Online Policy Adaptation for Ensemble Classifiers

Christos Dimitrakakis and Samy Bengio
IDIAP, P.O. Box 592, CH-1920 Martigny, Switzerland

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 addressed in the past using supervised learning methods. In this context, a set of N example patterns $\hat{D} = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ 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 $\varepsilon_i = 1$ if $h(x_i) \neq y_i$ and 0 otherwise. Thus, the classification error for the set of examples $\hat{D}$ can be summarised as the empirical error $\hat{L} = \sum_i \varepsilon_i / N$. If $\hat{D}$ is a sufficiently large representative sample taken from a distribution D, the generalization error $L = \int p_D(x)\varepsilon(x)$ would be close to $\hat{L}$. In practice, however, the training set provides limited sampling of the distribution D, leading to problems such as overfitting. Adding the effects of the classifier's inherent bias and variance, we will have $L > \hat{L}$.

Since the generalization error cannot be directly observed, 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.

Another possible solution is offered by ensemble methods, such as the mixtures of experts (MOE) architecture [7], bagging [4] and boosting [6]. The boosting algorithm AdaBoost has been shown to significantly outperform other

This work is supported by the Swiss National Science Foundation through the National Centre of Competence in Research on "Interactive Multimodal Information Management".

ensemble techniques. Theoretical results explaining the effectiveness of AdaBoost relate it to the *margin of classification* [11]. The margin distribution for the 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 $h(x) = \text{sign}(f(x))$. In this case, $|f(x)|$ can be interpreted as the confidence in the label prediction. For the multi-class case, let $f_y(x)$ be the model's estimate of the probability of class y given input x. In this case the margin is defined as:

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

Thus the margin can serve as a measure of how far away from the threshold classification decisions are made. A particular measure is the minimum margin over the set $\hat{D}$, i.e.:

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

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 had been used 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.

The rest of the paper is organised as follows. The framework of Reinforcement Learning (RL) is 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 RL classifier ensemble consists of a set of n base classifiers, or experts, $\mathcal{E} = \{e_1, e_2, ..., e_n\}$ and a controlling agent that selects the experts to make classification decisions and to train on particular examples. The specific RL algorithm employed in this work is outlined below. The following section describes in what manner it was used to control the classifier ensemble.

For the controlling agent we define a set of states $s \in \mathcal{S}$ and a set of actions $a \in \mathcal{A}$. At each time step t, the agent is at state $s_t = s$ and chooses action $a_t = a$. After the action is taken, the agent receives a reward r_t and it enters a new state $s_{t+1} = s'$. A policy $\pi : (\mathcal{S}, \mathcal{A}) \rightarrow [0, 1]$ is defined as a set of probabilities:

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

for selecting an action a given the state s. The objective is to find the policy that maximises the discounted future return of the system, starting at time t, which is defined as:

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

where $\gamma \in [0, 1)$ is a *discount factor*. This can be achieved by the iterative application of two steps. First, we estimate the return of actions under the current policy π. More specifically, we define $Q^\pi : (\mathcal{S}, \mathcal{A}) \rightarrow \Re$ as the expected return of taking action a when being at state s at time t and following π thereafter:

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

Q^π itself is unknown and we maintain instead an estimate Q for each state action pair. Herein we employ the Q-learning update when action a_j is selected in state s:

$$Q(s, a_j) \leftarrow Q(s, a_j) + \eta(r + \gamma \max_i Q(s', a_i) - Q(s, a_j)), \qquad \eta > 0 \qquad (2)$$

The second step involves deriving a policy π from the updated estimates Q. This 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 $e^{Q(s,a_j)} / \sum_i e^{Q(s,a_i)}$. Stochastic action selection is in general necessary so that all state-action pairs are sampled frequently enough for us to have accurate estimates of their expected return.

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 $\eta > 0$. 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. $\theta \leftarrow \theta + \eta \delta \nabla_\theta Q(s, a_j)$.
6. $s \leftarrow s'$.
7. Loop to 2, unless termination condition is met.

3 Experimental results

A 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 $\eta = 0.01$ 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.

A comparison was made between the RL-controlled mixture, a single MLP, the Mixture of Experts and AdaBoost using MLPs. As can be seen in Table 1, the ensembles generally manage to improve test performance compared to the base classifier, with the RL mixture outperforming AdaBoost and MOE 4 and

[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.

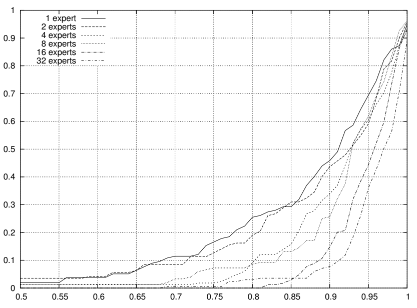

Figure 1: Cumulative margin distribution for RL on the ionosphere dataset, with an increasing number of experts.

MLP	Boost	MOE	RL
7.28%	1.21%	4.84%	2.43%
32.8%	16.2%	31.6%	29.1%
15.0%	16.2%	13.7%	15.0%
5.96%	5.96%	5.96%	3.08%
4.10%	2.52%	4.55%	3.73%
2.63%	1.42%	2.13%	2.3%
2.72%	3.10%	2.80%	2.69%
8.33%	6.48%	7.75%	7.41%
56.1%	61.9%	68.1%	48.3%

Table 1: Classification error on the UCI breast, forest, heart, ionosphere, letter, optdigits, pendigits, spambase and vowel datasets using 32 experts.

7 times out of 9 respectively. For each dataset we have also calculated the cumulative margin distribution resulting from equation (1). For the RL mixture there was a constant improvement in the distribution in most datasets when the number of experts was increased (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. However, perhaps the most promising direction in this domain would be to extend the set of possible actions so that more interesting policies can be developed. For such enlarged spaces 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.