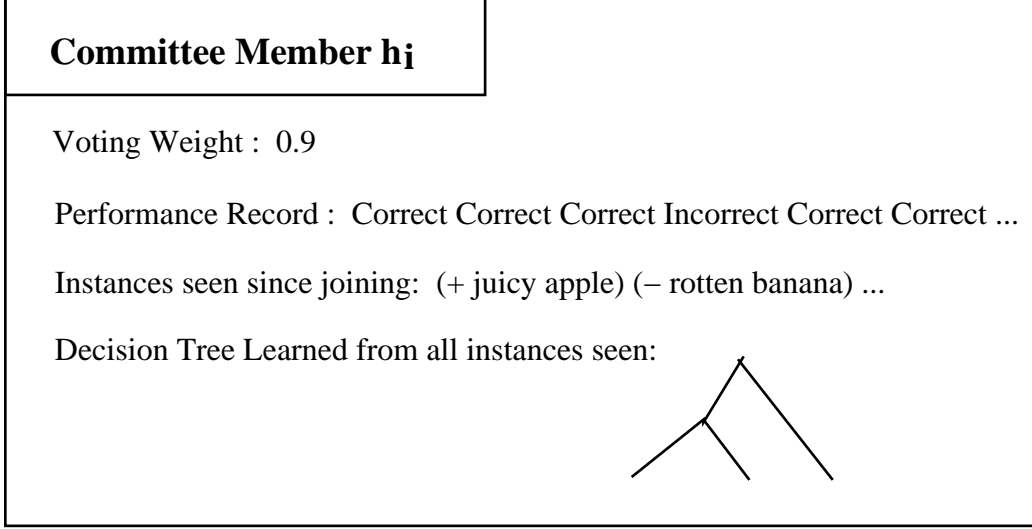


Figure 1: **Sample Committee Member.** The figure depicts a sample committee member and its internal components. The committee member makes its classifications using its decision tree, which is derived from all the instances it has seen since it joined the committee. The member votes with a voting weight derived from its recent record.

seen by committee member h_i be denoted s_i . Since committee members are introduced at different times, they are all trained on a different number of preceding instances, depending on when they first appeared (figure 1).

For example, let us say that the current instance is i_{25} and that committee member h_2 first appeared on i_{10} . Then the decision tree representing h_2 is trained on s_2 , which contains every instance $i_{10} \dots i_{25}$. Of course there is a computational cost to retraining every committee member each time a new instance arrives. However, because incremental algorithms exist for inducing decision trees, the algorithm can operate incrementally, training each decision tree in the committee on a single new instance each time a new instance arrives. ID5R is an example of an incremental decision tree induction algorithm (Utgoff 1989) ¹.

The committee is initially empty. As the first few instances arrive, the committee must form. Whenever a new instance arrives and the committee has less than its maximum n hypotheses, a single new committee member is added. The new member h_i is started out with its training instances s_i containing only the current instance. Thus, at the first instance, a single committee member h_1 joins the committee and is trained only on i_1 . On i_2 , a new committee member h_2 joins the committee and is trained only on i_2 , whereas h_1 is now trained on both i_1 and i_2 . When the committee reaches its maximum size n , new members are no longer added unless another member is forced to retire.

To test committee members, each member is allowed to vote on testing instances that are derived from the distribution (the target concept) of the current instance. The weight of each vote is the same as the record of the voter on the past n training instances. Thus, committee members that have been doing well recently have more say. When a committee member's record falls below some threshold t , the committee member is retired and replaced by a brand new committee member. Therefore, the committee is forced to be up to date by retiring members that are out of step with the current concept.

¹The code used in the experiments in this paper used ID3, retraining each committee member in batch each time a new instance arrived. However, because ID5R is guaranteed to produce the same decision trees as ID3, the results hold for an incremental version of CDC, which could easily be implemented simply by integrating a version of ID5R.

Because committee members are not reliable before they have seen a certain number of instances, they are assigned a voting weight of zero before they reach this *age of maturity*. The age of maturity is set to equal the size n of the committee, so that in the worst situation (where the entire committee has a bad record) there is always at least one mature member because only one committee member can be forced to retire at any one time. In addition, immature committee members cannot be purged, so they have a chance to see enough instances to learn a reliable concept.

In practice, the committee as a whole becomes mature and remains relatively stable when the target concept is not drifting. However, when the concept drifts there is a great deal of upheaval, with many members retiring. Instantaneous concept changes generally lead to the entire committee eventually retiring and being replaced, whereas gradual drifts allow a group of mature committee members to survive for a time in proportion to α , which determines which target concept is most likely at any given time. The idea is that the composition of the committee should reflect the distribution of α . In addition, because the committee is made up of decision trees, individual members can adapt to some extent to new concepts, although of course they cannot represent contradictory concepts. However, they may be able to change enough to represent some kind of middle ground concept for a duration of the drift that can still be useful. Of course, new members do not have to deal with the problem of reconciling old concepts with new ones, which is why they become increasingly dominant during drift. Both voting weights and retirement affect the balance of power in the vote.

Given a committee C processing instances $i_1 \dots i_{last}$, the CDC algorithm can be summarized in pseudo-code:

- $C \leftarrow NIL$
- Train h_1 on i_1
- Add h_1 to C
- For all remaining instances $i_2 \dots i_{last}$,
 - Let i_c be the current instance
 - Test all $h \in C$ *individually* on i_c , and update the record of each h to reflect the result
 - Update all $h \in C$ by incrementally training on i_c
 - Test current C on a test distribution from the same distribution as i_c ; record performance
 - Purge C : Remove $h_{min} \in C$ only if
 - * h_{min} is mature, and
 - * h_{min} has a performance record below a minimum threshold t , and
 - * h_{min} has the worst performance record in C
 - If $size(C) < n$ then train a new committee member h_{new} on i_c and add h_{new} to C
- Return the committee's testing performance on all instances

This section concludes with an intuitive example. Let us imagine we are tracking the preferences of a young automobile customer. The customer likes the color green, and he also likes big cars. However, before he graduates from college, he can only afford small cars. Thus, he is interested in small green cars.

The CDC committee sees instances of cars that the customer has indicated he might purchase over his college career. Let us say our customer made the following recent selections:

- i_1 Small blue Toyota: not interested
- i_2 Small green Toyota: interested
- i_3 Large blue Honda: not interested
- i_5 Large green Honda: not interested
- i_4 Small green Honda: interested

Now let us say there are 4 committee members, $c_1 \dots c_4$. The oldest committee members, c_1 and c_2 , have the hypothesis “any small green car.” However, c_3 and c_4 have not seen all the instances, so they have different hypotheses based on partial data. c_3 hypothesizes that the customer likes “green Hondas,” whereas c_4 believes the customer likes everything. The mature committee members clearly rule with their correct hypothesis.

However, suddenly our fortunate young customer graduates from college into a lucrative career as a CS391L teaching assistant. His dreams becoming reality, he realizes he can now afford those big green cars that he’s always loved. He now responds to recent questionnaires:

- i_5 Small blue Toyota: not interested
- i_6 Small green Toyota: not interested
- i_7 Large blue Honda: not interested
- i_8 Large green Honda: interested
- i_9 Small green Honda: not interested
- i_{10} Large green Toyota: interested

Suddenly the esteemed committee is in turmoil. Its most respected members, c_1 and c_2 , get i_6 , i_8 , i_9 and i_{10} incorrect. Their records tarnished, they are forced into retirement, quickly replaced by new members. The remaining members from before the graduation perform better. c_4 , who previously believed the customer liked everything, comes to correctly hypothesize that the customer likes “large green cars.” c_3 , who thought the customer liked “green Hondas” performs well until i_9 , lowering its voting weight somewhat, but not quite eliminating it. Thus, the committee is now in a period of transition, with c_4 emerging as the experienced sage and c_3 with a slightly incorrect hypothesis and a lower voting weight. Two brand new members have just joined the committee and are sure to quickly learn the new concept assuming it does not change again. We can see that this committee is already going to vote correctly for large green cars.

Of course the preceding example is oversimplified and does not include gradual drift. We will see how a larger committee performs on more difficult and varied problems in the next section, where I detail the experiments I performed with CDC.

3 Experimental Evaluation

This section addresses the hypothesis that CDC is a powerful algorithm for tracking varied rates of concept drift. The constantly changing committee allows CDC to closely mirror a changing distribution.

Three experiments were performed with CDC:

1. Instantaneous concept change
2. Moderate concept drift
3. Slow concept drift

In order to allow for comparison, I strictly duplicated these experiments as they were performed by Widmer and Kubat (1996) for testing their system, FLORA. It is unfortunate that the experimental paradigm introduced in their paper is not a real-world domain. However, the experiments *do* clearly demonstrate performance in well-defined cases of drift, and therefore give insight into how well concept drift algorithms might perform in the real world. Most importantly, the experiments serve as one of the few standardized benchmarks for comparison in the area of concept drift.

This section begins with descriptions of the experiments followed by results.

3.1 Experimental Methodology

The same CDC settings were used in all experiments. The maximum committee size was 10, the age of maturity for a committee member was 10, and the performance record for a particular hypothesis was taken over the last 10 instances it processed (i.e. the record is a queue of 10 correct or incorrect classifications). The minimum performance level to avoid retirement was 80%.

3.1.1 Instantaneous Concept Change

The instantaneous concept change experiment uses the same concept drift problem as Widmer and Kubat (1996), which originally appeared in Schlimmer and Granger (1986). The problem occurs in a block world with three attributes:

- $size \in \{small, medium, large\}$,
- $color \in \{red, green, blue\}$,
- $shape \in \{square, circular, triangular\}$

Three hidden concepts are used in the experiment:

1. $size = small \wedge color = red$,
2. $color = green \vee shape = circular$,
3. $size = (medium \vee large)$

120 training instances are chosen uniformly from the instance space. The first 40 are labeled according to concept 1, the second 40 according to concept 2, and the last 40 according to concept 3. Thus, 2 instantaneous concept changes occur at instance 40 and instance 80. 100 testing instances are also randomly generated for each experiment. For each training instance, the 100 testing instances are labeled according to the underlying concept of the training instance. The committee is then tested on those 100 testing instances, to give a score out of 100 for the accuracy of the committee at the particular training instance. The testing performance was completely independent from training and was not used to facilitate training in any way.

This experiment shows how CDC compares to the FLORA4 algorithm by Widmer and Kubat. FLORA4 is the best performing algorithm of the FLORA family on this problem. The FLORA algorithms are described in detail in Related Work (Section 4). Roughly, FLORA algorithms operate by keeping 3 groups of descriptors, each represented as a DNF formula, representing accepted descriptors that are believed to be currently correct, negative descriptors that are believed to be always incorrect, and potential descriptors that match some recent negative and positive examples. The algorithms use a variable sized instance window to adjust the sets of descriptors in an attempt to best capture the current underlying concept. The later versions of FLORA use more sophisticated statistical methods to rate reliability of predictors than earlier versions of FLORA. A third algorithm, IB3, is also depicted for comparison (Aha et al. 1991). . FLORA4 borrows the idea of using statistics to check the reliability of a predictor from IB3. Thus, IB3 is a predecessor of FLORA4.

The instantaneous drift experiment shows how quickly the algorithms can react to a sudden change. It is essentially an experiment about *recovery* after a new concept has stabilized, since there is no unstable period.

3.1.2 Moderate and Slow Concept Drift

Other than having different drift rates, the experiments with moderate and slow concept drift both use the same setup. The scenario is taken once again from Widmer and Kubat for the sake of comparison. Two concepts are defined over 6 boolean attributes $\{a_1 \dots a_6\}$:

1. Concept A : $a_1 \wedge a_2$,
2. Concept B : $(a_3 \wedge a_4) \vee (a_5 \wedge a_6)$

Concept A gradually changes into concept B over some period Δx as described in Section 2.1. The rate of the drift is controlled by the duration of the change. In the moderate drift problem, $\Delta x = 100$, and in the slow drift problem, $\Delta x = 200$.

In both problems, an experiment takes place over the course of 500 uniformly selected instances from the instances space. Before the 100th instance, $\alpha = 1$, meaning that concept A is stable as the underlying concept. After the 100th instance, α begins to shift downward such that concept A is more and more likely to be replaced by concept B on any given instance until instance $100 + \Delta x$, at which point concept B is stable. Concept B remains stable until the 500th instance, where the experiment ends.

A test set of 200 instances is also uniformly chosen at the start of an experiment. On each training instance, the 200 test instance are labeled according to the current value of α . In other words, the distribution of the underlying concepts in the test instances reflects the distribution from which the underlying concept of the current training instance was chosen. Thus, the committee’s score reflects how well it captures *the current distribution* α .

It is interesting both to see how an algorithm performs during the period of drift, and how well it recovers after the drift ceases. While a concept is drifting, the maximum possible accuracy drops because the closer α is to 0.5, the less predictable the underlying concept for a particular concept is. Thus, algorithms must take a hit during this drift period. However, once drift stops, the algorithms have a chance to stabilize on concept B . A good algorithm should be able to recover quickly.

The gradual drift experiments also include comparisons with the FLORA family of algorithms.

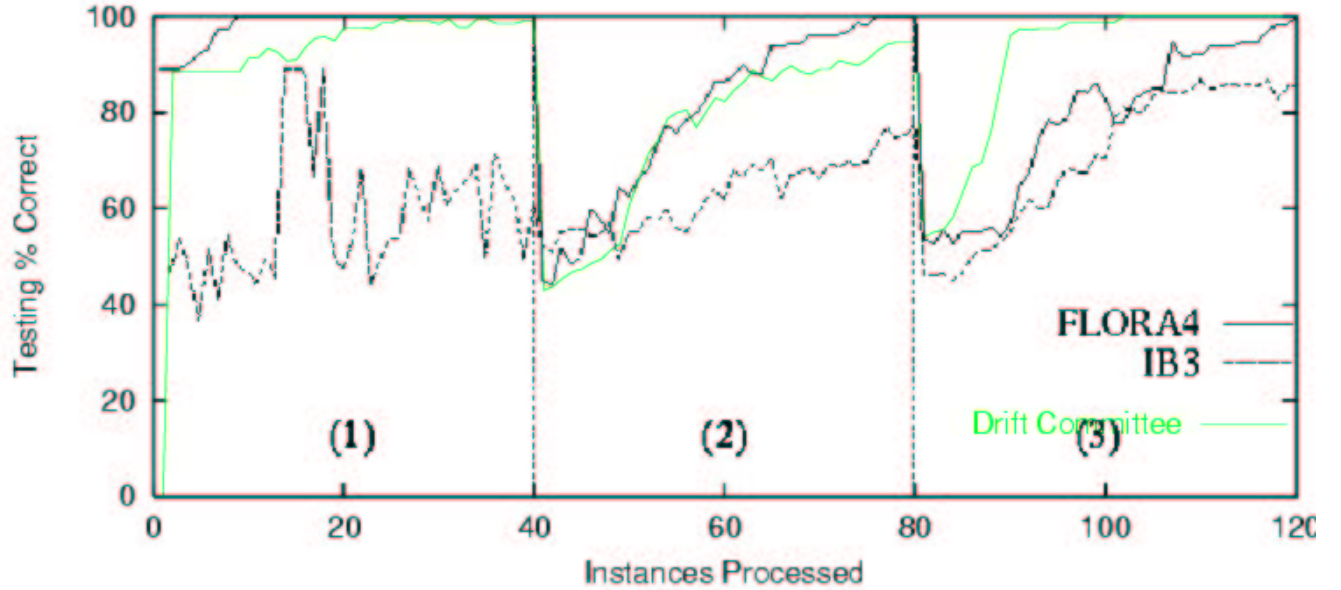


Figure 2: **Instantaneous Change Performance Comparison.** Performance of CDC (Drift Committee), FLORA4, and IB3.

3.2 Results

For each experiment, there are 2 graphs depicting the results. The first graph shows CDC's testing performance compared to other algorithms. CDC is always plotted in green. The second graph is the same as the first, except an additional dotted blue line is plotted. This line is the lower confidence bound for the results of CDC. The reason I include two graphs for each experiment is that the confidence bound can make it difficult to see the other lines (representing averages) in certain cases.

The lower confidence bound is computed using a 1-sample T-value based on the standard deviation and degrees of freedom of the particular experiment and instance. For the first experiment, since differences were pretty dramatic, I show a 99% confidence bound, whereas the second 2 experiments depict 95% confidence bounds. The reason for using confidence bounds is that we do not know the standard deviation or distribution of the data collected on other systems, so we cannot compute a t-test directly. However, if the lower confidence bound on CDC's performance is *above* the mean performance of a competing method, we can be reasonably sure that CDC's performance is superior by a statistically significant margin. Thus, the graphs that include lower confidence bounds are used to easily see statistical significance. The graphs without the bounds are included to make it easier to see the actual mean results.

3.2.1 Instantaneous Concept Change

The instantaneous change experimental results are averaged over 20 experiments for CDC, but over 10 experiments for FLORA and IB3. The plots are divided into three regions, each corresponding to the first, second, or third target concept.

The results indicate that both CDC and FLORA4 perform significantly better than the earlier algorithm IB3. Under the first concept, the problem is not yet a drift problem since nothing is changing. However, we can see FLORA4 performs slightly better than CDC here. After the first sudden change, FLORA4 and CDC

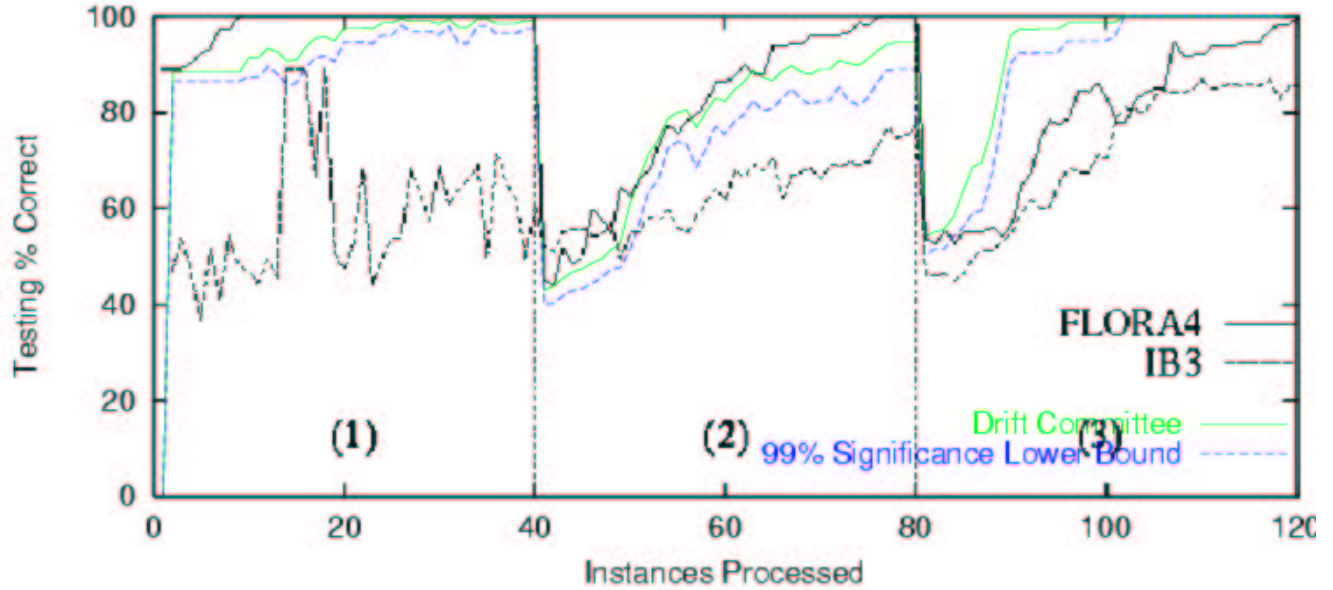


Figure 3: **Instantaneous Change Performance Significance.** Blue line shows 99% confidence lower bound on CDC performance on this task.

both recover in a similar manner. However, FLORA4 reaches a bit higher before the final concept shift. After this second shift, CDC recovers very significantly faster than FLORA4.

The main conclusion is that the results are mixed. FLORA4 seems biased to recovering from different kinds of concepts than CDC. I believe this is due to the form of representation used by the learning algorithms rather than the quality of their drift tracking procedures. CDC has an advantage on the last concept because it is a disjunction of two values for a single attribute, which is easy to represent in a decision tree. However, CDC, using actual DNF expressions, captures the second concept, a disjunction of values for 2 attributes, slightly more easily. It appears that for instantaneous drift, the form of representation may be more of an issue than the drift tracking method, assuming it is of suitable quality (IB3’s tracking method is bad enough that its performance is degraded for all 3 concepts).

It is perhaps more informative in evaluating concept drift tracking methods to compare their performance on actual drift, as in the following 2 experiments.

3.2.2 Moderate Concept Drift ($\Delta x = 100$)

Figures 4 and 5 compare the FLORA family of algorithms to CDC. All plots are averaged over 10 runs. The concept begins to drift at instance 100 and stops drifting at instance 200. The upper line in each plot shows what a method could achieve with perfect information (i.e. both α and what the concepts are). The lower line show the accuracy that would result from the “dumb” method of simply guessing the majority class.

The results show that CDC is on average slightly higher than FLORA4 (the best FLORA) throughout the run. However, the performance of FLORA4 is above the lower 95% confidence bound of CDC, indicating that the results are not statistically significant. The main result is that both CDC and FLORA4 perform similarly on moderate drift, with CDC perhaps a bit more accurate.

It is also informative to observe the performance of CDC in its own right, in order to understand what

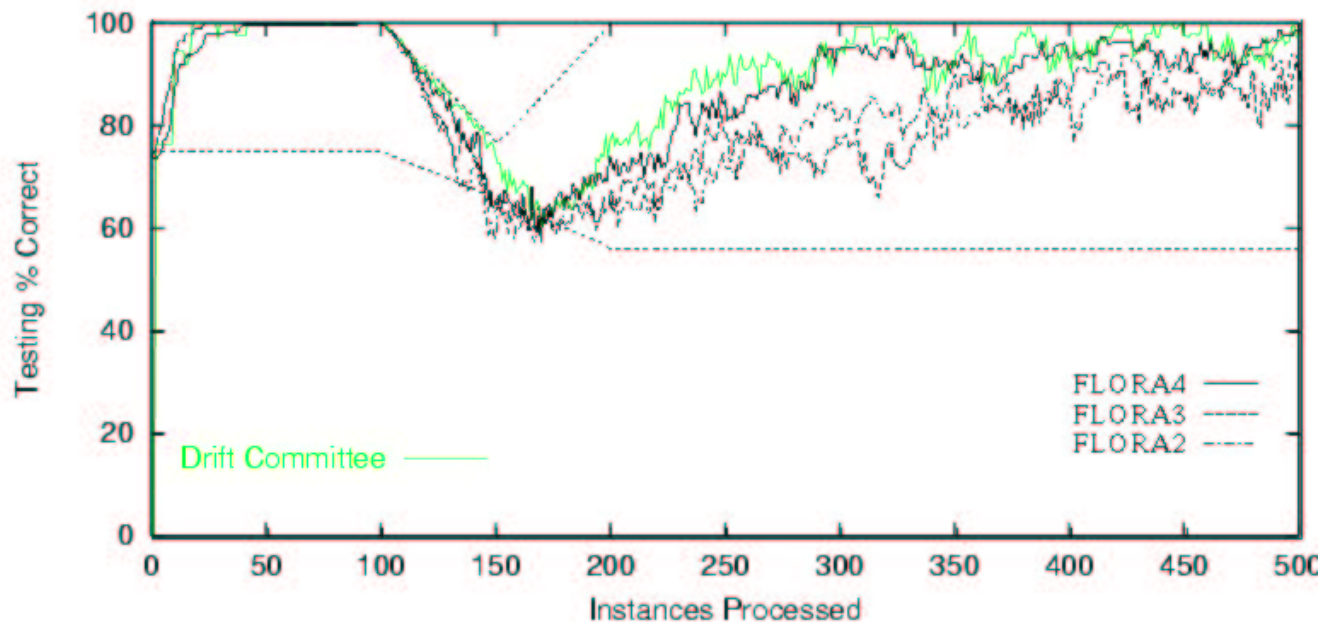


Figure 4: **Moderate Drift Performance Comparison.** Performance of CDC (Drift Committee) and FLORA algorithms.

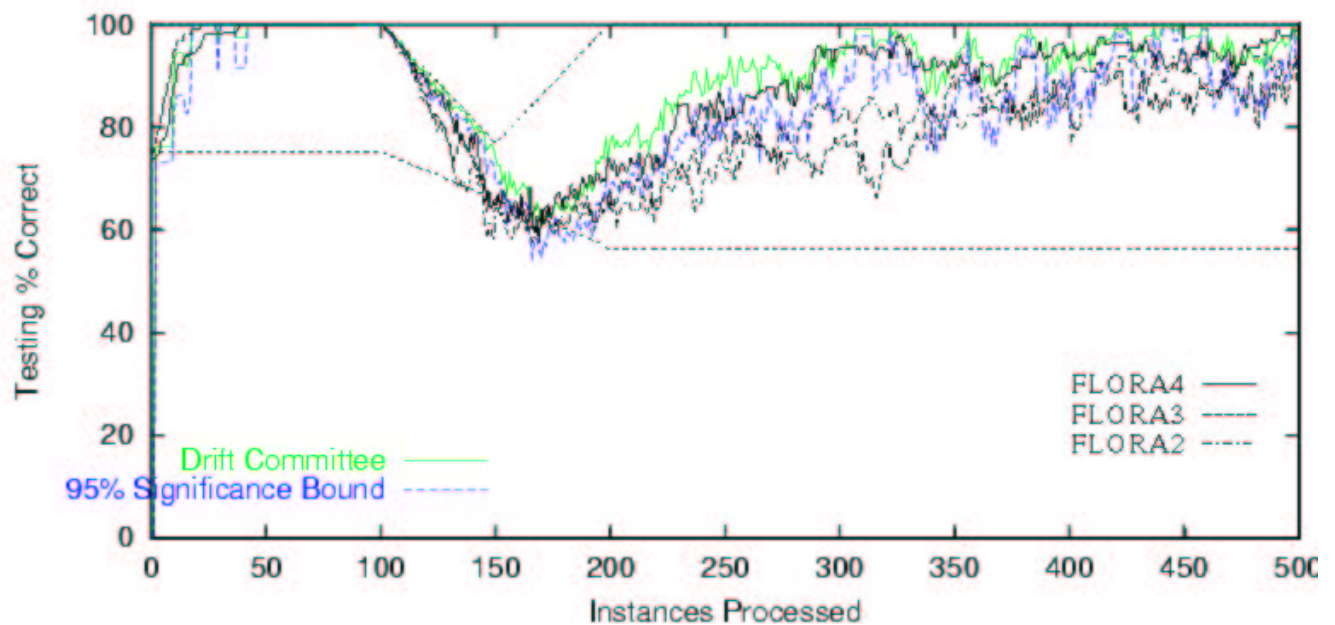


Figure 5: **Moderate Drift Performance Significance.** Blue line shows 95% confidence lower bound on CDC performance on this task.

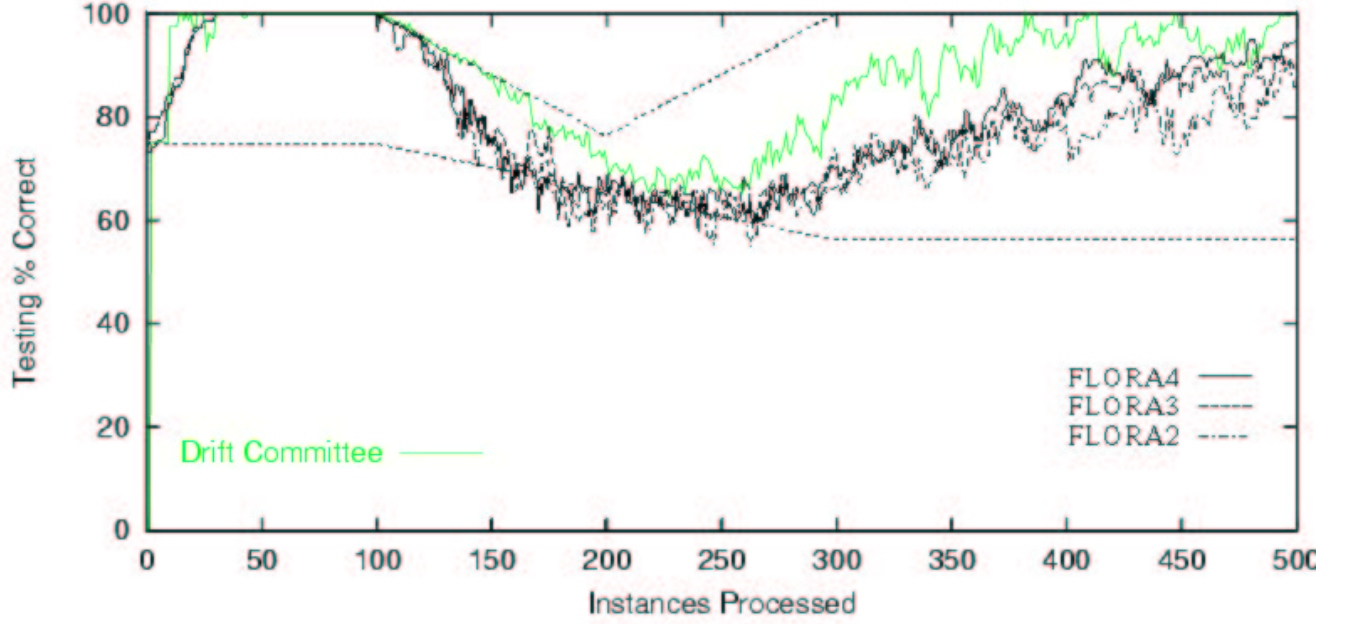


Figure 6: **Slow Drift Performance Comparison.** Performance of CDC (Drift Committee) and FLORA algorithms.

is going on. As the concept gradually shifts towards $\alpha = 0.5$, the committee is continuously purged until it hits bottom at around i_{160} . At that point the concept is stabilizing again at a new target and the committee becomes more and more entrenched in support of the new concept, rising in accuracy as the concept becomes more certain, all the way until it begins to oscillate between 90% and 100%. (CDC oscillates between 80% and 90%)

Why does the committee not completely stabilize on the second concept? I believe the reason is that the second concept is a disjunct of two conjuncts, and since the training examples are chosen randomly a lot of the time it might look like only one of the conjuncts is the actual target concept. In other words, the random chance of which conjunct has recently appeared might look deceptively like a concept change, and some new committee members might join for a brief time with a hypothesis containing only one of the two conjuncts. The initial concept is only a single conjunct and therefore both FLORA and CDC have any easier time stabilizing on it before i_{100} .

The next experiment, slow drift, is interesting because it allows us to see both algorithms attempting to grasp a moving target for an extended period of time. We can clearly see how they temporarily learn to fit the current distribution.

3.2.3 Slow Concept Drift ($\Delta x = 200$)

Figures 6 and 7 show how CDC performs compared to the FLORA family on a drift that occurs between i_{100} and i_{300} .

The advantage of CDC over the FLORA methods is clear throughout figure 6. Figure 7 shows that the difference is significant throughout both the fall and subsequent rise in accuracy from the changing α .

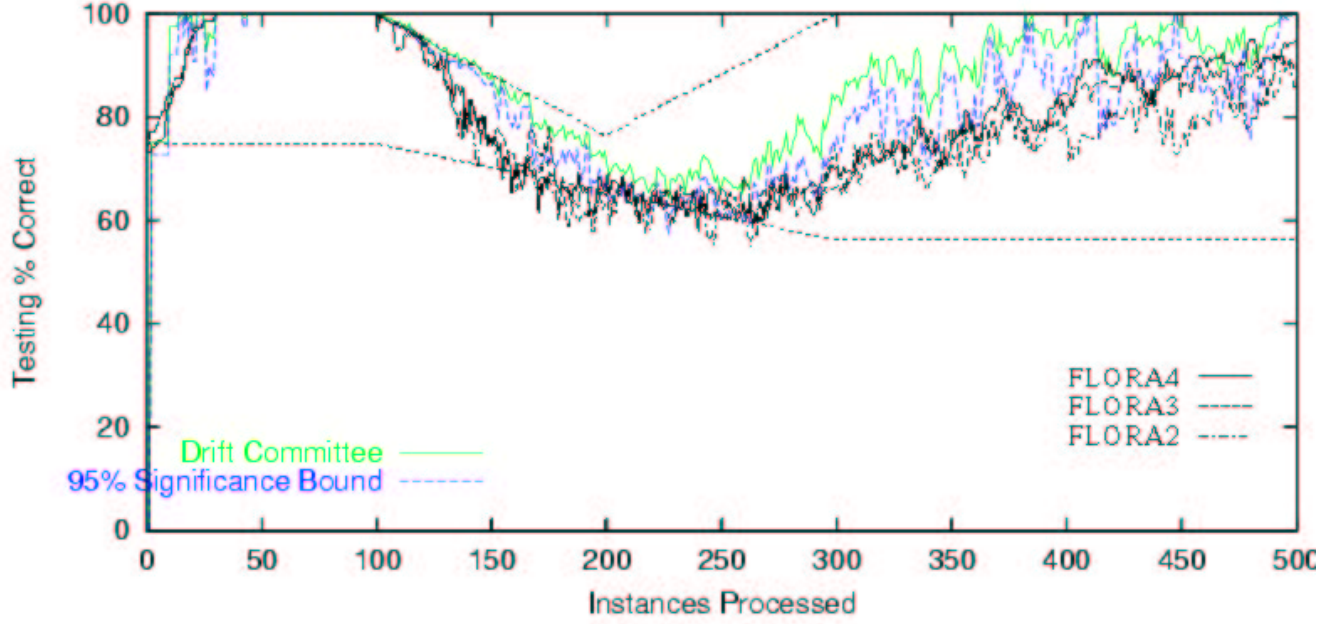


Figure 7: **Slow Drift Performance Significance.** Blue line shows 95% confidence lower bound on CDC performance on this task.

3.3 Discussion

Why is CDC relatively so much better than FLORA on slow drift? FLORA clearly has trouble recovering after such a long drift, barely climbing to the 90% level near the end. FLORA may be handicapped by its use of a window adjustment heuristic (WAH). The algorithm attempts to *detect* concept changes based on the theory that sudden drops in accuracy or a sudden explosion in accepted descriptors indicates a changing concept. The window size is then dropped by 20%. Clearly, this is a rough estimate of the change in duration of the current concept. Why should it necessarily be 20%? Particularly during slow drift, 20% drops in window size may be too radical. CDC, on the other hand, does not use a WAH and is not relying on an ability to detect changes. Instead, CDC has many hypotheses looking at different groups of instances, all competing to best capture the current target concept. Thus there is no need to detect a concept change explicitly, and there is no need for a heuristic to adjust window size.

One way to conceptualize the advantage of CDC is to consider that the more samples one has of an actual distribution, the less variance will occur in the sample distribution. A hypothesis can be compared to a single sample window, with a relatively roughly approximated size. However, because we have multiple hypotheses, the *average* of their respective window sizes is less rough than that of any individual hypothesis. Thus the average size of past instances observed by hypotheses in the committee is an *implicit* window averaged over many sample window sizes and thus less likely to be affected by variance. In addition, because each hypothesis is voting with a weight corresponding to its recent record, the average is weighted according to accuracy, making it even more resistant to variance.

The conclusion is that CDC is making very fine and accurate adjustments to its implicit window, while the FLORA methods make crude 20% adjustments to its explicit window. The result is that 20% adjustments are too severe for a slowly drifting distribution. CDC, on the other hand, can adapt to any drift rate. FLORA is probably by chance biased towards faster drift ($\Delta x = 100$) simply because of its 20% window cutting rate when it discovers drift. CDC is less biased. When drift is instantaneous, the ability of the algorithms to

adjust to drift becomes a moot point and the underlying learning algorithms become more important.

The question remains why FLORA methods fail to fully recover after slow drift ends. FLORA keeps a store of former concepts so that it can reuse expired concepts if they ever recur. I hypothesize that the protracted drift contains periods of instances that look deceptively like a concept change. FLORA tries to remember these deceptive “concepts” and recall them later. However, the supposed concepts being learned are actually just changing distributions of concepts A and B . Thus it may mistakenly be identifying recurring concepts after the drift is already over, because it learned numerous erroneous intermediate concepts.

In conclusion, FLORA’s WAH is rough compared to the weighted averaging of many hypotheses in CDC. The conclusion confirms the hypothesis that CDC is suited to any rate of drift.

4 Related Work

We have seen how the FLORA methods (Widmer and Kubat 1996) compare to CDC. The intuition behind FLORA is that an algorithm needs to be able to decide when a concept is changing. Once the change is detected, it can then change the window of instances it is observing to more accurately encompass the current target concept. FLORA is based on keeping groups of *descriptors* corresponding to accepted descriptors, negative descriptors, and possibly acceptable descriptors. The descriptors are conjunctions, and a set of descriptors can be considered a DNF formula. The current window is adjusted when accuracy suddenly drops, or when accepted descriptors balloons. When this happens, window size is dropped by 20%. If on the other hand the hypothesis is performing very well, the window size is unchanged. If the hypothesis is stable but not performing very well, window size is increased by 1 to incorporate more information.

The basic framework of FLORA as described above is actually the algorithm for FLORA2. FLORA3 elaborates on this idea by keeping old stable concepts around for later use. FLORA3 checks during concept changes whether an old hypothesis matches the current window. Thus, FLORA3 can avoid relearning the same concept over again. FLORA4 elaborates on FLORA3 in an attempt to be more resistant to noise. In FLORA4, accepted descriptors don’t necessarily have to match every positive instance in the window. Instead, FLORA4 attempts to rate the reliability of descriptors statistically.

As we have seen, FLORA’s main weakness is its rigid window adjustment heuristic. CDC instead has an implicit window size based on the average of many hypotheses, allowing it to make more appropriate fine adjustments.

I have claimed that CDC does not really have a window. It may seem that CDC actually has many windows, one for each hypothesis. However, I do not believe these are windows in the usual sense of concept drift algorithms because they are never individually adjusted. Each hypothesis simply learns from every instance it has ever seen, never analyzing or adjusting the instances it observes. It is really through the averaging of all these hypotheses that a kind of weighted window implicitly arises. The weighting comes from the decay of voting weights for hypotheses that are becoming out of date. Thus, CDC entirely avoids both answering the question “is the concept changing?” and “what should I do about it if it is?” The difference between CDC and window-based algorithms can be characterized as implicit versus explicit drift tracking. Another characterization is variance reduction versus precise change detection.

There are additional windowing algorithms to FLORA. Klinkenberg and Thorsten (2000) introduced a window-based drift tracker based on support vector machines (SVMs). The algorithm is able to have a more precise WAH than FLORA because of statistical properties of SVMs that can be theoretically shown to indicate appropriate window adjustments. However, I think the specific reliance on SVMs is a weakness for a concept drift algorithm, because it restricts the kinds of learning methods available for tracking drift. I believe that a drift-tracking strategy should be independent from the concept-learner because what happens

when a vastly superior concept learning algorithm appears and we can't use it because we have to use SVMs? There is nothing in CDC that necessitates the use of decision trees. Thus, CDC can be used with the latest and greatest concept learning algorithm from the future, or domain-specific learning algorithms biased towards particular kinds of concepts that SVMs might not be suited for.

There are some concept drift methods that don't use windows or committees. Salganicoff (1993) used weight decay on experiences based on how close the experience is to *subsequent* experiences. The idea of using weight decay is orthogonal to CDC, and could be incorporated into the CDC algorithm as a supplement². Lanquillon (1999) explored the problem of tracking drift *without* knowing the true labels of instances, which were texts to be classified. Instead, Lanquillon used recall and precision to produce a confidence measure that could be used to track drift. The question of unsupervised concept drift tracking is not addressed by CDC.

In summary, varied methods exist. Among them, CDC has the advantage of being able to precisely align to shifting distributions without being tied to a particular learning algorithm.

5 Future Work

Several areas still need to be addressed. First, it would be informative to test CDC (and perhaps competing algorithms as well) in a real-world domain like product preference tracking or the stock market. Because the stock market involves slow drift, CDC might be particularly well suited for classifying stocks as promising or not.

More experiments need to be performed to understand the contribution of committee size, age of maturity, and voting weights to performance. It is possible that voting weights could be exponentially weighted instead of linearly weighted. Also, does the algorithm become more and more powerful the larger the committee? Where is the point of diminishing returns?

Currently, examples are not weighted in the system, which could be a weakness. Perhaps examples should be weighted for each committee member according to the ones it has gotten wrong in the past, somewhat like in boosting. Or an example-weighting scheme based on similarity could be implemented as by Salganicoff (1993).

CDC should be tested with different learning methods. Particularly in the case of instantaneous concept shifts, decision trees might not be the best method. We would want a learning method that can learn approximations from the least number of examples possible. However, I do think decision trees are useful because of their potential for incremental learning.

Finally, I have not checked to see how robust CDC is with respect to noise. I expect CDC would do very well with noise because of its reliance on averaging, which tends to smooth out uneven distributions. Noisy experiments should be performed and compared with FLORA4, which was designed to be resistant to noise.

6 Conclusion

CDC is a powerful algorithm for tracking concept drift and performs significantly better than a leading algorithm on problems of protracted drift. The use of a committee allows CDC to make fine implicit adjustments to the group of instances from which the committee bases its predictions. The results show that an explicit

²I used my boosted-ID3 code in this project so that weighted examples could easily be added to the algorithm in the future

window may not be the best way to track concepts drift despite its intuitive appeal. In a larger context, the research confirms once again the utility of committees in learning, now in the domain of concept drift.

References

- Aha, D., Kibler, D., and Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.
- Bartlett, P., Ben-David, S., and Kulkarni, S. (1996). Learning changing concepts by exploiting the structure of change. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*. Desenzano del Garda, Italy: ACM Press.
- Barve, R. D., and Long, P. M. (1996). On the complexity of learning from drifting distributions. In *Proc. 9th Annu. Conf. on Comput. Learning Theory*, 122–130. ACM Press, New York, NY.
- Bauer, E., and Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging. *Machine Learning*, 36:105–139.
- Freund, Y., and Mansour, Y. (1997). Learning under persistent drift. In *Learning under persistent drift Computational learning theory: Third European conference*.
- Helmhold, D., and Long, P. (1994). Tracking drifting concepts by minimizing disagreements. *Machine Learning*, 14(1):27–46.
- Klinkenberg, R., and Thorsten, J. (2000). Detecting concept drift with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann.
- Lanquillon, C. (1999). Information filtering in changing domains. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*. San Francisco, CA: Morgan Kaufmann.
- Salganicoff, M. (1993). Density-adaptive learning and forgetting. In *Machine Learning: Proceedings of the Tenth Annual Conference*. San Francisco, CA: Morgan Kaufmann.
- Schlimmer, J. C., and Granger, R. H. (1986). Incremental learning from noisy data. *Machine Learning*, 1(3):317–354.
- Utgoff, P. E. (1989). Incremental induction of decision trees. *Machine Learning*, 4(2):161–186.
- Widmer, G., and Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101.