Neural Network Ensembles in Reinforcement Learning

Article *in* Neural Processing Letters · November 2013

DOI: 10.1007/s11063-013-9334-5

2 authors:

Stefan Faußer
Hochschule Neu-Ulm
14 PUBLICATIONS **121** CITATIONS
SEE PROFILE

Friedhelm Schwenker
Ulm University
314 PUBLICATIONS **4,505** CITATIONS
SEE PROFILE

Neural Processing Letters manuscript No.
(will be inserted by the editor)

Neural Network Ensembles in Reinforcement Learning

Stefan Faußer · Friedhelm Schwenker

Received: date / Accepted: date

Abstract The integration of function approximation methods into reinforcement learning models allows for learning state- and state-action values in large state spaces. Model-free methods, like Temporal-Difference or SARSA, yield good results for problems where the Markov property holds. However, methods based on a temporal-difference are known to be unstable estimators of the value functions, when used with function approximation. Such unstable behavior depends on the Markov chain, the discounting value and the chosen function approximator. In this paper, we propose a meta-algorithm to learn state- or state-action values in a Neural Network Ensemble, formed by a committee of multiple agents. The agents learn from joint decisions. It is shown that the committee benefits from the diversity on the estimation of the values. We empirically evaluate our algorithm on a generalized maze problem and on SZ-Tetris. The empirical evaluations confirm our analytical results.

Keywords neural network ensemble · learning from unstable estimations of value functions · reinforcement learning with function approximation · large environments

1 Introduction

Reinforcement Learning (RL) methods [13] offer an elegant way to learn to predict state values or to learn to take the best action by trial-and-error in a Markov Decision Process (MDP). Aside from the model-based Dynamic Programming methods, the model-free methods like Temporal-Difference (TD) or Monte-Carlo can be applied to learn the model by step-wise interaction with the environment. Large state spaces impose some problems on RL methods. If the values are stored in tables, then the RL methods need numerous iterations over the states to get values with

S. Faußer · F. Schwenker
Institute of Neural Information Processing, University of Ulm, 89069 Ulm, Germany
E-mail: stefan.fausser@uni-ulm.de, friedhelm.schwenker@uni-ulm.de

enough precision. Therefore, it is common to approximate the values by a continuous value function. Based on the selection of the feature space, the values are dependent on each other and may result in faster learning with fewer iterations.

Several RL methods have been combined with function approximation techniques, mainly linear or generalized linear models [2] such as radial basis function networks [12]. Unfortunately, not all RL methods converge to a fixed-point of the values dependent on the environment, the discounting value and the function approximator. In the star problem, Baird [1] has empirically shown that the TD method with linear function approximation diverges unbounded from the true values. Furthermore, Baird has introduced the Residual-Gradient (RG) algorithm being a gradient descent on the Mean Squared Bellman Error (MSBE). RG converges to the true values in the star problem, but is for problems with high discounting rate slower than TD. Later on, it has been proven [16] that TD with linear function approximation theoretically converges to the true values if the policy is kept fixed, and the states are sampled out of the policy that is evaluated (on-line state sampling or on-policy learning). Further, the worst-case error bounds of TD and RG with linear function approximation have been analytically compared by Li [10]. Applying TD results in lower prediction errors than with RG, while applying RG results in lower temporal-difference errors than with TD. Recently, the TD with gradient correction (TDC) and gradient temporal difference (GTD2) algorithms [14] have been introduced, which minimize the Mean Squared Projected Bellman Error (MSPBE). Both methods are able to learn from off-line state sampling (off-policy learning). While these methods solve Baird's star problem (like RG) and have been shown to converge theoretically to a fixed point, they need twice the weights and a second learning rate to tune and only seem to perform nearly as well as TD for 9×9 Computer Go [14]. Scherrer [11] mathematically analyzed and compared the minimization of the Bellman Residual (BR) with the computation of the TD fixed-point using his projected view and has observed that minimizing the BR error reduces the TD error as well but not vice-versa. Further, he has empirically observed the TD result is often better than the BR solution, but the TD and TDC methods fail more often.

In the following, we assume convergence of the value functions to a fixed-point. Still, the estimation of the value functions when combined with function approximation are unstable, i.e. multiple runs with different initialized weights result in estimators with significant variance on the estimated values. In supervised learning, Breiman [4] trained various unstable estimators with overlapping training sets, sampled from all available samples with random replacement. Breiman showed that the variance gained from the sampling-technique results in a better or equal error of the committee than each of the unstable estimators alone. Whereas ensemble or committee-based models have been successfully applied to supervised [6], semi-supervised and active learning [8, 18, 19], in RL not much research has been done to combine agents in a committee. Wiering and van Hasselt [17] combined different RL algorithms namely Q-Learning, SARSA and Actor-Critic with joint policies for learning state-action values and empirically evaluated them on several Grid World problems. As these methods have different objective functions, the estimators have different biases. Also for the more difficult Grid World problems, the Q-Learning method seemed to outperform the committees. Hans and Udluft [9] combined the Q-values of multiple neural networks, trained by the Neural Fitted Q-iteration (NFQ) algorithm. While the NFQ is known to be applicable to larger-scale state spaces, they empirically evaluated their ensembles with the NFQ algorithm on the simple Pole Balancing problem with large neural networks (4-layer up to 10-layer MLPs) with some success. More recently Fausser and Schwenker [7] introduced a committee that learns from joint decisions and average values with the TD or the RG method. In the empirical evaluations with the simple pen-and-pencil game Tic Tac Toe and a maze, some combinations of the ensemble methods showed promising results.

1.1 Contribution

Our contribution in this paper is a generalization of the ensemble methods in RL. We propose a meta-algorithm that is able to, depending on the chosen RL method, either learn state- or state-action values in a Neural Network Ensemble. It can be applied for a prediction problem or for a control problem. In a control problem, the agents in the committee learn from joint decisions such as Majority Voting or Averaging of the values. We analytically show that a committee benefits from the diversity on the estimation of the values. Empirical evaluations in two environments, each with a large state space and dynamics, namely the generalized maze problem with random upwind and stochastic SZ-Tetris confirm our analytical results.

In contrast to the works of [17], we only combine RL methods with alike objective functions to lower the biases of the estimators. Additionally to the Joint Majority Voting policy with the softmax action selection strategy, we also consider the epsilon-greedy strategy, and both with Joint Average and Joint Majority Voting decisions. Last, we apply joint decisions in the learning phase. Compared to our previous works in [7], this work introduces the theories about learning from unstable estimations of the value functions in the RL domain. Further, we evaluate the performance of the committees on more difficult problems with large state spaces.

2 Preliminaries

Given an environment with Markov property, an agent in state $s_t \in \mathcal{S}$ takes action $a_t \in \mathcal{A}(s_t)$ retrieved by policy $a_t = \pi(s_t)$, observes the next state $s_{t+1} = s'_t$ and observes reward $r_t(s_t, a_t, s_{t+1})$. $\mathcal{A}(s_t)$ is a discrete set of actions available in state s_t, $\mathcal{S}$ is a discrete set of states and s_t is the state at time t. With the Markov property, future states s_{t+1} only depend on the current state s_t and not on previous states $s_{t-1}, \dots, s_0$. In RL methods that do not separate the policy from the state or state-action values, e.g. Value Iteration, Temporal-Difference (TD) or Monte-Carlo (MC), any update to the values updates the policy as well. Improvements to the policy are desired in control problems, where in a prediction problem, the policy is kept fixed and is being evaluated. In a control problem, an agent tries to maximize its total reward. The expected total reward given policy π, state s and action a is:

$$J(s, a, \pi) = \mathbb{E}_\pi \Big\{ \sum_{t=0}^{\infty} \gamma^t r_t(s_t, a_t, s_{t+1}) \mid s_0 = s, a_0 = a, a_t = \pi(s_t), s_{t+1} \Big\} \quad (1)$$

where $0 < \gamma \leq 1$ is a discounting factor and $r_t(s_t, a_t, s_{t+1})$ is the immediate reward for taking action a_t in state s_t at time t and observing state s_{t+1}. For an optimal policy π^*, the optimal state value function is $V^*(s) = \max_{a \in \mathcal{A}} J(s, a, \pi^*)$ and the optimal state-action value function is $Q^*(s,a) = J(s, a, \pi^*)$. The model-free methods update their values either immediately (Online-TD, TD with eligibility trace, etc.) with the 5-tuple $\{s_{t-1}, a_{t-1}, r_{t-1}(s_{t-1}, a_{t-1}, s_t), s_t, a_t\}$ or collects these tuples and updates the values as soon as a terminal state is observed (Batch-TD, MC, etc.). If the values are generated by a function approximator (FA), i.e. $V_\theta(s) \sim V(s)$, $Q_\theta(s,a) \sim Q(s,a)$, then the weights θ are updated to update the values indirectly. The optimal value function fulfills the Bellman optimality equation:

$$Q^*(s,a) = \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \Big[r(s,a,s') + \gamma V^*(s') \Big] \tag{2}$$

where $\mathcal{P}^a_{ss'}$ is the probability to get to the next state s' from state s with action a. For state values $V^*(s) = \max_{a \in \mathcal{A}(s)} Q^*(s,a)$. Many RL methods try to minimize the Bellman error which is the (squared) difference between the two sides of (2). For practical reasons, we consider to update the values with the 3-tuple $\{s_t, r_t(s_t), s_{t+1}\}$ for state values. Therefore, the value of the terminal states are learned for state values, while they are zero for state-action values. A policy π is considered to be deterministic, if it assigns a state a single action with probability 1, and stationary, if it only depends on the last state visited. Likewise the environment is stationary, if the reward r_t does not depend on t. The model-free methods neither know the probabilities $\mathcal{P}^a_{ss'}$ nor all rewards $r(s,a,s')$, but observe the states and rewards interactively.

3 Learning from unstable estimations of the value functions

In the following, we will first justify our ensemble methods with theories based on Breiman's main theorem [4].

Theorem 1 *Let $\mathcal{L} = \{l_1, \ldots, l_N\}$ be a set with N elements $l_i = (x_i, y_i)$, $x_i = (s_i, a_i)$, $y_i = Q(x_i)$. $L^m \subseteq \mathcal{L}$, where the observations in L^m are drawn with replacement from $\mathcal{L}$ according to the distribution π^m, $m = 1, \ldots, M$. An estimator $h(x_i, L^m)$ is learned from the set L^m. The error is $e(x_i, L^m) = Q(x_i) - h(x_i, L^m)$. The average value estimation error of a single estimator is $E_{AV} = \mathbb{E}_X \mathbb{E}_Y (e(X,L))^2$, where X is the discrete, countable random variable with the N possible elements (x_i) and Y is the discrete, countable random variable with the M possible estimators $(h(x, L^m))$. If a committee with size $M > 1$ averages over the value estimations and $\pi^1 = \pi^2 = \ldots = \pi^M = \pi^*$ (best policy), then the average value estimation error of the committee is $E_{COM} \leq E_{AV}$.*

Proof (Proof of Theorem 1) For a committee with size M, whose value estimations are an average over the single value estimations, the average value estimation error of the combined estimator is $E_{COM} = \mathbb{E}_X (\mathbb{E}_Y e(X,L))^2$. We assume all samples in each L^m are observed with the same underlying distribution π^*. Due to Jensen's inequality we have $\Big(\mathbb{E}_Y h(X,L)\Big)^2 \leq \mathbb{E}_Y \Big(h(X,L)\Big)^2$.

$$\begin{aligned}E_{AV} &= \mathbb{E}_X\mathbb{E}_Y\Big(Q(X) - h(X,L)\Big)^2\\ &= \mathbb{E}_X Q(X)^2 - 2\mathbb{E}_X\mathbb{E}_Y Q(X)h(X,L)\\ &\quad + \mathbb{E}_X\mathbb{E}_Y\Big(h(X,L)\Big)^2\\ &\geq \mathbb{E}_X Q(X)^2 - 2\mathbb{E}_X\mathbb{E}_Y Q(X)h(X,L)\\ &\quad + \mathbb{E}_X\Big(\mathbb{E}_Y h(X,L)\Big)^2 = E_{COM} \end{aligned} \tag{3}$$

□

The following Corollary is from the book Bishop [3] and extended to incorporate the errors introduced in Theorem 1.

Corollary 1 *Let the errors of the estimators $e(x_i, L^m)$ be zero on average and be uncorrelated, i.e. $\mathbb{E}_X e(X, L^m) = 0$ and $\mathbb{E}_X e(X, L^m)e(X, L^l) = 0$, $\forall m, \forall l, l \neq m$. The average value estimation error of the committee is then $E_{COM} = \frac{1}{M}E_{AV}$.*

Proof (Proof of Corollary 1)

$$\begin{aligned}E_{COM} &= \mathbb{E}_X\Big(\frac{1}{M}\sum_{m=1}^{M} e(X,L^m)\Big)^2\\ &= \frac{1}{M^2}\sum_{m=1}^{M}\mathbb{E}_X\Big(e(X,L^m)\Big)^2\\ &= \frac{1}{M}E_{AV}\end{aligned} \tag{4}$$

□

The theories introduced in this section apply both for prediction problems and control problems in RL. Likewise, the state-action pairs (s_i, a_i) can be replaced by states (s_i) to learn V-values instead of Q-values. In a prediction problem, the policy π is a given fixed policy and is evaluated. According to Theorem 1, the estimators of the value functions, or agents, within a committee have a lower or equal average value estimation error than a single estimator, if the committee averages their estimated values. Thus, a committee that averages the estimated values is faster than a single agent, i.e. reaches faster a fixed-point, in case a fixed-point does exist.

In a control problem, the policy is iteratively evaluated and improved. A committee has a lower or equal average value estimation error than a single estimator and, therefore, can make more accurate decisions, if it does joint decisions with the single value estimations. With more accurate decisions and a greedy action selection policy, a committee gains a higher total reward than a single agent.

While Theorem 1 shows that the committee error can be lower than the error of a single agent, Corollary 1 shows how large the difference between these two errors can be. The requirement of the zero mean errors implies that the committee should learn from the same objective function. If the errors within a committee are uncorrelated, then the average value estimation error can be reduced drastically by $\frac{1}{M}$. Slight changes in the network architecture like different learning rates or varying number of neurons in the hidden layer in case of a Multi-Layer Perceptron (MLP) as a function approximator may contribute to more uncorrelated errors.

4 Neural Network Ensembles

Assume to have a committee with M agents $\{A_1, A_2, \ldots, A_M\}$, each with a function approximator with weights θ_m, learning rate α_m, $m = 1, \ldots, M$ and a set of starting states ST. At any time t, during the learning phase or during the simulation phase, the agents can either perform a single decision or a joint decision for the best action to take. The greedy Single policy for agent m is:

$$\pi_m(s_t) = \underset{a_t \in \mathcal{A}(s_t)}{\arg\max} \Big[Q_{\theta_m}(s_t, a_t) \Big] \tag{5}$$

where state s_t is the current, at time t, state of one of the M agents. Instead of performing a single decision, the agents in the committee can combine their estimated values in a joint decision. The greedy Joint Average policy for agent m in the committee is:

$$\pi_m^{AV}(s_t) = \underset{a_t \in \mathcal{A}(s_t)}{\arg\max} \Big[\sum_{l=1}^{M} w_m(l) Q_{\theta_l}(s_t, a_t) \Big] \tag{6}$$

where $w_m(l = m) = \frac{1}{2}$, $w_m(l \neq m) = \frac{1}{2(M-1)}$ in the training phase and $w_m(l) = \frac{1}{M}$, $\forall l$, $\forall m$ in the simulation phase. Further, the greedy Joint Majority Voting policy is:

$$\pi_m^{VO}(s_t) = \underset{a_t \in \mathcal{A}(s_t)}{\arg\max} \Big[\sum_{l=1}^{M} \hat{w}_m(l) N_l(s_t, a_t) \Big] \tag{7}$$

where $\hat{w}_m(l = m) = \lceil \frac{M-1}{2} \rceil + 0.01$, $\hat{w}_m(l \neq m) = 1$ in the training phase and $\hat{w}_m(l) = 1$, $\forall m$, $\forall l$ in the simulation phase, and $N_l(s_t, a_t)$ is the vote of agent l for action a_t in state s_t:

$$N_l(s_t, a_t) = \begin{cases} 1, & \text{if } \pi_l(s_t) = a_t \\ 0, & \text{else} \end{cases} \tag{8}$$

The Joint Majority Voting policy is expected to be more robust over large differences in the estimated values. However, a large action selection space may result in wide distributions of the votes, especially during the very first learning iterations. Likewise, the agents can perform softmax decisions, either independent as a single decision or with joint decisions. The single softmax action selection probabilities for agent m are:

$$\pi_m^{SO}(s_t, a_t) = \frac{exp\Big(Q_{\theta_m}(s_t, a_t)/\tau\Big)}{\sum_{b_t \in \mathcal{A}(s_t)} exp\Big(Q_{\theta_m}(s_t, b_t)/\tau\Big)} \tag{9}$$

where $\pi_m^{SO}(s_t, a_t)$ is the probability to select action a_t in state s_t, based on the values Q_{θ_m}, generated by the input coding of state s_t and the weights of agent

m. If parameter $\tau \to 0$, then the probability of the action with the highest value approaches 1, while for $\tau \to \infty$, the actions have equal probabilities. The Joint Average softmax action selection probabilities for agent m in a committee are:

$$\pi_m^{SO,AV}(s_t, a_t) = \frac{exp\Big(\sum_{l=1}^{M} w_m(l) Q_{\theta_l}(s_t, a_t)/\tau\Big)}{\sum_{b_t \in \mathcal{A}(s_t)} exp\Big(\sum_{l=1}^{M} w_m(l) Q_{\theta_l}(s_t, b_t)/\tau\Big)} \tag{10}$$

and the Joint Majority Voting softmax action selection policy is:

$$\pi_m^{SO,VO}(s_t) = \underset{a_t \in \mathcal{A}(s_t)}{\arg\max} \Big[\sum_{l=1}^{M} \hat{w}_m(l) N_l^{SO}(s_t, a_t)\Big] \tag{11}$$

where $N_l^{SO}(s_t, a_t)$ is the vote of agent l for action a_t in state s_t:

$$N_l^{SO}(s_t, a_t) = \begin{cases} 1, & \text{if } a_t \text{ is drawn from } \pi_l^{SO}(s_t, a_t) \\ 0, & \text{else} \end{cases} \tag{12}$$

$N_l^{SO}(s_t, a_t)$ includes probabilities in the action selection process. If $N_l^{SO}(s_t, a_t) = 1$ only when $a_t = \arg\max_{b_t \in \mathcal{A}(s_t)} \pi_l^{SO}(s_t, b_t)$ then $N_l^{SO}(s_t, a_t)$ reduces to (8). If an agent m learns state values instead of state-action values, then it directly decides for the next state $s_{t+1} \in \mathcal{S}$ with the estimated values $V_{\theta_m}(s_{t+1})$ instead of deciding upon the action a_t that then causes the agent to observe the next state. Considering this, the described policies apply for state values as well.

The complete algorithm for a Neural Network Ensemble for learning state- or state-action values, either in a prediction problem with a given policy, or in a control problem, is given in algorithm 1. To get diverse agents, we consider random weight initializations, a set with starting states ST, exploration in the action selection process, different learning rates and, if desired, slight changes in the network architecture, i.e. a varying number of neurons in the hidden layer. All agents in a committee are trained with the same chosen RL method. A single agent (parameter $M = 1$) uses the same algorithm with the random restarts (set ST), but can only perform single decisions. The algorithm is applied in the training phase of the agents to update the weights of the function approximators. In a committee with $M > 1$ agents that does joint decisions, the current acting agent m has a higher decision weight $w_m(l)$ (Joint Average) or $\tilde{w}_m(l)$ (Joint Majority Voting) than any of the other $M - 1$ agents in the committee, see (6), (7), (10) and (11). This ensures that the current acting agent has enough impact on the decisions, even with large ensembles. If the committee does single decisions (Single policy, see (5)) during the training phase, or it is a prediction problem with a fixed policy, then the loop over the episodes T (line 5 in algorithm 1) can be performed in parallel.

In the simulation phase, the weights are kept fixed and the agents are evaluated. The committee is formed by taking all of these agents. We consider taking the same joint decisions as in the learning phase with the single decisions as an exception. A committee, or multiple agents trained in parallel, that performed single decisions in the learning phase, uses joint decisions in the simulation phase. All agents are equally weighted in the joint decisions.

In the theoretical view, we have the large ensemble with all trained agents. However, for the empirical evaluations of the committees with single decisions, we

Algorithm 1 Neural Network Ensembles

```
Input: problem with Markov property, discounting value $\gamma$, set $ST$ with starting states, training iterations $T$, committee size $M$, FA (MLP, linear), learning rates $\alpha_m$, $m = 1, \dots, M$, basic RL method (TD, RG, TDC, GTD2, including batch methods, eligibility trace, same for all agents in the committee), prediction: fixed policy $\pi$, control: softmax action selection strategy with temperature parameter $\tau_m$ or $\epsilon$-greedy exploration strategy with exploration parameter $\epsilon_m$, single, average or voting policy function $\pi$ for exploiting states
Output: FA with trained weights: state values $V^\pi$ (prediction) or $V^*$ (control) or state-action values $Q^*$ (control)
 1: Initialize the weights $\theta_m$ of the FA of agent $m$, $\forall m$
 2: Select starting state $s_0^m$ randomly from $ST$, $\forall m$
 3: for $t = 0 \to T$ do
 4:   Set weights: $\tilde{\theta}_m := \theta_m$, $\forall m$
 5:   for $m = 1 \to M$ do in parallel
 6:     if softmax action selection strategy then
 7:       Select action $a_t^m$ with probabilities given by softmax policy and parameter $\tau_m$ (single decision see (9), average see (10), voting see (11)), using weights $\tilde{\theta}_m$
 8:     else if $\epsilon$-greedy action selection strategy then
 9:       Choose $e \in [0, 1]$ randomly
10:       if $e < \epsilon_m$ then
11:         Explore action: Select action $a_t^m$ randomly
12:       else
13:         Exploit action: Select action $a_t^m$ with greedy policy (single decision see (5), average see (6), voting see (7)), using weights $\tilde{\theta}_m$
14:       end if
15:     else if prediction problem, policy $\pi$ is fixed then
16:       Select action $a_t^m = \pi(s_t^m)$
17:     end if
18:     Perform action $a_t^m$
19:     Observe next state $s_{t+1}^m$
20:     Receive reward $r(s_t^m, a_t^m, s_{t+1}^m)$ for state-action values or reward $r(s_t^m)$ for state values
21:     Update FA weights $\theta_m$: $V^\pi$ with $\{s_t^m, r(s_t^m), s_{t+1}^m\}$, or $Q^\pi$ with $\{s_{t-1}^m, a_{t-1}^m, r(s_{t-1}^m, a_{t-1}^m, s_t^m), s_t^m, a_t^m\}$ with chosen RL method and learning rate $\alpha_m$, Batch or Off-Line methods: Collect the tuples and update episode-wise
22:     if $s_{t+1}^m$ is terminal state or time-out then
23:       Select starting state $s_{t+1}^m$ randomly from $ST$
24:     end if
25:   end for
26: end for
```

avert from this view for computational reasons as follows. On each benchmark run, a committee consists of $\tilde{M}$ agents, randomly drawn without replacement from the L agents trained in parallel, whereas $\tilde{M} \leq L$. In contrast, a committee with $M > 1$, that performed joint decisions during the learning phase, and were trained semi-parallel (see line 7 in the algorithm), keeps the same agents within the committee in the simulation phase.

5 Experiments

5.1 Generalized Maze

In the generalized maze problem, an agent tries to find the shortest path in a maze with some barriers and one goal. The positions of the barriers and the position of the goal are known to the agent. Therefore, it is considered to be a pathplanning problem. In our benchmark-type setting, we consider a 5×5 maze with $3 - 5$ randomly placed barriers. Out of 1000 generated unique mazes, 900 are kept fixed as a training set and the rest (100) as a validation set. Both sets are not overlapping. This way an agent tests its generalized performance in different mazes than in the learning phase. The validation set is smaller than the training set for computational reasons. In the benchmark, all mazes of the validation set are evaluated, while in the training phase, the mazes of the training set are randomly selected as soon as an agent reaches a terminal state. There are $(25 \cdot 24 \cdot 23 \cdot 22) + (25 \cdot 24 \cdot 23 \cdot 22 \cdot 21) + (25 \cdot 24 \cdot 23 \cdot 22 \cdot 21 \cdot 20) = 134191200 \sim 1.34 \cdot 10^8$ (excluding the position of the agent) different mazes exactly. Barriers and the goal are terminal states. The starting set ST includes all non-terminal states of the current active maze. The action set $\mathcal{A}$ includes four actions (north, east, south, west) in all states, except in corner states (two actions) and border states (three actions). The agent is not allowed to leave the maze. State-action values are learned with either TD-SARSA or with RG-SARSA, i.e. the agent follows the (joint or single) policy and updates the values on-policy. With a probability of 0.3, an up-wind influences the movements of the agent. On up-wind, the agent moves one state further to the north, if not blocked by a barrier. The reward function is as follows:

$$r(s, a, s') = \begin{cases} 1, & \text{if } s' \text{ is the goal} \\ 0, & \text{if } s' \text{ is a barrier or non-terminal state} \end{cases}$$

We evaluated different discounting values with a single agent and got the best results with a discounting value of $\gamma = 0.9$. With lower discounting values, an agent tends to learn faster, but with worse results in the long run. In contrast, with higher discounting values, an agent tends to learn slower, but with almost the same results in the long run. If the agent is observing a terminal state, then the next maze is randomly selected and the set ST is rebuilt. In the simulation phase, all valid starting positions, excluding the terminal states, of each maze in the validation set are evaluated, and the total reward is calculated. The agents do not learn in the simulation phase. As the environment is stochastic, due to the up-wind, the simulation is repeated 10 times per starting position. An upper bound for the total reward is calculated as follows:

$$r \begin{pmatrix} \gamma^0 & \gamma^1 & \gamma^2 & \gamma^3 \end{pmatrix} A$$

where r is the reward for reaching the goal and A is a vector with the number of required steps for reaching the goal in the middle of the maze. For no barriers, $A^T = \begin{pmatrix} 4 & 8 & 8 & 4 \end{pmatrix}$ and the (max) total reward is ~ 20.6. For 3 barriers, $A^T = \begin{pmatrix} 4 & 8 & 8 & 1 \end{pmatrix}$ and the (max) total reward is ~ 18.41 and for 5 barriers, $A^T = \begin{pmatrix} 4 & 8 & 7 & 0 \end{pmatrix}$ and the (max) total reward is 16.87. Applying the breadth-first search method, we got a total reward, averaged over the 100 mazes in the validation set, of ~ 15.79 without dynamic wind and ~ 14.38 with dynamic wind. Therefore, we roughly guess that the total reward, averaged over all mazes with different goal positions, up-wind, and with $3 - 5$ barriers is at ~ 15. A $2 - \mathit{layer}$ MLP is used as a FA with 5 neurons in hidden layer and 150 input neurons. The activation function in the hidden layer is the s-shaped hyperbolic tangent function, and the activation function in the output layer is the linear function. Out of the input neurons, 25 are for the barriers, 25 for the position of the goal and 25 per action (100 for all

Table 1 Empirical results with the Generalized Maze, ϵ-greedy exploration strategy. Evaluated is the total reward, measured in a benchmark setting after $2.5 \cdot 10^6$ to $15 \cdot 10^6$ training episodes (fifth to seventh column). The single agent ($M = 1$) reached a total reward of 13.64 ± 0.29 with RG and 13.90 ± 0.34 with TD at $25 \cdot 10^6$ training episodes. 10 repeated tests per starting state to capture the effects of the dynamic wind. 100 runs for the single agent, 50 runs for the committees with single decisions, and 20 runs for the committees with joint decisions in the training phase. Third column is the policy used for the training phase (S = Single, A = Average and V = Majority Voting), fourth column is the policy used for the simulation phase, i.e. for calculation of the total reward.

RL	M	π train	π sim.	reward $2.5M$	reward $5M$	reward $15M$
TD	1	S	S	11.99 ± 0.94	13.29 ± 0.46	13.80 ± 0.39
TD	3	S	A	13.23 ± 0.39	14.00 ± 0.23	14.34 ± 0.16
TD	5	S	A	13.80 ± 0.28	14.27 ± 0.14	14.48 ± 0.12
TD	10	S	A	14.08 ± 0.19	14.39 ± 0.10	14.58 ± 0.08
TD	3	S	V	13.02 ± 0.50	13.92 ± 0.27	14.31 ± 0.14
TD	5	S	V	13.68 ± 0.24	14.31 ± 0.14	14.50 ± 0.11
TD	10	S	V	14.10 ± 0.22	14.47 ± 0.10	14.65 ± 0.08
TD	3	A	A	13.84 ± 0.33	14.35 ± 0.14	14.54 ± 0.10
TD	5	A	A	14.11 ± 0.20	14.44 ± 0.09	14.63 ± 0.08
TD	3	V	V	13.32 ± 0.29	14.11 ± 0.18	14.44 ± 0.16
TD	5	V	V	13.89 ± 0.16	14.37 ± 0.08	14.52 ± 0.09
RG	1	S	S	9.58 ± 1.90	12.80 ± 0.69	13.53 ± 0.38
RG	3	S	A	11.74 ± 0.97	13.69 ± 0.25	14.10 ± 0.20
RG	5	S	A	12.73 ± 0.55	13.96 ± 0.21	14.31 ± 0.13
RG	10	S	A	13.31 ± 0.43	14.17 ± 0.13	14.44 ± 0.11
RG	3	S	V	11.19 ± 1.13	13.69 ± 0.31	14.07 ± 0.21
RG	5	S	V	12.24 ± 0.66	14.03 ± 0.15	14.31 ± 0.14
RG	10	S	V	13.04 ± 0.42	14.26 ± 0.11	14.50 ± 0.07
RG	3	A	A	12.83 ± 0.95	14.01 ± 0.20	14.37 ± 0.14
RG	5	A	A	13.53 ± 0.31	14.19 ± 0.19	14.50 ± 0.12
RG	3	V	V	11.09 ± 1.54	13.60 ± 0.28	14.17 ± 0.19
RG	5	V	V	12.59 ± 0.95	14.08 ± 0.18	14.40 ± 0.10

four actions) for the position of the agent. Learning rates are optimized for a single agent ($M = 1$), $\alpha = 0.01$.

Results for a single agent and committees with three ($M = 3$) to ten ($M = 10$) agents, with the epsilon-greedy exploration strategy with $\epsilon = 0.3$, and weight initializations in an interval $a = [-0.4, 0.4]$ are in table 1. 100 agents with randomly initialized weights were trained parallel. The committees with the single decisions during learning (π train is 'S') were formed by randomly selecting M out of these agents, 50 runs. An episode ends by reaching the goal or a barrier. The differences of the total rewards between $5,000,000$ episodes (column six) of the committees with three agents ($M = 3$) and $15,000,000$ episodes (column seven) of the single agent are statistically significant, i.e. the p-values of an Analysis of Variance (ANOVA) are below 0.01 with the values 13.92 ± 0.27 (line 5, TD) and 13.60 ± 0.28 (line 21, RG) as an exception (p-values are 0.04 and 0.46). Furthermore, the differences of the total rewards between $5,000,000$ episodes of the committees with five agents ($M = 5$) and $25,000,000$ episodes (see table description) of the single agent are statistically significant. The results with the softmax action selection strategy with parameter $\tau = 0.15$ for TD, respectively $\tau = 0.08$ for RG, and weight initializations in an interval $a = [-0.2, 0.2]$ are in table 2. In this table, the differences of the total rewards are statistically significant (p-values below 0.01),

Table 2 Empirical results with the Generalized Maze, softmax action selection strategy. The single agent ($M = 1$) reached a total reward of 13.87 ± 0.32 with RG and 14.12 ± 0.26 with TD at $25 \cdot 10^6$ training episodes. Otherwise same description as table 1.

RL	M	π train	π sim.	reward $2.5M$	reward $5M$	reward $15M$
TD	1	S	S	12.73 ± 0.60	13.74 ± 0.33	14.05 ± 0.27
TD	3	S	A	13.60 ± 0.29	14.23 ± 0.16	14.43 ± 0.14
TD	5	S	A	13.92 ± 0.25	14.32 ± 0.14	14.53 ± 0.09
TD	10	S	A	14.04 ± 0.17	14.41 ± 0.09	14.55 ± 0.07
TD	3	S	V	13.47 ± 0.29	14.13 ± 0.18	14.37 ± 0.18
TD	5	S	V	13.84 ± 0.25	14.36 ± 0.15	14.54 ± 0.13
TD	10	S	V	14.10 ± 0.19	14.49 ± 0.08	14.62 ± 0.08
TD	3	A	A	13.94 ± 0.13	14.31 ± 0.13	14.57 ± 0.10
TD	5	A	A	14.00 ± 0.21	14.40 ± 0.12	14.59 ± 0.10
TD	3	V	V	14.07 ± 0.20	14.48 ± 0.07	14.61 ± 0.09
TD	5	V	V	14.29 ± 0.12	14.56 ± 0.11	14.73 ± 0.09
RG	1	S	S	12.10 ± 0.81	13.36 ± 0.37	13.80 ± 0.34
RG	3	S	A	13.20 ± 0.48	13.99 ± 0.27	14.27 ± 0.22
RG	5	S	A	13.62 ± 0.33	14.05 ± 0.33	14.39 ± 0.27
RG	10	S	A	13.95 ± 0.20	14.21 ± 0.17	14.51 ± 0.12
RG	3	S	V	13.24 ± 0.44	14.04 ± 0.21	14.35 ± 0.15
RG	5	S	V	13.74 ± 0.28	14.31 ± 0.15	14.59 ± 0.10
RG	10	S	V	14.15 ± 0.15	14.48 ± 0.08	14.69 ± 0.09
RG	3	A	A	13.73 ± 0.32	14.29 ± 0.18	14.53 ± 0.10
RG	5	A	A	14.09 ± 0.19	14.35 ± 0.11	14.60 ± 0.09
RG	3	V	V	13.64 ± 0.28	14.25 ± 0.18	14.50 ± 0.13
RG	5	V	V	14.17 ± 0.18	14.54 ± 0.14	14.73 ± 0.11

compared similarly as done with table 1, with the value 14.13 ± 0.18 (line 5, RG) as an exception (p-value is 0.045). With the used value for parameter τ, both TD and RG selected the action with the highest value with a probability of ~ 0.7 on average and in the long run. Therefore, the differences between epsilon-greedy and softmax are in the actions with the lower values. With the initialized weights, the single agent and the committees started with a total reward of ~ 2. In Fig. 1, the single agent is compared to the committees with a single policy for training and a Joint Average policy (left column) or a Joint Majority policy (right column) for simulation, and for different committee sizes. The results with TD, softmax and RG, epsilon-greedy are similar to the results in this figure and are omitted.

5.1.1 Discussing the Results

The single agent (with $M = 1$, 100 test runs, $25,000,000$ episodes) with the epsilon-greedy exploration strategy (ϵ-strategy) reaches a final total reward (see table 1) of 13.90 with TD, averaged over 10 simulations per starting state and per maze, and 13.64 with RG. With the softmax action selection strategy (τ-strategy) and TD, the single agent has 14.12, and 13.87 with RG (see table 2). Comparing the total rewards between TD and RG, RG learns slower with the first iterations, but almost reaches the same performance as TD in the long run. This can be observed in the tables and in Fig. 1. The results with τ-strategy are better than the results with ϵ-strategy, both with TD and with RG.

All committees ($M > 1$) with several training and simulation policies surpass the total rewards of the single agent. Within the committees, all committees with

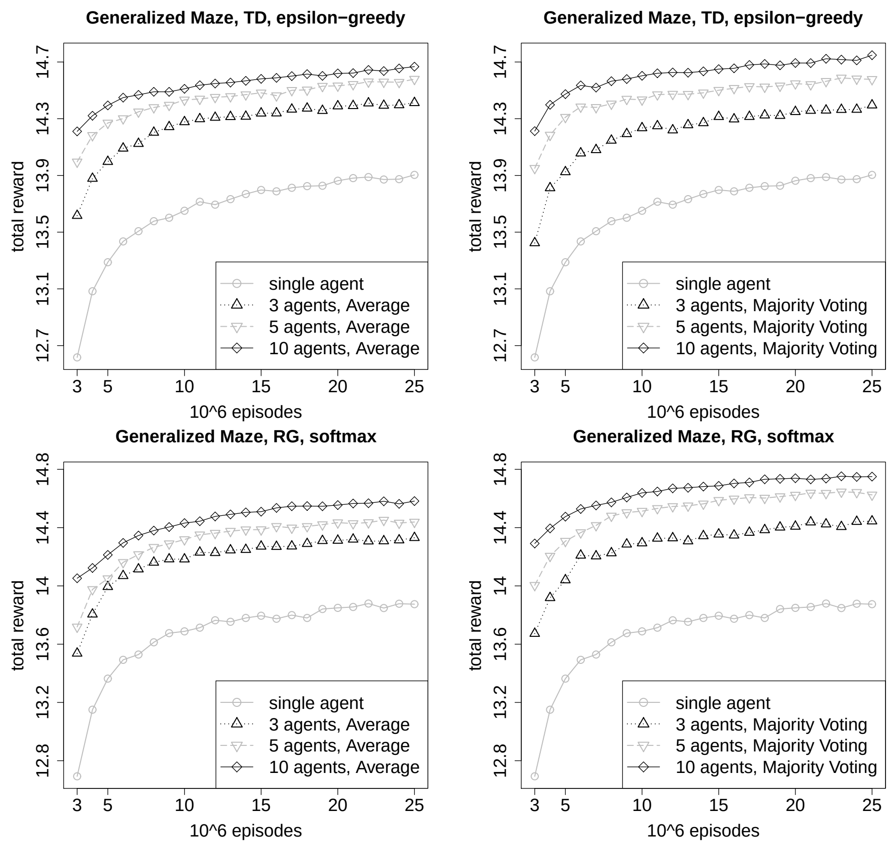

Fig. 1 Empirical results with Generalized Maze, starting from $3,000,000$ episodes. The total reward is averaged over 100 test runs for the single agent and 50 runs for the committees. Graphs in the left column are with the Joint Average policy and graphs in the right column are with the Joint Majority Voting policy.

joint decisions for training (20 runs) outperform the committees with single decisions for training (50 runs). In some cases, the committees with joint decisions and five agents are better than the committees with single decisions and ten agents. The best results are with the τ-strategy, where a committee of five agents with the Joint Majority Voting policy for training and for simulation reached a total reward of 14.73 with TD and with RG. In comparison to this, a committee of ten agents with the single policy for training and the Joint Majority Voting policy for simulation reached a total reward of 14.62 with TD and 14.69 with RG. The best committees have a better strategy than the breadth-first search method, that reaches a total reward of 14.38 (see section 5.1).

Further, the results of the committees with the single policy for training and with different committee sizes (parameter M) and up to $25,000,000$ episodes are in Fig. 1. The graphs show first, that the larger the committee size the higher the total reward, and second, that an increase in the total reward of a single agent is an increase in the total reward of a committee. For some values of iterations T, it can be observed that a committee of M agents with T training episodes is better than a single agent with $T \cdot M$ training episodes. All committees with three agents and $5,000,000$ episodes are better than the single agent with $15,000,000$ episodes. The same holds for five agents and $25,000,000$ episodes.

5.2 SZ-Tetris

In stochastic SZ-Tetris, an agent places tetrominos on top of already placed tetrominos and tries to complete rows. The board has a width of 10 and a height of 20. At one step either an S-shaped piece or a Z-shaped piece randomly appears with equal probabilities. The agent then chooses a rotation (lying or standing) and a horizontal position of the piece within the board. If the piece does not fit on the board, then the game ends. SZ-Tetris with its two tetrominos, out of the seven available, is considered to be the hard core of Tetris. Deterministic SZ-Tetris with an alternating sequence of SZ pieces has been proven to have a maximum episode length of $69,600$ [5]. In the benchmark-type setting as introduced in [15], the agent gets 1 point for completing one row and 2 points for completing two rows at once. Measured is the sum of completed rows until the board is full. The best learning agent uses the Bertsekas & Ioffe feature set and learns to weight these feature with the Cross-Entropy method [15]. The best score with this setting is 133. Up to now, Bellman error minimizing methods like TD, RG, TDC, GDT2, etc. were reported to reach a score of ~ 0 at best.

Only states with a full board are considered to be terminal states. A non-terminal state does not have completed rows, i.e. if rows are completed, then they are immediately removed. The starting set ST only includes the empty board. State values are learned with $TD(\lambda)$, $\lambda = 0.5$. On each step, the agent receives the following immediate reward:

$$r(s) = exp(-\beta \cdot \text{number of holes in s})$$

with $\beta = 1/33$ and counting the number of holes below the pieces (column-wise) in a state s. The discounting value is set to $\gamma = 0.95$. With lower values, the agent reached worse scores or even reached a score close to zero ($\gamma \leq 0.7$). The maximum total reward (upper bound) an agent can obtain is $1/(1-0.95) = 20$ (geometric series). With these smooth rewards, an agent tries to minimize the number of holes up to a terminal state, being the full board. An agent that directly uses the reward function for the decisions, instead of learning from the reward, has a score of ~ 17. A $2-layer$ MLP is used as a FA with 5 neurons in the hidden layer and 330 input neurons. The s-shaped hyperbolic tangent function is the activation function for the hidden layer. In the output layer, the activation function is the linear function. Out of 330 input neurons, 180 are the discretized height differences (i.e. 9 height differences with a maximum height of 20 each) and 150 are the discretized number of holes. On the rare occasion, a board has more than 150 holes, it is considered to be 150 holes as well. With this input coding exactly 10 neurons (nine for the height differences and one for the number of holes) are active (1) and 320 are inactive (0) in each state. Learning rates are optimized for a single agent ($M = 1$), $\alpha = 0.001$.

The results for a single agent and committees with three ($M = 3$) to ten ($M = 10$) agents, with the epsilon-greedy exploration strategy with $\epsilon = 0.04$,

Table 3 Empirical results with SZ-Tetris. Evaluated is a score, i.e. the number of cleared lines, measured in a benchmark setting after $0.5 \cdot 10^6$ to $3 \cdot 10^6$ training episodes (fourth to sixth column). The single agent ($M = 1$) reached a score of 129.70 ± 11.40 at $5 \cdot 10^6$ episodes. 100 repeated tests in the simulation phase. 100 runs for the single agent, 50 runs for the committees. Second column is the policy used for the training phase (S = Single, A = Average and V = Majority Voting), third column is the policy used for the simulation phase, i.e. for calculation of the score.

M	π train	π sim.	score $0.5M$	score $1M$	score $3M$
1	S	S	88.40 ± 22.10	108.60 ± 19.70	125.80 ± 13.90
3	S	A	114.60 ± 17.60	131.20 ± 11.30	141.40 ± 8.50
5	S	A	127.60 ± 12.30	135.20 ± 8.50	144.10 ± 6.60
10	S	A	130.40 ± 9.80	142.30 ± 7.70	146.20 ± 6.00
3	S	V	83.40 ± 18.60	106.60 ± 13.10	129.30 ± 10.80
5	S	V	109.90 ± 12.70	124.60 ± 10.90	137.80 ± 8.80
10	S	V	119.20 ± 11.30	133.60 ± 8.90	141.20 ± 5.50

and weight initializations in an interval $a = [-0.2, 0.2]$ are in table 3. With higher values of ϵ, i.e. higher exploration rates, the agent has more trouble filling the lines and ends up with a worse set-up of pieces. In contrast, with no exploration, the agent reaches a score close to zero. 100 agents with randomly initialized weights were trained in parallel. The committees with the single decisions during learning (π train is 'S') were formed by randomly selecting M out of these agents, 50 runs. An episode ends by reaching the full board. The differences of the scores between $1,000,000$ episodes (column five) of the committees with three agents ($M = 3$) and $3,000,000$ episodes (column six) of the single agent are statistically significant (p-values below 0.02). Furthermore, the differences of the total rewards between $1,000,000$ episodes of the committees with five agents ($M = 5$) and $5,000,000$ episodes (see table description) of the single agent are statistically significant. With the initialized weights, the single agent and the committees started with a score of ~ 0. In Fig. 2, the single agent is compared to the committees with a single policy for training and a Joint Average policy (left) or a Joint Majority policy (right) for simulation, and for different committee sizes.

5.2.1 Discussing the Results

Several learning agents were trained with a Bellman error minimizing method (TD with eligibility trace) and a nonlinear function approximator, specifically a 2-layer MLP. A single agent almost reaches, and some of the committees surpass the best reported performance [15]. The single agent (with $M = 1$, 100 test runs, $5,000,000$ episodes) has a final score (see table 3) of 129.70, averaged over 100 repeated simulations. Likewise, the agents get better than their teacher, i.e. they do more rewarding decisions than they could reach by using the reward function for their decisions alone (17, see section 5.2). The ascent in the score between $3,000,000$ episodes (column six) and $5,000,000$ episodes is considerably small. Thus, it is expected that the performance of the single agent will not increase with further training episodes.

Virtually all committees ($M > 1$, 50 test runs), with several simulation policies, have higher scores than the single agent. Committees with a Joint Average policy perform significantly better than committees with a Joint Majority policy. This

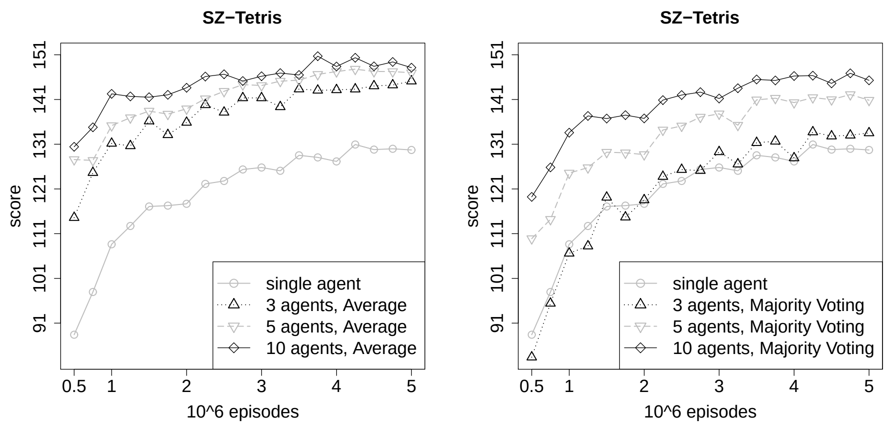

Fig. 2 Empirical results with SZ-Tetris, starting at 500,000 episodes. The score is averaged over 100 test runs for the single agent and 50 runs for the committees. Left graph is with the Joint Average policy, right graph is with the Joint Majority Voting policy.

may be because SZ-Tetris has up to 17 actions (or next states) to choose from, and the votes were widely distributed during the very first training iterations. The results of the committees with a joint policy for training are omitted. The scores they reached were similar to the scores of the committees with a single policy for training. Our best guess is that the gain of learning from joint decisions is not as large as simply adding agents to the committee and then doing joint decisions. The best-performing committee has ten agents and used the Joint Average policy for the simulation. It reached a final score of 142.30 (1,000,000 episodes), 146.20 (3,000,000 episodes) and ~ 150 (5,000,000 episodes, see Fig. 2). This committee outperforms the current best learning-agent, having a score of ~ 133.

The results of the committees with the single policy for training with different committee sizes, and a single agent, for up to 5,000,000 episodes are in Fig. 2. An increasing committee size (parameter M) results in better scores. For $M = 10$, the score is ~ 150. The committee with three agents with the Joint Average policy (left graph) is better at 1,000,000 episodes (131.20) than the single agent at 3,000,000 episodes (125.80). Same with five agents and 5,000,000 episodes. With the Joint Majority policy (right graph), the committee is better than the single agent with $M \geq 5$ (see explanation above).

6 Conclusion

In our analysis, we discussed the improvements on learning with unstable estimations of the value functions in the RL domain with various ensemble techniques, based on Breiman's theorem. We have shown analytically that a committee with joint decisions has a lower or equal average value estimation error than a single estimator and thus can perform more rewarding decisions in a control problem. We described a meta-algorithm to learn state- or state-action values in a Neural Network Ensemble. The algorithm applies for a control problem or for a prediction problem and uses model-free RL methods to update the FA-weights. Empirical evaluations confirm our analytical results in two environments with large states space and dynamics, i.e. generalized maze and stochastic SZ-Tetris. The committees with $M > 1$ agents and T training episodes had a higher total reward (generalized maze) or a higher score (SZ-Tetris) than a single agent with $T \cdot M$ training episodes. To date, our committees with three to ten agents outperforms the best-known SZ-Tetris learning-agent.

References

1. Baird LCI (1995) Residual Algorithms: Reinforcement Learning with Function Approximation. In: Prieditis A, Russell SJ (eds) Proc. 12th Internat. Conf. on Machine Learning (ICML'95), Morgan Kaufmann, pp 30–37
2. Bertsekas DP, Tsitsiklis JN (1996) Neuro-dynamic programming. Athena Scientific, Belmont, MA
3. Bishop CM (2006) Pattern Recognition and Machine Learning. Springer, New York
4. Breiman L (1996) Bagging Predictors. Machine Learning 24(2):123–140
5. Burgiel H (1997) How to Lose at Tetris. The Mathematical Gazette 81(491):194–200
6. Dietrich C, Palm G, Riede K, Schwenker F (2004) Classification of bioacoustic time series based on the combination of global and local decisions. Pattern Recognition 37(12):2293–2305
7. Faußer S, Schwenker F (2011) Ensemble Methods for Reinforcement Learning with Function Approximation. In: Sansone C, Kittler J, Roli F (eds) Proc. 10th Internat. Workshop on Multiple Classifier Systems (MCS 2011), Springer, Lecture Notes in Computer Science, vol 6713, pp 56–65
8. Hady MFA, Schwenker F (2010) Combining Committee-Based Semi-Supervised Learning and Active Learning. J Comput Sci Technol 25(4):681–698
9. Hans A, Udluft S (2010) Ensembles of Neural Networks for Robust Reinforcement Learning. In: Draghici S, Khoshgoftaar TM, Palade V, Pedrycz W, Wani MA, Zhu X (eds) ICMLA, IEEE Computer Society, pp 401–406
10. Li L (2008) A worst-case comparison between temporal difference and residual gradient with linear function approximation. In: Cohen WW, McCallum A, Roweis ST (eds) Proc. 25th Internat. Conf. on Machine Learning (ICML'08), ACM, ACM International Conference Proceeding Series, vol 307, pp 560–567
11. Scherrer B (2010) Should one compute the Temporal Difference fix point or minimize the Bellman Residual? The unified oblique projection view. In: Fürnkranz J, Joachims T (eds) Proc. 27th Internat. Conf. on Machine Learning (ICML'10), Omnipress, pp 959–966
12. Schwenker F, Kestler HA, Palm G (2001) Three learning phases for radial-basis-function networks. Neural Networks 14(4-5):439–458
13. Sutton RS, Barto AG (1998) Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA
14. Sutton RS, Maei HR, Precup D, Bhatnagar S, Silver D, Szepesvári C, Wiewiora E (2009) Fast gradient-descent methods for temporal-difference learning with linear function approximation. In: Danyluk AP, Bottou L,

Littman ML (eds) Proc. 26th Internat. Conf. on Machine Learning (ICML'09), ACM, ACM International Conference Proceeding Series, vol 382, p 125
15. Szita I, Szepesvári C (2010) SZ-Tetris as a Benchmark for Studying Key Problems of Reinforcement Learning. In: ICML 2010 Workshop on Machine Learning and Games
16. Tsitsiklis J, Roy BV (1997) An Analysis of Temporal-Difference Learning with Function Approximation. IEEE Trans on Automatic Control 42(5):674–690
17. Wiering MA, van Hasselt H (2008) Ensemble Algorithms in Reinforcement Learning. IEEE Transactions on Systems, Man, and Cybernetics, Part B 38(4):930–936
18. Zhou ZH (2009) When Semi-supervised Learning Meets Ensemble Learning. In: Benediktsson JA, Kittler J, Roli F (eds) MCS, Springer, Lecture Notes in Computer Science, vol 5519, pp 529–538
19. Zhou ZH (2011) Unlabeled Data and Multiple Views. In: Schwenker F, Trentin E (eds) PSL, Springer, Lecture Notes in Computer Science, vol 7081, pp 1–7