Consolidated actor–critic model for partially-observable Markov decision processes

I. Elhanany, C. Niedzwiedz, Z. Liu and S. Livingston

A method for consolidating the traditionally separate actor and critic neural networks in temporal difference learning for addressing partially-observable Markov decision processes (POMDPs) is presented. Simulation results for solving a five-state POMDP problem support the claim that the consolidated model achieves higher performance while reducing computational and storage requirements to approximately half those of the traditional approach.

Introduction: In temporal difference (TD) learning [1], an agent's goal is to maximise a long-term reward prospect, as provided by its environment. At each time step, the agent attempts to improve value estimation accuracy by minimising the differences in temporally separate measurements. The value of each state–action pair is dependent on the current policy, which maps states to actions. The policy, in turn, is improved based on the current value function approximation. By iteratively refining these two constructs, an optimal control policy is obtained [1].

In actor–critic (AC) algorithms, an agent typically consists of two building blocks: an actor that constructs the policy, and a critic that maintains a value function estimate for the actor's current policy. The value function maps state–action pairs to scalar values that represent the expected sum of future time-discounted rewards to be received by the agent when following the current policy. In partially-observable settings, explicit state information is not made available to the agent. Rather, the agent infers the state of its environment from a sequence of observations it receives.

For problems with large state and/or action spaces, nonlinear function approximators, notably artificial neural networks (ANNs), are often used to represent the actor and critic. When addressing partially-observable problems, recurrent neural networks have been proposed. In traditional AC learning-based systems, the actor and critic are realised using two separate ANNs. In previous work [2], an early attempt to combine the value and policy functions into a single network has been demonstrated; however, in that scheme, the agent has only two possible actions at every state, and the network learns action probabilities rather than actions, thus weakening scalability.

In this Letter, we present a method for consolidating these two networks into one, thereby substantially reducing storage and processing requirements, while supporting high-dimensional state and action spaces. Such consolidation is shown to yield substantial reduction in computational effort, storage requirements and convergence time.

Traditional actor–critic design: In the traditional AC design [3], the actor is approximated by a two-layer feedforward ANN, in which the hidden layer employs a nonlinear mapping, $f_i(t) = f(x_i)$, such as the sigmoid function, where $x_i(t)$ is the sum of all weighted activations going into neuron i and $y_i(t)$ is its output, all at time t. To facilitate state-inference, the output of the hidden layer at time t is fed back as input at time $t+1$, in accordance with an Elman network structure [4]. Similarly, the critic is approximated by an Elman network, in which the output layer activation function is typically linear.

The network learning process is governed by stochastic gradient descent. In particular, at time t, the error of the critic network is give by

$$E_c(t) = \frac{1}{2}([r(t) + \gamma J(t)] - J(t-1))^2 \tag{1}$$

while the error of the actor network is

$$E_a(t) = \frac{1}{2}(J(t) - R^*)^2 \tag{2}$$

where R^* is the optimal return, which is dependent on the problem definition. The expected return is expressed as the general cost function, $J(t)$, which is to be maximised by the agent. Specifically,

$$J(t) = r(t+1) + \gamma r(t+2) + \gamma^2 r(t+3) + \cdots \tag{3}$$

where $r(t)$ is the immediate reward and γ is the time-discounting factor ($0 \leq \gamma < 1$). Finally, following the procedure of gradient descent, the networks' mappings are updated after each time step by adding a scaled negative of the error gradient to their weights.

Consolidated actor–critic model: This Letter proposes a consolidated actor–critic model (CACM), for which the goal is to eliminate redundant implicit environmental modelling information. In order for the actor to optimally select actions and the critic to correctly evaluate a state, both must be able to accurately model the underlying dynamics of the agent's environment. The CACM is designed to eliminate this duplicated effort by embodying the actor and critic in a single ANN.

Let $nn_{CACM}(\cdot)$ represent the ANN of the combined actor and critic. Given a state vector $\boldsymbol{s}(t)$ and action vector $\boldsymbol{a}(t)$, nn_{CACM} is defined such that $nn_{CACM}(\boldsymbol{s}(t), \boldsymbol{a}(t)) = (J(t); \boldsymbol{a}(t+1))$, where $J(t)$ is the estimated value of the given state-action pair (obtained from the critic), and $\boldsymbol{a}(t+1)$ is the subsequent action to be taken by the agent (produced by the actor).

The error at the critic node output is the squared Bellman error [3]. However, the error at the action outputs is now defined by the gradient of the estimated value $J(t+1)$ with respect to the action vector $\boldsymbol{a}(t+1)$ selected by the CACM network at time t. Specifically,

$$\begin{aligned} \boldsymbol{e}_a(t) &= \alpha \nabla_{\boldsymbol{a}(t+1)} J(t+1) \\ &= \alpha\left(\frac{\partial J(t+1)}{\partial a_1(t+1)}, \ldots, \frac{\partial J(t+1)}{\partial a_d(t+1)}\right) \end{aligned} \tag{4}$$

where α is a scaling constant and d is the number of elements in the action vector $\boldsymbol{a}$. Combining the error for each component of the selected action, the overall actor error is

$$E_a(t) = \frac{1}{2}\left[\sum_{i=1}^{d} e_{a_i}^2(t)\right] \tag{5}$$

where $e_{a_i}(t)$ is the ith element of the action error gradient $\boldsymbol{e}_a(t)$.

In finding the gradient of the estimated value $J(t+1)$ with respect to the previously selected action $\boldsymbol{a}(t+1)$, the direction of change in action, which will improve the expected return at time step $t+1$, is obtained. Thus, by incrementally improving actions in this manner, an optimal policy can be achieved. The error to be minimised for the entire CACM network is now defined as $E(t) = E_c(t) + E_a(t)$.

Simulations: The problem chosen to be solved is optimisation of a five-state partially-observable Markov decision process (see Fig. 1). The agent begins each episode in state A, and an episode terminates once the agent enters state E. From each non-terminal state, two deterministic actions are available: 'down' and 'right'. For discounting factor $0 < \gamma < 1$, the optimal policy in terms of state transitions is $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$. To simulate partial observability, states B and C deliver identical observations to the agent. We find that such a problem conveys the intuition behind the proposed control method, while remaining simple enough to prevent application-specific heuristics, which would narrow the applicability of our results.

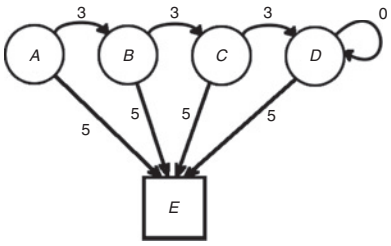

Fig. 1 *Partially-observable Markov decision process to be optimised*

Each state, except state E, is non-terminal. States B and C appear identical to agent (hence, partial observability). Each action is indicated by line with number which is associated transition reward

Three AC controllers are applied to the problem. First, the proposed consolidated actor–critic model (CACM) is labelled 'CACM-20' and implemented using the nonlinear hidden layer activation function tanh(·) and 20 hidden neurons, all of which are fed back as network input; the network learning rate is 0.01. Next, two controllers are implemented, which are similar in function to 'CACM-20', with the exception of hosting separate networks for the actor and critic functionalities. The activation function of the actor's output layer is tanh(·) (rather than a simple summation as in the critic network and in the output layer of 'CACM-20'), and the actor 5 network learning rate is 0.005. One of these controllers has ten hidden neurons per network

(20 total) and is thus labelled 'traditional AC-10' while the other controller has 20 hidden neurons per network (40 total) and is labelled 'traditional AC-20'. Finally, for all three agents, the actor learning step size $\alpha = 0.001$.

To learn the optimal policy, some amount of exploration must be included during training. Each of the tested agents selects a random action with probability $\varepsilon = 0.1$, otherwise it selects the greedy action given by the actor. The greedy action is defined as the one that has the highest expected return. In all simulations, the return discount parameter chosen was $\gamma = 0.9$. Also, a trial is defined as the number of training episodes required for convergence (i.e. to reach the optimal policy).

For the experiment, 100 trials were performed. The output errors (Bellman and action) are recorded after each discrete time step, averaged across the 100 trials, and then squared and multiplied by 0.5. The squared Bellman error of the three actor-critic agents is shown in Fig. 2. In addition, a running average filter is applied to the squared action error according to: $D_{avg}(t) = 0.95D_{avg}(t-1) + 0.05D_{raw}(t)$, where $D_{avg}(0) = 0$, $D_{avg}(t)$ is the running-average squared action error of time step t, and $D_{raw}(t)$ is the actual error. The squared action error of the three tested agents is also shown in Fig. 2. Finally, based on the 100 performed trials, the mean number of training episodes until convergence, required by each controller, is listed in Table 1.

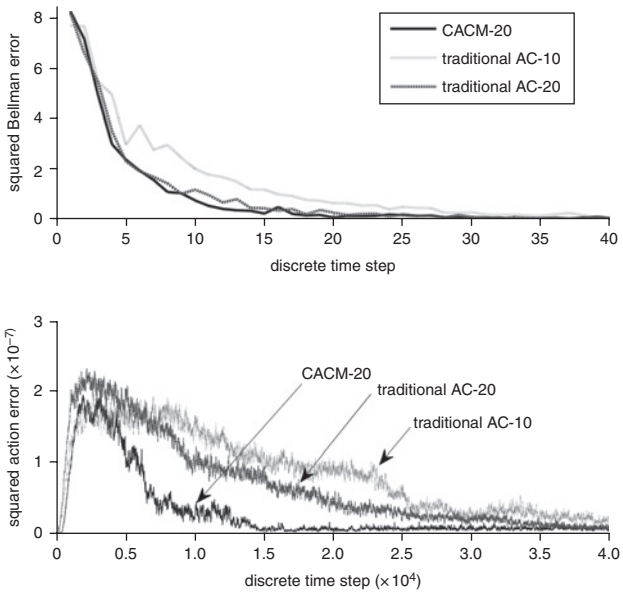

Fig. 2 *Squared Bellman error averaged over 100 independent trials (upper plot), and smoothed squared action error averaged over 100 independent trials (lower plot)*

Table 1: Mean training episodes required to learn optimal control policy, derived from 100 independent trials

Method	Mean episodes to convergence
CACM-20	41720
Traditional AC-10	104800
Traditional AC-20	102600

As reflected by the results, consolidating the actor and critic networks improves the mean convergence time by more than half while equally reducing computation and memory requirements. This supports the assertion that the traditionally separate neural networks perform at least partially redundant environmental modelling, and therefore combining the networks can both improve the convergence rate and reduce the required resources.

Conclusions: The consolidated actor–critic model, for eliminating redundant implicit environmental modelling in actor–critic temporal difference learning systems using artificial neural networks, is presented. Simulation results for learning the optimal policy of a five-state POMDP indicate substantially improved performance compared to that of a traditional, two network actor–critic approach. Not only is the mean number of training episodes leading to convergence more than halved, but also the required memory and computational resources are approximately halved. The proposed control method is particularly applicable to resource-constrained decision-making platforms, such as embedded systems or mobile robotics.

13 May 2008
Electronics Letters online no: 20081346
doi: 10.1049/el:20081346

I. Elhanany, C. Niedzwiedz and S. Livingston (*Department of Electrical Engineering and Computer Science, University of Tennessee, 414 Ferris Hall, 1508 Middle Way Dr., Knoxville, TN 37996-2100, USA*)

E-mail: itamar@ece.utk.edu

Z. Liu (*Yahoo Search, Inc.*)

References

1 Sutton, R.S., and Barto, A.G.: 'Reinforcement learning: an introduction' (MIT Press, Cambridge, MA, USA, 1998)
2 Mizutani, E., and Dreyfus, S.: 'Two stochastic dynamic programming problems by model-free actor-critic recurrent-network learning in non-Markovian settings'. Proc. IEEE Int. Joint Conf. on Neural Networks, 2004, Vol. 2, pp. 1079–1084
3 Si, J., Yang, L., and Liu, D.: 'Direct neural dynamic programming', *in* Si, J., Barto, A.G., Powell, W.B., Wunsch, D. (Eds.): 'Handbook of learning and approximate dynamic programming' (Institute of Electrical and Electronics Engineers, 2004, pp. 125–151)
4 Elman, J.L.: 'Finding structure in time', *Cogn. Sci.*, 1990, **14**, (2), pp. 179–211