Reinforcement Learning Reference Sheet

Constantin Cronrath, *Department of Electrical Engineering, Chalmers*, December 2019

***Abstract*—This document is a brief, and not at all comprehensive, overview of Reinforcement Learning (RL). It merely aims to serve as a starting point for further endeavours into the field.**

I. Foundations of RL

A. The MDP Model

An *Markov Decision Process* (**MDP**) is a tuple $\mathcal{M} = \langle \mathcal{X}, \mathcal{U}, \rho, f, \gamma \rangle$, where $\mathcal{X}$ is the set of states (*state space*), $\mathcal{U}$ is the set of actions (*action space*), $\rho(x,u) : \mathcal{X} \times \mathcal{U} \mapsto \mathscr{P}(\mathbb{R})$ is the stochastic reward function for each $(x,u) \in \mathcal{X} \times \mathcal{U}$, $f(x,u) : \mathcal{X} \times \mathcal{U} \mapsto \mathscr{P}(\mathcal{X})$ is the stochastic transition function for each $(x,u) \in \mathcal{X} \times \mathcal{U}$ (i.e. *dynamics model* $\mathrm{X}_{k+1} = f(x_k, u_k)$), γ is a discount factor (alternatively: planning horizon H). In the **MDP** model, time advances in discrete steps $k = 0, 1, 2, \ldots$. Each state has the Markov property, meaning the state definition contains all information necessary to predict the associated reward r_{k+1} and next state x_{k+1}. In other words, the state is representative of the control history.

B. The Objective

The goal in RL is to maximize the expected return R (i.e. expected future rewards):

$$\begin{aligned} \text{maximize} \quad & R = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{k+1} \right] \\ \text{subject to} \quad & \mathrm{X}_{k+1} = f(x_k, u_k). \end{aligned}$$

C. Key Formulas

The state-value function $V^\pi(x)$ of a control policy $\pi(x)$ is the expected return R if following π from state x. The state-action-value function $Q^\pi(x,u)$ of $\pi(x)$ is the expected return R if taking action u in state x and following π from the next state x' on-wards. Policies that achieve the highest expected return in all states are called the optimal policies π^* and yield the optimal value functions. Those can be defined recursively as the Bellman optimality equations:

$$V^*(x) = \max_{u \in \mathcal{U}} \sum_{x'} p^u_{xx'} \left[r^u_{xx'} + \gamma V^*(x') \right],$$

$$Q^*(x,u) = \sum_{x'} p^u_{xx'} \left[r^u_{xx'} + \gamma \max_{u \in \mathcal{U}} Q^*(x', u') \right],$$

where x' is the next state, $p^u_{xx'}$ is the probability of transitioning to x' from x after applying u, and $r^u_{xx'}$ is the reward received in that transition. These equations are at the core of most RL algorithms [1].

II. Taxonomy of RL Algorithms

RL algorithms can be primarily categorized into model-based (such as Dynamic Programming) and model-free methods. An exact (tabular) and an approximate ('deep') version exists for most algorithms. Model-free methods can be divided into temporal-difference (TD-learning) methods (such as Q-learning) that exploit the estimation errors in the value functions, and policy optimization methods (such as REINFORCE) that tune the parameters of the policy directly. Actor-Critic methods combine policy optimization and TD-learning. Further information can be found in [1] and [2].

Algorithm 1 Q-learning

```
Initialize $Q(x,u)$ arbitrarily
for all episodes do
    Initialize $x$
    for all steps of episode do
        Choose $u$ from $x$ using policy derived from $Q$
        Take action $u$, observe $r$, $x'$
        $Q(x,u) \leftarrow Q(x,u) + \alpha [r + \gamma \max_{u'} Q(x',u') - Q(x,u)]$
        $x \leftarrow x'$
    end for
end for
return  $\pi(u) = \arg\max_u Q(x,u)$
```

Algorithm 2 REINFORCE

```
Require: A differentiable policy parameterization $\pi(u|x,\theta)$, $\forall u \in \mathcal{U}, x \in \mathcal{X}, \theta \in \mathbb{R}^d$
Initialize $\pi$ with random weights $\theta$
repeat
    Generate trajectory $x_0, u_0, r_1, \ldots, x_{K-1}, u_{K-1}, r_K$ following $\pi$
    for all steps of episode $k = 0, \ldots, K-1$ do
        $G \leftarrow$ return from step $k$
        $\theta \leftarrow \theta + \alpha \gamma^k G \nabla_\theta \log \pi(u_k|x_k,\theta)$
    end for
until $\mathbb{F}$
return  $\pi(\cdot|\cdot,\theta)$
```

III. Issues

RL can solve some incredibly hard control problems, but: the generic, model-free approach must be paid for in data samples; designing a good real-world, control problem for RL is still not easy; synthesising an RL policy is not guaranteed to work; RL policies can lack robustness against disturbances; and model-based control (e.g. LQR) can often perform equally well out of the box.

A. The Curse of Dimensionality

States are usually described using state features (e.g. $\mathbf{x} \in \mathbb{R}^d$). $|\mathcal{X}|$ is exponential in the dimensionality d of the state features. The sample complexity of model-free RL algorithms, hence, becomes worse with each additional state feature.

B. Partial Observability

To optimally solve a partially observable MDP, the state definition must be the history. Any finite horizon RL problem, therefore, becomes an infinite horizon RL problem, making learning much harder (because no state is visited more than once) [3].

C. The Exploration/Exploitation Dilemma

The controller must explore the system to acquire new information, but it must also exploit what it knows to behave optimally. The exploration/exploitation dilemma is a *fundamental* problem in RL. Even with the best possible exploration strategy, model-free RL algorithms will always need a minimum amount of exploration, during which the controller acts sub-optimally.

D. The Deadly Triad

The deadly triad is the combination of: function approximation (e.g. neural nets), bootstrapping (e.g. TD-learning), and off-policy learning (training on data collected under a different policy). All three are used in most deep RL algorithms, but theoretically can lead to divergence. In practice, a less severe form of divergence occurs [4].

E. Distributional Shift

This denotes the issue of changes in the underlying distributions of the system (e.g. transition probabilities change due to disturbances). No established method exists to detect such shifts, rendering the control policy sub-optimal when they occur [5].

F. Reward Hacking

Reward function design is difficult [6]. A poorly specified reward function might cause the RL algorithm to game the reward function instead of solving the problem [5].

G. Final vs Transient Performance

RL algorithms are often judged on their final performance after millions of timesteps. In a real-world setting the transient performance, while learning, might be more important, since sampling data from the system is expensive [5] [7] [8] [9].

H. Interpretability

A controller in form of a neural net is hard to interpret or analyze.

I. Reproducibility

Results achieved by RL algorithms can be hard to reproduce due to their sensitivity to for example: random seeds, hyperparameter settings, implementation details, reward design or local optima [10].

IV. Considering Applying RL

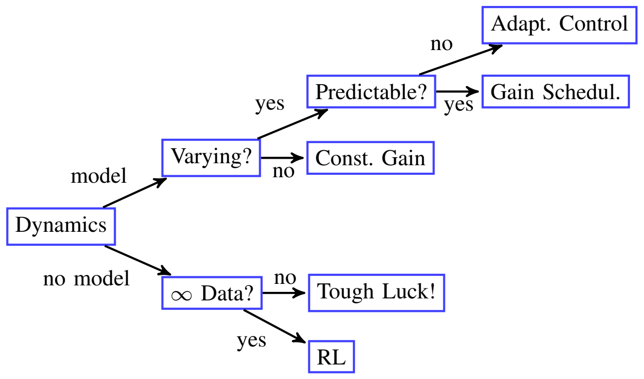

Fig. 1. Decision tree adapted from [11]

V. Further Information

A. Blog Posts

1) [9] contrasts RL with control theory.
2) [6] highlights problems in deep RL.
3) [2] provides an excellent starting point into deep RL.

B. Papers

1) [7] describes how RL can be applied in robotics.
2) [8] surveys safe RL methods.
3) [10] criticises how RL results are reported.
4) [12] compares some RL algorithms for continuous control.
5) [13] compares recent model-based RL algorithms.
6) Two surveys of deep RL: [14] [15]

C. Books

The three most cited books in RL are in descending order: Sutton & Barto [1], Puterman [16], Bertsekas [17].

D. Online Courses

1) Comprehensive 'classical' RL course based on [18] by the mind behind AlphaGo when he was at UCL: [19]
2) Deep RL course by one of the strongest academic groups: [20]
3) Control oriented PhD course based on [17]: [21]

References

[1] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018. [Online]. Available: http://incompleteideas.net/sutton/book/the-book-2nd.html

[2] Open AI, "Spinning Up in Deep RL!" [Online]. Available: https://spinningup.openai.com/en/latest/

[3] K. Åström, "Optimal control of Markov processes with incomplete state information," *Journal of Mathematical Analysis and Applications*, vol. 10, no. 1, pp. 174–205, 2 1965. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0022247X6590154X

[4] H. van Hasselt, Y. Doron, F. Strub, M. Hessel, N. Sonnerat, and J. Modayil, "Deep Reinforcement Learning and the Deadly Triad," 12 2018. [Online]. Available: http://arxiv.org/abs/1812.02648

[5] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete Problems in AI Safety," 6 2016. [Online]. Available: http://arxiv.org/abs/1606.06565

[6] A. Irpan, "Deep Reinforcement Learning Doesn't Work Yet," 2018. [Online]. Available: https://www.alexirpan.com/2018/02/14/rl-hard.html

[7] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 9 2013. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364913495721

[8] J. Garcia and F. Fernandez, "A Comprehensive Survey on Safe Reinforcement Learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015. [Online]. Available: http://www.jmlr.org/papers/volume16/garcia15a/garcia15a.pdf

[9] B. Recht, "An Outsider's Tour of Reinforcement Learning – arg min blog," 2018. [Online]. Available: http://www.argmin.net/2018/06/25/outsider-rl/

[10] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep Reinforcement Learning that Matters," 9 2017. [Online]. Available: http://arxiv.org/abs/1709.06560

[11] K. J. Astrom and B. Wittenmark, *Adaptive Control*, 2nd ed. Addison-Wesley, 1995.

[12] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking Deep Reinforcement Learning for Continuous Control," 4 2016. [Online]. Available: http://arxiv.org/abs/1604.06778

[13] T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, "Benchmarking Model-Based Reinforcement Learning," 7 2019. [Online]. Available: http://arxiv.org/abs/1907.02057

[14] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," pp. 26–38, 8 2017. [Online]. Available: https://arxiv.org/abs/1708.05866

[15] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and I. Palunko, "Reinforcement learning for control: Performance, stability, and deep approximators," pp. 8–28, 1 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1367578818301184

[16] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 2005.

[17] D. P. Bertsekas, *Reinforcement Learning and Optimal Control*. Athena Scientific, 2019.

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

[19] D. Silver, "UCL Course on RL," 2015. [Online]. Available: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

[20] S. Levine, J. Schulman, and C. Finn, "CS 294: Deep Reinforcement Learning, Spring 2017," 2017. [Online]. Available: http://rll.berkeley.edu/deeprlcoursesp17/

[21] D. P. Bertsekas, "Reinforcement Learning and Optimal Control Course at ASU," 2019. [Online]. Available: http://web.mit.edu/dimitrib/www/RLbook.html