Neurocomputing ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Selective neural network ensembles in reinforcement learning: Taking the advantage of many agents

Stefan Faußer *[,1], Friedhelm Schwenker [1]

Institute of Neural Information Processing, University of Ulm, 89069 Ulm, Germany

ARTICLE INFO

Article history:
Received 29 June 2014
Received in revised form
15 October 2014
Accepted 4 November 2014

Keywords:
Selective ensemble learning
Neural network ensembles
Reinforcement learning with function approximation
Large state environments

ABSTRACT

Ensemble models can achieve more accurate and robust predictions than single learners. A selective ensemble may further improve the predictions by selecting a subset of the models from the entire ensemble, based on a quality criterion. We consider reinforcement learning ensembles, where the members are artificial neural networks. In this context, we extensively evaluate a recently introduced algorithm for ensemble subset selection in reinforcement learning scenarios. The aim of the learning strategy is to select members whose weak decisions are compensated by strong decisions for collected states. The correctness of a decision is determined by the Bellman error. In our empirical evaluations, we compare the benchmark performances of the full ensembles and the selective ensembles in generalized maze and in SZ-Tetris. Both are large state environments. We found that while the selective ensembles have a small number of agents, they significantly outperform the large ensembles. We therefore conclude that selecting an informative subset of many agents may be more efficient than training single agents or full ensembles.

1. Introduction

Reinforcement Learning (RL) methods [1] offer an elegant way to learn to take the best action by trial-and-error in a Markov Decision Process. In this work, we focus on large state space environments with both a discrete set of states and actions. The environments are stationary and stochastic. For learning the state or state–action values in large state spaces, we consider Multi-Layer Perceptrons (MLP) for the function approximation (FA) of the values. From the RL methods, we focus on model-free methods, where the models are learned by step-wise interaction with the environment. The values are updated on-policy, i.e. with the state–action pairs or the states that an agent observed by following the policy.

Whereas ensemble or committee-based models have been successfully applied to supervised [2,3], semi-supervised and active learning [4–6], in RL not much research has been done to combine agents in a committee. One of the first attempts was done in [7], where fitted Q iteration used ensembles of regression trees. In [8], values learned from different RL algorithms were combined in joint decisions. However, Q-Learning single learners seemed to outperform committees in the more difficult problems. Ref. [9] combined the Q-values of multiple neural networks, trained by the Neural Fitted Q-iteration (NFQ) algorithm. While NFQ is known to be applicable for large state spaces, they empirically evaluated their ensembles on the simple pole balancing problem with large neural networks (4-layer up to 10-layer MLPs) with some success. In another work [10,11], it has been analytically shown that a committee with joint decisions, with estimated values from agents with FA, can perform more rewarding decisions than a single agent. Each agent was trained by the same RL method. An RL committee benefits from the diversities on the value estimations, both from unstable value estimators and from large state spaces. Empirical evaluations with SZ-Tetris and generalized maze confirmed the analytical results [10].

In a selective ensemble, an informative subset is taken of a large ensemble with the aim of performing better predictions. In [12], this was done for both classification and regression. Later on, unlabelled data were included to improve the performance of a selective ensemble [13]. In [14], we recently described a method to select an informative subset of agents from a full ensemble with the aim to achieve more accurate value estimations. To our knowledge, this was the first attempt to combine selective ensemble learning and RL.

1.1. Contribution

Our contribution in this paper is an extensive evaluation of the full ensembles [10] and the selective ensembles [14] in RL. In the

* Corresponding author.
E-mail addresses: stefan.fausser@uni-ulm.de (S. Faußer), friedhelm.schwenker@uni-ulm.de (F. Schwenker).
[1] Tel.: +49 731 50 24151; fax: +49 731 50 24156.

http://dx.doi.org/10.1016/j.neucom.2014.11.075
0925-2312/

following, the neural network ensembles refer to the full ensembles, and the selective neural network ensembles refer to the selective ensembles. As for the full ensembles, we do not include the ensembles with joint decisions in the learning phase, see [10]. In the same reference, an overview and a comparison of the non-selective ensembles in RL can be found. The key difference between our work in [10], the full ensembles, and the related works is that we consider RL and function approximation as a general tool in the ensemble methods. In contrast, in [8], different RL methods were combined and in [9], deep network architectures were utilized for performance improvements in comparably simple RL problems (basic maze and pole balancing). We evaluate empirically the full ensembles and the selective ensembles on the generalized maze and on SZ-Tetris. Compared to our work in [14], we further elaborate the analysis of the objective function of the proposed method. In the experiments, we additionally train the single agent by the Temporal-Difference with gradient correction (TDC) method [15,16]. Furthermore, we increased the size of the validation set for the generalized maze.

1.2. Outline

The outline of this paper is as follows. First, we summarize the neural network ensembles in Section 2. Second, we introduce and discuss the error function of the selective neural network ensembles in Section 2.1. The errors, used in the objective function, are defined in Section 2.2. In Section 3, we describe our performed experiments and empirically evaluate the neural network ensembles and the selective neural network ensembles on two large state environments. The results for the first environment, generalized maze, are in Section 3.1 and the results for the second environment, SZ-Tetris, are in Section 3.2. The paper ends with a conclusion.

2. Preliminaries

Given is a set $D=\{A_1, A_2, \ldots, A_M\}$ of M agents. Each agent is represented by an FA with weights θ_m and trained through some RL method, including Value Iteration, Temporal-Difference (TD) or Monte-Carlo. In this work, we focus on MLPs as FA and on the model-free methods. Each agent in D is trained by the same RL method. After the training of T iterations or episodes, the performance of these M agents is tested, keeping their weights fixed. From now on this is called simulation phase, benchmark or testing in contrast to the training phase, where the agents learn and update their weights. In the simulation phase, the agents can act as a committee and perform joint decisions. The Joint Average policy is

$$\Pi^{AV}(s) = \arg\max_{a \in \mathcal{A}(s)} \left[\sum_{m \in D} Q_{\theta_m}(s,a) \right] \tag{1}$$

where s is the current observed state, $\mathcal{A}(s)$ is a discrete set of actions available in state s and $Q_{\theta_m}(s,a)$ is the estimated value of agent m. Further, the Joint Majority Voting policy is

$$\Pi^{VO}(s) = \arg\max_{a \in \mathcal{A}(s)} \left[\sum_{m \in D} N_m(s,a) \right] \tag{2}$$

where $N_m(s,a)$ is one if agent m votes for action a in state s, else zero. In contrast to this, an agent that is acting outside a committee performs single decisions. The Single policy for an agent m is

$$\Pi_m(s) = \arg\max_{a \in \mathcal{A}(s)} \left[Q_{\theta_m}(s,a) \right] \tag{3}$$

An ensemble with all agents from set D is called a full ensemble. Such an ensemble can achieve more accurate value estimations than the single agents. Mainly, this is due to the diversities on the value estimations, both from unstable value estimators and from large state spaces. As the value estimations contribute to the decisions, it follows that more accurate value estimations result in more accurate decisions. In a control problem, an agent tries to maximize its total reward. With more accurate decisions, the committees can get higher total rewards than the single agents. The total reward is the sum of the discounted rewards received until the end of the episode.

2.1. Selective neural network ensembles

Out of the set D with agent indices of a large ensemble, we can select a subset $\tilde{D} \subset D$, $|\tilde{D}| = \tilde{M}$, by minimizing the following objective function:

$$E(x) = \sum_{s \in \mathcal{S}} \hat{p}(s) \left[\sum_{m \in D} x_m \sum_{a \in \mathcal{A}(s)} \alpha_m(s,a) e_m(s,a) \right]^d \tag{4}$$

under the constraints $\sum_{m \in D} x_m = \tilde{M}$, $x_m \in \{0,1\}$, $\forall m$, where $\mathcal{S}$ is a discrete set of states, $\hat{p}(s)$ is the probability to observe the state s, α is a weighting function with $\sum_{a \in \mathcal{A}(s)} \alpha_m(s,a) = 1$, $\forall s, \forall m$, $e_m(s,a) \geq 0$ is a bounded real-valued error of agent m for state–action pair (s,a), $d \geq 1$ is the degree and x are the variables. Both $\hat{p}$ and α are error weighting functions. With the weights from $\hat{p}$, the errors are weighted according to the occurrences of the states. Specifically, states that are more likely observed have a higher weight and, thus, are credited more in the objective function than rarely observed states. The weighting function α depends on the joint decisions and is described in Section 2.2. With $d=1$, the value of Eq. (4) is the sum of the weighted errors of the individual agents. From this it follows that the lowest value of Eq. (4), with $d=1$, is reached by selecting $\tilde{M}$ agents with the independently summed lowest errors. On the other hand, in this case, the relations of the errors between the agents are not considered. This may lead to situations where predominantly similar agents, in terms of similar errors, are selected. As we prefer to have varied agents, we set $d=2$. With $d=2$, Eq. (4) can be reformulated as a constrained binary quadratic programming (BQP) problem:

$$E(x) = x^T B x \tag{5}$$

under the same linear constraints as for (4) and with

$$B_{ij} = \sum_{s \in \mathcal{S}} \hat{p}(s) \left[\sum_{a \in \mathcal{A}(s)} \alpha_i(s,a) e_i(s,a) \sum_{a \in \mathcal{A}(s)} \alpha_j(s,a) e_j(s,a) \right] \tag{6}$$

Note that (5) is equivalent to (4). Solving the problem exactly requires a linearization of the problem as done in [17]. Unfortunately, these strategies are computationally expensive for large vector sizes of the variables. Thus, we approximately solve the problem by solving a quadratic programming problem with box constraints (QP-BOX):

$$E(w) = w^T B w \tag{7}$$

under the constraints $\sum_{m \in D} w_m = \tilde{M}$, $w_m \in \mathbb{R}$, $1 \leq w_m \leq 0$, $\forall m$ and with B defined as in (6). Here, B is a positive-definite matrix, as each entry in the matrix is the result of an inner product and $\sum_{a \in \mathcal{A}(s)} \alpha_i(s,a) e_i(s,a) \geq 0$, $\forall i$, by definition. Thus, the objective functions (4), (5) and (7) are convex functions. From this it follows that a local minimum for (7) is also a global minimum. Further, we can conclude that the problem of minimizing the quadratic function with QP-BOX is P, due to the convex objective function, while with BQP the problem is NP-hard. The difference between the two approaches is that QP-BOX allows for partially selecting the agents, while BQP selects the agents fully. Fortunately, the square function penalizes larger variables more than smaller variables and, thus, the solutions with variables close to zero are preferred. In contrast, the sum of the variables must be $\tilde{M}$ and,

Please cite this article as: S. Faußer, F. Schwenker, Selective neural network ensembles in reinforcement learning: Taking the advantage of many agents, Neurocomputing (2015), http://dx.doi.org/10.1016/j.neucom.2014.11.075