Using Additive Expert Ensembles to Cope with Concept Drift

Jeremy Z. Kolter and Marcus A. Maloof {JZK, MALOOF}@CS.GEORGETOWN.EDU
Department of Computer Science, Georgetown University, Washington, DC 20057-1232, USA

Abstract

We consider online learning where the target concept can change over time. Previous work on expert prediction algorithms has bounded the worst-case performance on any subsequence of the training data relative to the performance of the best expert. However, because these "experts" may be difficult to implement, we take a more general approach and bound performance relative to the actual performance of any online learner on this single subsequence. We present the additive expert ensemble algorithm **AddExp**, a new, general method for using any online learner for drifting concepts. We adapt techniques for analyzing expert prediction algorithms to prove mistake and loss bounds for a discrete and a continuous version of **AddExp**. Finally, we present pruning methods and empirical results for data sets with concept drift.

1. Introduction

We consider the online learning scenario where an algorithm is presented with a series of labeled examples, $\mathcal{S} = \{\mathbf{x}_t, y_t\}$, for $t = 1, 2, \ldots, T$. At each time step, the algorithm outputs a class prediction, $\hat{y}_t$, for the given feature vector $\mathbf{x}_t$, then updates its hypothesis based on the true class y_t. In the discrete case, y_t and $\hat{y}_t$ are in a finite set of classes Y, and under the mistake bound model (Littlestone, 1988), one bounds the number of mistakes an algorithm makes on some target concept. In the continuous case, $y_t, \hat{y}_t \in [0, 1]$, and one bounds the maximum loss suffered by the algorithm – we focus on the absolute loss function $L(\hat{y}_t, y_t) = |\hat{y}_t - y_t|$. In addition, we allow for the target concept to change over time.

Certain expert prediction algorithms (Littlestone & Warmuth, 1994; Herbster & Warmuth, 1998) provide theoretical performance guarantees under concept drift. These algorithms predict based on the advice of several prediction strategies, called *experts*. When trained on a series of different concepts, these algorithms perform "almost as well" as the best expert on each concept – the assumption being that different experts perform best on different concepts. In practice, however, such algorithms can be difficult to implement. If the experts are fixed prediction strategies, then they are limited to those that one can delineate prior to any training. Alternatively, if the experts are learning algorithms, then they must be able to adapt to concept drift individually, since all are trained over the entire example series.

In this paper, we take a different, more adaptive approach: we bound our algorithm's performance over changing concepts, not relative to the performance of any abstract expert, but relative to the actual performance of an online learner trained on each concept individually. Our algorithm, called Additive Expert (**AddExp**), maintains what we call an *additive expert ensemble*. It is similar to expert prediction algorithms, except that new experts can be added during the online learning process. We adapt techniques from Littlestone and Warmuth (1994) and Freund and Schapire (1997) to prove performance bounds under the mistake bound and online loss models. The primary result of the paper is, when trained on a series of changing concepts, **AddExp** makes $O(m)$ mistakes on any concept, where m is the number of mistakes that the base learner makes when trained only on this single concept. Similarly for the continuous case, **AddExp** will suffer a loss of $O(\ell)$ on any concept, where ℓ is the loss of the base learner when trained only on this single concept. We present pruning methods so the algorithm can be efficient on complex data sets. Finally, we present empirical results.

2. Background and Related Work

Much work on expert prediction can be traced back to the Weighted Majority (WM) algorithm (Littlestone & Warmuth, 1994) and Vovk's (1990) work on aggre-

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

gating strategies. Cesa-Bianchi et al. (1997) propose a general algorithm for randomized predictions and prove lower bounds on the net loss of any expert prediction algorithm. Freund and Schapire (1997) propose the **Hedge** algorithm, which uses these methods for an online-allocation problem. Haussler et al. (1998) and Vovk (1998) generalize previous results to general-case loss functions. Separately, several algorithms have been proposed to deal with concept drift (e.g., Kolter & Maloof, 2003; Schlimmer & Granger, 1986; Street & Kim, 2001; Widmer & Kubat, 1996). There has also been theoretical analysis (e.g., Helmbold & Long, 1994).

Expert prediction algorithms have been applied specifically to the problem of concept drift. A variant of the original WM algorithm, WML (Littlestone & Warmuth, 1994) adds mechanisms that allow new experts to become dominant in constant time. Blum (1997) implemented a version of WM for a calendar scheduling task that exhibits concept drift. Herbster and Warmuth (1998) generalize the loss bounds of such expert predictions algorithms to general-case loss functions, and Monteleoni and Jaakkola (2004) have recently extended this work by allowing the algorithm to adjust parameters during online learning to improve performance. Bousquet and Warmuth (2003) analyze a special case where the best experts come from a small subset of the ensemble.

Additive expert ensembles were introduced by Kolter and Maloof (2003) with the Dynamic Weighted Majority (DWM) algorithm, which shares many similarities with the **AddExp** algorithm we present in this paper. DWM was shown to perform well in situations with concept drift, but the evidence for this performance was entirely empirical, and in fact DWM's weighting scheme makes it impossible to bound performance in the worst case. DWM handles only discrete classes, can become unstable in noisy settings, and has a pruning mechanism that provides no guarantee as to the number of experts created. The **AddExp** algorithm, as we will describe, allows for theoretical analysis, generalizes to regression tasks, provides pruning mechanisms that cap the number of experts, and achieves better empirical results than DWM.

3. AddExp for Discrete Classes

In this section, we present the **AddExp.D** algorithm (see Figure 1) for handling discrete class predictions. As its concept description, **AddExp.D** maintains an ensemble of predictive models, referred to as *experts*, each with an associated weight. Experts use the same algorithm for training and prediction, but are created at different time steps. The performance element of the algorithm uses a weighted vote of all the experts. For each possible classification, the algorithm sums the weights of all the experts predicting that classification, and predicts the classification with the greatest weight. The learning element of the algorithm first predicts the classification of the training example. The weights of all the experts that misclassified the example are decreased by a multiplicative constant β. If the overall prediction was incorrect, a new expert is added to the ensemble with weight equal to the total weight of the ensemble times some constant γ. Finally, all experts are trained on the example.

Algorithm AddExp.D$(\{\mathbf{x}, y\}^T, \beta, \gamma)$
Parameters:
$\{\mathbf{x}, y\}^T$: training data with class $y \in Y$
$\beta \in [0, 1]$: factor for decreasing weights
$\gamma \in [0, 1]$: factor for new expert weight
Initialization:
1. Set the initial number of experts $N_1 = 1$.
2. Set the initial expert weight $w_{1,1} = 1$.

For $t = 1, 2, \ldots, T$:
1. Get expert predictions $\xi_{t,1}, ..., \xi_{t,N_t} \in Y$
2. Output prediction:
$$\hat{y}_t = \text{argmax}_{c \in Y} \sum_{i=1}^{N_t} w_{t,i}[c = \xi_{t,i}]$$
3. Update expert weights:
$$w_{t+1,i} = w_{t,i}\beta^{[y_t \neq \xi_{t,i}]}$$
4. If $\hat{y}_t \neq y_t$ then add a new expert:
$$N_{t+1} = N_t + 1$$
$$w_{t+1,N_{t+1}} = \gamma \sum_{i=1}^{N_t} w_{t,i}$$
5. Train each expert on example $\mathbf{x}_t, y_t$.

Figure 1. **AddExp** for discrete class predictions.

If trained on a large amount of data, **AddExp.D** has the potential to create a large number of experts. However, since this does not affect the theoretical mistake or loss bounds of the algorithm, we will not deal with this issue until Section 5, where we discuss pruning.

3.1. Analysis

Here we analyze the performance of **AddExp.D** within the mistake bound model (Littlestone, 1988) by adapting the methods used by Littlestone and Warmuth (1994) to analyze the WM algorithm. As this is a worst-case analysis, we allow for an adversary to choose, at any time step, whether any given expert is going to make a mistake. This may seem odd because, after all, not only do these experts all employ

the same learning algorithm, they are also all trained from the same data stream – the only difference being that some experts are trained on a larger subset of the stream than other experts. However, if we consider training noise and the sampling effects present in real-world problems, it becomes difficult to conclude anything about the performance of the different experts. Therefore we perform this analysis in the most general case.

We denote the total weight of the ensemble at time step t to be $W_t = \sum_{i=0}^{N_t} w_{t,i}$. We also let M_t be the number of mistakes that **AddExp.D** has made through time steps $1, 2, \ldots, t-1$. The bound rests on the fact that the total weight of the ensemble decreases exponentially with the number of mistakes.

Theorem 3.1 *For any time steps* $\mathrm{t}_1 < \mathrm{t}_2$*, if we stipulate that* $\beta + 2\gamma < 1$*, then the number of mistakes that* ***AddExp.D*** *will make between time steps* t_1 *and* t_2 *can be bounded by*

$$M_{t_2} - M_{t_1} \leq \frac{\log(W_{t_1}/W_{t_2})}{\log(2/(1+\beta+2\gamma))}.$$

Proof **AddExp.D** operates by multiplying the weights of the experts that predicted incorrectly by β. If we assume that a mistake is made at time step t then we know that at least $1/2$ of the weight in the ensemble predicted incorrectly. In addition, a new expert will be added with weight γW_t. So we can bound the weight with

$$W_{t+1} \leq \frac{1}{2}W_t + \frac{\beta}{2}W_t + \gamma W_t = \frac{1+\beta+2\gamma}{2}W_t \; .$$

Applying this function recursively leads to:

$$W_{t_2} \leq \frac{1+\beta+2\gamma}{2}^{M_{t_2}-M_{t_1}} W_{t_1} \; .$$

Taking the logarithm and rearranging terms gives the desired bound, which will hold since the requirement that $\beta + 2\gamma < 1$ ensures that $(1+\beta+2\gamma)/2 < 1$. □

Now suppose that the algorithm makes a mistake at time t_1, and so a new expert is added. The initial weight of this expert will be γW_{t_1}. Let m be the number of mistakes made by the new expert by time step t_2. The weight of the expert is multiplied by β for each mistake it makes, so its weight at time t_2 will be $\gamma W_{t_1}\beta^m$. Since the total weight of the ensemble must be greater than that of any individual member,

$$W_{t_2} \geq \gamma W_{t_1}\beta^m \; . \tag{1}$$

Corollary 3.2 *For any time steps,* $\mathrm{t}_1 < \mathrm{t}_2$*, if* ***AddExp.D*** *makes a mistake at time* t_1 *– and therefore adds a new expert to the pool – then the number of mistakes that* ***AddExp.D*** *makes between time steps* t_1 *and* t_2 *can be bounded by*

$$M_{t_2} - M_{t_1} \leq \frac{m\log(1/\beta) + \log(1/\gamma)}{\log(2/(1+\beta+2\gamma))} \; ,$$

where m *is the number of mistakes that the new expert makes in this time frame.*

Proof The bound is obtained by substituting (1) into the result of Theorem 3.1. □

Finally, we show how this analysis applies to concept drift.

Corollary 3.3 *Let* $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ *be any arbitrary partitioning of the input sequence* $\mathcal{S}$ *into* k *subsequences, such that* ***AddExp.D*** *makes a mistake on the first example of each subsequence. Then for each* $\mathcal{S}_i$*,* ***AddExp.D*** *makes no more than*

$$\min\left\{|\mathcal{S}_i|, \frac{m_i\log(1/\beta) + \log(1/\gamma)}{\log(2/(1+\beta+2\gamma))}\right\}$$

mistakes, where m_i *is the number of mistakes made by the base learner when trained only on* $\mathcal{S}_i$*.*

Proof The follows follows by applying Corollary 3.2 to each subsequence $\mathcal{S}_i$. □

It is important to note that the requirement that **AddExp.D** makes a mistake on the first example of each subsequence is not a practical limitation, since subsequences need not correspond directly to concepts. Consider the subsequence that begins whenever **AddExp.D** makes its first mistake on some concept, and ends at the last example of this concept. We can bound the performance of **AddExp.D** on this subsequence relative to the performance of the base learner. And since we know that **AddExp.D** made no mistakes on the concept before the subsequence begins, this also serves as a bound for the performance of **AddExp.D** over the entire concept. Additionally, while this bound is most applicable to scenarios of sudden drift, it is by no means limited to these situations. It only requires that at some point during online training, a new expert will perform better than experts trained on "old" data, which can also happen in situations of gradual drift.

However, there is another difficulty when considering the theoretical performance of **AddExp.D**. It can be shown that the coefficient on the m term in Corollary 3.2 can never be reduced lower than 2, which can be a significant problem if, for example, the base learner makes a mistake 20% of the time.

Theorem 3.4 *For any $\beta, \gamma \in [0,1]$,*

$$\frac{m\log(1/\beta)+\log(1/\gamma)}{\log(2/(1+\beta+2\gamma))} \geq 2m\ .$$

Proof (Sketch) The coefficient of m is minimized as $\gamma \to 0$, so it will always be greater than

$$\frac{\log(1/\beta)}{\log(2/(1+\beta))}\ .$$

It can be shown that $\ln(1/x) \geq 2(1-x)/(1+x)$ for $x \in [0,1]$. Using this inequality to bound the numerator, and the inequality $\ln(1+x) \leq x$ to bound the denominator gives the desired result. This can also be verified by plotting the function for $\beta \in [0,1]$. □

A similar situation occurs in the analysis of the original WM algorithm. The problem is that since only half the weight of the ensemble needs to predict incorrectly in order to make a mistake, an adversary will be able to let the experts conspire to drive up the number of global mistakes. For this reason, we shift our focus to a continuous version of the algorithm analyzed under the online loss model. In the case of binary classes, it is possible to use randomized prediction to obtain a bound on the expected number of mistakes equivalent to the loss bound.

4. AddExp for Continuous Classes

In this section, we present and analyze the **AddExp.C** algorithm (see Figure 2) for predicting values in the interval [0,1]. The general operation of the **AddExp.C** algorithm is the same as that of **AddExp.D** except that class predictions – both the overall prediction of **AddExp.C** and the predictions of each expert – are in $[0,1]$ rather than in a discrete set of classes Y. This necessitates other changes to the algorithm. Rather than predicting with a weighted majority vote, **AddExp.C** predicts the sum of all expert predictions, weighted by their relative expert weight. When updating weights, the algorithm uses the rule

$$w_{t+1,i} = w_{t,i}\beta^{|\xi_{t,i}-y_t|}\ . \tag{2}$$

As with other expert prediction algorithms, the following analysis also holds for a more general update function $w_{t+1,i} = w_{t,i}U_\beta(|\xi_{t,i}-y_t|)$, for any $U_\beta : [0,1] \to [0,1]$, where $\beta^r \leq U_\beta(r) \leq 1-(1-\beta)r$ for $\beta \geq 0$ and $r \in [0,1]$. Whereas we previously added a new member to the ensemble whenever the global prediction was incorrect, we now add a new member whenever the global loss of the algorithm on that time step is greater than some threshold $\tau \in [0,1]$. The weight of this new expert is calculated by:

$$w_{t+1,N_{t+1}} = \gamma\sum_{i=1}^{N_t} w_{t,i}|\xi_{t,i}-y_t|\ . \tag{3}$$

Algorithm AddExp.C$(\{\mathbf{x},y\}^T,\beta,\gamma,\tau)$
Parameters:
$\{\mathbf{x},y\}^T$: training data with class $y \in [0,1]$
$\beta \in [0,1]$: factor for decreasing weights
$\gamma \in [0,1]$: factor for new expert weight
$\tau \in [0,1]$: loss required to add a new expert
Initialization:
1. Set the initial number of experts $N_1 = 1$.
2. Set the initial expert weight $w_{1,1} = 1$.

For $t = 1,2,\ldots,T$:
1. Get expert predictions $\xi_{t,1},\ldots,\xi_{t,N_t} \in [0,1]$
2. Output prediction:
$$\hat{y}_t = \frac{\sum_{i=1}^{N_t} w_{t,i}\xi_{t,i}}{\sum_{i=1}^{N_t} w_{t,i}}$$
3. Suffer loss $|\hat{y}_t - y_t|$
4. Update expert weights:
$$w_{t+1,i} = w_{t,i}\beta^{|\xi_{t,i}-y_t|}$$
5. If $|\hat{y}_t - y_t| \geq \tau$ add a new expert:
$$N_{t+1} = N_t + 1$$
$$w_{t+1,N_{t+1}} = \gamma\sum_{i=1}^{N_t} w_{t,i}|\xi_{t,i}-y_t|$$
6. Train each expert on example $\mathbf{x}_t, y_t$.

Figure 2. **AddExp** for continuous class predictions.

4.1. Analysis

We adapt techniques from Littlestone and Warmuth (1994) and Freund and Schapire (1997) to analyze **AddExp.C**. Again we use the notation W_t to denote the total weight of the expert pool at time t. We let L_t refer to the total absolute loss suffered by the algorithm before time t, with $L_t = \sum_{i=1}^{t-1}|\hat{y}_i - y_i|$.

Theorem 4.1 *For any time steps $\mathrm{t}_1 < \mathrm{t}_2$, if we stipulate that $\beta+\gamma < 1$, then the absolute loss of **AddExp.C** between time steps t_1 and t_2 can be bounded by*

$$L_{t_2} - L_{t_1} \leq \frac{\ln(W_{t_1}/W_{t_2})}{1-\beta-\gamma}.$$

Proof It can be shown by a convexity argument (e.g., Littlestone & Warmuth, 1994, Lemma 5.1) that

$$\beta^r \leq 1-(1-\beta)r \tag{4}$$

for $\beta \geq 0$ and $r \in [0,1]$. At each time step **AddExp.C** updates the weights according to (2) and may add a

new expert with weight determined by (3). Combining these equations with (4) gives

$$\begin{aligned} W_{t+1} &\leq \sum_{i=0}^{N_t} w_{t,i}\beta^{|\xi_{t,i}-y_t|} + \gamma\sum_{i=1}^{N_t} w_{t,i}|\xi_{t,i}-y_t| \\ &\leq \sum_{i=0}^{N_t} w_{t,i}(1-(1-\beta-\gamma)|\xi_{t,i}-y_t|) \\ &\leq W_t(1-(1-\beta-\gamma)|\hat{y}-y|)\ . \end{aligned}$$

Applying repeatedly for $t = t_1, \ldots, t_2 - 1$ gives

$$W_{t_2} \leq W_{t_1}\prod_{t=t_1}^{t_2-1}(1-(1-\beta-\gamma)|\hat{y}_t-y_t|)\ .$$

The theorem follows by taking the natural logarithm and using the inequality $\ln(1+x) \leq x$. □

Now suppose that the loss at time step t_1 is greater than τ, so a new expert is added with initial weight $\gamma\sum_{i=1}^{N_t} w_{t,i}|\xi_{t,i}-y_t| = \gamma|\hat{y}_t-y_t|W_t$. Let ℓ be the loss that the new expert has suffered by time t_2. At time step t_2 the weight of this expert will be $\beta^\ell\gamma|\hat{y}_t-y_t|W_t$. And since the total weight of the ensemble must be greater than any individual member, we have

$$W_{t_2} \geq \beta^\ell\gamma|\hat{y}_{t_1}-y_{t_t}|W_{t_1}\ . \tag{5}$$

We use this to prove corollaries corresponding to 3.2 and 3.3 for the continuous case.

Corollary 4.2 *For any time steps,* $\mathrm{t}_1 < \mathrm{t}_2$*, if* ***AddExp.C*** *suffers a loss greater than* τ *at time* t_1 *– and therefore adds a new expert to the pool – then the loss suffered by* ***AddExp.C*** *between time steps* t_1 *and* t_2 *can be bounded by*

$$L_{t_2} - L_{t_1} \leq \frac{\ell\ln(1/\beta)+\ln(1/\gamma)+\ln(1/\tau)}{1-\beta-\gamma}\ ,$$

where ℓ *is the loss suffered by the new expert in this time frame.*

Proof The bound is obtained immediately by substituting (5) into the result of Theorem 4.1 and noticing that $|\hat{y}_{t_1}-y_{t_1}| \geq \tau$. □

Corollary 4.3 *Let* $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ *be any arbitrary partitioning of the input sequence* $\mathcal{S}$ *such that at the first example in each subsequence,* ***AddExp.C*** *suffers loss greater than* τ*. Then for each subsequence* $\mathcal{S}_i$*,* ***AddExp.C*** *suffers loss no greater than*

$$\min\left\{|\mathcal{S}_i|, \frac{\ell_i\ln(1/\beta)+\ln(1/\gamma)+\ln(1/\tau)}{1-\beta-\gamma}\right\}\ ,$$

where ℓ_i *is the loss suffered by the base learner when trained only on* $\mathcal{S}_i$*.*

Proof The bound is obtained by applying Corollary 4.2 to each partition $\mathcal{S}_i$. □

As with the discrete case, the requirement that **AddExp.C** suffer loss greater than τ at the first example of each subsequence is not a significant practical limitation. It is slightly more difficult in the continuous case, since **AddExp.C** may continuously suffer loss less than τ, and never add a new expert. However, since τ can be made arbitrary small, with only a $\log(1/\tau)$ penalty, this "extra" loss suffered can be reduced arbitrarily close to zero.

4.2. Optimizing Parameters

In this section, we present methods for choosing parameters β and γ based on previous knowledge of the problem. Specifically, if we have an upper bound $\tilde{\ell}$ on the loss of a given expert, then we can choose β and γ so as to minimize the loss bound for **AddExp.C**. However, due to the nature of the bound, it is very difficult to optimize the parameters precisely, and so approximations are used. Well-chosen values for β and γ result in analysis that is remarkably similar to the analysis of WM-style algorithms.

We make use of the following lemma from Freund and Schapire (1997), slightly rewritten:

Lemma 4.4 (Freund & Schapire, 1997) *Suppose* $0 \leq \mathrm{A} \leq \tilde{\mathrm{A}}$ *and* $0 \leq \mathrm{B} \leq \tilde{\mathrm{B}}$*. Let* $\alpha = 1+\sqrt{2\tilde{\mathrm{B}}/\tilde{\mathrm{A}}}$*. Then*

$$\frac{A\ln\alpha+B}{1-(1/\alpha)} \leq A+B+\sqrt{2\tilde{A}\tilde{B}}\ .$$

Empirical optimization of the loss bound suggests that the bound reaches a minimum when $\gamma = \beta/\ell$. Using this conjecture we can choose values for β and γ such that the resulting equation is an instance of Lemma 4.4. Specifically, we choose the values

$$\beta = \frac{\tilde{\ell}}{(\tilde{\ell}+1)g(\tilde{\ell})} \quad \text{and} \quad \gamma = \frac{1}{(\tilde{\ell}+1)g(\tilde{\ell})}\ , \tag{6}$$

where

$$g(\tilde{\ell}) = 1+\sqrt{2(\ln^*(\tilde{\ell}+1)+\ln(1/\tau))/(\tilde{\ell}+1)}\ .^1 \tag{7}$$

Substituting these values into the bound from Corollary 4.2 satisfies the conditions of Lemma 4.4 with

[1] Here we define the function

$$\ln^*(x) = x\ln x-(x-1)\ln(x-1).$$

We use the name $\ln^*$ since this function is asymptotically similar to the ln function, and can be bounded by $\ln^* x \leq \ln x+1$ for $x > 1$. The difference is that $\ln^* x$ approaches 0 as x approaches 1, so the bound is stronger.

$\tilde{A} = \tilde{\ell} + 1$, $\tilde{B} = \ln^*(\tilde{\ell} + 1) + \ln(1/\tau)$ and $\alpha = g(\tilde{\ell})$. So we can obtain the new loss bound,

$$L_{t_2} - L_{t_1} \leq \ell + 1 + \ln^*(\ell + 1) + \ln(1/\tau) + \sqrt{2(\tilde{\ell} + 1)(\ln^*(\tilde{\ell} + 1) + \ln(1/\tau))} .$$

If we choose $\tilde{\ell}$ and τ such that $\tilde{\ell} \geq 1/\tau$, then the net loss of **AddExp.C** over this interval, $L_{t_2} - L_{t_1} - \ell$, is $O\left(\sqrt{\tilde{\ell} \ln \tilde{\ell}}\right)$. This bears a strong resemblance to the bound proven by Cesa-Bianchi et al. (1997) and others for the worst-case net loss of WM-style algorithms: $O\left(\sqrt{\tilde{l} \ln n}\right)$, where n in the number of experts.

5. Pruning

Until now, we have ignored issues of efficiency. **AddExp** can potentially add a new expert every time step, leading to inefficiency as the length of the input grows. In this section, we present two pruning techniques, one that has theoretical guarantees but which may be impractical, and one that performs well in practice but which has no theoretical guarantees.

Ideally, pruning removes unnecessary experts from the ensemble while keeping the best experts. If done correctly, it will not affect the mistake or loss bounds proven in Sections 3 and 4. These bounds depended only on the fact that the weight of the ensemble decreases over time, and pruning does not interfere with this. If we are concerned with the bound between time steps t_1 and t_2, the only requirement is that we do not remove the best expert during this time period.

Pruning Method 1 (Oldest First) *When a new expert is added to the ensemble, if the number of experts is greater than some constant* K*, remove the oldest expert before adding the new member.*

Theorem 5.1 *Let $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ be a partitioning of the input series as discussed in Corollaries 3.3 and 4.3. Suppose that we have an upper bound $\tilde{m}$ or $\tilde{\ell}$ on the maximum mistakes or loss of the base learner on any partition. Then **AddExp** trained on $\mathcal{S}$, using the Oldest First pruning method with suitably chosen K, will have the same mistake or loss bound as **AddExp** without pruning.*

Proof For the discrete case, let

$$K = \frac{\tilde{m} \log(1/\beta) + \log(1/\gamma)}{\log(2/(1 + \beta + 2\gamma))} .$$

By Corollary 3.3, **AddExp.D** will make no more than K mistakes when trained on any partition $\mathcal{S}_i$. Since a new expert is added only when a mistake is made, this also serves as a bound for the number of experts that can be created when trained on one partition. Therefore, if more experts than K are created, the oldest expert must span more than one partition, and we can remove it, since the bound depends only on the expert added at the beginning of each partition.

In the continuous case we select

$$K = \frac{\tilde{\ell} \ln(1/\beta) + \ln(1/\gamma) + \ln(1/\tau)}{1 - \beta - \gamma} \cdot \frac{1}{\tau} ,$$

as this is a worst-case bound on the number of new experts that can be created. After this, the proof parallels that of the discrete case. □

Unfortunately, this method may be difficult to use in practice since we may not be able to obtain $\tilde{\ell}$ or $\tilde{m}$. Even if we could, the number of experts required might still be prohibitively large. Additionally, if we select a K that is too small, Oldest First may seriously degrade the performance of **AddExp** since, when trained on a static concept, it is precisely the oldest expert that is usually the most valuable. For this reason, we introduce a more practical pruning method.

Pruning Method 2 (Weakest First) *When a new expert is added to the ensemble, if the number of experts is greater than a constant* K*, remove the expert with the lowest weight before adding the new member.*

Weakest First performs well in practice, because it removes the worst performing experts. However, it is difficult to determine performance in the worst case since an adversary can always degrade the initial performance of the optimal expert, such that its weight falls below other experts. We must also be careful in choosing K in conjunction with γ, as we do not want new experts to be pruned before they are given a chance to learn. However, as we show in the next section, Weakest First works well in practice, even with small values of K.

6. Empirical Results

To provide empirical support for **AddExp**, we evaluated it on the STAGGER Concepts (Schlimmer & Granger, 1986), a standard benchmark for concept drift, and on classification and regression tasks involving a drifting hyperplane. For the STAGGER Concepts, each example consists of three attributes, *color* $\in$ {*green, blue, red*}, *shape* $\in$ {*triangle, circle, rectangle*}, and *size* $\in$ {*small, medium, large*}, and a class label $y \in \{+, -\}$. Training lasts for 120 time steps, and consists of three target concepts that last 40 time steps each: (1) *color* = *red* $\wedge$ *size* = *small*, (2) *color* = *green* $\vee$ *shape* = *circle*, and (3) *size* = *medium* $\vee$

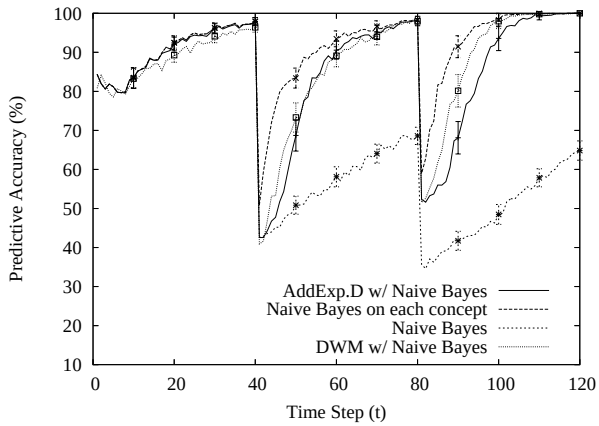

Figure 3. Predictive accuracy on the STAGGER concepts.

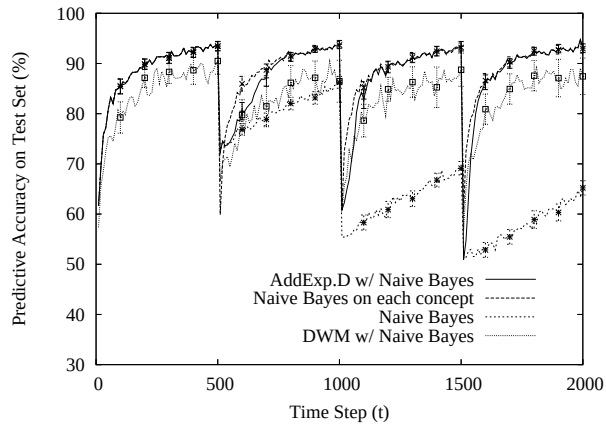

Figure 4. Predictive accuracy on the hyperplane problem.

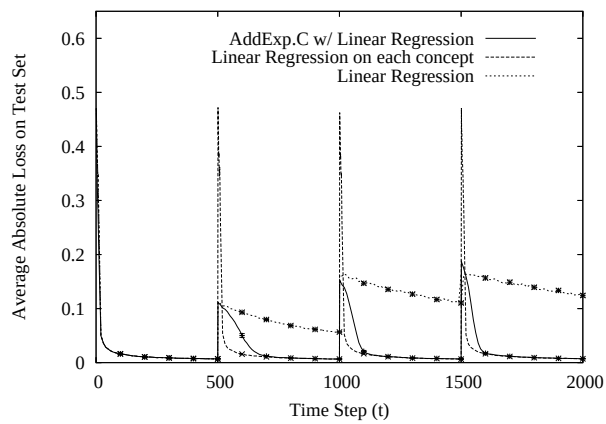

Figure 5. Average absolute loss on a continuous version of the hyperplane problem.

size = large. At each time step, the learner is trained on one example, and tested on 100 examples generated randomly according to the current concept.

To test performance on a larger problem involving noise and more gradual drift, and to evaluate **AddExp** on a regression task, we created a synthetic problem using a drifting hyperplane. Feature vectors consist of ten variables, $\mathbf{x} \in [0,1]^{10}$, each with a uniform distribution over this range. Training lasts for 2000 time steps, with 4 target concepts lasting for 500 examples each. At every time step, the learners are trained on one example and tested on 100. For the classification task, $y \in \{+,-\}$, and the general target concept is $(x_i + x_{i+1} + x_{i+2})/3 > 0.5$, with $i = 1$, 2, 4, and 7 for each of the 4 concepts, respectively. We also introduced 10% random class noise into the training data. The regression task is identical except that $y \in [0,1]$ and we eliminated the threshold, i.e., $y = (x_i + x_{i+1} + x_{i+2})/3$ with $i = 1$, 2, 4, and 7 for each of the 4 concepts. Instead of class noise, we introduced a ± 0.1 random variate to y, clipping y if it was not in $[0,1]$.

We evaluated **AddExp.D** on the STAGGER and hyperplane classification tasks, and **AddExp.C** on the hyperplane regression task. Based on pilot studies, we chose parameters of $\beta = 0.5, \gamma = 0.1$ for STAGGER, $\beta = 0.5, \gamma = 0.01$ for the hyperplane classification task, and $\beta = 0.5, \gamma = 0.1, \tau = 0.05$ for hyperplane regression. We used an incremental version of naive Bayes as the base learner for **AddExp.D** and an online batch version of least squares regression as the base learner for **AddExp.C**. For the sake of comparison, we evaluated the performance of the base learner over the entire data set, and on each concept individually. These serve as worst-case and best-case comparisons, respectively, since the base learners have no mechanisms for adjusting to concept drift, and conversely, the base learners trained on each concept individually are not hindered by previous data. We also evaluated DWM on the classification tasks – using naive Bayes as a base learner, $\beta = 0.5$, and allowing new experts to be added every time step, mirroring **AddExp.D**. For all simulations, we averaged accuracies over 50 runs, and calculated 95% confidence intervals.

Figure 3 shows the performance of **AddExp.D** on the STAGGER data set. All algorithms except DWM perform virtually the same on the first concept, since no drift takes place. Once the target concept changes, naive Bayes cannot adapt quickly to the new concept, but **AddExp.D** is able to quickly converge to performance similar to naive Bayes trained only on the new target concept. Figure 4 shows a similar result for the hyperplane classification task. **AddExp.D** converges quickly to all concepts. However, it converges slowest to the second concept, the most gradual of the three drifts since only one relevant variable changes, and x_2 and x_3 are relevant for both concepts. After the drift occurs, **AddExp.D**'s performance is initially better than naive Bayes', because of the "old" experts and the overlap of the two concepts. Convergence is slower because it takes longer for the new experts to "out-

weigh" the old ones. This shows **AddExp.D** balancing the predictions of its old and new experts, which is important for real-world applications, since it is not known in advance whether drift will be gradual or sudden. Finally, Figure 5 shows similar behavior for **AddExp.C** on the hyperplane regression task.

DWM performs slightly worse than **AddExp.D** on the first STAGGER concept (Figure 3), but outperforms it on the last. This is because DWM gives more weight to a new expert – set initially to the highest weight in the ensemble – which can accelerate convergence, but it can also let new experts become dominant too quickly, thereby decreasing performance. This is particularly harmful in noisy scenarios, as shown by Figure 4. To deal with this problem, DWM uses a parameter such that new experts can only be added periodically. However, this prevents DWM from recognizing drift during this period, and does not alleviate the problem entirely. Therefore, we argue that **AddExp** is a more robust solution to this problem, since the γ parameter allows one to precisely dictate the weight of new experts, and the bounds we have proven show that new experts can never dominate entirely if an older expert performs better.

By the end of the 2000 training examples on the hyperplane classification task, **AddExp.D** created an average of 385 experts, causing the algorithm to be much slower than naive Bayes. This is especially a problem in noisy domains, since new experts will continually be created when the algorithm misclassifies noisy examples. Fortunately, the Weakest First method of pruning works remarkably well on this task. The learning curves for **AddExp.D** with a maximum of 10 or 20 experts are virtually indistinguishable from the unpruned learning curve. The average accuracy over all time steps for unpruned **AddExp.D** is 87.34% $\pm$ 0.06, while Weakest First achieves average accuracies of 87.27 ± 0.06 for $K = 20$ and 87.08 ± 0.06 for $K = 10$.

7. Concluding Remarks

We presented the additive expert ensemble algorithm **AddExp**, a general method for using any online learner to handle concept drift. We proved mistake and loss bounds on a subsequence of the training data relative to the actual performance of the base learner, the first theoretical analysis of this type. We proposed two pruning methods that limit the number of experts in the ensemble, and evaluated the performance of the algorithm on two synthetic data sets. In future work, we would like to conduct more empirical analysis, analyze additive expert ensembles under other loss functions, and investigate methods for dynamically adjusting the parameters of the algorithm during online learning to maximize performance.

Acknowledgements. The authors thank the Georgetown Undergraduate Research Opportunities Program for funding, and thank Lisa Singh and the anonymous reviewers for helpful comments on earlier drafts of this article.

References

Blum, A. (1997). Empirical support for Winnow and Weighted-Majority algorithms. *Machine Learning*, *26*, 5–23.

Bousquet, O., & Warmuth, M. K. (2003). Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, *3*, 363–396.

Cesa-Bianchi, N., Freund, Y., Haussler, D., Helmbold, D. P., Schapire, R. E., & Warmuth, M. K. (1997). How to use expert advice. *Journal of the ACM*, *44*, 427–485.

Freund, Y., & Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Computer and Sys. Sciences*, *55*, 119–139.

Haussler, D., Kivinen, J., & Warmuth, M. (1998). Sequential predictions of individual sequences under general loss functions. *IEEE Transactions on Information Theory*, *44*, 1906–1925.

Helmbold, D. P., & Long, P. M. (1994). Tracking drifting concepts by minimising disagreements. *Machine Learning*, *14*, 27–45.

Herbster, M., & Warmuth, M. K. (1998). Tracking the best expert. *Machine Learning*, *32*, 151–178.

Kolter, J. Z., & Maloof, M. A. (2003). Dynamic weighted majority. *Proceedings of the 3rd IEEE ICDM Conference* (pp. 123–130). IEEE Press.

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

Littlestone, N., & Warmuth, M. K. (1994). The weighted majority algorithm. *Information and Computation*, *108*, 212–261.

Monteleoni, C., & Jaakkola, T. S. (2004). Online learning of non-stationary sequences. *Proceedings of the 16th NIPS Conference*. MIT Press.

Schlimmer, J., & Granger, R. (1986). Beyond incremental processing: Tracking concept drift. *Proceedings of the 5th AAAI Conference* (pp. 502–507). AAAI Press.

Street, W., & Kim, Y. (2001). A streaming ensemble algorithm (SEA) for large-scale classification. *Proceedings of the 7th SIGKDD Conference* (pp. 377–382). ACM Press.

Vovk, V. (1990). Aggregating strategies. *Proceedings of the 3rd COLT Workshop* (pp. 371–386). Morgan Kaufmann.

Vovk, V. (1998). A game of prediction with expert advice. *J. of Computer and System Sciences*, *56*, 153–173.

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