AI-Toolbox: A Framework for Fundamental Reinforcement Learning*

Eugenio Bargiacchi[1], Diederik M. Roijers[1,2], and Ann Nowé[1]

[1] Department of Computer Science, Vrije Universiteit Brussel
[2] Microsystems Technology, HU University of Applied Sciences
svalorzen@gmail.com

As is generally true for computer science, reinforcement learning (RL) curricula are mainly composed of two parts: theory, and hands on experience. The latter, in particular, often requires the students to implement the most fundamental algorithms from scratch. Students can then test these on toy examples in order to gain an intuitive understanding of the underlying mechanisms, e.g. value functions, belief updates, etc. While useful, this approach can be time-consuming, which prevents students from experimenting with less known or harder to implement methods and comparing their characteristics directly. To counteract this, providing a suite of already implemented methods can help to significantly expand the experience of a novice.

With the recent surge of interest in the field of deep learning, there are numerous resources for implementing and testing deep methods today. However, we argue that deep methods are not optimal for (human) learning, as it is difficult to inspect their internals and get an intuitive understanding of what is happening under the hood. Unfortunately, relatively few libraries organize discrete literature that can be used for this purpose [7, 6, 4].

AI-Toolbox[3] is a C++\Python library that tries to fill this gap. It is a research-grade repository of more than 40 implementations of several algorithms for single-agent and multi-agent bandit, MDP and POMDP algorithms, and also provides a large number of utility classes and functions available to implement additional methods. AI-Toolbox is one of the largest frameworks of its type available online, and it is free software.

AI-Toolbox is battle-tested, and has been referenced by numerous publications [9, 2, 8, 5, 1]. The goals of this framework are, in descending order of importance: usability and documentation, ease of modification, clarity and performance. These goals align well with student experimentation and discovery.

AI-Toolbox extensive documentation covers every public class, method and utility. The library provides a uniform, consistent interface throughout, emphasizing patterns across algorithms that might not otherwise be noticed. The code is written in C++17, taking advantage of all features of the language, and is built following modern standard practices, i.e. unit tests, continuous integration, separate concerns, etc.

To give a brief example of how easy it is to use the library, this is all the code needed to use the Incremental Pruning algorithm to solve the known POMDP

* Supported by FWO (Fonds Wetenschappelijk Onderzoek), grant #1SA2820N, and by the “Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen” programme.

tiger problem, and initialize a policy with the resulting value function. The policy can then be easily used to act in the environment; for example, to control a robot.

```cpp
// The model can be any custom class that respects a 10-method interface.
// In this case it is a problem provided by the toolbox.
auto model = AIToolbox::POMDP::makeTigerProblem();
unsigned horizon = 10; // The horizon of the solution.

// The 0.0 is the convergence parameter. It gives a way to stop the
// computation if the policy has converged before the horizon.
AIToolbox::POMDP::IncrementalPruning solver(horizon, 0.0);

// Solve the model and obtain the optimal value function.
auto [bound, valueFunction] = solver(model);

// We create a policy from the solution to compute the agent's actions.
// The parameters are the size of the model (SxAxO), and the value function
// obtained from solving the problem.
AIToolbox::POMDP::Policy policy(2, 3, 2, valueFunction);
```

References

1. Arming, S., Bartocci, E., Chatterjee, K., Katoen, J.P., Sokolova, A.: Parameter-independent strategies for pMDPs via POMDPs. In: International Conference on Quantitative Evaluation of Systems. pp. 53–70. Springer (2018)
2. Bargiacchi, E., Verstraeten, T., Roijers, D., Nowé, A., van Hasselt, H.: Learning to coordinate with coordination graphs in repeated single-stage multi-agent decision problems. In: International Conference on Machine Learning. vol. 80, pp. 482–490. PMLR (10–15 Jul 2018)
3. Bargiacchi, E., Roijers, D.M., Nowé, A.: AI-Toolbox: A C++ library for reinforcement learning and planning (with Python bindings). Journal of Machine Learning Research **21**(102), 1–12 (2020), http://jmlr.org/papers/v21/18-402.html
4. Chades, I., Chapron, G., Cros, M., Garcia, F., Sabbadin, R.: MDPtoolbox: a multi-platform toolbox to solve stochastic dynamic programming problems. In: Ecography. vol. 37, pp. 916–920 (2014)
5. Chatterjee, K., Elgyütt, A., Novotný, P., Rouillé, O.: Expectation optimization with probabilistic guarantees in pomdps with discounted-sum objectives. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18. pp. 4692–4699. International Joint Conferences on Artificial Intelligence Organization (7 2018). https://doi.org/10.24963/ijcai.2018/652, https://doi.org/10.24963/ijcai.2018/652
6. Egorov, M., Sunberg, Z., Balaban, E., Wheeler, T., Gupta, J., Kochenderfer, M.: POMDPs.jl: A framework for sequential decision making under uncertainty. Journal of Machine Learning Research **18**(26), 1–5 (2017), http://jmlr.org/papers/v18/16-300.html
7. Oliehoek, F., Spaan, M.T.J., Terwijn, B., Robbel, P., Messias, J.V.: The MADP toolbox: An open-source library for planning and learning in (multi-)agent systems. Journal of Machine Learning Research **18**(1), 3112–3116 (2017)
8. Petrák, J.: Maze Navigation via Monte Carlo Tree Search. Ph.D. thesis, Masaryk University (2019)
9. Verstraeten, T., Bargiacchi, E., Libin, P., Helsen, J., Roijers, D.M., Nowé, A.: Multi-agent Thompson sampling for bandit applications with sparse neighbourhood structures. Nature Scientific Reports **10**(1), 6728 (2020)