Using Advice in Model-Based Reinforcement Learning
Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, Sheila A. McIlraith
Department of Computer Science, University of Toronto
{rmitoro, toryn, rvalenzano, sheila}@cs.toronto.edu

Abstract. When a human is mastering a new task, they are usually not limited to exploring the environment, but also avail themselves of advice from other people. In contrast to constraints, advice is merely a recommendation about how to act that may be of variable quality or incomplete. In this work, we consider the use of advice expressed in Linear Temporal Logic to guide exploration in a model-based reinforcement learning algorithm. Our experimental results demonstrate the potential for good advice to significantly reduce the number of training steps needed to learn strong policies, while still maintaining robustness in the face of incomplete or misleading advice.

Motivation
When RL agents learn behavior
• Must explore environment to learn how to act
• This process can be prohibitively expensive

When people learn behavior
• Use different sources of info besides just experience, including advice from other people
• Guidance concerning prudent future action
• May be of variable quality or incomplete

Research question: How can an RL agent take advantage of linguistically expressed advice?

Advice-Based Exploration
Our method turns advice into a way to guide exploration. To do so, we need to address two problems.

Problem 1: Communicating with the agent.
We use a signature $Σ$ to define the predicates and constants that can be referred to when giving advice. A labelling function $I$ identifies what is true in any state.

$$e.g. \text{at}(c) \in I(x)$$

Problem 2: Satisfying a given advice formula.
The agent uses a background knowledge function, $h: S \times A \times X \rightarrow \mathbb{R}$, to estimate the number of actions needed to make a literal true.

$$e.g. h(s, a, \text{at}(c)) \text{ and } h(s, a, \neg\text{at}(c))$$

We extend this estimate to formulate as follows:

$$\hat{h}(s, a, \phi) = \min(h(s, a, \phi), h(s, a, \neg\phi))$$

This method can substantially improve performance in stochastic grid-world environments.

Figure 1: An example grid-world domain.

Actions: left, right, up, down
Rewards: door +1000; nail −10; step −1
MDP: $M = (S, A, γ, T, R)$

Model-Based RL
Idea: estimate $T$ and $R$ from experience (by counting):

$$\hat{R}(s, a) = \frac{1}{n_{a}(s)} \sum_{r=1}^{n_{a}(s)} \hat{T}(s', a) = \frac{n_{a}(s, s')}{n_{a}(s)}$$

• Compute $\hat{T}(s', a)$ using $\hat{R}(s, a)$ and $T(s', a)$.
• Execute an action following $\hat{T}(s', a)$.
• Update $\hat{T}$, $R$, and $n$, and repeat.

R-MAX: if $n(s, a) < m$, then assume $\hat{R}(s, a) = R_{\text{max}}$

A Language for Advice
Linear Temporal Logic (LTL) extends propositional logic with the following temporal operators: next ($\bigcirc \varphi$), always ($\bigcirc \varphi$), eventually ($\nu \varphi$), and until ($\varphi \Upsilon \psi$).

The following formulate state “get to the key and then open the door” and “always avoid the nails”, respectively:

$$\bigcirc(\text{at(key)} \land \bigcirc(\text{at(opendoor)}))$$

$$\bigcirc(\neg(\text{at(nail)}) \land \neg(\text{at(s))})$$

From LTL to NFAs
Any LTL formula can be transformed into an equivalent set of Non-Deterministic Finite Automatons (NFAs).

Advice was used in R-MAX as follows. If $n(s, a) < m$,

$$\hat{R}(s, a) = \begin{cases} 0 & (s', a') \notin T(s, a) \land \hat{h}(s', a') < \hat{h}(s, a) \\ R_{\text{max}} & \text{otherwise} \end{cases}$$

This method can substantially improve performance in deterministic grid-world maps when given advice.

Figure 2: NFAs for advice formulas (1) and (2).

Experiments with R-MAX
Advice can also be used provided to MBIE-EB [1].

$$\hat{Q}_{\text{MBIE}}(s, a) = \frac{R_{\text{max}}}{1 - \gamma} \left[ \hat{R}(s, a) - \gamma \sum_{s'} T(s', a) \max_{a'} \hat{Q}_{\text{MBIE}}(s', a') \right]$$

This algorithm was also shown to benefit from advice, in stochastic grid-world environments.

Figure 5: Results in the stochastic nail room.

Previous Work
Maclin & Shavlik (1996)
• IF
  An Enemy IS (Near and West) AND
  An Obstacle IS (Near and North)
• MULTIATION: MoveEast, MoveNorth
Maclin et al. (2005)
• IF (dist_goalcenter <= 15) AND
  (angle_goalcenter_you_goalie >= 25)
• THEN
  STICK trade to Pass
Krening et al. (2016)
• (Koopa, fireball), (Coin, JumpRight), ...

Conclusion
Our approach can use LTL advice to reduce the training required while being robust to misleading advice.

Future work
• Learn the background knowledge function.
• Extend to non-discrete domains.

References