ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

Ensembles for Continuous Actions in Reinforcement Learning

Siegmund Duell[1,2] and Steffen Udluft[1]

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

2- Berlin University of Technology, Machine Learning,
Marchstr. 23, D-10587 Berlin, Germany

Abstract. Data efficient reinforcement learning methods allow to optimize controllers (policies) for complex technical systems in a data-driven manner. Still, there is the risk that, when running such a policy on the real system, it performs considerably worse than expected. For policies with discrete actions it has been shown that this risk can be reduced considerably, when, instead of just using a single policy, that by chance might be inferior, a whole ensemble of policies is used to select the final policy by an aggregation like, e.g., majority voting. In this paper we extend the applicability of the ensemble approach to vector-valued, continuous actions.

1 Introduction

Reinforcement learning (RL) [1] offers means to optimize controllers for complex technical systems where no sufficiently accurate analytical description or simulation is available. Being a data-driven approach, RL uses actual observations of the system to be controlled, which, in turn, means that a sufficiently large number of observations has to be collected in order to produce good policies. This has been a major drawback to the applicability of RL to real world tasks in the past. To overcome this issue a couple of data-efficient RL-algorithms have been developed in the last decade [2–7], aiming for the best possible use of the rather small number of observations available.

In the kind of control problems we are concerned about, like optimizing the high-level control of a gas turbine [8], operating the system with an inferior policy is expensive due to possible load reduction or even damage to the hardware. Therefore, we have to start with a batch of observation data that has been produced during regular operation. Data-efficient RL-algorithms enable us to create policies, which possibly are an improvement over a default controller. The problem is that evaluating such a policy on the real system is too expensive, if, by chance, it performs considerably worse than normal operation. There are (at least) two strategies to overcome this problem: Using some kind of offline evaluation to determine the policy quality without actually executing it [9], or, reduce the risk to execute an inferior data-based policy. A solution to the latter strategy is to use not just a single policy, that by chance might be inferior, but a whole ensemble of policies [10, 11]. It has been shown that for

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

policies with discrete actions a significant improvement can be achieved when aggregating the set of actions proposed by the ensemble members to a final action by an aggregation scheme like Q-averaging, Boltzmann multiplication, or majority voting [10,11].Previous studies indicate that especially majority voting is a powerful aggregation scheme. The situation is similar to the effectiveness of ensembles in classification problems [12].

In this paper, we extend the ensemble technique to continuous action policies. In general, the actions will be k-dimensional real valued vectors. Thus, majority voting cannot be applied directly, because any action will usually only be proposed once by the ensemble.

2 Policy Generation

For our experiments, we use a data-efficient RL-algorithm for continuous state and action spaces, introduced in 2007 by Schneegass et al. [7], called the *policy gradient neural rewards regression* (PGNRR). It is an iteration-free method, based on the neural rewards regression for discrete actions [6]. The PGNRR architecture comprises a neural policy network and two instances of a neural network representing the Q-function. One instance of the Q-network is evaluated on the current state and action taken from the data, the other is evaluated on the successor state and the action generated by the neural policy network. Both instances share the same weights. The Q-network is trained to encode the Q-function of the policy network by back-propagation learning on batch data. This implements temporal difference learning of a Q-function with stochastic representation of previously recorded data. At the same time the neural policy is trained to maximize the Q-function's output, thus representing the greedy policy for the current Q-function.

3 Aggregation Methods

In general, any ensemble of policies for RL requires a set of policies and an aggregation method. Besides the mean and the component-wise median of all action vectors, a list of aggregation methods is given below. In all following aggregation methods a set of policies is assumed to be available to provide actions to calculate the final action.

Binning A naïve method to provide a robust aggregation approach is majority voting on a simple discretization (binning) of the actions space. This is accomplished by discretizing all dimensions of the action space into bins of equivalent size. The action results from the bin with the most hits. An action could be either derived by the mean of all actions within the winning bin or the action vector given by the center of that bin. Besides the number of bins, no further parametrization is required.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

Density Based The essence of majority voting is to search for the action with the highest density in the action space. We compute the density value d_i for each of the N action vectors a_i in the k-dimensional action space using Parzen windows [13], i.e.,

$$d_i = \sum_{j=1}^{N} e^{-\frac{\sum_{l=1}^{k}(a_{il}-a_{jl})^2}{r^2}}, \tag{1}$$

where r is a distance parameter. After calculating the density for all action vectors, the action with highest density is selected. For our experiments we normalized the action space to $[-1; 1]$ in each dimension and choose $r = 0.001$.

Data Center This parameter-free algorithm applies the following rule to select an action from the ensemble: In a first step, the center of all vectors is calculated by simply calculating the mean. Next, the Euclidean distance from the center to all action vectors is calculated. The vector with the greatest distance from the center is removed from the list of actions. The procedure repeats these steps until only two actions are left. Finally, either the average or one of the two remaining actions is randomly selected.

4 Experiments and Results

To show the benefits of policies based on ensembles in domains with continuous state and action spaces, a series of experiments is conducted. The pole swing-up benchmark as well as a gas turbine simulation are used as dynamics, policies are calculated by the PGNRR algorithm.

For an initial test of policy aggregation methods, a pole swing-up simulation is used with standard parameters. The pole swing-up simulation is a variant of the cart pole task [1], Sec. 3.3, where the cart is not able to move and the pole is free to rotate. Actions are continuous within -15 to 15 newton. For policy calculation, 5000 data points are used which were generated by random exploration. The data is used to run the PGNRR algorithm for 2500 epochs, resulting in the same number of successive policies. The quality of these policies is illustrated within Fig. 1, disclosing the risk of selecting a policy with insufficient performance.

4.1 Gas Turbine Simulation

To demonstrate the capabilities of the presented approach on a problem similar to the real world application of our interest, a gas turbine simulation is introduced [1]. A gas turbine consists of many sub-systems, among which we focus on the combustion process. Our goal is to minimize combustion humming and emissions while keeping a desired power output (load). To accomplish

[1]The source code of the simulation as well as additional information can be found at `https://github.com/duell/GTSim`.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

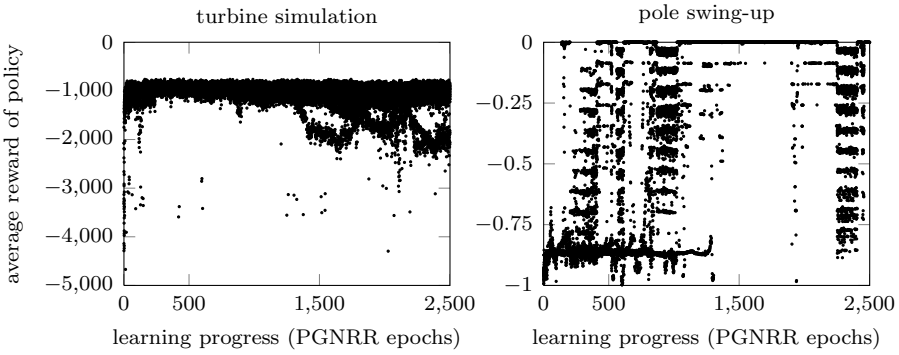

Fig. 1: For both, the gas turbine simulation (left) and the pole swing-up (right), 25 learning trials are performed over 2500 epochs. For each epoch, all policies are evaluated 25 times over 1000 time steps. The average reward of these evaluations is shown for all policies.

these goals, the simulation provides the controllable variables pilot and boost. Both variables affect humming and emissions. While pilot trades humming for emissions, boost is capable to reduce humming and trades it for thermal stress. This can cause a negative delayed reward if too much thermal debt is accumulated. In each time step, pilot and boost can be changed up to a certain amount: $pilot_{t+1} = pilot_t + \Delta pilot_t$ and $boost_{t+1} = boost_t + \Delta boost_t$, where $\Delta pilot_t$ and $\Delta boost_t$ define the two-dimensional continuous action vector. The following equations describe the dynamics of the turbine:

$$\begin{aligned}
\text{decay parameter, } d &= \frac{1}{1+e^{(-boost/c_d)}} \\
\text{boost temperature, } T_b &= T_b d + c_{T_b} boost + c_{T_l} load \\
\text{humming, } h &= \frac{c_h}{c_{h_P} pilot + c_{h_l}} - c_{h_b} boost^2 \\
\text{flame temperature, } f &= c_{f_l} load + c_{f_b} boost + c_{f_P} pilot + c_f \\
\text{emissions, } e_i &= e^{\frac{f - c_{e_f}}{c_e}} \\
\text{emissions measured, } e_c &= \text{conv}(e_i\{t, ...t_{-n}\})
\end{aligned}$$

conv() describes a convolution of n past time steps of emissions e_i. For each time step, the simulator provides observations about its current state. The reward function is based on humming, emissions, and the boost temperature. The reward signal is calculated as follows:

$$\begin{aligned}
r_T &= \begin{cases} 0, & \text{if } T_b < c_{crt} \\ \left(\frac{T_b - c_{T_b}}{c_{T_{b_\Delta}}} \min\right)^2, & \text{else} \end{cases} \\
r_{\text{total}} &= h \cdot c_{r_H} + e_c \cdot c_{r_E} + r_T \cdot c_{r_T}
\end{aligned}$$

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

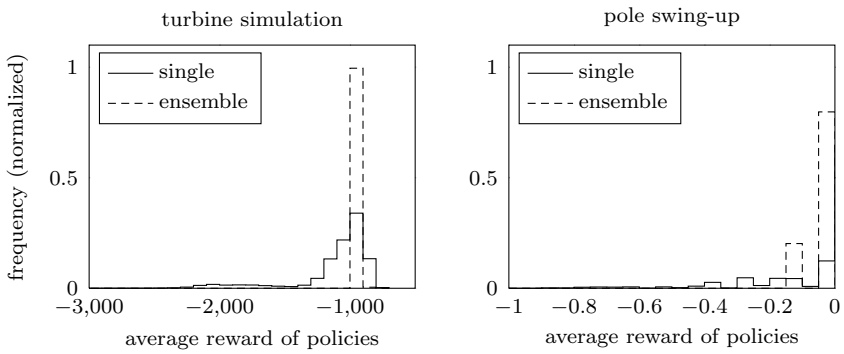

Fig. 2: Performance, i.e., average reward achieved during the evaluation process for both the gas turbine (left) and the pole swing-up (right) problem. For both benchmarks, all trained policies are evaluated individually (dashed line). The ensembles (solid line) are based on 25 independently trained single policies.

4.2 Results

As mentioned above and illustrated in Fig. 1, 25 trials of the PGNRR policy generation algorithm are performed over 2500 epochs for both the gas turbine as well as the pole swing-up problem. Within each training epoch, the resulting policy was stored for evaluation and further usage in ensembles. As discussed earlier, the performance of a resulting policy does not converge—and the likelihood of a policy performing poorly cannot be minimized by simply training for more epochs. To overcome the problem of probably poor performance at run time, an ensemble can be deployed. To demonstrate the capabilities of ensembles, 500 successive ensembles with 25 members are created. Each training trial of the PGNRR contributed a policy from the same training epoch, i.e., the first ensemble was created from all policies at epoch 2000, the last at epoch 2499. Fig. 2 illustrates the performance of all ensembles, aggregated with the density based approach. In addition, the performance of all members evaluated as single policies is shown. Note that for the evaluated benchmarks, all tested aggregation methods deliver comparable results. For the gas turbine simulation, all ensembles achieve robust results. For the pole swing-up problem, considerably more ensemble solutions reach a stable pole swing-up position, with some solutions (-0.1) solving the swing-up problem late or unstable. However, none of the ensembles fail to swing up.

5 Conclusion

A series of policy aggregation methods was introduced to use ensembles of policies for vector-valued, continuous actions. The experiments show that all investigated aggregation methods produce robust and well-performing policies, i.e., the risk of obtaining an inferior policy can be reduced considerably compared

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

to using just a single policy. This is of major importance if no simulation or analytical description of the domain of interest is available and evaluation on the real system is expensive.

Acknowledgment

Part of this work has been funded by the Federal German Ministry for Education and Research under the grant ALICE, 01 IB10003 A-C.

References

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

[2] D. Ernst, P. Geurts, and L. Wehenkel. Iteratively Extending Time Horizon Reinforcement Learning. In *Proceedings of the 14th European Conference on Machine Learning*. Springer, 2003.

[3] D. Ernst, P. Geurts, and L. Wehenkel. Tree-Based Batch Mode Reinforcement Learning. *Journal of Machine Learning Research*, 6, 2005.

[4] 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*, 2005.

[5] D. Schneegaß, S. Udluft, and T. Martinetz. Kernel Rewards Regression: An Information Efficient Batch Policy Iteration Approach. In *Artificial Intelligence and Applications*, 2006.

[6] 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*, 2007.

[7] D. Schneegaß, S. Udluft, and T. Martinetz. Improving Optimality of Neural Rewards Regression for Data-Efficient Batch Near-Optimal Policy Identification. In *Proc. of the International Conference on Artificial Neural Networks*, 2007.

[8] A.M. Schaefer, D. Schneegass, V. Sterzing, and S. Udluft. A Neural Reinforcement Learning Approach to Gas Turbine Control. In *Proc. of the International Joint Conference on Neural Networks*, 2007.

[9] A. Hans, S. Duell, and S. Udluft. Agent Self-Assessment: Determining Policy Quality Without Execution. In *Proceedings of the IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 2011.

[10] M.A. Wiering and H. van Hasselt. Ensemble Algorithms in Reinforcement Learning. *IEEE transactions on systems, man, and cybernetics*, 38(4), 2008.

[11] A. Hans and S. Udluft. Ensemble Usage for More Reliable Policy Identification in Reinforcement Learning. In *Proceedings of the 19th European Symposium on Artificial Neural Networks*, 2011.

[12] T.G. Dietterich. Ensemble Methods in Machine Learning. *Lecture Notes in Computer Science*, 1857, 2000.

[13] E. Parzen. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, 33(3), 1962.