Proxy functions for Approximate Reinforcement Learning

Alibekov, Eduard; Kubalík, Jiří; Babuška, Robert

DOI
10.1016/j.ifacol.2019.09.145

Publication date
2019

Document Version
Final published version

Published in
IFAC-PapersOnLine

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 52-11 (2019) 224–229

Proxy Functions for Approximate Reinforcement Learning

Eduard Alibekov * **Jiří Kubalík** ** **Robert Babuška** ***

* *Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague (e-mail: alibeedu@fel.cvut.cz).*
** *Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, Czech Republic (e-mail: jiri.kubalik@cvut.cz)*
*** *Cognitive Robotics department, Faculty of 3mE, Delft University of Technology, the Netherlands and also with the Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, Czech Republic, (e-mail: r.babuska@tudelft.nl)*

Abstract: Approximate Reinforcement Learning (RL) is a method to solve sequential decision-making and dynamic control problems in an optimal way. This paper addresses RL for continuous state spaces which derive the control policy by using an approximate value function (V-function). The standard approach to derive a policy through the V-function is analogous to hill climbing: at each state the RL agent chooses the control input that maximizes the right-hand side of the Bellman equation. Although theoretically optimal, the actual control performance of this method is heavily influenced by the local smoothness of the V-function; a lack of smoothness results in undesired closed-loop behavior with input chattering or limit-cycles. To circumvent these problems, this paper provides a method based on Symbolic Regression to generate a locally smooth proxy to the V-function. The proposed method has been evaluated on two nonlinear control benchmarks: pendulum swing-up and magnetic manipulation. The new method has been compared with the standard policy derivation technique using the approximate V-function and the results show that the proposed approach outperforms the standard one with respect to the cumulative return.

Keywords: reinforcement learning, continuous state space, optimal control, policy derivation, V-function

1. INTRODUCTION

Reinforcement learning (RL) has the potential to solve challenging decision-making and control problems in engineering and in a variety of other disciplines, such as economics or medicine (Busoniu et al., 2010; Sahba et al., 2006; Deng et al., 2017; Guan et al., 2015). There is a wide variety of RL architectures, which can be broadly classified into critic-only, actor-only, and actor-critic schemes (Konda and Tsitsiklis, 2000). In this paper, we focus on the critic-only architecture for continuous state spaces. The RL agent first learns an approximate value function (V-function), based on which it then derives the optimal control policy. We restrict ourselves to the model-based scenario: a nonlinear deterministic state-space system model is used for V-function learning and policy derivation (Polydoros and Nalpantidis, 2017; Kuvayev and Sutton, 1996).

To derive a control policy from the V-function, the RL agent chooses the control input that maximizes the right-hand side of the Bellman equation. The performance of such a control law depends on the local smoothness of the V-function and therefore on the type of approximator used. A wide spectrum of approximation techniques have been used in RL: local linear regression (Grondman et al., 2012), deep neural networks (Lillicrap et al., 2015), fixed or adaptive basis functions (Buşoniu et al., 2011), regression trees (Ernst et al., 2005). In this paper, we use approximation by means of triangular basis functions (BFs) as a baseline, since the convergence of value iteration can be guaranteed. However, approximation with basis functions may cause chattering of the state trajectory during transients or may render the goal state unreachable, as illustrated in Fig. 1. The bottom plot clearly shows that due to the odd shape of the surface (visualised as the diamond-shaped level curves), the state trajectory is deflected from the desired path to the goal and ends at some distance away from the desired position.

A possible approach to alleviating the above problem is to approximate the V-function by using a technique which would not be affected by any interpolation artifacts between the individual basis functions. One of the methods to construct a compact smooth approximation is symbolic regression (SR). Based on genetic programming, SR searches for an analytic expression which fits best the given data. However, we found (Alibekov et al., 2018) that it is difficult to directly apply SR to approximate an existing V-function. While the symbolic approximation can achieve a very low mean-squared error, the resulting policy can be very different and sub-optimal. This is because subtle,

2405-8963 © 2019, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2019.09.145

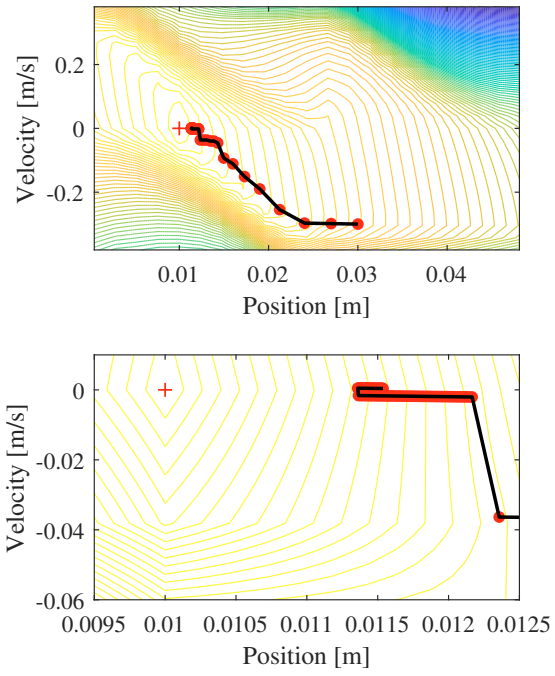


Fig. 1. An example of a state trajectory on the magnetic manipulation benchmark (Section 4) heavily influenced by the V-function approximator. Top: the state trajectory superimposed on the RHS of the Bellman equation level curves. Bottom: a close-up view of the state trajectory approaching, but not reaching the goal state $[0.01, 0]$.

yet important details of the V-function surface are not represented properly by the symbolic approximator.

Our first approach to overcoming this limitation by means of a smooth proxy function was described in (Alibekov et al., 2016). The method relies on a binary fitness function, which leads to a non-convex optimization problem and reduces chances of finding an accurate proxy function. Consequently, the originally proposed symbolic approximator had to be combined with a numeric approximator.

The current paper builds upon this result by resolving these outstanding issues of the originally proposed proxy function. In this paper, we formulate and propose an enhanced symbolic regression approach which uses linear programming in order to find more powerful models and solve the above problems. Moreover, we redesign it in a way that is applicable to any linear-in-parameters approximator.

The paper is structured as follows. Section 2 presents the necessary RL preliminaries. The proposed method for fitting a proxy function is described in Section 3. The description of benchmarks used and the testing procedure can be found in Section 4. Section 5 provides results with a detailed discussion of the benefits and drawbacks of the proposed method. Section 6 concludes the paper.

2. PRELIMINARIES

Define an n-dimensional state space $\mathcal{X} \subset \mathbb{R}^n$, and m-dimensional action space $\mathcal{U} \subset \mathbb{R}^m$. The system to be controlled is described by the state transition function $x_{k+1} = f(x_k, u_k)$, with $x_k, x_{k+1} \in \mathcal{X}$ and $u_k \in \mathcal{U}$. A user-defined reward function assigns a scalar reward $r_{k+1} \in \mathbb{R}$ for each transition from x_k to x_{k+1} using action u_k:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) \\ r_{k+1} &= \rho(x_k, u_k, f(x_k, u_k)) \end{aligned} \tag{1}$$

In this paper we restrict ourselves to the reward function depending on the next state only, omitting x_k and u_k for clear notation.

To solve the RL problem, we define a finite set of discrete control input values $U = \{u_1, u_2, \ldots, u_M\}$ drawn from $\mathcal{U}$. An approximate V-function denoted by $\hat{V}(x)$ is then computed by solving the Bellman equation:

$$\hat{V}(x) = \max_{u \in U} \left[\rho\big(f(x,u)\big) + \gamma \hat{V}\big(f(x,u)\big) \right] \tag{2}$$

where γ is a user-defined discount factor. The policy is the mapping:

$$h : \mathcal{X} \to \mathcal{U} \tag{3}$$

and the optimal discrete-valued policy corresponding to $\hat{V}(x)$ is given by:

$$\hat{h}(x) \in \operatorname*{argmax}_{u \in U} \left[\rho\big(f(x,u)\big) + \gamma \hat{V}\big(f(x,u)\big) \right], \forall x\,. \tag{4}$$

There are several algorithms to compute an approximate V-function for a continuous state space. For the purposes of this paper, the fuzzy V-iteration algorithm (Busoniu et al., 2010) is used because of its guaranteed convergence. The algorithm can be briefly described as follows. First, the fixed structure of the approximator is defined in terms of triangular basis (membership) functions. The V-function approximator is described as:

$$\hat{V}(x) = \theta^T \phi(x) \tag{5}$$

where $\phi = [\phi_1(x), \phi_2(x), \ldots, \phi_N(x)]^T$ is the vector of fixed triangular basis functions, with each $\phi_i(x)$ centered in s_i such that $\phi_i(s_i) = 1$ and $\phi_j(s_i) = 0$, $\forall j \neq i$. The basis functions are normalized so that $\sum_{j=1}^{N} \phi_j(x) = 1$, $\forall x \in \mathcal{X}$. Finally, $\theta \in \mathbb{R}^N$ is the corresponding parameter vector. The value iteration is then defined as:

$$\theta_i \leftarrow \max_{u \in U} \left[\rho\big(f(s_i,u)\big) + \gamma \theta^T \phi\big(f(s_i,u)\big) \right] \tag{6}$$

for $i = 1, 2, \ldots, N$. This algorithm guaranties convergence under certain conditions (Busoniu et al., 2010) and terminates when the convergence threshold ϵ is reached:

$$||\theta - \theta^-||_\infty \leq \epsilon \tag{7}$$

with θ^- the parameter vector calculated in the previous iteration.

3. PROPOSED METHOD

The main idea of the proxy function method (Alibekov et al., 2016) is to find through symbolic regression a smooth, analytically defined function P, which $\forall x \in \mathcal{X}$ satisfies the following equation:

$$\operatorname*{argmax}_{u \in U} P(f(x,u)) = \operatorname*{argmax}_{u \in U} \left[\rho\big(f(x,u)\big) + \gamma \hat{V}\big(f(x,u)\big) \right] \tag{8}$$

In order to generalize this method, assume that the proxy function has the form:

$$P(x) = \beta_1 p_1(x) + \beta_2 p_2(x) + \ldots + \beta_q p_q(x) \tag{9}$$

where $p_1, \ldots, p_q$ are continuous analytic functions generated by means of evolutionary programming, each of them

being defined over the whole state space and $\beta_1, \ldots, \beta_q$ are real-valued coefficients.

Assume that the $\hat{V}(\cdot)$ approximator is given. As described earlier, the policy derivation process can be regarded as hill climbing. At each time step, the agent selects the control input which leads to the highest value of the right-hand side of the Bellman equation:

$$u^* = \underset{u \in U}{\operatorname{argmax}} \left[\rho\big(f(x,u)\big) + \gamma \hat{V}\big(f(x,u)\big) \right] \tag{10}$$

The selected action u^* is then applied to the system, which leads to the new state:

$$x^* = f(x, u^*) \tag{11}$$

To find a P, or its close approximation, we generate a set of N state samples $X = \{x_1, x_2, \ldots, x_N\} \in \mathcal{X}$. By using the already defined set of discrete control inputs $U = \{u_1, u_2, \ldots u_M\}$, for each state $x_i \in X$ we construct the following set of next states:

$$X_i^n = \big\{ x_{ij} \mid x_{ij} = f(x_i, u_j), j = 1, 2, \ldots M \big\} \tag{12}$$

and partition it into optimal and suboptimal next states:

$$\tilde{X}_i = x_i^o \cup X_i^s \tag{13}$$

The optimal next state maximizes the right-hand side of the Bellman equation:

$$x_i^o = \underset{x_{ij} \in X_i^n}{\operatorname{argmax}} \left[\rho(x_{ij}) + \gamma \hat{V}(x_{ij}) \right] \tag{14}$$

and the suboptimal next states are all the remaining ones:

$$X_i^s = \tilde{X}_i \setminus x_i^o \tag{15}$$

For simplicity, we assume that the optimal state for each sample in X is unique. However, the proposed method can be trivially extended to handle multiple optimal next states.

To define the fitness function for symbolic regression, (8) is reformulated as follows:

$$P(x_{ik}^s) - P(x_i^o) < 0, \ \forall i, k \tag{16}$$

This means that for each state x_i, the proxy function value for the optimal next state must be larger than the value for the suboptimal next states. Index k runs over all L elements in X_i^s. Substituting from (9), the above inequality becomes:

$$\sum_{j=1}^{q} \beta_j \big(p_j(x_{ik}^s) - p_j(x_i^o) \big) < 0, \quad \forall i, k \tag{17}$$

To simplify the notation, define an auxiliary variable d_j^{ik} as:

$$d_j^{ik} = p_j(x_{ik}^s) - p_j(x_i^o) \tag{18}$$

To represent (17) for all the data in matrix form, define

$$O = \begin{pmatrix} d_1^{11} & d_2^{11} & \cdots & d_q^{11} \\ d_1^{12} & d_2^{12} & \cdots & d_q^{12} \\ \vdots & \vdots & \ddots & \vdots \\ d_1^{1L} & d_2^{1L} & \cdots & d_q^{1L} \\ d_1^{21} & d_2^{21} & \cdots & d_q^{21} \\ d_1^{22} & d_2^{22} & \cdots & d_q^{22} \\ \vdots & \vdots & \ddots & \vdots \\ d_1^{NL} & d_2^{NL} & \cdots & d_q^{NL} \end{pmatrix} \tag{19}$$

and solve the problem by linear programming:

$$\begin{aligned} \min_{\beta} \ & \emptyset \quad \text{such that} \\ & O\beta \leq \epsilon \end{aligned} \tag{20}$$

where ϵ represents a small negative constant.[1] Note that (17) defines a feasibility problem, rather than a minimization problem. To guide the evolutionary process toward a feasible solution, we introduce an infeasibility measure of the candidate solution. Define a vector of non-positive variables $s = [s_1, \ldots, s_{[N \times L]}]^T$. The fitness function for SR can now be defined as:

$$\begin{aligned} \min_{\beta, s} \ & \sum_{i=1}^{N \times L} -s_i \quad \text{such that} \\ & O\beta + s \leq \epsilon \\ & -\infty \leq s \leq 0 \end{aligned} \tag{21}$$

This formulation adds an extra variable to every inequality, which represents the measure of infeasibility of the resulting model, and which linear programming then minimizes. The β weights of the analytic expressions are defined as free variables with no restriction.

In this work, we use Single Node Genetic Programming (SNGP) (see Appendix for further details) to generate the non-linear analytic expressions $p_1(\cdot), \ldots, p_q(\cdot)$, which are then evaluated using (21). The whole process repeats until a stopping criterion is satisfied, such as a prescribed number of iterations or an improvement threshold.

4. EXPERIMENTAL EVALUATION

4.1 Benchmarks

The proposed method has been tested on two different benchmarks: the well-known pendulum swing-up and a nonlinear control problem called magnetic manipulation. Both of them are explicitly discussed in (Alibekov et al., 2018), including all necessary mathematical details. Here, only a high-level description of these problems is presented.

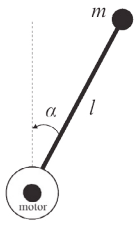
The first task is the classic under-actuated pendulum swing-up problem (abbreviated as pendulum), schematically depicted in Fig. 2a. The inverted pendulum consists of an actuated link that rotates in a vertical plane, and a weight of mass m attached to it. The motor torque is not sufficient to push the pendulum up in a single rotation. Instead, from some states, the pendulum needs to gather energy by swinging back and forth, in order to be pushed up and stabilized. The control goal is to stabilize the pendulum in the upright position $x_{des} = [0, 0]$ (rad, rad/s), which is formally described by the following reward function:

$$\rho(f(x,u)) = -abs(x_{des}^T - f(x,u))Q \tag{22}$$

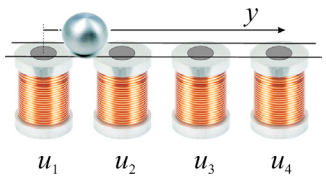
with $Q = [1, 0.1]^T$ a weighting vector to adjust the relative importance of the angle and angular velocity and $abs(\cdot)$ function working element-wise.

Magnetic manipulation (abbreviated as magman) is a benchmark of contactless manipulation with applications at micro scale. Our magnetic manipulation setup (see

[1] The purpose of ϵ is to make $O\beta$ strictly smaller than zero. From the practical point of view we recommend to choose the value with respect to the constraint tolerance the particular solver supports. In this paper, we have chosen $\epsilon = 0.001$.



(a) Pendulum task



(b) Magnetic manipulation task

Fig. 2. In the pendulum swing-up benchmark, pendulum needs to gather energy by swinging back and forth, in order to be pushed up and stabilized by controlling the torque applied by a motor. The goal of magnetic manipulation benchmark is to accurately position a steel ball on a 1-D track by dynamically shape the magnetic field above the magnets using current.

Table 1. Experiment parameters

Fuzzy V-iteration parameters: pendulum	
State space, $\mathcal{X}$	$[-\pi, \pi] \times [-30, 30]$
Input space, $\mathcal{U}$	$[-2, 2]$
State samples per dimension, B_X	$[21, 21]$
Action samples per dimension, B_U	11
Discount factor, γ	0.95
Convergence threshold, ϵ	10^{-4}
Desired state, x_{des}	$[0, 0]^T$
Sampling period, T_s [s]	0.02
Simulation time, T_{sim} [s]	3
Fuzzy V-iteration parameters: magman	
State space, $\mathcal{X}$	$[0, 0.05]\times$ $[-0.4, 0.4]$
Input space, $\mathcal{U}$	$[0, 0.6]$
State samples per dimension, B_X	$[21, 21]$
Action samples per dimension, B_U	$[3, 3]$
Discount factor, γ	0.99
Convergence threshold, ϵ	10^{-8}
Desired state, x_{des}	$[0.01, 0]^T$
Sampling period, T_s [s]	0.01
Simulation time, T_{sim} [s]	3
SNGP parameters	
Population size	1000
Elementary functions	+, -, ×, x^2, x^3, BentGeneral, Logistic3
Maximal depth of features	5
Maximal number of features	30
Epoch length	500
Local search iterations	500
Number of epochs	1
Number of threads	1

Fig. 2b) consists of four electromagnets in a line, but in this work only two of them have been used. The goal is to accurately position a steel ball on a 1-D track by dynamically shape the magnetic field above the magnets using current. This goal is formally described by the following reward function:

$$\rho(f(x,u)) = -abs(x_{des}^T - f(x,u))Q \tag{23}$$

with $Q = [10, 5]^T$ where the desired position x_{des} is set to $[0.01, 0]$ (m, m/s), Q is again a weighting vector and function $abs(\cdot)$ again works element-wise.

4.2 Testing procedure

For each benchmark, 30 different proxy functions have been constructed in 30 independent SNGP runs. Each function has been tested on each benchmark with $N = 100$ randomly chosen initial states via simulations. It should be noted that all the functions are presented "as is", which means that there is no selection procedure w.r.t. to some criterion.

Each result is compared with the baseline solution (4) (using the same initial states), which is computed beforehand by means of the fuzzy V-iteration algorithm. To evaluate each of proxy functions, the following criteria are defined:

- Improvement percentage

$$I = \left[\frac{1}{N} \sum_{j=1}^{N} \left[p^j_{baseline} / p^j_{method} \right] \cdot 100 \right] - 100\%$$

  where $p_{method} = \sum_{k=1}^{T_{sim}/T_s} \rho(f(x_k, u_k))$, with T_{sim} stands for the total simulation time, T_s is the sampling period, and *method* represents either *baseline* or *proxy* solution. The reward functions are defined to have maximum value zero in the goal state and to be negative otherwise. Therefore I equals 0% for the baseline and it is bigger than 0% if the proxy function outperforms the baseline approach.
- Mean distance between the last state (at the end of simulation) x_{end} and the desired goal state x_{des}

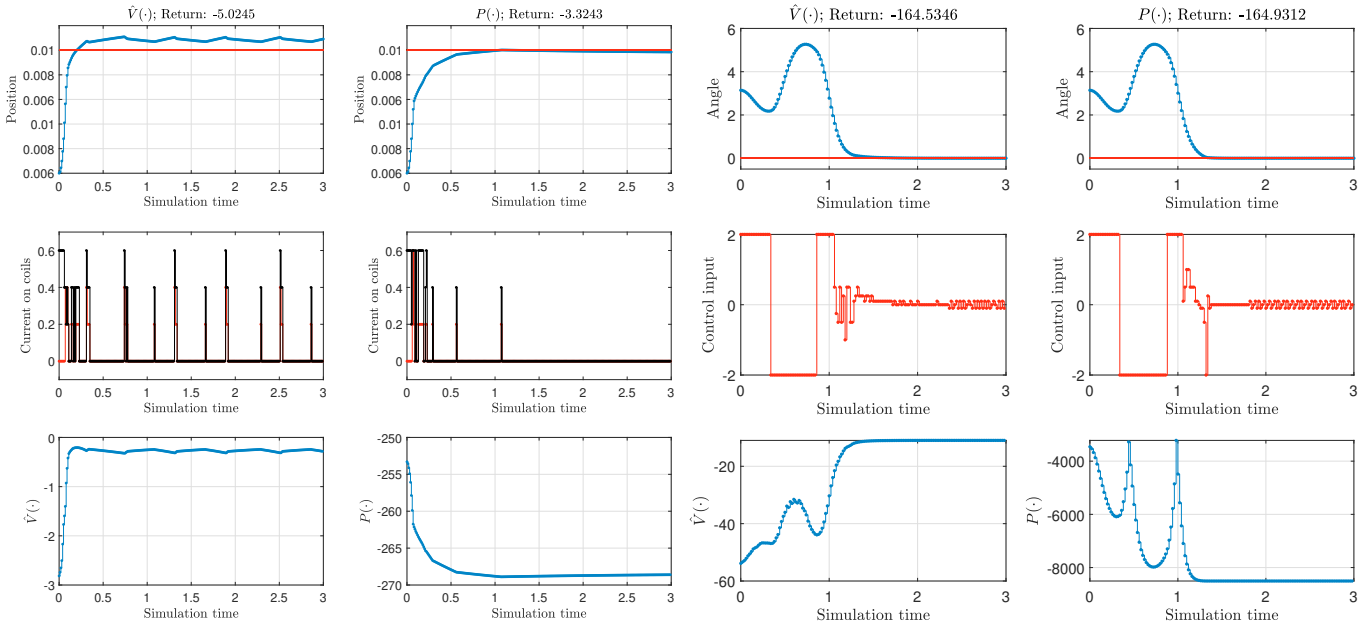
$$D = \frac{1}{N} \|x_{des} - x_{end}\|$$

  where $\|\cdot\|$ is the Mahalanobis norm.

The sampling parameters for both benchmarks, as well as SNGP parameters, are listed in Table 1.

5. RESULTS AND DISCUSSION

The simulation results for both tasks are listed in Table 2. The proposed method shows the potential to significantly outperform the baseline solution. According to the experiments, proxy functions demonstrate significant improvement in a range of 5%-82% w.r.t. the baseline. In the vast majority of cases, it is caused by alleviating of numerical artifacts. An example of it is depicted in Fig. 3a. The example demonstrates the comparison of derived policies for the magman task. The left column corresponds to the baseline value function, computed by fuzzy V-iteration algorithm, while the right column stands for one of the proxy functions. It can be seen that proxy function significantly alleviates steady-state error caused by artifacts. Another interesting note is that proxy function remarkably violates Lyapunov stability condition, as shown by the bottom row of Fig. 3a. The reason is that proxy function considers solely local properties of its surface, neglecting global geometry. However, for the pendulum task, the proposed method demonstrates a modest improvement percentage ratio. The reason for it is a small influence of the numerical artifacts on the policy. The comparison between derived policies using policy and baseline methods, respectively, is depicted in Fig. 3b. It can be seen, that, e.g., the steady-state error for both methods is relatively small (in comparison with magman task), which means that there is a small room for improvement for the proxy method.



(a) Magnetic manipulation task

(b) Pendulum swing-up task

Fig. 3. Example of the transient response using the original V-function $\hat{V}(x)$ and one of the proxy functions. The first row represents the position of the ball for the magman task and the angle for the pendulum task; the second stands for the control inputs; the last row shows changing a value of either value function or proxy function.

Table 2. Experimental study statistics

		Median D	Median I
Pendulum	Baseline	0.003	0%
	Proxy	0.002	5%
Magman	Baseline	0.038	0%
	Proxy	0.007	82%

The proposed method has several limitations. First of all, with the proposed design it is not possible to penalize input chattering. Input chattering can usually be reduced by penalizing the control input in the reward function, which is, however, not possible here due to the choice of the proxy function structure as $P(f(x,u))$. One way to overcome this limitation is to reformulate the proxy function as $P(x,u)$ and then use it in policy derivation in the same way as a Q-function. Another possible way is to combine MSE-like fitness and proxy function fitness in order to represent a trade-off between global V-function geometry and proxy local properties. This may be a part of our future work.

6. CONCLUSION

The proposed method offers an alternative way for the policy derivation. Instead of using V-function directly, the proposed approach build a smooth proxy function on top of it, from which the better policy can be derived. The proposed method may be combined with any kind of value function approximation. Moreover, due to the analytic nature of the proxy function, it can be combined with policy derivation methods (Alibekov et al., 2018) to further policy improvement.

ACKNOWLEDGEMENTS

This research was supported by the Grant Agency of the Czech Republic (GAČR) with the grant no. 15-22731S titled "Symbolic Regression for Reinforcement Learning in Continuous Spaces" and by the European Regional Development Fund under the project Robotics for Industry 4.0 (reg. no. CZ.02.1.01/0.0/0.0/15_003/0000470).

APPENDIX - SINGLE NODE GENETIC PROGRAMMING

This paper introduces an enhanced variant of Single Node Genetic Programming (SNGP) (Kubalík et al., 2017) to implement the symbolic regression. SNGP is a graph-based GP method that evolves a population of individuals, each consisting of a single program node. Our implementation differs from the above paper in the following aspects:

The following operators and functions are used to build analytic expressions:

$F = \{\times, +, -, x^2, x^3, \mathit{BentGeneral}, \mathit{Logistic3}\}$ where

$$\mathit{BentGeneral}(x_1,\ldots,x_N) = \prod_{i=1}^{N}\left[x_i + (\sqrt{x_i^2+1.0}-1)/2\right] \tag{24}$$

and

$$\begin{aligned}\mathit{Logistic3}(x_1,x_2,x_3) = x_1(1-(1/(1+e^{-x_3})))+\\ +x_2(1/(1+e^{-x_3}))\end{aligned} \tag{25}$$

and N is the arity of input.

Recalling that the result symbolic model $P(x)$ is composed from the linear combination of possibly non-linear analytic expressions $p_1,\ldots,p_q$, as:

$$P(x) = \beta_0 + \beta_1 p_1(x) + \beta_2 p_2(x) + \ldots + \beta_q p_q(x)$$

the following restrictions are applied in order to control overfitting:

- For every point in the dataset the result of an analytic expression lies within $[-10^8, 10^8]$ interval.

- Difference between the maximum and minimum values of an analytic expression computed on the given dataset lies within $[10^{-5}, 10^{3}]$ interval.
- Weights $\beta_1, \ldots, \beta_q$ lie within $[-1, 1]$ interval.

REFERENCES

Alibekov, E., Kubalík, J., and Babuška, R. (2016). Symbolic method for deriving policy in reinforcement learning. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, 2789–2795. IEEE.

Alibekov, E., Kubalík, J., and Babuška, R. (2018). Policy derivation methods for critic-only reinforcement learning in continuous spaces. *Engineering Applications of Artificial Intelligence*, 69, 178–187.

Buşoniu, L., Ernst, D., De Schutter, B., and Babuška, R. (2011). Cross-entropy optimization of control policies with adaptive basis functions. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 41(1), 196–209.

Busoniu, L., Babuska, R., De Schutter, B., and Ernst, D. (2010). *Reinforcement learning and dynamic programming using function approximators*, volume 39. CRC press.

Deng, Y., Bao, F., Kong, Y., Ren, Z., and Dai, Q. (2017). Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3), 653–664.

Ernst, D., Geurts, P., and Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6, 503–556.

Grondman, I., Vaandrager, M., Buşoniu, L., Babuška, R., and Schuitema, E. (2012). Efficient model learning methods for actor–critic control. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 42(3), 591–602.

Guan, C., Wang, Y., Lin, X., Nazarian, S., and Pedram, M. (2015). Reinforcement learning-based control of residential energy storage systems for electric bill minimization. In *Consumer Communications and Networking Conference (CCNC), 2015 12th Annual IEEE*, 637–642. IEEE.

Konda, V. and Tsitsiklis, J. (2000). Actor-critic algorithms. In *SIAM Journal on Control and Optimization*, 1008–1014. MIT Press.

Kubalík, J., Derner, E., and Babuška, R. (2017). Enhanced symbolic regression through local variable transformations. In *Proceedings of the 9th International Joint Conference on Computational Intelligence - Volume 1: IJCCI,*, 91–100. INSTICC, SciTePress. doi: 10.5220/0006505200910100.

Kuvayev, L. and Sutton, R.S. (1996). Model-based reinforcement learning with an approximate, learned model. In *Proc. Yale Workshop Adapt. Learn. Syst*, 101–105.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. ArXiv:1509.02971 [cs.LG].

Polydoros, A.S. and Nalpantidis, L. (2017). Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent and Robotic Systems*, 86(2), 153–173.

Sahba, F., Tizhoosh, H.R., and Salama, M.M. (2006). A reinforcement learning framework for medical image segmentation. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, 511–517. IEEE.