Reinforcement Learning for Critical Domains

Daan Bloembergen

joint work with

Richard Klima (UoL), Michael Kaisers (CWI), Karl Tuyls (DeepMind)
Motivation

• Many large scale **critical systems** are highly sensitive to the local performance of individual components.
  – traffic and transport networks, security systems, power grids, ...

• Local failure or attack could destabilise the whole system.

• **Goal**: explicitly encode robustness against significant rare events in the learning method.
What is Reinforcement Learning?

- Goal-oriented: learning about, from, and while *interacting* with an external environment
- Learning what to do — **how to map situations to actions** — so as to maximize a numerical reward signal
The Agent Learns a Policy

Policy at step $t$, $\pi_t$:
- a mapping from states to action probabilities
- $\pi_t(s, a) =$ probability that $a_t = a$ when $s_t = s$

- Reinforcement learning methods specify how the agent changes its policy as a result of experience.
- Roughly, the agent’s goal is to get as much reward as it can over the long run.
Formal Model of Decision Problem

- A Markov Decision Process is defined by:
  - state and action sets
  - next-state transition probabilities
  - reward expectations

- The value of a state $s$ in an MDP under policy $\pi$ is

$$V^\pi(s) = E_\pi \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right]$$

**return**: discounted sum of rewards
Bellman Equation (1950s)

State value function can be written as:

$$V^\pi(s) = E_\pi\{R_t|s_t = s\}$$
$$= E_\pi\{r_{t+1} + \gamma R_{t+1}\}$$
$$= \sum_a \pi(s, a) \sum_{s'} P^a_{ss'} [R^a_{ss'} + \gamma V^\pi(s')]$$

- The equation is **recursive**: $V(s)$ depends on $V(s')$.
- It sums over all possible **future returns**, weighted by their probability of occurring:
  - the **action probability** given by $\pi(s, a)$
  - the **state transition probability** given by $P^a_{ss'}$
Temporal Difference (TD) Learning

- **Temporal difference:**
  Look at the difference between the *current estimate* of the value of a state and the *sampled reward plus the discounted value of the next state*.

- Keep adjusting the value function aiming to reduce the **TD error** (until convergence).

  \[ V(s_t) \leftarrow V(s_t) + \alpha [r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \]

  **TD error:** target – current estimate

  **Target:** an estimate of the return
Estimate $Q^\pi$ for the current behaviour policy $\pi$.

After every transition, update your estimate for $Q$ as:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

(If $s_{t+1}$ is terminal, then $Q(s_{t+1}, a_{t+1}) = 0$)
Deriving the Policy

• For any (optimal) value function, we can easily derive a policy that yields it:

\[ \pi(s) = \arg\max_a Q(s, a), \forall s \in S \]

• Typically, during learning we want to balance exploration (random actions) and exploitation (greedy actions)

• We distinguish a target policy (what we wish to learn) and a behavior policy (how we collect experience)
On-Policy vs. Off-Policy

**Sarsa** (on-policy):

\[
Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]
\]

**Q-learning** (off-policy):

\[
Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[ r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]
\]
Robust RL for Critical Domains

• Our idea in a nutshell: encode the expected probability (and model) of attacks or failures in the TD error $\delta_t$.

• E.g., malicious attack with probability $\kappa$:

$$
\delta_t = r_{t+1} + \gamma \left( (1 - \kappa) \max_a Q(s_{t+1}, a) + \kappa \min_a Q(s_{t+1}, a) \right) - Q(s_t, a_t)
$$

modified estimate of next state value
Results

• For increasing attack probability $\kappa$ we learn an increasingly safe policy
κ-methods

We can easily build versions of standard TD methods based on this idea.

- **Modified TD update rule:**
  \[ Q^\pi(s, a) \leftarrow Q^\pi(s, a) + \alpha \left[ r + \gamma V^\kappa(s') - Q^\pi(s, a) \right] \]

- **Q(κ)-learning:**
  \[ V^\kappa(s) = (1 - \kappa) \max_a Q(s, a) + \kappa \min_a Q(s, a) \]

- **Expected SARSA(κ):**
  \[ V^\kappa(s) = (1 - \kappa) \mathbb{E}_{a \sim \pi} [Q(s, a)] + \kappa \min_a Q(s, a) \]
  \[ = (1 - \kappa) \sum_a \pi(a|s)Q(s, a) + \kappa \min_a Q(s, a) \]

- **General κ-model:**
  \[ V^\kappa(s) = \sum_{\sigma \in C} p(\sigma|s) \sum_a \sigma(s, a)Q^\pi(s, a) \]
Results

- Our new methods (green and purple) outperform the original TD methods on which they are based
Theoretical Analysis

• We prove convergence of both $Q(\kappa)$ and Expected SARSA($\kappa$) to two different fixed points:
  - to the optimal value function $Q^*$ of the original MDP in the limit where $\kappa \to 0$; and
  - to the optimal robust value function $Q^*_\kappa$ of the MDP that is generalized w.r.t. $\kappa$ for constant parameter $\kappa$.

• Note that optimality in this sense is purely induced by the relevant operator.
Conclusion

• $\kappa$-versions of standard TD methods learn a safer policy \textbf{before experiencing the deviation}
  – Especially beneficial in the early learning stage.
  – Robust against model mis-specification.

• Proven \textit{convergence} to \textit{optimal} value function.

• Promising empirical results in single- and multi-agent settings.