Improving Optimality of Neural Rewards Regression for Data-Efficient Batch Near-Optimal Policy Identification

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

[1] Information & Communications, Learning Systems
Siemens AG, Corporate Technology, D-81739 Munich, Germany
[2] Institute for Neuro- and Bioinformatics
University at Lübeck, D-23538 Lübeck, Germany
daniel.schneegass.ext@siemens.com

Abstract. In this paper we present two substantial extensions of Neural Rewards Regression (NRR) [1]. In order to give a less biased estimator of the Bellman Residual and to facilitate the regression character of NRR, we incorporate an improved, Auxiliared Bellman Residual [2] and provide, to the best of our knowledge, the first Neural Network based implementation of the novel Bellman Residual minimisation technique. Furthermore, we extend NRR to Policy Gradient Neural Rewards Regression (PGNRR), where the strategy is directly encoded by a policy network. PGNRR profits from both the data-efficiency of the Rewards Regression approach and the directness of policy search methods. PGNRR further overcomes a crucial drawback of NRR as it extends the accordant problem class considerably by the applicability of continuous action spaces.

1 Introduction

Neural Rewards Regression [1] has been introduced as a generalisation of Temporal Difference Learning (TD-Learning) and Approximate Q-Iteration with Neural Networks. It further offers the trade between minimising the Bellman Residual as well as approaching the fixed point of the Bellman Iteration. The method works in a full batch learning mode, explicitly finds a near-optimal Q-function without an algorithmic framework except Back Propagation for Neural Networks, and can, in connection with Recurrent Neural Networks, also be used in higher-order Markovian and partially observable environments. The approach is motivated by Kernel Rewards Regression (KRR) [3] for data-efficient Reinforcement Learning (RL) [4]. The RL problem is considered as a regression task fitting a reward function on the observed signals, where the regressor 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 [5,6] allows us to restrict the hypothesis space appropriately.

Policy Gradient (PG) methods [7] convince due to their capability to identify a near-optimal policy without using the loop way through the Value Function. They are known to be robust, convergence guarantees are given under mild conditions, and they behave well in partially observable domains. But PG methods usually base on Monte Carlo and hence high variances estimations of the discounted future rewards. Recent work addressing this problem are e.g. provided by Wang and Dietterich [8] as well as

Ghavamzadeh and Engel [9]. We introduce Policy Gradient Neural Rewards Regression (PGNRR), which, from the PG perspective, reduces the variance of the Value estimator and, from the Rewards Regression perspective, improves the considered problem class and enables generalisation over the policy space.

The remainder of this paper is arranged as follows. After a brief introduction to Reinforcement Learning, we recapitulate NRR and explain the underlying idea in sec. 3. 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. Subsequently (sec. 4) we generalise further by applying the new Bellman Residual minimisation technique to NRR. Finally we extend NRR by a policy network for direct policy search (sec. 5) and motivate an advancement in data-efficiency (sec. 6). Preliminary practical results (sec. 7) on a common benchmark problem and a real-world application are shown for the purpose of supporting the theoretical model.

2 Markov Decision Processes and Reinforcement Learning

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) [4]. 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, the chosen action, and the successor state. 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 $R = \int_{\mathbb{R}} r P_R(s, a, s', r) dr$, $s, s' \in S, a \in A$.

In most RL tasks one is interested in maximising the discounting Value Function

$$V^\pi(s) = \mathbf{E}^\pi_\mathbf{s} \left(\sum_{i=0}^{\infty} \gamma^i R \left(s^{(i)}, \pi(s^{(i)}), s^{(i+1)} \right) \right)$$

for all possible states s where $0 < \gamma < 1$ is the discount factor, s' the successor state of s, $\pi : S \to A$ the used policy, and $\mathbf{s} = \{s', s'', \ldots, s^{(i)}, \ldots\}$. Since the dynamics of the given state-action space cannot be modified by construction one has to maximise V over the policy space. One typically takes one intermediate step and constructs a so-called Q-function depending on the current state and the chosen action approaching

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

We define $V^* = V^{\pi_{\text{opt}}}$ as the Value Function of the optimal policy and Q^* respectively. This is the function we want to estimate. It is defined as

$$Q^*(s, a) = \mathbf{E}_{s'}(R(s, a, s') + \gamma V^*(s')) = \mathbf{E}_{s'} \left(R(s, a, s') + \gamma \max_{a'} Q^*(s', a') \right),$$

which is called the Bellman Optimality Equation. Therefore the best policy is apparently the one using the action $\pi(s) = \arg\max_a Q^*(s, a)$ maximising the (best) Q-function. We define the Bellman Operator T and the variance of the discounted sum of future rewards using T' as $(TQ)(s, a) = \mathbf{E}_{s'}(R(s, a, s') + \gamma \max_{a'} Q(s', a'))$ as well as $(T'Q)(s,a)^2 = \mathbf{Var}_{s'}(R(s,a,s') + \gamma \max_{a'} Q(s',a'))$ for any Q. There are several optimality criteria as approximations to the optimal Q-function. In this work, we concern the Bellman Residual minimisation, provided by the Q-function minimising the distance $\|Q - TQ\|$ (w.r.t. some appropriate norm) and the Temporal-Difference solution $Q = \mathbf{Solve}(TQ)$ given by the fixed point of the Bellman Operator followed by its projection on the Q-function's hypothesis space H_Q. For details we refer to [4,10,11].

3 Neural Rewards Regression

In the standard setting for NRR [1], we describe the Q-function for each possible action as Feed-Forward-Networks $N_a(s) = Q(s,a)$. The reward function to be fitted is hence given as $R(s,a,s') = N_a(s) - \gamma \max_{a'} N_{a'}(s')$, where the $\max$-operator has to be modelled by an appropriate architecture. Alternatively, a monolithic approach with one network $N(s,a) = Q(s,a)$ which takes the state and action as input $R(s,a,s') = N(s,a) - \gamma \max_{a'} N(s',a')$, can also be applied. The second model has the advantage

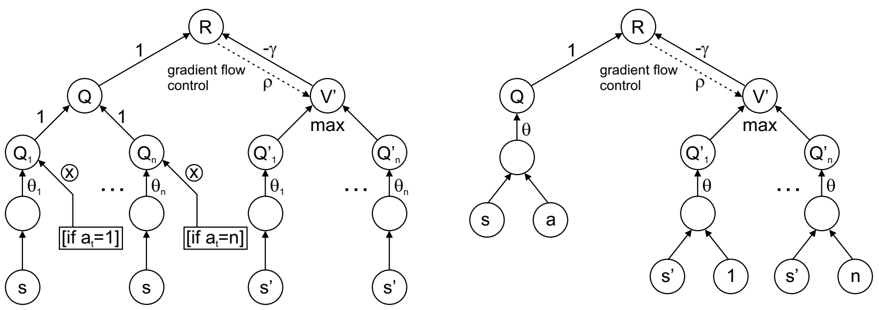

Fig. 1. Both alternatives of the NRR architecture (left: one network for each action, right: the monolithic approach). Connectors with the same parameters have shared weights.

of performing a generalisation over the action space as well. Using the Back Propagation algorithm this problem setting approaches the minimum of the (regularised) sampled Bellman Residual over all l observed transitions

$$L = \sum_{i=1}^{l} F(L_i) + \Omega(\theta) = \sum_{i=1}^{l} F\left(Q(s_i,a_i) - \gamma V(s_{i+1}) - r_i\right) + \Omega(\theta),$$

where θ are the network parameters, F an error function, and Ω an appropriate regulariser. The observed r_i and s_{i+1} have to be unbiased estimates for their expectations. The error function's gradient $\frac{dL}{d\theta} = \sum_{i=1}^{l} F'(L_i) \frac{d}{d\theta}\left(Q(s_i,a_i) - \gamma V(s_{i+1})\right) + \frac{d\Omega(\theta)}{d\theta}$ depends on the current Q-function and the successor Value Function (fig. 1). To obtain the fixed point of the Bellman Iteration instead, one retains $y_i := r_i + \gamma V(s_{i+1})$ and minimises iteratively $L = \sum_{i=1}^{l} F\left(Q(s_i,a_i) - y_i\right) + \Omega(\theta)$, until convergence of Q. Its gradient is then given as $\frac{dL}{d\theta} = \sum_{i=1}^{l} F'(L_i) \frac{d}{d\theta} Q(s_i,a_i) + \frac{d\Omega(\theta)}{d\theta}$. 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 backward path of the Back Propagation algorithm, the error propagation between the R-cluster and the V'-cluster is multiplied by a constant ρ (see fig. 1). The setting $\rho = 1$ leads to the sampled Bellman Residual minimisation, while for $\rho = 0$ one obtains the Bellman Iteration. Any other compromise is a possible trade-off between these two error definitions [12]. Optimality is then defined as the solution of the general equation system

$$\Gamma = \sum_{i=1}^{l} F'\left(Q(s_i,a_i) - \gamma V(s_{i+1}) - r_i\right) \frac{d}{d\theta}\left(Q(s_i,a_i) - \rho\gamma V(s_{i+1})\right) + \frac{d\Omega(\theta)}{d\theta} = 0.$$

A closer look at the approach reveals the similarities to TD-Learning [4,13]. Combining $\rho = 0$, $F(x) = x^2$, 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 Neural Networks [14].

4 Auxiliared Bellman Residual

It is well-known, that minimising the Bellman Residual on the one hand has the advantage of being a well-controllable learning problem as it is close to a Supervised Learning scheme, but on the other hand, tends to minimise terms of higher-order moments of the discounted sum of future rewards in the stochastic case, if no further, uncorrelated samples of every transition can be given [4,10]. In general the solutions are biased to Q-functions which are smoother for successor states of stochastic transitions. Specifically, if s_{i+1} and r_i are unbiased estimates for the successor states and rewards, respectively, the expression $\left(Q(s_i,a_i) - \gamma V(s_{i+1}) - r_i\right)^2$ is not an unbiased estimate for the true squared Bellman Residual $\left(Q(s,a) - (TQ)(s,a)\right)^2$, but for $\left(Q(s,a) - (TQ)(s,a)\right)^2 + (T'Q)(s,a)^2$. As an alternative to the usage of doubled trajectories, which is no option in our setting, Antos et. al. [2] introduced a modified squared Bellman Residual as a better approximation of the true Bellman Residual. They left it unnamed, so that we choose the working name Auxiliared Bellman Residual, defined as

$$L_{\text{aux}} = \sum_{i=1}^{l} \left(Q(s_i,a_i) - \gamma V(s_{i+1}) - r_i\right)^2 - \sum_{i=1}^{l} \left(h(s_i,a_i) - \gamma V(s_{i+1}) - r_i\right)^2$$

The task is then to solve $\hat{Q} = \arg\min_{Q \in H_Q} \max_{h \in H_h} L_{\text{aux}}$. The idea behind is to find an h that fits the Bellman operator over the observations. Its unavoidable error cancels the variance of the original expression down. We obtain

$$\begin{aligned} Z &= \mathbf{E}_{s'}(Q(s,a) - \gamma V(s') - R(s,a,s'))^2 - \mathbf{E}_{s'}(h(s,a) - \gamma V(s') - R(s,a,s'))^2 \\ &= (Q(s,a) - (TQ)(s,a))^2 - \mathbf{Err}(h(s,a),(TQ)(s,a)), \end{aligned}$$

which is the true loss function with an additional error term due to the suboptimal approximation of h, if H_h is unable to fit the Bellman operator arbitrarily precisely. This technique allows us to upper-bound the true Bellman Residual, if the error of

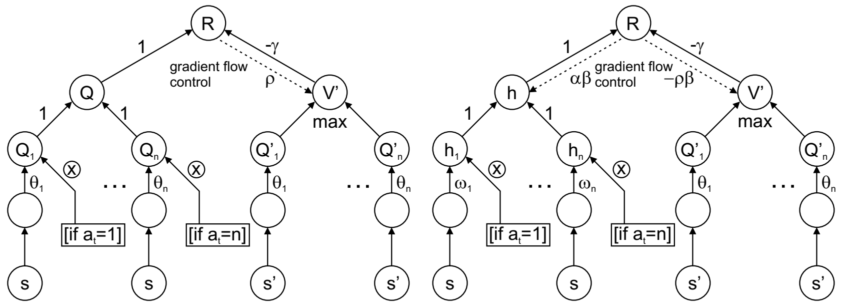

Fig. 2. The Auxiliared Bellman Residual NRR architecture. Both networks are trained simultaneously. The right network is used to obtain an approximation of $T'Q^2$.

h on TQ can be bounded. However, it is easy to see, that $\hat{L} \leq L$ within a saddle point of L_{aux}, if $H_Q = H_h$. Otherwise h would not provide the minimum of $\hat{L}$. Therefore, an optimum of L_{aux} would be provided by any Temporal-Difference fixed point, if it exists, as only in that case Q can fit the Bellman Operator as well as h and $L_{\text{aux}} = 0$. In contrast to the original proposal [2], we hence choose H_h either as a considerable richer function class than H_Q or with prior knowledge of the true Bellman Operator, such that $\hat{L}$ basically provides a better estimate of $T'Q^2$. As any such estimate of the variance is still biased, the method converges to a biased estimator of the true Bellman Residual minimising function $\hat{Q}^* \in H_Q$ within its function space only, but obviously provides a better approximation than the estimator given in sec. 3. In the following we consider the standard setting $F(x) = x^2$ and obtain the gradients

$$\Delta\theta = \Gamma + \beta\rho\gamma \sum_{i=1}^{l} \left(h(s_i, a_i) - \gamma V(s_{i+1}) - r_i\right) \frac{d}{d\theta} V(s_{i+1})$$

$$\Delta\omega = \alpha\beta \sum_{i=1}^{l} \left(h(s_i, a_i) - \gamma V(s_{i+1}) - r_i\right) \frac{d}{d\omega} h(s_i, a_i)$$

where ω are the accordant parameters describing h, $0 \leq \beta \leq 1$ controlling the influence of the auxiliary function h, and $\alpha \geq 1$ the strength of the optimisation of h in comparison with Q. This is incorporated into the NRR architecture as follows. In addition to the original NRR network, responsible for minimising L, we install a second similar, but auxiliary network, where the Q-clusters are replaced by respective h-clusters while the network part for Q' remains the same (see fig. 2). All three Q-function networks are still connected via shared weights. Finally, the auxiliary network has to fulfil two tasks, maximising $\hat{L}$ w.r.t. θ and minimising $\hat{L}$ w.r.t. ω. Because of the negative sign of $\hat{L}$, the gradient through the Q' part of the auxiliary network has to be sent in the opposite direction. The parameter α has to be chosen sufficiently large, such that h can fit the Bellman Operator almost instantaneously.

5 Policy Gradient Neural Rewards Regression

We now consider our second improvement, by introducing PGNRR, which is primarily a further generalisation of NRR to continuous action spaces. In general it is feasible for special function classes only to detect $V(s) = \max_a Q(s,a)$ in polynomial time. One possibility is to apply the idea of wire-fitting [15] or comparable approaches to NRR. Much more general and simpler is an approach combining the regression scheme on the rewards with PG methods. Therefore we replace the ensembles of Q-function

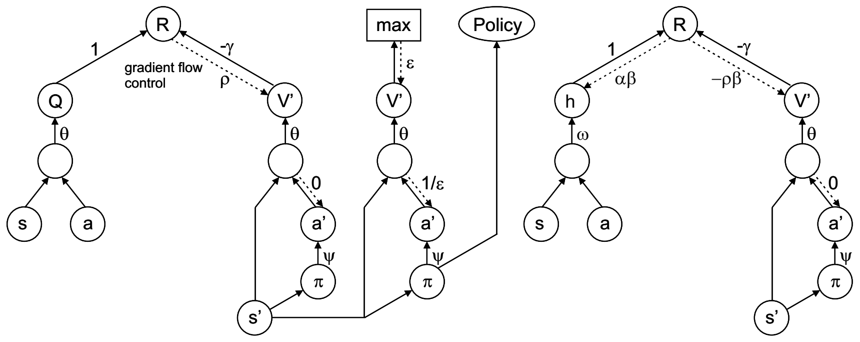

Fig. 3. The Policy Gradient NRR architecture. The additional architectural elements are used to train the policy network, whose task is to perform a near-optimal policy.

networks for the discrete actions by a single network evaluating $Q(s,a)$ for continuous states and actions. For the successor states we evaluate $Q(s', \pi(s'))$ where we assume that $\pi : S \rightarrow A$ with parameters ψ tend to perform the optimal policy (see fig. 3). Hence $Q(s', \pi(s'))$ gets close to $\max_{a'} Q(s', a')$. This is achieved by the maximisation of the Q-function for the successor states, simultaneously with the regression on the rewards. Therefore, a kind of Batch On-Policy-Iteration or Batch Actor-Critic Iteration is performed, by exploiting the intrinsic interrelationship between Q-function and policy. The gradient flow control technique in connection with shared weights is again sufficient to construct the appropriate architecture. In the standard network part for the successor state, the gradient flow through the policy network is simply cut off, such that the policy does not influence the regression on the rewards. In the extended part, a sufficiently small ϵ enables that only the policy contributes to the Q-function's maximisation. The common gradients are given as

$$\begin{aligned}\varDelta\theta = \sum_{i=1}^{l} \Big(& \big((Q(s_i,a_i) - \gamma Q(s_{i+1}, \pi(s_{i+1})) - r_i\big) \\ & \frac{d}{d\theta}\left(Q(s_i,a_i) - \rho\gamma Q(s_{i+1}, \pi(s_{i+1}))\right) + \frac{\epsilon d}{d\theta} Q(s_{i+1}, \pi(s_{i+1}))\Big) + \frac{d\varOmega(\theta)}{d\theta} \\ & + \beta\rho\gamma \left(h(s_i,a_i) - \gamma Q(s_{i+1}, \pi(s_{i+1})) - r_i\right) \frac{d}{d\theta} Q(s_{i+1}, \pi(s_{i+1}))\end{aligned}$$

$$\Delta\omega = \alpha\beta \sum_{i=1}^{l} \left(h(s_i, a_i) - \gamma Q(s_{i+1}, \pi(s_{i+1})) - r_i\right) \frac{d}{d\omega} h(s_i, a_i)$$

$$\Delta\psi = \sum_{i=1}^{l} \frac{d}{\epsilon d\psi} \pi(s_{i+1}) \frac{\epsilon d}{d\pi(s_{i+1})} Q(s_{i+1}, \pi(s_{i+1})) = \sum_{i=1}^{l} \frac{d}{d\psi} Q(s_{i+1}, \pi(s_{i+1})).$$

Still, a near-optimal policy is found by using Back-Propagation with shared weights only. After convergence, the policy can be evaluated by the network for π without using the Q-function as interim result.

6 Improving Data-Efficiency

With the problem class extension, we obtain an advancement of data-efficiency as well. This is based on two different arguments. The first one concerns the improvement w.r.t. the approachable optimality criteria. As we are now able to find the solution of a less biased Bellman Residual minimisation, we profit from that optimality criterion, which has several advantages [11,13]. Secondly, it has often been stated in the literature, that PG approaches are better able to profit from prior knowledge as the policies are usually known to be much simpler than the Q-functions and hence appropriate function classes can be given. Beside this argument, the question arises, if the combination of learning a Q-function and a policy, provides an improvement w.r.t. the bias-variance-dilemma [16] without special prior knowledge. This could be achieved by trading between the complexity of both hypothesis spaces $H_Q \subset C_Q$ and $H_\pi \subset C_\pi$, which are subsets of assumed concept spaces. However, there exists a function $\Pi : 2^{C_Q} \to 2^{C_\pi}$ mapping from all possible Q-function hypothesis spaces to the set of all possible policy hypothesis spaces and for $\Pi(H_Q) = H_\pi$ it holds

$$\forall Q \in H_Q : \exists \pi \in H_\pi : \forall s \in S : \pi(s) = \arg\max_a Q(s, a)$$
$$\forall \pi \in H_\pi : \exists Q \in H_Q : \forall s \in S : \pi(s) = \arg\max_a Q(s, a).$$

That is, $\Pi(H_Q)$ contains a corresponding policy for any $Q \in H_Q$ and vice versa. For illustration, we tested the Cerebellar Model Articulation Controller (CMAC) architecture [4,17] as regressor for the true Q-function together with the Wet-Chicken benchmark problem [18]. The bias-variance-trade-off is controlled by the number of tiles. The policy is chosen out of the respective function class $\Pi(H_Q)$, which can easily be given for this example, and is fitted, such that the average of $Q(s, \pi(s))$ is maximal over the observations. If we decrease the parameters of π, such that $H_\pi \subset \Pi(H_Q)$, then we additionally trade bias against variance over the policy space. In general, it is indeed possible to obtain the maximal values at a point that does not lie on the diagonal, where $H_\pi = \Pi(H_Q)$. Fig. 4 shows the results. However, a detailed theoretical analysis, under which conditions an improvement can be obtained, is part of future work.

7 Benchmark Results

In order to support the theoretical considerations on data-efficiency with empirical evidence, we tested the novel architecture on the Cart-Pole problem as described in [10] and on a simulation of a gas turbine, which has been used for a Reinforcement Learning project to reduce acceleration and pressure intensities in the combustion chamber (RMS) while maintaining high and stable load. A more detailed description can be found in [20]. We have chosen these two benchmarks, because standard NRR already shows competitive results in comparison with TD-Learning [13] on human-designed as well as automatically trained local basis functions, the Adaptive Heuristic Critic [4], Neural Fitted Q-Iteration [19], and the Recurrent Control Neural Network [20] as a model-based approach exploiting the special properties of the considered problem class. Unfortunately a fair comparison of NRR and PGNRR is difficult as each setting redounds to one method's advantage. We decided to rediscretise the continuous action policy. But it is apparent that a better performance in comparison to the natural discrete method is only a sufficient criterion for its prevalence as the continuous method could have found a policy which especially exploits its continuous action character.

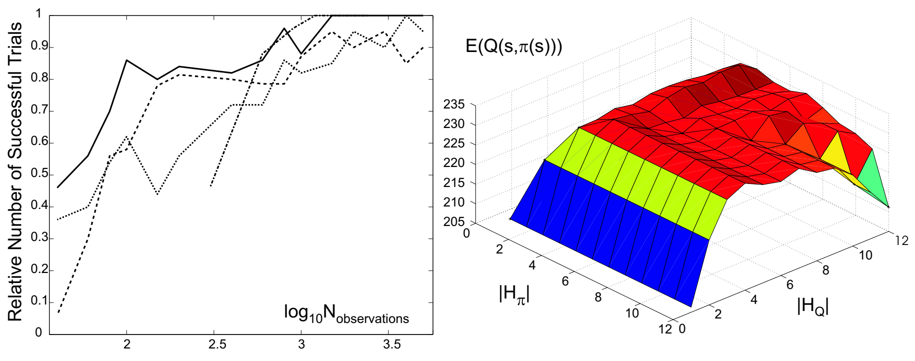

Fig. 4. Left: Performance comparison on the Cart-Pole problem. The figure shows the number of successful learning trials, where the controller balances the pole for at least 100000 steps, dashed: NRR, solid: PGNRR, dotted: PGNRR rediscretised, and dash-dotted: Neural Fitted Q-Iteration [19] for at least 3000 steps. Right: Bias-variance-trade-off for Q-function and policy for a tile-coding example. The maximal values are averaged over 100 test trials.

Nevertheless, for the Cart-Pole problem (see fig. 4), it can be seen that PGNRR performs considerably better than NRR and also its discretised version is quite comparable to the standard NRR. Both methods were trained with 4-layer networks. The most successful setting for NRR was already verified with Q-function networks with 16 and 8 neurons in the first and the second hidden layer, respectively. In accordance with our observations w.r.t. the bias-variance-trade-off, for PGNRR we improved the complexity of the Q-function (30 and 15 hidden neurons) and have chosen a smaller policy network (10 and 5 hidden neurons). As learning algorithm we applied the Vario-Eta method [21], where we observed best results for this problem. We have chosen a batch size of 5 and

Table 1. Performance comparison on a controller for a gas turbine simulation. We opposed our best NRR controller with a PGNRR controller for the same discrete action data set. As the underlying MDP's action space is two-dimensional we applied each a one- and two-action discretisation. Interestingly, the performance of the two discretised controllers is indeed better than on the natural discrete controller.

Gas Turbine	NRR Discrete	PGNRR Simple Discretised	PGNRR Double Discretised	PGNRR Continuous
Average Reward	0.8511 ± 0.0005	0.8531 ± 0.0005	0.8560 ± 0.0005	0.8570 ± 0.0005

$\eta = 0.05$. The input data was rescaled to constrain the inputs between -1 and $+1$. The net training was performed with an exponentially decreasing learning rate. For comparison we plotted the number of successful learning trials with Neural Fitted Q-Iteration [19], where the pole is balanced for at least 3000 steps, apparently the results are an upper bound for the presented setting.

As a second benchmark we applied NRR and PGNRR on a gas turbine simulation [20]. There are two different control variables, the so-called pilot gas and inlet guide vane (IGV). These variables are, beside other control variables and measurements, part of the state space. The actions are to wait and do nothing, to increase and decrease pilot gas, and to increase and decrease IGV each by a certain amount. Only one of these five actions is selectable in each time step. For the continuous version we allow a maximum of the discrete modification simultaneously for both control variables. We used the Recurrent Neural Rewards Regression (RNRR) model as described in [1] and an appropriate Recurrent PGNRR architecture for comparison. The recurrent substructure is necessary, because the state space is not Markovian, specifically the underlying MDP is of higher order.

For both architectures we used the same settings as described for the Cart-Pole problem, but additionally applied weight decay as regularisation with a factor $\lambda = 0.001$. Applying this setting we obtained the best results with the NRR architecture and also reproducible results with PGNRR. For comparison with the standard NRR, we rediscretised the continuous actions independent from each other as well as to the simple discrete case, where the larger relative action was discretised and the appropriate other action not executed. It can be seen in tbl. 1 that with both discretised versions indeed a reward improvement could achieved. Note, that a reward difference of 0.002 and 0.006 corresponds to an additional power output of 0.14 and 0.42 MW, respectively, with stable RMS for a power plant with a maximum power of 200 MW.

8 Conclusion

We introduced two generalisations of Neural Rewards Regression to extend the approachable optimality criteria and the considered problem class substantially. Of course, PGNRR can also be combined with Recurrent Neural Networks [1], so that a very broad problem class is addressed. Future work will mainly concern the broadening of its application for gas turbine control and the theoretical analysis of the improvement of data-efficiency.

References

1. Schneegass, D., Udluft, S., Martinetz, T.: Neural rewards regression for near-optimal policy identification in markovian and partial observable environments. In: Verleyen, M. (ed.) Proc. of the European Symposium on Artificial Neural Networks, pp. 301–306 (2007)
2. Antos, A., Szepesvári, C., Munos, R.: Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. In: Lugosi, G., Simon, H.U. (eds.) COLT 2006. LNCS (LNAI), vol. 4005, pp. 574–588. Springer, Heidelberg (2006)
3. Schneegass, D., Udluft, S., Martinetz, T.: Kernel rewards regression: An information efficient batch policy iteration approach. In: Proc. of the IASTED Conference on Artificial Intelligence and Applications, pp. 428–433 (2006)
4. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
5. Hinton, G.E., McClelland, J.L., Rumelhart, D.E.: Distributed representations. In: Parallel Distributed Processing, pp. 77–109 (1986)
6. Haykin, S.: Neural Networks: A Comprehensive Foundation. Macmillan, New York (1994)
7. Sutton, R.S., McAllester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems, vol. 12 (2000)
8. Wang, X., Dietterich, T.: Model-based policy gradient reinforcement learning. In: International Conference on Machine Learning (2003)
9. Ghavamzadeh, M., Engel, Y.: Bayesian policy gradient algorithms. In: Advances in Neural Information Processing Systems (2006)
10. Lagoudakis, M.G., Parr, R.: Least-squares policy iteration. Journal of Machine Learning Research, 1107–1149 (2003)
11. Munos, R.: Error bounds for approximate policy iteration. In: Proc. of the International Conference on Machine Learning, pp. 560–567 (2003)
12. Baird III, L.C.: Residual algorithms: Reinforcement learning with function approximation. In: Proc. of the International Conference on Machine Learning, pp. 30–37 (1995)
13. Tsitsikilis, J.N., Van Roy, B.: An analysis of temporal difference learning with function approximation. IEEE Transactions on Automatic Control 42(5), 674–690 (1997)
14. Pearlmutter, B.: Gradient calculations for dynamic recurrent neural networks: A survey. IEEE Transactions on Neural Networks 6(5), 1212–1228 (1995)
15. Baird, L., Klopf, A.: Reinforcement learning with high-dimensional, continuous actions. Technical Report WL-TR-93-1147, Wright Laboratory, Wright-Patterson Air Force Base, OH 45433-7301 (1993)
16. Geman, S., Bienenstock, E., Doursat, R.: Neural networks and the bias/variance dilemma. Neural Computation 4(1), 1–58 (1992)
17. Timmer, S., Riedmiller, M.: Fitted q iteration with cmacs. In: ADPRL. Proceedings of the IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, pp. 1–8. IEEE Computer Society Press, Los Alamitos (2007)
18. Tresp, V.: The wet game of chicken. Siemens AG, CT IC 4, Technical Report (1994)
19. Riedmiller, M.: Neural fitted q-iteration - first experiences with a data efficient neural reinforcement learning method. In: Proceedings of the 16th European Conference on Machine Learning, pp. 317–328 (2005)
20. Schaefer, A.M., Schneegass, D., Sterzing, V., Udluft, S.: A neural reinforcement learning approach to gas turbine control. In: Proc. of the International Joint Conference on Neural Networks (2007) (to appear)
21. Neuneier, R., Zimmermann, H.G.: How to train neural networks. In: Orr, G.B., Müller, K.-R. (eds.) Neural Networks: Tricks of the Trade. LNCS, vol. 1524, pp. 373–423. Springer, Heidelberg (1998)