

Managing Domain Knowledge and Multiple Models with Boosting

Peng Zang

College of Computing
Georgia Tech
Atlanta, GA 30332
pengzang@cc.gatech.edu

Charles Isbell

College of Computing
Georgia Tech
Atlanta, GA 30332
isbell@cc.gatech.edu

Abstract

We present MBoost, a novel extension to AdaBoost that extends boosting to use multiple weak learners explicitly, and provides robustness to learning models that overfit or are poorly matched to data. We demonstrate MBoost on a variety of problems and compare it to cross validation for model selection.

1 Introduction

Recent efforts in ensemble learning methods have shown empirically that the use of many methods and models is more powerful than using any single method alone. For example, Caruana [2004] shows how constructing an ensemble classifier from a library of around 2000 models ranging from decision trees to SVMs produces classifiers that outperform the best of any single model. Oddly, Boosting [Schapire, 1990]—a particularly popular ensemble technique with strong theoretical and empirical support—is rarely used in such a fashion. In principle, Boosting should be able to handle multiple learning methods automatically simply by using a learner that itself chooses among multiple weak learners. On the other hand, such a learner introduces additional complexity and may compound overfitting effects.

In this paper, we present a new boosting algorithm, MBoost, that incorporates multiple models into Boosting explicitly. In particular, we extend AdaBoost [Schapire and Singer, 1999; Freund and Schapire, 1995] to act as an arbitrator between multiple models. Our contributions are:

- A principled technique for extending Boosting to enable explicit use of multiple weak learners.
- A technique for controlling weak learners that overfit or are poorly matched to the data (*e.g.*, embody poor domain knowledge).

In the sections that follow we first present MBoost itself, comparing it to AdaBoost. We analyze the theoretical and practical consequences of MBoost, and validate our analysis by presenting several empirical results, including a comparison of MBoost to model selection via cross validation. Finally, we situate our work in the literature and conclude with some discussion.

2 MBoost

AdaBoost [Schapire and Singer, 1999; Freund and Schapire, 1995] is an ensemble learning technique that iteratively constructs an ensemble of hypotheses by applying a weak learner repeatedly on different distributions of data. Distributions are chosen to focus on the “hard” parts of the data space, that is, where the hypotheses generated thus far perform poorly. If h_1, h_2, \dots, h_T is a set of hypotheses generated by AdaBoost, the final boosted hypothesis is then: $H(x) = \text{sign}(f(x)) = \text{sign}(\sum_{t=1}^T \alpha_t h_t(x))$, where α_t denotes the weighting coefficient for h_t .

Although described as a single learner, the weak learner could be a “bag” of several learning models. For example, rather than choosing between a decision tree and a perceptron as the weak learner, one could use a learner that contained both, presumably choosing the better for any given iteration.

Using such a composite weak learner requires some attention to detail. Whether the bag chooses the best model or combines the models in some way, it introduces additional inductive bias: one is no longer boosting over a decision tree and a perceptron, but over a potentially complex ensemble learner that happens to include a decision tree and perceptron. To ensure that no additional inductive bias is introduced, we propose that the boosting algorithm itself should be the *arbitrator*, acting as the mechanism for choosing which model to use.

The MBoost algorithm explicitly supports multiple weak learners and formalizes the notion of boosting as the arbitrator. In each round, each weak learner proposes a hypothesis and MBoost selects the “best” one. Here, “best” is determined by how boosting would reweight the distribution from round to round, allowing MBoost to arbitrate among the weak learners without introducing any inductive bias not already intrinsic to the boosting framework.

Boosting is known to be susceptible to overfitting when its weak learner overfits. Imagine for example, that the weak learner is a rote learner such as a hash table. The training error will be zero, but there will also be no generalization. No matter how many rounds are run, the rote learner will generate the same hypothesis (assuming there is no noise in the labels). As a result, the final Boosting classifier would show the same (lack of) generalization capabilities. Using multiple weak learners can only compound this problem.

Boosting’s susceptibility to this kind of overfitting lies in

Algorithm 1 MBoost

Require: Weak learners b_1, \dots, b_m Data $(x_1, y_1), \dots, (x_n, y_n)$ where $x_i \in \mathcal{X}, y_i \in \{-1, +1\}$

1. Initialize $D_0(i) = 1/n$
2. **for** $t = 1, \dots, T - 1$ **do**
3. Split data randomly into D_{train_t} and D_{val_t} .
4. Train learners $b_1 \dots b_m$ on D_{train_t} to generate hypotheses $h_1 \dots h_m$.
5. Choose hypothesis $h_t = \operatorname{argmin}_{Z_t}(h_1 \dots h_m)$
6. Choose α_t for h_t per usual
7. Update: $D_{val_{t+1}}(i) = D_{val_t}(i)e^{-\alpha_t y_i h_t(x_i)}$
8. Normalize $D_{t+1} = D_{t+1}/Z_t$ such that $\sum_i D_{t+1}(i) = 1$
9. **end for**
10. **return** Final hypothesis: $H(x) = \operatorname{sign}(\sum_{t=1}^T \alpha_t h_t(x))$

Where \mathcal{X} is the input space and Z_t is the normalization constant.

part in its use of training error for hypothesis evaluation. Training error can be an overly-optimistic and poor measure of true performance. Rather than avoid “stronger” learners, MBoost divides the data every round into training and validation sets, using the latter to evaluate generated hypotheses. This approach yields a more accurate measure of the hypotheses’ performance which in turn is used to choose the best hypothesis and its weight. Additionally, MBoost only reweights the distribution on data from the validation set. Points from the training set are suspect as their prediction may be due to overfitting causes. Reweighting these points may incorrectly imply that they have been learned. Reweighting only the validation set avoids making such a mistake. Note that the convergence of the algorithm does not change. All points will eventually be reweighted since points in the training set of one round may very well be in the validation set of the next.

The combination of these two techniques means MBoost has a much more accurate measure of the hypotheses’ generalization errors. Further, by being “inside” the bag of competing models, MBoost can choose the hypothesis that directly minimizes the loss function. The net effect is that users can insert a variety of models into boosting to take advantage of the empirical power of using multiple models while mitigating the effects of poor weak learners that either overfit or embody incorrect domain knowledge.

Algorithm 1 shows pseudo-code for MBoost. We note that while MBoost as shown is derived from AdaBoost, any boosting scheme can be similarly adapted.

3 Analysis and Implications

3.1 Using multiple weak learners

Boosting can be viewed as a coordinate gradient descent method that searches for the set of hypotheses which, when combined with their weights, minimizes some loss. In each round of AdaBoost, a hypothesis h_t (the coordinate) is given, and AdaBoost optimizes the weight α_t for that coordinate to minimize the loss function. An interpretation of this process

is that given the coordinate, AdaBoost is taking the largest possible loss-minimizing step along that coordinate.

When Boosting is extended to multiple weak learners, not only must the optimal weight be chosen, but the optimal hypothesis (among a set of proposed hypotheses) must be chosen as well. In other words, we must find the best coordinate (the one with the largest component in the direction of the gradient) in addition to optimizing the loss-minimizing step.

Because we do not know our learning models *a priori*, we are precluded from analytical optimizations that find the best hypothesis. Instead, we perform an exhaustive search. For each hypothesis proposed by the different models, we find its optimal weight and calculate the resulting loss. The hypothesis with the lowest loss is chosen. Because the number of weak learners is typically small and computation of the loss requires only hypothesis predictions and not training, the cost of this step is low.

Applying this technique to AdaBoost, we see that the best hypothesis is the one minimizing $G(f) = \sum_{i=1}^N e^{-y_i f(x_i)}$, where y_i is the true label and $f(x) = \sum_{t=1}^T \alpha_t h_t(x)$ is the hypotheses ensemble classifier. The cost of this minimization grows linearly with the number of rounds run. Over many rounds, it can become expensive. Fortunately, minimizing $G(f)$ is equivalent to minimizing Z_t , the normalization constant in AdaBoost, in that round.

Proof. In any particular round t , the update rule gives us:

$$D_{t+1}(i) = D_t(i)e^{-y_i \alpha_t h_t(x_i)} / Z_t \quad (1)$$

$$= \frac{e^{\sum_t -y_i \alpha_t h_t(x_i)}}{m \prod_t Z_t} \quad (2)$$

$$= \frac{e^{-y_i f(x_i)}}{m \prod_t Z_t} \quad (3)$$

Because Z_t must normalize $D_{t+1}(i) = 1$, we have:

$$Z_t = \sum_i D_t(i)e^{-y_i \alpha_t h_t(x_i)} \quad (4)$$

$$= \sum_i D_{t+1}(i)Z_t \quad (5)$$

$$= \sum_i \frac{e^{-y_i f(x_i)}}{m \prod_{j=1}^{t-1} Z_j} \quad (6)$$

Thus

$$\operatorname{argmin}(G(f)) = \operatorname{argmin}\left(\sum_{i=1}^N e^{-y_i f(x_i)}\right) \quad (7)$$

$$= \operatorname{argmin}\left(\frac{1}{m \prod_{j=1}^{t-1} Z_j} \sum_i e^{-y_i f(x_i)}\right) \quad (8)$$

$$= \operatorname{argmin}(Z_t) \quad (9)$$

□

We note that in the MBoost formulation, Z_t is defined somewhat differently than in AdaBoost. This is because MBoost’s use of a validation set changes the loss function; however, the difference is slight and the proof still applies.

Finally, it is worth noting that because MBoost is choosing among multiple hypotheses, the weak learner requirement (in the PAC sense) can be relaxed. The ensemble of learners must act as a weak learner but no individual learner must.

3.2 Using an internal validation set

MBoost needs an accurate error estimate for candidate hypotheses each round. Following [Langford, 2005], we model hypothesis error as a biased coin flip where the bias is the true error, the number of tails is the number of correct predictions and the number of heads is the number of incorrect predictions.

The probability that one makes k or fewer mistakes in n predictions given that the true error rate is ϵ is given by the binomial cumulative distribution function: $CDF(k, n, \epsilon) = \sum_{i=0}^k \binom{n}{i} \epsilon^i (1 - \epsilon)^{n-i}$.

The largest true error rate r such that the probability of having k or fewer mistakes in those n predictions is at least δ , is then: $\max_r (r : CDF(k, n, r) \geq \delta)$. We will refer to this as the maximum reasonable true error: $m rte(k, n, \delta)$. A bound on the true error can then be formed as: $P(\epsilon \leq m rte(k, n, \delta)) \geq 1 - \delta$.

In MBoost, a hypothesis is only used if its $m rte(k, n, \delta)$ is less than 0.5. That is, a hypothesis is only used if the upper bound of its true error is less than 0.5, suggesting with high confidence that the hypothesis is better than random.

Recall that we can view Boosting as a coordinate gradient descent method optimizing a particular loss function. Use of the validation set for hypothesis selection, weighting and distribution reweighting essentially amounts to a different loss function: $\sum_{i=1}^N e^{-y_i \sum_t h_t(x) I(x_i \notin D_{train_t})}$. MBoost's loss function reflects our understanding that the real goal of interest is the ensemble's generalization error. We want to avoid being misled by atypical results on training data. Use of the test set for evaluation allows MBoost to effectively apply weak learners prone to overfitting (as a "strong" learner may do).

One could easily imagine extending the use of a single internal validation set to perform full 10-fold cross-validation per round of boosting. We note however, that such a treatment aims towards finding the best weak learner, not the best hypothesis. In building an ensemble of hypotheses, our goal is to find the best hypothesis per round.

Finally, we note that internal validation is also mildly helpful when overfitting is due to noise. Such error is a side effect of boosting's emphasis on "hard" examples, or in noisy situations, noise. If noise is non-systematic, MBoost should determine that the generated hypotheses cannot predict the noise better than random. As we note in the next section, this will cause MBoost to halt.

3.3 Automatic stopping condition

It is natural for MBoost to stop when none of its weak learners perform better than random. Standard boosting uses training data for evaluation so it may be easier for weak learners to perform better than random than in MBoost. In running our experiments, MBoost usually stops due to this condition.

For MBoost to determine that none of its weak learners can perform better than random, a single round in which no generated hypotheses perform is insufficient as the training

and validation sets are chosen randomly. To determine that the learners are indeed "exhausted", many consecutive rounds in which no hypothesis beats random is required.

We note that in these rounds, the distribution is static. MBoost is essentially performing an internal Monte-Carlo cross-validation on each of its weak learners. As we run more rounds, we acquire tighter bounds on each learner's true error. Eventually, this will either show that at least one of the weak learners does better than random, in which case MBoost will continue, or that none of the weak learners can do better than random (with some confidence), in which case MBoost halts.

3.4 Error bounds

MBoost can be approximately reduced to AdaBoost. To see this, consider the weak learner L which when trained on data set D_t in round t does the following:

- Splits the data set into a training set and a validation set, D_{train_t} and D_{val_t} .
- Trains each of its member learners $b_1 \dots b_m$ on D_{train_t} , generating hypotheses $h_1 \dots h_m$.
- Simulates for each hypothesis the boosting reweighting on D_{val_t} and chooses the hypothesis $h_{best,t}$ with the lowest hypothetical normalization constant Z .
- Returns $h'_t(x) = h_{best,t}(x) I(x \notin D_{train_t})$, where $I(p) \in \{0, 1\}$ is the indicator function that returns 1 when the predicate p is true, and 0 otherwise. In other words, return $h'_t(x)$ as a copy of $h_{best,t}(x)$ except that when asked to predict on data it has been trained on it abstains.

Applying AdaBoost to the weak learner L , we construct a learner L_{ada} . This is essentially MBoost. The sole difference is that MBoost's final hypothesis is $H(x) = \text{sign}(\sum_{t=1}^T \alpha_t h_{best,t}(x))$ instead of $H'(x) = \text{sign}(\sum_{t=1}^T \alpha_t h'_t(x))$; however, $h'_t(x)$ and $h_{best,t}(x)$ differ only in predictions on a randomly-drawn finite set, namely, D_{train_t} . Thus, $H'(x)$ and $H(x)$ can differ only on points in the union of $D_{train_1} \dots D_{train_T}$, a subset of the overall training data D . In the limit, as the size of the instance space approaches infinity, the probability of predicting on a data point seen in training is vanishingly small. More formally, $\lim_{|\chi| \rightarrow \infty} Pr(x \in D) = 0$

and therefore $\lim_{|\chi| \rightarrow \infty} |H'(x) - H(x)| = 0$ where χ is the instance space and x is a point randomly drawn from χ . Further:

- The sum of all errors is bounded by the size of the training set: $\sum_{x \in \chi} |H'(x) - H(x)| < |D|$.
- $H'(x)$ and $H(x)$ differ only on a randomly drawn, finite set, and the errors $|H'(x) - H(x)|$, are not systematic. In other words, $\sum |H'(x) - H(x)| \approx 0$. $H'(x)$ and $H(x)$ represent the same decision boundary.

$H'(x) \approx H(x)$ and thus, L_{ada} is approximately *MBoost*.

This reduction now allows us to inherit the generalization error bounds that have been derived for AdaBoost. In particular, both the VC-theory based generalization error bound found in [Freund and Schapire, 1995], as well as the margin-based generalization bounds in [Schapire *et al.*, 1997] and [Schapire and Singer, 1999] can be applied straightforwardly.

4 Empirical evaluation

We performed four sets of experiments to examine MBoost’s performance.

4.1 Effects of domain knowledge

This set of experiments focus on exploring MBoost’s basic ability to exploit multiple domain knowledge sets, as encoded by the weak learners; the validation set is not used. We used a 2D synthetic dataset composed of three Gaussians whose points are labeled according to three different lines, one for each Gaussian. Two weak learners were used. The first is a simple perceptron. The second is a Gaussian estimator combined with a perceptron. This represents the accurate domain knowledge case.

As we can see in Figures 1a and 1b, this is an easy dataset and both AdaBoost and MBoost are able to approach perfect classification. MBoost, however, by leveraging the additional domain knowledge in the second weak learner, is able to do so five times faster.

To explore the effect of inaccurate domain knowledge, we replace the second weak learner with one that learns vertical bands of width 0.1. This hypothesis class can yield no helpful hypotheses, simulating poor domain knowledge. In this case, MBoost discards the hypotheses generated by the second weak learner. MBoost and AdaBoost perform identically from round to round.

4.2 MBoost versus AdaBoost on individual learners

In this set of experiments, we explore the effect of boosting multiple models together, versus boosting each model individually. We want to determine if heterogeneous ensembles are superior to homogeneous ones. We performed the experiments on five of the largest binary classification UCI datasets¹. A set of twenty-five weak learners were used:

- Five Naive Bayes learners. One uses empirical probabilities, the other four use m-estimates with m at 4, 16, 64, and 256. We used the Naive Bayes learner provided in the Orange data mining package.
- Five kNN learners with k at 1, 4, 16, 64, and 256. We used the kNN learner provided in the Orange data mining package.
- Five C4.5 decision tree learners with *minObjs*, the minimum number of examples in a leaf, set at 1, 4, 16, 64 and 256. All other options were set to defaults. We used R8 of Quinlan’s implementation.
- Ten SVM learners with C , the parameter for the cost of misclassification, set at 0.125, 0.5, 2, 8, 32, 128, 512, 2048, 8192, and 32768. The SVMs used a RFB kernel. All other options were set to defaults. We used the libsvm library by Chih-Chung Chang and Chih-Jen Lin.

We compared two learners. The first, *MBoost10*, is given all twenty-five weak learners and set to run for ten rounds. The second is a learner that runs AdaBoost for 10 rounds with

¹The full ADULT data set is quite large, we used a subsample of 1000 examples.

Table 1: CV Accuracy for MBoost and BestAda

<i>Dataset/Learner</i>	<i>MBoost10</i>	<i>BestAda</i>	<i>p-value</i>
ADULT	0.837	0.765	0.0000
BREAST-CANCER	0.751	0.706	2.12e-04
CRX	0.874	0.809	0.0000
HORSE-COLIC	0.816	0.777	0.0016
IONOSPHERE	0.947	0.883	2.89e-12

Table 2: CV Accuracy for MBoost10 and BestCV-Ada

<i>Dataset/Learner</i>	<i>MBoost10</i>	<i>BestCV-Ada</i>	<i>p-value</i>
ADULT	0.837	0.812	0.0038
B-CANCER ²	0.751	0.731	0.0213
CRX ²	0.874	0.861	0.0323
HORSE-COLIC	0.816	0.818	0.9601
IONOSPHERE	0.947	0.946	0.6145

each of the weak learners in turn (for a total of twenty-five runs). The best AdaBoosted model (according to their reported training error) is chosen and used. We call this second learner *BestAda*. Note that the computational complexity of the two learners are equivalent, they are $O(nm)$ where n is the number of rounds and m is the number of weak learners.

Table 1 shows the accuracy results on the five datasets. All reported accuracy scores use subsample cross-validation 50 times; that is, in each cross validation round we randomly split the data into training and validation sets with 90% in the training set. We use this variation of cross validation throughout our evaluation because it can be performed any number of times. We find that 50 rounds typically produce enough samples for our significance tests.

We used one-way ANOVA tests to determine if the reported accuracies are statistically significant, $pvalue = F(1, 98)$. All datasets show MBoost performing statistically significantly better.

These results are quite surprising. Detailed analysis shows that BestAda suffered from significant overfitting. For example, AdaBoosted SVM would report a higher accuracy rate than AdaBoosted C4.5 when in fact AdaBoosted C4.5 generalized better.

One might imagine using cross-validation to avoid this problem. To this end, we create a third learner, *BestCV-Ada*, that is the same as *BestAda* but uses ten-fold cross validation to select the best AdaBoosted weak learner. Note that this increases the computational cost of *BestCV-Ada* by a factor 10. We present results comparing *BestCV-Ada* and *MBoost10* in Table 2.

These results are in line with expectations. As commonly reported in the literature, the best single-model booster is quite effective. Nevertheless, we see that MBoost, using one tenth of the computational cost, can perform better on three of the five data sets and as good on the rest. Heterogeneous ensembles appear to perform better than homogeneous ones. MBoost is able to take advantage of this effect and combine

²Additional CV rounds were performed to ascertain statistical significance.

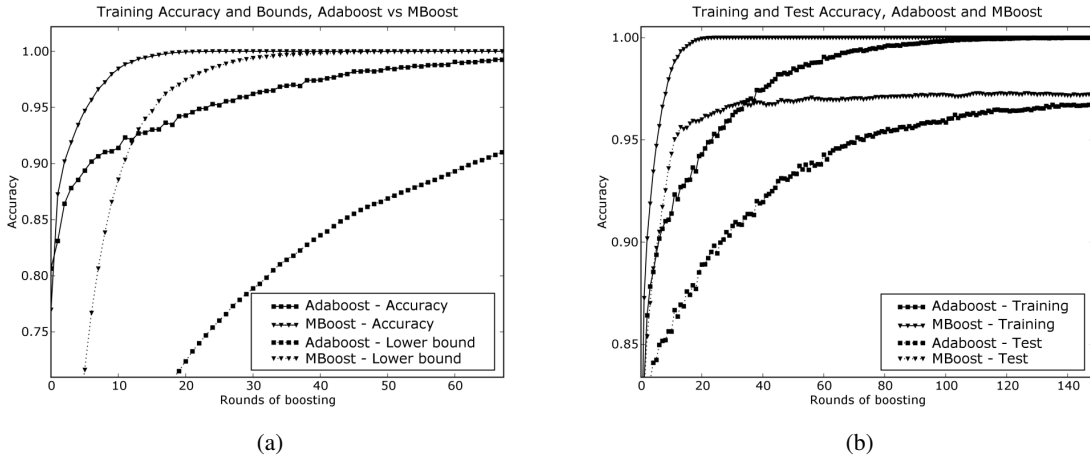


Figure 1: Comparing AdaBoost and MBoost on Gaussian dataset. Left shows training accuracy and theoretical bound, right shows training and testing accuracy. Averaged over 50 runs

Table 3: CV Accuracy for MBoost and MBoostAcc

Dataset/Learner	MBoost	MBoostAcc	<i>p</i> -value
ADULT	0.837	0.836	0.8685
B-CANCER	0.751	0.739	0.1752
CRX	0.874	0.876	0.8494
HORSE-COLIC	0.816	0.815	0.8867
IONOSPHERE	0.947	0.945	0.7399

Table 4: CV Accuracy for MBoost-10 and BestCV-Ind

Dataset/Learner	MBoost10	BestCV-Ind	<i>p</i> -value
ADULT	0.837	0.833	0.5162
B-CANCER	0.751	0.745	0.5233
CRX ²	0.874	0.862	0.0326
HORSE-COLIC	0.816	0.813	0.7761
IONOSPHERE	0.947	0.947	0.9294

the ensembles appropriately.

4.3 MBoost as weak learner arbitrator

Here, we perform a set of experiments to explore the effects of using MBoost’s loss function as a weak learner arbitrator. We compare two versions of MBoost. The first is standard MBoost using its exponential loss function as the arbitrator, the second, *MBoostAcc* using validation accuracy of the weak learners. Both boosters are run for 10 rounds.

Table 3 shows that the two versions’ performance are quite similar. Only one dataset shows any difference that might be statistically significant with *MBoost* outperforming *MBoostAcc*. In further exploration, we ran the same experiment but allowed both versions to run until exhaustion. In this experiment, all performance differences disappeared. This is unsurprising as the performance difference should be in terms of speed of convergence, not final asymptomatic performance. The exponential loss function is more aggressive, so it is reasonable that it might show some early gains.

4.4 MBoost versus the best individual learners

Here we perform a series of experiments comparing MBoost with the best individual model as selected via 10 fold cross validation (*BestCV-Ind*). We ran two versions of MBoost, one running for 10 rounds and one with the automatic stopping condition, though never more than 50 rounds. *MBoost10* has the same computational complexity as selecting the best

Table 5: CV Accuracy for MBoost10 and MBoostAuto

Dataset/Learner	MBoost10	MBoostAuto	<i>p</i> -value
ADULT	0.837	0.838	0.8142
B-CANCER	0.751	0.750	0.9362
CRX	0.874	0.876	0.7147
HORSE-COLIC	0.816	0.818	0.8454
IONOSPHERE	0.947	0.949	0.7518

individual model via 10-fold cross validation. *MBoost-Auto* typically requires two to five times the computational cost.

Table 4 shows *MBoost10* performing better on one of the five datasets and equivalently on the rest. Further, as we can see in Table 5, the accuracy scores do not degrade even as we run MBoost to exhaustion. MBoost is quite resistant to overfitting. These factors combined with MBoost’s low computational complexity lead us to suggest it as an alternative to cross-validation for model selection. Instead of trying many models and finding the best, we suggest Boosting the models together and synthesizing an ensemble whole.

5 Related Work

MBoost’s use of multiple weak learners is similar to a boosting extension known as localized or dynamic boosting ([Moerland and Mayoraz, 1999; Meir *et al.*, 2000; Avnimelech and Intrator, 1999; Maclin, 1998]) where α_t —the weight on the hypothesis in round t —is generalized to be a function depen-

dent on the data, x . Localized boosting can be viewed as a way to boost across two weak learners: one in the hypothesis and one in the $\alpha_t(x)$. Our work aims to explicitly enable multiple weak learners in general, where any number of weak learners are allowable. MBoost can emulate localized boosting by treating $\alpha_t(x)h_t(x)$ as one weak learner, but we point out that—ignoring MBoost’s use of internal validation—both extensions are mathematically reducible back to boosting. The difference lies in that MBoost makes boosting across multiple learners explicit. We hope that this formulation is easier to reason about, and that at the very least, provides a clearer mechanism for the user to apply multiple weak learners and the domain knowledge they embody.

We know of no other work in boosting for controlling weak learner overfitting that is similar to our use of internal validation; however, there is general work in ensemble learning (again, see [Caruana *et al.*, 2004]). Our work differs in that it is integrated into the boosting framework, and so we leverage boosting’s distribution perturbation and the use of hypotheses trained on different parts of the data space.

MBoost’s ability to mitigate the effects of poor weak learners also makes it suitable for arbitrating between and merging different sets domain knowledge embodied in the weak learners. There has been some prior work focused on incorporating knowledge into Boosting, eg. [Schapire *et al.*, 2002]. Their approach is to add a prior model, informed by domain knowledge, mapping each instance of the data to a probability over the possible label values. The Boosting process is modified to also fit the prior. Requiring a prior can be burdensome. Further, if the prior is incorrect, performance will suffer. Priors also offer no way to reconcile different sets of domain knowledge. MBoost relaxes these constraints. Weak learners provide a very flexible domain knowledge insertion point. MBoost can further mitigate any negative effects should incorporated knowledge be inaccurate. Finally, MBoost can merge the knowledge sets into an ensemble whole.

6 Conclusion

We have introduced a novel boosting algorithm, MBoost, that explicitly chooses among multiple models, controlling for those weak learners that overfit or are otherwise poor model fits for the data. MBoost takes advantage of results in ensemble learning while retaining the theoretical and empirical strengths of boosting. It provides us an effective way to manage and use multiple models and the domain knowledge they embody.

Empirical evaluations show MBoost can outperform any single model and any boosted single model on some data sets and performs at least as well on the rest. Furthermore, MBoost provides a natural stopping criterion that appears robust to overfitting. Favorable performance and MBoost’s low computational complexity also lead us to suggest it as an alternative to model selection via cross validation.

Acknowledgments

We thank Alex Gray and John Cassel for providing useful pointers to *hunch.net* among other helpful discussions. The breast cancer databases used in our benchmarks was obtained

from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. We acknowledge the support of DARPA under contract No. HR0011-04-1-0049.

References

- [Avnimelech and Intrator, 1999] Ran Avnimelech and Nathan Intrator. Boosted mixture of experts: An ensemble learning scheme. *Neural Computation*, 11(2):483–497, 1999.
- [Caruana *et al.*, 2004] Rich Caruana, Alexandru Niculescu, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proc. 21st International Conference on Machine Learning*, pages 137–144, 2004.
- [Demsar J, 2004] Leban G Demsar J, Zupan B. Orange: From experimental machine learning to interactive data mining, 2004.
- [D.J. Newman and Merz, 1998] C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.
- [Freund and Schapire, 1995] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.
- [Langford, 2005] John Langford. Tutorial on practical prediction theory for classification. 2005.
- [Maclin, 1998] Richard Maclin. Boosting classifiers regionally. In *AAAI/IAAI*, pages 700–705, 1998.
- [Mangasarian and Wolberg, 1990] O. L. Mangasarian and W. H. Wolberg. Cancer diagnosis via linear programming. *SIAM News*, 23(5):1,18, 1990.
- [Meir and Ratsch, 2003] Ron Meir and Gunnar Ratsch. An introduction to boosting and leveraging. *Advanced lectures on machine learning*, pages 118–183, 2003.
- [Meir *et al.*, 2000] Ron Meir, Ran El-Yaniv, and Shai Ben-David. Localized boosting. In *Proc. 13th Annual Conference on Computational Learning Theory*, pages 190–199. Morgan Kaufmann, 2000.
- [Moerland and Mayoraz, 1999] Perry Moerland and Eddy Mayoraz. Dynaboost: Combining boosted hypotheses in a dynamic way. IDIAP-RR 09, IDIAP, 1999.
- [Schapire and Singer, 1999] Robert E. Schapire and Yoram Singer. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.
- [Schapire *et al.*, 1997] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. 14th International Conference on Machine Learning*, pages 322–330. Morgan Kaufmann, 1997.
- [Schapire *et al.*, 2002] Robert E. Schapire, Marie Rochery, Mazin G. Rahim, and Narendra Gupta. Incorporating prior knowledge into boosting. In *ICML*, pages 538–545, 2002.
- [Schapire, 1990] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.