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

Technical Report UT-AI-TR-03-302

Abstract

Concept drift occurs when a target concept changes over time. We 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 minimum 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 several benchmarks. The results indicate that using a committee to track drift has several advantages over traditional window-based approaches.

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 called *concept drift*.

Concept drift frequently occurs in the real world. For example, 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 it 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.

Much 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) established 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) established 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 showed what is necessary in order to learn a sequence of functions determining changing distributions. Other theoretical results established bounds given assumptions such as known linearity (Freund and Mansour 1997) or slow drift (Barve and Long 1996).

Several practical algorithms also have been developed. For example, Klinkenberg and Thorsten (2000) developed a method for detecting concept drift using support vector machines. Widmer and Kubat (1996) used sets of disjunctive normal form formulae to characterize the current hypothesis. Both of these methods utilize 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. Similarly, Maloof and Michalski (2000) used forgetting to maintain a relevant set of examples. Other algorithms, with well-established mistake bounds, track drift by dynamically adjusting the voting weights of a set of "experts" or "specialists" that each make independent classifications (Auer and Warmuth 1998; Blum 1997; Herbster and Warmuth 1998; Mesterharm 2002). Each expert or specialist is assigned its own small subset of the total set of features, and votes based on experience with only that feature set.

In this paper, we present a novel drift tracking algorithm that also employs weighted voting instead of using a window. The method presented is called *Concept Drift Committee* (CDC). This method differs from other voting schemes in that each member of the committee learns from *all* the features. A committee of independent concept learners is maintained. Each member of the committee casts a vote weighted according to its recent record. When a committee member's performance drops too low, a completely new member replaces it. Each committee member maintains a hypothesis based on every example seen in its lifetime. Thus, there is no explicit window. However, an *implicit window* arises through new members learning only from the latest examples, and through committee members retiring when the concept changes. In the experiments presented in this paper, we used a decision tree as the concept learning algorithm for each committee member, but any supervised learning algorithm could have been substituted.

The windowless committee is like an 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 while the older members are most reliable during times of stability. The clique is exclusive because it does not tolerate older members who become set in their ways. When older members start showing signs of age, they are quickly removed 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, and additional comparisons are needed with other drift tracking algorithms, 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.

We begin with a formal definition of concept drift, followed by a description of the CDC algorithm. Finally, experimental comparisons measure the performance of CDC.

2 Definition of Concept Drift

Let a concept be 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_d, the target concept A is stable and does not change. After some number of instances Δx beyond i_d, the concept is once again stable, this time at concept B. Between instance i_d and $i_{d+\Delta x}$ 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. 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 (Widmer and Kubat 1996). Thus, before i_d, $\alpha = 1$, and after $i_{d+\Delta x}$, $\alpha = 0$. 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_d and $i_{d+\Delta x}$ by setting $\alpha = \frac{c-d}{\Delta x}$. 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. CDC was tested with $\Delta x = 100$, a moderate drift, and $\Delta x = 200$, a slow drift (Section 4).

Instantaneous and gradual drift problems present different challenges. A concept drift algorithm may handle sudden changes 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) examined both types of drift in their research, and we will compare our results directly with theirs.

The next section describes the implementation of the CDC algorithm.

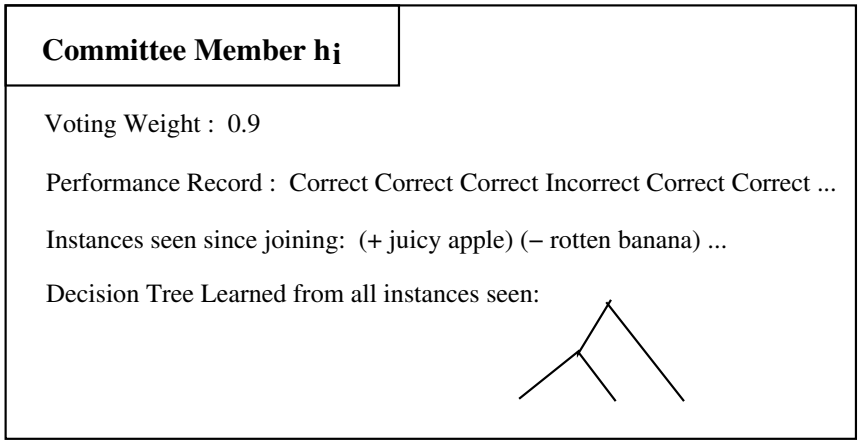

Figure 1: **Sample Committee Member.** A sample committee member and its internal components. The committee member makes its classifications using its decision tree, which is derived from all the positive and negative instances it has seen since it joined the committee. The member votes with a voting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

3 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. In our implementation, each hypothesis is a decision tree. On an arbitrary instance i_a, each committee member is the decision tree derived from training on every instance since it was first introduced. Let the sequence of instances 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, assume 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 $i_{10} \ldots i_{25}$. In the experiments in this paper, decision trees were retrained in batch using ID3. However, because incremental algorithms exist for inducing decision trees, the algorithm can also 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).

The committee is initially empty. As the first few instances arrive, the committee adds members. 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 begins 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 the committee, each member votes on testing instances that are generated 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 recently been doing well in training have more clout.

When a committee member's training record falls below some threshold t, it 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.

Because committee members are not reliable before they have seen a sufficient 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 (i.e. when 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 drift allows 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. New members do not have to reconcile 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 $i_1 \ldots i_{last}$, 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 $i_2 \ldots i_{last}$,
  - Let i_c be the current instance
  - Evaluate all $h \in C$ *individually* on i_c, and update the record of each h to 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 results
  - 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 results on all instances

The main intuition behind the CDC algorithm is that averaging a variety of hypotheses implicitly approximates an accurate window of relevant instances to the current concept. The next section tests this idea.

4 Experimental Evaluation

This section tests the hypothesis that CDC is a promising algorithm for tracking varied rates of concept drift. The constantly changing committee should allow CDC to closely mirror a changing distribution.

Three experiments were performed with CDC: (1) Instantaneous Concept Change, (2) Moderate Concept Drift, and (3) Slow Concept Drift.

In order to allow for comparison, we strictly duplicated these experiments as they were performed by Widmer and Kubat (1996) for testing their system, FLORA. Although their experimental domain is not from the real world, 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. Moreover, this domain serves as one of the few benchmarks for comparison of concept drift algorithms.

The intuition behind the FLORA family of algorithms, which we will compare with CDC, is that a tracker 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 keeps 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 algorithm uses a variable sized instance *window* to adjust the sets of descriptors in an attempt to best capture the current underlying concept. The current window is adjusted when accuracy suddenly drops, or when the set of accepted descriptors balloons. In the latter case, 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 one to incorporate more information.

This basic framework is 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. We compare CDC directly with the results reported by Widmer and Kubat (1996) for the FLORA family of algorithms. Thus, the FLORA results below are from Widmer and Kubat (1996). The results of the entire FLORA family are included.

4.1 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%.

We estimated significance by calculating 95% lower bounds on the average performance of CDC for every instance in every experiment. Although significance data was not available for FLORA, we can infer a significant difference if the performance of FLORA is below the lower bound of CDC performance.

4.1.1 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 each training instance. Testing was completely independent from training and was not used to facilitate training in any way.

This experiment compares CDC to the FLORA4 algorithm by Widmer and Kubat. An even earlier algorithm, IB3, is also included 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. Because there is no unstable period, this experiment reveals only how each algorithm *recovers* after a new concept has stabilized.

4.1.2 Moderate and Slow Concept Drift

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

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. 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 instance 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 increasingly 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.

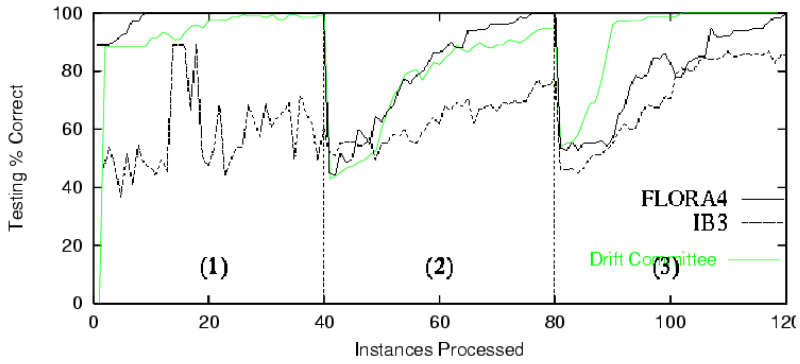

Figure 2: **Instantaneous Change Performance Comparison.** Performance of CDC (Drift Committee), FLORA4 (Widmer and Kubat 1996), and IB3 (Aha et al. 1991). CDC performs significantly better throughout the third concept.

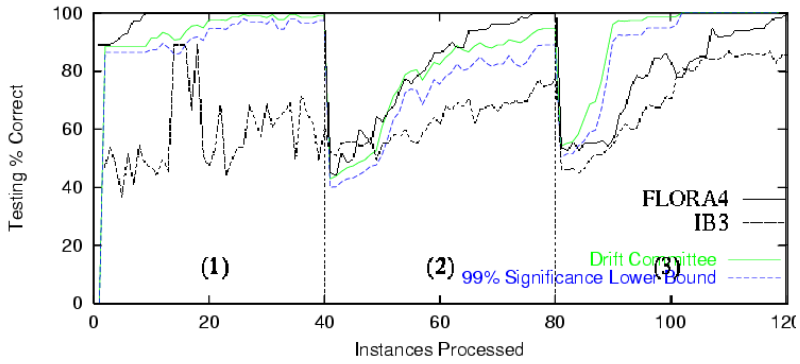

Figure 3: **Instantaneous Change Performance Significance.** The blue line (directly below the CDC line) shows 99% confidence lower bound on CDC performance on this task.

A test set of 200 instances is also uniformly chosen at the start of an experiment. On each training instance, the 200 test instances 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 is. Thus, algorithms must lose accuracy 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.

4.2 Results

4.2.1 Instantaneous Concept Change

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

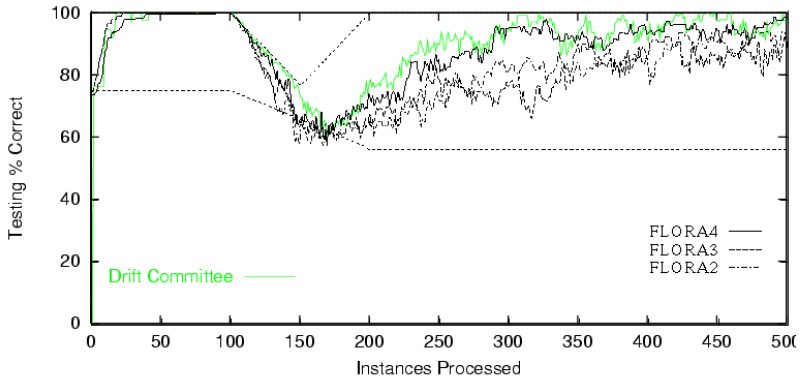

Figure 4: **Moderate Drift Performance Comparison.** Performance of CDC (Drift Committee) and FLORA algorithms (Widmer and Kubat 1996) on moderate drift ($\Delta x = 100$). Although CDC performance is slightly higher than FLORA4, this difference is not statistically significant.

The results (figure 2) indicate that both CDC and FLORA4 perform significantly better than the earlier algorithm IB3. Figure 3 shows the 99% lower confidence bound. During the first concept, the problem is not yet a drift problem since nothing is changing. However, FLORA4 performs slightly better than CDC here. After the first sudden change, FLORA4 and CDC both recover similarly. However, FLORA4 reaches a slightly higher level before the final concept shift. After this second shift, CDC recovers 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. This may be 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, FLORA4, 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 sufficiently inaccurate to perform lower on all 3 concepts).

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

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

Figure 4 compares 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.

Figure 4 shows that CDC is on average slightly more accurate 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 conclusion is that both CDC and FLORA4 perform similarly on moderate drift, with CDC perhaps a bit more accurate.

It is informative to analyze the behavior of CDC. As the concept gradually shifted towards $\alpha = 0.5$, the committee was continuously purged until it hit bottom at around i_{160}. At that point the concept was stabilizing again at a new target and the committee became increasingly entrenched in support of the new

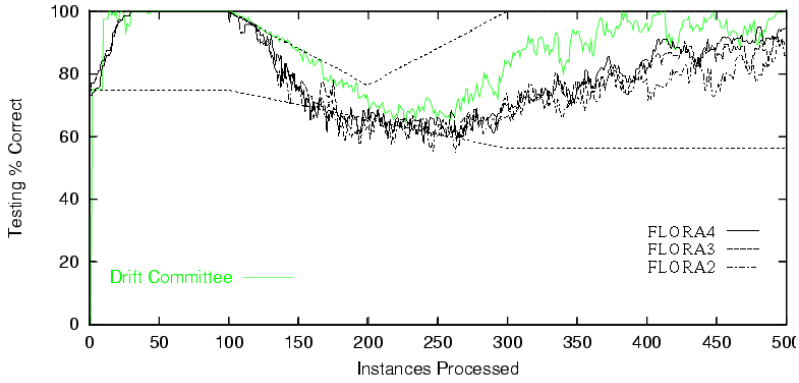

Figure 5: **Slow Drift Performance Comparison.** Performance of CDC (Drift Committee) and FLORA algorithms (Widmer and Kubat 1996) on on slow drift ($\Delta x = 200$). CDC performance is significantly higher than FLORA4 between instances 150 and 200, and between instances 280 and 400.

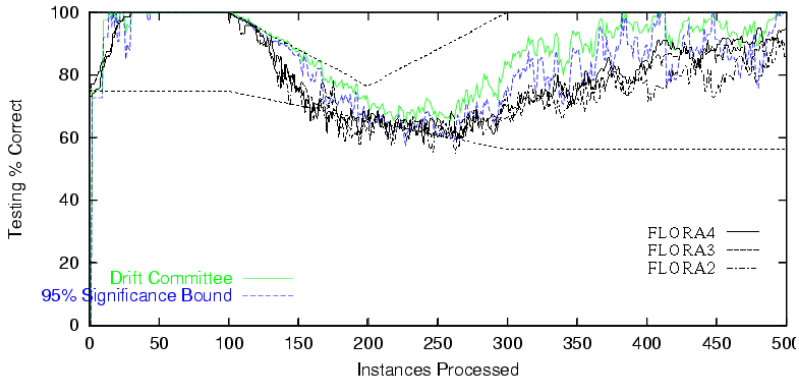

Figure 6: **Slow Drift Performance Significance.** The blue line (directly below the CDC line) shows 95% confidence lower bound on CDC performance on this task.

concept, rising in accuracy as the concept became more certain, all the way until it begins to oscillate between 90% and 100%. (FLORA4 oscillates between 80% and 90%)

Why does the committee not completely stabilize on the second concept? The reason is that the second concept is a disjunct of two conjuncts, and since the training examples are chosen randomly, 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 had an easier time stabilizing on it before i_{100}.

The next experiment, slow drift, is interesting because it features both algorithms attempting to grasp a moving target for an extended period of time.

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

Figure 5 shows 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 5. The difference is significant throughout both the fall and subsequent rise in accuracy from the changing α, as shown in figure 6.

4.3 Discussion

Why is the difference between CDC and FLORA most significant on slow drift as opposed to the other experiments? FLORA clearly has trouble recovering after such a long drift, barely climbing to the 90% level near the end. FLORA is 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 extreme. 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 advantage of CDC is that the more samples one has of an actual distribution, the less variance will occur in the sample distribution. A hypothesis is like a single sample window, with a relatively roughly approximated size. However, because CDC has 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 the accuracy of the hypotheses, 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 their explicit window. 20% adjustments turn out to be too severe for a slowly drifting distribution. CDC, on the other hand, can adapt to any drift rate. FLORA probably happens to be biased towards faster drift ($\Delta x = 100$) simply because of it's 20% window cutting rate when it discovers drift. In contrast, when drift is instantaneous, the ability of the algorithms to adjust to drift becomes less important than the underlying learning algorithm.

The question remains why FLORA methods fail to fully recover after slow drift ends. FLORA3 and FLORA4 keep a store of former concepts so that they can reuse expired concepts if they ever recur. We 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 comparable to the weighted averaging of many hypotheses in CDC. This conclusion supports the hypothesis that CDC is suited to any drift rate. CDC avoids answering both the question, "is the concept changing?" and, "what should we do about it if it is?"

5 Related Work

FLORA is not the only window-based algorithm. Klinkenberg and Thorsten (2000) introduced a window-based drift tracker based on support vector machines (SVMs). This algorithm is able to have a more precise

WAH than FLORA by using statistical properties of SVMs that can be theoretically shown to indicate appropriate window adjustments. However, the exclusive reliance on SVMs is a weakness for a concept drift algorithm, because it restricts the kinds of learning methods available for tracking drift. An ideal drift-tracking strategy should be independent from the concept-learner; if a superior concept learning algorithm appears, it should be possible to employ it in an existing concept drift framework. There is nothing in CDC that necessitates the use of decision trees. Thus, CDC can be used with future concept learning algorithms, or domain-specific learning algorithms biased towards particular kinds of concepts that SVMs might not be suited for.

As pointed out in Section 1, several other algorithms use committees to track drift. Most of these methods are variants of Winnow (Littlestone 1988), in which each committee member only looks at a small subset of the features, and casts a weighted vote if the the current instance contains the features with which it is concerned (Auer and Warmuth 1998; Blum 1997; Herbster and Warmuth 1998; Mesterharm 2002). The vote is based on a constant number of past instances that include the relevant features. Thus, the committee members are specialists that make classification decisions based on the fixed past history of a small set of features. In contrast, all CDC committee members see *all* the features, and each member bases its decision on a differently sized history of past instances. In addition, CDC committee members are themselves generic learners. In this paper, CDC used decision trees, but committee members can employ any concept learner, including neural networks, SVMs, or even complete Winnow classifiers. Moreover, CDC committee members can retire and be replaced by neophytes. Thus, CDC substantially differs from other committee-based tracking methods. Because mistake bounds have been established for Winnow-based trackers (Auer and Warmuth 1998; Herbster and Warmuth 1998; Mesterharm 2002), CDC should be compared to these algorithms in the future to gain a better understanding of its performance.

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 weigh decay is orthogonal to CDC, and could be incorporated into the CDC algorithm as a supplement.

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 efficacy of unsupervised concept drift tracking is not addressed by CDC.

In summary, a variety of methods exist. Among them, CDC is able to precisely align to shifting distributions without being tied to a particular learning algorithm. However, more comparisons are necessary.

6 Future Work

In addition to the need to compare with Winnow-based methods, several issues remain to be addressed. First, CDC should be tested with alternate base learning algorithms in order to distinguish the contribution of decision trees from CDC in general.

Second, CDC (and other drift tracking algorithms) should be tested 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.

Third, more experiments need to be performed to understand the contribution of committee size, age of maturity, and voting weights to performance. Does the algorithm become increasingly accurate the larger the committee? What is the point of diminishing returns? Because using the accuracy of a committee member as the voting weight has not been theoretically justified, alternate vote weighting schemes, such as log-likelihood, should be tested.

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, like in boosting. Or an example-weighting scheme based on similarity could be added, as implemented by Salganicoff (1993).

Finally, we have not checked to see how robust CDC is with respect to noise. We expect CDC would perform 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 also designed to be resistant to noise.

7 Conclusion

CDC is a promising algorithm for tracking concept drift. It outperformed a leading algorithm on a protracted drift problem. 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 experimental results imply that an explicit window may not be the best way to track concept drift despite its intuitive appeal. In a larger context, this research confirms once again the utility of committees in learning.

References

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

Auer, P., and Warmuth, M. K. (1998). Tracking the best disjunction. *Machine Learning*, 32(2):127–150.

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.

Blum, A. (1997). Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. *Machine Learning*, 26(1):5–23.

Freund, Y., and Mansour, Y. (1997). Learning under persistent drift. In *Computational Learning Theory: Third European conference (EuroCOLT '97)*, 109–118. Springer Verlag.

Helmbold, D., and Long, P. (1994). Tracking drifting concepts by minimizing disagreements. *Machine Learning*, 14(1):27–46.

Herbster, M., and Warmuth, M. K. (1998). Tracking the best regressor. In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT 1998)*, 14–31. New York, NY: ACM Press.

Klinkenberg, R., and Thorsten, J. (2000). Detecting concept drift with support vector machines. In *Machine Learning: Proceedings of the 17th Annual Conference*. San Francisco, CA: 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.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(1):285–318.

Maloof, M. A., and Michalski, R. S. (2000). Selecting examples for partial memory learning. *Machine Learning*, 41(1):27–52.

Mesterharm, C. (2002). Tracking linear-threshold concepts with winnow. In *Proceedings of the 15th Annual Conference on Computational Learning Theory (COLT 2002), Volume 2375 of Lecture Notes in Artificial Intelligence*, 138–152. Springer Verlag.

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.