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
{rntoro, 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

Advice

- 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?

Running Example

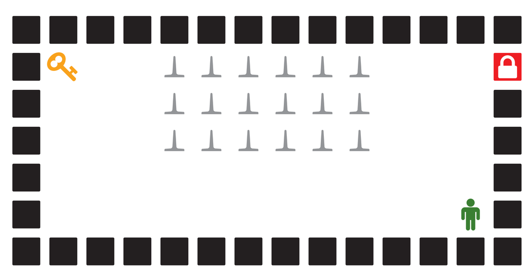

Figure 1: An example grid-world domain.

Actions: left, right, up, down
Rewards: door +1000; nail -10; step -1
MDP: $\mathcal{M} = \langle S, s_0, A, \gamma, T, R \rangle$

Model-Based RL

Idea: estimate T and R from experience (by counting).

$$\hat{R}(s,a) = \frac{1}{n(s,a)} \sum_{i=1}^{n(s,a)} r_i \qquad \hat{T}(s'|s,a) = \frac{n(s,a,s')}{n(s,a)}$$

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

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

A Language for Advice

Linear Temporal Logic (LTL) extends propositional logic with the following temporal operators: next ($\bigcirc\varphi$), always ($\square\varphi$), eventually ($\diamond\varphi$), and until ($\varphi_1 \,\mathsf{U}\, \varphi_2$).

The following formulae state "get to the key and then the door" and "always avoid the nails", respectively:

$$\diamond(\mathtt{at}(\mathtt{key}) \wedge \bigcirc\diamond(\mathtt{at}(\mathtt{door}))) \qquad (1)$$

$$\square(\forall(x \in \mathtt{nails}).\neg\mathtt{at}(x)) \qquad (2)$$

From LTL to NFAs

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

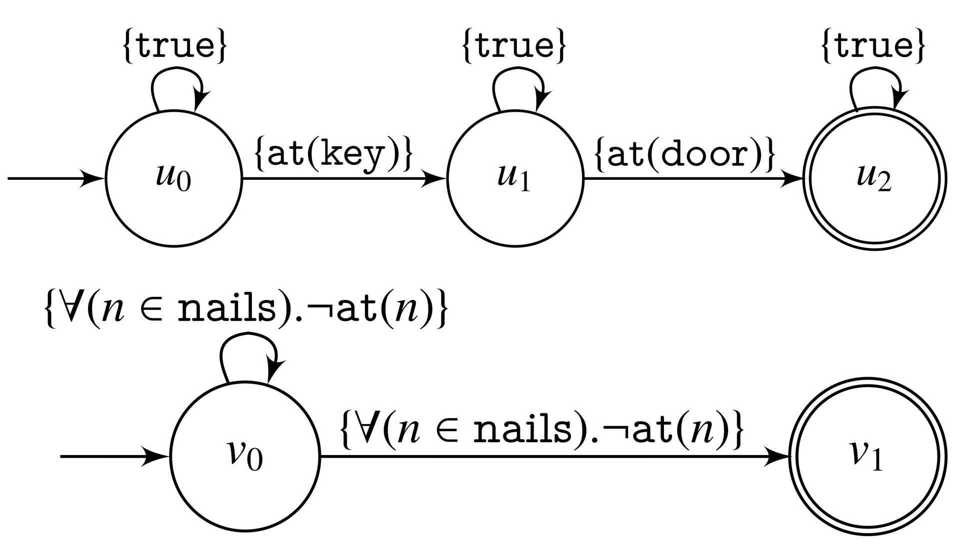

Figure 2: NFAs for advice formulas (1) and (2).

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 L identifies what is true in any state.

e.g. $\mathtt{at}(c) \in L(s)$

Problem 2: Satisfying a given advice formula.

The agent uses a background knowledge function, $h_B : S \times A \times \text{literals}(\Sigma) \to \mathbb{N}$, to estimate the number of actions needed to make a literal true.

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

We extend this estimate to formulae as follows:

$h : S \times A \times \mathcal{L}_\Sigma \to \mathbb{N}$

- $h(s,a,\ell) = h_B(s,a,\ell)$ for $\ell \in \text{literals}(\Sigma)$
- $h(s,a,\psi \wedge \chi) = \max\{h(s,a,\psi), h(s,a,\chi)\}$
- $h(s,a,\psi \vee \chi) = \min\{h(s,a,\psi), h(s,a,\chi)\}$

e.g. $h(s, a, \mathtt{at}(\mathtt{key}_1) \vee \mathtt{at}(\mathtt{key}_2))$

$$\hat{\varphi} = \bigvee_{i=0}^{m} \left[\bigvee_{(q,\Gamma,q') \in \delta^{(i)} \text{ and } (q,q') \in \mathit{useful}(q^{(i)})} \mathtt{to\_DNF}(\Gamma) \right]$$

$$\hat{\varphi}_w = \bigwedge_{i=0}^{m} \left[\bigvee_{q \in q^{(i)} \text{ and } (q,\Gamma,q') \in \delta^{(i)}} \mathtt{to\_DNF}(\Gamma) \right]$$

e.g. $\hat{\varphi} = \mathtt{at}(\mathtt{key})$; $\hat{\varphi}_w = \forall(x \in \mathtt{nails}).\neg\mathtt{at}(x)$

$$\hat{h}(s,a) = \begin{cases} h(s,a,\hat{\varphi}) & \text{if } h(s,a,\hat{\varphi}_w) = 0 \\ h(s,a,\hat{\varphi}) + 1000 & \text{otherwise} \end{cases}$$

Experiments with R-MAX

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

$$\hat{R}(s,a) = \begin{cases} 0 & \exists(s',a') \text{ s.t. } \hat{h}(s',a') < \hat{h}(s,a) \\ \mathrm{R}_{\max} & \text{otherwise} \end{cases}$$

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

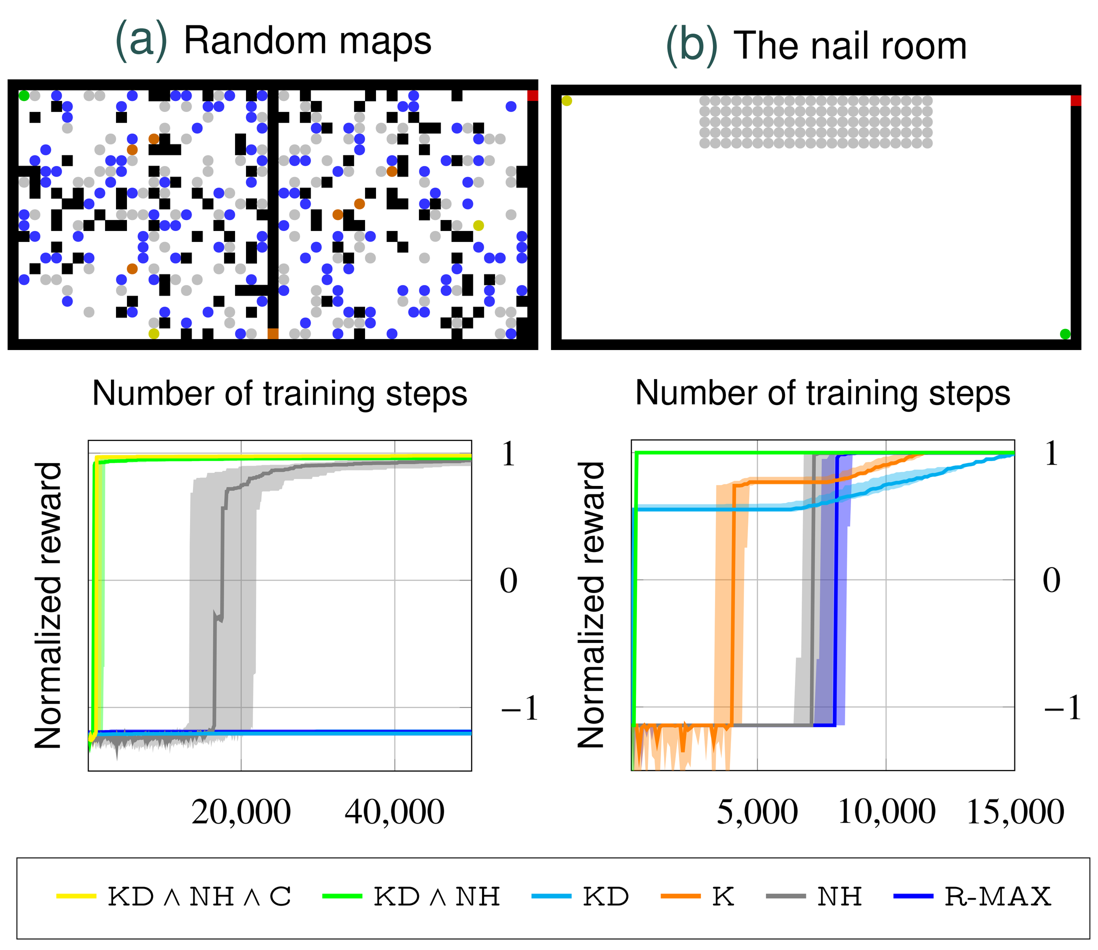

Figure 3: Results using advice in R-MAX.

Abbreviation	Advice formula (and informal meaning)
C	$\forall(c \in \mathtt{cookies}).\diamond\mathtt{at}(c)$ Get all the cookies.
NH	$\square(\forall(x \in \mathtt{nails} \cup \mathtt{holes}).\neg\mathtt{at}(x))$ Avoid nails and holes.
K	$\forall(k \in \mathtt{keys}).\diamond(\mathtt{at}(k))$ Get all the keys.
KD	$\forall(k \in \mathtt{keys}).\diamond(\mathtt{at}(k) \wedge \bigcirc\diamond(\exists(d \in \mathtt{doors}).\mathtt{at}(d)))$ For every key, get it and then go to a door.
Adversarial	$\forall(n \in \mathtt{nails}).\diamond\mathtt{at}(n)$ Step on every nail.

Figure 4: Advice formulae used in our experiments.

Experiments with MBIE-EB

Advice can also be used provided to MBIE-EB [1].

$$\hat{Q}_{\text{init}}(s,a) = \frac{\mathrm{R}_{\max}}{1-\gamma}(1-\alpha) + (-\hat{h}(s,a))\alpha$$

$$\hat{Q}_*(s,a) = \frac{\beta}{\sqrt{n(s,a)}} + \hat{R}(s,a)(1-\alpha) + (-1)\alpha + \gamma \textstyle\sum_{s'} \hat{T}(s'|s,a) \max_{a'} \hat{Q}_*(s',a')$$

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

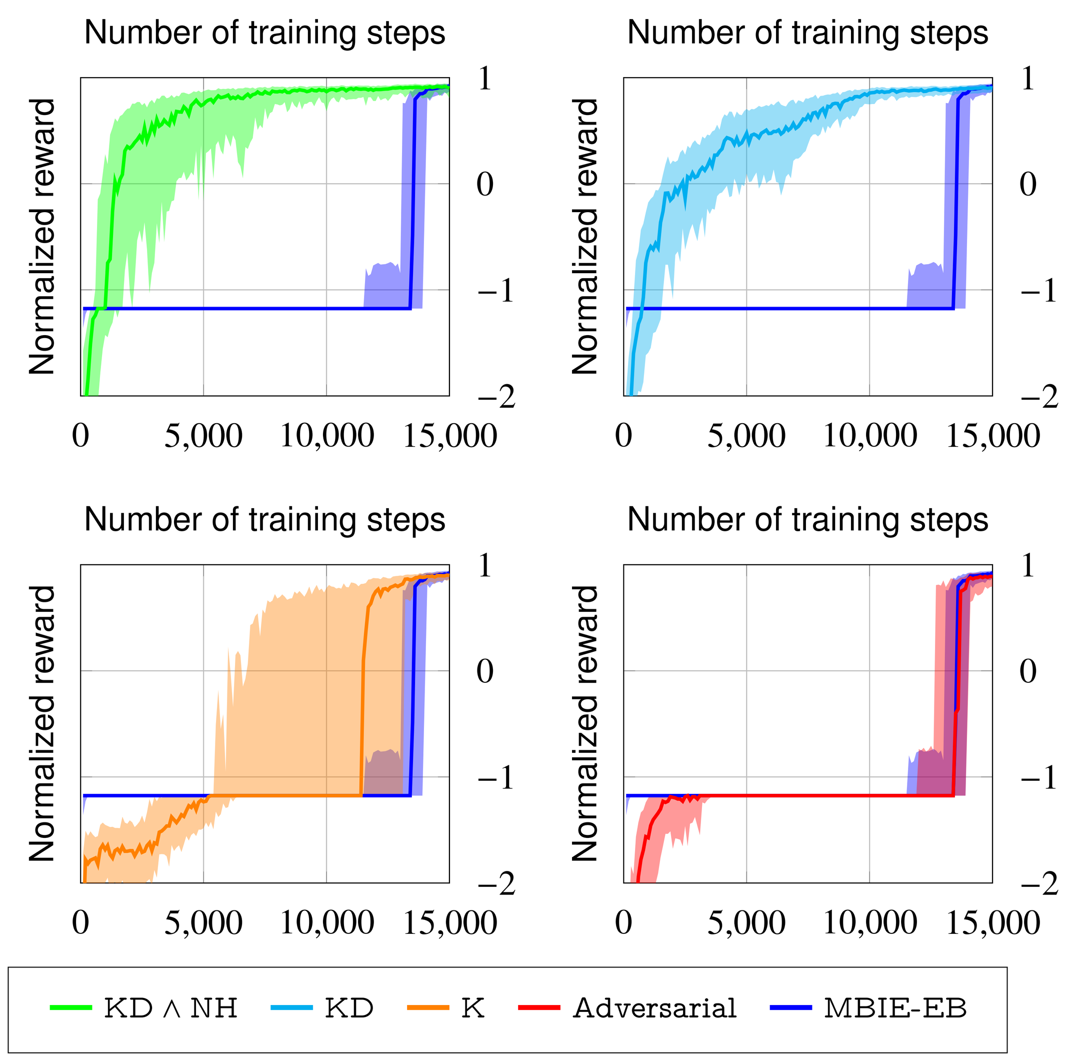

Figure 5: Results in the stochastic nail room.

Previous Work

Maclin & Shavlik (1996)

- IF An Enemy IS (Near and West) AND An Obstacle IS (Near and North)
  - MULTIACTION: MoveEast, MoveNorth

Maclin et al. (2005)

- IF (dist_goalcenter <= 15) AND (angle_goalcenter_you_goalie >= 25)
  - THEN PREFER Shoot 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

[1] Strehl, A. L., & Littman, M. L. (2008). An analysis of model-based interval estimation for Markov decision processes. *JCSS*.

[2] Maclin, R., & Shavlik, J. W. (1996). Creating advice-taking reinforcement learners. *Machine Learning*.

[3] Maclin, R., et al. (2005). Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. In *AAAI*.

[4] Krening, S., et al. (2016). Learning from Explanations using Sentiment and Advice in RL. *TCDS*.