IDIAP RESEARCH REPORT

Online Policy Adaptation for Ensemble Classifiers

Christos Dimitrakakis [a] Samy Bengio [b]

IDIAP–RR 03-69

December 2003

Dalle Molle Institute for Perceptual Artificial Intelligence • P.O.Box 592 • Martigny • Valais • Switzerland

phone +41 – 27 – 721 77 11
fax +41 – 27 – 721 77 12
e-mail secretariat@idiap.ch
internet http://www.idiap.ch

a IDIAP, CP952, 1920 Martigny, Switzerland, dimitrak@idiap.ch
b IDIAP, CP952, 1920 Martigny, Switzerland, bengio@idiap.ch

IDIAP Research Report 03-69

Online Policy Adaptation for Ensemble Classifiers

Christos Dimitrakakis Samy Bengio

December 2003

Abstract. Ensemble algorithms can improve the performance of a given learning algorithm through 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 on several UCI benchmark databases.

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 $D_N = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, 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 class y_i.

Let us denote the output of a learning machine for a particular vector x_i as $h(x_i)$. 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 can be summarised 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 such as 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 to use a part of the training data for validation for its estimation. 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.

An alternative solution is offered by ensemble methods, such as the mixtures of experts architecture [7], bagging [4] and boosting [6]. The boosting algorithm AdaBoost in particular, has been shown to significantly outperform other ensemble techniques. Theoretical results explaining the effectiveness of AdaBoost relate it to the *margin of classification* [11].

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 $f : \mathcal{X} \to [-1, 1]$. 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 output is predicted, i.e. $y^x = \text{argmax}_y f_y(x)$ and thus:

$$\text{margin}_f(x, y) = f_y(x) - \max_{y \neq y'} f_{y'}(x). \tag{1}$$

Thus the classification margin is a particular classification decision's distance from the decision threshold. A particularly useful measure is the minimum margin over the set D_N, i.e.

$$\text{margin}_{min}(D_N) = \min_{(x,y) \in D_N} \text{margin}_f(x, y).$$

It is argued [11] that AdaBoost is indirectly maximising this margin, leading to more robust performance. Although there exist counterexamples for which the minimum margin is not an adequate predictor of generalisation [5], attempts to apply algorithms that directly maximise the margin have obtained some success[10, 9].

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 [12] provides natural candidates for use in adaptive policies. In particular, the policy is adapted using Q-learning [13], a method that improves a policy through the iterative approximation of an evaluation function Q. Previously, Q-learning has been applied in a similar mixture model applied to a control task [1]. An Expectation Maximisation based mixtures of experts (MOE) algorithm for supervised learning was presented in [8]. In this paper, we attempt to solve the same task as in the standard MOE model, but through the use of reinforcement learning rather than expectation maximization techniques.

In the rest of this paper, the framework of Reinforcement Learning will be introduced in section 2. Section 2.1 outlines how the RL methods are employed in this work and describes how the system is implemented. Experiments are described in section 3, followed by conclusions and suggestions for future research.

2 General Architecture

The Reinforcement Learning classifier ensemble consists of a set of n base classifiers, or experts, $\mathcal{E} = \{e_1, e_2, ..., e_n\}$ and a controlling agent. By recasting the classification problem as a control problem, it is possible to apply reinforcement learning techniques to implement the agent. The agent will be given the task of making a classification decision based on the output h_i of each expert e_i and to the problem of choosing which experts to train on a particular example.

For the controlling agent we define a set of states, $s \in \mathcal{S}$, a set of actions $a \in \mathcal{A}$, with $\mathcal{A} = \{a_1, a2, ..., a_n\}$.

At each time step t, the agent is at state s and can choose any action from $\mathcal{A}$. After the action is taken, the agent receives a reward r_t and it enters a new state, s'.

The aim is to maximise the discounted future return of the system, starting at time t:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

where $\gamma \in [0, 1)$ is a *discount factor*.

A policy $\pi : (\mathcal{S}, \mathcal{A}) \rightarrow [0, 1]$, is defined as the set of probabilities

$$\pi = \Big\{ p(a_j|s) \Big| (a_j, s) \in (\mathcal{A}, \mathcal{S}) \Big\}$$

for selecting an action a_j given the state s.

We define $Q^\pi(s, a_j)$, with $Q^\pi : (\mathcal{S}, \mathcal{A}) \rightarrow \Re$, as the expected return when taking action a_j and following π thereafter.

$$Q^\pi(s, a_j) = E_\pi \Big\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \Big| s_t = s, a_t = a_j \Big\}$$

Since Q^π is unknown, we evaluate it by maintaining an estimate Q. Our aim then is to improve these estimates by exploring the state-action space, while at the same time improving the policy π, based on the updated estimates.

A policy π can be derived from the evaluations $Q(s, a)$ either deterministically, by always selecting the action a_j with the largest expected return, or stochastically. $\epsilon - greedy$ action selection selects the highest evaluated action with probability $(1 - \epsilon)$, with $\epsilon \in [0, 1]$, otherwise it selects a random action. Softmax action selection selects action a_j with probability $e^{Q(s,a_j)} / \sum_i e^{Q(s,a_i)}$. After the action is taken, the next state s' is observed and the current evaluation is adjusted according to:

$$Q(s, a_j) \leftarrow Q(s, a_j) + \alpha(r + \gamma \max_i Q(s', a_i) - Q(s, a_j)) \qquad \text{with } \alpha > 0. \tag{2}$$

2.1 Implementation

We employ an architecture with n experts, implemented as multi-layer perceptrons (MLPs), and a further MLP with n outputs and parameters θ which acts as the controlling agent. At each time step t a new example x is presented to the ensemble and each expert e_i emits a decision $h_i(x)$. The state space of the controlling agent is $\mathcal{S} \equiv \mathcal{X}$, the same as the classifiers' input space. Its outputs approximate $Q(s, a_j)$ in order to select actions from $\mathcal{A}$. We examine the case in which each action a_j corresponds to selecting expert e_j for training on the current example.

The decisions of the experts themselves can be combined with a weighted sum: $f(x) = \sum_i w_i h_i(x)$, where $w_i = \frac{e^{Q(x,a_i)}}{\sum_j e^{Q(x,a_j)}}$. Alternatively we can make hard decisions by setting $f(x) = h_j(x)$, where $j = \arg\max_i Q(s, a_i)$. The classification decision results in a return $r \in \{0, 1\}$, which is 1 if $f(x) = y$ and 0 otherwise.

The Q-learning update remains essentially the same as in (2), but because of the parameterised representation, we perform gradient descent to update our estimates, with the back-propagated error being $\delta = r + \gamma \max_i Q(s', a_i) - Q(s, a_j)$ and learning rate $\alpha > 0$. The algorithm is implemented as follows:

1. Select example x_t randomly from $\mathcal{X}$.
2. Given $s = x_t$, choose $a_j \in \mathcal{A}$ according to a policy derived from Q (for example using ϵ-greedy action selection) .
3. Take action a_j, observe r and the next state $s' = x_{t+1}$, chosen randomly from $\mathcal{X}$.
4. Calculate $\delta \leftarrow r + \gamma \max_i Q(s', a_i) - Q(s, a_j)$.
5. $\theta \leftarrow \theta + \alpha \delta \nabla_\theta Q(s, a_j)$.
6. $s \leftarrow s'$.
7. Loop to 2, unless termination condition is met.

3 Experimental results

A small set of experiments has been performed in order to evaluate the effectiveness of this approach, on 9 datasets that are available from the UCI Machine Learning Repository [3]. For each dataset, cross-validation was used to select the number of hidden units for the base classifier. Each classifier was trained for 100 iterations and a learning rate $\alpha = 0.01$ was used. The discount parameter γ for the controlling agent was set to 0[1]. The results reported here are for ϵ-greedy action selection and for the hard combination method. Results with softmax action selection and weighted combination are not significantly different.

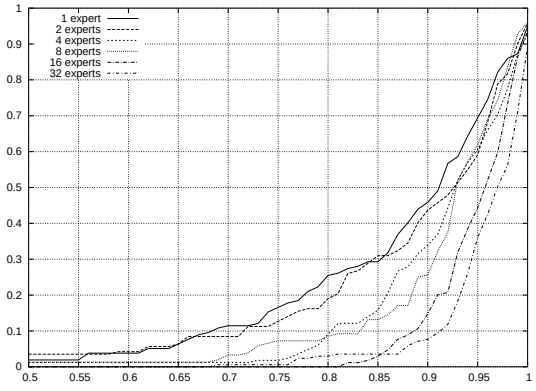

Figure 1: Margin distribution for RL on the ionosphere dataset, with an increasing number of experts

Dataset	MLP	Boost	RL
breast	7.28%	1.21%	2.43%
forest	32.8%	16.2%	29.1%
heart	15.0%	16.2%	15.0%
ionosphere	5.96%	5.96%	3.08%
letter	4.10%	2.52%	3.73%
optdigits	2.63%	1.42%	2.3%
pendigits	2.72%	3.10%	2.69%
spambase	8.33%	6.48%	7.41%
vowel	56.1%	61.9%	48.3%

Table 1: Classification error on the UCI datasets using 32 experts

The RL-controlled mixture was compared with a single MLP and AdaBoost using MLPs, which represents the state of the art in ensemble classifiers. As can be seen in Table 1, both ensembles manage to improve test performance compared to the base classifier, with the RL mixture outperforming

[1]The classification task is similar to an n-armed bandit problem, since the next state is not influenced by the agent's actions. However it is more accurately described as a partially observable process, since the parameters of the classifiers constitute a state which changes depending on the agent's actions.

AdaBoost 4 times out of 9. For each dataset we have also calculated cumulative margin distribution resulting from equation (1). For the RL mixture, in most, but not all, datasets there was a constant improvement in the distribution with an increasing number of experts (c.f. figure 1), though this did not always result in an improvement in generalisation performance.

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, there still remain some questions to be answered, such as under what conditions the margin of classification is increased when using this approach.

In the future we would like to explore the relationship between RL and EM techniques for training ensembles. Furthermore, it would be interesting to investigate the application of RL when the agent's state space is extended to include information about each expert. In this case it would no longer constitute of i.i.d samples, so the agent's actions will affect its future state.

For enlarged spaces however it would appear necessary to replace action-value methods for policy improvement, with direct gradient descent 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 state-action spaces.

References

[1] C. Anderson and Z. Hong. Reinforcement learning with modular neural networks for control, 1994.

[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. http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998.

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

[5] Leo Breiman. Arcing the edge. Technical report, Department of Statistics, University of California, Berkeley, CA., 1997.

[6] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[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] Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.

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

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

[11] 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.

[12] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

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