Configurable Markov Decision Processes

A. M. Metelli, M. Mutti and M. Restelli

{albertomaria.metelli, marcello.restelli}@polimi.it
{mirco.mutti}@mail.polimi.it

POLITECNICO MILANO 1863

Motivations and Problem

- In many real-world problems it is possible to **configure the environment** during the learning process:
  - *Driving on a track*: the driver/engineer configures the vehicle settings (e.g., seasonal tires, stability and vehicle attitude, engine model, ...)
  - *Student-Teacher interaction*: the teacher adapts its teaching style to the student's needs (e.g., simplify the questions, slow down the presentation of concepts, ...)
- Configuring the environment can bring some **benefits**:
  - learn policy with **higher performances** (e.g., configure the car to learn a better driving policy)
  - **speed up** the learning process (e.g., configure the car to learn faster a good driving policy)
- How to choose an **environment configuration** so that the performance of the **optimal policy** is maximized?

Contributions

1. We propose a new framework, **Configurable Markov Decision Process** (Conf-MDP), to model sequential decision-making problems in which the transition model is configurable.
2. We present a safe learning algorithm, **Safe Policy-Model Iteration** (SPMI), able to *jointly* and *adaptively* optimize the policy and the environment configuration.
3. We provide an **empirical evaluation** to highlight the benefits of the environment configuration and the effectiveness of SPMI.

Configurable Markov Decision Processes

Markov Decision Process (MDP)

$(\mathcal{S}, \mathcal{A}, R, P, \gamma, \mu)$

- Learning in an MDP means to recover an **optimal policy** π^* under a **fixed environment** P
- The environment P is fixed and **cannot** be controlled
- The policy π belongs to a policy space Π

Configurable Markov Decision Process (Conf-MDP)

$(\mathcal{S}, \mathcal{A}, R, \gamma, \mu, \mathcal{P}, \Pi)$

- Learning in a Conf-MDP means to recover an **optimal policy** π^* and an **optimal environment configuration** P^*
- The environment P belongs to a the model space $\mathcal{P}$ and **can** be configured
- The policy π belongs to a policy space Π

Safe Policy-Model Iteration

- SPMI finds the optimal model-policy pair:

$$P^*, \pi^* = \arg\max_{P \in \mathcal{P}, \pi \in \Pi} J_\mu^{P,\pi}$$

- At each iteration SPMI decides whether to update the **model** P, the **policy** π or **both** (*joint* and *adaptive*)
- SPMI is **safe** as it ensures a monotonic *performance improvement* [Pirotta et al., 2013]
- SPMI optimizes a *lower-bound* of the performance improvement:

$$\underbrace{J_\mu^{P',\pi'} - J_\mu^{P,\pi}}_{\text{performance improvement}} \geq \underbrace{B(P', \pi', P, \pi)}_{\text{lower-bound}}$$

Bound on Performance Improvement

$$B(P', \pi', P, \pi) = \frac{\underbrace{\mathbb{A}_{P,\pi,\mu}^{P',\pi}}_{\text{model advantage}} + \underbrace{\mathbb{A}_{P,\pi,\mu}^{P,\pi'}}_{\text{policy advantage}}}{1-\gamma} - \frac{\gamma \Delta Q^{P,\pi} \overbrace{D}^{\text{dissimilarity}}}{2(1-\gamma)^2}$$

Policy and Model Update

$$P' = (1-\beta)P + \beta \overline{P} \qquad \pi' = (1-\alpha)\pi + \alpha \overline{\pi}$$

Algorithm

initialize π_0, P_0
for $i = 0, 1, 2, \dots$ **until** convergence **do**
  $\overline{P}_i = ModelChooser(P_i)$
  $\overline{\pi}_i = PolicyChooser(\pi_i)$
  $\alpha_i, \beta_i = \arg\max_{\alpha,\beta \in [0,1]} B(P', \pi', P, \pi)$
  $\pi_{i+1} = \alpha_i \overline{\pi}_i + (1-\alpha_i)\pi_i$
  $P_{i+1} = \beta_i \overline{P}_i + (1-\beta_i)P_i$
end for

More on Conf-MDPs

Model and Policy Chooser

How to select the *target* $\overline{P}_i$ and $\overline{\pi}_i$?

- **greedy** chooser
  - select the policy/model that maximizes the advantage function
  - might generate oscillations
- **persistent** chooser
  - select between the *greedy* and the old target
  - avoids oscillations

Model and Policy Spaces

How to represent the model space $\mathcal{P}$?

- **unconstrained** model space: any model is valid
- **parametric** model space: the model $P_{\boldsymbol{\omega}}$ depends on parameters $\boldsymbol{\omega}$
  - e.g., convex hull of vertex models

P-Gradient Theorem

Extension of the *Policy Gradient Theorem* [Sutton et al., 2000] to model learning:

$$\nabla_{\boldsymbol{\omega}} J_\mu^{P_{\boldsymbol{\omega}}} = \int \delta_\mu^{P_{\boldsymbol{\omega}}}(s,a) \nabla_{\boldsymbol{\omega}} P_{\boldsymbol{\omega}}(s'|s,a) U^{P_{\boldsymbol{\omega}}}(s,a,s') \,\mathrm{d}s'\mathrm{d}a\mathrm{d}s.$$

$$U^{P_{\boldsymbol{\omega}}}(s,a,s') = R(s,a) + \gamma V^{P_{\boldsymbol{\omega}}}(s').$$

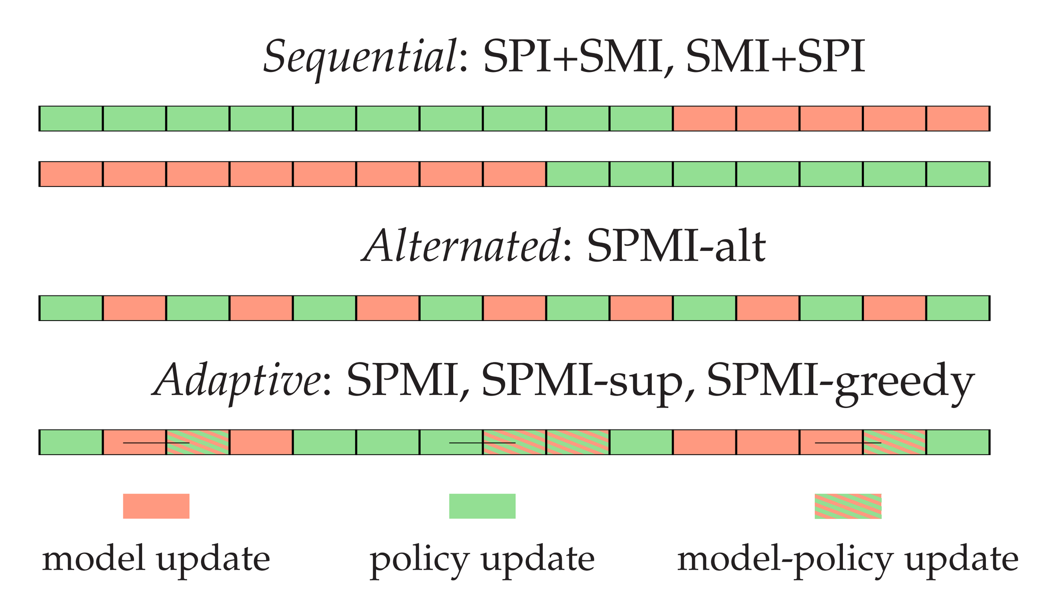
References

Matteo Pirotta, Marcello Restelli, Alessio Pecorino, and Daniele Calandriello. Safe policy iteration. In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, volume 28 of *ICML'13*, pages 307–315, 2013.

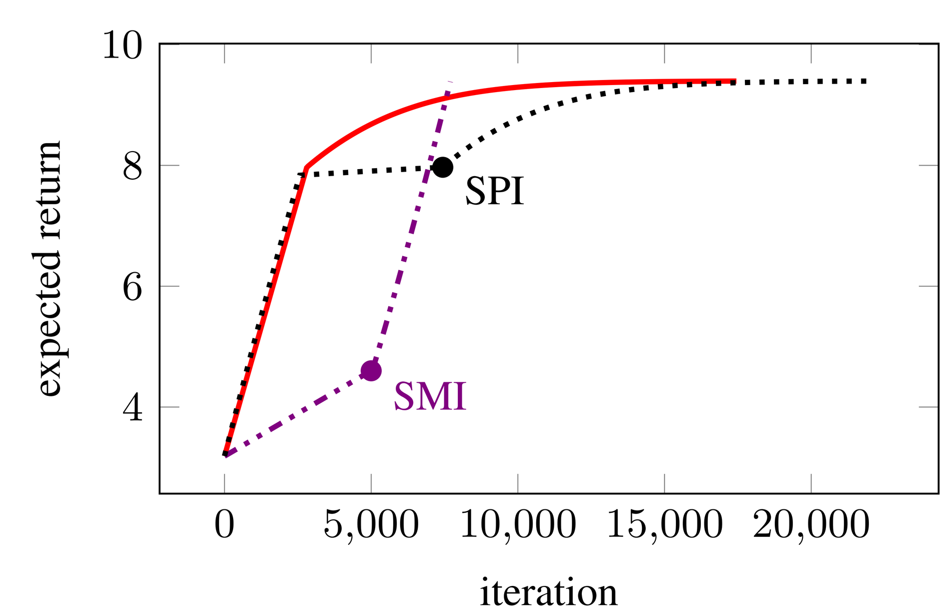
Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063, 2000.

Experimental Evaluation

Compared Algorithms



Student-Teacher



RaceTrack