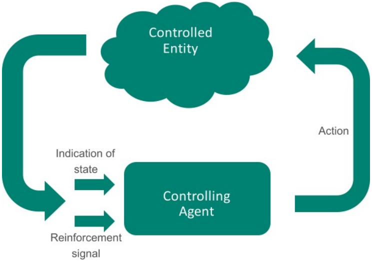
Robust Deep Reinforcement Learning

Shie Mannor

Technion - Israel Institute of Technology & Nvidia Research

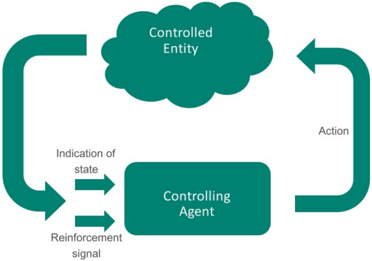
October, 2020

Model = Markov Decision Process (MDP) = $\{S, P, R, A\}$



Classical Reinforcement Learning

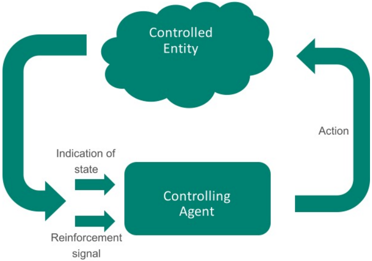
Model = Markov Decision Process (MDP) = $\{S, P, R, A\}$



States S, actions A are known and given
Transitions P and rewards R are not known.

Classical objective: $\max_\pi \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_{\mathbf{t}}\right]$, $\gamma < 1$

Model = Markov Decision Process (MDP) = $\{S, P, R, A\}$

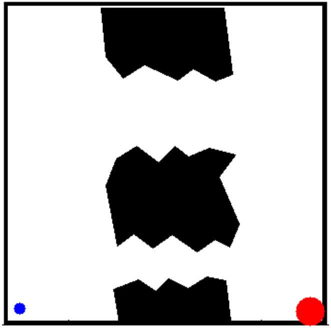


States S, actions A are known and given
Transitions P and rewards R are not known.

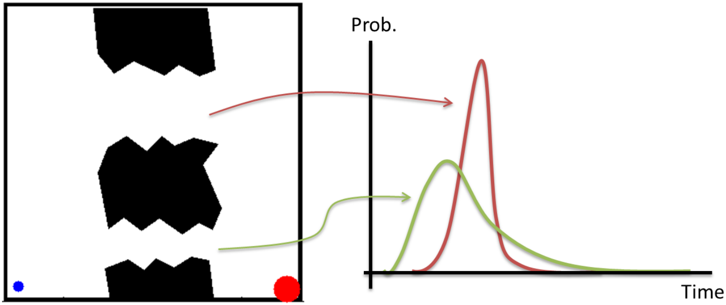
Classical objective: $\max_\pi \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t\right]$, $\gamma < 1$

"All models are wrong, but some are useful", G. Box

Why should we be robust?



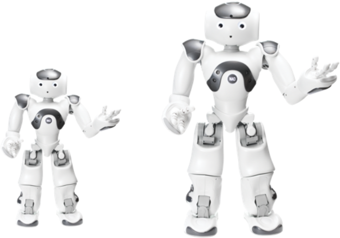
Why should we be robust?



Meaning of robustness

Abrupt disturbances

Model uncertainty



Three Types of Uncertainties

1. Parmeter uncertainty

- Uncertainty in MDP *parameters* (transitions, rewards)
- Objective:

$$\max_{\pi} \min_{P \in \text{possible MDP parameters}} \mathbb{E}^{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right]$$

- Origins in robust control

Three Types of Uncertainties

2. Inherent uncertainty

- Cumulative reward is stochastic
- Expectation does not capture *variability*
- Objective:

$$\max_{\pi} \rho \left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \right]$$

- ρ is a *risk measure*, e.g., $\rho(X) = \mathbb{E}[X] - \beta \mathrm{Var}[X]$
- Explicit safety against ‘unluckiness’
- Humans tend to be risk aware

Three Types of Uncertainties

3. Model uncertainty

- Model itself not known (observations/features/order)
- Objective:

$$\max_{\pi} \min_{\text{possible models}} \mathbb{E}_{model}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right]$$

- Model mismatch handled explicitly
- Origins in multi-model control

Applicability

When is robustness important?

- Cost of failure is high
  - Finance
  - Smart-grids
  - Health
  - Robotics (e.g., safety)
- Model is not known (always) and created from a few samples

Applicability

When is robustness important?

- Cost of failure is high
  - Finance
  - Smart-grids
  - Health
  - Robotics (e.g., safety)
- Model is not known (always) and created from a few samples

We desire:

- Scalability
- Adaptivity
- Accountability

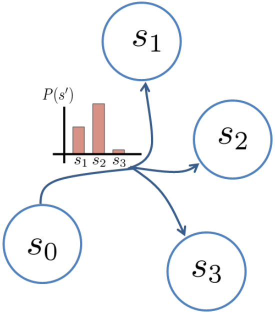
Robust MDPs with function approximation

A. Tamar, SM, and H. Xu, ICML 2014
A. Tamar, Y. Chow, M. Ghavamzadeh, and SM, NIPS 2015

Introduction - Planning with *Parameter* Uncertainty

Setting:

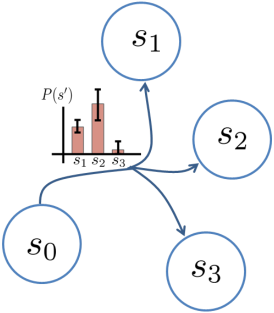
- Planning problem
- Uncertain transitions
  - Confidence intervals
  - Heuristic simulator
  - Time changing dynamics
  - etc.



Introduction - Planning with *Parameter* Uncertainty

Setting:

- Planning problem
- Uncertain transitions
  - Confidence intervals
  - Heuristic simulator
  - Time changing dynamics
  - etc.
- Potentially large impact [SM et. al, Management Science 2010]
  - Uncertainty amplification
  - Disasters / safety
  - Smart grids, finance

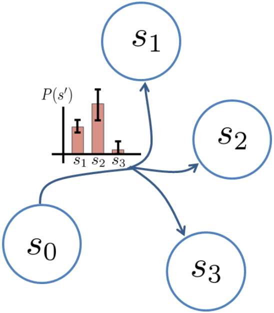


Background: Robust MDPs

Definitions:

- Robust Markov decision processes:
- State, actions and rewards as in the standard model
- Transitions $P(s'|s,a) \in \mathcal{P}$
- Policy π
- Worst-case objective

$$\sup_{\pi} \inf_{P \in \mathcal{P}} \mathbb{E}^{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right]$$



Background: Robust MDPs

Dynamic programming solution

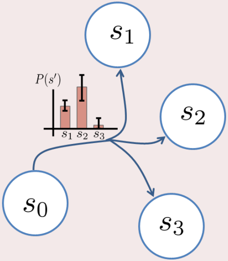
- Robust value function (fixed policy)

$$V^\pi(s) \doteq \inf_{P \in \mathcal{P}} \mathbb{E}^{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) | s_0 = s\right]$$

Dynamic programming solution

- Robust value function (fixed policy)

$$V^{\pi}(s) \doteq \inf_{P \in \mathcal{P}} \mathbb{E}^{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^{t} r(s_t) | s_0 = s\right]$$

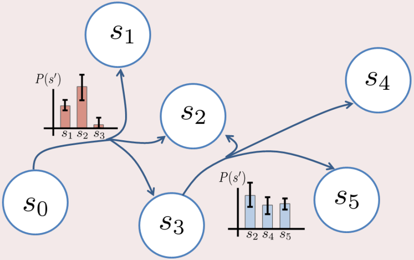


Background: Robust MDPs

Dynamic programming solution

- Robust value function (fixed policy)

$$V^{\pi}(s) \doteq \inf_{P \in \mathcal{P}} \mathbb{E}^{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) | s_0 = s\right]$$



Dynamic programming solution

- Robust value function (fixed policy)

$$V^{\pi}(s) \doteq \inf_{P \in \mathcal{P}} \mathbb{E}^{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) | s_0 = s\right]$$

- Robust Bellman equation (fixed policy)

$$V^{\pi}(s) = r(s) + \gamma \inf_{P \in \mathcal{P}(s)} \mathbb{E}^{P}\left[V^{\pi}(s') | s, \pi(s)\right]$$

Background: Robust MDPs

Dynamic programming solution

- Robust value function (fixed policy)

$$V^\pi(s) \doteq \inf_{P \in \mathcal{P}} \mathbb{E}^{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) | s_0 = s\right]$$

- Robust Bellman equation (fixed policy)

$$V^\pi(s) = r(s) + \gamma \inf_{P \in \mathcal{P}(s)} \mathbb{E}^P\left[V^\pi(s') | s, \pi(s)\right]$$

- Small problems: solved Policy iteration [Iyengar, 2005] and value iteration approach [Nilim et al. 2005]
- Large problems: Dynamic Programming cannot handle large spaces ("the curse of dimensionality")

Robust Policy Evaluation

Approximate value function

- Given state-dependent features $\phi(s)$
- Linear function approximation

$$\tilde{V}^{\pi}(s) = \phi(s)^{\top} w$$

- How to select w?

- For standard (non-robust) problems:

$$V^{\pi}(s) \doteq \mathbb{E}^{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^{t} r(s_t) | s_0 = s\right]$$

Sample and regress w.

Robust Policy Evaluation

Approximate value function

- Given state-dependent features $\phi(s)$
- Linear function approximation

$$\tilde{V}^{\pi}(s) = \phi(s)^{\top} w$$

- How to select w?

- For robust problems

$$V^{\pi}(s) = \inf_{P \in \mathcal{P}} \mathbb{E}^{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^{t} r(s_t) | s_0 = s\right]$$

Cannot regress w: how to sample trajectories from worst-case model?

Our approach

- Recall the Bellman equation

$$V^\pi(s) = r(s) + \gamma \inf_{P \in \mathcal{P}(s)} \mathbb{E}^P \left[V^\pi(s') | s, \pi(s) \right]$$

- Idea: **bootstrap!**

Robust Policy Evaluation

Our approach

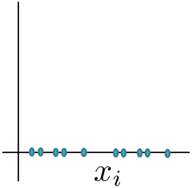
- Recall the Bellman equation

$$V^\pi(s) = r(s) + \gamma \inf_{P \in \mathcal{P}(s)} \mathbb{E}^P\left[V^\pi(s') | s, \pi(s)\right]$$

- Idea: **bootstrap!**

Algorithm

Given: initial weights w_0, sample states $x_1 \ldots x_N$



Robust Policy Evaluation

Our approach

- Recall the Bellman equation

$$V^\pi(s) = r(s) + \gamma \inf_{P \in \mathcal{P}(s)} \mathbb{E}^P \left[V^\pi(s') | s, \pi(s) \right]$$

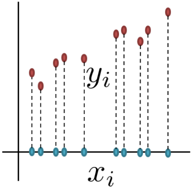
- Idea: **bootstrap!**

Algorithm

Given: initial weights w_0, sample states $x_1 \ldots x_N$

- At iterate $k+1$ generate regression targets

$$y_i = r(x_i) + \gamma \inf_{P \in \mathcal{P}(x_i)} \sum_{x'} P(x'|x_i, \pi(x_i)) \underbrace{\phi(x')^\top w_k}_{\tilde{V}_k^\pi(x')}$$



Robust Policy Evaluation

Our approach

- Recall the Bellman equation

$$V^\pi(s) = r(s) + \gamma \inf_{P \in \mathcal{P}(s)} \mathbb{E}^P \left[V^\pi(s') | s, \pi(s)\right]$$

- Idea: **bootstrap!**

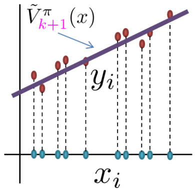
Algorithm

Given: initial weights w_0, sample states $x_1 \ldots x_N$

- At iterate $k+1$ generate regression targets

$$y_i = r(x_i) + \gamma \inf_{P \in \mathcal{P}(x_i)} \sum_{x'} P(x'|x_i, \pi(x_i)) \underbrace{\phi(x')^\top w_k}_{\tilde{V}^\pi_k(x')}$$

- Solve for w_{k+1} using least squares



Results:

Guarantees

- The magic: Convergence + Error bounds

Policy improvement

- Can iterate between policy evaluation and policy improvement
- Can derive deep Q-learning (model free) simulation based algorithm
- Error bounds follow through

Part 2

Two issues remain:

1. Uncertainty set construction
2. Online adaptivity

C. Tessler, Y. Efroni, and SM, ICML 2019

E. Derman, D. Mankowitz, T. Mann, and SM, UAI 2019.

Action Robustness

A trembling hand model

$$\pi_{\alpha}^{mix}(\pi, \pi') = \begin{cases} \pi, & \text{w.p. } 1 - \alpha. \\ \pi', & \text{w.p. } \alpha. \end{cases}$$

The policy π' is potentially adversarial.
Continuous extension: agent chooses a, adversary can modify to $(1 - \alpha)a + \alpha a'$.

Action Robustness

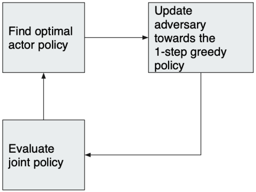
A trembling hand model

$$\pi_\alpha^{mix}(\pi, \pi') = \begin{cases} \pi, & \text{w.p. } 1-\alpha. \\ \pi', & \text{w.p. } \alpha. \end{cases}$$

The policy π' is potentially adversarial.
Continuous extension: agent chooses a, adversary can modify to $(1-\alpha)a + \alpha a'$.

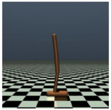
AR-DDPG:

1. Train Actor
2. Train Adversary
3. Train Critic for the joint policy

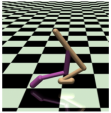


Theorem: This procedure converges to the Nash equilibrium.

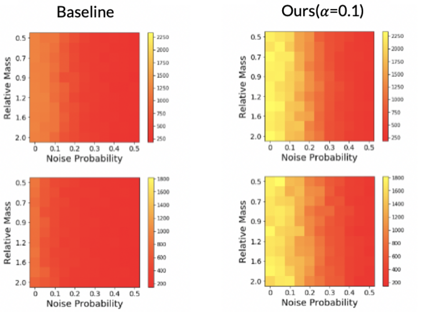
Some results



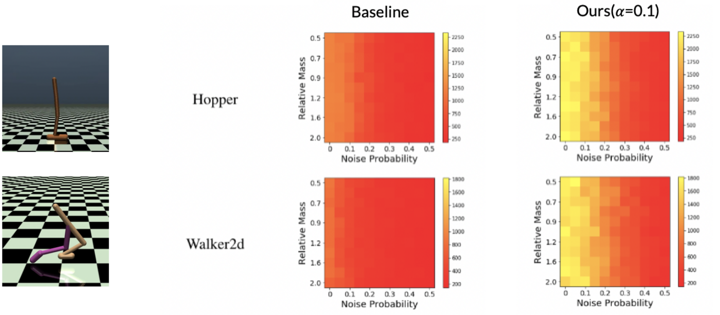
Hopper



Walker2d



Some results



- Robustness: uncertainty + transfer to unseen domains
- A gradient based approach for robust reinforcement learning with convergence guarantees
- Does not require explicit definition of the uncertainty set

Posterior Uncertainty Sets: Online Construction of Uncertainty Sets

- Dirichlet prior on distribution over next states.
- Observation history $\mathcal{H}$ up to time h
- Time h - current step and t - current episode

$$\widehat{\mathcal{P}}^h_{sa}(\psi_{sa}) = \{p_{sa} \in \Delta_{\mathcal{S}} : \|p_{sa} - \bar{p}_{sa}\|_1 \leq \psi_{sa}\}$$

$\bar{p}_{sa} = \mathbb{E}[p_{sa} \mid \mathcal{H}]$ is the *nominal* transition.

This uncertainty set is

- Rectangular:

$$\widehat{\mathcal{P}}^h = \bigotimes_{s \in \mathcal{S}, a \in \mathcal{A}} \widehat{\mathcal{P}}^h_{s,a}$$

- Updated **online** according to new observations

Uncertainty Robust Bellman Equation

- Posterior robust Q-value random variables satisfy a **robust Bellman recursion**

$$\hat{Q}^h_{sa} \overset{D}{=} r^h_{sa} + \gamma \inf_{p \in \widehat{\mathcal{P}}^h_{sa}} \sum_{s',a'} \pi^h_{s'a'} p_{sas'} \widehat{Q}^{h+1}_{s'a'}$$

- Posterior worst-case transition:
$\widehat{p}^h_{sa} \in \arg\min_{p \in \widehat{\mathcal{P}}^h_{sa}} \sum_{s',a'} \pi^h_{s'a'} p_{sas'} \widehat{Q}^{h+1}_{s'a'}$

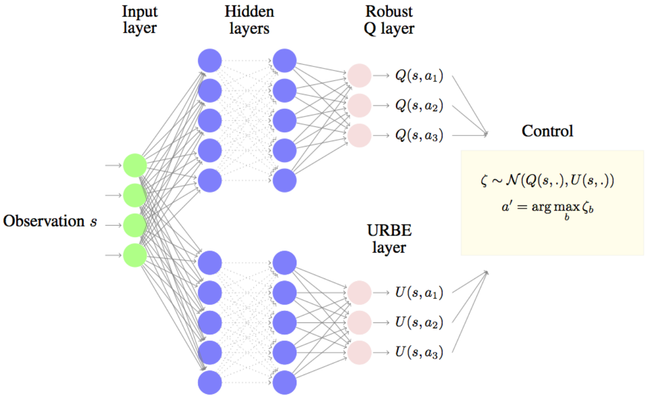
Theorem (Solution of URBE)

There exists a unique mapping w that satisfies the URBE:

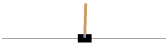
$$w^h_{sa} = \nu^h_{sa} + \gamma^2 \sum_{s' \in \mathcal{S}, a' \in \mathcal{A}} \pi^h_{s'a'} \mathbb{E}_t(\widehat{p}^h_{sas'}) w^{h+1}_{s'a'}$$

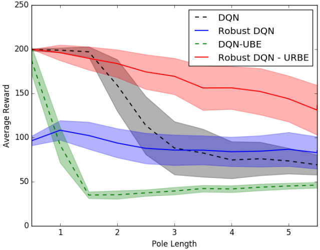
- Approximate Q-values as $\mathcal{N}(Q, diag(w))$.

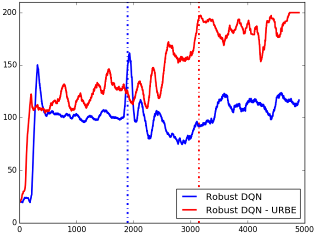
Deep Learning Approximation



Q-head uses robust TD error. URBE layer uses approximation.







- DQN/DQN-UBE: Overly **sensitive** to change of dynamics
- Robust DQN: Overly **conservative**

Conclusion

Robustness is essential for learning

- Handles ‘unluckiness’
- Overcomes model misspecification
- Works online with deep models (scalability)

Take home message: solve robust MDPs

Scalable, works, and even has theoretical guarantees!

Applications: health, energy, finance, robotics, cyber, e-commerce

Thanks

Joint work with:

E. Boccara (Technion), Y. Chow (Google AI), G. Dallal (Nvidia), Y. Efroni (MSR), M. Ghavamzadeh (FAIR), A. Hallak (Nvidia), M. Kozdoba (Technion), O. Maillard (CNRS Lille), D. Mankowitz (Google DeepMind), T. Mann (Google DeepMind), M. Pavone (Stanford), A. Tamar (Technion), C. Tessler (Technion), J. Tsitsiklis (MIT), H. Xu (Alibaba).