Effects of the Strength of the Weak Learner on Boosting

Thaxila Karunatillake†, Sean Holden‡, and Bernard Buxton‡

†Cambridge University Engineering Department, Trumpington Street, Cambridge CB2 1PZ, UK.
‡Department of Computer Science, University College London, Gower Street, London WC1E 6BT, UK
†jtw28@eng.cam.ac.uk,‡{s.holden, b.buxton}@cs.ucl.ac.uk

Abstract. AdaBoost boosts the performance of a weak learner by training a committee of weak learners which learn different features of the training sample space with different emphasis and jointly perform classification or regression of each new data sample by a weighted cumulative vote. We use RBF kernel classifiers to demonstrate that boosting a Strong Learner generally contributes to performance degradation, and identify three patterns of performance degradation due to three different strength levels of the underlying learner. The strength is a function of both training and generalization performances of the learner. We demonstrate that boosting productivity increases, peaks and then falls as the strength of the underlying learner increases. We highlight patterns of behavior in the distribution and argue that AdaBoost's characteristic of forcing the strong learner to concentrate on the very hard samples or outliers with too much emphasis is the cause of performance degradation in Strong Learner boosting. However, by boosting an underlying classifier of appropriately low strength, we are able to boost the performance of the committee to achieve or surpass the performance levels achievable by strengthening the individual classifier with parameter or model selection in many instances. We conclude that, if the strength of the underlying learner approaches the identified strength levels, it is possible to avoid performance degradation and achieve high productivity in boosting by weakening the learner prior to boosting ...

1 Introduction

Freund & Schapire's algorithm AdaBoost and its variants have been applied extensively for boosting the performance of weak learning algorithms. However, boosting tends to fail or degrade performance in some instances. Quinlan [9] first attracted attention to performance degradation in his early experiments of boosting C4.5. Freund & Schapire [3] record further results on performance degradation of boosting C4.5. In their recent work, Freund & Schapire [4] pose this as an open problem: can we characterize or predict when boosting will fail in this manner?

In Freund & Schapire's study [3] it is notable that the underlying classifier is already a very strong learner with over 90% generalization performance. In further applications researchers have observed that boosting sometimes fails when the weak learner is overly complex. Furthermore, Schapire records [10] that, when the number of outliers is very large, the emphasis placed on the hard samples can become detrimental to the performance of AdaBoost. Onoda, Ratsch and Muller [7] argue that AdaBoost is not independent of the hypothesis class being used, and that the character of the annealing process in boosting will vary greatly depending on the strength of the learner.

We are strongly motivated by the above to research the effects of the strength of the learner on boosting performance, paying specific attention to instances of performance degradation. This paper is an exposition of the effects of the strength of the underlying learner on the performance of AdaBoost and a response to the open problem posed by Freund & Schapire.

For our analysis it is important to form a clear notion of the strength of the learner on a given problem. The learnability of a problem by a classifier is dependent on the inherent difficulty of the problem, the capacity of the classifier, and the size of the training dataset relative to the size of the problem. Given a learning problem posed by a training dataset and a test dataset of fixed size, we form a judgement on the strength of a learner of fixed capacity, based on how well it learns the original training data (training accuracy) supported by how well it performs on the test data (test/generalisation accuracy). The capacity of the classifier is fixed for a committee, so that each committee is made up of instances of the same learner having learned different regions of the feature space. Furthermore, the classifier strength is defined with respect to the given instance of the problem, as posed by the given fixed training and test dataset, and not the abstract problem. Committee performances and subsequent performance improvements are judged with respect to the same dataset.

Variation in learner strength is achieved by varying the capacity of the learner from one committee to another. It is possible to achieve strength variation in kernel machines by varying a free parameter or the kernel itself. Thus, to achieve a weaker classifier relative to the given problem instance, it is only necessary to vary a parameter or the kernel, so that the classifier gives a lower training accuracy and a lower test accuracy according to the given training and test datasets respectively. We choose to vary the number of RBF kernels (K) in the RBF classifier in our experiments. It is noted that the main measure of classifier strength, the training accuracy, increases with K.

We fix K for each committee and watch the annealing process variations as the strength of the learner increases with K from one committee to another. We watch the effects of the strength of the learner on the distribution and highlight very insightful patterns in its behavior. The results highlight the influence of the strength of the learner on the boosting process at different levels of classifier strength. They highlight three different patterns of performance degradation depending on three different high-strength levels of the base learner. From our results we argue that AdaBoost's characteristic of forcing an underlying strong

learner to concentrate on a few hard-to-learn samples or outliers with too much emphasis is the probable cause of performance degradation in boosting. We observe that boosting a learner of moderate strength is optimally productive and often achieves performance levels that surpass the performance of the strengthened individual classifier. We conclude by proposing that a learner approaching the high strength levels identified on a given problem instance be weakened prior to boosting, to achieve optimal productivity and to lower the probability of performance degradation.

2 Background and the Experiments

Given: Training dataset: $(x_1, y_1), \ldots, (x_N, y_N)$
Test dataset: $(x_1, y_1), \ldots, (x_M, y_M)$
where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize: $D_1(i) = \frac{1}{N}$
For $t = 1, \ldots, T$

- Train weak learner with fixed capacity according to distribution D_t
- If $(t = 1)$
  $\epsilon_{train} \leftarrow$ error on Training dataset
  $\epsilon_{test} \leftarrow$ error on Test dataset
  Classifier Strength $:= f(1 - \epsilon_{train}, 1 - \epsilon_{test})$
- Get weak hypothesis $h_t : X \rightarrow \{-1, +1\}$
  with error $\epsilon_t = Pr_{i-D_t}[h_t(x_i) \neq y_i]$
- Update
  $$D_{t+1} = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha t} & , \text{if } h_t(x_i) = y_i \\ e^{\alpha t} & , \text{if } h_t(x_i) \neq y_i \end{cases}$$
  where Z_t is the normalization factor chosen so that D_{t+1} is a distribution

Output Classifier Strength and the final hypothesis
$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Fig. 1. AdaBoost.M1 (Freund & Schapire) with a measure of the underlying classifier strength.

AdaBoost: The boosting algorithm AdaBoost.M1 is used for boosting RBF classifiers on the two class classification problems Monks1, Monks2, Monks3, Poisonous Mushroom, Credit Scoring and Tic-Tac-Toe Endgame from the UCI data repository.

Training the Learner: A Radial Basis Function classifier is trained as the underlying learner with variations in its strength being achieved by changing the number K of kernels in the second layer. K is an input parameter to the training

algorithm and is allowed to take a value between 2 and N, the number of training samples. The first layer projects the input samples to the K dimensional feature space by using spherical Gaussian kernels. The K function centers are trained by K-means clustering of the input samples. The function widths are set to twice the mean distance between the function centers. The final layer optimizes the input samples projected onto the feature space by linear optimization of the sum-of-squares error, and the real valued output is thresholded to provide a binary classification. The training algorithm allows us to train an RBF classifier of desired complexity on the training dataset.

Given K and T, the number of iterations, AdaBoost trains a committee of T RBF classifiers, each with approximately K number of kernels. A classifier is trained according to the distribution by sampling the training data according to the distribution with replacement.

The graphs of Fig.2 plot the performance of the learners against K for the six datasets.

The Strength of the Learner: The strength of the classifier is defined with respect to the problem instance given by a fixed dataset, not the unknown problem. Subsequent performance improvements are judged relative to the fixed dataset. The complexity of the classifiers for a particular AdaBoost run is also fixed, so that all the classifiers of a particular committee have approximately same capacity.

Within this context, an intuitive measure of the strength of the classifier being boosted in a particular committee is formed, based primarily on its performance on the training samples and supported by its performance on the test samples. The accuracy on the training samples is an important measure of the strength of the classifier in boosting as the committee concentrates on reducing its training error in a greedy manner by AdaBoost's choice of the voting parameter α_t. [10] However, we find that the generalization accuracy must also support its strength. A classifier showing 95% accuracy on the given training dataset and 95% accuracy on the given test dataset is clearly a strong learner of the problem instance. The RBF classifier with 25 function centers on the Tic-Tac-Toe dataset having approximately 90% training accuracy and 95% test accuracy is a significantly stronger learner of its problem instance than the RBF classifier with 30 function centers on the Monks1 dataset also having approximately 90% training accuracy, but only 82% test accuracy. However, a very high training accuracy (e.g. 94%) may be supported by a moderate test accuracy (e.g. 75%) and still show strong learner behavior. Alternatively, when the test accuracy is very high (e.g. 97% for Tic-Tac-Toe) strong learner behavior can start at classifiers with a lower training accuracy.

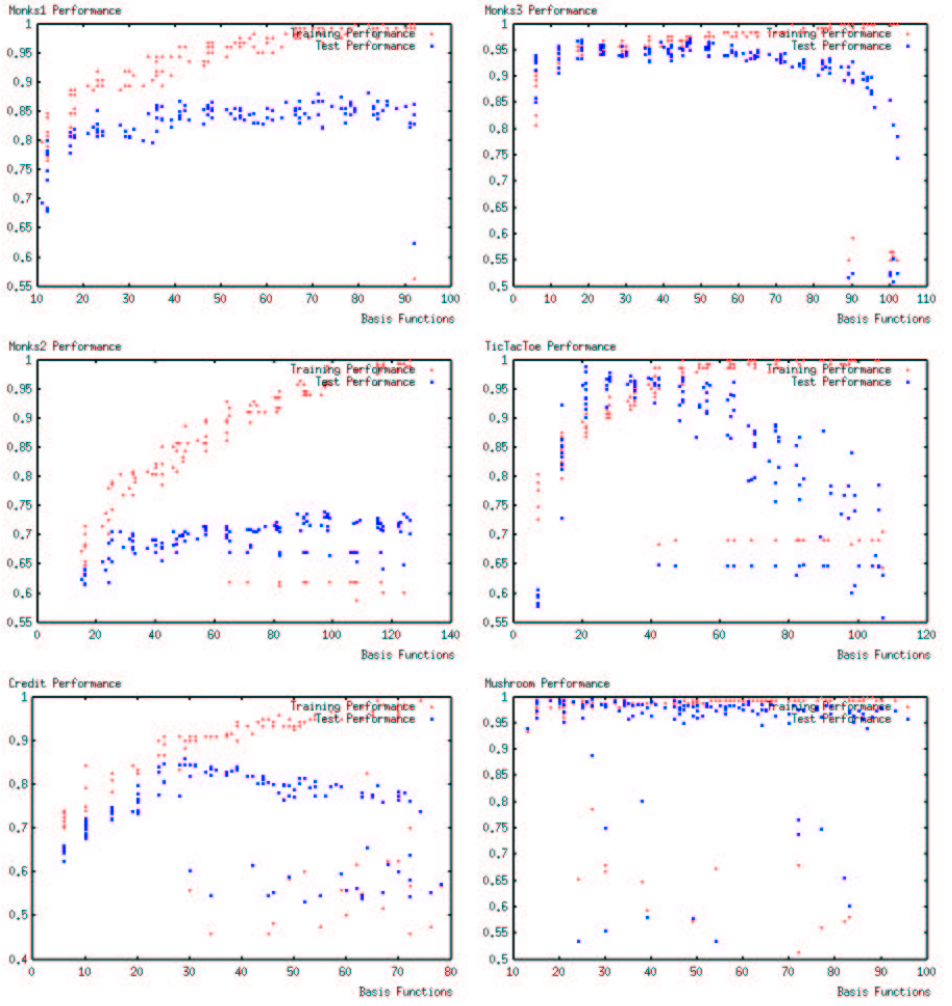


Fig. 2. RBF performance graphs for Monks1, Monks2, Credit, Monks3, Tic-Tac-Toe and Mushroom datasets. Performance on the training data (plotted in red dots) and on the test data (plotted in blue squares). The RBF classifier starts as a weak learner and becomes a strong learner as the complexity increases for Monks1, Monks2, and Credit Scoring datasets. It is a strong learner of Monks3, Tic-Tac-Toe and Mushroom datasets. (Unusually sub-optimal classifiers are trained due to the inverted matrix in linear optimization being near degenerate.)

3 Boosting a Strong Learner is Generally Counterproductive or Unproductive

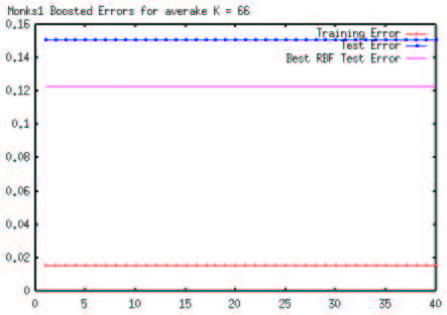
As the strength of the classifier increases beyond approximately 90% (training accuracy above 90% and supported by high test performance), boosting consistently becomes counterproductive or unproductive. For the particular datasets the onset of unproductive or counterproductive boosting behavior occurs when the base classifier is stronger than (approximately) the classifiers of Table 1. For all RBF classifiers stronger than these on the particular datasets, unproductive or counterproductive boosting behavior is consistent.

Table 1. The classifier strength at which consistent unproductive or counterproductive behavior starts for datasets tested

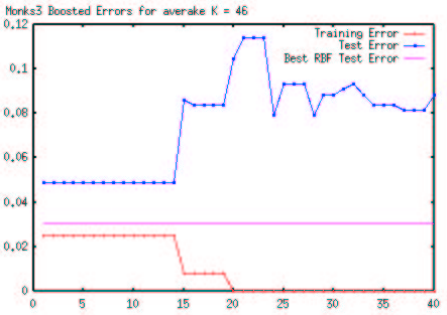
Dataset/Problem Title	*Complexity*	*Training Performance*	*Test Performance*	*Behavior*
Monks1	47	92.5	85.0	Unproductive
Monks2	75	92.5	72	Unproductive
Credit Scoring	45	93.9	81.0	Unproductive
Monks3	18	93.0	93.0	Counterproductive
Tic-Tac-Toe Endgame	20	89.0	95.0	Counterproductive
Poisonous Mushroom	8 (lowest tested)	96.0	98.0	Counterproductive

It is noted that, in the case of unproductive behavior, the high training accuracy is supported by a relatively modest test performance. Careful observation of the graphed distribution indicates that a relatively small number of samples gets strongly highlighted and that their emphasis in the distribution keeps growing at the expense of the emphasis on the other samples throughout the boosting process, but never comes down. The low generalization performance of the base classifier seems to contribute to AdaBoost's being unable to train a classifier in subsequent iterations that is capable of learning the highlighted very hard samples. This results in AdaBoost's being unable to affect either training or test error.

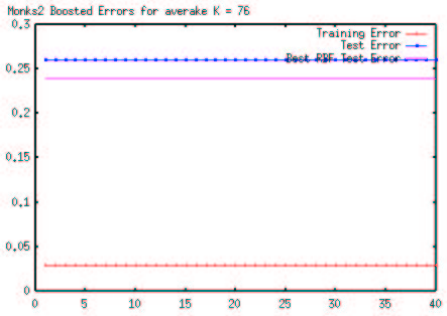
When a high training accuracy is supported by a high generalization performance, AdaBoost is able to train classifiers that learn the highly emphasized hard samples. Then the training error drops to zero, but the test error continues to increase throughout. The distribution repeatedly highlights a small number (generally less than 5) of samples very strongly; even though their weighting comes down with a good hypothesis that learns them, they are soon highlighted strongly again and again. Hence too much emphasis is placed on the hard-to-learn outliers or a few very hard samples in the training data throughout, causing AdaBoost to continuously overfit the training data.



Monks1: Boosting is unproductive
for strong learner with K=66,
training accuracy 96%,
test performance 84%

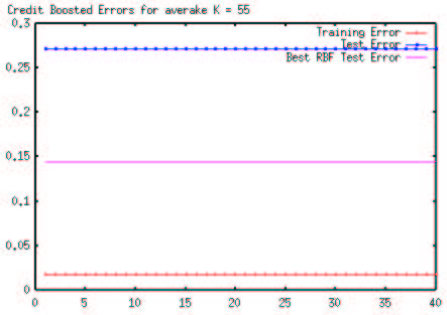


Monks3: Boosting is counterproductive
for strong learner with K=46,
training accuracy 96%,
test performance 95%



Monks2: Boosting is unproductive
for strong learner with K=76,
training accuracy 93%,
test performance 72%

Mushroom: Boosting is counterproductive
for strong learner with K=24,
training accuracy 97.5%,
test performance 98%



Credit: Boosting is unproductive
for strong learner with K=55,
training accuracy 95%,
test performance 80%

Tic-Tac-Toe: Boosting is counterproductive
for strong learner with K=24,
training accuracy 92%,
test performance 96%

Patterns I: Unproductive Boosting:
When the learner training accuracy
is high and this is supported by a
modest test accuracy

Pattern II - Continuously Counter-
productive Boosting: When the learner
training accuracy is high and this is
supported by a high test accuracy

Fig. 3. Patterns of Performance Degradation for Strong Learners

Onoda, Ratsch and Muller [7] conduct a theoretical analysis that confirms the conclusion drawn from our experimental results. The analysis considers a vector $\underline{b}$, which is the unnormalized weighting of hypotheses $h_t(t = 1 \ldots T)$ on a particular sample. They consider an annealing parameter $|\underline{b}| = \sum_{t=1}^{T} b_t$ and observe that when ϵ_t of AdaBoost (Fig 1) takes a small value (which is the case when the learner is strong) b_t becomes large, contributing to a large $|\underline{b}|$. They analyze: "when the annealing parameter $|\underline{b}|$ takes a very big value, AdaBoost type learning become a hard competition case: only the patterns with smallest margins (hardest samples) will get high weights; other patterns are effectively neglected in the learning process". This theoretical analysis confirms the conclusion from our observations and leads us to conclude that too much emphasis placed on a few hard samples in the training dataset, and the neglect of important features of the other samples is the probable cause for performance degradation in strong learner boosting.

We note the counterproductive boosting results recorded by Freund and Schapire [3] on boosting C4.5 over 100 iterations for Soybean-small, House-votes-84, Votes1 and Hypothyroid datasets all have very high test performances; the training accuracy is unknown. However, unlike in all instances of boosting strong RBF classifiers, Freund and Schapire record a number of productive instances of boosting C4.5 where the generalization performance is high; the training accuracy is unknown. The (unknown) training accuracy is a strong factor in our measure of learner strength. However, this warrants further investigation.

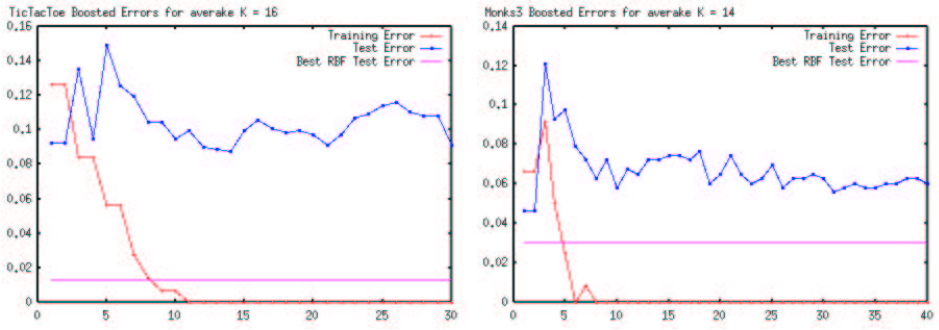


Fig. 4. Patterns of Performance Degradation III - Initially high degradation as the training error drops to zero, followed by a correction that is unable to compensate for the initial overfitting; when the base learner strength level is just approaching the strength levels of the strong learners showing Unproductive or Continuously Counter-productive boosting behaviors

4 Behavior of the Distribution

The distribution over the training samples was graphed for each iteration. Careful observation of the behavior of the behavior of the distribution gives an intuitive insight to the error reduction process.

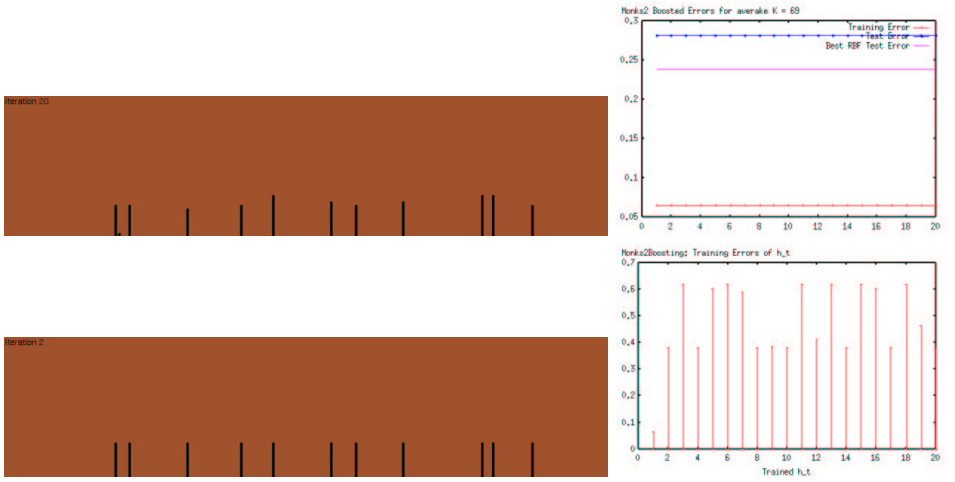


Fig. 5. Unproductive Boosting of Monks2 for K = 72: The top right graph shows the committee errors; the bottom right, the error of the hypotheses in the committee according to the distribution. The same few samples remain highlighted from iteration 1 to iteration 20. There is no error reduction in boosting and AdaBoost is unable to train a single good hypothesis which learns the hard samples

When the classifier is very weak it highlights a large number of samples as hard and the weighting is distributed over a large number of samples. The distinction between the hardness of these samples is small. There is a lot of activity in the distribution as the weightings change by small amounts over a large set of samples from iteration to iteration. Thus each iteration poses a large region in the feature space with little distinction among sample hardness, to be learned by a relatively weak classifier. This contributes to the slow error reduction process ridden with fluctuations.

When the classifier is a moderate performer fewer samples are highlighted in the hard distributions. This allows a classifier of moderate strength to focus on a moderate size region in the feature space at a time. When they are learned, their weighting comes down allowing another moderate set of samples to be highlighted. Their features are quickly learned by AdaBoost to achieve very fast and optimal error reduction.

As the classifier gets even less weak, a smaller set of somewhat harder samples is highlighted. It takes a number of attempts to learn their features during which they remain highly emphasized with little activity. As soon as these samples are

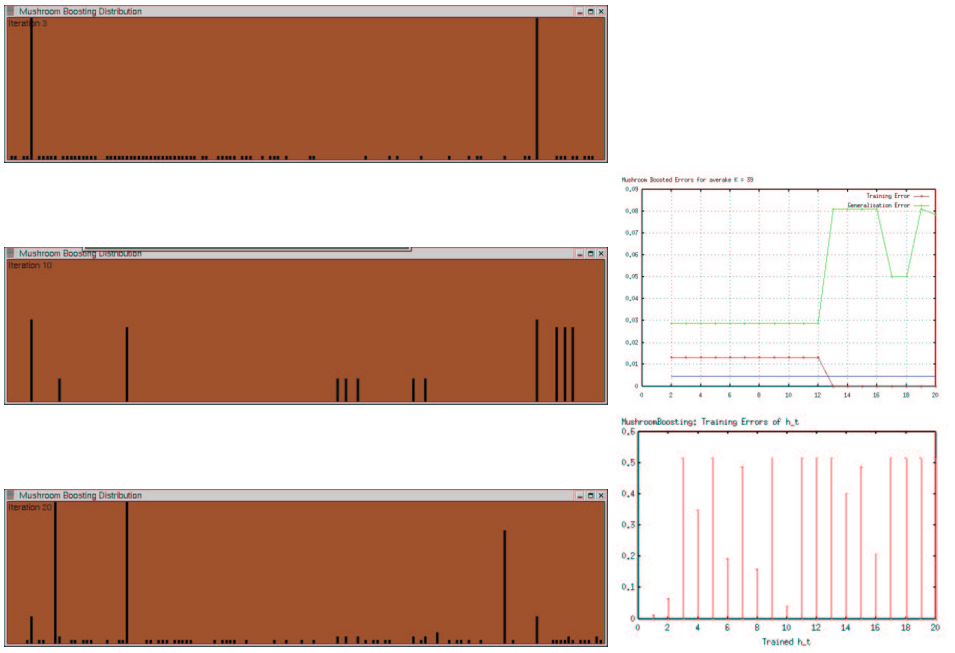


Fig. 6. Continuously counterproductive Boosting of Poisonous Mushroom for K = 39: The top right graph shows the committee errors; the bottom right, the error of the individual hypotheses in the committee according to the distribution. A very few very hard samples or outliers get repeatedly highlighted. AdaBoost is able to train good hypotheses that learn them; but they immediately rise in emphasis again and again. Specifically the two noted samples in distribution 2 are in constant emphasis. In iterations 9 and 19 AdaBoost trains very good hypotheses that learn them, but in iterations 10 and 20 they have soon begun to rise in emphasis again

learned (noted by a strong performing hypothesis) their weighting comes down suddenly allowing a different small set to be highlighted; this activity generally coincides with a "step reduction" of the errors.

When the classifier is somewhat stronger (but has not yet reached the strength levels of classifiers discussed in section 3) a few "significantly hard" samples are very strongly highlighted by the distribution initially. In its attempt to learn the features of these few samples particular to the training data, AdaBoost seems forced to ignore the less hard but important low-highlighted samples (the weighting on the few very hard samples keep increasing at the cost of the others) and overfits the training dataset as it does so. Once they are learned, a correction process immediately follows, where AdaBoost relearns the important features of the other hard samples with the right emphasis (their weighting is allowed to rise), thereby reducing the generalization error somewhat. However, the subsequent reduction is often unable to compensate for the initial overfitting. This contributes to a different form of performance degradation to the instances discussed in section 3, when the classifier strength is just below the strength levels of those discussed in 3.

The behavior of the distribution when the underlying classifier is very strong is analyzed in section 3.

5 Boosting Performance Increases, Peaks and then Falls as the Strength of the Weak Learner Increases

Boosting performance graphs are taken for committees with decreasing levels of learner weakness for Monks1, Monks2 and Credit datasets. The decrease in weakness from one committee to the next is achieved by appropriately increasing the number of kernels K in the underlying classifier.

When the classifier is significantly weak the boosting process is slow and ridden with fluctuations over both errors. When the weakness of the classifier decreases to a certain level, AdaBoost achieves very fast and maximum error reduction. In accordance with the theoretical predictions [10] the training error drops to zero very fast. In accordance with the theoretical predictions [10] the generalization error reduces with no clear sign of overfitting long after the training error has dropped to zero.

AdaBoost is able to also boost the performance of a somewhat strong learner. A clear distinction of such a boosting instance is the "step reduction process" where the errors remain stable for a number of iterations and suddenly reduce by a significant amount.

AdaBoost's ability to boost the generalization performance lessens as the RBF classifiers get stronger beyond a particular level. When the classifier's strength increases further, it overfits the training data and increases the generalization error by a significant amount as the training error rapidly drops to zero. After the training error has dropped to zero the generalization error resumes the reduction process. However, in many instances the subsequent reduction is unable to compensate for the initial overfitting. This shows a third pattern of

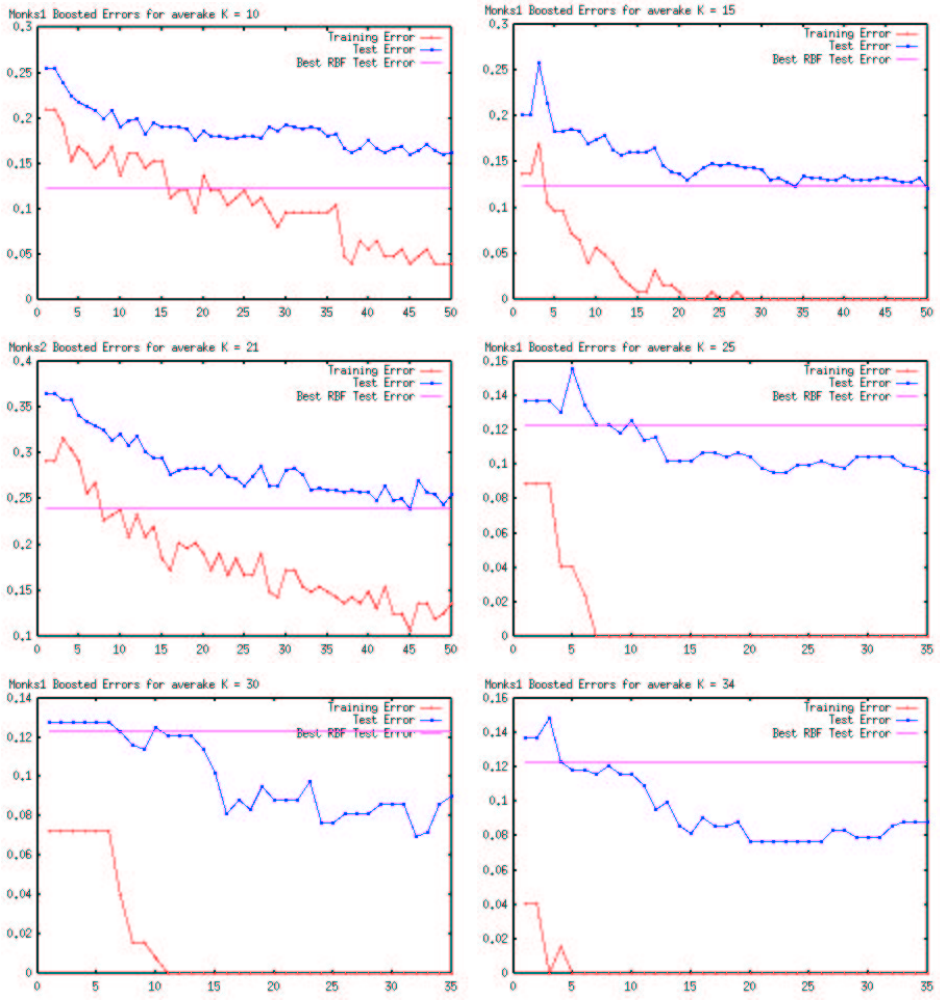


Fig. 7. Monks1 boosting for K=10,15,21,25,30,34. Error reduction is slow and full of fluctuations for very weak learner boosting. However, boosting achieves very fast and optimal error reduction as the classifier weakness decreases and achieves an optimal level

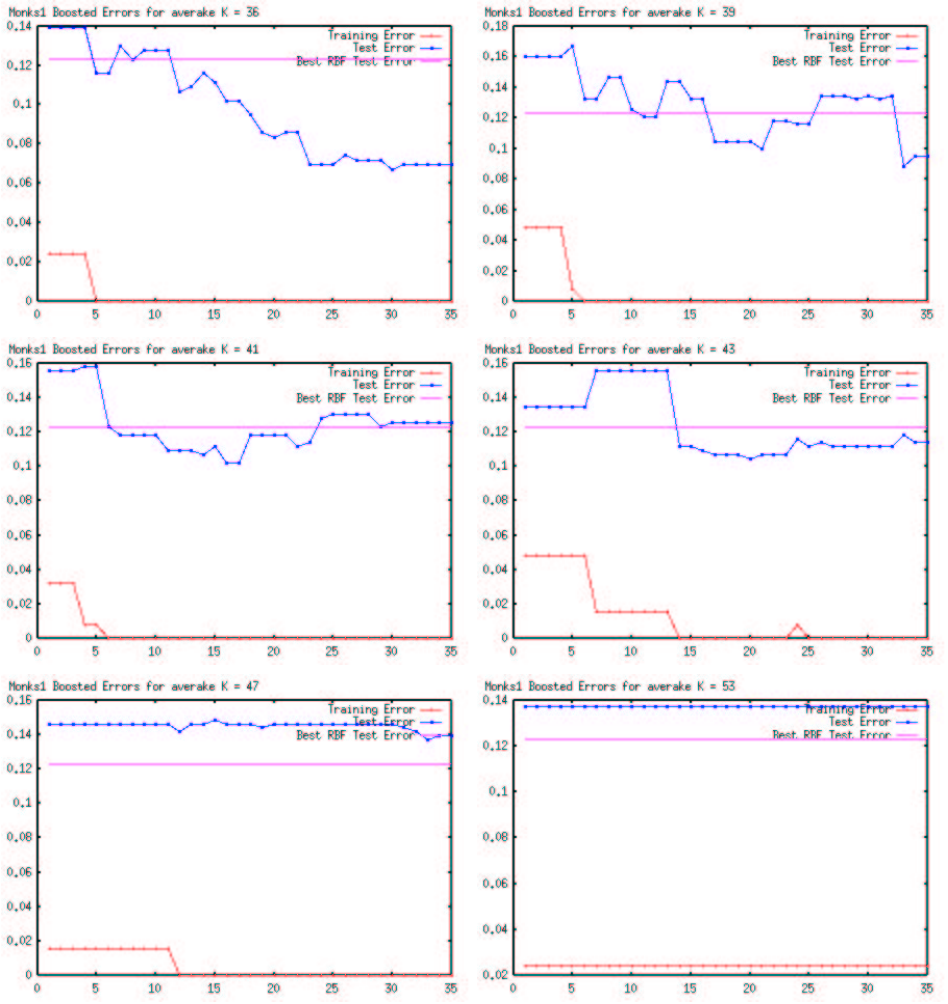


Fig. 8. Monks1 boosting for K=36,39,41,43,47,53. Overfitting effects shown as the training error drops to zero and "step reduction" in errors shown as the weakness decreases further. Boosting becomes unproductive as the classifier becomes a Strong Learner

performance degradation when the classifier is somewhat less strong than the learners described in section 3. The performance degradation graph recorded by Quinlan [9] in boosting C4.5 on the Colic dataset over 50 iterations also demonstrates this behavior. The test error rises sharply as the training error rapidly drops to zero; then the test error reduction is resumed at a slow pace, but is unable to compensate for the initial rise. The test performance of the base C4.5 classifier is reported as 85.08%; the training accuracy is unknown.

When the strength of the classifier is very high AdaBoost is unable to effect either error or there is continued overfitting even after the training error has dropped to zero, as analyzed in detail in section 3.

AdaBoost is hence optimally productive when the underlying classifier is moderately weak. It is slow and has low productivity when the learner is significantly weak. It is generally unproductive or counterproductive in boosting the performance of an already strong learner. If it is not possible to weaken a strong classifier to an appropriate level, it is better to train a good individual classifier by optimizing on parameter and model selection.

The AdaBoost annealing process is graphed for Monks1 for RBF classifiers with increasing strength in Fig. 7 and Fig. 8. The annealing process follows a very similar pattern for datasets Monks2 and Credit Scoring, on which RBF classifiers of strengths varying from very weak to strong (with modest supporting test performances) can be achieved.

6 Error Falls are Generally Caused by Notably Well Performing Hypotheses

In this section, we look at the individual hypotheses of a committee. Each hypothesis is trained according to a hard distribution of an iteration. Its error on the original training distribution is graphed.

Except in the situation of continuous overfitting where chaotic behavior can be observed, a reduction of either error coincides with or immediately follows a notably well performing hypothesis. Such a hypothesis is generally shown by a low (approximately less than 25%) error on the training samples usually preceded by hypotheses with high (about or more than 40%) error on the training samples. This indicates that effective error reduction occurs when there are a number of weak performers that highlights a set of hard samples followed by a good performer that effectively learns the hard sample space highlighted. This supports that there is an optimal weakness of the underlying learner that offers a good trade-off: the ability to produce weak hypotheses that highlight a representative region in the sample space, and the ability to generate a strong hypothesis that effectively learns the highlighted sample space. The ability to generate a strong hypothesis for learning the hard regions is closely related to the generalization ability of the learner. Hence, classifier strength must also be a function of the generalization error.

In instances that boost well, a fair proportion of the hypotheses have less than error as well as a number of notably high performers. There is generally

an abundance of low performers (with error close to or above) and a dearth of notably strong performers in instances that do not boost.

7 Boosting an Appropriately Weakened Classifier Can Improve on the Performance of the Same Classifier Strengthened by Parameter or Model Selection

The strong classifier is weakened prior to boosting. In our definition of learner strength, and our experiments, the capacity of all the classifiers in a particular committee is fixed. Hence the weakening is achieved by varying a free parameter or the kernel (thereby changing this capacity), so that the learner shows a higher training error and a higher test error with respect to the same dataset. Weakening in our experiments is achieved by decreasing the number of kernels. The boosting performance of the weakened classifier is graphed together with the peak performance achievable by the classifier individually(straight line in graph). It is clear from the graphs of Fig. 7 and Fig. 8 that the boosted performance is able to improve on the performance of the improved classifier for weak learners of Monks1. The weaker learner similarly improves on the peak performance for Monks2 and achieves performance close to the peak performance for Credit Scoring. A vast amount of published literature, in particular [3] and [9], report further results and analysis of significant performance improvements achieved by the boosted committee over the individual classifier when the learner is not strong.

The weakened learner causes AdaBoost to concentrate on the samples modelling the more representative regions in the feature space, and reduces the excessive emphasis on the few boundary samples. Hence, the probability of performance degradation is decreased, and the committee is often able to learn more regions in the feature space than an individual learner.

8 Conclusion

We have demonstrated that boosting an already strong learner is generally counterproductive, and that the boosting performance increases, peaks and then falls as the strength of the underlying weak learner increases. The learner strength is a function of both training and generalization abilities of the weak learner. We have identified three patterns of performance degradation depending on three different identified strength levels of the underlying learner:

1. Unproductive boosting behavior, when the base learner has high training accuracy supported by a modest test accuracy.
2. Continuously counterproductive boosting behavior when the base learner has high training accuracy supported by a high test accuracy.
3. Initially highly counterproductive boosting behavior followed by a slightly productive phase that is unable to compensate for the initial overfitting, when the base learner is just approaching the strength levels of instances discussed in 1 and 2.

We have highlighted patterns of behavior in the distribution and have argued that performance degradation in boosting is due to:

1. An underlying strong learner
2. AdaBoost's characteristic of forcing the learner to concentrate on the very hard samples and outliers with too much emphasis when the earner is strong.

However, boosting a learner of appropriately low strength achieves good error reduction and, in some instances, improves on the peak performance the strengthened learner is capable of achieving by parameter or model selection. We have proposed therefore that, when the underlying learner approaches the Strong Learner strength levels identified, it is possible to avoid performance degradation and to achieve higher boosting productivity by weakening the learner. It is possible to weaken a kernel machine that is a Strong Learner by using a weaker kernel, decreasing the number of kernels or varying a free parameter (such as ε in SVMR, or the regularization parameter) as applicable, so that the learner has a higher training error and a higher test error with respect to the given dataset.

The proposed solution works by eliminating the first identified cause of performance degradation. Future work would address regularizing AdaBoost so that it curbs the excessive emphasis it places on the few extremely hard samples and outliers when the underlying learner is strong.

References

1. Bishop C.M. (1995): *Neural Networks for Pattern Recognition.* Oxford University Press.
2. Freund Y., Iyer R., Schapire R.E. & Singer Y.(1998): An Efficient Boosting Algorithm for Combining Preferences. *Proc. of the Fifteenth International Conference on Machine Learning,* pp. 332-341.
3. Freund Y. & Schapire R.E. (1999): A Brief Introduction to Boosting. *Proceedings of the Sixteenth International Joint Conference on Artificial intelligence.*
4. Freund Y. & Schapire R.E. (1998): Discussions of the paper "Arching Classifiers" by Leo Brieman. *The Annals of Statistics,* 26(3):824-832.
5. Freund Y. & Schapire R.E. (1996): Experiments with a New Boosting Algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning,* pp.148-156.
6. Kearns M.J. & Vazirani U.V. (1994): *An Introduction to Computational Learning Theory.* The MIT Press.
7. Onoda T., Ratsch G. & Muller K. (1998): An Asymptotic Analysis of AdaBoost in the Binary Classification Case. In L. Niklasson, M. Bodn, and T. Ziemke, editors, *Proc. of the Int. Conf. on Artificial Neural Networks (ICANN'98),* pp. 195-200.
8. Press W.H., Teukolsky S.A., Vetterling W.T. & Flannery B.P (1992): *Numerical Recipes in C: The Art of Scientific Computing. (Second edition):* Singular Value Decomposition. Numerical Recipes On-Line Software Store.
9. Quinlan, J. (1996): Bagging, Boosting and C4.5. *Proc. of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference,* pp. 725-730.

10. Schapire R. & Robert E. (1999): Theoretical Views of Boosting and Applications. *Tenth International Conference on Algorithmic Learning Theory.*