Reinforcement Learning
Reinforcement Learning: Advanced Techniques in the Tabular Setting

E. Le Pennec

M2DS - Reinforcement Learning – Fall 2023
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
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 iterative 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 deducted by a statewise optimization of the value function over the actions.
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).
Outline

1. *n*-step Algorithms
2. Eligibility Traces
3. Off-policy vs on-policy
4. Bandits
5. Model Based Approach
6. Replay Buffer and Prioritized Sweeping
7. Real Time Planning
8. References
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

1. \textit{n-step Algorithms}
2. Eligibility Traces
3. Off-policy vs on-policy
4. Bandits
5. Model Based Approach
6. Replay Buffer and Prioritized Sweeping
7. Real Time Planning
8. References
How many steps before backup?

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

$$\left( \mathcal{T}^n \right)^n \nu(s) = \mathbb{E}_n \left[ \frac{R_{t+1} + \gamma R_{t+2} + \gamma^{n-1} R_{t+n} + \gamma^n \nu(S_{t+n})}{G_{t:t+n}} \right]_{S_t = s}$$

- Family of stochastic approximation algorithms:

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

**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 \( \Pi \)...i.e. SARSA!

- Best convergence often for intermediate \( n \).
- No proof beside TD for \( n > 1 \)!
**n-steps TD**

### Discounted: Prediction by n-steps TD

**input:** MDP environment, initial state distribution $\mu_0$, policy $\Pi$ and discount factor $\gamma$

**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 $\mu_0$

  **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

The policy $\Pi$ is known so that we can use it in a formula:

$$R_t + \gamma Q(S_t, A_t) \rightarrow R_t + \gamma \sum_a \pi(a|S_t) Q(S_t, a)$$

- 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 $\mu_0$, policy $\Pi$ and discount factor $\gamma$

**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 $\mu_0$

**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 $Q$
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:

$$G_{t:t+2} = R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1})Q_{t+1}(S_{t+1}, a)$$

$$+ \gamma \pi(A_{t+1}|S_{t+1}) \left( R_{t+2} + \gamma \sum_a \pi(a|S_{t+2})Q(S_{t+2}, a) \right)$$

$$= R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1})Q(S_{t+1}, a) + \gamma \pi(A_{t+1}|S_{t+1})G_{t+1:t+2}$$
\( n\)-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:
  \[
  G_{t:t+2} = R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1})Q(S_{t+1}, a) + \gamma \pi(A_{t+1}|S_{t+1})G_{t+1:t+2}
  \]
  \[
  = R_{t+1} + \gamma \sum_a \pi(a|S_{t+1})Q(S_{t+1}, a) + \gamma \pi(A_{t+1}|S_{t+1}) (G_{t+1:t+2} - Q(S_{t+1}, A_{t+1}))
  \]

- Recursive definition of \( n\)-step return:
  \[
  G_{t:t+n} = R_{t+1} + \gamma \sum_a \pi(a|S_{t+1})Q(S_{t+1}, a)
  \]
  \[
  + \gamma \pi(A_{t+1}|S_{t+1}) (G_{t+1:t+n} - Q(S_{t+1}, A_{t+1}))
  \]

- TD update
  \[
  Q(S_{t-n}, A_{t-n}) = Q(S_{t-n}, A_{t-n}) + \alpha (N(S_{t-n}, Q_{t-n})) (G_{t-n:t} - Q(S_{t-n}, A_{t-n}))
  \]
Unifying algorithm!

Recursive definition of \( n \)-step return:

\[
G_{t:t+n} = R_{t+1} + \sigma G_{t+1:t+n} \\
+ (1 - \sigma) \left( \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) \\
+ \gamma \pi(A_{t+1}|S_{t+1}) (G_{t+1:t+n} - Q(S_{t+1}, A_{t+1})) \right)
\]
**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)**
  \[ G_t^\omega = \sum_{n=1}^{\infty} \omega_n G_{t:t+n} \text{ with } \sum_{i=1}^{\infty} \omega_n = 1 \]

- **TD(\lambda): specific averaging**
  \[ G_t^\lambda = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_{t:t+n} \]
  \[ = (1 - \lambda) \sum_{n=1}^{T-t} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t} G_t \text{ (Episodic)} \]

  interpolating between TD (a.k.a TD(0)) and MC for \( \lambda = 1 \).

- Can be mixed with tree backup strategies (TB(\lambda))
**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.
$\lambda$-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 $G_\lambda$ 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

1. *n*-step Algorithms
2. Eligibility Traces
3. Off-policy vs on-policy
4. Bandits
5. Model Based Approach
6. Replay Buffer and Prioritized Sweeping
7. Real Time Planning
8. References
Forward and Backward Point of View

From a forward view

To a backward one:
Eligibility Traces

Returns and Temporal Differences

- **n-step returns:**
  \[
  G_{t:t+n} - Q(S_t, A_t) = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} \\
  + \gamma^n Q(S_{t+n}, A_{t+n}) - Q(S_t, A_t)
  \]
  \[
  = \sum_{l=1}^{n} \gamma^{l-1}(R_{t+l} + \gamma Q(S_{t+l}, A_{t+l}) - Q(S_{t+l-1}, A_{t+l-1}))
  \]
  \[
  = \sum_{l=0}^{n-1} \gamma^{l-1} \delta_{t+l}
  \]

- **\(\lambda\) return:**
  \[
  G_{t}^{\lambda} - Q(S_t, A_t) = (1 - \lambda) \sum_{n} \lambda^n(G_{t:t+n} - Q(S_t, A_t))
  \]
  \[
  = \sum_{n=0} G_{t}^{\lambda} \gamma^n \delta_{t+n}
  \]
# Forward View and Backward View

## Forward View

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

- **Cumulative updates:**
  \[
  Q_t(s, a) = Q_0(s, a) + \sum_{t' \leq t} 1_{(s, a) = (S_{t'}, A_{t'})} \alpha_{t'}(s, a) \left( \sum_{t'' \geq t'} \lambda^{t''-t'} \gamma^{t''-t'} \delta_{t''} \right)
  \]

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

- **Focus on the update place.**
Forward View and Backward View

Limit(s)

Limit:

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

\[ = Q_0(s, a) + \sum_{t''} \delta_{t''} \sum_{t' \leq t''} 1(s, a) = (S_{t'}, A_{t'}) \alpha_{t'}(s, a) \lambda^{t'' - t'} \gamma^{t'' - t'} \]

Focus on the update place or and the temporal differences...
### Backward View

- **Same limit with cumulative udpates over temporal differencies**

  \[
  Q_t(s, a) = Q_0(s, a) + \sum_{t'' \leq t} \delta_t \sum_{t' \leq t''} 1(s, a) = (S_{t'}, A_{t'}) \alpha_{t'}(s, a) \lambda^{t''-t'} \gamma^{t''-t'}
  \]

- **Updates**

  \[
  Q_t(s, a) = Q_{t-1}(s, a) + \delta_t \sum_{t' \leq t} 1(s, a) = (S_{t'}, A_{t'}) \alpha_{t'}(s, a) \lambda^{t-t'} \gamma^{t-t'}
  \]

- **Pseudo Eligibility trace:**

  \[
  z_t(s, a) = \sum_{t' \leq t} 1(s, a) = (S_{t'}, A_{t'}) \alpha_{t'}(s, a) \lambda^{t-t'} \gamma^{t-t'}
  \]

  \[
  = \lambda \gamma z_{t-1}(s, a) + \alpha_t(s, a) 1(s, a) = (S_t, A_t)
  \]

- **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 differences with simultaneous update on all states.
- TD(\( \lambda \)) eligibility trace: \( z_t(s, a) = \lambda \gamma z_{t-1}(s, a) + 1(s, a) = (S_t, A_t) \)
- 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:
    \[
    z_t(s, a) = c_t \gamma z_{t-1}(s, a) + 1(s, a) = (S_t, A_t)
    \]
    where \( c_t \) is defined in an 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:**
  \[ \delta_t = R_{t+1} + \gamma V(S_{t+1}) - Q(S_t, A_t) \]
  \[ = R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \]

- **Average of both:**
  \[ \delta_t = R_{t+1} + \gamma \sigma Q(S_{t+1}, A_{t+1}) + \gamma (1 - \sigma) V(S_{t+1}) - Q(S_t, A_t) \]
  \[ = R_{t+1} + \gamma V(S_{t+1}) + \gamma \sigma (Q(S_{t+1}, A_{t+1}) - V(S_{t+1})) - Q(S_t, A_t) \]

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

1. n-step Algorithms
2. Eligibility Traces
3. Off-policy vs on-policy
4. Bandits
5. Model Based Approach
6. Replay Buffer and Prioritized Sweeping
7. Real Time Planning
8. References
On-Policy vs Off-Policy

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

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

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 $p(x) = 0$.

- $\mathbb{V}ar_q \left[ \frac{p(x)}{q(x)} g(x) \right]$ may be large with respect to $\mathbb{V}ar_p \left[ g(x) \right]$ if the ratio $p(x)/q(x)$ is large...

Importance Sampling for Trajectories

- For any trajectory $\tau_{t:t'} = S_t, A_t, R_{t+1}, S_{t+1}, \ldots, R_{t'}, S_{t'}, A_{t'}(, R_{t'+1}, S_{t'+1})$,

$$\mathbb{P}_\pi(S_t, A_t, R_{t+1}, S_{t+1}, \ldots, R_{t'}, S_{t'}, A_{t'}(, R_{t'+1}, S_{t'+1}) | S_t) \equiv \pi(A_t | S_t) \ldots \pi(A_{t'} | S_{t'})$$

$$\mathbb{P}_b(S_t, A_t, R_{t+1}, S_{t+1}, \ldots, R_{t'}, S_{t'}, A_{t'}(, R_{t'+1}, S_{t'+1}) | S_t) \equiv b(A_t | S_t) \ldots b(A_{t'} | S_{t'})$$
Importance Sampling and Returns

\[ E_\pi[g(\tau_{t:t'})|S_t = s] = E_b[\rho_{t:t'}g(\tau_{t:t'})|S_t = s] \quad \text{with} \quad \rho_{t:t'} = \frac{\pi(A_t|S_t) \cdots \pi(A_{t'}|S_{t'})}{b(A_t|S_t) \cdots b(A_{t'}|S_{t'})} \]

From \( b \) to \( \Pi \)

- **Returns:**

\[
E_\pi[G_{t:t'}|S_t = s] = E_\pi \left[ \sum_{t''=t+1}^{t'} \gamma^{t''-t-1} R_{t''} + \gamma^{t'-t} V(S_{t'}) \bigg| S_t = s \right] \\
= E_b \left[ \rho_{t:(t'-1)} \left( \sum_{t''=t+1}^{t'} \gamma^{t''-t-1} R_{t''} + \gamma^{t'-t} V(S_{t'}) \right) \bigg| S_t = s \right] \\
= E_b \left[ \sum_{t''=t+1}^{t'} \rho_{t:(t''-1)} \gamma^{t''-t-1} R_{t''} + \rho_{t:(t'-1)} \gamma^{t'-t} V(S_{t'}) \bigg| S_t = s \right]
\]
Importance Sampling and Returns

\[ E_{\Pi}[g(\tau_{t:t'})|S_t, A_t] = E_b \left[ \rho_{(t+1):t'} g(\tau_{t:t'}) | S_t, A_t \right] \quad \text{with} \quad \rho_{t:t'} = \frac{\pi(A_t|S_t) \ldots \pi(A_{t'}|S_{t'})}{b(A_t|S_t) \ldots b(A_{t'}|S_{t'})} \]

**From \( b \) to \( \Pi \)**

- **Returns:**
  \[
  E_{\pi}[G_{t:t'}|S_t, A_t] = E_{\pi} \left[ \sum_{t''=t+1}^{t'} \gamma^{t''-t-1} R_{t''} + \gamma^{t'-t} Q(S_{t'}, A_{t'}) \right| S_t, A_t \]
  \[
  = E_b \left[ \rho_{(t+1):(t'-1)} \left( \sum_{t''=t+1}^{t'} \gamma^{t''-t-1} R_{t''} + \gamma^{t'-t} Q(S_{t'}, A_{t'}) \right) \right| S_t, A_t \]
  \[
  = E_b \left[ \rho_{(t+1):(t''-1)} \sum_{t''=t+1}^{t'} \gamma^{t''-t-1} R_{t''} + \rho_{(t+1):t'} \gamma^{t'-t} Q(S_{t'}, A_{t'}) \right| S_t, A_t \]

- No correction if \( t' = t + 1 \)
\textbf{\(\lambda\)-return}

- **Recursive definition of the \(\lambda\)-return:**

  \[
  G^\lambda_t|S_t = R_{t+1} + \gamma \left( (1 - \lambda) V(S_{t+1}) + \lambda G^\lambda_{t+1} \right)
  \]

  \[
  G^\lambda_t|S_t, A_t = R_{t+1} + \gamma \left( (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^\lambda_{t+1} - Q(S_{t+1}, A_{t+1}) \right)) + \lambda G^\lambda_{t+1} \right)
  \]

- **Off-line correction**

  \[
  G^\lambda_t|S_t = \rho_{t:t} \left( R_{t+1} + \gamma \left( (1 - \lambda) V(S_{t+1}) + \lambda G^\lambda_{t+1} \right) \right)
  \]

  \[
  G^\lambda_t|S_t, A_t = R_{t+1} + \gamma \left( (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^\lambda_{t+1} - Q(S_{t+1}, A_{t+1}) \right)) + \lambda \rho_{t+1:t+1} G^\lambda_{t+1} \right)
  \]

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

### $\delta_t$?

#### 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 $\pi$.

#### Expected temporal differencies:

\[ \delta_t = R_{t+1} + \gamma V(S_{t+1}) - Q(S_t, A_t) \]

\[ = R_{t+1} + \gamma \sum_a \pi(a | S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \]

without any correction.

#### Average of both:

\[ \delta_t = R_{t+1} + \gamma \sigma Q(S_{t+1}, A_{t+1}) + \gamma (1 - \sigma) V(S_{t+1}) - Q(S_t, A_t) \]

\[ = R_{t+1} + \gamma V(S_{t+1}) + \gamma \sigma (Q(S_{t+1}, A'_{t+1}) - V(S_{t+1})) - Q(S_t, A_t) \]

with $A'_{t+1}$ drawn using $\pi$. 

---

**Discounted**

### 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 $\pi$.

- **Expected temporal differencies:**
  \[ \delta_t = R_{t+1} + \gamma V(S_{t+1}) - Q(S_t, A_t) \]

  \[ = R_{t+1} + \gamma \sum_a \pi(a | S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \]

  without any correction.

- **Average of both:**
  \[ \delta_t = R_{t+1} + \gamma \sigma Q(S_{t+1}, A_{t+1}) + \gamma (1 - \sigma) V(S_{t+1}) - Q(S_t, A_t) \]

  \[ = R_{t+1} + \gamma V(S_{t+1}) + \gamma \sigma (Q(S_{t+1}, A'_{t+1}) - V(S_{t+1})) - Q(S_t, A_t) \]

  with $A'_{t+1}$ drawn using $\pi$. 

---
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.
\( \tilde{T} Q(s, a) = Q(s, a) + \mathbb{E}_b \left[ \sum_{t \geq 0} \gamma^t \left( \prod_{t' = 1}^{t} c_{t'} \right) \delta_t \right] \) if \( S_0 = s, A_0 = a \)

\[
c_t = c(A_t, S_t, A_{t-1}, S_{t-1}, \ldots, A_0, S_0)
\]

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

**Generic Off-Policy Algorithm**

- Generic off-line algorithm including
  - Importance sampling: \( c_t = \rho_{t:t} = \pi(A_t|S_t)/b(A_t|S_t) \)
  - \( \text{TB}(\lambda): c_t = \lambda \pi(A_t|S_t) \)
  - \( \text{Retrace}(\lambda): c_t = \lambda \min(1, \pi(A_t|S_t)/b(A_t/S_t)) \)

- **Prop:** \( Q_\pi \) is a fixed point as \( \mathbb{E}_b[\delta_t|S_t, A_t] = \mathbb{E}[T^\pi Q(S_t, A_t) - Q(S_t, A_t)|S_t, A_t] \).

- **Prop:** \( \tilde{T} \) is a contraction provided \( c_t \leq \rho_t = \pi(A_t|S_t)/b(A_t|S_t) \).

- Convergence for Importance sampling, \( \text{TB}(\lambda) \) and \( \text{Retrace}(\lambda) \) for any \( b \).

- Partial results for policy improvement under more assumptions.

- For \( Q(\lambda), c_t = \lambda \), convergence if \( \|\pi(|s) - b(|s)|\|_1 \leq \epsilon \) and \( \lambda \leq (1 - \gamma)/(\gamma \epsilon) \).
1. $n$-step Algorithms
2. Eligibility Traces
3. Off-policy vs on-policy
4. Bandits
5. Model Based Approach
6. Replay Buffer and Prioritized Sweeping
7. Real Time Planning
8. References
**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**: 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:

  $$ \sum_t \mathbb{E}_{n^*}[R_t] - \mathbb{E}_{n_t}[R_t] $$

  which forces us to be good as fast as possible.
- No natural definition in the discounted setting.
Very simple toy model where there is only one state!

Optimal policy: pick $a_\star \in \arg\max r_a$.

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 maximise the rewards during the learning 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 (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,

\[ \mathbb{P}(a_T = a_\ast) \geq 1 - \sum_{a \neq a_\ast} \exp(-m\Delta(a)^2/4) \]
\( \epsilon \)-greedy Strategy

- Estimate \( Q(a) = r_a \) by MC:
  \[
  Q_t(a) = \frac{\sum_{t'=1}^{t-1} 1_{A_{t'}=a} R_{t'}}{\sum_{i=1}^{t-1} 1_{A_{t'}=a}}
  \]

- Pick arm \( a \) at time \( t \) using
  \[
  \pi(a) = \begin{cases} 
  \frac{\epsilon_t}{k} + (1 - \epsilon) & \text{if } a = \arg\max_{a'} Q_t(a') \text{ (only the smallest if necessary)} \\
  \frac{\epsilon_t}{k} & \text{otherwise}
  \end{cases}
  \]

- Prop:
  \[
  r_T \geq \sum_{t=1}^T \frac{\epsilon_t}{k} \sum_{a=1}^k \Delta(a)
  \]
\( \epsilon \)-greedy Strategy

\( \epsilon \)-greedy Strategy

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

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

Furthermore,

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

If \( \epsilon_t = c/t \),

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

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

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

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

**Upper Confidence Bound**

- Use an optimistic strategy to pick the best arm

\[ A_t = \text{argmax } Q_t(a) + \sqrt{\frac{c \log t}{N_t(a)}} \]

- 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 \( \epsilon \)-greedy may not be optimal.
Outline

1. $n$-step Algorithms
2. Eligibility Traces
3. Off-policy vs on-policy
4. Bandits
5. Model Based Approach
6. Replay Buffer and Prioritized Sweeping
7. Real Time Planning
8. References
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.
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

1. $n$-step Algorithms
2. Eligibility Traces
3. Off-policy vs on-policy
4. Bandits
5. Model Based Approach
6. Replay Buffer and Prioritized Sweeping
7. Real Time Planning
8. References
Replay Buffer and Prioritized Sweeping

- Can we reuse previous interactions?
- In which order?
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 mildly evolved...
Prioritized Sweeping

- Plain Replay Buffer: subsequence drawn uniformly.
- Prioritized Sweeping: subsequence drawn favoring states with large temporal differences.
- Can be combined with a model approach.
Outline

1. n-step Algorithms
2. Eligibility Traces
3. Off-policy vs on-policy
4. Bandits
5. Model Based Approach
6. Replay Buffer and Prioritized Sweeping
7. Real Time Planning
8. References
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 computationally efficient.
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

- 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 highest value.
- The deeper the better... but the more expensive due to branching!
- Requires a suitable heuristic...
Rollout Algorithm

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

- Simultaneous 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

- Real Time Planning

Monte Carlo Tree Search

- Simultaneous 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

- Simultaneous 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

- Simultaneous 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

- Simultaneous 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

- Simultaneous 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

- Simultaneous 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

- Simultaneous 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

- Simultaneous 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

- 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.
Outline

1. n-step Algorithms
2. Eligibility Traces
3. Off-policy vs on-policy
4. Bandits
5. Model Based Approach
6. Replay Buffer and Prioritized Sweeping
7. Real Time Planning
8. References
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.
Wiley, 2005

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

W. Powell.
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

Creative Commons Attribution-ShareAlike (CC BY-SA 4.0)

- You are free to:
  - **Share:** copy and redistribute the material in any medium or format
  - **Adapt:** remix, transform, and build upon the material for any purpose, even commercially.
- Under the following terms:
  - **Attribution:** You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
  - **ShareAlike:** If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
  - **No additional restrictions:** You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Contributors

- **Main contributor:** E. Le Pennec