From *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04), July 25-29, 2004, San Jose, California, MIT PRESS, Cambridge*

Online Parallel Boosting

Jesse A. Reichler[1], Harlan D. Harris[2], and Michael A. Savchenko[3]

[1]Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA, reichler@uiuc.edu
[2]Department of Psychology, Columbia University, New York, NY 10027, USA, harlan@psych.columbia.edu
[3]Department of Aeronautical and Astronautical Engineering, University of Illinois, Urbana, IL 61801, USA, savchenk@uiuc.edu

Abstract

This paper presents a new boosting (arcing) algorithm called POCA, Parallel Online Continuous Arcing. Unlike traditional boosting algorithms (such as Arc-x4 and Adaboost), that construct ensembles by adding and training weak learners sequentially on a round-by-round basis, training in POCA is performed over an entire ensemble continuously and in parallel. Since members of the ensemble are not frozen after an initial learning period (as in traditional boosting) POCA is able to adapt rapidly to non-stationary environments, and because POCA does not require the explicit scoring of a fixed exemplar set, it can perform online learning of non-repeating data. We present results from experiments conducted using neural network experts that show POCA is typically faster and more adaptive than existing boosting algorithms. Results presented for the UCI letter dataset are, to our knowledge, the best published scores to date.

Introduction

Boosting (also known as arcing) is an ensemble learning strategy that works by iteratively constructing and combining experts that are increasingly forced to concentrate on "difficult" training exemplars.

Traditionally, boosting proceeds in a series of sequential rounds. During each round, a new expert is trained and added to the ensemble. Training data for all experts is drawn from a single fixed data set with weights on each exemplar. After each round, exemplars are re-weighted to increase the importance of those exemplars that have been the most difficult for prior experts to learn. Ensemble output is a weighted combination of all expert outputs.

It has been shown that efficient boosting algorithms exist for constructing arbitrarily accurate (on training data) ensembles from individual learners which perform no better than chance (Schapire 1999). One such algorithm, Adaboost, has received a great deal of attention because of its efficiency and unexpectedly good generalization properties (Freund and Schapire 1998). Recently, Breiman introduced Arc-x4 (Breiman 1997), a similar but simpler algorithm, that assigns fixed, equal voting weights for each expert, and re-weights exemplars using a simpler function of the number of mistakes made by prior experts. Breiman (Breiman 1999) has demonstrated that Arc-x4 usually performs as well as Adaboost, and has argued that the success of boosting algorithms is due to the general process of increasing weights on difficult exemplars, rather than to any specific re-weighting or recombination scheme. Pseudo-code for Arc-x4 is shown in Table 1-A.

Current boosting algorithms require that explicit performance scores for each exemplar be calculated after each training round. This can require large memory resources and is unsuitable for use with changing concepts or in environments that lack a fixed, finite training set.

This paper introduces a new multi-class boosting algorithm, POCA (Parallel Online Continuous Arcing), that does not require the use of a fixed, finite training set, and is thus capable of learning in online environments with non-repeating data. Unlike traditional boosting, where prior experts are frozen as new experts are added, POCA continuously trains an entire ensemble in parallel, allowing ensembles to adapt rapidly to non-stationary environments.

Results from multi-class classification tests show that even with a fixed data set POCA performs competitively with existing boosting algorithms and learns faster in terms of improvement per training iteration. Results on the UCI letter dataset are, to our knowledge, the best published scores to date.

Other work on parallel boosting includes parallel boosting of binary concepts using decision trees (Fern and Givan 2003), and boosting with disjoint distributed data sets (Fan et al. 1999; Lazarevic and Obradovic 2001; Chawla et al. 2002).

POCA: Parallel Online Continuous Arcing

The basic idea behind the POCA algorithm is straightforward: Each exemplar is received from the environment and delivered to an entire ensemble of experts in parallel. Every expert is trained on every exemplar as it is delivered, and experts dynamically modulate each other's learning rates in a manner that approximates traditional boosting schemes for re-weighting training data. In essence, experts form a virtual chain (Figure 1), such that the learning rate of an expert is a function of the performance, on that exemplar only, of the experts preceding it in the chain. After an exemplar is used for training, it is discarded from memory. Pseudo-code for POCA is shown in Table 1-B.

Copyright © 2004, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

A Virtual Chain

POCA was designed to approximate the behavior of standard boosting algorithms, but in a parallelizable manner. Here we provide a brief inductive proof outline for the assertion that POCA approximates traditional boosting. Consider the case of learning a fixed set of training data. The first (leftmost) expert in POCA is trained on the un-weighted data, exactly as in traditional boosting. That expert will converge to the same solution as would the initial expert in Adaboost or Arc-x4, except that its training never ends. After the first expert has converged, it sends to the rightward experts an error signal that is weighted in the same way that exemplar mistakes are weighted in standard boosting. Therefore, as experts sequentially converge from left to right, POCA ensembles come to mirror traditional boosting ensembles. While this equivalence holds in the limit, we will show that POCA benefits from not waiting until experts converge before initiating the training of subsequent experts.

Choice of Experts and Weighting Functions

Because the POCA algorithm must operate in non-stationary environments, it requires experts that are capable of gradual and continuous non-monotonic learning. Here we have used backpropagation and small multi-layer neural network experts, where the learning rate of each expert is simply a baseline rate multiplied by the current output of the expert's paired weighting node. In multi-class classification, weighting nodes compute vector weightings by treating each output class as a separate classification task (Schapire 1999), and we use a slightly modified form of backpropagation described in a subsequent section.

For experts that cannot accept explicit learning rates, a variant of POCA analogous to boosting-by-resampling (Freund and Schapire 1999) can be used: The value computed by each weighting node (scaled from 0 to 1) is used as the probability of training the paired expert on each exemplar, and each expert makes an independent and probabilistic decision about whether to train on each exemplar as it is delivered from the environment.

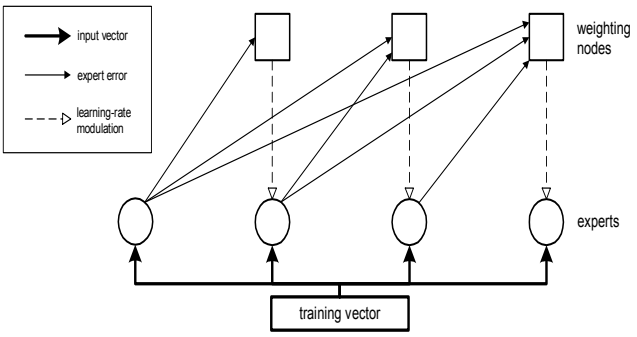

Figure 1: The POCA architecture. Each expert processes the input in parallel and computes its own prediction error. This error is forwarded in parallel to all rightward weighting nodes. Each weighting node computes some function over the incoming errors and projects this value down to its paired expert in order to modulate the learning rate for the current exemplar.

Table 1-A. Arc-x4 Algorithm

Given: Training dataset $\{<x_1,y_1>,\ldots,<x_n,y_n>\}$

Maximum # of rounds, R, Expert (weak) learning algorithm

Initialize weights and mistakes:

$w_1(i) = 1/n \quad \forall i \qquad m_0(i) = 0 \quad \forall i$

For r=1 to R:

Build new expert h with weighted dataset.

Calc. weighted error of current expert:

$e_r(i) = [\![1 \ \ if \ \ h_r(x_i) \neq y_i \ \ else \ \ 0]\!] \quad \forall i$

$\varepsilon_r = \sum_i w_r(i) e_r(i)$

If weighted error is greater than chance:

Reset weights, delete expert, loop

Update exemplar mistakes, weights:

$m_r(i) = m_{r-1}(i) + e_r(i) \quad \forall i$

$w_{r+1}(i) = 1 + m_r(i)^4 \quad \forall i$

Normalize exemplar weights:

$w_{r+1}(i) = w_{r+1}(i) / \sum_i w_{r+1}(i) \quad \forall i$

End for loop.

Ensemble output: $h(x) = \arg\max_{y \in Y} \left\| \{h_r : h_r(x) = y\} \right\|$

Table 1-B. POCA Algorithm

Given: Expert learning algorithm,

# of experts, R; Decay, d

Build experts $h_1,\ldots,h_R$

Initialize weighted error approximations:

$\varepsilon(r) = q(r) = 0.5 \quad \forall r \qquad s(r) = 1 \quad \forall r$

While there is an available training exemplar <x,y>:

Calc. expert errors on <x,y>: $e(r) = [\![1 \ \ if \ \ h_r(x) \neq y \ else \ 0]\!] \quad \forall r$

Calc. expert weights for <x,y>:

$m(r) = \sum_{\{i : i<r \ \wedge \ \varepsilon(i)<0.5\}} e(i) \quad \forall r$

$w_t(r) = 1 + m(r)^4 \quad \forall r$

$z(r) \approx \max(w_{t-windowsize}(r), \ldots, w_t(r))$

Train experts on <x,y> using $w_t(r)/z(r)$

Update norm. weighted error estimates of experts:

$q(r) = q(r)(1-d) + w_t(r)e(r)d \quad \forall r$

$s(r) = s(r)(1-d) + w_t(r)d \quad \forall r$

$\varepsilon(r) = q(r) / s(r) \quad \forall r$

End while loop.

Ensemble output: $h(x) = \arg\max_{y \in Y} \left\| \{h_r : h_r(x) = y\} \right\|$

Table 1: Pseudo-code for (two-class) Arc-x4 and POCA. Indices in table 1-A index over exemplars in the fixed training set; indices in table 1-B index over experts in the ensemble. See text for discussion of decay, windowsize, and normalization constants in POCA.

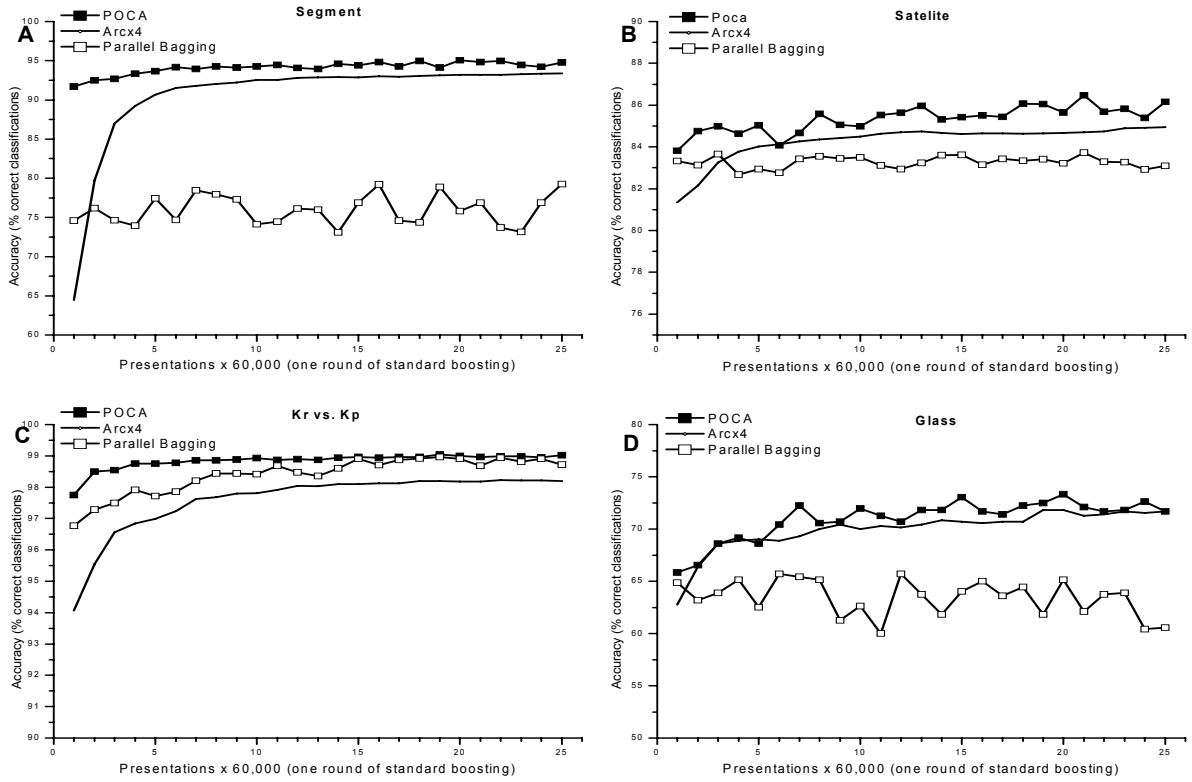

Figure 2: Does POCA boost? Performance on some UCI datasets using simple experts (standard deviations are omitted for clarity, but are generally within one percentage point). Note that significantly better results can easily be obtained on these datasets with more powerful experts; simple experts were chosen to examine the effect of boosting.

In addition to selecting a particular architecture for the experts, it is necessary to choose a particular function to be computed by the weighting nodes. We have implemented parallelizable (Fern and Givan 2003) weighting functions that emulate both the Arc-x4 and Adaboost weighting schemes, and found only minor differences in performance. All experiments in this paper were conducted using the Arc-x4 version of POCA.

Dynamic Normalization of Weighting Nodes

Traditional boosting algorithms employ an explicit normalization step after each round to ensure that exemplar weights form a valid distribution for sampling. This is important because, depending on the performance of prior experts, the highest weights that a given expert receives may be very small. If left un-normalized, excessively small weights could significantly slow down learning. The difficulty for POCA is that it is impossible to systematically normalize weighting node outputs (i.e. learning rates or training probabilities) between rounds, not only because POCA does not work in discrete rounds, but because we assume an online environment with possibly non-repeating, non-stationary training exemplars.

We employ a heuristic mechanism to approximate the normalization of weighting node outputs: Each expert keeps track of the highest weight it has seen in the recent past, and this value is used to normalize new weights as they are generated. The heuristic ensures that the relative weighting of exemplars is largely preserved, while each expert rescales its weights so that the highest weighted exemplars receive weights (probabilities or learning rates) near 1. The optimal window size for normalization depends on learning rates and concept drift rate, but experimental results show little difference over a wide range of window sizes; experiments presented in this paper normalize over the last 2000 exemplars. Another technique employed in boosting algorithms (Breiman 1999) is to exclude experts that perform worse than chance. We emulate this by tracking each expert's "decaying-weighted-error" over time, which can be used continuously and is suitable for non-stationary concepts. Experiments presented here use a decay of .001.

Differentially Weighted Backpropagation

When training multi-class neural networks, one commonly presents a vector where the correct class has a value near 1,

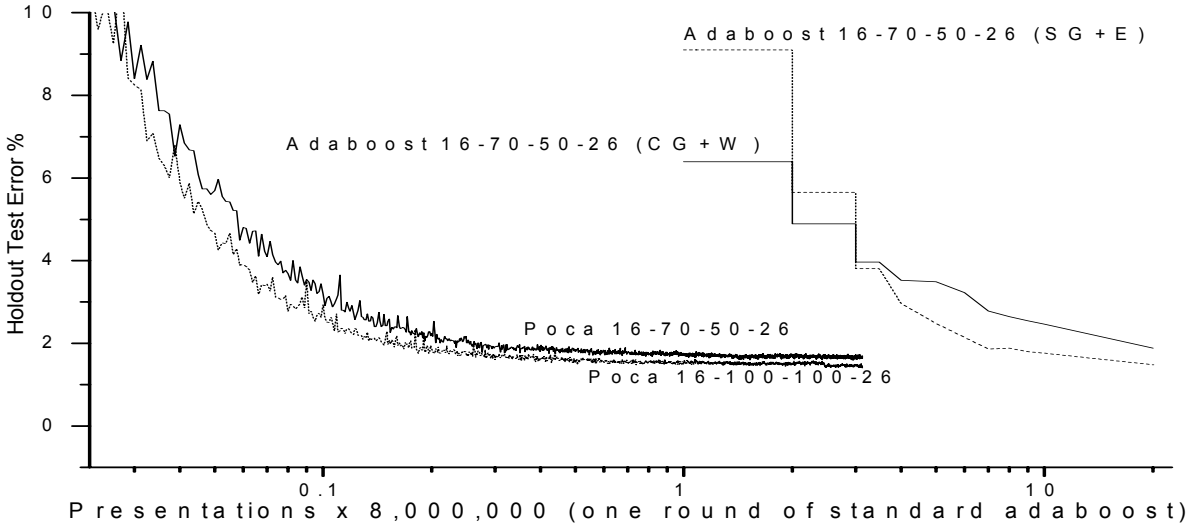

Figure 3: Comparison with previously published (state-of-the-art) results on the UCI Letter dataset by Schwenk et al. Note that test scores in Adaboost are calculated after the completion of each discrete round (which is accompanied by the addition of a new expert to the ensemble), and therefore begin only after completion of the first round. For clarity, standard deviations are not shown, but are consistently below 0.1 during the second half of POCA training. SG+E is stochastic-gradient descent training using weighted resampling; CG+W is conjugate gradient descent training using weighted training exemplars; results taken from (Schwenk and Bengio 2000).

and other classes have a value near 0. Prediction error is the mean-squared-error over the entire output vector. When boosting such multi-class neural network experts, the common practice is to calculate a scalar weight for each training exemplar and use this scalar weight to either modify the learning rate of the neural network or to weight the exemplars for resampling (Drucker et al. 1994; Drucker 1999; Optiz and Maclin 1999; Schwenk and Bengio 2000).

We have obtained slightly better results using a modification of backpropagation that differentially weights individual components of the error vector, rather than modifying the global learning rate of the network. We simply multiply the backpropagation error term on each output node by the associated (normalized) weight of that component, as calculated using the applicable weighting rule (i.e. Arc-x4 variant or Adaboost variant), treating each component of the output as an independent binary learning task, along the lines of Adaboost.M2 (Schapire 1999). This forces the neural networks to concentrate learning on hard-to-distinguish output classes (specific output components). Results using standard backpropagation (not shown) are not qualitatively different, but tend to converge slightly slower.

Results

This section presents some representative results comparing POCA to other ensemble methods using datasets from the UCI machine learning repository (Blake and Merz 1998).

Does POCA Boost?

The first set of experiments compares POCA to Arc-x4 and Parallel Bagging, using very simple experts. Experts are standard MLP (multi-layer feed-forward perceptron) neural networks with a single hidden layer of only 3 nodes; training was performed using a slightly modified version of stochastic backpropagation that multiplies output component errors by the component output of the weighting nodes, emulating Schapire's Adaboost.M2 (Schapire 1999) training scheme for multi-class boosting. The baseline learning rate was 0.4, with a momentum of 0.2; each run consists of 25 "rounds" with 60,000 training presentations per round. Note that in POCA and parallel bagging, rounds demarcate chunks of training exemplars seen, and have no greater significance; all algorithms receive the same number of total training exemplars.

Figure 2 shows that POCA performs comparably to standard Arc-x4, and is not simply averaging the votes of independent experts, as in bagging. Plots show accuracy over time, on held-out test data, averaged over 10 runs. Results of Adaboost and POCA-adaboost were similar and are not shown.

It is important to note that significantly better results can be achieved on these UCI datasets with more complex experts; for this set of experiments we specifically chose simple experts as base learners in order to tax the abilities of individual experts and test the ability of POCA to boost. As has been reported elsewhere, with more complex individual experts, the advantages of boosting over bagging become less pronounced (Optiz and Maclin 1999).

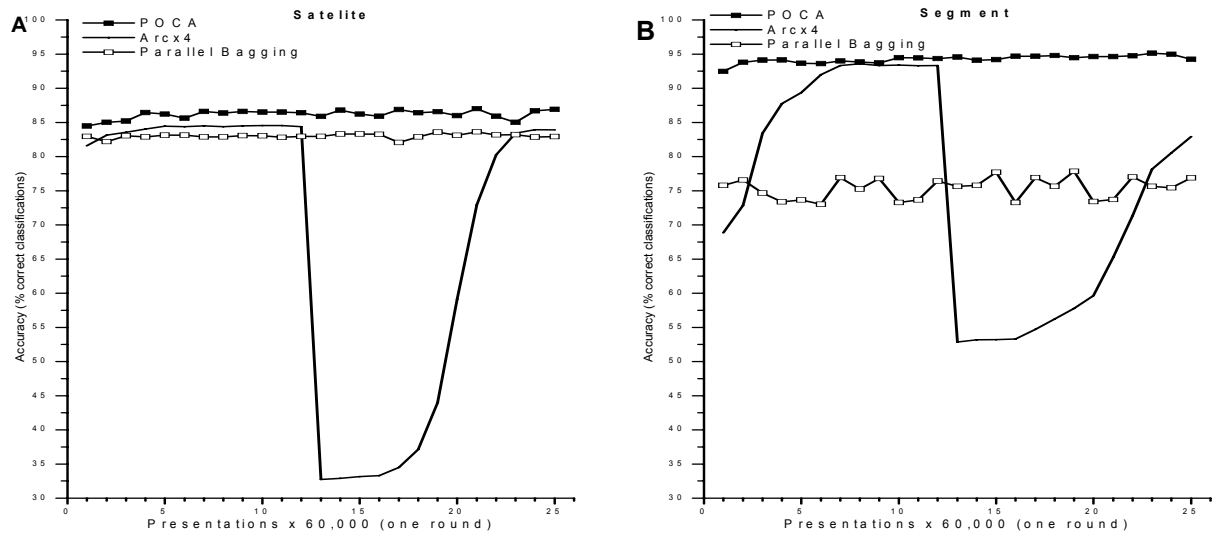

Figure 4: Performance on datasets where the concept changes during training. In traditional boosting, only over time do new experts "drown-out" the old incorrect ones.

Comparison with State-of-the-art

Recently, Schwenk and Bengio (Schwenk and Bengio 2000) presented state-of-the-art results on the large (20,000 cases) and difficult UCI Letter dataset using boosted neural networks. Schwenk and Bengio's results plateau at 1.47% error after a total of 200 million training-exemplar presentations (25 rounds of 500 epochs each, with 16,000 training exemplars per epoch), which they cite as the best results obtained to date on this data.

Figure 3 shows a comparison of POCA to the Schwenk et al. Adaboost results, using identically sized ensembles and experts (25 neural network experts of size 16-70-50-26) , and also using larger ensembles (50 experts) of larger neural networks (16-100-100-26). Other parameters were identical to the previous experiment. To reduce training time, results for POCA have been averaged over only 3 runs and we stopped training after 25 million exemplars were presented, when learning seemed to plateau.

With neural network experts of identical size to those used in Schwenk et al., POCA plateaus at a slightly higher error of 1.6%, but reaches this much more rapidly. With larger networks, POCA plateaus at an error rate of 1.44%, which is the best published score on this dataset to our knowledge.

We believe that the significance of the results shown in figure 3 lies not so much in the final error rates, but rather in the speed of the learning (in terms of improvement per iteration and potential running time on parallel hardware), and in the flexibility of the underlying algorithm (i.e. suitability for online learning of non-repeating data).

We further note that Schwenk et al. were unable to obtain satisfactory results when using stochastic gradient descent training with a directly-weighted cost function (not shown), and had to use weighted sampling or conjugate-gradient descent, both of which greatly increase computational processing requirements. POCA is able to achieve identical results using a computationally cheaper, online-suitable gradient descent procedure with a directly-weighted cost function (differentially weighted backprop).

Suitability for Changing Concepts

A major motivation for the development of the POCA algorithm was the possibility of applying boosting principles to online learning in non-stationary environments. Standard boosting algorithms are ill-suited for such tasks both because they require a fixed dataset over which to track explicit performance scores, and because traditional boosting algorithms build ensembles sequentially, so that previously built experts are incapable of adapting to changes in a target concept.

Figure 4 compares the performance of POCA against the standard Arc-x4 algorithm and Parallel Bagging on two UCI Datasets, Satellite and Segment. After the 12th round we change half of the target classes, for both training and testing data, effectively changing the target concept. Arc-x4 exhibits a catastrophic failure and can only recover slowly by building enough new experts to drown out the newly-incorrect votes of previously constructed experts.

Conclusions

In a recent paper, Breiman (Breiman 1999) states that "in terms of handling large databases, an advantage for bagging, randomized construction, and output randomization is that they can be easily parallelized, while boosting algorithms are essentially sequential." Drucker (Drucker 1999) similarly states that "The big advantage of

bagging over boosting is that each machine can be trained independently, hence in parallel." And in (Bauer and Kohavi 1999), Bauer and Kohavi state that "In parallel environments, Bagging has a strong advantage because the sub-classifiers can be built in parallel" and go on to question whether an efficient parallel implementation of boosting/arcing is possible.

We believe that POCA represents a viable, efficient, parallelizable boosting algorithm. It requires no memory for storing exemplars or exemplar scores and can be used for online learning of difficult multi-class concepts in non-stationary environments.

Although the POCA algorithm itself is clearly parallel, one might ask whether the actual adaptation of experts is still in some sense inherently serial. Empirically, the parallel training procedure yields consistently better results over its serial counterpart. We speculate that this is due to the fact that, even though a given expert cannot be expected to completely stabilize until those preceding it in the virtual chain completely stabilize, all experts in POCA are learning gradually, as if tracking a drifting distribution. When training begins, learning in the ensemble resembles a form of parallel bagging, but as experts learn to specialize on certain regions of the input space, they perturb the distribution of weights seen by those experts that follow them in the virtual chain, forcing those experts to adapt to the changing distribution as if the target concept had changed. Because of this, POCA typically learns faster (in improvement per training iteration) than traditional boosting algorithms like Arc-x4 and Adaboost, as can be seen in Figures 2, 3, and 4.

In traditional boosting, at the start of each round a new expert is created, given the full data set, and "trained to completion." Because neural network experts (and similar structures) can get stuck in local minima for prolonged periods, traditional boosting algorithms must adopt criteria for deciding when to stop training a given expert and move on to the next round. Stopping prematurely may result in an unnecessarily large number of rounds, while overtraining experts wastes computational time. The continuous training of experts in POCA ensures that no time is wasted overtraining experts, while still providing each expert with the entire training time in which to reach asymptotic performance. We believe that the higher overall accuracy of POCA seen in our results is due to the ability of POCA to avoid the under and over training of individual experts.

A drawback of the POCA algorithm is that on a single CPU it can run orders of magnitude slower in actual runtime than traditional boosting algorithms. We are currently working on a version of POCA that runs on parallel hardware, using a pipeline system to minimize communication bandwidth. We also hope to explore alternative weight-propagation topologies and learning schemes, which would allow POCA to generalize to parallel forms of other ensemble algorithms such as Multiboosting (Webb 2000) and Mixtures of Experts (Jacobs et al. 1991).

References

Bauer, E., and Kohavi, R. 1999. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning* 36(1-2): 105-39.

Blake, C. L. and Merz, C. J. 1998. "UCI Repository of machine learning databases." Web page. Available at http://www.ics.uci.edu/~mlearn/MLRepository.html.

Breiman, L. 1997. *Arcing the Edge*, Technical Report #486. University of California, Berkeley.

Breiman, L. 1999. Combining Predictors. *Combining Artificial Neural Nets.* ed. A. J. C. Sharkey, 31-50. London: Springer-Verlag.

Chawla, N. V, Hall, L. O, Bowyer, K. W, Moore, T. E, and Kegelmeyer, W. P. 2002. Distributed Pasting of Small Votes. *In Proceedings of Multiple Classifier Systems 2002*.

Drucker, H. 1999. Boosting Using Neural Networks. *Combining Artificial Neural Nets.* ed. A. J. C. Sharkey, 51-77. London: Springer-Verlag.

Drucker, H., Cortes, C., Jackel, L. D., Lecun, Y., and Vapnik, V. 1994. Boosting and Other Ensemble Methods. *Neural Computation* 6(6): 1289-301.

Fan, W., Stolfo, S. J., and Zhang, J. 1999. The Application of AdaBoost for Distributed, Scalable and On-line Learning. *In Proceedings of the fifth ACM SIGKDD International Conference on Knowledge discovery and Data Mining, KDD-99*, San Diego, California.

Fern, A., and Givan, R. 2003. Online Ensemble Learning: an Empirical Study. *Machine Learning* 53(1-2): 71-109.

Freund, Y., and Schapire, R. E. 1998. Arcing Classifiers - Discussion. *Annals of Statistics* 26(3): 824-32.

Freund, Y., and Schapire, R. E. 1999. A Short Introduction to Boosting. *Journal of the Japanese Society for Artificial Intelligence* 14(5): 771-80.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. 1991. Adaptive mixtures of local experts. *Neural Computation* 3(1): 79-87.

Lazarevic, A, and Obradovic, Z. 2001. The Distributed Boosting Algorithm. *In Proceedings of the International Conference on Knowledge Discovery and Data Mining, KDD-2001*.

Optiz, D., and Maclin, R. 1999. Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research* 11: 169-98.

Schapire, R. E. 1999. A Brief Introduction to Boosting. *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*.

Schwenk, H., and Bengio, Y. 2000. Boosting Neural Networks. *Neural Computation* 12(8): 1869-87.

Webb, G. I. 2000. Multiboosting: a Technique for Combining Boosting and Wagging. *Machine Learning* 40(2): 159-96.