Summary of Part I: Reinforcement Learning in Finite State and Action Spaces

Oliver Wallscheid
Common Key Ideas to all Discussed RL Methods so far

1. Estimating and comparing value functions
2. Backing up values along actual or possible state trajectories
3. Usage of GPI mechanism to maintain an approximate value function and policy trying to improve each of them on the basis of the other

Fig. S-I.1: Generalized policy iteration (GPI) as a mutual building block of all previously discussed RL methods (source: R. Sutton and G. Barto, Reinforcement learning: an introduction, 2018, CC BY-NC-ND 2.0)
Figure 8.11: A slice through the space of reinforcement learning methods, highlighting the two of the most important dimensions explored in Part I of this book: the depth and width of the updates.

Ranging from one-step TD updates to full-return Monte Carlo updates. Between these is a spectrum including methods based on $n$-step updates (and in Chapter 12 we will extend this to mixtures of $n$-step updates such as the $\lambda$-updates implemented by eligibility traces).

Dynamic programming methods are shown in the extreme upper-right corner of the space because they involve one-step expected updates. The lower-right corner is the extreme case of expected updates so deep that they run all the way to terminal states (or, in a continuing task, until discounting has reduced the contribution of any further rewards to a negligible level). This is the case of exhaustive search. Intermediate methods along this dimension include heuristic search and related methods that search and update up to a limited depth, perhaps selectively. There are also methods that are intermediate along the horizontal dimension. These include methods that mix expected and sample updates, as well as the possibility of methods that mix samples and distributions within a single update. The interior of the square is filled in to represent the space of all such intermediate methods.

A third dimension that we have emphasized in this book is the binary distinction between on-policy and $\pi$-policy methods. In the former case, the agent learns the value function for the policy it is currently following, whereas in the latter case it learns the value function for a target policy, possibly following a different one.

Fig. S-I.2: A slice through the RL method space (source: R. Sutton and G. Barto, Reinforcement learning: an introduction, 2018, CC BY-NC-ND 2.0)
Other Important RL Dimensions

Selected, non-exhaustive list:

- **Problem space**: How many states and actions? Stochastic vs. deterministic environment? Stationary?
- **Policy objective**: on-policy vs. off-policy? Explicit vs. implicit policy?
- **Task**: Episodic vs. continuing?
- **Return definition**: Discounting? General reward design?
- **Value**: State vs. action value estimation?
- **Model**: Required? Distribution vs. sample models? Learning vs. a priori (expert) knowledge?
- **Exploration**: How to search for new policies?
- **Update order**: synchronous vs. asynchronous? If latter, which order?
- **Experience**: simulated vs. real experience? Memory length and style?
- ...
First part of the course:

**Reinforcement learning on small finite action and state spaces**

The problem space is such small that RL methods based on look-up tables are applicable.

Second part of the course:

**Reinforcement learning using function approximators**

The problem space is either continuous or contains an unfeasible large amount of discrete state-action pairs. Value estimates, models or explicit policies stored in look-up tables would let the memory demand explode. Modifications and extensions of available RL algorithms using function approximators are required.
The End for Today

Thanks for your attention and have a nice week!