ESANN'2007 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2007, d-side publi., ISBN 2-930307-07-2.

Neural Rewards Regression for Near-optimal Policy Identification in Markovian and Partial Observable Environments

Daniel Schneegaß[1,2], Steffen Udluft[1], and Thomas Martinetz[2]

1- Siemens AG, Corporate Technology, Learning Systems,
Otto-Hahn-Ring 6, D-81739 Munich, Germany

2- University of Luebeck, Institute for Neuro- and Bioinformatics,
Ratzeburger Allee 160, D-23538 Luebeck, Germany

Abstract. Neural Rewards Regression (NRR) is a generalisation of Temporal Difference Learning (TD-Learning) and Approximate Q-Iteration with Neural Networks. The method allows to trade between these two techniques as well as between approaching the fixed point of the Bellman iteration and minimising the Bellman residual. NRR explicitly finds a near-optimal Q-function without an algorithmic framework except Back Propagation for Neural Networks. We further extend the approach by a recurrent substructure to Recurrent Neural Rewards Regression (RNRR) for partial observable environments or higher order Markov Decision Processes. It allows to transport past information to the present and the future in order to reconstruct the Markov property.

1 Introduction

In industrial applications, Reinforcement Learning [1] problems are usually very complex, of high dimensionality, and just few data is available or the learning methods make high demands on time and space resources. Therefore, solving such tasks with powerful function approximators like problem-adapted architectural-designed Neural Networks (NN) [2] and data-efficient approximation techniques are obvious and promising strategies. Our Neural Rewards Regression approach is motivated by Kernel Rewards Regression (KRR) [3] for data-efficient RL. The RL problem is considered as a regression task fitting a reward function on the observed signals, where it is chosen from a hypothesis space, such that the Q-function can be gained out of the reward function. NRR is formulated similarly. The usage of shared weights with Back Propagation (BP) [4] allows us to restrict the hypothesis space appropriately.

In RL the main objective is to achieve a policy, that optimally moves an agent within an environment, which is defined by a Markov Decision Process (MDP) [1]. An MDP is generally given by a state space S, a set of actions A selectable in the different states, and the dynamics, defined by a transition probability distribution $P_T : S \times A \times S \to [0, 1]$ depending on the current state s, the chosen action a, and the successor state s'. The agent collects so-called rewards $R(s, a, s')$ while transiting. They are defined by a reward probability distribution P_R with the expected reward $\mathbf{E}R = \int_{\mathbb{R}} r P_R(s, a, s', r) dr, s, s' \in S, a \in A$.

Typically one is interested in maximising the discounting Q-function, defined by the Bellman Equation

$$Q^\pi(s,a) \quad = \quad \mathbf{E}_{s'}(R(s,a,s') + \gamma Q^\pi(s', \pi(s'))),$$

over the policy space $\Pi = (S \rightarrow A)$ for all possible states s and actions a, where $0 < \gamma < 1$ is the discount factor, s' the successor state of s, and $\pi \in \Pi$ the used policy. The best policy is the one using the actions maximising the (best) Q-function $Q^{\pi_{\text{opt}}}(s,a) = Q(s,a) = \mathbf{E}_{s'}(R(s,a,s') + \gamma \max_{a'} Q(s',a'))$, that is

$$\pi(s) \quad = \quad \arg\max_a Q(s,a).$$

We further define the Value Function over S as $V(s) = \max_a Q(s,a)$. For details we refer to Sutton and Barto [1]. In our setting, we assume a discrete set of actions, while the set of states is continuous and the dynamics probabilistic.

The remaining paper is arranged as follows. In section 2 we introduce NRR and explain briefly, what the core idea is. We describe, in which way the RL task is interpreted as a regression problem and how the trade-off between either the two learning procedures as well as the two optimality criteria is realised. In section 3 NRR is extended to RNRR to handle higher order MDPs. We show, how a Recurrent Neural Network (RNN) can be connected with NRR in a quite simple way and claim, that the architecture is indeed able to reconstruct the Markov property and increases NRR's applicability. Finally we consider its potential to solve the Optimal Control problem on two benchmark tasks in the context of Markovian as well as partial observable (PO) environments.

2 Neural Rewards Regression

We describe the Q-function for each possible action as Feed-Forward-Networks[1] $N_a(s) = Q(s,a)$. The reward function to be fitted is hence given as

$$R(s,a,s') \quad = \quad N_a(s) - \gamma \max_{a'} N_{a'}(s'),$$

where the max-operator has to be modeled by an appropriate architecture. Using the BP algorithm this problem setting approaches the minimum of the (squared) Bellman residual over all l observed transitions

$$L \quad = \quad \sum_{i=1}^{l} L_i^2 + \Omega(\boldsymbol{\theta}) = \sum_{i=1}^{l} \left(Q(\mathbf{s}_i, a_i) - \gamma V(\mathbf{s}_{i+1}) - r_i\right)^2 + \Omega(\boldsymbol{\theta}),$$

where $\boldsymbol{\theta}$ are the network parameters and Ω may be an appropriate regulariser. The observed r_i and $\mathbf{s}_{i+1}$ have to be unbiased estimates for their expectations [5] as the expectation operator is omitted. The error function's gradient $\frac{dL}{d\boldsymbol{\theta}} = 2\sum_{i=1}^{l} L_i \frac{d}{d\boldsymbol{\theta}} \left(Q(\mathbf{s}_i, a_i) - \gamma V(\mathbf{s}_{i+1})\right) + \frac{d}{d\boldsymbol{\theta}}\Omega(\boldsymbol{\theta})$ depends on the current Q-function

[1] A monolithic approach with one network $N(s,a) = Q(s,a)$ which takes the state and action as input, can also be used.

and the successor Value Function (fig. 1, the upper part of the network). But to obtain the fixed point of the Bellman Iteration, which provides better solutions in most applications [5], one retains $\mathbf{y}_i := r_i + \gamma V(\mathbf{s}_{i+1})$ and minimises iteratively

$$L \quad = \quad \sum_{i=1}^{l} \left(Q(\mathbf{s}_i, a_i) - \mathbf{y}_i\right)^2 + \Omega(\boldsymbol{\theta}),$$

until convergence of Q. Its gradient is then given as $\frac{dL}{d\boldsymbol{\theta}} = 2\sum_{i=1}^{l} L_i \frac{d}{d\boldsymbol{\theta}} Q(s_i, a_i) + \frac{d}{d\boldsymbol{\theta}}\Omega(\boldsymbol{\theta})$. By resubstituting $\mathbf{y}_i$ we further obtain the equation system

$$\sum_{i=1}^{l} \left(Q(\mathbf{s}_i, a_i) - \gamma V(\mathbf{s}_{i+1}) - r_i\right) \frac{d}{d\boldsymbol{\theta}} Q(\mathbf{s}_i, a_i) + \frac{d}{d\boldsymbol{\theta}}\Omega(\boldsymbol{\theta}) \quad = \quad 0,$$

whose solution is the fixed point of the (regularised) Bellman Iteration by construction. It can be seen, that both gradients differ only in their direction terms, but not in the error term. Blocking the gradient flow through the Value Function part of the network hence reconstructs the latter gradient. Therefore, in the backwards path of the BP algorithm, the error propagation between the R-cluster and the V'-cluster is multiplied by a constant ρ (see fig. 1). For $\rho = 1$ we have the classical Bellman residual minimisation, while the setting $\rho = 0$ leads to the Bellman Iteration. Any other compromise is a possible trade-off between these two error definitions [6]. The optimality is each defined as the solution of the more general equation system

$$\sum_{i=1}^{l} \left(Q(\mathbf{s}_i, a_i) - \gamma V(\mathbf{s}_{i+1}) - r_i\right) \frac{d}{d\boldsymbol{\theta}} \left(Q(\mathbf{s}_i, a_i) - \rho\gamma V(\mathbf{s}_{i+1})\right) + \frac{d}{d\boldsymbol{\theta}}\Omega(\boldsymbol{\theta}) \quad = \quad 0.$$

A closer look at the approach reveals the similarities to TD-Learning [1]. Combining $\rho = 0$ and an incremental learning scheme, NRR is identical to that classical approach. But the architecture allows us to apply the whole palette of established learning algorithms for NNs [7].

3 Recurrent Neural Rewards Regression

If e.g. in PO environments, the Markov property is not fulfilled, but can be reconstructed by using information from the past, it is desirable to memorise necessary observed information. One possibility is to apply RNNs [2] to simulate the underlying dynamical system (see fig. 1, lower part). This is typically done by internal states $\mathbf{x}_t, \mathbf{z}_t, t \in i-\tau, \cdots, i+1$ and their transitions recursively defined by

$$\begin{aligned} \mathbf{x}_t &= \tanh(F\mathbf{s}_t + J\mathbf{z}_{t-1}) \\ \mathbf{z}_t &= Ga_t + H\mathbf{x}_t \end{aligned}$$

using (linear) operators F, G, H, J. Applying a matrix M, mapping to the external state, the successor state has to be reconstructed by approaching

$$\|M\mathbf{z}_t - \mathbf{s}_{t+1}\|^2 \quad = \quad \min.$$

In doing so the internal states $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ can be used as inputs for NRR, which now may work on an MDP, if τ is appropriately large. The weighting of learning the dynamical system in comparison with solving the RL problem can be controlled by a factor μ similarly to ρ (see fig. 1, lower part).

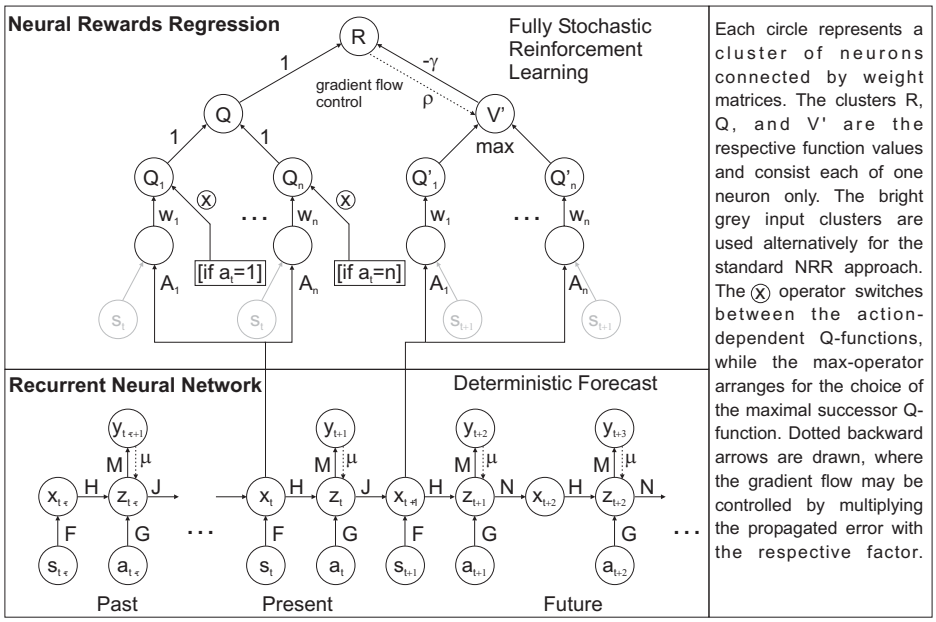

Fig. 1: The RNRR architecture.

Interestingly, the NRR architecture allows us to use only one RNN for both the Q-function and the Value Function. Note, that these RNNs are indeed able to approximate deterministic dynamical systems arbitrarily precisely [8], but this is not the case for stochastic dynamical systems. Fortunately, this is not a true restriction as the construction of the internal state can just be seen as the transformation into an appropriate feature space for a full stochastic Reinforcement Learner, built from past and present observations. In the deterministic case, this feature space is identical to the perfect description of all information determining the future, while in the general stochastic case the internal state has to be designed in order to forecast the expected future. Therefore one includes an autonomous future part, similarly to [2], where external states are forecasted without using observations from the Markov Process as

$$\mathbf{x}_t \quad = \quad \tanh(N\mathbf{z}_{t-1}), t > i+1.$$

This is indeed sufficient as the Markov property can be reconstructed by knowing the expected future states. By construction of a linear system of these expectations, we are able to prove the following accordant theorem (in the forthcoming full paper). Note, that that U and U' are Markov Processes, i.e. S joins states and action for a certain exploration.

Theorem 1 *Let $U = (S, P)$ be a countable stationary ergodic Markov Process of order $\tau + 1$. Let further be K a Recurrent Neural Network with unfolding at least of order τ and autonomous future extension of ν steps. Then $U' = (X, P')$ with $P' = P'(P, S, X)$ is a Markov Process of order 1, identical to U, i.e. $P(s^i_{j+1}|\mathbf{x}^i_j) = P(s^i_{j+1}|(s^i_{j-\tau}, \ldots, s^i_j))$, if $\nu \to \infty$ and K is able to reconstruct the deterministic part of the basic dynamics for all $s \in S$, i.e., K forecasts the expected future states arbitrarily precisely.*

4 Practical Results and Conclusion

We tested NRR and RNRR on two benchmark tasks, the Cart-Pole problem [1] and the so-called Wet-Chicken problem [9]. We examined both tasks regarding data-efficiency, their manageability in PO environments, and the influence of the parameter ρ. For comparison with the results presented in [10] and [5], we used the setting as described in [5] for the Cart-Pole problem, with the pole's angle $\boldsymbol{\theta}$ and its derivation. The reward function was shaped in order to offer more information about the nature of the problem to the learning method, that is, in transitions, where the pole is still balanced, we subtracted $\frac{|\theta|}{2\theta_{\max}}$ from the reward. Standard NRR is able to solve the Cart-Pole task perfectly (balancing for at least 3000 steps), stable, and reproducible with at most 1500 single observations (including restart transitions) and random exploration. We observed two thirds successful adaptations with only 80 single observations. For a detailed analysis see table 1. We further used RNRR with a four-dimensional internal state space, observing the angle θ only. Table 2 shows the linear correlation between the best linear combination of the internal state and the external variables. The reconstructibility of the full external state is a sufficient condition to recover the Markov property. Moreover, the Cart-Pole task is fully deterministic apart from tossing the dice for new starting states. Therefore, the setting of ρ is not very important. The most stable results could be achieved with $\rho = 0$, but $\rho > 0$ leads to successful results as well. Nevertheless, for (near-)deterministic tasks, it could be advisable to choose a greater ρ as the learning procedure is then well controllable and understood, the setting is close to a Supervised Learning scheme.

Table 1: Successful learning for Cart-Pole (standard and partial observable).

Std., Obs.	1500	1000	600	200	150	100	80	60	40
Succ. (%)	100	97	94	94	92	74	68	36	6
PO., Obs.	20000	10000	5000	2000	1000				
Succ. (%)	75	40	45	5	5				

In contrast, the Wet-Chicken problem is a quite simple, but probabilistic benchmark RL task. A canoeist has to paddle on a river until reaching a waterfall at maximal position $x = x_{\max}$. If he falls down, he has to restart. The reward increases linearly with the proximity to the waterfall. Simulated turbulences make the state transition probabilistic. These turbulences t themselves make a random walk, such that $t' = t + N(0, c)$. The canoeist has the possibility to drift,

hold the position, and row back. He has to row back as late as possible and as early as necessary. The task can be solved as well as with KRR [3] and achieves policies with nearly-optimal performance with 10000 observations for $\rho < 0.25$. While for ρ close to 0 the action changes from drift to row back directly, there is a wider hold-the-position area for greater ρ. For ρ close to 1, the policy tends to become too risky. In case of stochastic and unobservable turbulences, the two state variables x and t can be reconstructed quite well (see table 2).

Table 2: Best correlations between internal and external states.

Benchmark	**Vr.**	**Obs.**	**Corr.**	**Benchmark**	**Vr.**	**Obs.**	**Corr.**
Cart-Pole	θ	yes	0.9977	Wet-Chicken	x	yes	0.9996
	$\dot{\theta}$	no	0.9896		t	no	0.6552

5 Conclusion

The key aim of data-efficiency can be pointed out explicitly. As NRR is motivated by KRR, it works as a batch regression method and uses several regularisation techniques, hence results from Statistical Learning Theory can be applied. Further, the intrinsic Markovisation of the state space is an important technique to handle higher order MDPs. The parameter ρ, the recurrent substructure, and the usage of different learning methodologies realises a generalisation of certain RL methods for NNs in various aspects.

References

[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, 1998.

[2] H. G. Zimmermann, R. Grothmann, A. M. Schaefer, and Ch. Tietz. *New Directions in Statistical Signal Processing: From Systems to Brain*, chapter Identification and Forecasting of Large Dynamical Systems by Dynamical Consistent Neural Networks. MIT Press, 2006.

[3] Daniel Schneegass, Steffen Udluft, and Thomas Martinetz. Kernel rewards regression: An information efficient batch policy iteration approach. In *Proc. of the IASTED Conference on Artificial Intelligence and Applications*, pages 428–433, 2006.

[4] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Distributed representations. In *Parallel Distributed Processing*, pages 77–109, 1986.

[5] Michael G. Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, pages 1107–1149, 2003.

[6] Leemon C. Baird III. Residual algorithms: Reinforcement learning with function approximation. In *International Conference on Machine Learning*, pages 30–37, 1995.

[7] B. Pearlmutter. Gradient calculations for dynamic recurrent neural networks: A survey. *IEEE Transactions on Neural Networks*, 6(5):1212–1228, 1995.

[8] A. M. Schaefer and H. G. Zimmermann. Recurrent neural networks are universal approximators. In Stefanos Kollias, Andreas Stafylopatis, Wlodzislaw Duch, and Erkki Oja, editors, *ICANN 2006: 16th International Conference. Proceedings, Part I*, pages 632–640, 2006.

[9] Volker Tresp. The wet game of chicken. *Siemens AG, CT IC 4, Technical Report*, 1994.

[10] Martin Riedmiller. Neural fitted q-iteration - first experiences with a data efficient neural reinforcement learning method. In *Proceedings of the 16th European Conference on Machine Learning*, pages 317–328, 2005.