Online Policy Adaptation for Ensemble Algorithms

Christos Dimitrakakis
dimitrak@idiap.ch

Samy Bengio
bengio@idiap.ch

IDIAP–RR 02-28

July 2002

IDIAP Research Report 02-28

Online Policy Adaptation for Ensemble Algorithms

Christos Dimitrakakis
dimitrak@idiap.ch

Samy Bengio
bengio@idiap.ch

July 2002

Abstract. Ensemble algorithms are general methods for improving the performance of a given learning algorithm. This is achieved by the combination of multiple base classifiers into an ensemble. In this paper, the idea of using an adaptive policy for training and combining the base classifiers is put forward. The effectiveness of this approach for online learning is demonstrated by experimental results.

1 Introduction

The problem of pattern classification has been attacked in the past using supervised learning methods. In this context, a set of N example patterns $S = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, is presented to the learning machine, which adapts its parameter vector so that when input vector x_i is presented to it, the machine outputs the corresponding vector y_i.

Let us denote the output of a learning machine for a particular vector x as $h(x)$. The classification error for that particular example can be designated as $\varepsilon_i = [h(x_i) \neq y_i]$. Thus, the classification error for the set of examples D_N that is used to train the machine's parameters can be summarized as the empirical error $\hat{L} = 1/N \sum_i \varepsilon_i$, where N is the number of instances in D_N.

In general, if D_N is a sufficiently large representative sample, taken from a distribution D, then the generalization error $L = \int p_D(x)\varepsilon(x)$, would be close to $\hat{L}$. In practice the training set provides limited sampling of the distribution D, leading to problems with overfitting. Adding the effect of classifier's inherent bias and variance, ultimately it will be true that $L > \hat{L}$.

The generalization error cannot be directly observed, so it has been common in the past to use a part of the training data for validation, in order for it to be estimated. This has led to the development of techniques mainly aimed at reducing overfitting caused by limited sampling, such as early stopping and K-fold cross-validation.

A possible solution to the bias and variance problem is offered by ensemble methods, such as the mixtures of experts architecture [7], bagging[5]and boosting [6]. In particular, AdaBoost, has been shown to significantly outperform other ensemble techniques, as shown empirically for example in [10]; however, theoretical results regarding its apparent effectiveness have at best been tentative.

Of these, the most widely accepted one holds that the effectiveness of AdaBoost in reducing the actual errors relates to a quantity called the *margin of classification.*

The margin distribution for the general two class case can be defined as $\text{margin}_f(x, y) = yf(x)$, where $x \in \mathcal{X}$, $y \in \{-1, 1\}$ and $f : \mathcal{X} \rightarrow [-1, 1]$. In general, the hypothesis $h(x)$ can be derived from $f(x)$ by setting $h(x) = sign(f(x))$. . In this case, $|f(x)|$ can be interpreted as the confidence in the label prediction. For the multi-class case, we assume that label with the highest confidence is predicted, i.e. $y = \text{argmax}_y f_y(x)$ and thus:

$$\text{margin}_f(x) = f_y(x) - \max_{y \neq y'} f_{y'}(x)$$

In effect, it is argued that AdaBoost is indirectly maximizing this margin - something that should lead to more robust performance. However its importance as a measure of generalization ability is not certain. There exist counterexamples for which the minimum margin is not an adequate predictor of generalization[4]. Empirically, however, attempts to apply algorithms that directly maximize the margin have been met with success. [1]

During the past few years, there has been a resurgence in the development of new ensemble machine learning techniques, sometimes combining aspects of older, well known techniques, as in [1]. Other researchers attempt to take advantage of theoretical results in generalization error bounds related to the minimum margin, such as the work reported in [9] and [8].

1.1 Adaptive Policies for Mixing

In this work, the possibility of using an adaptive, rather than a fixed, policy for training and combining base classifiers is investigated. The field of reinforcement learning [11] provides natural candidates for use in adaptive policies. In particular, the policy is adapted using Q-learning[12], a method that improves a policy through the iterative approximation of an evaluation function Q.

The framework within which Q-learning will be employed is outlined below.

[1]Perhaps it is useful to note here, that, although Support Vector Machines are designed to maximize such a margin, the margin in that case is defined as a distance to a hyperplane in a kernel space, rather than the input space itself.

2 General Architecture

The OLAP (OnLine Adaptive Policy) classifier ensemble consists of a set of base classifiers, or experts, $\mathcal{E} = \{e_1, e_2, ..., e_n\}$ and a set of predictors $\mathcal{P} = \{p_1, p_2, ..., p_n\}$, one for each base classifier.

Choosing which experts to train on which examples, and how to derive an ensemble decision from the hypothesis of the individual experts can be formulated as a control problem. In this case we wish to maximize the discounted future return of the system at time t:

$$R^t = \sum_{k=0}^{\inf} \gamma^k r^{t+k+1}$$

where $\gamma \in [0, 1)$ is a *discount factor* and r^t is the *reward* received at time step t. For classification tasks, r^t is chosen to be -1 for an incorrect decision and 0 for a correct classification of the current example.

At each time step t, an example $x \in \mathcal{X}$ is presented to the ensemble. The algorithm then chooses an *action* a_j from the set of actions $\mathcal{A} = \{a_1, a2, ..., a_n\}$.

A policy $\pi : (\mathcal{X}, \mathcal{A}) \to [0, 1]$, is defined as the set of probabilities $p(a_j|x) \quad \forall (a_j, x) \in (\mathcal{A}, \mathcal{X})$, of selecting each action a_j for each input is x.

We define $Q^\pi_j(x)$, with $Q^\pi_j : \mathcal{X} \to \Re$, as the expected return when taking action a_j and following π thereafter.

$$Q^\pi_j(x) = E_\pi \Big\{ \sum_{k=0}^{\inf} \gamma^k r^{t+k+1} | x^t = x, a^t = a_j \Big\}$$

As a matter of fact, $Q^\pi_j(x)$ is unknown and must be evaluated. The evaluation function is performed by the predictors $\mathcal{P}$, whose output shall be defined as $Q_j(x)$. Multi-layer perceptrons trained with the back-propagation algorithm have been chosen as the function approximation method with which to implement the predictors.

2.1 Action Selection and Policy Improvement

A policy π can be derived from the evaluations $Q_j(x)$ as follows:

At each presented input x, $Q_i(x)$ is the estimated value of action a_i. The simplest way to derive a policy from the estimated values is to simply pick the action with the largest expectation. However, for exploration reasons, we will use $\epsilon - greedy$ action selection. In this method, for $\epsilon \in (0, 1)$ a random action is selected with probability ϵ. Otherwise, action a_j is selected, with $Q_j(x) = max_i Q_i(x)$.

It has to be noted here that the evaluations $Q_j(x)$ might not be the true expected return $Q^\pi_j(x)$. In order to set meaningful targets for the predictors p_i, a method of policy evaluation is needed. In this paper, a state-action value method called Q-learning [12] is used.[2]

2.2 Using Q-learning to Control the Ensemble

For this particular problem, at each time-step t one example is presented to the ensemble. An action a_j is selected from a set of actions $A = \{a_1, a_2, ..., a_n\}$, where n is the number of experts. The selected action a_j corresponds to selecting expert e_j to process the current example, and its parameters are adjusted towards the error gradient. The ensemble decision then is the hypothesis emitted by e_j. A reward $r^t \in \{-1, 0\}$ is returned, it being -1 in the case of incorrect classification, 0 otherwise.[3]

[2]Q-learning performs what is referred to as *policy iteration*. At each step, the evaluation function is adjusted towards the expected return (evaluation step). Then, the policy π is itself changed to take into account the improved estimates by the new evaluation function.

[3]There is particular reason for choosing these values. All predictors p_i are initialized with low random weights so that initially they emit evaluations close to 0. By giving only negative rewards, the Q-learning algorithm is forced to explore the policy space. See [11] for more details

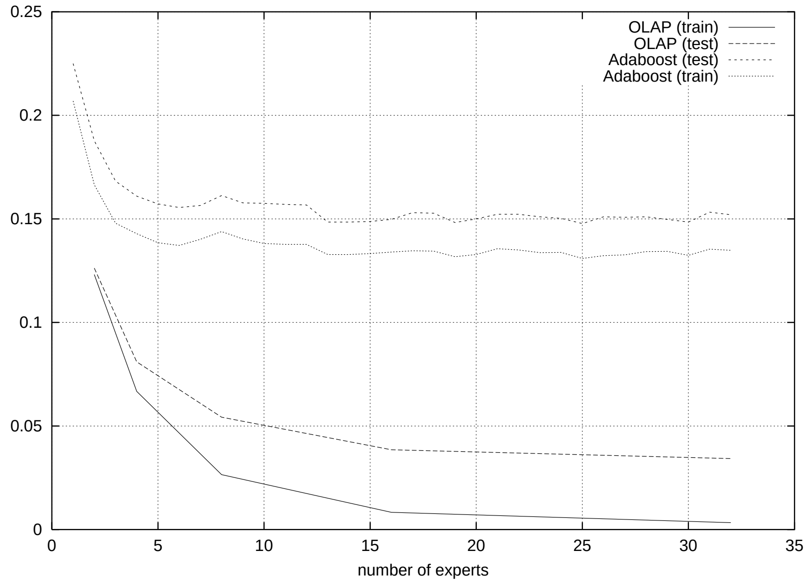


Figure 1: Test and training error for the letter data set, comparing the performance of AdaBoost.M1 and the on-line adaptive policy

Assuming that p_i can be expressed as a parameterized function with parameter vector θ_i, and $\alpha > 0$, the algorithm is used as follows:

1. For each observed example x, choose a_j derived from Q (for example using ϵ-greedy action selection)
2. Take action $a_j \in A$, observe r and the next example x'
3. Calculate $\delta \leftarrow r + \gamma \max_i Q_i(x') - Q_j(x)$
4. $\theta_j \leftarrow \theta_j + \alpha\delta\nabla_{\theta_j} Q_j(x)$
5. $x \leftarrow x'$.
6. Repeat until termination.

3 Experimental results

A small set of experiments has been performed in order to evaluate the effectiveness of this approach. These were done on the Letters and OptDigits datasets that are available from the UCI Machine Learning Repository[3].

OptDigits, the hand-written digit recognition set consisted of 3224 training examples and 2394 test examples, randomly selected from the original data. The base classifier used was a Multi-Layer Perceptron with 25 hidden units.[4] For the On-Line Adaptive Policy method, Q-values were generated by MLPs with 10 hidden units, one MLP evaluating each action.

Letters, The hand-written character recognition set consisted of 16,000 training examples and 4,000 test examples. The base classifier used was a Multi-Layer Perceptron with 140 hidden units.

[4]A number of hidden units sufficient for a single MLP to have reasonable performance was selected. In the same manner, the number of hidden unit for the evaluation networks p_i was chosen that could drive δ sufficiently close to 0.

Dataset	Method	Train	Test
OptDigits	AdaBoost	0.000	0.014
	OLAP	0.000	0.030
	MLP	0.020	0.040
Letters	AdaBoost	0.130	0.152
	OLAP	0.033	0.034
	MLP	0.210	0.230

Table 1: Classification error results for the two data sets. In both OLAP and AdaBoost, a comparison is made between the values obtained for 32 experts. Base methods used the same Multi-Layer Perceptron as the base classifier, whose result is shown also on this table for reference. The learning rate used was 0.01, while the number of hidden units was set to 25 and 140 for the Digits and Letters datasets respectively.

For the On-Line Adaptive Policy method, Q-values were generated by MLPs with 50 hidden units, one MLP for each possible action.

The learning parameters chosen was a learning rate $\alpha = 0.01$ for both the evaluation networks and the classifiers themselves. For Q-learning, a range of values between 0 and 0.9 were chosen for γ. However this did not measurably affect the performance of the ensembles. The experiments reported here are for $\gamma = 0$. ϵ-greedy action-selection was used, with $\epsilon = 0.1$.

Because the algorithm described here is working online, samples are presented randomly from the training set at each iteration. In order for a meaningful comparison to be made with AdaBoost, the epoch in OLAP has been defined as a number of iterations equal to the number of examples present in the training set. In both datasets, each base classifier was trained for 250 epochs[5]

As can be seen in Table 1, both ensembles manage to improve performance in the test cases, compared to the base classifier. However, in the digit dataset, the improvement is minimal for the online ensemble. In fact, for the online ensemble in this particular dataset, the generalization error remains constant after 4 experts have been added. This is despite the improvement in the minimum margin that is shown in Figure2(a).

On the letter recognition dataset, the online algorithm performs significantly better than AdaBoost, while AdaBoost's performance levels off after 5 experts, as can be clearly seen in Figure 1.

4 Conclusions and Future Research

The aim of this work was to demonstrate the feasibility of using adaptive policies to train and combine a set of base classifiers. While this purpose has arguably been reached, the variance in performance cannot be easily accounted for. One reason would be that the sampling performed by AdaBoost and OLAP is different. AdaBoost concentrates its efforts for each successive expert into the most difficult examples, while OLAP concentrates each expert into a particular part of the input space, probably performing some kind of clustering. A further argument towards the difference of the sampling is that the evaluation function used in OLAP is smooth, which facilitates the division of the input space into large, homogeneous regions. One future consideration then, would be using some class of parameterized functions other than MLPs for the predictors p_i.

Another thing to consider would be the space of policies that is searched. In this work, the set of possible policies was limited to ones that can only generate actions to select a single base classifier to be trained and used as an output. However, more complicated policies cannot be practically implemented in the context of Q-learning[12], because the number of actions would become unmanageably

[5]For the online algorithm, this is only an average value, as the number of training iterations each base classifier will undergo is proportional to the examples present in the volume of input space that has been allocated to it.

large, thus limiting the ability to accurately explore sufficiently explore the value-function space, and subsequently correctly evaluate $Q_i(x)$.

In that case it would appear necessary to replace action-value methods for policy improvement, such as Q-learning, with direct gradient descend in policy space[2]. The latter methods have also been theoretically proven to converge in the case of multiple agents and are much more suitable for problems in partially observable environments and with large action spaces.

Acknowledgments

Christos Dimitrakakis would like to thank the Swiss NSF for financial support (project FN2100-061243.00)

References

[1] Ran Avnimelech and Nathan Intrator. Boosted mixture of experts: An ensemble learning scheme. *Neural Computation*, 11(2):483–497, 1999.

[2] Jonathan Baxter and Peter L. Bartlett. Reinforcement learning in POMDP's via direct gradient ascent. In *Proc. 17th International Conf. on Machine Learning*, pages 41–48. Morgan Kaufmann, San Francisco, CA, 2000.

[3] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[4] L. Breiman. Arcing the edge, 1997.

[5] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[6] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, San Francisco, CA, 1996.

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

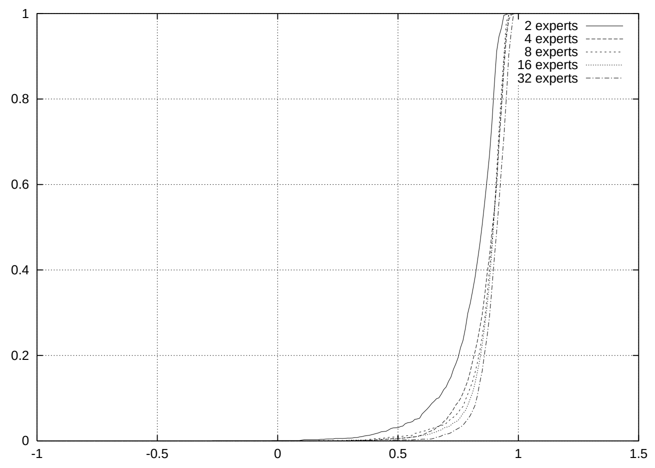
[8] Yi Li and Philip M. Long. The relaxed online maximum margin algorithm. *Machine Learning*, 46(1/3):361, 2002.

[9] Llew Mason, Peter L. Bartlett, and Jonathan Baxter. Improved generalization through explicit optimization of margins. *Machine Learning*, 38(3):243, 2000.

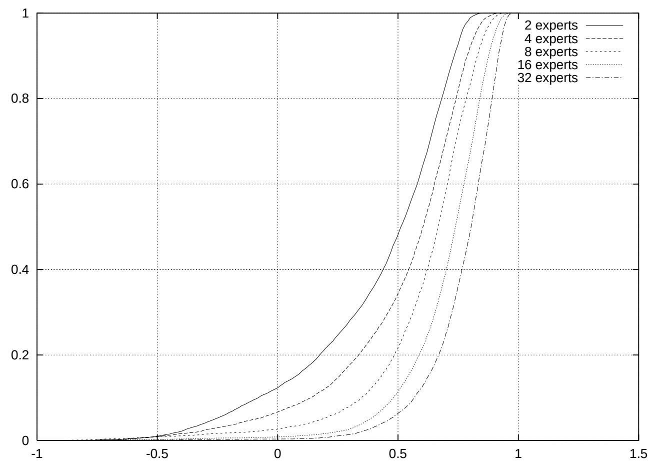
[10] Holger Schwenk and Yoshua Bengio. Boosting neural networks. *Neural Computation*, 12(8):1869–1887, 2000.

[11] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, Massachusets 02142, 1998.

[12] Christopher J.C.H. Watkins and Peter Dayan. Technical note q-learning. *Machine Learning*, 8:279, 1992.



(a) Margin distribution of OLAP for the OptDigits dataset



(b) Margin distribution of OLAP for the Letter dataset

Figure 2: Cumulative margin distribution plots for the OnLine Adaptive Policy. 2(a) and 2(b) show the change in the distribution as the number of experts in the ensemble increases.