ESANN 2010 proceedings, European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning. Bruges (Belgium), 28-30 April 2010, d-side publi., ISBN 2-930307-10-2.

The Markov Decision Process Extraction Network

Siegmund Duell[1,2], Alexander Hans[1,3], and Steffen Udluft[1]

1- Siemens AG, Corporate Research and Technologies, Learning Systems,
Otto-Hahn-Ring 6, D-81739 Munich, Germany
{duell.siegmund.ext|alexander.hans.ext|steffen.udluft}@siemens.com

2- Berlin University of Technology, Machine Learning,
Franklinstr. 28-29, D-10587 Berlin, Germany

3- Ilmenau University of Technology, Neuroinformatics and Cognitive Robotics Lab,
P.O.Box 100565, D-98684 Ilmenau, Germany

Abstract. This paper presents the Markov decision process extraction network, which is a data-efficient, automatic state estimation approach for discrete-time reinforcement learning (RL) based on recurrent neural networks. The architecture is designed to model the minimal relevant dynamics of an environment, capable of condensing large sets of continuous observables to a compact state representation and excluding irrelevant information. To the best of our knowledge, it is the first approach published to automatically extract minimal relevant aspects of the dynamics from observations to model a Markov decision process, suitable for RL, without requiring special knowledge of the regarded environment. The capabilities of the neural state estimation approach are evaluated using the cart-pole problem and standard table-based policy iteration.

1 Introduction

Reinforcement learning (RL) [1] describes a class of methods to solve optimal control problems. Usually the problem is assumed to be a Markov decision process (MDP) $M := (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ with a state space $\mathcal{S}$, a set of possible actions $\mathcal{A}$, the system dynamics, defined as state transition probability distribution $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, which gives the probability of reaching state s' by executing action a in state s, and a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, which determines the reward for a given transition. In many real world problem settings the current state s_t is hidden and an observation $z_t \in \mathcal{Z}$ is the only source of information about the considered environment. The mapping of states and actions to observations is described by the observation function $\mathcal{O} : \mathcal{S} \times \mathcal{A} \to \mathcal{Z}$. To apply standard RL algorithms to a partially observable Markov decision process (POMDP) $M' := (\mathcal{S}, \mathcal{Z}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{O})$, a state estimator is required to provide a Markovian representation of the environment.

The Markov decision process extraction network (MPEN) is based on the approach presented in [2], where a hidden layer of a dynamic consistent recurrent neural network is used as state representation. We extend this work to automatically select the required aspects of the considered dynamics and therefore only model the minimal required MDP. Previous approaches were only able to

model the minimal dynamics in terms of redundant information including all dynamical aspects of the regarded set of observations.

The paper is divided into four parts. Subsequent to the introduction Sec. 2 gives a short overview of state estimation from a history of observations using neural networks. In Sec. 3 we describe in detail the architecture of the MPEN, where Sec. 4 underlines our theoretic findings by an application to the well known cart-pole problem. A conclusion and further prospects are given in Sec. 5.

2 State Estimation from a History of Observations

RL problems can be viewed as an agent interacting with an environment by performing different actions and observing a state transition as well as a reward signal. As described in the introduction section, the original state of an environment might not be available to the agent. Therefore the only source of information about the system's state is provided by an observation, which often does not satisfy the Markov property.

A naïve approach to model a Markovian state is to accumulate a sufficient number of prior time slices into a single state. Although the resulting state space fulfills the Markov property, in many cases it is not applicable due to its high dimensionality.

To overcome this problem, a neural bottleneck network (e.g., [3]) can be used to condense all relevant information into a compact state representation. This approach allows to reduce all redundant information and decorrelates the state variables. The bottleneck network approach can further be rewritten as a dynamic consistent recurrent neural network [4] to account for the time invariance of sequential data and therefore avoid overfitting by reducing the number of free parameters. Recurrent state estimators have shown remarkable results in various industrial applications and are therefore preferably used for sequential data [5]. For discrete time grids ($t = 1, \ldots, T, T \in \mathbb{N}$) the dynamics can be described as

$$s_{t+1} = f(s_t, z_t, a_t), \tag{1}$$

$$z_t = g(s_t). \tag{2}$$

The dynamic consistent recurrent neural network (RNN) approach was proven to model a Markovian state space [2]. In either using an RNN or a neural bottleneck architecture, the goal is to optimize

$$\sum_{t=1}^{T} (\tilde{z}_t - z_t)^2 \rightarrow \min_{f,g}, \tag{3}$$

where $\tilde{z}_t$ is the neural approximation of the measured z_t. As the complete observation vector z_t needs to be predicted, the modeled dynamics can well be much richer than the minimal dynamics required for RL. Therefore, the applicability of these approaches can be limited in case of industrial problem settings with high dimensional observation spaces containing various measurements of possibly irrelevant dynamic subsystems. In such scenarios manual preselection

of relevant observables may be required, demanding expert knowledge, and may lead to a loss of relevant information.

The MPEN approach is based on the idea of RNNs but only models relevant aspects. To accomplish this goal, several changes to the standard dynamic consistent network, described above, have been made. First, the network is split into past and future subnetworks to match

$$s_{t+1} = \begin{cases} f_{\text{past}}(s_t, z_t, a_t), & t < 0, \\ f_{\text{present}}(s_0, a_t), & t = 0, \\ f_{\text{future}}(s_t, a_t), & t > 0. \end{cases} \tag{4}$$

This allows us to use the reward signal itself as target instead of the naïve target of predicting all observables:

$$r_t = g(s_t, a_t),\ t \geq 0. \tag{5}$$

Note that the current state s_t and the applied action a_t are sufficient to describe the expected value of all relevant reward functions since all information about the successor state s_{t+1} must be included in these two arguments. Through this target the objective of the neural network becomes

$$\sum_{t=1}^{T} (\tilde{r_t} - r_t)^2 \rightarrow \min_{f,g}, \tag{6}$$

which causes the network to model only the required dynamical aspects in s_t. The network accumulates all information required for the Markov property from the sequence of past observations in the past network, while the future network optimizes the prediction of the state transitions. The exact specification of the MPEN architecture is described in the next section.

3 The Markov Decision Process Extraction Network

The MPEN is a recurrent neural network consisting of a past ($t < 0$) and a future ($t \geq 0$) subnetwork (fig. 1). There are no explicit targets in the past but required information can be extracted from the complete considered history. The past network is connected to the future network via an arbitrary neural structure (e.g., a weight matrix, a multi-layer perceptron). The future network uses the information provided by past network as well as future actions to learn the dynamics capable of predicting a sequence of future rewards. Note that the goal of the network is not a forecast of the sequence of actions which induced the state transitions, because action selection can be based on information which is not included in the set of observables or might be even unpredictable (e.g., due to random exploration).

As proven in [6], an RNN can be used to approximate an MDP by predicting all expected future successor states based on a history of observations. Because of the RNN structure each state must encode all necessary information to estimate

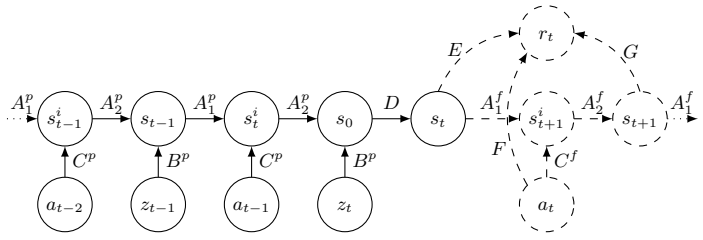

Fig. 1: The MPEN architecture. Clusters of neurons are depicted as circles. The connection arrows can be either a weight matrix or a multi-layer perceptron. Connectors with identical label (e.g., A_1^p) are identical due to shared weights. The dashed neural structures are only required for learning. Cluster s_t contains a Markovian state representation for recall. If the dimensionality of matrices A^p and A^f are equal, clusters s_0 and s_t can be merged.

a successor state accounting for the influence of an action. For this reason an RNN must be able to estimate the expected rewards for each future state, as a reward function can only use a state, action, and successor state as arguments. From this follows:

Theorem 3.1 *To estimate a state suitable for RL it is sufficient to model the dynamics predicting the reward for all future time slices.*

Based on this theorem the MPEN architecture is introduced to model the minimal dynamics of a regarded problem. In practice the horizon of forecasts is usually limited. For RL the discount factor γ introduces a natural limit of the significant horizon. Built on these observations the MPEN is derived from an RNN with separated past and future weight sets [6]. As mentioned above, the reward function arguments or the evaluated reward function is used as target. The major advantage of the MPEN over other RNN-based state estimators is the capability to model the minimal dynamics from a set of observables without manual selection of variables. A further advantage is the capability to extract the minimal state space in comparison to a network with dynamic consistent overshooting, where all considered input variables are predicted and therefore encoded in the state space.

4 Experiments and Results

To demonstrate the capabilities of the MPEN approach, a partially observable version of the cart-pole problem was used to calculate a policy using an MPEN for state estimation. The classical cart-pole problem [1] consists of a pole mounted on a cart and a limited track. The overall objective is to balance the pole as long as possible without hitting a boundary of the track or tilt the pole for more than 12°. We define the problem to be solved when balancing the pole for at least 100.000 steps. The reward signal is defined to be -1 when hitting any of the boundaries and 0 otherwise. The (Markovian) state of the

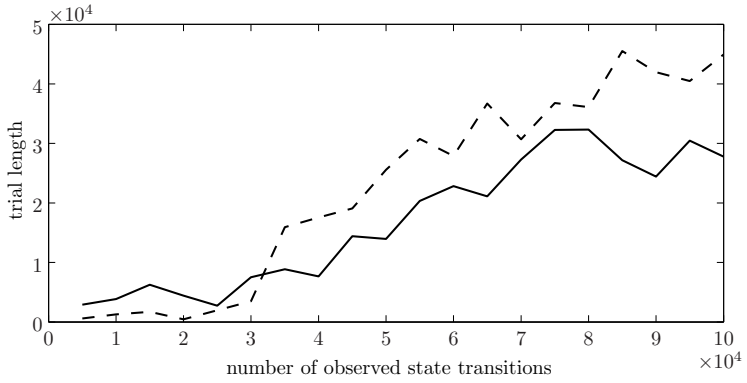

Fig. 2: Demonstration of the MPEN approach. Shown is the performance of policies based on the original state space (full line) and the neural state space as modeled by the MPEN from the partially observable state. The results were averaged over 25 trials.

cart-pole problem at any time step t is fully described by the position of the cart x_t, the velocity of the cart $\dot{x}_t$, the angle of the pole perpendicular to the cart α_t, and the angular velocity of the pole $\dot{\alpha}_t$. Possible influences by three discrete actions are to push the cart to the left or to the right by a constant force F or to apply no force at all.

In previous work, the cart-pole problem has been studied in various forms. The focus of this work is to demonstrate the capabilities of the MPEN architecture, i.e., the capability to model a suitable MDP for RL by predicting a future sequence of expected rewards. Therefore, we used the cart's position and velocity and the pole's angle as observables since this modification sufficiently violates the Markov property for table-based policy iteration (PI) to fail.

Policies based on a discretized four-dimensional neural state space generated by an MPEN (from observations of the three-dimensional partially observable state space) are compared to policies based on observations of the four-dimensional original state space. The MPEN with lags from $t \in \{-10, -9, \ldots, 25\}$ was trained from 25.000 partially observable transitions of the cart-pole problem. The available state variables were normalized (mean = 0, standard deviation = 1). No other transformation on the data was performed. To apply standard table-based PI, the continuous state representation of the original state space as well as the neural state space had to be discretized. Both state spaces were approximated by 4.096 discrete states each (8 bins per dimension).

PI was applied using a fixed discount factor $\gamma = 0.9$. After an initial 1.000 steps of random exploration a policy was calculated and subsequently used for ε-greedy exploration (random action with probability $\varepsilon = 0.15$). A new policy was generated every 1.000 state transitions.

As shown in fig. 2, the MPEN approach is able to model states which are suitable to be discretized and used for PI. Note that we chose PI because it is very sensitive to a violation of the Markov property and therefore adequate

to verify the reconstruction of the Markov property by the neural state space. Further it is notable that discretizing the state space of the cart-pole problem also violates the Markov property. Policies based on the neural state space outperform those based on the original state space since the neural network rescales and decorrelates the state variables. This results in a better estimation of the state transition probability distribution as well as the expected rewards based on the discretized states.

5 Conclusion and Outlook

A new RNN topology using separated weight sets for past and future states to avoid dynamic inconsistency was introduced. The topology uses the reward function or reward function arguments as targets and a set of observables as inputs. This is of remarkable value for data based RL, since an MDP including only relevant aspects of a considered observation space can be modeled. This approach reduces the possible loss of information due to manual input preselection and is a significant step towards autonomous neural applications in RL. The capabilities of the MPEN approach were successfully demonstrated on the cart-pole problem.

We are currently working on the combination of continuous RL methods with the MPEN such as neural fitted Q-iteration [7], neural rewards regression [8], and the recurrent control neural network [9].

References

[1] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[2] A.M. Schäfer and S. Udluft. Solving Partially Observable Reinforcement Learning Problems with Recurrent Neural Networks. In *Workshop Proc. of the European Conf. on Machine Learning*, pages 71–81, 2005.

[3] G.E. Hinton and R.R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, July 2006.

[4] H.G. Zimmermann, R. Grothmann, A.M. Schäfer, and C. Tietz. Identification and Forecasting of Large Dynamical Systems by Dynamical Consistent Neural Networks. In *New Directions in Statistical Signal Processing: From Systems to Brain*, pages 203–242. MIT Press, 2006.

[5] A.M. Schäfer, D. Schneegaß, V. Sterzing, and S. Udluft. A Neural Reinforcement Learning Approach to Gas Turbine Control. In *Proc. of the Int. Joint Conf. on Neural Networks*, pages 1691–1696, 2007.

[6] D. Schneegaß. *Steigerung der Informationseffizienz im Reinforcement-Learning*. PhD thesis, Luebeck University, 2008.

[7] M. Riedmiller. Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method. In *Proc. of the European Conf. on Machine Learning*, pages 317–328, 2005.

[8] D. Schneegaß, S. Udluft, and T. Martinetz. Neural Rewards Regression for Near-Optimal Policy Identification in Markovian and Partial Observable Environments. In *Proc. of the European Symposium on Artificial Neural Networks*, pages 301–306, 2007.

[9] A.M. Schäfer, S. Udluft, and H.-G. Zimmermann. The Recurrent Control Neural Network. In *Proc. of the European Symposium on Artificial Neural Networks*, pages 319–324, 2007.