Reinforcement Learning

Erwan Le Pennec
Erwan.Le-Pennec@polytechnique.edu

M2DS - Reinforcement Learning – Winter 2023-2024

Outline

Outline

Decision or Decisions

Source: W. Powell

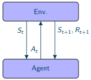
Sequential Decision Setting

Sequential Decision Setting

- In many (most?) settings, not a single decision but a sequence of decisions.
- Need to take into account the (not necessarily immediate) consequences of the sequence of decisions/actions rather than of each decisions.
- Different framework than supervised learning (no immediate feedback here) and unsupervised learning (well defined goal here).

Source: W. Powell

From Sequential Decision to Reinforcement Learning



Sequential Decision

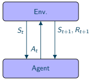
Sequential Decision

- Sequence of action A_t as a response of an environment defined by a state S_t
- Feedback through a reward R_t

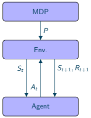
Actions?

- Is my current way of choosing actions good?
- How to make it better?

From Sequential Decision to Reinforcement Learning



Sequential Decision



MDP Modeling

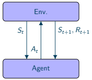
Markov Decision Process Modeling

- Specific modeling of the environment.
- Goal as as a (weighted) sum of a scalar reward.

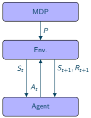
Actions?

- Is my current way of choosing actions good?
- How to make it better?

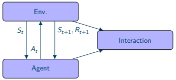
From Sequential Decision to Reinforcement Learning



Sequential Decision



MDP Modeling



Reinforcement Learning

Reinforcement Learning

- Same modeling...
- But no direct knowledge of the MDP.

Actions?

- Is my current way of choosing actions good?
- How to make it better?

Sequential Decision Settings

Sequential Decisions

- MDP / Reinforcement Learning:

$$\max_{\pi} \mathbb{E}_{\pi}\left[\sum_t R_t\right]$$

- Optimal Control:

$$\min_{u} \mathbb{E}\left[\sum_t C(x_t, u_t)\right]$$

Related settings. . .

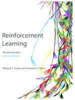
- (Stochastic) Search:

$$\max_{\theta} \mathbb{E}[F(\theta, W)]$$

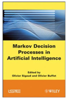
- Online Regret:

$$\max \sum_k \mathbb{E}[F(\theta_k, W)]$$

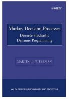
References



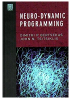
R. Sutton and A. Barto.
Reinforcement Learning, an Introduction (2nd ed.)
MIT Press, 2018



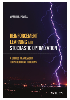
O. Sigaud and O. Buffet.
Markov Decision Processes in Artificial Intelligence.
Wiley, 2010



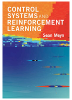
M. Puterman.
Markov Decision Processes. Discrete Stochastic Dynamic Programming.
Wiley, 2005



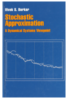
D. Bertsekas and J. Tsitsiklis.
Neuro-Dynamic Programming.
Athena Scientific, 1996



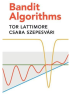
W. Powell.
Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions.
Wiley, 2022



S. Meyn.
Control Systems and Reinforcement Learning.
Cambridge University Press, 2022



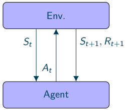
V. Borkar.
Stochastic Approximation: A Dynamical Systems Viewpoint.
Springer, 2008



T. Lattimore and Cs. Szepesvári.
Bandit Algorithms.
Cambridge University Press, 2020

Outline

Environnement

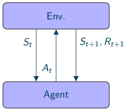


Decision Process and Environment

- At time step $t \in \mathbb{N}$:
  - State $S_t \in \mathcal{S}$: representation of the environment
  - Action $A_t \in \mathcal{A}(S_t)$: action chosen
  - Reward $R_{t+1} \in \mathcal{R}$: instantaneous reward
  - New state S_{t+1}

- Focus on the discrete setting, i.e. $\mathcal{S}$ finite, $\mathcal{A}(s)$ finite and $\mathcal{R}$ finite.
- Extension: Non finite bounded $\mathcal{R}$: easy / Non finite $\mathcal{S}$: hard / Non finite $\mathcal{A}$: harder.

Environnement



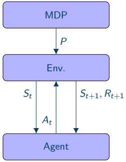
Stochastic Model

- Dynamic defined by:

$$\mathbb{P}\big(S_{t+1} = s', R_{t+1} = r \big| (S_{t'}, A_{t'}, R_{t'}), t' \leq t\big)$$
$$= \mathbb{P}\big(S_{t+1} = s', R_{t+1} = r \big| S_t = s, A_t = a, H_t\big)$$

where $H_t = (R_t, S_{t-1}, A_{t-1}, R_{t-1}, S_{t-2}, \ldots)$ is the past and (S_t, A_t) the present.

Markov Decision Process and Environment



Markovian Environment

- Markovian Dynamic Assumption: S_{t+1} and R_{t+1} are independent of the past $H_t = (R_t, S_{t-1}, A_{t-1}, R_{t-1}, S_{t-2}, \dots)$ conditionally to the present (S_t, A_t).
- Dynamic entirely defined by state-reward transition probabilities

$$\begin{aligned}\mathbb{P}(S_{t+1} = s', R_{t+1} = r \big| S_t = s, A_t = a, H_t) &= \mathbb{P}(S_{t+1} = s', R_{t+1} = r \big| S_t = s, A_t = a) \\ &= p(s', r | s, a)\end{aligned}$$

in the discrete setting.

- Informally, this means that S_t encodes all the information related to the past.

Markov Decision Process and State-Action

- State-Reward transition probabilities for a given state-action:

$$\begin{aligned}\mathbb{P}(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a, H_t) &= \mathbb{P}(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a)\\ &= p(s', r|s, a)\end{aligned}$$

Induced State-action laws

- State transition probabilities for a given state-action:

$$\begin{aligned}\mathbb{P}(S_{t+1} = s' | S_t = s, A_t = a, H_t) &= \mathbb{P}(S_{t+1} = s' | S_t = s, A_t = a)\\ &= p(s'|s, a) = \sum_r p(s', r|s, a)\end{aligned}$$

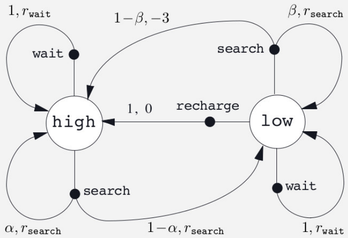
- Expected reward for a given state-action:

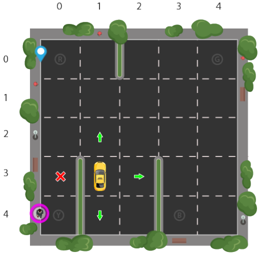
$$\begin{aligned}\mathbb{E}[R_{t+1} | S_t = s, A_t = a, H_t] &= \mathbb{E}[R_{t+1} | S_t = s, A_t = a]\\ &= r(s, a) = \sum_r r \sum_{s'} p(s', r|r, a)\end{aligned}$$

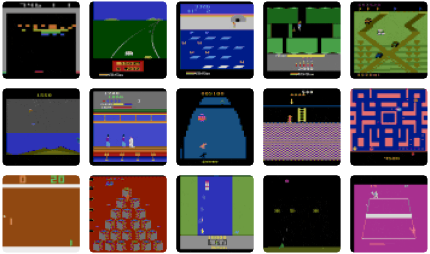
- From now on, we will always assume that the Markovian property holds for the environment.

Examples

s	a	s'	$p(s' \mid s, a)$	$r(s, a, s')$
high	search	high	α	$r_{\texttt{search}}$
high	search	low	$1-\alpha$	$r_{\texttt{search}}$
low	search	high	$1-\beta$	-3
low	search	low	β	$r_{\texttt{search}}$
high	wait	high	1	$r_{\texttt{wait}}$
high	wait	low	0	-
low	wait	high	0	-
low	wait	low	1	$r_{\texttt{wait}}$
low	recharge	high	1	0
low	recharge	low	0	-

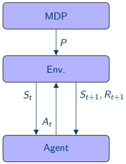






Sources:: Sutton and Barto / learndatasci.com / danijar.com

Decision Process, Agent and Policy



Agent

- Interact with the environment by choose the action given the past.

Policy Π : specification of how to choose the action

- General stochastic policy $\Pi = (\pi_0, \pi_1, \dots, \pi_t, \dots)$:
$$\Pi_t(A_t = a) = \pi_t(A_t = a | S_t = a, A_t = a, H_t)$$
- General deterministic policy $\Pi = (\pi_0, \pi_1, \dots, \pi_t, \dots)$ (with as slight abuse of notation):
$$\Pi_t(A_t = a) = \mathbf{1}_{A_t = \pi_t(S_t = a, A_t = a, H_t)}$$

Markov Decision Process, Agent and Policy

Agent

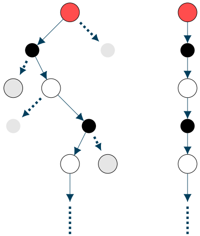
- Interact with the environment by choose the action given the past.

Policy Π: specification of how to choose the action

- History dependent stochastic policy $\Pi = (\pi_0, \pi_1, \ldots, \pi_t, \ldots)$:
$$\Pi_t(A_t = a) = \pi_t(A_t = a | S_t = s, H_t)$$
- Markovian stochastic policy $\Pi = (\pi_0, \pi_1, \ldots, \pi_t, \ldots)$:
$$\Pi_t(A_t = a) = \pi_t(A_t = a | S_t = s) = \pi_t(a|s)$$
- Stationary Markovian stochastic policy $\Pi = (\pi, \pi, \ldots, \pi, \ldots)$:
$$\Pi_t(A_t = a) = \pi(A_t = a | S_t = s) = \pi(a|s)$$

- Similar deterministic policy definition.
- Partially Observed Markov Decision Process extension: the Agent has only access to a partial observation O_t at each time step... (not the focus of the lectures)

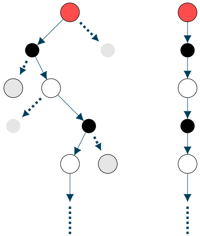
Decision Process and Trajectories



Trajectories

- Trajectory $(S_0, A_0, R_1, S_1, A_1, \ldots)$ defined by
  - an initial distribution $\mathbb{P}_0$ for S_0,
  - a policy $\Pi = (\pi_0, \pi_1, \ldots, \pi_t, \ldots)$ specifying
  $$\Pi_t(A_t = a) = \pi_t(A_t = a | S_t, H_t)$$
  - an environment specifying
  $$\mathbb{P}(S_{t+1}, R_{t+1} | S_t, A_t, H_t)$$

Decision Process and Trajectories

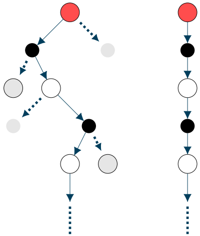


Trajectories

- Induced probability:

$$
\begin{aligned}
&\mathbb{P}(S_0 = s_0, A_0 = a_0, R_1 = r_1, S_1 = s_1, A_1 = a_1, \ldots S_t = s_t, R_t = r_t) \\
&\quad = \mathbb{P}_0(S_0 = s_0) \\
&\qquad \times \pi_0(A_0 = a_0 | S_0)\, \mathbb{P}(S_1, R_1 | S_0, A_0)\, \pi_1(A_1 = a_1 | S_1 = s_1, H_1) \\
&\qquad\quad \times \cdots \times \mathbb{P}(S_t = s_t, R_t = r_t | S_{t-1} = s_{t-1}, A_{t-1} = a_{n-1}, H_{t-1})
\end{aligned}
$$

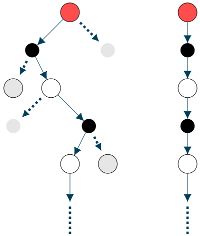
Markov Decision Process and Trajectories



Trajectories

- Trajectory $(S_0, A_0, R_1, S_1, A_1, \ldots)$ defined by
  - an initial distribution $\mathbb{P}_0$ for S_0,
  - a policy $\Pi = (\pi_0, \pi_1, \ldots, \pi_t, \ldots)$ specifying
    $$\Pi_t(A_t = a) = \pi_t(A_t = a | S_t, H_t)$$
  - a Markovian environment specifying
    $$\mathbb{P}(S_{t+1}, R_{t+1} | S_t, A_t)$$

Markov Decision Process and Trajectories

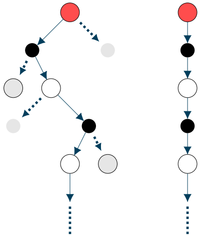


Trajectories

- Induced probability:

$$
\begin{aligned}
&\mathbb{P}(S_0 = s_0, A_0 = a_0, R_1 = r_1, S_1 = s_1, A_1 = a_1, \ldots S_t = s_t, R_t = r_t) \\
&\quad = \mathbb{P}_0(S_0 = s_0) \\
&\qquad \times \pi_0(A_0 = a_0 | S_0)\, \mathbb{P}(S_1, R_1 | S_0, A_0)\, \pi_1(A_1 = a_1 | S_1 = s_1, H_1) \\
&\qquad\quad \times \cdots \times \mathbb{P}(S_t = s_t, R_t = r_t | S_{t-1} = s_{t-1}, A_{t-1} = a_{t-1})
\end{aligned}
$$

Markov Decision Process and Trajectories



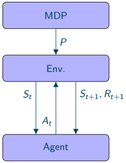
Markovian Trajectories only if the policy is Markovian

- $$\begin{aligned}&\mathbb{P}(R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, S_{t+2}, \dots R_{t+k}, S_{t+k} | S_t, A_t, H_t)\\ &\quad = \mathbb{P}(R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, S_{t+2}, \dots R_{t+k}, S_{t+k} | S_t, A_t)\\ &\quad = \mathbb{P}(S_{t+1}, R_{t+1} | S_t, A_t)\, \pi_{t+1}(A_{t+1} | S_{t+1})\\ &\qquad \times \cdots \times \mathbb{P}(S_{t+k}, R_{t+k} | S_{t+k-1}, A_{t+k-1})\end{aligned}$$

- Stationary if the policy is stationary.

Outline

Rewards and Total Return



Rewards and Total Returns

- MDP: Rewards R_t encode all the feedbacks!
- Quality of a policy Π measured from the remaining total return:

$$G_t = \sum_{t'=t+1}^{\infty} R_{t'}$$

- Expected total return following Π starting from s:

$$\mathbb{E}_\Pi[G_t | S_t = s] = \sum_{t'=t+1}^{\infty} \mathbb{E}_\Pi[R_{t'} | S_t = s]$$

Total Return: Issue and Fixes

Issues

- G_t is a limiting process and thus may not be defined!
- Can diverge to $\pm\infty$ and not converge at all.

Fixes?

- Finite horizon: $G_t^T = \sum_{t'=t+1}^{T} R_{t'}$
- Episodic setting: it exists a random T such that $\forall t' \geq R, R_{t'} = 0$ and $\mathbb{E}[T] < \infty$ so that $G_t = \sum_{t'=t+1}^{\infty} R_{t'}$ is well defined.
- Discounted setting: for $0 < \gamma < 1$, $G_t^\gamma = \sum_{t'=t+1}^{\infty} \gamma^{t'-(t+1)} R_{t'}$
- Average return: $\overline{G}_t = \lim \frac{1}{T} \sum_{t'=t+1}^{t+T} R_{t'}$

Finite Horizon

$$G_t^T = \sum_{t'=t+1}^{T} R_{t'}$$

Finite Horizon Setting

- Always well defined and easy to interpret.
- Loss of Markovianity as we need to know the time step. . .

- Can be put in a classical Markov framework!
  - Define an absorbing state s_{abs} (a state that cannot be escaped and from which the reward is always 0).
  - Extend the state space $\mathcal{S}$ to $(\mathcal{S} \times \{0, \ldots, T\}) \cup \{s_{\text{abs}}\}$.
  - Define an state reward transition probability:

$$p\left(\tilde{s}', r | \tilde{s}, a\right) = \begin{cases} p(s', t|s, a) & \text{if } \tilde{s} = (s, t),\ t < T \text{ and } \tilde{s'} = (s', t+1) \\ 1 & \text{if } \tilde{s} = (s, t),\ t = T,\ \tilde{s'} = s_{\text{abs}} \text{ and } r = 0 \\ 1 & \text{if } \tilde{s} = s_{\text{abs}}\ ,\ \tilde{s'} = s_{\text{abs}} \text{ and } r = 0 \\ 0 & \text{otherwise} \end{cases}$$

Episodic Setting

$$G_t = \sum_{t'=t+1}^{\infty} R_{t'}$$

Episodic Setting

- Assumption: for any policy Π, the average duration before $R_t = 0$ is smaller than a finite horizon H: $\mathbb{E}_\Pi \left[\min_{t, R_{t'}=0, \forall t' \geq t} t \right] \leq H < +\infty$
- Strong assumption...
- Easy to interpret.

- Equivalent def.:
  - Replace all the states from which R_t remains equal to 0 whatever the policy by a single absorbing state s_{abs},
  - Assumption: for any policy Π, the average duration to reach this state is smaller than a finite horizon H: $\mathbb{E}_\Pi \left[\min_{t, S_t = s_{\text{abs}}} t \right] \leq H < +\infty$

Discounted

$$G_t^\gamma = \sum_{t'=t+1}^{T} \gamma^{t'-(t+1)} R_{t'}$$

Discounted

- Always defined but not that easy to interpret.
- Easiest theoretical setting!

- Equivalent to an episodic setting if one adds an absorbing state s_{abs} and changes all state-reward transition probabilities to:

$$p(s', r|s, a) = \begin{cases} \gamma p(s', r|s, a) & \text{if } s' \neq s_{\text{abs}}, s \neq s_{\text{abs}} \\ (1-\gamma) & \text{if } s' = s_{\text{abs}}, r = 0, s \neq s_{\text{abs}} \\ 1 & \text{if } s' = s_{\text{abs}}, r = 0, s = s_{\text{abs}} \\ 0 & \text{otherwise} \end{cases}$$

- Horizon $H = 1/(1-\gamma)$.

Average Return Setting

$$\overline{G}_t = \lim \frac{1}{T} \sum_{t'=t+1}^{t+T} R_{t'}$$

Average Return

- Not always defined. (Cesaro Average)
- Always equal to 0 in the episodic setting!
- Natural definition in a *stationary* setting...
- Complex theoretical analysis!

- Under a strict stationarity assumption ($R_t \sim R_{t'}$), link with discounted setting as

$$\mathbb{E}_\Pi[G_t^\gamma] = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_\Pi[R_{t+1}] = \frac{1}{1-\gamma}\mathbb{E}_\Pi[R_t] = \frac{1}{1-\gamma}\mathbb{E}_\Pi\left[\overline{G}_t\right]$$

State Value Functions

State Value Functions

- Return expectation for a policy Π starting from s at time t
  - Finite horizon setting:
$$v_{t,\Pi}^{T}(s) = \mathbb{E}_{\Pi}\left[G_t^T | S_t = s\right] = \sum_{t'=t+1}^{T} \mathbb{E}_{\Pi}[R_{t'} | S_t = s]$$
  - Episodic setting:
$$v_{t,\Pi}(s) = \mathbb{E}_{\Pi}[G_t | S_t = s] = \sum_{t'=t+1}^{\infty} \mathbb{E}_{\Pi}[R_{t'} | S_t = s]$$
  - Discounted:
$$v_{t,\Pi}^{\gamma}(s) = \mathbb{E}_{\Pi}[G_t^{\gamma} | S_t = s] = \sum_{t'=t+1}^{\infty} \gamma^{t'-(t+1)} \mathbb{E}_{\Pi}[R_{t'} | S_t = s]$$
  - Average return setting:
$$\overline{v}_{t,\Pi}(s) = \mathbb{E}_{\Pi}\left[\overline{G}_t | S_t = s\right] = \lim_{T \to \infty} \frac{1}{T} \sum_{t'=t+1}^{t+T} \mathbb{E}_{\Pi}[R_{t'} | S_t = s]$$

- Depends on t for a history dependent policy!

Markovian Policy and State Value Functions

State Value Functions

- Return expectation for a Markovian policy Π starting from s at time t.
  - Finite horizon setting (with time extended state space):
$$v^T_{t,\Pi}(s) = \mathbb{E}_\Pi\left[G^T_t | S_t = s\right] = \sum_{t'=t+1}^{T} \mathbb{E}_\Pi[R_{t'} | S_t = s]$$
  - Episodic setting:
$$v_{t,\Pi}(s) = \mathbb{E}_\Pi[G_t | S_t = s] = \sum_{t'=t+1}^{\infty} \mathbb{E}_\Pi[R_{t'} | S_t = s]$$
  - Discounted:
$$v^\gamma_{t,\Pi}(s) = \mathbb{E}_\Pi[G^\gamma_t | S_t = s] = \sum_{t'=t+1}^{\infty} \gamma^{t'-(t+1)} \mathbb{E}_\Pi[R_{t'} | S_t = s]$$
  - Average return setting:
$$\overline{v}_{t,\Pi}(s) = \mathbb{E}_\Pi\left[\overline{G}_t | S_t = s\right] = \lim_{T\to\infty} \frac{1}{T} \sum_{t'=t+1}^{t+T} \mathbb{E}_\Pi[R_{t'} | S_t = s]$$

- Becomes independent on t if the policy is stationary and Markovian the generic case (except in the finite horizon setting).

State-Action Value Functions

- Return expectation for a policy Π starting from s and an action a at time t.
  - Finite horizon setting:
$$q^T_{t,\Pi}(s,a) = \mathbb{E}_\Pi\left[G^T_t | S_t = s, A_t = a\right] = \sum_{t'=t+1}^{T} \mathbb{E}_\Pi[R_{t'} | S_t = s, A_t = a]$$
  - Episodic setting:
$$q_{t,\Pi}(s,a) = \mathbb{E}_\Pi[G_t | S_t = s, A_t = a] = \sum_{t'=t+1}^{\infty} \mathbb{E}_\Pi[R_{t'} | S_t = s, A_t = a]$$
  - Discounted:
$$q^\gamma_{t,\Pi}(s,a) = \mathbb{E}_\Pi[G^\gamma_t | S_t = s, A_t = a] = \sum_{t'=t+1}^{\infty} \gamma^{t'-(t+1)} \mathbb{E}_\Pi[R_{t'} | S_t = s, A_t = a]$$
  - Average return setting:
$$\overline{q}_{t,\Pi}(s,a) = \mathbb{E}_\Pi\left[\overline{G}_t | S_t = s, A_t = a\right] = \lim_{T\to\infty} \frac{1}{T} \sum_{t'=t+1}^{t+T} \mathbb{E}_\Pi[R_{t'} | S_t = s, A_t = a]$$

- Different strategy for action at time t than after...
- Independent of t for a Markovian policy except for the finite horizon setting!

State Value Function vs State-Action Value Functions

$$v_{t,\Pi}(s) = \mathbb{E}_\Pi[G_t | S_t = s] \quad q_{t,\Pi}(s, a) = \mathbb{E}_\Pi[G_t | S_t = s, A_t = a]$$

State vs State-Action

- Performance measure of a policy Π:
  - starting from a state s for the state value function,
  - starting from a state s and an action a (not necessarily related to Π) for the state-action value function.
- State value function at time t from state-action value function:

$$v_{t,\Pi}(s) = \sum_a \Pi_t(a) q_t(s, a)$$

Do We Really Need The History Dependent Policies?

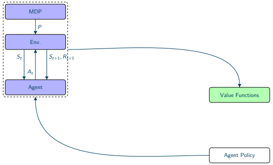
Equivalent Markovian policy in terms of value function

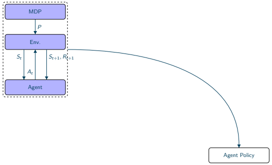
- **Thm:** For any policy Π and any initial distribution $\mathbb{P}_0(S_0)$, it exists a Markovian policy $\widetilde{\Pi}$ such that
$$\forall t, \forall s, v_{t,\Pi}(s) = v_{t,\widetilde{\Pi}}(s).$$

- Relies on the Markovian environment.
- Possible choice:
$$\widetilde{\pi}_t \left\{A_t = a_t | S_t = s_t\right\} = \mathbb{E}_{\mathbb{P},\mathbb{P}_0}[\pi_t(A_t = a_t | S_t = s_t, H_t) | S_t = s_t, S_0]$$
- **No need to consider non Markovian policy** if the goal is entirely defined in terms of value functions.

Outline

Goals





Prediction

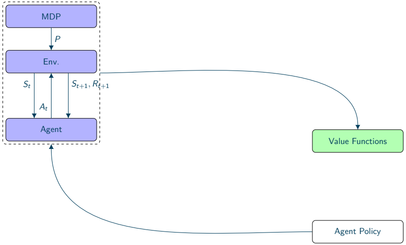
- What is the performance of a given policy?

Planning

- What is the *best* policy?

- Planning is harder than predicting.

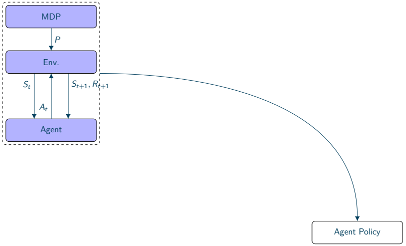
Prediction



Prediction

- What is the performance of a given policy?
- Compute/Approximate/Estimate
$$v_{t,\Pi}(s) = \mathbb{E}_{\Pi}[G_t | S_t = s]$$
- Well defined provided the expectation exists.

Planning

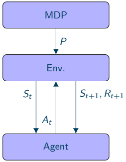


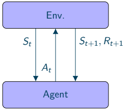
Planning

- What is the *best* policy?
- A possible definition: $\underset{\Pi}{\operatorname{argmax}} \sum_{s,t} \mu(s,t) v_{t,\Pi}(s)$
- Not necessarily well defined. . .
- Several choices for μ!
- More realistic goal: find a *good* policy. . .

Outline

What Do We Know?





Model

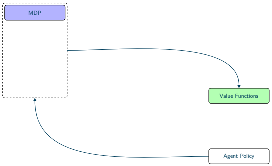
- Able to use the MDP transition probabilities.
- Markov Decision Process / Operations Research.
- Probability world.

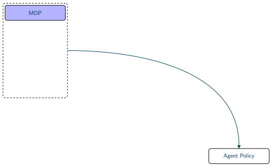
Only Observations

- No access to the MDP transition probabilities.
- Reinforcement Learning.
- Statistic world.

- Reinforcement Learning is harder than Markov Decision Process / Operations Research.

Markov Decision Process / Operations Research

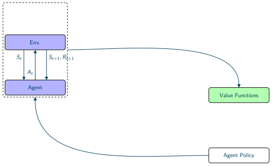


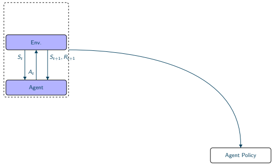


MDP / OR

- Stochastic setting in which the world is known.
- MDP model assumption.
- Probability world / Idealized setting...
- Lots of insight for the RL problem.

Reinforcement Learning





RL

- Stochastic setting in which the world is observed through interactions.
- Still MDP model assumption.
- More realistic setting?
- More difficult theoretical analysis.

Outline

MDP vs Discrete Control

MDP

- State s and action a
- Dynamic model:
$$\mathbb{P}(s'|s,a)$$
- Reward r defined by $\mathbb{P}(r|s',s,a)$.
- Policy Π: $a_t = \pi_t(S_t, H_t)$
- Goal:
$$\max \mathbb{E}_\Pi \left[\sum_t R_t \right]$$

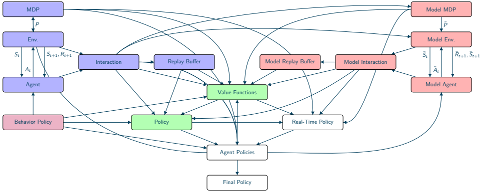
Discrete Control

- State x and control u
- Dynamic model:
$$x' = f(x,u,W)$$
with W a stochastic perturbation.
- Cost: $C(x,u,W)$.
- Control strategy U: $u_t = u(x_t, H_t)$
- Goal:
$$\min_U \mathbb{E}_U \left[\sum_t C(x_t, u_t, W_t) \right]$$

- Almost the same setting but with a different vocabulary!

Outline

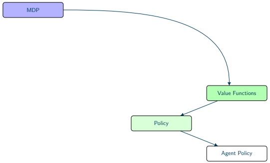
RL: What Are We Going To See?



Outline

- Operations Research and MDP.
- Reinforcement learning and interactions.
- More tabular reinforcement learning.
- Reinforcement and approximation of value functions.
- Actor/Critic: a Policy Point of View
- Extensions

Operations Research and MDP

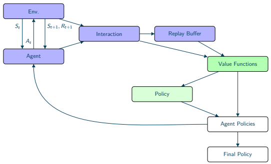


How to find the best policy knowing the MDP?

- Is there an optimal policy?
- How to estimate it numerically?

- Finite states/actions space assumption (tabular setting).
- Focus on interative methods using value functions (dynamic programming).
- Policy deduced by a statewise optimization of the value function over the actions.
- Focus on the discounted setting.

Reinforcement Learning and Interactions

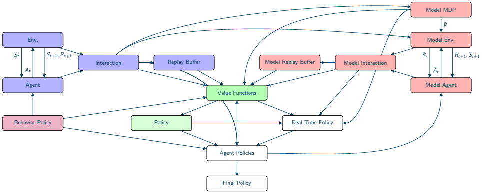


How to find the best policy not knowing the MDP?

- How to interact with the environment to learn a good policy?
- Can we use a Monte Carlo strategy outside the episodic setting?
- How to update value functions after each interaction?

- Focus on stochastic methods using tabular value functions (Q learning, SARSA...)
- Policy deduced by a statewise optimization of the value function over the actions.

More Tabular Reinforcement Learning

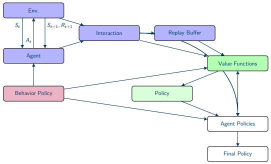


Can We Do Better?

- Is there a gain to wait more than one step before updating?
- Can we interact with a different policy than the one we are estimating?
- Can we use an estimated model to plan?
- Can we plan in real-time instead of having to do it beforehand?

- Finite states/actions space setting (tabular setting).

Reinforcement and Approximation of Value Functions

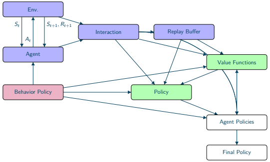


How to Deal with a Large/Infinite states/action space?

- How to approximate value functions?
- How to estimate good approximation of value functions?

- Finite action space setting.
- Stochastic algorithm (Deep Q Learning...).
- Policy deduced by a statewise optimization of the value function over the actions.

Actor/Critic: a Policy Point of View

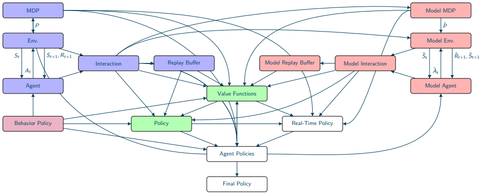


Could We Directly Parameterized the Policy?

- How to parameterize a policy?
- How to optimize this policy?
- Can we combine parametric policy and approximated value function?

- State Of The Art Algorithms (DPG,PPO, SAC. . .)

Extensions

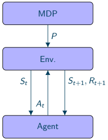


Can We Do Something Different in This Setting?

- How to deal with the total and average returns?
- How to deal with partial observations?
- How to learn a policy or an implicit reward by observing an actor?

Outline

Markov Decision Process / Operations Research



MDP / OR

- Known MDP model
- Focus on the finite horizon setting

$$G_t^T = \sum_{t'=t+1}^{T} R_{t'}$$

  and the discounted setting:

$$G_t^\gamma = \sum_{t'=t+1}^{\infty} \gamma^{t'-(t+1)} R_{t'}$$

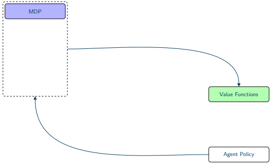
- We will later consider the other settings.

Policy

Policy

- Finite horizon : emphasis on Markovian policies
$$\Pi_t(A_t = a_t) = \pi_t(A_t = a_t | S_t = s_t) = \pi_t(a_t | s_t)$$
- Discounted return: emphasis on stationary Markovian policies
$$\Pi_t(A_t = a_t) = \pi(A_t = a_t | S_t = s_t) = \pi(a_t | s_t)$$

Prediction



Prediction

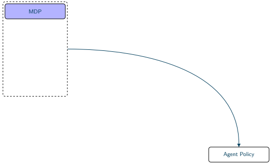
- How to efficently evaluate the quality of a policy

$$v_{t,\Pi}(s) = \mathbb{E}_{\Pi}\left[\sum_{t'=t+1}^{T} \gamma^{t'-(t+1)} R_{t'} \middle| S_t = s\right]$$

when we can ensure that the sum is finite?

- $v_{t,\Pi}$ independent of t in the discounted setting if the policy is stationary.

Planning



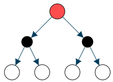
Policy

- How to find a policy π such that
$$\sum_{s,t} \mu(s,t) v_{t,\Pi}(s)$$
is as large as possible?
- Emphasis on $\mu(s,t) = 0$ if $t \neq 0$ and $\mu(s,0) = \mathbb{P}_0(S_0 = s_0)$.

Outline

Bellman Equation

$$
\begin{aligned}
v_{t,\Pi}(s) &= \sum_a \pi_t(a|s) \sum_{s',r} p(s', r|s, a) \left(r + \gamma v_{t+1,\Pi}(s')\right) \\
&= \sum_a \pi_t(a|s) r(s, a) + \gamma \sum_{s'} \sum_a p(s'|s, a) \pi_t(a|s) v_{t+1,\Pi}(s')
\end{aligned}
$$



Bellman Equation

- Link between $v_{t,\Pi}$ and $v_{t+1,\Pi}$.
- Straightforward consequence of

$$G_t = \sum_{t'=t+1}^{T} \gamma^{t'-(t+1)} R_{t'} = R_{t+1} + \gamma \sum_{t'=t+2}^{T} \gamma^{t'-(t+2)} R_{t'} = R_{t+1} + \gamma G_{t+1}$$

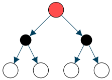
and thus

$$\mathbb{E}[G_t|S_t = s] = \mathbb{E}[R_{t+1}|S_t = s] + \gamma \mathbb{E}[\mathbb{E}[G_{t+1}|S_{t+1}]|S_t = s]$$

Bellman Operator

$$\mathcal{T}^{\pi_t} : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$$

$$\mathcal{T}^{\pi_t} v(s) = \underbrace{\sum_a \pi_t(a|s) r(s,a)}_{r_{\pi_t}(s)} + \gamma \sum_{s'} \underbrace{p(s'|s,a) \sum_a \pi_t(a|s)}_{P^{\pi_t}(s,s')} v(s')$$



Bellman Operator

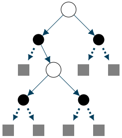
- Affine operator from the space of state value functions to the space of state value functions.
- By construction,
$$v_{t,\Pi} = \mathcal{T}^{\pi_t} v_{t+1,\Pi}$$
- r_{π_t} is the vector of average immediate rewards using policy π_t while P^{π_t} is the one step state transition matrix using policy π_t.

Outline

Finite Horizon: Naive Approach

$$v^T_{t,\Pi}(s) = \sum_{a_t, r_{t+1}, s_{t+1}, \cdots, r_T} \left(\sum_{t'=t+1}^{T} r_{t'} \right) \mathbb{P}_\Pi(A_t = a_t \ldots, R_T = r_T | S_t = s)$$

$$= \sum_{a_t, r_{t+1}, s_{t+1}, \cdots, r_T} \left(\sum_{t'=t+1}^{T} r_{t'} \right) \pi_t(a_t|s) \times \cdots \times p(s_T, r_T | s_{T-1}, a_{T-1})$$



Finite Horizon: Naive Approach

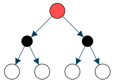
- Exhaustive exploration of the trajectories.
- Complexity of order $(|\mathcal{A}| \times |\mathcal{S}| \times |\mathcal{R}|)^{T-t}$ for the value function at time t.

- Complexity can be reduced to $(|\mathcal{A}| \times |\mathcal{S}|)^{T-t}$ by noticing that

$$v^T_{t,\Pi}(s) = \sum_{a_t, s_{t+1}, \cdots, s_{t-1}, a_{t-1}} \left(\sum_{t'=t+1}^{T} r(s_t, a_t) \right) \pi_t(a_t|s) \times \cdots \times p(s_T | s_{T-1}, a_{T-1})$$

Finite Horizon: Recursive Prediction

$$v^T_{T,\Pi} = 0$$
$$v^T_{t-1,\Pi} = \mathcal{T}^{\pi_{t-1}} v^T_{t,\Pi}$$



Finite Horizon: Recursive Prediction

- After time T, the finite horizon return $G^T_t = 0$ hence $v^T_{T,\Pi} = 0$ whatever the policy.
- The Bellman equation yields second equation.
- Equivalent rewriting
$$v^T_{t-1,\Pi}(s) = r_{\pi_{t-1}}(s) + \sum_{s'} P_{\pi_{t-1}}(s,s') v^T_t$$
- Complexity of order only $T \times |\mathcal{S}|^2(|\mathcal{A}| + |\mathcal{S}|)$ to compute all the value functions.

Finite Horizon: Value Iteration

Finite Horizon: Prediction by Value Iteration

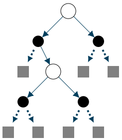
input: MDP model $\langle(\mathcal{S}, \mathcal{A}, \mathcal{R}), P\rangle$ and policy Π
parameter: Horizon T
init: $v_T^T(s) = 0 \,\forall\, s \in \mathcal{S},\ t = T$
repeat
  $t \leftarrow t - 1$
  for $\forall\, s \in \mathcal{S}$ **do**

$$v_t^T(s) \leftarrow \sum_{a \in \mathcal{A}} \pi_t(a|s) \left(r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) v_{t+1}^T(s') \right)$$

  end
until $t = 0$
output: Value functions v_t^T

- Most classical formulation

Discounted: Naive Approach



$$v^{\gamma}_{t,\Pi}(s) = \sum_{t'=t+1}^{\infty} \gamma^{t'-(t+1)} \mathbb{E}_{\Pi}[R_{t'}|S_t = s] \simeq \sum_{t'=t+1}^{T} \gamma^{t} \mathbb{E}_{\Pi}[R_{t'}|S_t = s] = v^{\gamma,T}_{t,\Pi}(s)$$

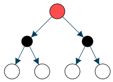
$$v^{\gamma,T}_{t,\Pi}(s) = \sum_{a_t,s_{t+1},\cdots,s_{t-1},a_{t-1}} \left(\sum_{t'=t+1}^{T} \gamma^{t'-(t+1)} r(s_t, a_t) \right) \pi_t(a_t|s) \times \cdots$$
$$\times\, p(s_T|s_{t-1}, a_{t-1})$$

Naive approach

- Exhaustive exploration of truncated trajectories.
- Back to the finite horizon setting...
- **Prop:** Control on the error as $\left| v^{\gamma}_{\Pi} - v^{\gamma,T}_{t,\Pi} \right|_{\infty} \leq \frac{\gamma^{T+1-t}}{1-\gamma} \max_{r \in \mathcal{R}} |r|$
- Relation between the error $\epsilon \simeq \gamma^{T-t}$ and the numerical complexity $C = (|\mathcal{A}| \times |\mathcal{S}|)^{T-t}$ of order $C \simeq \epsilon^{-1}$.

Discounted: Recursive Prediction with Naive Initialization

$$v^{\gamma}_{T,\Pi} \simeq v^{\gamma,T'}_{T,\Pi} = \tilde{v}_{T,\Pi}$$
$$v^{\gamma}_{t-1,\Pi} = \mathcal{T}^{\pi_{t-1}} v^{\gamma}_{t,\Pi} \simeq \tilde{v}_{t-1,\Pi} = \mathcal{T}^{\pi_{t-1}} \tilde{v}_{t,\Pi}$$



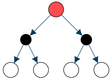
Recursive Prediction

- Requires an initialization at time T with a horizon T'.
- The Bellman equation yields the second equation.
- Complexity of order only $T \times |\mathcal{S}|^2(|\mathcal{A}| + |\mathcal{S}|)$ to compute all the value functions after the initialization of cost $(|\mathcal{A}| \times |\mathcal{S}|)^{T'-T}$.
- **Prop:** If the approximation error between $v^{\gamma}_{T,\Pi}$ and $v^{\gamma,T'}_{T,\Pi}$ is bounded by ϵ then
$$\|v^{\gamma}_{t,\Pi} - \tilde{v}_{t,\Pi}\|_\infty \leq \gamma^{T-t}\epsilon, \quad \forall t \leq T$$

Discounted and stationary: Bellman Equation

$$v_\Pi = \mathcal{T}^\pi v_\Pi$$

$$v_\Pi(s) = \sum_a \pi(a|s) r(s,a) + \gamma \sum_{s'} \sum_a p(s'|s,a) \pi(a|s) v_\Pi(s')$$

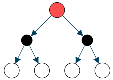


Bellman Equation

- Time independent value function v_Π.
- **Prop:** Unique solution of the linear equation $v_\Pi = \mathcal{T}^\pi v_\Pi$
- Complexity of order $(|A| + |S|) \times |S|^2$ to obtain the solution.

Discounted and stationary: Recursive Implementation

$$v_\Pi = \mathcal{T}^\pi v_\Pi$$
$$v_{k+1} = \mathcal{T}^\pi v_k \quad \text{with arbitrary } v_0$$



Bellman Iteration

- **Prop:** Unique fixed point of the Bellman operator $v \mapsto \mathcal{T}^\pi v$.
- **Prop:** The iterates $v_{k+1} = \mathcal{T}^\pi v_k$ converges toward v_Π and
$$\|v_k - v_\Pi\|_\infty \leq \gamma^k \|v_0 - v_\Pi\|_\infty$$
- Complexity of order $(k + |A|)|S|^2$ to obtain the kth iterate.
- Exponential decay of the error with respect to the complexity.

Bellman Operator and Contraction

$$\|\mathcal{T}^\pi v - \mathcal{T}^\pi v'\|_\infty \leq \gamma \|v - v'\|_\infty$$

Proof

- By definition
$$\|\mathcal{T}^\pi v - \mathcal{T}^\pi v'\|_\infty = \gamma \|P^\pi (v - v')\|_\infty$$
- It suffices then to notice that P^π is a transition matrix, so that
$$\sum_j P^\pi_{i,j} = 1$$
and thus $|\sum_j P^\pi_{i,j} z_j| \leq \max |z_j|$

Consequences

- Unicity of the solution of $\mathcal{T}^\pi v = v$.
- Linear decay γ^k of the error with the iterates.

Bellman Operator and Bellman Equation Solution

$$v_\Pi = \left(\sum_{k=0}^{\infty} \gamma^k \left(P^\pi\right)^k \right) r_\pi$$

A Closed Formula for the State Value Function

- $v_\Pi = \mathcal{T}^\pi v_\Pi \Leftrightarrow \left(I - \gamma P^\pi\right) v_\Pi = r_\pi$
- As P^π is a transition matrix, its eigenvalues are smaller than 1 and thus $\left(I - \gamma P^\pi\right)$ is invertible of inverse

$$\left(I - \gamma P^\pi\right)^{-1} = \sum_{k=0}^{\infty} \gamma^k \left(P^\pi\right)^k$$

- Could have been obtained without the Bellman equation as the $\left(\left(P^\pi\right)^k\right)_{s,s'}$ is, by construction, the probability of being at state s' at time k starting from s at time 0 and following Π.

Discounted and stationary: Value Iteration

Discounted: Prediction by Value Iteration

input: MDP model $\langle(\mathcal{S},\mathcal{A},\mathcal{R}),P\rangle$, discount factor γ, and stationary policy π
init: $\tilde{v}(s)\,\forall\, s\in\mathcal{S}$
repeat
  $\tilde{v}_{\text{prev}} \leftarrow \tilde{v}$
  for $s\in\mathcal{S}$ **do**

$$\tilde{v}(s) \leftarrow \sum_{a\in\mathcal{A}} \pi(a|s)\left(r(s,a)+\gamma\sum_{s'\in\mathcal{S}} p(s'|s,a)\tilde{v}_{\text{prev}}(s')\right)$$

  end
output: Value function $\tilde{v}$

- When to stop?

Discounted and stationary: Value Iteration

Discounted: Prediction by Value Iteration

input: MDP model $\langle(\mathcal{S},\mathcal{A},\mathcal{R}),P\rangle$, discount factor γ, and stationary policy π
parameter: $\delta > 0$ as accuracy termination threshold
init: $\tilde{v}(s)\,\forall\, s\in\mathcal{S}$
repeat
  $\tilde{v}_{\text{prev}} \leftarrow \tilde{v}$
  $\Delta \leftarrow 0$
  for $s\in\mathcal{S}$ **do**

$$\tilde{v}(s) \leftarrow \sum_{a\in\mathcal{A}} \pi(a|s)\left(r(s,a)+\gamma\sum_{s'\in\mathcal{S}} p(s'|s,a)\tilde{v}_{\text{prev}}(s')\right)$$

    $\Delta \leftarrow \max\left(\Delta, |\tilde{v}(s)-\tilde{v}_{\text{prev}}(s)|\right)$
  end
until $\Delta < \delta$
output: Value function $\tilde{v}$

- **Prop:** when the algorithms stops

$$\|\tilde{v}-v_\Pi\|_\infty \leq \frac{\gamma}{1-\gamma}\delta$$

Discounted: Prediction by Value Iteration - Gauss-Seidel Version

input: MDP model $\langle(\mathcal{S}, \mathcal{A}, \mathcal{R}), P\rangle$, discount factor γ, and stationary policy π
parameter: $\delta > 0$ as accuracy termination threshold
init: $\tilde{v}(s) \, \forall \, s \in \mathcal{S}$
repeat
  $\Delta \leftarrow 0$
  for $s \in \mathcal{S}$ **do**
    $\tilde{v}_{\text{prev}} \leftarrow \tilde{v}(s)$

$$\tilde{v}(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a|s) \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \tilde{v}(s') \right)$$

    $\Delta \leftarrow \max\left(\Delta, |\tilde{v}(s) - \tilde{v}_{\text{prev}}|\right)$
  end
until $\Delta < \delta$
output: Value function $\tilde{v}$

- Gauss-Seidel variation mostly used in practice.
- No need to store the previous value function.

Outline

Optimal Policy

Optimal Policy

- An optimal policy $\Pi_\star$ should be better than any other policies:

$$\forall s, \forall t, v_{t,\Pi_\star}(s) = \sup_{\Pi} v_{t,\Pi}(s)$$

Several Questions

- Do this policy exists?
- Is it unique?
- How to characterize it?
- How to obtain it?

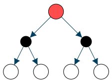
- Even the sup above could be an issue if it is not attained!

Finite Horizon and Optimal Policy

Explicit Recursive Solution

- After horizon T, any policy leads to a 0 return.
- At time $T-1$,
  - the total return G_T is the immediate return at time T and thus
  $$v_{T,\Pi^\star}(s) = \sup_{\pi(a|s)} \sum_a \pi(a|s) r(a,s) = \sup_a r(a,s)$$
  - the optimal policy $\pi^\star_{T-1}$ exists and is deterministic.
- By recursion,
  - the total return at time $t-1$ is the immediate return at time t plus the total return at time $t-1$ and thus
  $$\begin{aligned} v_{t-1,\Pi^\star}(s) &= \sup_{\pi(a|s)} \sum_a \pi(a|s) \left(r(a,s) + \sum_{s'} p(s'|s,a) v_{t,\Pi^\star} \right) \\ &= \sup_a \left(r(a,s) + \sum_{s'} p(s'|s,a) v_{t,\Pi^\star} \right) \end{aligned}$$
  - the optimal policy $\pi^\star_{t-1}$ exists and is deterministic.

Discounted Setting and Optimal Stationary Policy



Heuristic

- Optimal policy: $v^{\Pi^\star}(s) = \sup_\pi v_\Pi(s)$
- Stationary solution:

$$\begin{aligned} v_{\Pi^\star}(s) &= \sup_\pi \left(\mathcal{T}^\pi v_{\Pi^\star}\right)(s) \\ &= \sup_{\pi_t(\cdots|s)} \sum_a \pi(a|s) \left(r(a,s) + \gamma \sum_{s'} p(s'|s,a) v_{\Pi^\star}(s') \right) \\ &= \sup_a \left(r(a,s) + \gamma \sum_{s'} p(s'|s,a) v_{\Pi^\star}(s') \right) \end{aligned}$$

- Optimal deterministic policy: $\pi^\star(s) \in \operatorname{argmax}\left(r(a,s) + \gamma \sum_{s'} p(s'|s,a) v_{\Pi^\star}(s')\right)$.

- Is everything well defined? Yes but one has to be more cautious!

Optimal Value Function and Bellman Operator

Optimal Value Function

- Optimal value function: $v_\star(s) = \sup_\Pi v_\Pi(s)$
- Defined state by state so that it is not necessarily attained by a single $\Pi^\star$

Optimal Bellman operator

- Similar to the Bellman operator but do not depend on a policy:

$$\mathcal{T}^\star v(s) = \sup_a \left(r(a,s) + \gamma \sum_{s'} p(s'|s,a) v(s') \right)$$

Link between the two

- $v \geq \mathcal{T}^\star v$ implies $v \geq v_\star$.
- $v \leq \mathcal{T}^\star v$ implies $v \leq v_\star$.

Optimal Value Function and Bellman Operator

Bellman Operator and Fixed Point

- **Prop:** $\mathcal{T}^\star$ is a γ-contraction for the sup-norm and thus it exists a unique $v_{\star\star}$ such that $v_{\star\star} = \mathcal{T}^\star v_{\star\star}$.

Fixed Point and Optimal Value Function

- **Prop:** : $v_\star = v_{\star\star}$ and is thus the unique fixed point of $\mathcal{T}^\star$.
- **Proof:** $v_{\star\star} = \mathcal{T}^\star v_{\star\star}$ and thus $v_{\star\star} = v_\star$ according the link between the optimal value function and the Bellman operator.

- Does this mean something about policies?

Optimal Policy and Bellman Operator

Bellman Operator and Policy

- **Prop:** For any v, any policy π_v satisfying
$$\pi_v(s) \in \operatorname*{argmax}_a \left(r(a,s) + \gamma \sum_{s'} p(s'|s,a) v(s') \right)$$
is such that $\mathcal{T}^\star v(s) = \sup_\pi \mathcal{T}^\pi v(s) = \mathcal{T}^{\pi_v} v(s)$

Bellman Operator and Optimal Policy

- **Prop:** Any stationary policy $\pi_\star$ satisfying
$$\pi_\star(s) \in \operatorname*{argmax}_a \left(r(a,s) + \gamma \sum_{s'} p(s'|s,a) v^\star(s') \right)$$
is optimal.
- **Proof:** Indeed by construction, $\mathcal{T}^\star v_\star = \mathcal{T}^{\pi_\star} v_\star$ and thus, as $\mathcal{T}^\star v_\star = v_\star$, $v_{\pi_\star} = v_\star$.

Optimal Policy and Bellman Operator

Summary

- It exists a unique $v_\star$ such that $\mathcal{T}^\star v_\star = v_\star$
- $\forall s, v_\star(s) = \sup_\pi v_\pi(s)$
- Any policy $\pi_\star$ satisfying:

$$\forall s, \pi_\star(s) \in \operatorname*{argmax}_a \left(r(a,s) + \gamma \sum_{s'} p(s'|s,a) v^\star(s') \right)$$

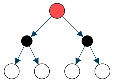
is optimal as $\forall s, v_{\pi_\star}(s) = v_\star(s) = \sup_\pi v_\pi(s)$

- Existence result but not (yet) a constructive algorithm!

Outline

Linear System and Linear Programming

$$v_\pi = \mathcal{T}^\pi v_\pi \qquad v_\star = \mathcal{T}^\star v_\star$$



Explicit Resolution of the Equations?

- Prediction:
  - Simple linear system for v_π.
  - Already mentionned before...
  - Complexity of order $(|A| + |S|)|S|^2$.
- Planning:
  - More complex linear programming system for $v_\star$ due to the max operator.
  - Optimal policy easily deduced from $v_\star$.
  - Complexity of order $(|A||S|)^3$.

Linear Programming

$$\text{From } \forall s, v(s) = \sup_a r(s,a) + \gamma \sum_{s'} p(s'|s,a)v(s')$$

$$\text{to } \min_v \sum_s \mu(s)v(s)$$

$$\text{such that } \forall (s,a), v(s) \geq r(s,a) + \gamma \sum_{s'} p(s'|s,a)v(s')$$

Different formulations but same solution

- Using $v \geq \mathcal{T}^\star v \Leftrightarrow v \geq v_\star$, the condition implies $v \geq v_\star$
- Now for any μ satisfying $\mu(s) > 0$, $\sum_s \mu(s)v(s) \geq \sum_s \mu(s)v_\star(s)$ as soon as the condition is satisfied, hence $v_\star$ is a solution.
- If for any state $v(s) > v_\star(s)$ then $\sum_s \mu(s)v(s) > \sum_s \mu(s)v_\star(s)$ and thus $v_\star$ is the unique minimizer.

Primal Problem

$$\text{Primal:} \quad \min_v \sum_s \mu(s) v(s)$$
$$\text{such that } \forall (s,a), v(s) \geq r(s,a) + \gamma \sum_{s'} p(s'|s,a) v(s')$$

Some properties

- Can be solved with a linear programming solver.
- Unicity of solution (and thus independence with respect to μ) can be proved without using $v_\star$.
  - **Proof:** let v_1 a solution for μ_1 and v_2 a solution for μ_2 then $\min(v_1, v_2)$ satifies the constraints. Furthermore if exists $v_2(s) < v_1(s)$ then $\min(v_1, v_2)$ is a strictly better solution for μ_2 which is impossible.

Dual Problem

Primal: $$\min_v \sum_s \mu(s)v(s)$$

such that $$\forall (s,a), v(s) \geq r(s,a) + \gamma \sum_{s'} p(s'|s,a)v(s')$$

Dual: $$\max_{\lambda(s,a)\geq 0} \sum_{s,a} \lambda(s,a)r(s,a)$$

such that $$\forall s, \sum_a \lambda(s,a) = \mu(s) + \gamma \sum_{s',a} p(s|s',a)\lambda(s',a)$$

Derivation

- Usual derivation through the Lagrangian:

$$\mathcal{L}(v,\lambda) = \sum_s \mu(s)v(s) + \sum_{s,a} \lambda(s,a) \left(r(s,a) + \gamma \sum_{s',a} p(s|s',a)v(s') - v(s) \right)$$

- Strong duality as Slater condition holds when $\gamma < 1$ with $v = \frac{1+\epsilon}{1-\gamma} \max_{s,a} r(s,a)$.

Dual and Interpretation

$$\text{Dual:} \quad \max_{\lambda(s,a)\geq 0} \sum_{s,a} \lambda(s,a) r(s,a)$$

$$\text{such that } \forall s, \sum_a \lambda(s,a) = \mu(s) + \gamma \sum_{s',a} p(s|s',a)\lambda(s',a)$$

$$\text{Interpretation} : \max_{\pi} \sum_{k=0}^{\infty} \gamma^k \sum_{s,a} \mathbb{P}(S_t = a, A_t = a | S_0 \sim \mu, \pi)\, r(s,a)$$

Interpretation in terms of policy

- For any feasible λ, define $u(s) = \sum_a \lambda(s,a)$ and the policy $\pi(a|s) = \lambda(s,a)/u(s)$.
- **Prop:** $u = (\mathrm{Id} - \gamma P^\pi)\mu = \sum_{k=0}^{\infty} \gamma^k \left(P^\pi\right)^k \mu$.
- **Prop:** $\lambda(s,a) = \pi(a|s)u(s) = \sum_{k=0}^{\infty} \gamma^k \mathbb{P}(S_t = a, A_t = a | S_0 \sim \mu, \pi)$
- Conversely for any π they is a feasible λ.
- Any optimal $\lambda_\star$ (and thus policy) satisfies $\lambda_\star(s,a) = 0$ if $v_\star(s) > r(s,a) + \gamma \sum_{s'} p(s'|s,a) v_\star(s')$ (optimal policy support)

Outline

Finite Horizon: Planning by Value Iteration

input: MDP model $\langle(\mathcal{S},\mathcal{A},\mathcal{R}),P\rangle$
parameter: Horizon T
init: $v_T^T(s)=0\,\forall\, s\in\mathcal{S},\ t=T$
repeat
  $t\leftarrow t-1$
  for $s\in\mathcal{S}$ **do**

$$v_t^T(s)\leftarrow \max_{a\in\mathcal{A}}\left(r(s,a)+\gamma\sum_{s'\in\mathcal{S}}p(s'|s,a)v_{t+1}^T(s')\right)$$

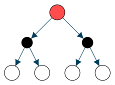
  end
until $t=0$

output: Deterministic policy $\pi_t(s)\in\operatorname{argmax}_{a\in\mathcal{A}}\left(r(s,a)+\gamma\sum_{s'\in\mathcal{S}}p(s'|s,a)v_{t+1}^T(s')\right)$

- Algorithm used to prove the existence of an optimal policy.
- No necessarily unique as argmax may not be unique.

Optimal Value Function, Fixed Point and Contraction

$$v_\star = \mathcal{T}^\star v_\star \quad \text{and} \quad \|\mathcal{T}^\star v - \mathcal{T}^\star v'\|_\infty \leq \gamma \|v - v'\|_\infty$$
$$\implies v_{k+1} = \mathcal{T}^\star v_k \to v_\star$$



Bellman Operator

- Properties of Optimal Bellman Operator:
  - $v_\star$ is a fixed point of $\mathcal{T}^\star$.
  - $\mathcal{T}^\star$ is a γ-contraction for the $\|\cdot\|_\infty$ norm.
- Classical fixed point theorem setting.
- Practical algorithm to approximate $v_\star$.

Value Iteration Algorithm

Discounted: Value Iteration Planning

input: MDP model $\langle(\mathcal{S}, \mathcal{A}, \mathcal{R}), P\rangle$, and discount factor γ
parameter: $\delta > 0$ as accuracy termination threshold
init: $\tilde{v}(s)\,\forall\, s \in \mathcal{S}$
repeat

  $\tilde{v}_{\text{prev}} \leftarrow \tilde{v}$
  $\Delta \leftarrow 0$
  for $s \in \mathcal{S}$ **do**

    $$\tilde{v}(s) \leftarrow \max_{a \in \mathcal{A}} r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a)\tilde{v}_{\text{prev}}(s')$$
    $$\Delta \leftarrow \max\left(\Delta, |\tilde{v}(s) - \tilde{v}_{\text{prev}}(s)|\right)$$

  end

until $\Delta < \delta$
output: Value function $\tilde{v}$

- Same convergence criterion (and similar proof) than in the planning case.
- Which policy?

Value Iteration Algorithm

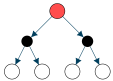
Discounted: Value Iteration Planning

input: MDP model $\langle(\mathcal{S},\mathcal{A},\mathcal{R}),P\rangle$, and discount factor γ
parameter: $\delta > 0$ as accuracy termination threshold
init: $\tilde{v}(s)\,\forall\, s \in \mathcal{S}$
repeat
  $\tilde{v}_{\mathrm{prev}} \leftarrow \tilde{v}$
  $\Delta \leftarrow 0$
  for $s \in \mathcal{S}$ **do**
    $\tilde{v}(s) \leftarrow \max_{a\in\mathcal{A}} r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a)\tilde{v}_{\mathrm{prev}}(s')$
    $\Delta \leftarrow \max\left(\Delta, |\tilde{v}(s) - \tilde{v}_{\mathrm{prev}}(s)|\right)$
  end
until $\Delta < \delta$
output: Deterministic policy $\tilde{\pi}(s) \in \operatorname{argmax}_a r(s,a) + \gamma \sum_{s'\in\mathcal{S}} p(s'|s,a)\tilde{v}(s')$

- Natural idea: define a policy using the argmax of the existence proof.
- Do we have a convergence guarantee on the resulting policy?

Value and argmax Policy

$$\tilde{\pi}(s) \in \operatorname*{argmax}_a r(s,a) + \gamma \sum_{s'} p(s'|s,a)\tilde{v}(s')$$
$$\implies \|v_{\tilde{\pi}} - v_\star\|_\infty \leq \frac{2\gamma}{1-\gamma}\|\tilde{v} - v_\star\|_\infty$$



Value and argmax Policy

- Bound on the loss of the final policy!
- Rely on the fact that, by construction, $\mathcal{T}^{\tilde{\pi}}\tilde{v} = \mathcal{T}^\star\tilde{v}$
- **Proof:**

$$\begin{aligned}
\|v_{\tilde{\pi}} - v_\star\|_\infty &= \|\mathcal{T}^{\tilde{\pi}} v_{\tilde{\pi}} - \mathcal{T}^{\tilde{\pi}}\tilde{v} + \mathcal{T}^\star\tilde{v} - \mathcal{T}^\star v_\star\|_\infty \\
&\leq \|\mathcal{T}^{\tilde{\pi}} v_{\tilde{\pi}} - \mathcal{T}^{\tilde{\pi}}\tilde{v}\|_\infty + \|\mathcal{T}^\star\tilde{v} - \mathcal{T}^\star v_\star\|_\infty \\
&\leq \gamma\|v_{\tilde{\pi}} - \tilde{v}\|_\infty + \gamma\|\tilde{v} - v_\star\|_\infty \\
&\leq \gamma\|v_{\tilde{\pi}} - v_\star\|_\infty + 2\gamma\|\tilde{v} - v_\star\|_\infty
\end{aligned}$$

Value Iteration Algorithm

Discounted: Value Iteration Planning

input: MDP model $\langle(\mathcal{S}, \mathcal{A}, \mathcal{R}), P\rangle$, and discount factor γ
parameter: $\delta > 0$ as accuracy termination threshold
init: $\tilde{v}(s) \, \forall \, s \in \mathcal{S}$
repeat

$\quad \tilde{v}_{\text{prev}} \leftarrow \tilde{v}$

$\quad \Delta \leftarrow 0$

$\quad$ **for** $s \in \mathcal{S}$ **do**

$$\tilde{v}(s) \leftarrow \max_{a \in \mathcal{A}} r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \tilde{v}_{\text{prev}}(s')$$

$$\Delta \leftarrow \max\left(\Delta, |\tilde{v}(s) - \tilde{v}_{\text{prev}}(s)|\right)$$

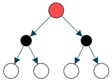
$\quad$ **end**

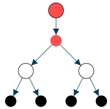
until $\Delta < \delta$

output: Deterministic policy $\tilde{\pi}(s) \in \operatorname{argmax}_a r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \tilde{v}(s')$

- **Prop:** $\|v_{\tilde{\pi}} - v_\star\|_\infty \leq \dfrac{2\gamma}{1-\gamma}\delta$

From State Value to State-Action Value Functions





$$v_\pi(s) = \mathbb{E}_\pi\left[\sum_k \gamma^k R_t | S_0 = s\right] \qquad q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_k \gamma^k R_t | S_0 = s, A_0 = a\right]$$

$$\mathcal{T}^\pi v(s) = \sum_a \pi(a|s)\left(r(s,a) + \gamma \sum_{s'} p(s'|s,a) v(s')\right) \qquad \mathcal{T}^\pi q(s,a) = r(s,a) + \sum_{s'} p(s'|s,a) \sum_a \pi(a|s') q(s',a)$$

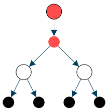
$$\mathcal{T}^\star v(s) = \max_a r(s,a) + \gamma \sum_{s'} p(s'|s,a) v(s') \qquad \mathcal{T}^\star q(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_a q(s',a)$$

Two equivalent point of view?

- Everything could have been defined using the state-action point of view.
- Knowing v_π is equivalent to knowing q_π as

$$v_\pi(s) = \sum_a \pi(s|a) q_\pi(s,a) \quad \text{and} \quad q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) v_\pi(s').$$

State-Action Bellman Operators



$$\mathcal{T}^{\pi} q(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \sum_{a} \pi(a|s') q(s',a)$$

$$\mathcal{T}^{\star} q(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a} q(s',a)$$

Properties

- **Prop:** $\mathcal{T}^{\pi}$ and $\mathcal{T}^{\star}$ are γ contractions for the $\|\cdot\|_{\infty}$ norm.
- **Prop:** q_{π} is the unique solution of $\mathcal{T}^{\pi} q = q$
- **Prop:** $q_{\star}$ defined $q_{\star}(s,a) = \sup_{\pi} q_{\pi}(s,a)$ is the unique solution of $q = \mathcal{T}^{\star} q$ and is attained for any policy $\pi_{\star}$ satisfying $\pi_{\star}(s) \in \operatorname{argmax} q_{\star}(s,a)$.
- **Prop:** Any such policy satisfies: $v_{\pi_{\star}}(s) = q_{\pi_{\star}}(s, \pi_{\star}(s)) = v_{\star}(s)$.

State-Action Value Iteration Algorithm

Discounted: Planning by State-Action Value Iteration

input: MDP model $\langle(\mathcal{S}, \mathcal{A}, \mathcal{R}), P\rangle$, and discount factor γ
parameter: $\delta > 0$ as accuracy termination threshold
init: $\tilde{q}(s, a)\,\forall\,(s, a) \in \mathcal{S} \times \mathcal{A}$
repeat

- $\tilde{q}_{\text{prev}} \leftarrow \tilde{q}$
- $\Delta \leftarrow 0$
- **for** $s \in \mathcal{S}$ **do**
  - **for** $a \in \mathcal{A}$ **do**
    - $$\tilde{q}(s, a) \leftarrow \left(r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) \max_{a'} \tilde{q}_{\text{prev}}(s', a') \right)$$
    - $\Delta \leftarrow \max\left(\Delta, |\tilde{q}(s, a) - \tilde{q}_{\text{prev}}(s, a)|\right)$
  - **end**
- **end**

until $\Delta < \delta$
output: Deterministic policy $\tilde{\pi}(s) \in \operatorname{argmax}_a \tilde{q}(s, a)$

- Same complexity but more storage than with state value function...
- but will be useful later!

Outline

Value Fonction vs Policy Point of View

$$v, q \longrightarrow \Pi \quad \text{or} \quad \Pi \longrightarrow v, q?$$

Planning

- Focus so far on value-fonction point of view!
- Heuristic: find a good approximation of the optimal value function and deduce a good policy.
- Can we work directly on the policy itself?

- For prediction, only the policy point of view makes sense!

Toward Policy Improvement

$$\forall s, \pi_+(s) \in \operatorname*{argmax}_a q_\pi(s, a) \implies \forall v_{\pi_+}(s) \geq v_\pi(s)$$

Classical Policy Improvement Lemma

- **Prop:** Given a policy π and its q value-function, one can obtain a better policy with the argmax operator.
- **Prop:** If no improvement is possible, it means that π is already optimal.
- **Proof:** Use $\mathcal{T}^{\pi_+} v_\pi = \mathcal{T}^\star v_\pi \geq \mathcal{T}^\pi v_\pi = v_\pi$ to prove $\left(\mathcal{T}^{\pi_+}\right)^k v_\pi \geq v_\pi$ which implies the result by letting k goes to $+\infty$.

- Leads to a sequential improvement algorith...

Policy Improvement Lemma

$$\mathbb{E}[v_{\pi'}(S_0)] - \mathbb{E}[v_\pi(S_0)] = \sum_{k=0}^{\infty} \gamma^k \mathbb{E}_{\pi'}\left[\sum_a \pi'(a|S_t)\left(q_\pi(S_t,a) - v_\pi(S_t)\right)\right]$$
$$= \sum_{k=0}^{\infty} \gamma^k \mathbb{E}_{\pi'}\left[\sum_a \left(\pi'(a|S_t) - \pi(a|S_t)\right) q_\pi(S_t,a)\right]$$

A Generic Improvement Lemma

- No assumptions on π and π'!
- Easy proof.
- Imply the previous lemma as $\max_a Q_\pi(s,a) - v_\pi(s) \geq 0$.
- Show that improvement choices are possible.

- Will prove to be useful later. . .

Discounted: Planning by Policy Iteration

input: MDP model $\langle(\mathcal{S}, \mathcal{A}, \mathcal{R}), P\rangle$, and discount factor γ
parameter: Initial policy $\tilde{\pi}$
repeat
    Compute $q_{\tilde{\pi}}$.
    for $s \in \mathcal{S}$ **do**
        for $a \in \mathcal{A}$ **do**
            $\tilde{pol}(s) \leftarrow \text{argmax}\, q_{\tilde{\pi}}(s, a)$
        end
    end
output: Deterministic policy $\tilde{\pi}$.

Some issues

- How to obtain q_π?
- When to stop?

Policy Iteration

Discounted: Planning by Policy Iteration

input: MDP model $\langle(\mathcal{S},\mathcal{A},\mathcal{R}),P\rangle$, and discount factor γ
parameter: Initial policy $\tilde{\pi}$
repeat
    stable $\leftarrow 0$
    Compute $q_{\tilde{\pi}}$.
    for $s \in \mathcal{S}$ **do**
        $old - action \leftarrow \tilde{\pi}(s)$
        $\tilde{\pi}(s) \leftarrow \mathrm{argmax}\, q_{\tilde{\pi}}(s,a)$
        if $\tilde{\pi}(s) \neq old - action$ **then**
            stable $\leftarrow 0$
        end
    end
until *stable* $=$*1*
output: Deterministic policy $\tilde{\pi}$.

Finite Setting

- Finite set of action-states implies a finite set of policy.
- Convergence of the algorithm in finite time!

Convergence Rate

- Crude analysis:
  - Bound after k steps of the algorithm

$$\|v_{\pi_k} - v_\star\|_\infty \leq \gamma \|v_{\pi_{k-1}} - v_\star\|_\infty \leq \gamma^k \|v_{\pi_0} - v_\star\|_\infty$$

$$\|v_{\pi_k} - v_\star\|_\infty \leq \frac{\gamma}{1-\gamma} \|v_{\pi_k} - v_{\pi_{k-1}}\|_\infty$$

  - Not much better than value iteration but much higher complexity as q_{π_k} is obtained by solving the Bellman equation!
- Much faster in practice...
- Clever analysis (Putterman):
  - Under some mild assumptions and provided $\|P^{\pi_k} - P^\star\| \leq K \|v_{\pi_k} - v_\star\|_\infty$ then

$$\|v_{\pi_k} - v_\star\|_\infty \leq \frac{K\gamma}{1-\gamma} \|v_{\pi_{k-1}} - v_\star\|_\infty^2$$

  - May explain the better convergence in practice!

Outline

Value Iteration: (Relaxed) First Order Method

Value Iteration

- Iteration:

$$\begin{aligned} v_k &= \mathcal{T}^\star v_{k-1} \\ &= v_{k-1} + (\mathcal{T}^\star - \mathrm{Id})\, v_{k-1} \end{aligned}$$

- Relaxation

$$v_k = v_{k-1} - \alpha \,(\mathrm{Id} - \mathcal{T}^\star)\, v_{k-1}$$

  can be proved to converge for any $\alpha < \frac{2}{1+\gamma}$.
- Can be interpreted as a first order method with pseudo-gradient $(\mathcal{T}^\star - \mathrm{Id})\, v_{k-1}$.
- No function corresponding to this gradient!

- Is there a better choice for α than $\alpha = 1$?
- No as the resulting operator is a contraction of constant

$$|1-\alpha| + \alpha\gamma \geq \gamma$$

Policy Iteration: Newton-Raphson Method

Policy Iteration

- Explicit iteration:

$$\text{Solve } v_{\pi_{k-1}} = \mathcal{T}^{\pi_k} v_{\pi_{k-1}}$$
$$\text{Let } \pi_k \text{ such that } \mathcal{T}^{\pi_k} v_{\pi_{k-1}} = \mathcal{T}^\star v_{\pi_{k-1}}$$

- Implicit iteration on v_{π_k}:

$$\begin{aligned} v_{\pi_k} &= (\mathrm{Id} - \gamma P^{\pi_k})^{-1} r_{\pi_k} \\ &= (\mathrm{Id} - \gamma P^{\pi_k})^{-1} \left(r_{\pi_k} + (\gamma P^{\pi_k} - \mathrm{Id}) v_{\pi_{k-1}} + (\mathrm{Id} - \gamma P^{\pi_k}) v_{\pi_{k-1}} \right) \\ &= v_{\pi_{k-1}} - (\mathrm{Id} - \gamma P^{\pi_k})^{-1} (\mathrm{Id} - \mathcal{T}^{\pi_k}) v_{\pi_{k-1}} \end{aligned}$$

- Can be interpreted as a second order method with pseudo-gradient $(\mathrm{Id} - \mathcal{T}^{\pi_k}) v_{\pi_{k-1}} = (\mathrm{Id} - \mathcal{T}^\star) v_{\pi_{k-1}}$ and pseudo-Hessian $(\mathrm{Id} - \gamma P^{\pi_k})$.

- Not a formal analysis but give a good insight on the better convergence of policy iteration.

Outline

Stability of Value and Policy Iteration

Ideal Value and Policy Iteration?

- Iterative algorithms.
- Convergence proofs assume perfect computation.
- What happens if we make a (small) error at each step?

- Particularly important for Policy Iteration in which one resolves a linear system at each step!

Value Iteration Stability

$$v_k = \mathcal{T}^\star v_{k-1} + \epsilon_{k-1}$$

$$\implies \|v_k - v_\star\|_\infty \leq \gamma^k \|v_0 - v_\star\|_\infty + \frac{\max_{0\leq k'<k} \|\epsilon_{k'}\|_\infty}{1-\gamma}$$

$$\implies \|v_{\pi_k} - v_\star\|_\infty \leq \frac{2\gamma^{k+1}}{1-\gamma}\|v_0 - v_\star\|_\infty + \frac{2\gamma \max_{0\leq k'<k} \|\epsilon_{k'}\|_\infty}{(1-\gamma)^2}$$

Stability with respect to the error

- Proof relies on the contraction property of $\mathcal{T}^\star$ (hence similar results for $\mathcal{T}^\pi$).
- Error term $\frac{\max_{0\leq k'<k} \|\epsilon_{k'}\|_\infty}{1-\gamma}$ can be replaced by $\sum_{k'=0}^{k-1} \gamma^{k-k'} \|\epsilon_{k'}\|_\infty$
- Convergence if $\|\epsilon_k\|_\infty$ tends to 0.
- Remains in a neighborhood of the optimal solution if $\|\epsilon_k\|_\infty$ is bounded.

$$v_{k-1} = v_{\pi_{k-1}} + \epsilon_{k-1} \quad \text{and} \quad \mathcal{T}^{\pi_k} v_{k-1} = \mathcal{T}^{\star} v_{k-1}$$

$$\implies \|v_{\pi_k} - v_\star\|_\infty \leq \gamma^k \|v_{\pi_0} - v_\star\|_\infty + \frac{\gamma(2-\gamma) \max_{0 \leq k' < k} \|\epsilon_{k'}\|_\infty}{(1-\gamma)^2}$$

Stability with respect to the error

- Quite involved proof but crude results.
- Error term $\frac{\max_{0 \leq k' < k} \|\epsilon_{k'}\|_\infty}{1-\gamma}$ can be replaced by $\sum_{k'=0}^{k-1} \gamma^{k-k'} \|\epsilon_{k'}\|_\infty$
- Convergence if $\|\epsilon_k\|_\infty$ tends to 0.
- Remains in a neighborhood of the optimal solution if $\|\epsilon_k\|_\infty$ is bounded.

- Policy Iteration only requires an approximate estimate of $v_{\pi_{k-1}}$, for instance obtained by Bellman iteration...

Outline

Modified Policy Iteration

Discounted: Planning by Generalized Policy Iteration

input: MDP model $\langle(\mathcal{S}, \mathcal{A}, \mathcal{R}), P\rangle$, and discount factor γ
parameter: Initial q
repeat
  for $s \in \mathcal{S}$ **do**
    $\tilde{\pi}(s) \leftarrow \operatorname{argmax}_a q(s, a)$
  end
  repeat
    $q_{\text{prev}} \rightarrow q$
    for $(s, a) \in \mathcal{S} \times \mathcal{A}$ **do**
      $q(s, a) \leftarrow r(s, a) + \gamma \sum_{s, a'} p(s'|s, a)\tilde{\pi}(a'|s)q_{\text{prev}}(s, a)$
    end
output: Deterministic policy $\tilde{\pi}$.

- Algorithm driven by q.
- Flexibility in the number of prediction steps after each policy improvement steps.
- Special cases:
    - Large number: Policy Iteration with (small) error.
    - One: Value Iteration!

MPI Analysis

$$\mathcal{T}^{\pi_k} v_k = \mathcal{T}^\star v_k \quad \text{and} \quad v_{k+1} = \left(\mathcal{T}^{\pi_k}\right)^{m_k} v_k$$

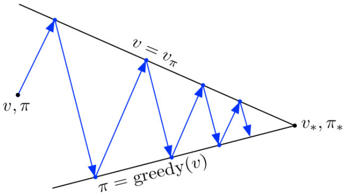
$$\Longrightarrow \|v_{k+1} - v_\star\|_\infty \leq \gamma \left(\frac{1-\gamma^{m_k}}{1-\gamma} \|P^{\pi_k} - P^\star\| + \gamma^{m_k} \right) \|v_k - v_\star\|_\infty$$

Convergence Results

- Quite technical proof.
- Valid only under the mild assumption $\mathcal{T}^\star v_0 \geq v_0$.
- Very fast decay provided $\|P^{\pi_k} - P^\star\|$ is small.

- No stability with arbitrary errors. . .

Generalized Policy Iteration



General Policy Iteration

- Two simultaneous interacting processes:
  - One forcing the policy to correspond to the current value function (Policy Improvement)
  - One trying to male the current value function coherent with the current policy (Policy Evaluation)
- Several variations possible on the two processes.

- In GPI, the policy is driven by the value function.
- Typically, stabilizes only if one reaches the optimal value/policy pair.

Source: Sutton and Barto

Discounted: Prediction by Value Iteration - State Update Order

input: MDP model $\langle(\mathcal{S},\mathcal{A},\mathcal{R}),P\rangle$, discount factor γ, and stationary policy π
init: $\tilde{v}(s)\,\forall\, s\in\mathcal{S}$
repeat
  $\tilde{v}_{\text{prev}} \leftarrow \tilde{v}$
  for $s\in\mathcal{S}'\subset\mathcal{S}$ **do**

$$\tilde{v}(s) \leftarrow \sum_{a\in\mathcal{A}} \pi(a|s)\left(r(s,a)+\gamma\sum_{s'\in\mathcal{S}} p(s'|s,a)\tilde{v}_{\text{prev}}(s')\right)$$

  end
output: Value function $\tilde{v}$

Classical strategies

- $\mathcal{S}'=\mathcal{S}$: classical iteration
- $\mathcal{S}'=\{s\}$: Gauss-Seidel
- $\mathcal{S}'=\{s, |\mathcal{T}^{\pi}\tilde{v}(s)-\tilde{v}(s)|>\epsilon\}$: Prioritized sweeping

- Converges provided all states are visited infinitely often...
- Gain in term of storage or focus on most interesting states...

Policy Improvement Variation

$$\text{Greedy}: \pi(s) \in \operatorname*{argmax}_a q(s,a) \iff \pi(\cdot|s) \in \operatorname*{argmax}_{\tilde{\pi}} \sum_a \tilde{\pi}(a) q(s,a)$$

$$\text{Restricted}: \pi(\cdot|s) \in \operatorname*{argmax}_{\tilde{\pi}\in\tilde{\Pi}_\epsilon} \sum_a \tilde{\pi}(a) q(s,a)$$

$$\text{Regularized}: \pi(\cdot|s) \in \operatorname*{argmax}_{\tilde{\pi}} \sum_a \tilde{\pi}(a) q(s,a) + \epsilon P(\tilde{\pi})$$

Classical Variations

- ϵ-greedy: Restrict $\tilde{\pi}$ to the set of policy s.t. $\tilde{\pi}(a) \geq \epsilon$
  - Explicit solution: $\pi(a|s) = \epsilon + (1-\epsilon) \operatorname{argmax} q(s,a)$
  - Policy improvement property if ϵ decreases.
- Soft-max: Regularize by $\epsilon H(\tilde{\pi})$ where H is the entropy.
  - Explicit solution: $\pi(a|s) \propto \exp(q(s,a)/\epsilon)$
  - No classical policy improvement...

- Tends to greedy when ϵ goes to 0.
- Turn out to be interesting later...

Outline

Episodic Setting

$$\mathbb{E}_\pi\left[\min_t\{t, \forall t' \geq t,\ R_{t'} = 0\}\right] < H \Rightarrow \|\mathcal{T}v - \mathcal{T}v'\|_\xi \leq \frac{H-1}{H}\|v - v'\|_\xi$$

Proper Policy

- A policy π is said to be H-proper if $\mathbb{E}_\pi\left[\min_t\{t, \forall t' \geq t,\ R_{t'} = 0\}\right] \leq H < \infty$
- $\Leftrightarrow$ average duration of an episode using this policy less than a finite horizon H!

Bellman operators

- If a policy π is H-proper, the Bellman operator $\mathcal{T}^\pi$ is a $(H-1)/H$- contraction for a weighted sup-norm.
- If all the policies are H-propers, the optimal Bellman operator $\mathcal{T}^\star$ is a $(H-1)/H$-contraction for a weighted sup-norm.

- Under those strong assumptions, episodic setting $\simeq$ discounted setting with $\gamma = (H-1)/H$.
- Some results can be obtained under the much milder assumption that there is one proper policy and that any non-proper policy has at least one state for which $v_\pi(s) = -\infty$.

Infinite Setting

- No issue with the rewards, as only the expectation is used.
- All the theory remains valid if the states are countable, but there is an issue in the algorithms, as we need to store/update an infinite number of states.
- The proof of existence of an optimal policy requires the max to be attained, which cannot be ensured in an infinite (even countable setting).

Some results. . .

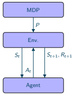
- **Thm:** If S is countable, there exists an ϵ-optimal (stationary) policy for any $\epsilon > 0$.
- **Thm:** If S is a Polish space (completely metrizable topological space),
  - there exists a (P, ϵ)-optimal (stationary policy) for any $\epsilon > 0$.
  - if each A_s is countable, there exists an ϵ-optimal (stationary) policy for any $\epsilon > 0$.
  - if each A_s is finite, there exists an optimal (stationary) policy.
  - if each A_s is a compact metric space, $r(s, a)$ is a bounded u.s.c. function on A_s and $p(B|s, a)$ is continuous in a for each Borel subset B and any s, there exists an optimal (stationary) policy.

- **Mainly technical difficulties. . .**

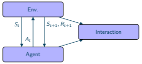
Outline

Reinforcement Learning

From



to



From Probability to Statistics?

- What to do if one has no knowledge of the underlying MDP?
- Only information through interactions!
- Prediction? Planning?
- Focus on the discounted setting

Outline

Monte Carlo, i.e. Just Play!

- Most simple way to evaluate a policy.

Just Play Following Policy Π

- Play N episodes following the policy.
- During each episode, compute the (discounted) gain.
- Compute the average gain.

- What is computed?

Average Gain or Value Function

$$\mathbb{E}[G_0] \quad \text{vs} \quad v_{t,\Pi}(s) = \mathbb{E}[G_t | S_t = s]$$

Prediction as Value Function Evaluation

- Not the same goal.
- By construction,

$$\mathbb{E}[G_0] = \sum_s \mu_0(s) v_{t,\Pi}(s)$$

- Much easier to compute the average gain than the value function (even if we use a stationary policy)

- Average gain is nevertheless the most classical way to evaluate a policy (with a single number).
- Implicit episodic setting if we do not want to use approximated gain.

Average Gain Estimation

Episodic: Evaluation by MC

input: MDP environment, initial state distribution μ_0, policy Π and discount factor γ
parameter: Number of episodes N
init: $V = 0$,$n = 0$
repeat
  $n \leftarrow n + 1$
  $t \leftarrow 0$
  $G \leftarrow 0$
  Pick initial state S_0 following μ_0
  repeat
    Pick action A_t according to $\pi(\cdot|S_t)$
    $G \rightarrow G + \gamma^t R_{t+1}$
    $t \leftarrow t + 1$
  until *episod ends at time* T
  $V \leftarrow V + G$
until $n == N$
$V \leftarrow V/N$
output: Average gain V

Monte Carlo Prediction

- How to estimate $v_{t,\Pi}$?

Just Play Following Policy Π

- Play N episodes following the policy.
- During episode, record S_t and R_t.
- After each episode, compute recursively for each time t the gain G_t.
- Estimate $v_{t,\Pi}(s)$ by the average G_t over all trajectories such that $S_t = s$

- **May require a lot of game to have a non empty set for each state s at each time t**

Monte Carlo Prediction

- How to estimate v_Π for a stationary policy?

Just Play Following Policy Π

- Play N episodes following the policy.
- During each episode, record S_t and R_t.
- After each episode, compute recursively for each time t the gain G_t.
- Estimate $v_\Pi(s)$ by the average over all trajectories of all G_t such that $S_t = s$, whatever t.

- The same state may be reached several time during a single episode...
- First-visit variant: Use only the first visit of s for each episode.

Monte Carlo Prediction

Episodic: Prediction by MC

input: MDP environment, initial state distribution μ_0, policy Π and discount factor γ
parameter: Number of episodes N
init: $\forall s$, $V(s)$, $n = 0$, $N(s) = 0$
repeat
  $n \leftarrow n + 1$
  $t \leftarrow 0$
  Pick initial state S_0 following μ_0
  repeat
    (If First-visit) $N(S_t) \leftarrow N(S_t) + 1$
    Pick action A_t according to $\pi(\cdot|S_t)$
    Record R_{t+1}, S_{t+1}
    $t \leftarrow t + 1$
  until *episod ends at time* T
  $G_{T+1} = 0$
  $t \rightarrow T + 1$
  repeat
    $t \leftarrow t - 1$
    Compute $G_t = R_{t+1} + \gamma G_{t+1}$
    (If First-visit) $V(S_t) = V(S_t) + G_t$
  until $t = 0$
until $n == N$
for $s \in \mathcal{S}$ **do**
  $V(s) \leftarrow V(s)/N(s)$
end
output: Value function V

Monte Carlo Prediction Analysis

First-Visit Variant Analysis

- Straightforward analysis as all the used values for a given state s are independent.
- Variance of order $1/N(s)$ where $N(s)$ is the number of episod where s is visited.
- Convergence if the number of visit goes to ∞.
- Strong assumption is practice as some states may not be visited by a given policy (if we cannot play on the initial state).

- Every-visit works... but not necessarily better!

Outline

Monte Carlo Planning

- Can we use a MC approach to find a good policy?

A First Attempt

- Estimate $v_\pi(s)$ by $V_\pi(s)$ using MC.
- Compute $Q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_\pi(s)$
- Enhance the current policy by setting $\pi(s) = \operatorname{argmax}_a Q_\pi(s,a)$

- Inspired by the Operations Research results. . .
- But unusable as r and p are unknown!

Monte Carlo Planning

A Second Attempt

- Estimate $q_\pi(s, a)$ by $Q_\pi(s, a)$ using MC.
- Enhance the current policy by setting $\pi(s) = \operatorname{argmax}_a Q_\pi(s, a)$

- Requires that $N(s, a)$ the number of times that an episode contains the state s followed by action a goes to ∞.
- Impossible with a deterministic policy!

Monte Carlo Planning

Classical Exploratory Policies...

- Stochastic policies ensuring that any action can occurs at any state.
- ϵ-exploratory policy: use a determistic policy and replace it with a random action with probability ϵ.
- Gibbs policy: use a policy where $\pi(a|s) \propto e^{G(a,s)} > 0$.

A Final Attempt

- Start from an exploratory policy.
- Estimate $q_\pi(s,a)$ by $Q_\pi(s,a)$ using MC.
- Enhance the current policy while remaining a exploratory policy.

- Last step is not straightforward...
- except for ϵ-deterministic policy for which the ϵ-exploratory policy with base policy $\pi(s) = \operatorname{argmax}_a Q_\pi(s,a)$ works.
- No convergence proof.

Outline

Advanced Implementation of Monte Carlo Prediction

$$V_\pi(S_t) \leftarrow V_\pi(S_t) + \alpha(N(S_t))(G_t - V_\pi(S_t))$$

On-Line Monte Carlo

- Average for a given state can be updated each time we have the gain G_t for a state S_t.
- Just use $\alpha(N) = 1/N$ and increment $N(S_t)$.
- No need to record the values between episodes...

- We still need to wait until the end of each episode to compute G_t.
- Can we do better?

Advanced MC Prediction

Episodic: Prediction by MC

input: MDP environment, initial state distribution μ_0, policy Π and discount factor γ
parameter: Number of episodes N
init: $\forall s, V(s), n = 0, N(s) = 0$
repeat
    $n \leftarrow n + 1$
    $t \leftarrow 0$
    Pick initial state S_0 following μ_0
    repeat
        (If First-visit) $N(S_t) \leftarrow N(S_t) + 1$
        Pick action A_t according to $\pi(\cdot|S_t)$
        Record R_{t+1}, S_{t+1}
        $t \leftarrow t + 1$
    until *episod ends at time* T
    $G_{T+1} = 0$
    $t \rightarrow T + 1$
    repeat
        $t \leftarrow t - 1$
        Compute $G_t = R_{t+1} + \gamma G_{t+1}$
        (If First-visit) $V(S_t) = V(S_t) + \frac{1}{N(S_t)}(G_t - V(S_t))$
    until $t = 0$
until $n == N$
output: Value function V

- We still need to wait until the end of each episode to compute G_t.
- Can we do better?

Prediction with Temporal Differencies

$$\text{From} \quad V_\pi(S_t) \leftarrow V_\pi(S_t) + \alpha(N(S_t))(G_t - V_\pi(S_t))$$
$$\text{to} \quad V_\pi(S_t) \leftarrow V_\pi(S_t) + \alpha(N(S_t))\underbrace{(R_{t+1} + \gamma V_\pi(S_{t+1}) - V_\pi(S_t))}_{\delta_t}$$

Bootstrap Strategy

- Replace G_t by an instantaneous estimate $R_{t+1} + \gamma V_\pi(S_{t+1})$.
- Amounts to replace $\gamma R_{t+2} + \gamma^2 R_{t+1}$ by an approximation of its expectation given S_{t+1}: $v_\pi(S_{t+1})$.
- Bootstrap as we use the current estimate $V_\pi(S_{t+1})$ instead of the true value.
- $\delta_t = R_{t+1} + \gamma V_\pi(S_{t+1}) - V_\pi(S_t)$ is called a temporal difference.

- No need to wait until the end of the episodes!
- Can be used in the discounted setting.

TD Prediction

Discounted: Prediction by TD

input: MDP environment, initial state distribution μ_0, policy Π and discount factor γ
parameter: Number of step T
init: $\forall s$, $V(s), n = 0, N(s) = 0,\ t' = 0$
repeat
  $t \leftarrow 0$
  Pick initial state S_0 following μ_0
  repeat
    $N(S_t) \leftarrow N(S_t) + 1$
    Pick action A_t according to $\pi(\cdot|S_t)$
    $V(S_t) \leftarrow V(S_t) + \alpha(N(S_t))\,(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$
    $t \leftarrow t + 1$
  until *episod ends at time* T' *or* $t' = T$
until $t' = T$
output: Value function V

- **But does this work?**

Prediction with Temporal Differencies

$$\mathbb{E}[\delta_t|S_t]\,\mathbb{E}[R_{t+1}+\gamma V_\pi(S_{t+1})-V_\pi(S_t)|S_t]=(\mathcal{T}^\pi-\mathrm{Id})\,V_\pi(S_t)$$

TD and Bellman Operator

- TD as an approximate Policy Iteration:
$$\mathbb{E}[V_\pi](S_t)\leftarrow V_\pi+\alpha(N(S_t))\,(\mathcal{T}^\pi-\mathrm{Id})\,V_\pi(S_t)$$
- Proof of convergence of this algorithm to a zero of $\mathcal{T}^\pi-\mathrm{Id}$, i.e. the fixed point of $\mathcal{T}^\pi$!
- Proof requires a mild assumption of α (satisfied by $\alpha(N)=1/N$) and the strong assumption that $N(s)$ goes to ∞.

- MC could be interpreted in a similar way (stochastic approximation) by noticing that $\mathbb{E}[G_t-V_\pi(S_t)|S_t]=v_\pi(S_t)-V_\pi(S_t)$.
- Often use with a constant α

MC vs TD

$$
\begin{aligned}
& V_\pi(S_t) \leftarrow V_\pi(S_t) + \alpha(N(S_t))(G_t - V_\pi(S_t)) \\
\text{or} \quad & V_\pi(S_t) \leftarrow V_\pi(S_t) + \alpha(N(S_t)) \underbrace{(R_{t+1} + \gamma V_\pi(S_{t+1}) - V_\pi(S_t))}_{\delta_t}
\end{aligned}
$$

MC vs TD

- Both are based on stochastic approximation.
- Both converges (under similar assumptions) to the correct value function.
- TD does not require to wait until the end of the episode.
- No theorical difference in the speed of convergence but often TD is better...
- Solve different approximate problems when used with a finite set of episodes:
  - MC compute the empirical gain from any state.
  - TD compute the value function of the empirical Bellman operator (the one obtained by using the empirical transition probabilities)

- If V_π is kept constant during an episode

$$G_t - V_\pi(S_t) = \sum \gamma^{t'-t} \delta_t$$

Outline

Stochastic Approximation

$$\theta_{k+1} = \theta_k + \alpha_k h_k(\theta_k) \quad \text{with} \quad h_k(\theta) = H(\theta) + \epsilon_k + \eta_k$$
$$\implies \theta_k \to \{\theta, H(\theta) = 0\}$$

Stochastic Approximation

- Family of sequential stochastic algorithm converging to a zero of a function.
- Classical assumptions:
  - $\mathbb{E}[\epsilon_k] = 0$, $\mathbb{V}\text{ar}\,[\epsilon_k] < \sigma^2$, and $\mathbb{E}[\|\eta_k\|] \to 0$,
  - $\sum_k \alpha_k \to \infty$ and $\sum_k \alpha_k^2 < \infty$,
  - the algorithm converges if we replace h_k by H.

- Convergence toward a neighborhood if α is kept constant (as often in practice).
- Most famous example are probably Robbins-Monro and SGD.
- Proof quite technical in general.
- The convergence with H is easy to obtain for a contraction.

Stochastic Approximation and ODE

From $\theta_{k+1} = \theta_k + \alpha_k h_k(\theta_k)$ with $h_k(\theta) = H(\theta) + \epsilon_k + \eta_k$

to $\dfrac{d\tilde{\theta}}{dt} = H(\tilde{\theta})$

ODE Approach

- General proof showing that the algorithm converges provided the ODE converges.
- Rely on the rewriting the equation
$$\frac{\theta_{k+1} - \theta_k}{\alpha_k} = h_k(\theta_k) = H(\theta_k) + \epsilon_k + \eta_k$$
- α_k can be interpreted as a time difference allowing to define a time $t_k = \sum_{t' \leq t} \alpha_k$.
- $\theta(t)$ is piecewise affine and defined through its derivative at time $t \in (t_k, t_{k+1})$.
- This piecewise function remains close to any solution of the ODE starting from θ_k for an arbitrary amount of time provided k is large enough.

- More general proofs based on martingale.

Asynchronous Update

$$\text{From} \quad \theta_{k+1} = \theta_k + \alpha_k h_k(\theta_k) \quad \text{with} \quad h_k(\theta) = H(\theta) + \epsilon_k + \eta_k$$
$$\text{to} \quad \forall i, \theta_{k+1}(i) = \theta_k(i) + \alpha_k(i) h_k(\theta_k)(i)$$

Asynchronous Update

- Componentwise action on θ.
- Not necessarily the same stepsize $\alpha_k(i)$ for all components.
- $\alpha_k(i) = 0$ is permitted!
- Previous results hold provided for every component i, $\sum_k \alpha_k(i) \to \infty$ and $\sum_k \alpha_k^2(i) < \infty$,

- Exact setting of TD approximation!

Outline

Planning with Temporal Differencies

A State Value Function Attempt

- $V_\star$ is the fixed point of $\mathcal{T}^\star$.
- Approximate it as the zero of $\mathcal{T}^\star - \mathrm{Id}$.
- By construction

$$\mathcal{T}^\star v(S_t) = \max_a \mathbb{E}[R_{T+1} + \gamma v(S_{t+1}) | S_t, a]$$

- Not an expectation!

A State-Action Value Function Attempt

- $q_\star$ is the fixed point of $\mathcal{T}^\star$.
- Approximate it as the zero of $\mathcal{T}^\star - \mathrm{Id}$.
- By construction

$$\mathcal{T}^\star q(S_t, A_t) = \mathbb{E}\left[R_{t+1} + \gamma \max_a q(S_{t+1}, a) \middle| S_t, A_t\right]$$

- An expectation!

Q Learning

Discounted: Planning by Q-Learning

input: MDP environment, initial state distribution μ_0, policy Π and discount factor γ
parameter: Number of step T
init: $\forall s, a, Q(s, a), N(s, a) = 0$, n=0, $t' = 0$
repeat
- $t \leftarrow 0$
- Pick initial state S_0 following μ_0
- **repeat**
  - $N(S_t) \leftarrow N(S_t) + 1$
  - Pick action A_t according to $\pi(\cdot|S_t)$
  - $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(N(S_t, A_t)) \left(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right)$
  - $t \leftarrow t + 1$
  - $t' \leftarrow t' + 1$
- **until** *episod ends at time T' or $t' = T$*

until $t' = T$
output: Deterministic policy $\tilde{\pi}(s) = \text{argmax}_a Q(s, a)$

Planning with Q Learning

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha(N(S_t, A_t)) \left(\underbrace{R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)}_{\delta_t} \right)$$

Q-Learning

- Update is independent of the policy Π.
- Convergence of the Q-value function provided the policy is such that $N(s, a)$ tends to ∞ for any state and any action.
- Implies a convergence of the policy.
- Relies on temporal difference.

- Most classical (tabular) planning algorithm!

Outline

Planning with Policy Improvement

$$\text{from} \quad Q(S_t, A_t) = Q(S_t, A_t) + \alpha(N(S_t, A_t)) \left(\underbrace{R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)}_{\delta_t} \right)$$

$$\text{to} \quad Q(S_t, A_t) = Q(S_t, A_t) + \alpha(N(S_t, A_t)) \left(\underbrace{R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)}_{\delta_t} \right)$$

$$\Pi(S_t) = \operatorname*{argmax}_a Q(S_t, a) (\text{plus exploration})$$

Policy Improvement

- More emphasis on the policy with a link between the policy used to play and the optimized policy.
- Almost equivalent to use the current policy in the Q-Learning algorithm.

Discounted: Planning by SARSA

input: MDP environment, initial state distribution μ_0, policy Π and discount factor γ
parameter: Number of step T
init: $\forall s, a, Q(s,a), N(s,a) = 0$, n=0, $t' = 0$
repeat
- $t \leftarrow 0$ Pick initial state S_0 following μ_0
- **repeat**
  - $N(S_t) \leftarrow N(S_t) + 1$
  - Pick action A_t according to $\pi(\cdot|S_t)$
  - $Q(S_{t-1}, A_{t-1}) \leftarrow Q(S_{t-1}, A_{t-1}) + \alpha(N(S_{t-1}, A_{t-1}))\,(R_t + \gamma Q(S_t, A_t) - Q(S_{t-1}, A_{t-1}))$
  - $\Pi(S_{t-1}) = \operatorname{argmax}_a Q(S_{t-1}, a)$ (plus exploration)
  - $t \leftarrow t + 1$
  - $t' \leftarrow t' + 1$
- **until** *episod ends at time T' or $t' = T$*

until $t' = T$
output: Deterministic policy $\tilde{\pi}(s) = \operatorname{argmax}_a Q(s,a)$

- Does this work?

SARSA and exploration

$$\Pi(S_t) = \operatorname*{argmax}_a Q(S_t, a)(\text{plus exploration})$$

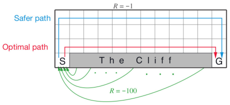
SARSA and Exploration

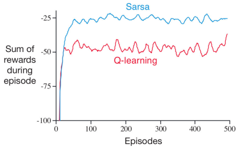
- No hope of convergence if we do not explore all possible actions (and states).
- Impossible if the policy used is deterministic.
- Exploration is required!
- Most classical choice: ϵ-greedy policy with a decaying ϵ.

- Convergence proof is harder than for Q-Learning.
- Relies on the similarity in the limit (when ϵ goes to 0) with the Q-Learning algorithm.

Outline

Q-Learning vs SARSA





How different are they?

- In Q-learning, the exploratory policy used is decoupled from the optimized policy.
- This exploratory policy may yield low rewards on average.
- In SARSA, the two policies are linked with the hope on having higher rewards during the learning step.
- Subtle different behavior even if we modify the exploratory policy in Q-Learning.

Exploration vs Exploitation

Exploration vs Exploitation

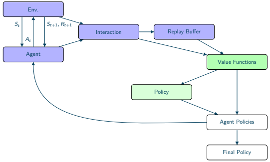
- Exploration: explore new policies to be able to discover the best ones.
- Exploitation: use good policies to obtain a good return.
- Exploration is a requirement.

- No tradeoff if we optimize only the final result!
- Tradeoff between the two if we consider that the returns during training matters!
- Q-learning use the first approach and SARSA try to tackle the second.
- Tradeoff if we study a regret:
$$\sum_t \mathbb{E}_{\Pi_\star}[R_t] - \mathbb{E}_{\Pi_t}[R_t]$$
which forces us to be good as fast as possible.
- No natural definition in the discounted setting.

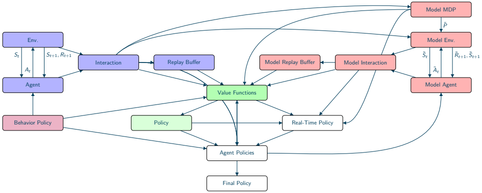
Outline

Advanced Tabular Reinforcemcent Learning

From



to



- Core idea: Approximate Bellman Operators with Stochastic Approximation...

Advanced Ideas?

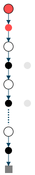
- Between MC and TD?
- Off-policy vs on-policy?
- Exploration vs Exploitation?
- Model? Replay?
- Real-Time Planning?

Outline

n-steps

or

or



How many steps before backup?

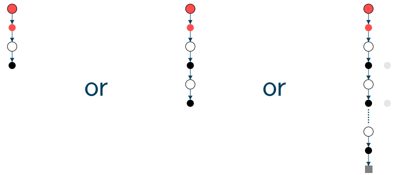
- One step: TD.
- As many steps as required to end the episod: MC.
- n-steps: n-steps TD.

$$\left(\mathcal{T}^{\Pi}\right)^{n} v(s) = \mathbb{E}_{\Pi}\left[\underbrace{R_{t+1} + \gamma R_{t+2} + \gamma^{n-1} R_{t+n} + \gamma^{n} v(S_{t+n})}_{G_{t:t+n}} \middle| S_t = s\right]$$

- Family of stochastic approximation algorithms:

$$V(S_t) \leftarrow V(S_t) + \alpha(N(S_t))\left(G_{t:t+n} - V(S_t)\right)$$

n-steps TD



$$V(S_t) \leftarrow V(S_t) + \alpha(N(S_t))\left(G_{t:t+n} - V(S_t)\right)$$

n-steps TD

- Convergence for prediction.
- Need to be combined with Policy Improvement for planning: n-steps SARSA.
- n-steps Q-learning could be an extension of API... but this means following the optimized policy Π... i.e. SARSA!

- Best convergence often for intermediate n.
- No proof beside TD for $n > 1$!

Discounted: Prediction by n-steps TD

input: MDP environment, initial state distribution μ_0, policy Π and discount factor γ
parameter: Number of step T
init: $\forall s, a, Q(s,a), N(s,a) = 0$, n=0, $t' = 0$
repeat
- $t \leftarrow 0$
- Pick initial state S_0 following μ_0
- **repeat**
  - $N(S_t) \leftarrow N(S_t) + 1$
  - Pick action A_t according to $\pi(\cdot|S_t)$
  - $Q(S_{t-n}, A_{t-n}) \leftarrow Q(S_{t-n}, A_{t-n}) + \alpha(N(S_t, A_t))\,(G_{t-n:t} - Q(S_t, A_t))$
  - $t \leftarrow t + 1$
  - $t' \leftarrow t' + 1$
- **until** *episod ends at time* T' *or* $t' = T$

until $t' = T$
output: State-Action value function Q

Expected SARSA

Expected SARSA

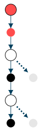
- The policy Π is known so that we can use it in a formula:
$$R_t + \gamma Q(S_t, A_t) \longrightarrow R_t + \gamma \sum_a \pi(a|S_t) Q(S_t, a)$$
- Make the update independent of the action chosen (and thus of the policy used to play).
- Reduce the variance for a computational cost.
- Amount to use the current estimate for $V(S_t)$...

Expected SARSA

Discounted: Prediction by Expected SARSA

input: MDP environment, initial state distribution μ_0, policy Π and discount factor γ
parameter: Number of step T
init: $\forall s, a, Q(s,a), N(s,a) = 0$, n=0, $t' = 0$
repeat
  $t \leftarrow 0$
  Pick initial state S_0 following μ_0
  repeat
    $N(S_t) \leftarrow N(S_t) + 1$
    Pick action A_t according to $\pi(\cdot|S_t)$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(N(S_t, A_t)) \left(R_{t+1} + \gamma \sum_a \pi(a|S_t) Q(S_{t+1}, a) - Q(S_t, A_t)\right)$
    $t \leftarrow t + 1$
    $t' \leftarrow t' + 1$
  until *episod ends at time* T' *or* $t' = T$
until $t = T$
output: State-Action value function Q

n-steps Tree Backup



n-steps Tree Backup

- At each time step, use the expected SARSA average over the action while replacing the Q value for the picked action by a deeper estimate.
  - 1-step return (Expected Sarsa)
  $$G_{t:t+1} = R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a)$$
  - 2-step return:
  $$\begin{aligned} G_{t:t+2} = R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q_{t+1}(S_{t+1}, a) \\ + \gamma\pi(A_{t+1}|S_{t+1}) \left(R_{t+2} + \gamma \sum_a \pi(a|S_{t+2}) Q(S_{t+2}, a) \right) \\ = R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q(S_{t+1}, a) + \gamma\pi(A_{t+1}|S_{t+1}) G_{t+1:t+2} \end{aligned}$$

n-steps Tree Backup

- 1-step return (Expected Sarsa)

$$G_{t:t+1} = R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a)$$

- 2-step return:

$$\begin{aligned} G_{t:t+2} &= R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q(S_{t+1}, a) + \gamma \pi(A_{t+1}|S_{t+1}) G_{t+1:t+2} \\ &= R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) + \gamma \pi(A_{t+1}|S_{t+1}) \left(G_{t+1:t+2} - Q(S_{t+1}, A_{t+1})\right) \end{aligned}$$

- Recursive definition of n-step return:

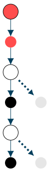
$$\begin{aligned} G_{t:t+n} = R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) \\ + \gamma \pi(A_{t+1}|S_{t+1}) \left(G_{t+1:t+n} - Q(S_{t+1}, A_{t+1})\right) \end{aligned}$$

- TD update

$$Q(S_{t-n}, A_{t-n}) = Q(S_{t-n}, A_{t-n}) + \alpha(N(S_{t-n}, Q_{t-n})) \left(G_{t-n:t} - Q(S_{t-n}, A_{t-n})\right)$$

Discounted

$Q(\sigma)$

Between  and 

Sampling or Averaging

- Unifying algorithm!
- Recursive definition of n-step return:

$$
\begin{aligned}
G_{t:t+n} = R_{t+1} &+ \sigma G_{t+1:t+n} \\
&+ (1-\sigma)\Big(\gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) \\
&\qquad + \gamma\pi(A_{t+1}|S_{t+1}) \left(G_{t+1:t+n} - Q(S_{t+1}, A_{t+1})\right)\Big)
\end{aligned}
$$

λ-Return

Averaged n-steps return?

- n-step return:

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

- Averaged n-step return: (compound update)

$$G_t^\omega = \sum_{n=1}^{\infty} \omega_n G_{t:t+n} \quad \text{with} \sum_{i=1}^{\infty} \omega_n = 1$$

- TD(λ): specific averaging

$$\begin{aligned} G_t^\lambda &= (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_{t:t+n} \\ &= (1-\lambda) \sum_{n=1}^{T-t} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t} G_t \quad \text{(Episodic)} \end{aligned}$$

  interpolating between TD (a.k.a TD(0)) and MC for $\lambda = 1$.
- Can be mixed with tree backup strategies (TB(λ))

λ-return and Temporality

True λ-return

- Require to wait until the end of an episode before we can update.
- Unusable in a non episodic setting!

Truncated λ-return

- Truncated λ-return:

$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{H-t} \lambda^{n-1} G_{t:t+n} + \lambda^{H-t} G_{t:H}$$

- The virtual horizon H may vary during the algorithm.

λ-return and Temporality

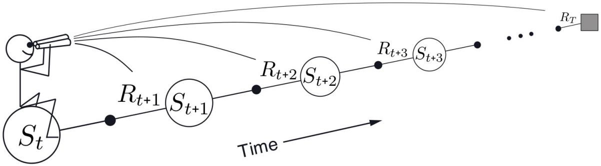
Temporality

- n-step return
$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$
depends on a current estimate V (or Q)!
- In G_λ should we use
  - an estimate available at time t?
  - an estimate available at time $t+n$?
  - an estimate available at time H?
- Off-Line vs On-Line!
  - Off-line: keep V constant during the episodes.
  - On-line: Used updated V when available.
  - True on-line (Sutton and Barto): restart algorithm with a growing horizon.

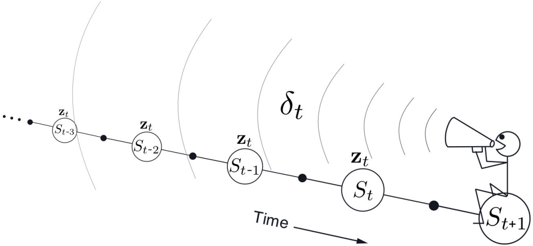
Outline

Forward and Backward Point of View

From a forward view



To a backward one:



Source: Sutton and Barto

Returns and Temporal Differencies

Returns and Temporal Differencies

- n-step returns:

$$
\begin{aligned}
G_{t:t+n} - Q(S_t, A_t) &= R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} \\
&\quad + \gamma^n Q(S_{t+n}, A_{t+n}) - Q(S_t, A_t) \\
&= \sum_{l=1}^{n} \gamma^{l-1} (R_{t+l} + \gamma Q(S_{t+l}, A_{t+l}) - Q(S_{t+l-1}, A_{t+l-1})) \\
&= \sum_{l=0}^{n-1} \gamma^{l-1} \delta_{t+l}
\end{aligned}
$$

- λ return:

$$
\begin{aligned}
G_t^\lambda - Q(S_t, A_t) &= (1-\lambda) \sum_n \lambda^n (G_{t:t+n} - Q(S_t, A_t)) \\
&= \sum_{n=0} \lambda^n \gamma^n \delta_{t+n}
\end{aligned}
$$

Forward View and Backward View

Forward View

- Updates:

$$Q_t(s,a) = Q_{t-1}(s,a) + \mathbf{1}_{(s,a)=(S_t,A_t)} \alpha_t(s,a) \left(\sum_{t'' \geq t} \lambda^{t''-t} \gamma^{t''-t} \delta_{t''} \right)$$

- Cumulative updates:

$$Q_t(s,a) = Q_0(s,a) + \sum_{t' \leq t} \mathbf{1}_{(s,a)=(S_{t'},A_{t'})} \alpha_{t'}(s,a) \left(\sum_{t'' \geq t'} \lambda^{t''-t'} \gamma^{t''-t'} \delta_{t''} \right)$$

- Limit:

$$Q_\infty(s,a) = Q_0(s,a) + \sum_{t'} \mathbf{1}_{(s,a)=(S_{t'},A_{t'})} \alpha_{t'}(s,a) \left(\sum_{t'' \geq t'} \lambda^{t''-t'} \gamma^{t''-t'} \delta_{t''} \right)$$

- Focus on the update place.

Forward View and Backward View

Limit(s)

- Limit:

$$
\begin{aligned}
Q_\infty(s,a) &= Q_0(s,a) + \sum_{t'} \mathbf{1}_{(s,a)=(S_{t'},A_{t'})} \alpha_{t'}(s,a) \left(\sum_{t'' \geq t'} \lambda^{t''-t'} \gamma^{t''-t'} \delta_{t''} \right) \\
&= Q_0(s,a) + \sum_{t''} \delta_{t''} \sum_{t' \leq t''} \mathbf{1}_{(s,a)=(S_{t'},A_{t'})} \alpha_{t'}(s,a) \lambda^{t''-t'} \gamma^{t''-t'}
\end{aligned}
$$

- Focus on the update place or and the temporal differencies. . .

Forward View and Backward View

Backward View

- Same limit with cumulative udpates over temporal differencies
$$Q_t(s,a) = Q_0(s,a) + \sum_{t''\leq t} \delta_{t''} \sum_{t'\leq t''} \mathbf{1}_{(s,a)=(S_{t'},A_{t'})}\alpha_{t'}(s,a)\lambda^{t''-t'}\gamma^{t''-t'}$$
- Updates
$$Q_t(s,a) = Q_{t-1}(s,a) + \delta_t \underbrace{\sum_{t'\leq t} \mathbf{1}_{(s,a)=(S_{t'},A_{t'})}\alpha_{t'}(s,a)\lambda^{t-t'}\gamma^{t-t'}}_{z_t(s,a)}$$
- Pseudo Eligibility trace:
$$\begin{aligned} z_t(s,a) &= \sum_{t'\leq t} \mathbf{1}_{(s,a)=(S_{t'},A_{t'})}\alpha_{t'}(s,a)\lambda^{t-t'}\gamma^{t-t'} \\ &= \lambda\gamma z_{t-1}(s,a) + \alpha_t(s,a)\mathbf{1}_{(s,a)=(S_t,A_t)} \end{aligned}$$

- Proof of convergence toward the same target.

Eligibility Trace

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha_t \delta_t z_t(s,a)$$

Eligibility Trace

- Focus on temporal differencies with simultaneous update on all states.
- TD(λ) eligibility trace: $z_t(s,a) = \lambda\gamma z_{t-1}(s,a) + \mathbf{1}_{(s,a)=(S_t,A_t)}$
- Strictly equivalent to the previous scheme for constant stepsize
- Other eligibility trace:
  - Replacing trace:
  $$z_t(s,a) = \begin{cases} 1 & \text{if } (s,a) = (S_t,A_t) \\ \lambda\gamma z_{t-1}(s,a) & \text{otherwise} \end{cases}$$
  - Time dependent trace:
  $$z_t(s,a) = c_t\gamma z_{t-1}(s,a) + \mathbf{1}_{(s,a)=(S_t,A_t)}$$
  where c_t is defined *in a appropriate way* to ensure the convergence of the algorithm.

- Need to store (and update) this information...

Temporal Differencies

$$\delta_t?$$

Temporal Differencies

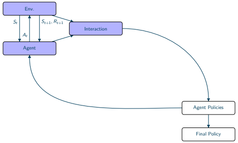
- Basic temporal differencies:
$$\delta_t = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$$
- Expected temporal differencies:
$$\begin{aligned}\delta_t &= R_{t+1} + \gamma V(S_{t+1}) - Q(S_t, A_t)\\ &= R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)\end{aligned}$$
- Average of both:
$$\begin{aligned}\delta_t &= R_{t+1} + \gamma\sigma Q(S_{t+1}, A_{t+1}) + \gamma(1-\sigma) V(S_{t+1}) - Q(S_t, A_t)\\ &= R_{t+1} + \gamma V(S_{t+1}) + \gamma\sigma \left(Q(S_{t+1}, A_{t+1}) - V(S_{t+1})\right) - Q(S_t, A_t)\end{aligned}$$

- Only expected temporal average is independent of the next action.
- No generic proof of convergence...

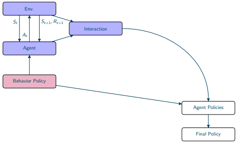
Outline

On-Policy vs Off-Policy

From



to



On-Policy vs Off-Policy

- On-Policy: the policy b used to interact is the same than the policy Π evaluated or optimized.
- Off-Policy: the policy b used to interact may be different from the policy Π evaluated or optimized.

- Off-Policy allows in particular to (re)use interactions from previous experiments.
- Q-learning was possible in off-policy setting.

Importance Sampling

$$\rho_{t:t'} = \frac{\mathbb{P}_\Pi(S_t, A_t, R_{t+1}, S_{t+1}, \dots, R_{t'}, S_{t'}, A_{t'} | S_t)}{\mathbb{P}_b(S_t, A_t, R_{t+1}, S_{t+1}, \dots, R_{t'}, S_{t'}, A_{t'} | S_t)} = \frac{\pi(A_t|S_t) \dots \pi(A_{t'}|S_{t'})}{b(A_t|S_t) \dots b(A_{t'}|S_{t'})}$$

Importance Sampling

- For any law p and q, and any function g
$$\mathbb{E}_p[g(x)] = \mathbb{E}_q\left[\frac{p(x)}{q(x)} g(x)\right]$$
provided $q(x) = 0$ implies $p(x) = 0$.
- $\mathbb{V}\mathrm{ar}_q\left[\frac{p(x)}{q(x)} g(x)\right]$ may be large with respect to $\mathbb{V}\mathrm{ar}_p\left[g(x)\right]$ if the ratio $p(x)/q(x)$ is large. . .

Importance Sampling for Trajectories

- For any trajectory $\tau_{t:t'} = S_t, A_t, R_{t+1}, S_{t+1}, \dots, R_{t'}, S_{t'}, A_{t'}(, R_{t'+1}, S_{t'+1})$,,
$$\frac{\mathbb{P}_\Pi(S_t, A_t, R_{t+1}, S_{t+1}, \dots, R_{t'}, S_{t'}, A_{t'}(, R_{t'+1}, S_{t'+1}) | S_t)}{\mathbb{P}_b(S_t, A_t, R_{t+1}, S_{t+1}, \dots, R_{t'}, S_{t'}, A_{t'}(, R_{t'+1}, S_{t'+1}) | S_t)} = \frac{\pi(A_t|S_t) \dots \pi(A_{t'}|S_{t'})}{b(A_t|S_t) \dots b(A_{t'}|S_{t'})}$$

Importance Sampling and Returns

$$\mathbb{E}_\Pi[g(\tau_{t:t'})|S_t = s] = \mathbb{E}_b[\rho_{t:t'} g(\tau_{t:t'})|S_t = s] \quad \text{with} \quad \rho_{t:t'} = \frac{\pi(A_t|S_t)\dots\pi(A_{t'}|S_{t'})}{b(A_t|S_t)\dots b(A_{t'}|S_{t'})}$$

From b to Π

- Returns:

$$\begin{aligned}
\mathbb{E}_\pi[G_{t:t'}|S_t = s] &= \mathbb{E}_\pi\left[\sum_{t''=t+1}^{t'} \gamma^{t''-t-1} R_{t''} + \gamma^{t'-t} V(S_{t'}) \middle| S_t = s\right] \\
&= \mathbb{E}_b\left[\rho_{t:(t'-1)} \left(\sum_{t''=t+1}^{t'} \gamma^{t''-t-1} R_{t''} + \gamma^{t'-t} V(S_{t'})\right) \middle| S_t = s\right] \\
&= \mathbb{E}_b\left[\sum_{t''=t+1}^{t'} \rho_{t:(t''-1)} \gamma^{t''-t-1} R_{t''} + \rho_{t:(t'-1)} \gamma^{t'-t} V(S_{t'}) \middle| S_t = s\right]
\end{aligned}$$

Importance Sampling and Returns

$$\mathbb{E}_\Pi[g(\tau_{t:t'})|S_t, A_t] = \mathbb{E}_b\left[\rho_{(t+1):t'} g(\tau_{t:t'})\middle|S_t, A_t\right] \quad \text{with} \quad \rho_{t:t'} = \frac{\pi(A_t|S_t)\dots\pi(A_{t'}|S_{t'})}{b(A_t|S_t)\dots b(A_{t'}|S_{t'})}$$

From b to Π

- Returns:

$$\begin{aligned}
\mathbb{E}_\pi[G_{t:t'}|S_t, A_t] &= \mathbb{E}_\pi\left[\sum_{t''=t+1}^{t'} \gamma^{t''-t-1} R_{t''} + \gamma^{t'-t} Q(S_{t'}, A_{t'})\middle|S_t, A_t\right]\\
&= \mathbb{E}_b\left[\rho_{(t+1):(t'-1)}\left(\sum_{t''=t+1}^{t'} \gamma^{t''-t-1} R_{t''} + \gamma^{t'-t} Q(S_{t'}, A_{t'})\right)\middle|S_t, A_t\right]\\
&= \mathbb{E}_b\left[\sum_{t''=t+1}^{t'} \gamma^{t''-t-1} \rho_{(t+1):(t''-1)} R_{t''} + \rho_{(t+1):t'} \gamma^{t'-t} Q(S_{t'}, A_{t'})\middle|S_t, A_t\right]
\end{aligned}$$

- No correction if $t' = t + 1$

λ-return

- Recursive definition of the λ-return:

$$G_t^\lambda | S_t = R_{t+1} + \gamma \left((1-\lambda) V(S_{t+1}) + \lambda G_{t+1}^\lambda \right)$$

$$G_t^\lambda | S_t, A_t = R_{t+1} + \gamma \Big((1-\lambda) (\sigma Q(S_{t+1}, A_{t+1}) + (1-\sigma) (\sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) + \pi(A_{t+1}|S_{t+1}) \left(G_{t+1}^\lambda - Q(S_{t+1}, A_{t+1}) \right))) + \lambda G_{t+1}^\lambda \Big)$$

- Off-line correction

$$G_t^\lambda | S_t = \rho_{t:t} \left(R_{t+1} + \gamma \left((1-\lambda) V(S_{t+1}) + \lambda G_{t+1}^\lambda \right) \right)$$

$$G_t^\lambda | S_t, A_t = R_{t+1} + \gamma \Big((1-\lambda) (\sigma Q(S_{t+1}, A'_{t+1}) + (1-\sigma) (\sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) + \pi(A_{t+1}|S_{t+1}) \left(G_{t+1}^\lambda - Q(S_{t+1}, A_{t+1}) \right))) + \lambda \rho_{t+1:t+1} G_{t+1}^\lambda \Big)$$

where A'_{t+1} is drawn following π (or multiply by $\rho_{t+1:t+1}$ to use A_{t+1}).

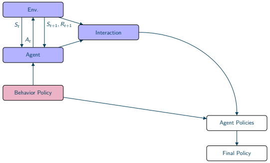
Temporal Differencies

$$\delta_t?$$

Temporal Differencies

- Basic temporal differencies:
$$\delta_t = R_{t+1} + \gamma Q(S_{t+1}, A'_{t+1}) - Q(S_t, A_t)$$
with A'_{t+1} drawn using π.
- Expected temporal differencies:
$$\begin{aligned}\delta_t &= R_{t+1} + \gamma V(S_{t+1}) - Q(S_t, A_t)\\ &= R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)\end{aligned}$$
without any correction.
- Average of both:
$$\begin{aligned}\delta_t &= R_{t+1} + \gamma\sigma Q(S_{t+1}, A_{t+1}) + \gamma(1-\sigma) V(S_{t+1}) - Q(S_t, A_t)\\ &= R_{t+1} + \gamma V(S_{t+1}) + \gamma\sigma \left(Q(S_{t+1}, A'_{t+1}) - V(S_{t+1})\right) - Q(S_t, A_t)\end{aligned}$$
with A'_{t+1} drawn using π.

Off-Policy Algorithm



Off-Policy Correction

- Replace any estimate of the gain by an importance-sampling corrected one.
- Works well for prediction.
- Can be combined with policy improvement (a la SARSA) but less (no?) theoretical guarantees.

Retrace(λ)

$$\widetilde{\mathcal{T}}Q(s,a) = Q(s,a) + \mathbb{E}_b\left[\sum_{t\geq 0}\gamma^t\left(\prod_{t'=1}^{t} c_{t'}\right)\delta_t \middle| S_0 = s, A_0 = a\right]$$

$$c_t = c(A_t, S_t, A_{t-1}, S_{t-1}, \cdots, A_0, S_0)$$

$$\mathbb{E}_b[\delta_t | S_t, A_t] = \mathbb{E}[R_{t+1} + \gamma \mathbb{E}_\pi[Q(S_{t+1}, \cdot)] - Q(S_t, A_t) | S_t, A_t]$$

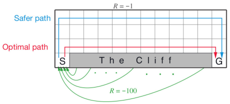
Generic Off-Policy Algorithm

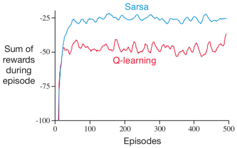
- Generic off-line algorithm including
  - Importance sampling: $c_t = \rho_{t:t} = \pi(A_t|S_t)/b(A_t|S_t)$
  - TB(λ): $c_t = \lambda\pi(A_t|S_t)$
  - Retrace(λ): $c_t = \lambda \min(1, \pi(A_t|S_t)/b(A_t/S_t))$
- **Prop:** Q_π is a fixed point as $\mathbb{E}_b[\delta_t|S_t, A_t] = \mathbb{E}[\mathcal{T}^\pi Q(S_t, A_t) - Q(S_t, A_t)|S_t, A_t]$.
- **Prop:** $\widetilde{\mathcal{T}}$ is a contraction provided $c_t \leq \rho_t = \pi(A_t|S_t)/b(A_t|S_t)$.
- Convergence for Importance sampling, TB(λ) and Retrace(λ) for any b.
- Partial results for policy improvement under more assumptions.

- For Q(λ), $c_t = \lambda$, convergence if $\|\pi(|s) - b(|s)\|_1 \leq \epsilon$ and $\lambda \leq (1-\gamma)/(\gamma\epsilon)$.

Outline

Q-Learning vs SARSA





How different are they?

- In Q-learning, the exploratory policy used is decoupled from the optimized policy.
- This exploratory policy may yield low rewards on average.
- In SARSA, the two policies are linked with the hope on having higher rewards during the learning step.
- Subtle different behavior even if we modify the exploratory policy in Q-Learning.

Exploration vs Exploitation

Exploration vs Exploitation

- Exploration: explore new policies to be able to discover the best ones.
- Exploitation: use good policies to obtain a good return.
- Exploration is a requirement.

- No tradeoff if we optimize only the final result!
- Tradeoff between the two if we consider that the returns during training matters!
- Q-learning use the first approach and SARSA try to tackle the second.
- Tradeoff if we study a regret:
$$\sum_t \mathbb{E}_{\Pi_\star}[R_t] - \mathbb{E}_{\Pi_t}[R_t]$$
which forces us to be good as fast as possible.
- No natural definition in the discounted setting.

Bandits

$$\mathcal{S} = \{0\} \quad \text{and} \quad A = \{1, \ldots k\} \quad \text{and} \quad r(s,a) = r_a$$

Bandits

- Very simple toy model where there is only one state!
- Optimal policy: pick $a_\star \in \operatorname{argmax} r_a$.
- Q estimation: estimate r_a by playing action a.
- Strategy:
  - Every arm has to be played until we are sure they are bad.
  - Best arm should be played as often as possible to maximime the rewards during the learnig phase.
- Simple enough setting to obtain result on the regret

$$r_T = \sum_{t \leq T} (r_{a_\star} - R_t)$$

- We will use $\Delta_a = r_{a_\star} - r_a$ and assume that $R|a$ is 1-subgaussian.

Explore Then Commit

Explore Then Commit (Random Exploration)

- Play the arm successively during Km steps and then play the optimal one during $T - Km$ steps.
- **Prop:**

$$r_T \leq \min(m, T/K) \sum_{a=1}^{k} \Delta(a) + \max(T - mK, 0) \sum_{a=1}^{k} \Delta(a) \exp(-m\Delta(a)^2/4)$$

  Furthermore,

$$\mathbb{P}(a_T = a_*) \geq 1 - \sum_{a \neq a_*} \exp(-m\Delta(a)^2/4)$$

- $R_T \leq O(\log T)$ for $m \propto \log T$,
- but $R_T = O(T)$ for any fixed m.

ϵ-greedy Strategy

ϵ-greedy Strategy

- Estimate $Q(a) = r_a$ by MC:

$$Q_t(a) = \frac{\sum_{t'=1}^{t-1} \mathbf{1}_{A_{t'}=a} R_{t'}}{\sum_{i=1}^{t-1} \mathbf{1}_{A_{t'}=a}}$$

- Pick arm a at time t using

$$\pi(a) = \begin{cases} \epsilon_t/k + (1-\epsilon) & \text{if } a = \operatorname{argmax}_{a'} Q_t(a') \text{ (only the smallest if necessary)} \\ \epsilon_t/k & \text{otherwise} \end{cases}$$

- **Prop:**

$$r_T \geq \sum_{t=1}^{T} \frac{\epsilon_t}{k} \sum_{a=1}^{k} \Delta(a)$$

ϵ-greedy Strategy

ϵ-greedy Strategy

- **Prop:**

$$\mathbb{P}(A_T = a_*) \geq 1 - \epsilon_T - \Sigma_t \exp(-\Sigma_T/(6k)) - \sum_{a \neq a_*} \frac{4}{\Delta(a)^2} e^{-\Delta(a)^2 \Sigma_T/(4k)}$$

with $\Sigma_T = \sum_{s=1}^{T} \epsilon_s$.

Furthermore,

$$\mathbb{P}(a_* = \operatorname{argmax} Q_{T,a}) \geq 1 - \Sigma_t \exp(-\Sigma_T/(6k)) - \sum_{a \neq a_*} \frac{4}{\Delta(a)^2} e^{-\Delta(a)^2 \Sigma_T/(4k)}$$

If $\epsilon_t = c/t$,

$$r_T \leq \sum_{a \neq a_*} \left(\Delta(a) \left(c \frac{\log(T) + 1}{k} + C \right) + \frac{4}{\Delta(a)} C' \right)$$

as soon as $c/(6k) > 1$ and $c \min_{a \neq a_*} \Delta(a)/4k < 1$.

If $\epsilon_t = c \log(t)/t$ then

$$r_T \leq \sum_{a \neq a_*} \left(\Delta(a) \left(c \frac{\log(T)(\log(T) + 1)}{k} + C \right) + \frac{4}{\Delta(a)} C' \right)$$

UCB Strategy

Upper Confidence Bound

- Use an optimistic strategy to pick the best arm

$$A_t = \operatorname{argmax} Q_t(a) + \sqrt{\frac{c \log t}{N_t(a)}}$$

- **Prop:**

$$r_n(t) \leq C_c \sum_a \Delta(a) + \sum_a \frac{4c \ln t}{\Delta(a)}.$$

with $C_c < +\infty$ as soon as $c > 3/2$
Furthermore

$$\mathbb{P}(A_t = a_*) \geq 1 - 2kt^{-2c+2}$$

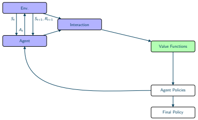
as soon as $t \geq \max_a \frac{4c \ln t}{\Delta(a)^2}$.

- Optimal regret!
- Hard to extend to RL setting but shows that ϵ-greedy may not be optimal.

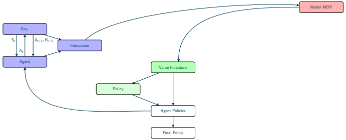
Outline

Model Based Approach

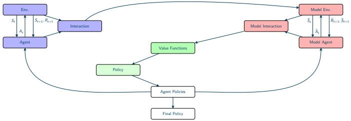
From



to



or

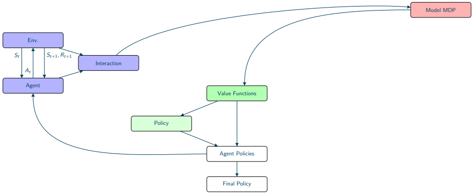


Model Based Approach

- Use the interactions to learn a model. . .
- that can be used to learn a good policy.
- This model can be:
  - a MDP,
  - a simulator.

- Often easier to obtain a simulator.

Model based and MDP

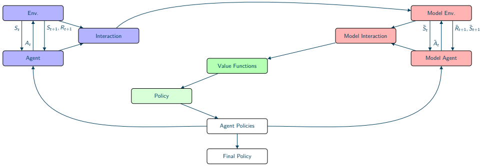


Estimated MDP: back to OR

- MDP can be estimated from trajectories.
- Simple (but maybe slow) even in an off-line setting.
- Once we have an estimated MDP, prediction and planning can be done using OR.

- Implicitely done by TD(0) when doing several passes.
- Model should be checked/improved as much as possible when new trajectories arrive.

Model based and RL

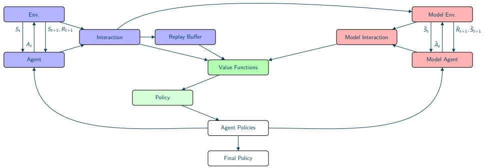


Estimated Simulator: back to RL

- Simulator can be estimated from trajectories.
- Simple (but maybe slow) even in an off-line setting.
- Once we have an estimated simulator, prediction and planning can be done using RL.

- Model should be checked/improved as much as possible when new trajectories arrive.

Model Free and Model Based Approach



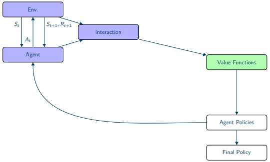
Dyna

- Combine true interactions with simulated ones.
- Simultaneous acting, model learning, OR learning and RL learning.
- Search for a tradeoff between the (slow) learning RL algorithm and the (wrong) model OR algorithm.
- Need to deal with schedule!

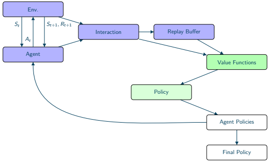
Outline

Replay Buffer and Prioritized Sweeping

From



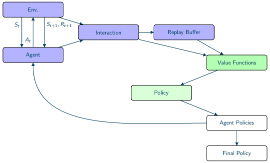
to



Replay Buffer and Prioritized Sweeping

- Can we reuse previous interactions?
- In which order?

Replay Buffer

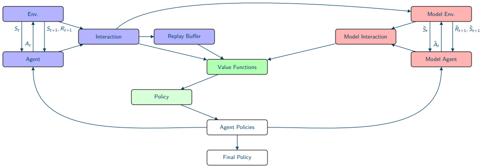


Replay Buffer

- Store previous interactions (trajectories) in a first-in first-out buffer.
- Draw a subsequence from those interactions (trajectories) and use it in a RL algorithm:
  - On-line: if the trajectory comes from the same policy.
  - Off-line: if the trajectory comes from a different policy.

- Similar to a simulator but no arbitrary choice of state or action.
- Often use with on-line algorithm if the policy has only mildy evolved...

Prioritized Sweeping



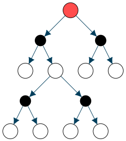
Prioritized Sweeping

- Plain Replay Buffer: subsequence drawn uniformly.
- Prioritized Sweeping: subsequence drawn favoring states with large temporal differencies.

- Can be combined with a model approach.

Outline

Real-Time Planning

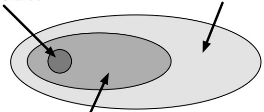


Real-Time Planning

- Can we optimize the policy at the current state?
- Do we need to optimize it everywhere?
- What is required?

- Planning at decision time. . .

Real-Time Dynamic Programming



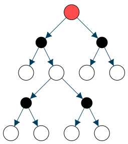
- Warmup in Dynamic Programming...

RT DP

- Use trajectories to sample the states to update.
- Convergence holds with exploratory policy.
- Optimal policy does not require to specify the action in irrelevant states.
- Convergence holds even without full exploration in some specific cases!

- In practice, seems to be computationaly efficient.

Planning At Decision Time



Planning At Decision Time

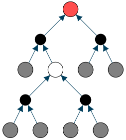
- Can we find a good action A_t at S_t... without having it precomputed?
- Policy Improvement

$$A_t = \text{argmax}\, Q_t(S_t, \cdot)$$

  can be seen as a first step.
- How to go deeper?

- **A model or a simulator will be required!**

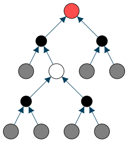
Heuristic Search



Heuristic Search

- Requires the knowledge of the MDP and of a heuristic based value function V.
- Strategy:
  - Build a limited depth tree by stopping after a few steps and at some specific states.
  - Backup the heuristic based value function using Dynamic Programming (Optimal Bellman operator).
  - Pick the action having the hight value.
- The deeper the better... but the more expensive due to branching!
- Requires a *suitable* heuristic...

Rollout Algorithm

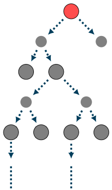


Rollout Policy

- Use a MC estimate with a default policy instead of a heuristic.
- Backup those estimates using Dynamic Programming.
- Simulation can even start after the first action (as in Policy Improvement).

- The values are (most of the time) discarded for the next state.

Monte Carlo Tree Search



Monte Carlo Tree Search

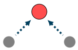
- Simultaneour tree growing, rollout and backup by DP.
- Repeat 4 steps:
  - Selection of a sequence of actions according to the current values with a tree policy.
  - Expansion of the tree at the last node without values.
  - Simulation with a rollout policy to estimate the values at this node.
  - Backup of the value by relaxed Dynamic Programming.
- MCTS focuses on promising paths using a UCB approach.

Monte Carlo Tree Search

Monte Carlo Tree Search

- Simultaneour tree growing, rollout and backup by DP.
- Repeat 4 steps:
  - Selection of a sequence of actions according to the current values with a tree policy.
  - Expansion of the tree at the last node without values.
  - Simulation with a rollout policy to estimate the values at this node.
  - Backup of the value by relaxed Dynamic Programming.
- MCTS focuses on promising paths using a UCB approach.

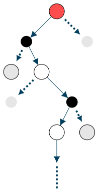
Monte Carlo Tree Search



Monte Carlo Tree Search

- Simultaneour tree growing, rollout and backup by DP.
- Repeat 4 steps:
  - Selection of a sequence of actions according to the current values with a tree policy.
  - Expansion of the tree at the last node without values.
  - Simulation with a rollout policy to estimate the values at this node.
  - Backup of the value by relaxed Dynamic Programming.
- MCTS focuses on promising paths using a UCB approach.

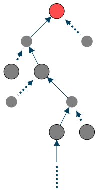
Monte Carlo Tree Search



Monte Carlo Tree Search

- Simultaneour tree growing, rollout and backup by DP.
- Repeat 4 steps:
  - Selection of a sequence of actions according to the current values with a tree policy.
  - Expansion of the tree at the last node without values.
  - Simulation with a rollout policy to estimate the values at this node.
  - Backup of the value by relaxed Dynamic Programming.
- MCTS focuses on promising paths using a UCB approach.

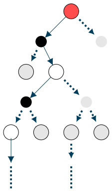
Monte Carlo Tree Search



Monte Carlo Tree Search

- Simultaneour tree growing, rollout and backup by DP.
- Repeat 4 steps:
  - Selection of a sequence of actions according to the current values with a tree policy.
  - Expansion of the tree at the last node without values.
  - Simulation with a rollout policy to estimate the values at this node.
  - Backup of the value by relaxed Dynamic Programming.
- MCTS focuses on promising paths using a UCB approach.

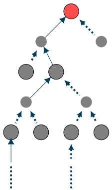
Monte Carlo Tree Search



Monte Carlo Tree Search

- Simultaneour tree growing, rollout and backup by DP.
- Repeat 4 steps:
  - Selection of a sequence of actions according to the current values with a tree policy.
  - Expansion of the tree at the last node without values.
  - Simulation with a rollout policy to estimate the values at this node.
  - Backup of the value by relaxed Dynamic Programming.
- MCTS focuses on promising paths using a UCB approach.

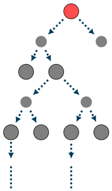
Monte Carlo Tree Search



Monte Carlo Tree Search

- Simultaneour tree growing, rollout and backup by DP.
- Repeat 4 steps:
  - Selection of a sequence of actions according to the current values with a tree policy.
  - Expansion of the tree at the last node without values.
  - Simulation with a rollout policy to estimate the values at this node.
  - Backup of the value by relaxed Dynamic Programming.
- MCTS focuses on promising paths using a UCB approach.

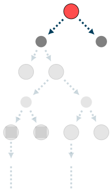
Monte Carlo Tree Search



Monte Carlo Tree Search

- Simultaneour tree growing, rollout and backup by DP.
- Repeat 4 steps:
  - Selection of a sequence of actions according to the current values with a tree policy.
  - Expansion of the tree at the last node without values.
  - Simulation with a rollout policy to estimate the values at this node.
  - Backup of the value by relaxed Dynamic Programming.
- MCTS focuses on promising paths using a UCB approach.

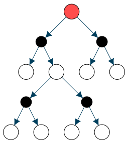
Monte Carlo Tree Search



Monte Carlo Tree Search

- Simultaneour tree growing, rollout and backup by DP.
- Repeat 4 steps:
  - Selection of a sequence of actions according to the current values with a tree policy.
  - Expansion of the tree at the last node without values.
  - Simulation with a rollout policy to estimate the values at this node.
  - Backup of the value by relaxed Dynamic Programming.
- MCTS focuses on promising paths using a UCB approach.

Model Predictive Control?



Model Predictive Control

- Open loop optimization:

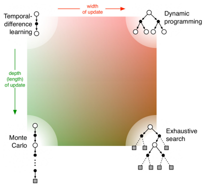
$$\max_{a_t, a_{t+1}, \ldots, a_{t+h}} \mathbb{E}\left[\sum_{t'=t}^{t+h} R_t\right]$$

  using a predictive model (simulator).
- Do not take into account state uncertainties in the control choice...
- But much simpler optimization...
- and equivalence for a linear Gaussian model.

- Extensively used for short-term planning in Control.
- May be combined with value functions after $t + h$.

Outline

Approximation?



Tabular Setting

- Require to store the state(-action) values (a table).
- Requirement in both OR and RL.

Approximation!

- Use instead approximated value functions.
- What is a good approximation?
- How to use them?

- Focus on value-functions. . .

Outline

Approximated Value Functions

$$V(s) \Longrightarrow V_{\boldsymbol{w}}(s)$$
$$Q(s,a) \Longrightarrow Q_{\boldsymbol{w}}(s,a)$$

Parametric Model

- Reduce dimensionality by storing $\boldsymbol{w}$ instead of all the values.
- Linear: $V_{\boldsymbol{w}}(s) = \langle \Phi(s), \boldsymbol{w} \rangle$ and $Q_{\boldsymbol{w}}(s,a) = \langle \Phi(s,a), \boldsymbol{w} \rangle$
  - $\Phi(s)$ and $\Phi(s,a)$ are features associated to the states(-actions).
  - Tabular setting corresponds to $(\Phi)_{s'(,a')}(s(,a)) = \mathbf{1}_{s'=s(,a'=a)}$.
  - Often used in theoretical analysis.
- Deep Learning: $V_{\boldsymbol{w}}(s) = \mathrm{NN}_{\boldsymbol{w}}(\Phi(s))$ and $Q_{\boldsymbol{w}}(s,a) = \mathrm{NN}_{\boldsymbol{w}}(\Phi(s,a))$
  - NN is any (deep) learning network.
  - Often used in practice.

- Other parametrization (or even non parametric coding) could be used (at least in theory...).

Approximated Value Functions Usage

$$\begin{aligned} v_\pi(s) &\simeq V_{\mathbf{w}_\pi}(s) \\ q_\pi(s,a) &\simeq Q_{\mathbf{w}_\pi}(s,a) \\ \operatorname*{argmax}_a q_\pi(s,a) &\simeq \operatorname*{argmax}_a Q_{\mathbf{w}_\pi}(s,a) \end{aligned}$$

$$\begin{aligned} v_\star(s) &\simeq V_{\mathbf{w}_\star}(s) \\ q_\star(s,a) &\simeq Q_{\mathbf{w}_\star}(s,a) \\ \operatorname*{argmax}_a q_\star(s,a) &\simeq \operatorname*{argmax}_a Q_{\mathbf{w}_\star}(s,a) \end{aligned}$$

Approximated Value Functions Usage

- *Drop-in* replacements for all the value functions?
- Prediction and Planning?
- Quality and Stability?

Approximation Quality

$$v_\pi(s) \simeq V_{\boldsymbol{w}_\pi}(s) \qquad v_\star(s) \simeq V_{\boldsymbol{w}_\star}(s)$$
$$q_\pi(s,a) \simeq Q_{\boldsymbol{w}_\pi}(s,a) \qquad q_\star(s,a) \simeq Q_{\boldsymbol{w}_\star}(s,a)$$
$$\operatorname*{argmax}_a q_\pi(s,a) \simeq \operatorname*{argmax}_a Q_{\boldsymbol{w}_\pi}(s,a) \qquad \operatorname*{argmax}_a q_\star(s,a) \simeq \operatorname*{argmax}_a Q_{\boldsymbol{w}_\star}(s,a)$$

Approximation Quality Norm

- Ideal loss:

$$\|v - V_{\boldsymbol{w}}\|_\infty \quad \text{or} \quad \|q - Q_{\boldsymbol{w}}\|_\infty$$

as this is the error used in all the previous analysis.

- Practical loss:

$$\|v - V_{\boldsymbol{w}}\|_{\mu,p}^p = \sum_s \mu(s)|v(s) - V_{\boldsymbol{w}}(s)|^p$$
$$\text{or} \quad \|q - Q_{\boldsymbol{w}}\|_{\mu,p}^p = \sum_{s,a} \mu(s,a)|q(s,a) - Q_{\boldsymbol{w}}(s,a)|^p$$

often with $p = 2$ and μ related to the behavior policy.

Approximation Target(s)

$$q(s,a) = \mathcal{T} q(s,a) \sim Q_{\mathbf{w}}(s,a) \longrightarrow \begin{cases} \|q - Q_{\mathbf{w}}\|_{\mu,p} \text{ small} \\ \|\mathcal{T} Q_{\mathbf{w}} - Q_{\mathbf{w}}\|_{\mu,p} \text{ small} \end{cases}$$

Approximation Targets(s)

- Direct measurement.
- Bellman residual error.

Extended Measurement

- Projection (with linear parametrization): $\|P_\Phi \left(\mathcal{T} Q_{\mathbf{w}} - Q_{\mathbf{w}}\right)\|_{\mu,p}$ small
- Probes Z:

$$\mathbb{E}_Z[|\left\langle \mathcal{T} Q_{\mathbf{w}} - Q_{\mathbf{w}}, Z \right\rangle|^p]$$

- Lots of freedom but hard to link with optimality of derived policy!

Outline

Prediction, Approximation and Gradient Descent

$$\min_{\boldsymbol{w}} \sum_{s,a} \mu_\pi(s,a) \left| q_\pi(s,a) - Q_{\boldsymbol{w}}(s,a) \right|^2$$

Prediction, Approximation and Gradient Descent

- Prediction objective:
$$\overline{\mathrm{VE}}(\boldsymbol{w}) = \sum_{q} \mu_\pi(s,a) \left| q_\pi(s,a) - Q_{\boldsymbol{w}}(s,a) \right|^2$$
- Gradient:
$$\nabla \overline{\mathrm{VE}}(\boldsymbol{w}) = -2 \sum_{s,a} \mu_\pi(s,a) \left(q_\pi(s,a) - Q_{\boldsymbol{w}}(s,a) \right) \nabla Q_{\boldsymbol{w}}(s,a)$$
- Stochastic gradient:
$$\widehat{\nabla} \overline{\mathrm{VE}}(\boldsymbol{w}) = -2 \left(q_\pi(S_t,A_t) - Q_{\boldsymbol{w}}(S_t,A_t) \right) \nabla Q_{\boldsymbol{w}}(S_t,A_t)$$

- Not a practical algorithm as q_π is unknown.

Prediction, Approximation and MC

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + 2\alpha_t \left(G_t - Q_{\boldsymbol{w}_t}(S_t, A_t)\right) \nabla Q_{\boldsymbol{w}_t}(S_t, A_t)$$

Monte Carlo Approach

- Replace $q_\pi(S_t, A_t)$ by its Monte Carlo estimate G_t.
- Still a Stochastic Gradient of the original problem with limit (if it exists) satisfying

$$\begin{aligned} &\mathbb{E}_\pi[(G_t - Q_{\boldsymbol{w}_\infty}(S_t, A_t)) \nabla Q_{\boldsymbol{w}_\infty}(S_t, A_t)] \\ &= \mathbb{E}[(q_\pi(S_t, A_t) - Q_{\boldsymbol{w}_\infty}(S_t, A_t)) \nabla Q_{\boldsymbol{w}_\infty}(S_t, A_t)] = 0 \end{aligned}$$

- Convergence ensured for the linear parametrization as it is a convex problem.

- Correspond exactly to the tabular MC prediction algorithm for the tabular parametrization.
- For the linear parametrization:
  Limiting equation: $\mathbb{E}_\pi[q_\pi(S_t, A_t)\Phi(S_t, A_t)] = \mathbb{E}_\pi\left[\Phi(S_t, A_t)\Phi(S_t, A_t)^\top\right] \boldsymbol{w}_\infty$

Prediction, Approximation and TD

$$\mathbf{w}_{t+1} = \mathbf{w}_t + 2\alpha_t \left(R_{t+1} + \gamma Q_{\mathbf{w}_t}(S_{t+1}, A_{t+1}) - Q_{\mathbf{w}_t}(S_t, A_t)\right) \nabla Q_{\mathbf{w}_t}(S_t, A_t)$$

Temporal Differencies Approach

- Replace $q_\pi(S_t, A_t)$ by $R_{t+1} + \gamma Q_{\mathbf{w}_t}(S_{t+1}, A_{t+1})$.
- Not a Stochastic Gradient of the original problem but a Stochastic Approximation algorithm with limit (if it exists) satisfying
$$\begin{aligned}\mathbb{E}_\pi[(R_t + \gamma Q_{\mathbf{w}_\infty}(S_{t+1}, A_{t+1}) - Q_{\mathbf{w}_\infty}(S_t, A_t)) \nabla Q_{\mathbf{w}_\infty}(S_t, A_t)]\\ = \mathbb{E}_\pi[((\mathcal{T}^\pi Q_{\mathbf{w}_\infty} - Q_{\mathbf{w}_\infty})(S_t, A_t)) \nabla Q_{\mathbf{w}_\infty}(S_t, A_t)] = 0\end{aligned}$$
- No simple argument to justify the convergence...

- In general, no straightforward relation with Bellman operator.
- Correspond exactly to the tabular TD prediction algorithm for the tabular parametrization.

$$\mathbf{w}_{t+1} = \mathbf{w}_t + 2\alpha_t \left(\tilde{G}_t - Q_{\mathbf{w}_t}(S_t, A_t)\right) \nabla Q_{\mathbf{w}_t}(S_t, A_t)$$

Temporal Differencies Approach

- Replace $q_\pi(S_t, A_t)$ by any advanced return $\tilde{G}_t$.
- Not a Stochastic Gradient of the original problem but a Stochastic Approximation algorithm with limit (if it exists) satisfying

$$\begin{aligned} &\mathbb{E}_\pi\left[\left(\tilde{G}_t - Q_{\mathbf{w}_t}(S_t, A_t)\right) \nabla Q_{\mathbf{w}_\infty}(S_t, A_t)\right] \\ &= \mathbb{E}_\pi\left[\left((\widetilde{\mathcal{T}}^\pi Q_{\mathbf{w}_\infty} - Q_{\mathbf{w}_\infty})(S_t, A_t)\right) \nabla Q_{\mathbf{w}_\infty}(S_t, A_t)\right] = 0 \end{aligned}$$

- No simple argument to justify the convergence...

- In general, no straightforward relation with Bellman operator.
- Correspond exactly to the tabular TD prediction algorithm for the tabular parametrization.

Prediction, Approximation and Eligibility Trace

$$
\begin{aligned}
z_t &= \gamma\lambda z_{t-1} + \nabla Q_{\mathbf{w}_t}(S_t, A_t) \\
\delta_t &= R_{t+1} + \gamma Q_{\mathbf{w}_t}(S_{t+1}, A_{t+1}) - Q_{\mathbf{w}_t}(S_t, A_t) \\
\mathbf{w}_{t+1} &= \mathbf{w}_t + \alpha_t \delta_t z_t
\end{aligned}
$$

Eligibility Trace

- Rewrite the TD(λ) updates using the backward point of view.
- No strict equivalence due to time evolution of the parameterization.
- Stochastic Approximation with limit (if it exists) satisfying
$$
\begin{aligned}
&\mathbb{E}_\pi[(R_{t+1} + \gamma Q_{\mathbf{w}_\infty}(S_{t+1}, A_{t+1}) - Q_{\mathbf{w}_\infty}(S_t, A_t))\, z_t] \\
&= \mathbb{E}_\pi[(\mathcal{T}^\pi Q_{\mathbf{w}_\infty} - Q_{\mathbf{w}_\infty})(S_t, A_t) z_t] = 0
\end{aligned}
$$
- No simple argument to justify the convergence.

Outline

Linear Parametrization

$$Q_{\boldsymbol{w}}(S_t, A_t) = \Phi(S_t, A_t)^\top \boldsymbol{w} \quad \text{and} \quad \nabla Q_{\boldsymbol{w}}(S_t, A_t) = \Phi(S_t, A_t)$$

Linear Parametrization

- Extension of the tabular setting.
- Derivative is independent of $\boldsymbol{w}$.
- Analysis of Stochastic Approximation often possible!
- More than a toy model as an algorithm not converging in the linear case will almost certainly not converge in a more general setting.

Linear Parametrization and MC

$$\text{Iteration:} \boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha_t (G_t - \Phi(S_t, A_t)^\top \boldsymbol{w}_t) \Phi(S_t, A_t)$$

$$\text{Limiting equation: } \mathbb{E}_\pi[q_\pi(S_t, A_t)\Phi(S_t, A_t)] = \mathbb{E}_\pi\left[\Phi(S_t, A_t)\Phi(S_t, A_t)^\top\right] \boldsymbol{w}_\infty$$

$$\text{ODE: } \frac{d\boldsymbol{w}}{dt} = -\mathbb{E}_\pi\left[\Phi(S_t, A_t)\Phi(S_t, A_t)^\top\right](\boldsymbol{w} - \boldsymbol{w}_\infty)$$

Linear Parametrization and MC

- Limiting equation is a linear equation.
- Under asymptotic stationarity assumption, convergence of ODE as $\mathbb{E}_\pi\left[\Phi(S_t, A_t)\Phi(S_t, A_t)^\top\right]$ is a Gram Matrix with positive eigenvalues (provided Φ is not redundant and under an ergodicity assumption).
- Need to explore all state-action pairs!

Linear Parametrization and TD

Iteration: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha_t (R_{t+1} + \gamma \Phi(S_{t+1}, A_{t+1})^\top \boldsymbol{w}_t - \Phi(S_t, A_t)^\top \boldsymbol{w}_t)\Phi(S_t, A_t)$

Lim. eq.: $\mathbb{E}_\pi[r(S_T, A_t)\Phi(S_t, A_t)] = \mathbb{E}_\pi\left[\Phi(S_t, A_t)\left(\Phi(S_t, A_t)^\top - \gamma\Phi(S_{t+1}, A_{t+1})^\top\right)\right]\boldsymbol{w}_\infty$

ODE: $\dfrac{d\boldsymbol{w}}{dt} = -\mathbb{E}_\pi\left[\Phi(S_t, A_t)\left(\Phi(S_t, A_t)^\top - \gamma\Phi(S_{t+1}, A_{t+1})^\top\right)\right](\boldsymbol{w} - \boldsymbol{w}_\infty)$

Linear Parametrization and TD

- Convergence of ODE if $\mathbb{E}_\pi\left[\Phi(S_t, A_t)\left(\Phi(S_t, A_t)^\top - \gamma\Phi(S_{t+1}, A_{t+1})^\top\right)\right]$ has complex eigenvalues with positive real parts...
- which can be proved to be true under an ergodicity assumption!
- Need to explore all state-action pairs!
- Different solution than MC! Minimization of the Projected Bellman Residual...
- **Prop:**

$$\overline{VE}(\boldsymbol{w}_{\mathrm{TD}}) \leq \frac{1}{1-\gamma}\overline{VE}(\boldsymbol{w}_{\mathrm{MC}}) = \frac{1}{1-\gamma}\min_{\boldsymbol{w}}\overline{VE}(\boldsymbol{w})$$

Least-Squares TD

$$b = \mathbb{E}_\pi[r(S_T, A_t)\Phi(S_t, A_t)] \sim \frac{1}{t}\sum_{t'=0}^{t-1} R_{t'+1}\phi(S_{t'}, A_{t'})$$

$$A = \mathbb{E}_\pi\left[\Phi(S_t, A_t)\left(\Phi(S_t, A_t)^\top - \gamma\Phi(S_{t+1}, A_{t+1})^\top\right)\right]$$

$$\sim \frac{1}{t}\sum_{t'=0}^{t-1} \Phi(S_{t'}, A_{t'})\left(\Phi(S_{t'}, A_{t'})^\top - \gamma\Phi(S_{t'+1}, A_{t'+1})^\top\right)$$

Least-Squares TD

- Bypass the Stochastic Approximation scheme by estimating directly its limit:
$$\boldsymbol{w}_\infty = A^{-1}b$$
- Much more sample efficient.
- Recursive implementation possible.
- Recursive implementation maintaining an estimate of A^{-1} is also possible.

Advanced Returns

$$\text{Return: } \tilde{G}_t = \tilde{R}_{t+1} + \tilde{\Phi}_t^\top \boldsymbol{w} \quad \text{(affine formula)}$$

$$\text{Iteration: } \boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha_t(\tilde{R}_t + \tilde{\Phi}_t^\top \boldsymbol{w}_t - \Phi(S_t, A_t)^\top \boldsymbol{w}_t)\Phi(S_t, A_t)$$

$$\text{Lim. eq.: } \mathbb{E}_\pi\left[\tilde{R}_t \Phi(S_t, A_t)\right] = \mathbb{E}_\pi\left[\Phi(S_t, A_t)\left(\Phi(S_t, A_t)^\top - \Phi_t{}^\top\right)\right] \boldsymbol{w}_\infty$$

$$\text{ODE: } \frac{d\boldsymbol{w}}{dt} = -\mathbb{E}_\pi\left[\Phi(S_t, A_t)\left(\Phi(S_t, A_t)^\top - \tilde{\Phi}_t^\top\right)\right](\boldsymbol{w} - \boldsymbol{w}_\infty)$$

Linear Parametrization and TD

- Convergence of ODE if $\mathbb{E}_\pi\left[\Phi(S_t, A_t)\left(\Phi(S_t, A_t)^\top - \tilde{\Phi}_t^\top\right)\right]$ has complex eigenvalues with positive real parts...
- which can be proved to be true for the advanced returns under an ergodicity assumption!

Outline

On-Policy Prediction

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + 2\alpha_t \left(\tilde{G}_t - Q_{\boldsymbol{w}_t}(S_t, A_t)\right) \nabla Q_{\boldsymbol{w}_t}(S_t, A_t)$$

On-line TD Algorithm

- Use the policy Π to obtain the interactions $S_t A_t R_{t+1} S_{t+1} A_{t+1} \dots$
- Convergence. . . for linear parametrization under stationarity and coverage assumptions!
- Appear to *converge* even with more complex parametrization.

- Monte Carlo can be used for short episodes.
- Similar observations for elegibility trace.

On-Policy Control

$$\mathbf{w}_{t+1} = \mathbf{w}_t + 2\alpha_t \left(\tilde{G}_t - Q_{\mathbf{w}_t}(S_t, A_t)\right) \nabla Q_{\mathbf{w}_t}(S_t, A_t)$$
$$\pi_{t+}(s) = \operatorname{argmax} Q_{\mathbf{w}_t}(s, \cdot) \quad \text{(plus exploration)}$$

On-Policy Control

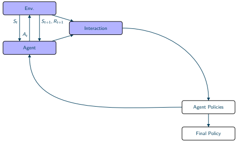
- SARSA type algorithm: update Q values and policy π while using policy π.
- Not a Stochastic Approximation algorithm anymore...
- Not approximate policy improvement as no sup-norm control...
- No proof of convergence... but appear to work well in practice.

- Non trivial scheduling issue in the definition of $\tilde{G}_t$.
- More constraints with eligibility trace.

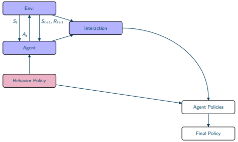
Outline

On-Policy vs Off-Policy

From



to



On-Policy vs Off-Policy

- On-Policy: the policy b used to interact is the same than the policy Π evaluated or optimized.
- Off-Policy: the policy b used to interact may be different from the policy Π evaluated or optimized.

- Off-Policy correction available for the return.

Off-Policy Prediction

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha_t \left(\tilde{G}_t - Q_{\boldsymbol{w}_t}(S_t, A_t)\right) \nabla Q_{\boldsymbol{w}_t}(S_t, A_t)$$

Off-policy TD Algorithm

- Use a policy b to obtain the interactions $S_t A_t R_{t+1} S_{t+1} A_{t+1} \ldots$
- Compute an (importance-sampling based) corrected return.
- Use it in the algorithm.

- **Can fail spectacularly!**
- Monte Carlo will work.

Off-Policy Divergence

Simplest Example?

- Simple transition with a reward 0.
- TD error:

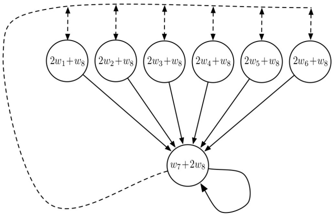
$$\begin{aligned}\delta_t &= R_{t+1} + \gamma V_{\boldsymbol{w}_t}(S_{t+1}) - V_{\boldsymbol{w}_t}(S_t) \\ &= 0 + \gamma 2\boldsymbol{w}_t - \boldsymbol{w}_t = (2\gamma - 1)\boldsymbol{w}_t\end{aligned}$$

- Off-policy semi-gradient TD(0) update:

$$\begin{aligned}\boldsymbol{w}_{t+1} &= \boldsymbol{w}_t + \alpha_t \rho_t \delta_t \nabla V(S_{t+1}, \boldsymbol{w}_t) \\ &= \boldsymbol{w}_t + \alpha_t \times 1 \times (2\gamma - 1)\boldsymbol{w}_t = (1 + \alpha_t(2\gamma - 1))\boldsymbol{w}_t\end{aligned}$$

- Explosion if this transition is explored without $\boldsymbol{w}$ being update on other transitions as soon as $\gamma > 1/2$.

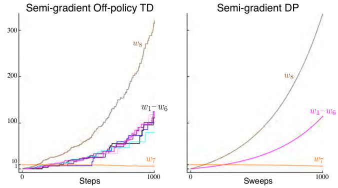
Off-Policy Divergence



$\pi(\text{solid}|\cdot) = 1$

$b(\text{dashed}|\cdot) = 6/7$
$b(\text{solid}|\cdot) = 1/7$

$\gamma = 0.99$



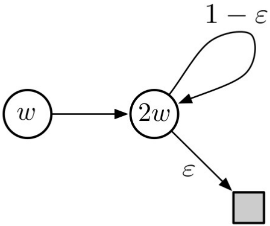
Baird's Counterexample

- Divergence of off-policy algorithm even without sampling, i.e. in Dynamic Programming.

Discounted

Source: Sutton and Barto

Off-Policy Divergence



Tsistiklis and Van Roy's Counterexample

- Exact minimization of bootstrapped $\overline{VE}$ at each step:

$$\begin{aligned} \boldsymbol{w}_{t+1} &= \operatorname*{argmin}_{\boldsymbol{w}} \sum_{s} \left(V_{\boldsymbol{w}_t}(s) - \mathbb{E}_\pi[R_{t+1} + \gamma V_{\boldsymbol{w}_t}(S_{t+1}) | S_t = s]\right)^2 \\ &= \operatorname*{argmin}_{\boldsymbol{w}} (\boldsymbol{w} - \gamma 2\boldsymbol{w}_t)^2 + (2\boldsymbol{w} - (1-\epsilon)\gamma 2\boldsymbol{w}_t)^2 \\ &= \frac{6-4\epsilon}{5}\gamma \boldsymbol{w}_t \end{aligned}$$

- Divergence if $\gamma > 5/(6-4\epsilon)$.

Linear Parametrization and TD

Iteration: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha_t (R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) \Phi(S_{t+1}, a)^\top \boldsymbol{w}_t - \Phi(S_t, A_t)^\top \boldsymbol{w}_t) \Phi(S_t, A_t)$

Lim. eq$\mathbb{E}_b[r(S_T, A_t)\Phi(S_t, A_t)] = \mathbb{E}_b\left[\Phi(S_t, A_t)\left(\Phi(S_t, A_t)^\top - \gamma \sum_a \pi(a|S_{t+1})\Phi(S_{t+1}, a)^\top\right)\right] \boldsymbol{w}_\infty$

ODE: $\frac{d\boldsymbol{w}}{dt} = -\mathbb{E}_b\left[\Phi(S_t, A_t)\left(\Phi(S_t, A_t)^\top - \gamma \sum_a \pi(a|S_{t+1})\Phi(S_{t+1}, q^\top\right)\right](\boldsymbol{w} - \boldsymbol{w}_\infty)$

Linear Parametrization and TD

- Convergence of ODE if
$$\mathbb{E}_b\left[\Phi(S_t, A_t)\left(\Phi(S_t, A_t)^\top - \gamma \sum_a \pi(a|S_{t+1})\Phi(S_{t+1}, q^\top\right)\right] = \Phi\Xi(I - \gamma P^\pi)\Phi^\top$$
(with $\Phi = (\Phi(s, a))$, $\Xi = \mathrm{diag}(\mu(s, a))$) and $P\pi$ the transition matrix associated to π) has complex eigenvalues with positive real parts...
- Proof for on-policy relies on $\mu = \mu_\pi$ which satisfies $\mu_\pi{}^\top P_\pi = \mu_\pi{}^\top$.
- Not true anymore with an arbitrary behavior policy!

Deadly Triad

Deadly Triad

- **Function approximation**
- **Bootstrapping**
- **Off-policy training**

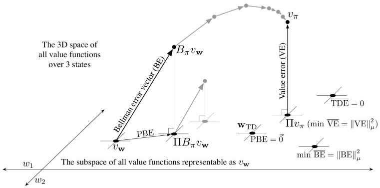
- **Instabilities as soon as the three are present!**

Issue

- Function approximation is unavoidable.
- Bootstrap is much more computational and data efficient.
- Off-policy may be avoided. . . but essential when dealing with extended setting (learn from others or learn several tasks)

- Dead End?

Objective?

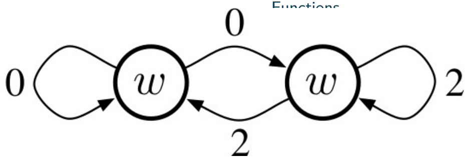


Linear Parametrization Target?

- Prediction objective $\overline{VE}$:
$$\|q_\pi - Q_{\boldsymbol{w}}\|_\mu^2$$
- Bellman Error $\overline{BE}$:
$$\|\mathcal{T}^\pi Q_{\boldsymbol{w}} - Q_{\boldsymbol{w}}\|_\mu^2$$
- Projected Bellman Error $\overline{PBE}$:
$$\|\operatorname{Proj}\mathcal{T}^\pi Q_{\boldsymbol{w}} - Q_{\boldsymbol{w}}\|_\mu^2$$

with $\operatorname{Proj} = \Phi(\Phi^\top \Xi \Phi)\Phi\,(\Phi)\,\Xi$.

Prediction Objective



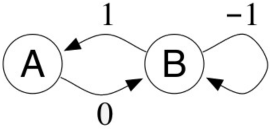
Prediction Objective

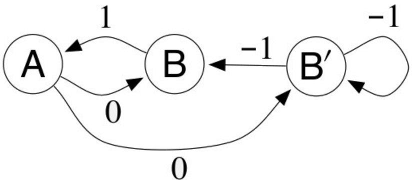
- Two MRP with the same outputs (because of approximation).
- but different $\overline{VE}$.
- Impossibility to learn $\overline{VE}$.
- Minimizer however is learnable:

$$\begin{aligned}\overline{RE}(\boldsymbol{w}) &= \mathbb{E}\left[(G_t - V_{\boldsymbol{w}_t}(S_t))^2\right] \\ &= \overline{VE}(\boldsymbol{w}) + \mathbb{E}\left[(G_t - v_\pi(S_t))^2\right]\end{aligned}$$

- MC method target.

Bellman Error



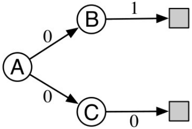


Bellman Error

- Two MRP with the same outputs (because of approximation).
- Different $\overline{BE}$.
- Different minimizer!
- $\overline{BE}$ is not learnable!

TD Error

$$\overline{TDE}(\mathbf{w}) = \|\mathbb{E}_\pi\left[\delta_t^2 | S_t, A_t\right]\|_\mu$$



Mean-Squares TD Error

- $\overline{TDE}(\mathbf{w}) = \mathbb{E}_b\left[\rho_t \delta^2\right]$
- Gradient: $\nabla \overline{TDE}(\mathbf{w}) =$ $\mathbb{E}_b[\rho_t\,(R_t + \gamma Q_{\mathbf{w}}(S_{t+1}, A_{t+1})) - Q_{\mathbf{w}_t}(S_t, A_t))\,(\gamma \nabla Q_{\mathbf{w}_t}(S_{t+1}, A_{t+1}) - \nabla Q_{\mathbf{w}_t}(S_t, A_t))]$
- SGD algorithm. . .
- but solutions often converge to not to a *desirable place* even without approximation!

Projected Bellman Error

$$\| \operatorname{Proj} \mathcal{T}^\pi Q_{\boldsymbol{w}} - Q_{\boldsymbol{w}} \|_\mu^2 \quad \text{with } \operatorname{Proj} = \Phi(\Phi^\top \Xi \Phi)^{-1} \Phi^\top \Xi.$$

Projected Bellman Error

- Rewriting

$$\overline{PBE}(\boldsymbol{w}) = \| \operatorname{Proj} \mathcal{T}^\pi q_{\boldsymbol{w}} - q_{\boldsymbol{w}} \|_\mu^2 = \| \operatorname{Proj} \delta_{\boldsymbol{w}} \|_\mu^2$$
$$= (\operatorname{Proj} \delta_{\boldsymbol{w}})^\top \Xi (\operatorname{Proj} \delta_{\boldsymbol{w}}) = (\Phi^\top \Xi \delta_{\boldsymbol{w}})^\top \left(\Phi^\top \Xi \Phi\right)^{-1} (\Phi^\top \Xi \delta_{\boldsymbol{w}})$$

- Gradient:

$$\nabla \overline{PBE}(\boldsymbol{w}) = 2 \nabla (\Phi^\top \Xi \delta_{\boldsymbol{w}})^\top \left(\Phi^\top \Xi \Phi\right)^{-1} (\Phi^\top \Xi \delta_{\boldsymbol{w}})$$

- Expectations:

$$\Phi^\top \Xi \delta_{\boldsymbol{w}} = \mathbb{E}_b[\rho_t \delta_t \Phi(S_t, A_t)]$$
$$\nabla (\Phi^\top \Xi \delta_{\boldsymbol{w}})^\top = \mathbb{E}_b\left[\rho_t (\gamma \Phi(S_{t+1}, A_{t+1}) - \Phi(S_t, A_t)) \Phi(S_t, A_t)^\top\right]$$
$$\Phi^\top \Xi \Phi = \mathbb{E}_b\left[\Phi(S_t, A_t) \Phi(S_t, A_t)^\top\right]$$

- Not yet a SGD/SA as the gradient is a product of several terms...

Discounted

Projected Bellman Error

Gradient and Stochastic Approximation

- Gradient:

$$\nabla \overline{PBE}(\boldsymbol{w}) = 2\mathbb{E}_b\left[\rho_t(\gamma\Phi(S_{t+1}, A_{t+1}) - \Phi(S_t, A_t))\Phi(S_t, A_t)^\top\right]$$
$$\left(\mathbb{E}_b\left[\Phi(S_t, A_t)\Phi(S_t, A_t)^\top\right]\right)^{-1}\mathbb{E}_b[\rho_t\delta_t\Phi(S_t, A_t)]$$

- Least-squares inside:

$$v = \left(\mathbb{E}_b\left[\Phi(S_t, A_t)\Phi(S_t, A_t)^\top\right]\right)^{-1}\mathbb{E}_b\left[\rho_t\delta_t\Phi(S_t, A_t)^\top\right]$$
$$\Leftrightarrow v = \operatorname*{argmin}_v \mathbb{E}_b\left[\left(\Phi(S_t, A_t)^\top v_t - \rho_t\delta_t\right)^2\right]$$

which can be estimated by

$$v_{t+1} = v_t + \beta_t\Phi(S_t, A_t)(\delta_t - \rho_t\Phi(S_t, A_t)^\top v_t)$$

- Plugin pseudo gradient (SA):

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - 2\alpha_t\rho_t(\gamma\Phi(S_{t+1}, A_{t+1}) - \Phi(S_t, A_t))\Phi(S_t, A_t)^\top v_t$$

- Same target than Pseudo Gradient but converging algorithm provided $\alpha_t \ll \beta_t$.

Gradient TD Algorithm

GTD

- Simultaneous update:

$$v_{t+1} = v_t + \beta_t \Phi(S_t, A_t)(\delta_t - \rho_t \Phi(S_t, A_t)^\top v_t)$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - 2\alpha_t \rho_t (\gamma \Phi(S_{t+1}, A_{t+1}) - \Phi(S_t, A_t)) \Phi(S_t, A_t)^\top v_t$$

- As $\alpha_t \ll \beta_t$, $\boldsymbol{w}$ is seen as constant by v...

TDC

- Simultaneous update:

$$v_{t+1} = v_t + \beta_t \Phi(S_t, A_t)(\delta_t - \rho_t \Phi(S_t, A_t)^\top v_t)$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - 2\alpha_t \rho_t (\delta_t \Phi(S_t, A_t) - \gamma \Phi(S_{t+1}, A_{t+1})) \Phi(S_t, A_t)^\top v_t$$

- Obtained by a similar derivation but faster in practice...
- As $\alpha_t \ll \beta_t$, $\boldsymbol{w}$ is seen as constant by v...

- Restricted to the linear setting but interesting insight.

Outline

Stochastic Approximation

$$\theta_{k+1} = \theta_k + \alpha_k h_k(\theta_k) \quad \text{with} \quad h_k(\theta) = H(\theta) + \epsilon_k + \eta_k$$
$$\implies \theta_k \to \{\theta, H(\theta) = 0\}$$

Stochastic Approximation

- Family of sequential stochastic algorithm converging to a zero of a function.
- Classical assumptions:
  - $\mathbb{E}[\epsilon_k] = 0$, $\mathbb{V}\text{ar}\,[\epsilon_k] < \sigma^2$, and $\mathbb{E}[\|\eta_k\|] \to 0$,
  - $\sum_k \alpha_k \to \infty$ and $\sum_k \alpha_k^2 < \infty$,
  - the algorithm converges if we replace h_k by H.

- Convergence toward a neighborhood if α is kept constant (as often in practice).
- Most famous example are probably Robbins-Monro and SGD.
- Proof quite technical in general.
- The convergence with H is easy to obtain for a contraction.

Stochastic Approximation and ODE

From $\theta_{k+1} = \theta_k + \alpha_k h_k(\theta_k)$ with $h_k(\theta) = H(\theta) + \epsilon_k + \eta_k$

to $\dfrac{d\tilde{\theta}}{dt} = H(\tilde{\theta})$

ODE Approach

- General proof showing that the algorithm converges provided the ODE converges.
- Rely on the rewriting the equation
$$\frac{\theta_{k+1} - \theta_k}{\alpha_k} = h_k(\theta_k) = H(\theta_k) + \epsilon_k + \eta_k$$
- α_k can be interpreted as a time difference allowing to define a time $t_k = \sum_{t' \leq t} \alpha_k$.
- $\theta(t)$ is piecewise affine and defined through its derivative at time $t \in (t_k, t_{k+1})$.
- This piecewise function remains close to any solution of the ODE starting from θ_k for an arbitrary amount of time provided k is large enough.

- More general proofs based on martingale.

Stochastic Approximation

$$\begin{cases} \theta_{k+1} = \theta_k + \alpha_k h_k(\theta_k, \nu_k) \\ \nu_{k+1} = \nu_k + \beta_k g_k(\theta_k, \nu_k) \end{cases} \quad \text{with} \quad \begin{cases} h_k(\theta, \nu) = H(\theta, \nu) + \epsilon_k + \eta_k \\ g_k(\theta, \nu) = G(\theta, \nu) + \epsilon'_k + \eta'_k \end{cases}$$

$$\implies \theta_k \to \{\theta, H(\theta, \nu(\theta)) = 0, \nu(\theta) \in \{\nu, G(\theta, \nu) = 0\}\}$$

Stochastic Approximation

- Family of sequential stochastic algorithm converging to a zero of a function.
- Classical assumptions:
  - $\mathbb{E}[\epsilon_k] = 0$, $\mathbb{V}\mathrm{ar}\,[\epsilon_k] < \sigma^2$, and $\mathbb{E}[\|\eta_k\|] \to 0$,
  - $\sum_k \alpha_k \to \infty$ and $\sum_k \alpha_k^2 < \infty$,
  - $\sum_k \beta_k \to \infty$ and $\sum_k \beta_k^2 < \infty$,
  - $\alpha_k/\beta_k \to 0$ (two-scales assumption),
  - the algorithm converges if we replace h_k and g_k by H and G.

- Convergence toward a neighborhood if $\alpha \ll \beta$ are kept constant (as often in practice).

Stochastic Approximation and ODE

From $\begin{cases} \theta_{k+1} = \theta_k + \alpha_k h_k(\theta_k, \nu_k) \\ \nu_{k+1} = \nu_k + \beta_k + g_k(\theta_k, \nu_k) \end{cases}$ with $\begin{cases} h_k(\theta, \nu) = H(\theta, \nu) + \epsilon_k + \eta_k \\ g_k(\theta, \nu) = G(\theta, \nu) + \epsilon'_k + \eta'_k \end{cases}$

to $\dfrac{d\tilde{\theta}}{dt} = H(\tilde{\theta}, \tilde{\nu}(\tilde{\theta}))$ with $\tilde{\nu}(\theta)$ the limit of $\dfrac{d\tilde{\nu}}{dt} = G(\theta, \tilde{\nu})$

ODE Approach

- General proof showing that the algorithm converges provided the two ODE converge.
- Quite generic setting and source of new algorithm or insight on existing ones.
- Importance of having two scales...

- Can be used to prove the convergence of GTD and TDC!

Outline

Simplified Deep Q-Learning

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \beta_t (R_{t+1} + \gamma \max_a Q_{\nu_t}(S_{t+1}, a) - Q_{\mathbf{w}_t}(S_t, A_t)) \nabla Q_{\mathbf{w}_t}(S_t, A_t)$$

$$\nu_t = \mathbf{w}_{\lceil t/T \rceil T}$$

Simplified Deep Q-Learning

- Stochastic Approximation for a fixed ν:
  - Limiting equation:
    $$\mathbb{E}_b[(\mathcal{T}^\star Q_\nu(S_t, A_t) - Q_{\mathbf{w}_\infty}(S_t, A_t)) \nabla Q_{\mathbf{w}_\infty}(S_t, A_t)] = 0$$
  - Stochastic Gradient Descent of
    $$\mathbb{E}_b\left[(\mathcal{T}^\star Q_\nu(S_t, A_t) - Q_{\mathbf{w}}(S_t, A_t))^2\right]$$
  - $Q_{\mathbf{w}} \to \mathcal{T}^\star Q_\nu$
- Approximate Value Iteration Scheme!

- Two-scales algorithm flavour as ν is kept constant.
- Explicit two scales with $\nu_{t+1} = \nu_t + \alpha_t(\mathbf{w}_t - \nu_t)$ variation.
- Could be used for prediction with $R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q_{\nu_t}(S_{t+1}, a)$

Deep Q-Learning

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \beta_t(R_t + \gamma \max_a Q_{\nu_t}(S_{t+1}, a) - Q_{\mathbf{w}}(S_t, A_t))\nabla Q_{\mathbf{w}}(S_t, A_t)$$

$$\nu_t = \mathbf{w}_{\lceil t/T \rceil T}$$

- **Who are $S_t, A_t, R_{t+1}, S_{t+1}$?** and thus to what corresponds $\mathbb{E}_b$?

Simplified Deep Q-Learning

- Use a behaviour policy b.
- The current greedy plus exploration Q-policy can be used.

Neural Fitted-Q

- Instead of a policy b, use a fix dataset $\mathcal{D}$ of $S_t, A_t, R_{t+1}, S_{t+1}$.
- Several pass on the data can be made.

Deep Q-Learning

- Use the current greedy plus exploration Q-policy to populate a FIFO buffer $\mathcal{D}$.
- Use random samples of the buffer $\mathcal{D}_t$ (more than one per interaction is OK).

Deep Q-Learning

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \beta_t (R_t + \gamma \max_a Q_{\nu_t}(S_{t+1}, a) - Q_{\mathbf{w}}(S_t, A_t)) \nabla Q_{\mathbf{w}}(S_t, A_t)$$

$$\nu_t = \mathbf{w}_{\lceil t/T \rceil T}$$

Plus tricks

Deep Q-Learning Tricks

- Replay buffer
- Double Q-Learning
- Better Exploration
- Advanced Return and Distributional
- Network Architecture

- Rainbow paper. . .

Replay Buffer

Replay Buffer

- Replace an expectation over real trajectories by an empirical average over past (short) sub-trajectories stored in a replay buffer.
- The empirical average corresponds to uniform sampling.
- If the policy is changing across time, we should use a importance sampling correction to be faithful with the theory...
- Not necessary for one-step Q learning but required for most of the other methods where replay buffer is used.
- Often no correction in practice if the policies used in the buffer are closed to the current one.
- Prioritized sweeping variant possible...

- Buffer can be constructed in parallel of the learning part.
- Only requires to transmit the *current* greedy plus exploration Q-policy.

Double Q-Learning

Q-Learning and overestimation

- Target: $R_{s,a} + \gamma \max_{a'} Q_{\mathbf{w}}(s', a')$
- Approximation issue: $Q_{\mathbf{w}}(s', a') \sim Q(s, a) + \epsilon(s, a)$
- Consequence: $\mathbb{E}[\max_a Q_{\mathbf{w}}(S_t, a)] \geq \max\left(Q(s, a) + \mathbb{E}[\epsilon(s, a)]\right)$

Double Q-Learning with two Q functions: $Q_{\mathbf{w}_1}$ and $Q_{\mathbf{w}_2}$

- Used in a crossed way for the target of $Q_{\mathbf{w}_i}$:

$$R_{s,a} + \gamma Q_{\mathbf{w}_{i'}}(s', \operatorname*{argmax}_{a'} Q_{\mathbf{w}_i}(s', a'))$$

- Mitigates the bias.

Clipped Q-Learning with several Q functions: $Q_{\mathbf{w}_i}$

- Used in a pessimistic way for the target of $Q_{\mathbf{w}_i}$:

$$R_{s,a} + \gamma \min_{i'} Q_{\mathbf{w}_{i'}}(s', \operatorname*{argmax}_{a'} Q_{\mathbf{w}_i}(s', a'))$$

- Seems even more efficient.

Outline

Continuous Action

- Case (almost) not yet covered in the lectures.
- Most complex theoretical extension.

Prediction

- No algorithmic issue if one can sample π.
- Off-policy can be considered under a domination assumption.

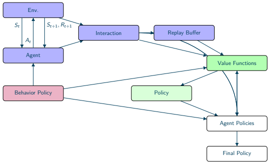
Planning

- Main issue is the argmax of the greedy policy (or the sampling of Gibbs policy).
- May be impossible to compute.
- Possible if the parametrization of Q with respect to a is simple (e.g. explicit quadratic dependency in a).
- An alternative could be to approximate the argmax operator, or to learn how to approximate the argmax directly, which is very close to approximating directly the policy itself...

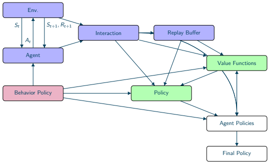
Outline

Policy Point of View

From



to



Policy Point of View

- Optimize policy directely instead of deriving it from a value function.
- Avoid the argmax operator.
- Most natural POV?

- Pontryagin vs Hamilton-Jacobi(-Bellman) in control!

Outline

Policy and Goal

$$J_\mu(\pi) = \sum_s \mu(s) v_\pi(s)$$

Goal: average expected return over the states

- Target used to define the linear programming formulation of an optimal policy in the tabular setting.
- μ can be the initial distribution of the states (independent of π)...
- but may also depends on π (for instance the associated stationary measure)
- Other choices will appear.

- Goal: optimize $J_\mu(\pi)$ in π!

Parametric Policy

$$\pi_\theta(a|s) = \begin{cases} \frac{e^{h_\theta(a,s)}}{\sum_{a'} e^{h_\theta(a,s')}} & \text{(softmax)} \\ P_{h_\theta(s)}(a) & \text{(parametric conditional model)} \\ \mathbf{1}_{a=h_\theta(s)} & \text{(deterministic)} \end{cases}$$

Parametric Policy

- Restriction of the set of policy to a parametrized one.
- Most classical parametrizations:
  - Soft-max with a preference function $h_\theta(a, s)$,
  - Parametric conditional model with parameter $h_\theta(s)$
- To be useful need to be able to sample the distribution.
- h_θ: from linear model to deep learning...
- Most of our result will assume that $\pi_\theta(a|s)$ is differentiable with respect to θ.

- Deterministic policies will be considered with a different analysis.

Episodic Setting: Gradient of Expected Returns

$$v_{\pi_\theta}(s) = \mathbb{E}_{\pi_\theta}\left[G_0 | S_0 = s\right]$$

$$\nabla_\theta v_{\pi_\theta}(s) = \mathbb{E}_{\pi_\theta}\left[\left(\sum_{t=0}^{T_\tau - 1} \nabla \log \pi_\theta(A_t|S_t)\right) G_0 \middle| S_0 = s\right]$$

Expected Returns

- Rely on $v_{\pi_\theta}(s) = \sum_\tau \mathbb{P}_{\pi_\theta}(\tau | S_0 = s)\, G_0(\tau)$ and

$$\begin{aligned}
\nabla \mathbb{P}_{\pi_\theta}(\tau|S_0 = s) &= \mathbb{P}_{\pi_\theta}(\tau|S_0 = s)\, \nabla \log \mathbb{P}_{\pi_\theta}(\tau|S_0 = s) \\
&= \mathbb{P}_{\pi_\theta}(\tau|S_0 = s) \sum_t \left(\nabla \log \pi_\theta(A_t|S_t) + \nabla p(R_{t+1}, S_{t+1}|S_t, A_t)\right) \\
&= \mathbb{P}_{\pi_\theta}(\tau|S_0 = s) \sum_t \nabla \log \pi_\theta(A_t|S_t)
\end{aligned}$$

- In an episodic setting, any trajectory τ ends at a finite time T_τ.

Episodic Setting: Policy Gradient Theorem

$$J_{\mu_0}(\pi_\theta) = \sum_s \mathbb{P}(S_0 = s)\, v_{\pi_\theta}(s)$$

$$\nabla J_{\mu_0}(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\left(\sum_{t=0}^{T_\tau - 1} \nabla \log \pi_\theta(A_t | S_t)\right) G_0\right]$$

Policy Gradient Theorem

- Natural μ: initial state distribution.
- Gradient is an expectation: MC type algorithm...

- Can be interpreted as the gradient of a the maximum likelihood of the actions weighted by the return.
- Favors good actions over bad ones.

Baseline and Variance Reduction

$$J_{\mu_0}(\pi_\theta) = \sum_s \mathbb{P}(S_0 = s)\, v_{\pi_\theta}(s)$$

$$\nabla J_{\mu_0}(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\left(\sum_{t=0}^{T_\tau - 1} \nabla \log \pi_\theta(A_t|S_t)\right)(G_0 - b)\right]$$

Variance Reduction and Baseline

- The previous formulae are valid if one replace G_0 by any function of τ.
- For any constant b, this leads to

$$\nabla \mathbb{E}_{\pi_\theta}[b] = 0 = \mathbb{E}_{\pi_\theta}\left[\left(\sum_{t=0}^{T_\tau - 1} \nabla \log \pi_\theta(A_t|S_t)\right) b\right]$$

- Optimal value for
$b = \mathbb{E}_{\pi_\theta}\left[\left(\sum_{t=0}^{T_\tau-1} \nabla \log \pi_\theta(A_t|S_t)\right)^2 G_0\right] / \mathbb{E}_{\pi_\theta}\left[\left(\sum_{t=0}^{T_\tau-1} \nabla \log \pi_\theta(A_t|S_t)\right)^2\right]$
- Most used value $b = \mathbb{E}_{\pi_\theta}[G_0]$.

Gradient(s) of Expected Return

$$v_{\pi_\theta}(s) = \mathbb{E}_{\pi_\theta}\left[\sum \gamma^t R_t \middle| S_0 = s\right]$$

$$\nabla v_{\pi_\theta}(s) = \sum_t \gamma^t \mathbb{E}_{\pi_\theta}\left[\left(\sum_{t'=0}^{t-1} \nabla \log \pi_\theta(A_{t'}|S_{t'})\right) R_t \middle| S_0 = s\right]$$

$$= \sum_{t'} \mathbb{E}_{\pi_\theta}\left[\nabla \log \pi_\theta(A_{t'}|S_{t'}) \left(\sum_{t \geq t'} \gamma^t R_t\right) |S_0 = s\right]$$

$$= \sum_{t'} \gamma^{t'} \mathbb{E}_{\pi_\theta}[\nabla \log \pi_\theta(A_{t'}|S_{t'}) q_{\pi_\theta}(S_{t'}, A_{t'})|S_0 = s]$$

$$= \sum_{t'} \gamma^{t'} \mathbb{E}_{\pi_\theta}\left[\nabla \log \pi_\theta(A_{t'}|S_{t'}) \underbrace{(q_{\pi_\theta}(S_{t'}, A_{t'}) - v_{\pi_\theta}(S_{t'}))}_{a_{\pi_\theta}(S_{t'}, A_{t'})} |S_0 = s\right]$$

From Returns to Value Functions

- Action point of view and use of value functions.

More Gradient(s)

$$
\begin{aligned}
\nabla v_{\pi_\theta}(s) &= \sum_{t'} \gamma^{t'} \mathbb{E}_{\pi_\theta}[\nabla \log \pi_\theta(A_{t'}|S_{t'}) q_{\pi_\theta}(S_{t'}, A_{t'})|S_0 = s] \\
&= \sum_{t'} \gamma^{t'} \mathbb{E}_{\pi_\theta}[\nabla \log \pi_\theta(A_{t'}|S_{t'}) a_{\pi_\theta}(S_{t'}, A_{t'})|S_0 = s] \\
&= \sum_{s'} \left(\sum_t \gamma^t \mathbb{P}_{\pi_\theta}(S_t = s'|S_0 = s) \right) \left(\sum_a \pi_\theta(a|s') \nabla \log \pi_\theta(a|s') q_{\pi_\theta}(s', a) \right) \\
&= \sum_{s'} \left(\sum_t \gamma^t \mathbb{P}_{\pi_\theta}(S_t = s'|S_0 = s) \right) \left(\sum_a \pi_\theta(a|s') \nabla \log \pi_\theta(a|s') a_{\pi_\theta}(s', a) \right)
\end{aligned}
$$

Focus on states

- Even more stochastic gradients!

Policy Gradient(s)

$$J_{\mu_0}(\pi_\theta) = \sum_s \mu_0(s) v_{\pi_\theta}(s)$$

$$\begin{aligned}\nabla J_{\mu_0}(\pi_\theta) &= \sum_s \left(\sum_t \gamma^t \mathbb{P}_{\pi_\theta}(S_t = s)\right)\left(\sum_a \pi_\theta(a|s) \nabla \log \pi_\theta(a|s) q_{\pi_\theta}(s,a)\right) \\ &= \sum_s \left(\sum_t \gamma^t \mathbb{P}_{\pi_\theta}(S_t = s)\right)\left(\sum_a \pi_\theta(a|s) \nabla \log \pi_\theta(a|s) (q_{\pi_\theta}(s,a) - v_{\pi_\theta}(s,a))\right)\end{aligned}$$

Discounted Setting

- Average (discounted) return from the beginning.
- Focus on early steps in discounted setting...

Policy Improvement Lemma

$$J_{\mu_0}(\pi') - J_{\mu_0}(\pi) = \sum_s \sum_t \gamma^t \mathbb{P}_{\pi'}(S_t = s) \left(\sum_a (\pi'(a|s) - \pi(a|s)) \, q_\pi(s, a) \right)$$
$$= \sum_s \sum_t \gamma^t \mathbb{P}_{\pi'}(S_t = s) \left(\sum_a (\pi'(a|s) - \pi(a|s)) \, a_\pi(s, a) \right)$$

Proof

- By construction, if S_t is a trajectory using policy π':

$$v_{\pi'}(S_t) - v_\pi(S_t) = \sum_a (\pi'(a|S_t) - \pi(a|S_t)) \, q_\pi(S_t, a) + \sum_a \pi'(a|s_t) \left(q_{\pi'}(S_t, a) - q_\pi(S_t, a) \right)$$
$$= \sum_a (\pi'(a|s_t) - \pi(a|S_t)) \, v_\pi(S_t, a) + \mathbb{E}_{\pi'}[v_{\pi'}(S_{t+1}) - v_\pi(S_{t+1}) | S_t]$$

- Discounted setting shortcut

$$v_{\pi'} - v_\pi = r_{\pi'} + \gamma P^{\pi'} v_{\pi'} - r_\pi - \gamma P^\pi v_\pi = r_{\pi'} - r_\pi + \gamma \left(P^{\pi'} - P^\pi \right) v_\pi + \gamma P^{\pi'} (v_{\pi'} - v_\pi)$$
$$v_{\pi'} - v_\pi = (I - \gamma P^{\pi'})^{-1} \left(r_{\pi'} - r_\pi + \gamma \left(P^{\pi'} - P^\pi \right) v_\pi \right)$$

Approximate Policy Improvement Lemma

$$\left| J_{\mu_0}(\pi') - J_{\mu_0}(\pi) - \sum_s \sum_t \gamma^t \mathbb{P}_\pi(S_t = s) \left(\sum_a (\pi'(a|s) - \pi(a|s))\, a_\pi(s,a) \right) \right|$$

$$= \left| \sum_s \sum_t \gamma^t \left(\mathbb{P}_{\pi'}(S_t = s) - \mathbb{P}_\pi(S_t = s) \right) \left(\sum_a (\pi'(a|s) - \pi(a|s))\, a_\pi(s,a) \right) \right|$$

$$\leq \frac{2\gamma}{(1-\gamma)^2} \max_s \|\pi'(\cdot|s) - \pi(\cdot|s)\|_1^2 \max_{s,a} |a_\pi(s,a)|$$

Approximate Policy Improvement Lemma

- If $\max_s \|\pi'(\cdot|s) - \pi(\cdot|s)\|_1 \leq \epsilon$
$$\mathbb{P}_{\pi'}(S_t = s) = (1-\epsilon)^t \mathbb{P}_\pi(S_t = s) + (1 - (1-\epsilon)^t)\mathbb{P}_{\text{mistake}}(S_t = s)$$
$$\to |\mathbb{P}_{\pi'}(S_t = s) - \mathbb{P}_\pi(S_t = s)| \leq 2(1 - (1-\epsilon)^t) \leq 2\epsilon t$$
- $\sum_t 2\gamma^t t = \frac{2\gamma}{(1-\gamma)^2}$

Discounted

Approximate Policy Improvement Lemma

$$\left| J_{\mu_0}(\pi') - J_{\mu_0}(\pi) - \sum_s \sum_t \gamma^t \mathbb{P}_\pi(S_t = s) \left(\sum_a (\pi'(a|s) - \pi(a|s))\, a_\pi(s,a) \right) \right|$$

$$\leq \frac{2\gamma}{(1-\gamma)^2} \max_s \|\pi'(\cdot|s) - \pi(\cdot|s)\|_1^2 \max_{s,a} |a_\pi(s,a)|$$

Approximate Policy Improvement Lemma and Policy Gradient Theorem

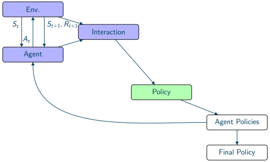
- Let $\pi' = \pi_{\theta+h}$ and π_θ
  - $\pi_{\theta+h}(a|s) - \pi_\theta(a|s) = \pi_\theta(a|s)\langle \nabla \log \pi_\theta(a|s), h\rangle + O(\|h\|^2)$
  - $\|\pi_{\theta+h}(\cdot|s) - \pi_\theta(\cdot|s)\|_1 \leq \|h\| \max_a \|\nabla \log \pi_\theta(a|s)\| + O(\|h\|^2)$
- Implies Policy Gradient Theorem:

$$J_{\mu_0}(\pi_{\theta+h})$$

$$= J_{\mu_0}(\pi_\theta) + \sum_s \sum_t \gamma^t \mathbb{P}_{\pi_\theta}(S_t = s) \left(\sum_a \pi_\theta(a|s) \langle \nabla \log \pi_\theta(s,a), h\rangle a_\pi(s,a) \right) + O(\|h\|^2)$$

Outline

Monte Carlo Approach



$$G_t = \sum_{t' \geq t} R_{t+1}$$

$$Q_{t,\pi_\theta}(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a]$$

Monte Carlo

- Replace every return by an empirical estimate along episodes.
- Need to wait until the end of the episods.

Episodic

REINFORCE: Monte Carlo Based Policy Gradient

$$J_{\mu_0}(\pi_\theta) = \sum_s \mathbb{P}(S_0 = s)\, v_{\pi_\theta}(s)$$

$$\nabla J_{\mu_0}(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\left(\sum_{t=0}^{T_\tau - 1} \nabla \log \pi_\theta(A_t|S_t)\right) G_0\right]$$

$$= \sum_s \left(\sum_t \mathbb{P}_{\pi_\theta}(S_t = s)\right)\left(\sum_a \pi_\theta(a|s) \nabla \log \pi_\theta(a|s) q_{\pi_\theta}(s,a)\right)$$

$$\widehat{\nabla} J_{\mu_0}(\pi_\theta) = \left(\sum_{t=0}^{T_\tau - 1} \nabla \log \pi_\theta(A_t|S_t)\right) G_0 \quad \text{or} \quad \widehat{\nabla} J_{\mu_0}(\pi_\theta) = \sum_t \nabla \log \pi_\theta(A_t|S_t) G_t$$

REINFORCE

- Plain MC (SGD) algorithm.
- Need to wait until the end of the episods.
- Convergence guarantees (even in off-line setting with importance sampling).

REINFORCE with Baseline

$$\nabla J_{\mu_0}(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\left(\sum_{t=0}^{T_\tau-1} \nabla \log \pi_\theta(A_t|S_t)\right)(G_0 - b)\right]$$

$$= \sum_s \left(\sum_t \mathbb{P}_{\pi_\theta}(S_t = s)\right)\left(\sum_a \pi_\theta(a|s) \nabla \log \pi_\theta(a|s)\,(q_{\pi_\theta}(s,a) - b(s))\right)$$

$$\widehat{\nabla} J_{\mu_0}(\pi_\theta) = \left(\sum_{t=0}^{T_\tau-1} \nabla \log \pi_\theta(A_t|S_t)\right)(G_0 - b)$$

or $$\widehat{\nabla} J_{\mu_0}(\pi_\theta) = \sum_t \nabla \log \pi_\theta(A_t|S_t)\,(G_t - b(S_t))$$

REINFORCE with baseline

- Several choices for b...
- and for $b(s)$ which can be any function (a crude estimate of $V_{t,\pi}(s)$ for instance)!
- Convergence guarantees (even in off-line setting with importance sampling).

Discounted REINFORCE?

$$\nabla J_{\mu_0}(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\left(\sum_{t=0}^{T_\tau - 1} \nabla \log \pi_\theta(A_t|S_t)\right)(G_0 - b)\right]$$

$$= \sum_s \left(\sum_t \gamma^t \mathbb{P}_{\pi_\theta}(S_t = s)\right)\left(\sum_a \pi_\theta(a|s) \nabla \log \pi_\theta(a|s)\left(q_{\pi_\theta}(s,a) - b(s)\right)\right)$$

$$\widehat{\nabla} J_{\mu_0}(\pi_\theta) = \left(\sum_{t=0}^{T_\tau - 1} \nabla \log \pi_\theta(A_t|S_t)\right)(G_0 - b)$$

or $$\widehat{\nabla} J_{\mu_0}(\pi_\theta) = \sum_t \gamma^t \nabla \log \pi_\theta(A_t|S_t)\left(G_t - b(S_t)\right)$$

Discounted REINFORCE

- Can be defined...
- but still requires an episodic setting for the discounted return G_t to be computed.

Discounted Measure?

$$\widehat{\nabla} J_{\mu_0}(\pi_\theta) = \sum_t \gamma^t \nabla \log \pi_\theta(A_t|S_t)\,(G_t - b(S_t))$$

$$\longrightarrow \widehat{\nabla} J_{\mu_{\pi_\theta}}(\pi_\theta) = \frac{1}{1-\gamma} \nabla \log \pi_\theta(A_t|S_t)\,(G_t - b(S_t))?$$

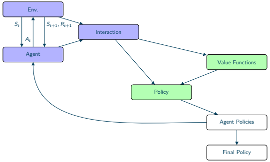
Discounted Measure?

- Much less weights for later states if μ corresponds to the initial state distribution!
- Equal weights corresponds to an averaged probability independent t, which is well defined if the initial distribution is the stationary distribution μ_{π_θ} corresponding to π_θ (it it exists).
- Approximately true after a burning stage if we reach stationarity...
- Better handled by the average return!

- More on this later...

Outline

Actor/Critic



Actor/Critic

- Actor: Parametric policy π_θ used.
- Critic: Q-value function $Q_{\boldsymbol{w}}(\cdot,\cdot)$ approximating Q_{π_θ}.
- Critic follows the Actor, which is optimized using the Critic.

- In Value Approximation, the Actor follows the Critic (through the argmax operator).
- In on-line methods, the Actor is used to interact with the environment.

Actor/Critic

$$J_(\mu_0)(\pi_\theta) = \sum_s \mu_0(s) v_{\pi_\theta}(s)$$

$$\nabla J_{\mu_0}(\pi_\theta) = \sum_s \left(\sum_t \gamma^t \mathbb{P}_{\pi_\theta}(S_t = s) \right) \left(\sum_a \pi_\theta(a|s) \nabla \log \pi_\theta(a|s) (q_{\pi_\theta}(s,a) - v_{\pi_\theta}(s,a)) \right)$$

$$\widehat{\nabla} J_{\mu_0}(\pi_\theta) = \sum_t \gamma^t \pi_\theta(A_t|S_t) \nabla \log \pi_\theta(A_t|S_t) \left(q_{\pi_\theta}(S_t, A_t) - \sum_a \pi(a|S_t) q_{\pi_\theta}(S_t, A_t) \right)$$

$$\simeq \sum_t \gamma^t \pi_\theta(A_t|S_t) \nabla \log \pi_\theta(A_t|S_t) \left(Q_{\boldsymbol{w}}(S_t, A_t) - \sum_a \pi(a|S_t) Q_{\boldsymbol{w}}(S_t, A_t) \right)$$

Actor/Critic

- Critic update: Stochastic Policy Gradient with plugin.
- Actor update: any Q-value methods estimating q_{π_θ}.
- Requires a two-scales algorithm so that $Q_{\boldsymbol{w}}$ is always a good estimate of q_{π_θ}.

- Is this a real algorithm in a non-episodic setting?

Actor/Critic

$$J_{\mu_{\pi_\theta}}(\pi_\theta) = \sum_s \mu_{\pi_\theta}(s) v_{\pi_\theta}(s)$$

$$\nabla J_{\mu_{\pi_\theta}}(\pi_\theta) = \sum_s \frac{1}{1-\gamma} \mathbb{P}_{\pi_\theta}(S_t = s) \left(\sum_a \pi_\theta(a|s) \nabla \log \pi_\theta(a|s) (q_{\pi_\theta}(s,a) - v_{\pi_\theta}(s,a)) \right)$$

$$\widehat{\nabla} J_{\mu_{\pi_\theta}}(\pi_\theta) \simeq \frac{1}{1-\gamma} \pi_\theta(A_t|S_t) \nabla \log \pi_\theta(A_t|S_t) \left(Q_{\mathbf{w}}(S_t, A_t) - \sum_a \pi(a|S_t) Q_{\mathbf{w}}(S_t, A_t) \right)$$

Actor/Critic

- Critic update: Stochastic Policy Gradient with plugin.
- Actor update: any Q-value methods estimating q_{π_θ}.
- Requires a two-scales algorithm so that $Q_{\mathbf{w}}$ is always a good estimate of q_{π_θ}.

- Require the existence of a stationary measure... and that this stationary measure is reached *quickly*.
- Much harder to do off-policy algorithm as the stationary measure is not known!

Critic in Actor/Critic

$$Q_{\boldsymbol{w}} \simeq q_{\pi_\theta}$$

Critic

- On-line TD learning with interaction following π_θ.
- Off-Policy TD learning is possible if the policy used for any action is stored.
- Approximate off-policy TD learning is possible using a replay buffer providing π_θ is changing slowly.

- May lead to 3 scales algorithm (Actor/Critic Target/Critic)
- As mentionned in the previous slide, much harder to do off-line update for the actor.

Off-Line Actor

$$J'_\mu(\pi) = \sum_s \mu(s) v_\pi(s)$$

Off-Line Actor

- Idea proposed in 2012.
- Key lemma in the paper
$$\nabla J'_\mu(\pi_\theta) \simeq \sum_s \mu(s) \sum_a \pi_\theta(a|s) \nabla \pi_\theta(a|s) q_{\pi_\theta}(s,a)$$
turns out to be wrong!
- Still used as a heuristic justification of many algorithms!
- Explicit formula for $\nabla J'_\mu(\pi_\theta)$ can be obtained but much harder to use...

Outline

PPO: Minorize-Majorization Algorithm

$$J_{\mu_0}(\pi') \geq J_{\mu_0}(\pi) + \sum_t \gamma^t \mathbb{P}_\pi(S_t = s) \left(\sum_a (\pi'(s|a) - \pi(s|a))\, a_\pi(s,a) \right) - \frac{2\gamma}{(1-\gamma)^2} \max_s \|\pi'(\cdot|s) - \pi(\cdot|s)\|_1^2 \max_{s,a} |a_\pi(s,a)|$$

Ideal Minorize-Majorization Algorithm

- At step k, find θ_{k+1} maximizing

$$J_{\mu_0}(\pi_\theta|\pi_{\theta_k}) = \sum_s \sum_t \gamma^t \mathbb{P}_{\pi_{\theta_k}}(S_t = s) \left(\sum_a (\pi_\theta(s|a) - \pi_{\theta_k}(s|a))\, a_{\pi_{\theta_k}}(s,a) \right) - \frac{2\gamma}{(1-\gamma)^2} \max_s \|\pi_\theta(\cdot|s) - \pi_{\theta_k}(\cdot|s)\|_1^2 \max_{s,a} |a_{\pi_{\theta_k}}(s,a)|$$

- By construction, $J_{\mu_0}(\pi_{\theta_{k+1}}) \geq J_{\mu_0}(\pi_{\theta_k})$

- Sample efficient algorithm as the same trajectory can be (re)used in the optimization.

PPO: Optimization

$$J_{\mu_0}(\pi_\theta) \geq J_{\mu_0}(\pi_{\theta_k}) + \sum_s \sum_t \gamma^t \mathbb{P}_{\pi_{\theta_k}}(S_t = s) \left(\sum_a (\pi_\theta(s|a) - \pi_{\theta_k}(s|a))\, a_{\pi_{\theta_k}}(s,a) \right)$$
$$- \frac{2\gamma}{(1-\gamma)^2} \max_s \|\pi_\theta(\cdot|s) - \pi_{\theta_k}(\cdot|s)\|_1^2 \max_{s,a} |a_{\pi_{\theta_k}}(s,a)|$$

Optimization

- Gradient descent is possible.
- Gradient of the first term can be approximated using a critic by
$$\sum_s \sum_t \gamma^t \mathbb{P}_\pi(S_t = s) \left(\sum_a \pi_\theta \nabla \pi_\theta(s|a) A_{\pi_{\theta_k}}(s,a) \right)$$
- Gradient of the second term more involved.

- Simpler (TRPO like) strategy: optimize
$$\sum_s \sum_t \gamma^t \mathbb{P}_{\pi_{\theta_k}}(S_t = s) \left(\sum_a (\pi_\theta(s|a) - \pi_{\theta_k}(s|a))\, a_{\pi_{\theta_k}}(s,a) \right)$$
under $\max_s \|\pi_\theta(\cdot|s) - \pi_{\theta_k}(\cdot|s)\|_1^2 \leq \epsilon$ and reduce ϵ there is no gain.

PPO: KL Relaxation

$$
\begin{aligned}
J_{\mu_0}(\pi_\theta) \geq J_{\mu_0}(\pi_{\theta_k}) &+ \sum_s \sum_t \gamma^t \mathbb{P}_{\pi_{\theta_k}}(S_t = s) \left(\sum_a (\pi_\theta(s|a) - \pi_{\theta_k}(s|a))\, a_{\pi_{\theta_k}}(s,a) \right) \\
&- \frac{2\gamma R_{\max}}{(1-\gamma)^2} \max_s \mathrm{KL}(\pi_{\theta_k}(\cdot|s), \pi_\theta(\cdot|s))
\end{aligned}
$$

TRPO/PPO Optimization

- Replace the ℓ_1 norm by a KL divergence.
- In practice, replace the max by an average and replace $\frac{2\gamma R_{\max}}{(1-\gamma)^3}$ by parameter β and replace the a_{π_k} by an estimate A_{π_k}.
- PPO: Gradient descent of the relaxed goal.
- TRPO: Constrained optimization.

- Adaptive scheme to set β.
- Can be used with continuous action.

PPO: Clipped Objective

$$\sum_s \sum_t \gamma^t \mathbb{P}_{\pi_{\theta_k}}(S_t = s) \left(\sum_a \pi_{\theta_k}(s|a) \min \left(\frac{\pi_\theta(s|a)}{\pi_{\theta_k}(s,a)} a_{\pi_{\theta_k}}(s,a), \operatorname{clip}(1-\epsilon, \frac{\pi_\theta(s|a)}{\pi_{\theta_k}(s,a)}, 1+\epsilon) a_{\pi_{\theta_k}}(s,a) \right) \right)$$

Clipped Objective

- Insight by (re)substracting $\sum_a \pi_{\theta_k}(s|a) a_{\theta_k}(s,a) = 0$:

$$\sum_a \min \left(\left(\pi_\theta(s|a) - \pi_{\theta_k}(s,a)\right) a_{\pi_{\theta_k}}(s,a), \operatorname{clip}(-\epsilon, \pi_\theta(s|a) - \pi_{\theta_k}(s,a), \epsilon) a_{\pi_{\theta_k}}(s,a) \right)$$
$$= \sum_a \operatorname{clip}(-\epsilon \pi_{\theta_k}(s,a), \pi_\theta(s|a) - \pi_{\theta_k}(s,a), \epsilon \pi_{\theta_k}(s,a)) a_{\pi_{\theta_k}}(s,a)$$
$$- \max \left(0, -(\pi_\theta(s|a) - \pi_{\theta_k}(s,a)) a_{\pi_{\theta_k}}(s,a) - \epsilon \pi_{\theta_k}(s,a) |a_{\pi_{\theta_k}}(s,a)| \right)$$

- First term amount to replace π_θ by a policy
$$\tilde{\pi}_\theta(a|s) = \operatorname{clip}(\pi_{\theta_k}(a|s)(1-\epsilon), \pi_\theta(a|s), \pi_{\theta_k}(a|s)(1+\epsilon)) + \eta_s \pi_{\theta_k}(a|s)$$
where η is so that $\tilde{\pi}$ is a probability for all s and $\|\tilde{\pi}_\theta(\cdot,s) - \pi_{\theta_k}(\cdot,s)\|_1 \leq \epsilon$
- Second term: hinge loss type penalization of policy π_θ penalizing *bad* actions.

- Very efficient for discrete actions.

Discounted

PPO: Stationary Objective

$$\sum_{s,t} \mathbb{P}_{\pi_{\theta_k}}(S_t = s) \left(\sum_a \left(\pi_\theta(s|a) - \pi_{\theta_k}(s|a)\right) a_{\pi_{\theta_k}}(s,a) \right) - \beta \max_s \mathrm{KL}(\pi_{\theta_k}(\cdot|s), \pi_\theta(\cdot|s))$$

$$\sum_{s,t} \mathbb{P}_{\pi_{\theta_k}}(S_t = s) \left(\sum_a \pi_{\theta_k}(s|a) \min\left(\frac{\pi_\theta(s|a)}{\pi_{\theta_k}(s,a)} a_{\pi_{\theta_k}}(s,a), \mathrm{clip}(1-\epsilon, \frac{\pi_\theta(s|a)}{\pi_{\theta_k}(s,a)}, 1+\epsilon) a_{\pi_{\theta_k}}(s,a) \right) \right)$$

Stationary Objective

- Amount to replace $J_{\mu_0}(\pi)$ by $J_{\mu_\pi}(\pi)$
- Most common implementation of PPO...
- But no way to justify it mathematically!
- May explain the (possible) instabilities of PPO.

- More on this later...

DPG: Deterministic Policy Gradient

$$J_{\mu_0}(\pi_\theta) = \sum_s \mu_0(s) v_{\pi_\theta}(s) \quad \text{with deterministic policy } \pi_\theta(a|s) = \mathbf{1}_{a=h_\theta(s)}$$

$$\nabla J_{\mu_0}(\pi_\theta) = \sum_s \sum_t \gamma^t \mathbb{P}_{\pi_\theta}(S_t = s) \nabla_a q(S_t, h_\theta(S_t)) \nabla h_\theta(S_t)$$

Deterministic Policy Gradient

- Deterministic policy replaced by a randomized one centered on $h_{\theta(s)}$ in the interactions!.
- Critic trained with a TD variant of DQN.
- Same formula by using a policy $\pi_\theta = \mathrm{N}(h_\theta(s), \sigma^2 \mathrm{Id})$ and letting σ goes to 0.
- Off-Policy as claimed?
- Yes for the actor but no theoretical justification for the critic!
- In practice, the buffer contains only samples using a policy close to the current one...

SAC: A New Goal

$$R_t \to R_t + \lambda \mathcal{H}(\pi(S_t))$$

A Modified Reward

- Modification of the reward to favor high entropy policy:
$$R_t \to R_t + \lambda \mathcal{H}(\pi(S_t))$$
- Goal:
$$J(\pi) = \mathbb{E}_\pi \left[\sum_t \gamma^t \left(R_t + \lambda \mathcal{H}(\pi(S_t)) \right) \right]$$
- Soft value function implicitly defined as the fixed point of
$$\mathcal{T}^\pi q_\pi(s,a) = r_\pi(s,a) + \gamma \sum_{s'} p(s'|s,a) v_\pi(s')$$
$$\text{where} \quad v_\pi(s,a) = \sum_a \pi(a|s) \left(q_\pi(s,a) - \log \pi(a|s) \right)$$

Discounted

SAC: Policy Improvement and Optimal Policy

$$R_t \to R_t + \lambda \mathcal{H}(\pi(S_t))$$

A Modified Policy Improvement Lemma

- Policy improvement rule:
$$\pi^+(\cdot|s) = \operatorname*{argmax}_{\pi(\cdot|s)} \sum_a \pi(a|s)\left(q(s,a) - \lambda \log(\pi(a|s))\right)$$
$$\pi^+(a|s) \propto \exp(-\frac{1}{\lambda} q(s,a))$$
implies $G_{\pi^+}(s,a) \geq G_\pi(s,a)$.
- At convergence, $J(\pi^\star)$ is optimal!
- Convergence in the finite setting.

SAC: Parametrization

$$\pi \sim \pi_\theta \quad \text{and} \quad q(s,a) \sim Q_{\boldsymbol{w}}$$

SAC Choices

- Fitted TD learning for Q:
$$\boldsymbol{w} \simeq \operatorname{argmin} \sum_{(S,A,R,S')\in\mathcal{B}} \left(R + \mathbb{E}_{\pi_\theta}\left[\gamma Q_{\overline{\boldsymbol{w}}}(S',a) - \lambda \log \pi_\theta(a|S')\right] - Q_{\boldsymbol{w}}(S,A)\right)^2$$
where the trajectory pieces are samples from a replay buffer and $\overline{\boldsymbol{w}}$ is a slowdown version of $\boldsymbol{w}$ (two-scales algorithm).
- Online version rather than batch...
- Fitted KL for π:
$$\theta \simeq \operatorname{argmin} \sum_{(S,A,R,S')\in\mathcal{B}} \mathrm{KL}(\pi_\theta(\cdot|S)|\exp -\lambda Q_{[\overline{\boldsymbol{w}}]}(S,\dot{})/Z_{\overline{\boldsymbol{w}}}(S))$$
$$\simeq \sum_{(S,A,R,S')\in\mathcal{B}} \mathbb{E}_{\pi_\theta}\left[\frac{1}{\lambda}\log \pi_\theta(a|S) - Q_\theta(a|s)\right]$$

Outline

Outline

Total Reward

$$v_\Pi(s) = \mathbb{E}_\Pi\left[\sum_{t'=1}^{+\infty} R_{t+1} \middle| S_0 = s\right]$$

$$= \underbrace{\mathbb{E}_\Pi\left[\sum_{t'=1}^{+\infty} \max(0, R_{t+1}) \middle| S_t = s\right]}_{v_{+,\Pi}(s)} - \underbrace{\mathbb{E}_\Pi\left[\sum_{t'=t+1}^{+\infty} \max(0, -R_{t+1}) \middle| S_t = s\right]}_{v_{-,\Pi}(s)}$$

- Total reward not necessarily well defined!
- Need to **assume** this is the case!

Classical Assumptions

- Episodic model: $\forall \Pi, s,\ \mathbb{E}_\Pi\left[\min_{t, \forall t' \geq t, R_{t'}=0} t \middle| S_0 = s\right] \leq H < +\infty$
- Stochastic Shortest Path: $\exists \Pi, \forall s,\ \mathbb{E}_\Pi\left[\min_{t, \forall t' \geq t, R_{t'}=0} t \middle| S_0 = s\right] \leq H < +\infty.$
- More general assumption: $\forall \Pi, s$ either $v_{+,\Pi}(s)$ or $v_\Pi(s)$ is finite.

Bellman Operator and Optimality Equation

$$\sup_{\Pi} v_{\Pi}(s) = v_\star(s) = \underbrace{\max_a r(s,a) + \sum_{s'} p(s'|s,a) v_\star(s')}_{\mathcal{T}^\star(v_\star)(s)}$$

- Similar to the discounted setting as:
  - We can focus on Markovian policy.
  - The optimal value $v_\star$ satisfies the Bellman optimality equation.

But. . .

- $\mathcal{T}^\star$ is not a contraction and thus there may be several solutions of the equation.
- If π is such that $\mathcal{T}^\pi v_\star = \mathcal{T}^\star v_\star$, we need to assume that $\limsup (P^\pi)^n v_\star(s) \leq 0$ to prove that $\Pi = (\pi, \pi, \ldots)$ is optimal.
- There may not exist an optimal policy!

- Existence of optimal policies in the finite state-action setting by defining the total reward to the limit of discounted setting when $\gamma \to 1$ and using the finiteness of the policy set. . .

Stochastic Shortest Path

$$\forall s,\ \mathbb{E}_\Pi\left[\min_{t, \forall t' \geq t, R_{t'}=0} t \middle| S_0 = s\right] \leq H < +\infty$$

- A policy is said to be H-proper if it satisfies this property.

Extended Stochastic Shortest Path

- Assumptions:
  - It exists a proper policy.
  - For any improper policy, it exists s such that $v_\Pi(s) = -\infty$.
- Results:
  - $v_\star$ is the unique solution of $v = \mathcal{T}^\star v$.
  - Value Iteration converges and Policy Iteration converges provided $v_0 \leq \mathcal{T}^\star v_0$ (or finite setting).
  - If all stationary policies are proper then $\mathcal{T}^\star$ is a contraction for a weighted sup-norm.

- Any discounted model can be put in this framework by adding an absorbing state reached at random at each step with probability $1-\gamma$ and $H = 1/(1-\gamma)$.

$$\delta_t = R_t + Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$$

Prediction

- Convergence of TD-learning algorithms for any proper policy.

$$\delta_t = R_t + \max_Q(S_{t+1}, a) - Q(S_t, A_t)$$

Planning

- Convergence of Q-learning algorithms is the Stochastic Shortest Path setting (It exists a proper policy and for any improper policy, it exists s such that $v_\Pi(s) = -\infty$) if the Q estimates remain bounded.

- See *Neuro-Dynamic Programming* from Bertsekas and Tsitsiklis!
- May be very slow in practice!

Stochastic Shortest Path and Policy Gradient

$$
\begin{aligned}
\nabla v_{\pi_\theta}(s) &= \sum_{t'} \mathbb{E}_{\pi_\theta}[\nabla \log \pi_\theta(A_{t'}|S_{t'}) a_{\pi_\theta}(S_{t'}, A_{t'}) | S_0 = s] \\
&= \sum_s \left(\sum_t \mathbb{P}_{\pi_\theta}(S_t = s | S_0 = s) \right) \left(\sum_a \pi_\theta(a|s) \nabla \log \pi_\theta(a|s) q_{\pi_\theta}(s, a) \right)
\end{aligned}
$$

Policy Gradient

- Formula valid in the Stochastic Shortest Path Assumption (if the current policy is proper).
- Approximate Policy Improvement Lemma with a H^2 multiplicative constant (instead of $O(H)$).

Actor-Critic

- Valid approach provided all the policies considered remain propers.
- Main difficulty is to maintain a good estimate of $q_{\pi_\theta}\ldots$

Positive Bounded and Negative Models

Positive Bounded Models

- $\forall \Pi, s, \ v_{+,\Pi}(s) < \infty$
- $\forall s, \exists a, r(s,a) \geq 0$

- Often stronger assumption: $r(s,a) \geq 0$.
- Any discounted model can be put in this framework by adding an absorbing state reached at random at each step with probability $1-\gamma$.

Negative Models

- $\forall \Pi, s, \ v_{+,\Pi}(s) = 0$ and $v_{-,\Pi}(s) < \infty$
- There exists a policy Π such that $\forall s, v_\Pi(s) > -\infty$

- Maximization of v_Π amounts to the minimization of $v_{-,\Pi}$ and the negative reward can be interpreted as the opposite of costs.
- Classical Stochastic Shortest Path within this framework.
- See *Markov Decision Processes. Discrete Stochastic Dynamic Programming* from Puterman.

Positive Bounded and Negative Models Results

Result	Positive Bounded Models	Negative Models
Optimality equation	$v^\star$ is a minimal solution within $v \leq \mathcal{T}^\star v$	$v^\star$ is a maximal solution within $v \geq \mathcal{T}^\star v$
$\mathcal{T}^\pi v_\star = \mathcal{T}^\star v_\star \Rightarrow \pi$ optimal	Only if $\limsup (P^\pi)^n v_\star(s) = 0$	Always
Existence of optimal stationary policy	S and A finite or existence of optimal policy and $r \geq 0$	A_s finite or A_s compact, r and *p continuous* with respect to a.
Existence of stationary ϵ-optimal policy	If $v^\star$ is bounded	Not always (Always for non stationary policy)
Value Iteration converges	$0 \leq v_0 \leq v_\star$	$0 \geq v_0 \geq v_\star$ and A_s finite or S finite if $v_\star > -\infty$
Policy Iteration converges	Yes	Not always
Modified Policy Iteration converges	$0 \leq v_0 \leq v_\star$ and $v_0 \leq \mathcal{T}^\star v_0$	Not always
Solution by linear programming	Yes	No

- No RL analysis?

Outline

Average Return

$$\overline{v}_{\Pi}(s) = \lim_{T\to\infty} \frac{1}{T} v_{T,\Pi}(s) = \lim_{T\to\infty} \frac{1}{T} \mathbb{E}_{\Pi}\left[\sum_{t=1}^{T} R_t \middle| S_0 = s\right]$$

$$\longrightarrow \overline{v}_{+,\Pi}(s) = \limsup_{T\to\infty} \frac{1}{T} v_{T,\Pi}(s)$$

$$\overline{v}_{-,\Pi}(s) = \liminf_{T\to\infty} \frac{1}{T} v_{T,\Pi}(s)$$

Average Return(s)

- Limit $\overline{v}_{\Pi}$ may not be defined!
- **Prop:** $\overline{v}_{\Pi}$ is well defined if Π is stationary and $\frac{1}{T}\sum_{t=1}^{T}(P^{\pi})^{t-1}$ tends to a stochastic matrix.
- Limits $\overline{v}_{+,\Pi}$ and $\overline{v}_{-,\Pi}$ always defined!

Average Returns and Optimality

$$\overline{v}_{+,\star}(s) = \sup_{\Pi} \overline{v}_{+,\Pi}(s) \quad \text{and} \quad \overline{v}_{-,\star}(s) = \sup_{\Pi} \overline{v}_{-,\Pi}(s)$$

Optimality of $\Pi_\star$

- Average optimal:
$$\overline{v}_{-,\Pi_\star} \geq \overline{v}_{+,\star}(s)$$
- Lim-sup average optimal (best case analysis):
$$\overline{v}_{+,\Pi_\star} \geq \overline{v}_{+,\star}(s)$$
- Lim-inf average optimal (worst case analysis):
$$\overline{v}_{-,\Pi_\star} \geq \overline{v}_{-,\star}(s)$$

- More complex setting!
- Let's start with Prediction...

Prediction for a Stationary Markov Policy

$$\overline{v}_\Pi(s) = \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} P_\pi^{t-1} r_\pi = \left(\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} P_\pi^{t-1}\right) r_\pi = P_\pi^\infty r_\pi$$

Stochastic Matrix P_π^∞

- Measures the average amount of time spend on a state s' starting from state s at $t = 0$ when using policy π.
- Structure linked to the properties of the resulting Markov chain:
  - If aperiodic, $P_\pi^\infty = \lim_T P_\pi^T$ i.e. P_π^∞ is close to the probability of reaching s' from s at any large T.
  - If unichain, then P_π^∞ has identical rows and corresponds to the stationary distribution.
  - If multichhain, then P_π^∞ has a diagonal block structure with rows equal withing each block corresponding to the stationary distribution in each chain.

- Implies that $\overline{v}_\Pi(s) = \overline{v}_\Pi(s')$ in the Markov process is unichain.
- Limit P_π^∞ may be hard to compute. . .

Average Reward and Relative Value Functions

$$U_\pi(s) = \mathbb{E}_\pi\left[\sum_{t=1}^{\infty}(R_t - \overline{v}_\pi(S_t))\middle| S_0 = s\right] \quad \Leftrightarrow U_\pi = \underbrace{(\mathrm{Id} - P_\pi + P_\pi^\infty)^{-1}(\mathrm{Id} - P_\pi^\infty)}_{H_\pi} r_\pi$$

Link between U_π and $\overline{v}_\pi$

- $(\mathrm{Id} - P_\pi)\overline{v}_\pi = 0$
- $\overline{v}_\pi + (I - P_\pi)U_\pi = r_\pi$

Characterization by a system

- If $(\mathrm{Id} - P_\pi)\overline{v} = 0$ and $\overline{v} + (I - P_\pi)U = r_\pi$ then
  - $\overline{v} = \overline{v}_\pi$,
  - $U = U_\pi + u$ with $(I - P_\pi)u = 0$,
  - If $P_\pi^\infty U = 0$ then $u = 0$.

- Prediction possible by solving this system as we do not need U_π.

Optimality Equations

$$\overline{v}(s) = \max_a \sum_{s'} p(s'|s,a)\overline{v}(s')$$

$$U(s) + \overline{v}(s) = \max_{a \in B_s} r(s,a) + \sum_{s'} p(s'|s,a)U(s) \text{with } B_s = \{a | \sum_{s'} p(s'|s,a)\overline{v}(s') = \overline{v}(s)\}$$

$$\pi_\star(s) \in \operatorname{argmax}_{a \in B_s} r(s,a) + \sum_{s'} p(s'|s,a)U(s)$$

Existence

- If there is a solution $(\overline{v}, U)$ of the system then $\overline{v} = \overline{v}_\star$ and $\pi_\star$ is an optimal policy.
- There may exist other optimal policies not satisfying the argmax property.
- There may not exist solutions to the system.

- Associated relative value iteration and modified policy iteration can be defined.
- Convergence under strong assumptions. . .

Average Return and Relative Value Functions

$$r(\pi) = \lim_T \mathbb{E}_\pi \left[\frac{1}{T} \sum_{t=0}^{T-1} R_t \right] = \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) r$$

$$G_t = \sum_{t' \geq t} (R_t - r(\pi))$$

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] \quad \text{and} \quad q_\pi(s,a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$$

Connection with Stochastic Shortest Path

- Provided there is a state s that is visited with positive probability in the first m steps for any starting state and any policy.
- $r(\pi)$ is the average cost between a visit s and the next one...

Reinforcement Learning Algorithms

- Simultaneous estimation of q and r...
- Much less theory as there is no contraction!

Algorithm(s)

Average: Planning by SARSA

input: MDP environment, initial state distribution μ_0, policy Π and discount factor γ
parameter: Number of step T
init: $\forall s, a, Q(s,a), N(s,a) = 0$, n=0, $t = 0$, $r = 0$
Pick initial state S_0 following μ_0
repeat
- $N(S_t) \leftarrow N(S_t) + 1$
- Pick action A_t according to $\pi(\cdot|S_t)$
- $Q(S_{t-1}, A_{t-1}) \leftarrow Q(S_{t-1}, A_{t-1}) + \alpha(N(S_{t-1}, A_{t-1}))\,(R_t - r_{t-1} + \gamma Q(S_t, A_t) - Q(S_{t-1}, A_{t-1}))$
- $r \leftarrow r + \alpha_t(R_t - r)$
- $\Pi(S_{t-1}) = \text{argmax}_a\, Q(S_{t-1}, a)$ (plus exploration)
- $t \leftarrow t + 1$

until $t == T$
output: Deterministic policy $\tilde{\pi}(s) = \text{argmax}_a\, Q(s,a)$

- Q-learning variant (known as R-learning) and other estimations of r exist.
- No convergence proof.

Policy Gradient

$$\nabla r(\pi) = \lim_T \frac{1}{T} \mathbb{E}_\pi \left[\sum_{i=1}^{T} \nabla \log \pi(A_t | S_t) q_\pi(S_t, A_t) \right]$$

$$\nabla r(\pi) = \lim_T \frac{1}{T} \mathbb{E}_\pi \left[\sum_{i=1}^{T} \nabla \log \pi(A_t | S_t) a_\pi(S_t, A_t) \right]$$

Policy Gradient

- REINFORCE type algorithms, using MC estimate of q and a are possible,
- but q and a are the relative ones, not the classical ones, and are much harder to estimate.

- Actor/Critic algorithms combining parametric estimation of q (or a) and gradient exist.

Outline

To Discount or Not?

To Discount: $J(\pi) = \mathbb{E}_\pi \left[\sum_t \rho^t R_t \right]$ $\quad Q_\pi(s,a) = \mathbb{E}_\pi \left[\sum_t \rho^t R_t \middle| s_0 = s, a_0 = a \right]$

or Not (SSP): $J(\pi) = \mathbb{E}_\pi \left[\sum_t R_t \right]$ $\quad Q_\pi(s,a) = \mathbb{E}_\pi \left[\sum_t R_t \middle| s_0 = s, a_0 = a \right]$

To Discount or Not? Open Question!

- Discount is (quite) artificial.
- No discount in the evaluation part most of the time.
- Discount often used in training due to better convergence for value functions. . . toward a (quite) artificial policy target!

- In practice, often hybrid scheme with no discount for the policy gradient part, but discount for the value functions part! No strong justification but often better numerical performance!
- Average reward much less used!

Outline

POMDP

$$o \sim \mathbb{P}(\cdot|s, a)$$

Partially Observed Markov Decision Process

- MDP strongest assumption is that s is observed!
- POMDP replaces this assumption by the observation of o with a known law of $\mathbb{P}(o|s, a)$.
- Can be recasted as a MDP where the state is the probability of being in a state s given the current observation!
- Much higher dimensional setting!

- Policy gradient algorithms remain valid in the POMDP setting when replacing s with o.
- Difficult part is to obtain a good value function estimate.

Outline

Imitation Learning

$$\text{Good } S_t, A_t, (R_{t+1},)S_{t+1}, A_{t+1} \rightarrow \pi$$

$$\operatorname*{argmin}_\theta \sum_{i=1}^{t} \log \pi_\theta(A_t|S_t)$$

Imitation Learning

- Learn policy from demonstrations (observations).
- Most classical approach: maximum likelihood.
- Need to cover all states (possibly through the approximation)
- Reward is not used.

- DAGGER: Sequential approach to add feedback from trajectory with an estimated policy through the decision that would have been made.

Inverse Reinforcement Learning

$$\text{Good } S_t, A_t, S_{t+1}, A_{t+1} \text{ or } \pi \to R \to \pi^\star$$

Inverse Reinforcement Learning

- **Heuristic:** Learn a reward which **explains** the observed policy and used it to obtain a better policy (or to generalize to different models).
- No clear mathematical formulation:
  - Reward so that the observed policy is optimal (with a margin).
  - Expected return/optimal value function linked to observed policy (trajectories) probability (with entropic regularization)
  - Most generic formulation?

$$\min_{\pi'} \max_{R} \mathbb{E}_\pi[R] - \mathbb{E}_{\pi'}[[]\,R] + K(\pi') - C(R)$$

- Exact problem considered not always clear for a given algorithm (and different from one algorithm to another)!

- Very hard problem!

Learning from Preferences

$$S_t, A_t, S_{t+1}, A_{t+1} \text{ vs } S_t, A'_t, S'_{t+1}, A'_{t+1} \rightarrow R \rightarrow \pi^\star$$

Learning from Preferences

- Often easier to compare trajectories than to make a demonstration.
- **Reinforcement Learning from Human Feedback**: Learn a reward from the demonstration using a preference model (Bradley-Terry?) and use it to find a policy.
- **Direct Policy Optimization**: shortcut to optimize directly the policy thanks to the explicit preference model used.
- Proximity constrains are often added to avoid moving too fast from a current policy.

- Key to the performances of current LLMs.

LLM: Large Language Model

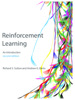
Outline

More!

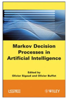
- Regrets
- Sample optimality
- Robustness
- Multi-agents (Games. . .)
- LLM and world models. . .

Outline

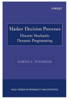
References



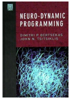
R. Sutton and A. Barto.
Reinforcement Learning, an Introduction (2nd ed.)
MIT Press, 2018



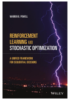
O. Sigaud and O. Buffet.
Markov Decision Processes in Artificial Intelligence.
Wiley, 2010



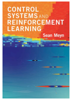
M. Puterman.
Markov Decision Processes. Discrete Stochastic Dynamic Programming.
Wiley, 2005



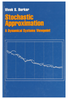
D. Bertsekas and J. Tsitsiklis.
Neuro-Dynamic Programming.
Athena Scientific, 1996



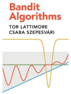
W. Powell.
Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions.
Wiley, 2022



S. Meyn.
Control Systems and Reinforcement Learning.
Cambridge University Press, 2022



V. Borkar.
Stochastic Approximation: A Dynamical Systems Viewpoint.
Springer, 2008



T. Lattimore and Cs. Szepesvári.
Bandit Algorithms.
Cambridge University Press, 2020

Licence and Contributors

Contributors

- Main contributor: E. Le Pennec
- Contributors: S. Boucheron, A. Dieuleveut, A.K. Fermin, S. Gadat, S. Gaiffas, A. Guilloux, Ch. Keribin, E. Matzner, M. Sangnier, E. Scornet.