Reinforcement Learning

Prof. Volkan Cevher
volkan.cevher@epfl.ch

Lecture 2: Dynamic Programming II

Laboratory for Information and Inference Systems (LIONS)
École Polytechnique Fédérale de Lausanne (EPFL)

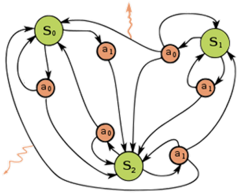
EE-568 (Spring 2025)

License Information for Reinforcement Learning (EE-568)

From planning to reinforcement learning

Fundamental Challenge 1

The dynamic programming approaches (VI and PI) as well as the linear programming approach all require the full knowledge of the transition model P and the reward.

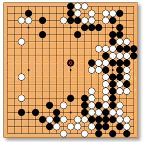


Fundamental Challenge 2

The computation and memory cost can be very expensive for large scale MDP problems.

Chess: 10^{120}

Go: 3^{361}

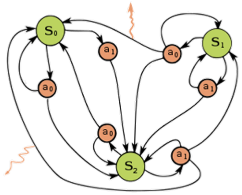


From planning to reinforcement learning

Fundamental Challenge 1

The dynamic programming approaches (VI and PI) as well as the linear programming approach all require the full knowledge of the transition model P and the reward.

⇒Need sampling approaches

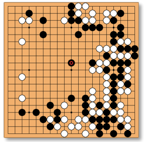


Fundamental Challenge 2

The computation and memory cost can be very expensive for large scale MDP problems.

Chess: 10^{120}

Go: 3^{361}

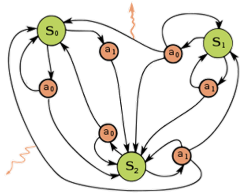


From planning to reinforcement learning

Fundamental Challenge 1

The dynamic programming approaches (VI and PI) as well as the linear programming approach all require the full knowledge of the transition model P and the reward.

⇒Need sampling approaches



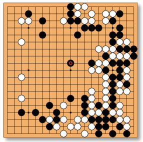
Fundamental Challenge 2

The computation and memory cost can be very expensive for large scale MDP problems.

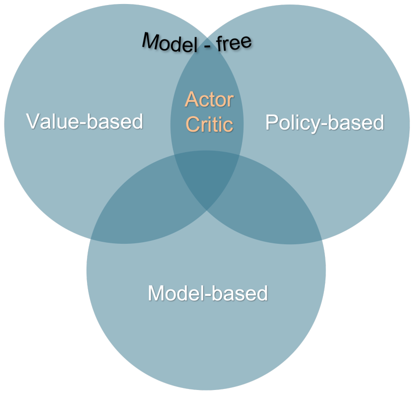
⇒Need new representations

Chess: 10^{120}

Go: 3^{361}



Overview of reinforcement learning approaches



- **Value-based RL**
  - Learn the optimal value functions $V^\star, Q^\star$
- **Policy-based RL**
  - Learn the optimal policy $\pi^\star$
- **Model-based RL**
  - Learn the model P, r and then do planning

Model-based vs model-free methods

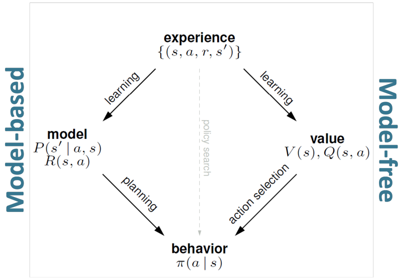


Figure: [7]

- ◦ Make full use of “experiences”
- ◦ Can reason about model uncertainty
- ◦ Sample efficient for easy dynamics

- ◦ Direct and simple
- ◦ Not affected by poor model estimation
- ◦ Not sample efficient

Online vs offline reinforcement learning

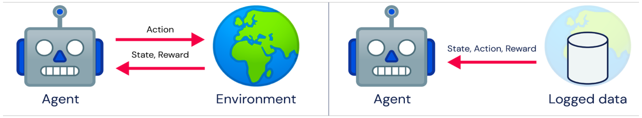


Figure: [3]

Online RL

- Collect data by interacting with environment
- Exploitation-exploration tradeoff

Offline/Batch RL

- Use previously collected data
- Data is static, no online data collection

On-policy vs off-policy reinforcement learning

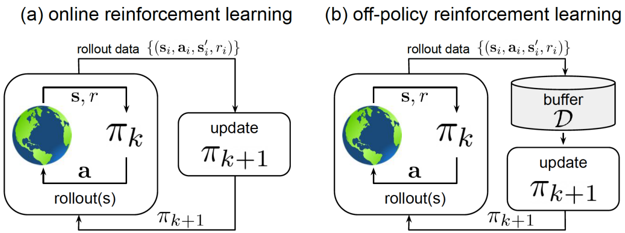

(a) online reinforcement learning
(b) off-policy reinforcement learning
rollout data $\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$
$\mathbf{s}, r$
π_k
$\mathbf{a}$
rollout(s)
update
π_{k+1}
buffer
$\mathcal{D}$

Figure: [11]

On-policy RL

- Learn based on data from current policy
- Always online

Off-policy RL

- Learn based on data from other policies
- Can be online or offline

Representation learning

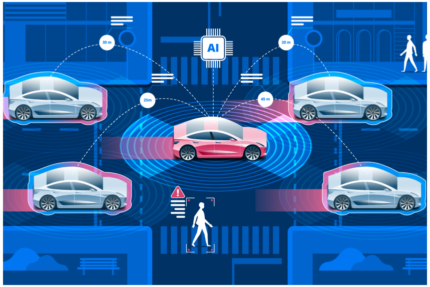


Figure: `http://selfdrivingcars.space/?p=68`

Large or continuous state and action spaces

$\Longrightarrow$

Function approximation

$$V(s) \approx V_\theta(s)$$
$$Q(s,a) \approx Q_\theta(s,a)$$
$$\pi(a|s) \approx \pi_\theta(a|s)$$
$$\mathsf{P}(s'|s,a) \approx \mathsf{P}_\theta(s'|s,a)$$

Value function representations

Linear function approximations

- The value function can be represented linearly using some known basis functions ϕ as follows:

$$V_\theta(s) = [\phi_1(s), \ldots, \phi_d(s)] \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{bmatrix}$$

- Reproducing kernel Hilbert space (RKHS) [14]
- Neural tangent kernel [6]

Nonlinear Function Approximation

- Fully connected neural networks [10]
- Convolutional neural networks [9]
- Residual networks [4]
- Recurrent networks [5]
- Self-attention [21]
- Generative adversarial networks [2]

Remarks: ○ In continuous d dimensional state space linear function approx (LFA) is better then discretizing.

○ Discretizing we would end up with a number of states which is exponential in d.

○ Using features we will have algorithms that find an almost optimal policy in poly(d)-time.

The advantage of LFAs

- The improvement is possible because each state is treated independently in the tabular setting.
- Instead, with LFA we can exploit similarities between states.

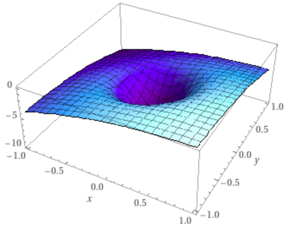


Figure: Reward function in a continuous grid world example

Example:
- Reward in the figure changes smoothly between neighboring states.
- After discretization, the states are independent.
- In contrast, LFA allows to exploit similarities.

Towards non-linear function approximations

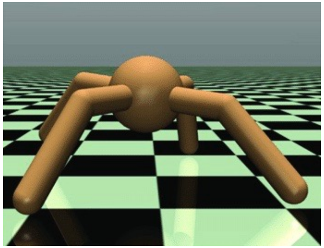


Figure: The ant environment in MuJoCo.

- In some important applications, like robotics, LFA is not expressive enough.
- For instance, in MuJoCo [19], we usually use a 3 layers neural network to parameterize the value function.
- The input is a vector containing position and velocity of the "ant" joints.

Our first goal: Model-free prediction

Goal:

Estimate $V^\pi(s)$ or $Q^\pi(s,a)$ from trajectories $\tau = \{s_0, a_0, r_0, s_0, \ldots\}$ collected with a given $\pi : \mathcal{S} \to \Delta(\mathcal{A})$:

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

where the expectation is taken with respect to the randomness of the environment and the randomness of the actions due to π. We assume that the transition dynamics is not available.

Observations:

- Similar to the policy evaluation phase in Policy Iteration.
- But without knowledge of the dynamics!
- We will explain and analyze two methods:
  - Temporal Differences (TD).
  - Monte Carlo (MC).

Monte Carlo: the optimization problem

∘ Let us consider a state distribution $\rho \in \Delta_{\mathcal{S}}$.

∘ In MC, this is typically the initial state distribution.

Monte-Carlo optimization problem

We will use the following optimization problem to explain the main ideas in Monte-Carlo approaches:

$$\min_{V} \mathcal{L}(V) \tag{MC}$$

where $\mathcal{L}_{\text{MC}}(V) = \frac{1}{2}\left\|V^{\pi} - V\right\|_{\rho}^{2}$ is a ρ-weighted norm loss.

Observations:

∘ The gradient is simply $\nabla\mathcal{L}_{\text{MC}}(V) = \operatorname{diag}(\rho)(V - V^{\pi})$.

∘ The role of the distribution ρ will be made clearer in the next slides.

∘ We will apply stochastic gradient descent method to solve this problem.

Solving the problem via stochastic gradient descent (SGD)

◦ Ingredients of SGD algorithm:

- the gradient estimate $g_t = (V(s_0) - \mathbb{E}[G^\pi(s_0)])\mathbf{e}_{s_0}$ for $s_0 \sim \rho$;
- a step-size (i.e., learning rate) η_t;
- a simple iteration invariant for SGD applied to the MC optimization problem: $\boxed{V_{t+1} = V_t - \eta g_t}$.

Recall: A general SGD bound for convex minimization[1]

Let us apply SGD to the problem $\min_{x\in\mathcal{X}} f(x)$, where $f : \mathcal{X} \to \mathbb{R}$ is a convex function: $x_{t+1} = x_t - \eta_t g(x_t)$.

We assume that the stochastic gradient $g(x)$ is a random vector such that

- $\mathbb{E}[g(x)] = \nabla f(x)$ for all $x \in \mathcal{X}$.
- $\mathbb{E}\left[\|g(x)\|^2\right] \leq \sigma^2$ for all $x \in \mathcal{X}$.

If we use the step-size $\eta_t = 1/(\sigma\sqrt{T})$ (i.e., constant step-size depending on the horizon T), it holds that

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} f(x_t) - f(x^\star)\right] \leq \frac{\|x_0 - x^\star\|^2 \sigma}{\sqrt{T}}. \qquad \text{(SGD Bound)}$$

[1]See EE-556 Math of Data Lecture 6 for further details.

The Monte Carlo proof roadmap

◦ We are going to the convergence of the Monte Carlo method with the following steps.

Step 1: Proof that the MC gradient estimate is unbiased, i.e. $\mathbb{E} g_t = \nabla \mathcal{L}_{\mathsf{MC}}(V)$.

Step 2: Proof that the MC gradient estimate has bounded second moment, i.e. $\mathbb{E}\left\|g_t\right\|^2 \leq \frac{1}{(1-\gamma)^2}$.

Step 3: Plug in the convergence bound of SGD for general convex functions.

Step 1: Establishing an unbiased gradient estimate

○ Let us define a (random) vector $G^\pi \in \mathbb{R}^{|\mathcal{S}|}$ with entries $G^\pi(s_0) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$ with $s_0 \sim \rho$, given π.

- G^π is unbiased estimator of V^π: By the definition of (V^π), we indeed have $\mathbb{E}[G^\pi(s_0)] = V^\pi(s_0)$.
- Recall that the expectation is taken with respect to $s_0 \sim \rho$.
- Let $\mathbf{e}_s \in \{0,1\}^{|\mathcal{S}|}$ be the vector such that the s^{th} entry equals 1, and is zero, elsewhere.
- Therefore, for any V, the MC gradient estimate is unbiased g_t is an unbiased gradient estimator:

$$\begin{aligned}
\mathbb{E}_{s_0\sim\rho}[g_t] &= \mathbb{E}_{s_0\sim\rho}[(V(s_0) - \mathbb{E}[G^\pi(s_0)])\mathbf{e}_{s_0}] \\
&= \mathbb{E}_{s_0\sim\rho}[(V(s_0) - V^\pi(s_0))\mathbf{e}_{s_0}] \\
&= \operatorname{diag}(\rho)(V - V^\pi) \\
&= \nabla\mathcal{L}(V).
\end{aligned}$$

Step 2: Proving a bounded second-order moment for the gradient estimate

◦ We can prove that g_t has bounded second moment, $\mathbb{E}[\|g_t\|^2] \leq \frac{1}{(1-\gamma)^2}$ as follows:

$$\begin{aligned}\mathbb{E}[\|g_t\|^2] &= \mathbb{E}[\|(G^\pi(s_0) - V^\pi(s_0))\mathbf{e}_{s_0}\|^2] \\ &= (G^\pi(s_0) - V^\pi(s_0))^2 \\ &\leq \frac{1}{(1-\gamma)^2}.\end{aligned}$$

Remark: where the final bound holds because the entries are bounded between 0 and $(1-\gamma)^{-1}$.

Step 3: Applying the known SGD bound

∘ We proved that g_t is unbiased.

∘ g_t has bounded second moment, $\mathbb{E}\left[\|g_t\|^2\right] \leq \sigma^2$ with $\sigma = (1-\gamma)^{-1}$.

∘ Therefore, plugging into the general SGD bound (SGD Bound), we obtain the following convergence result.

The guarantee for the SGD algorithm for (MC): $\boxed{V_{t+1} = V_t - \eta g_t}$

Plugging into the SGD bound for convex functions with

- $\sigma^2 = \frac{1}{(1-\gamma)^2}$,
- $\eta = \frac{1}{\sigma\sqrt{T}}$.
- $x_0 = V_0$, $x^\star = V^\pi$.

we can obtain the following guarantee for SGD:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|V^\pi - V_t\|_\rho^2] \leq \frac{\|V_0 - V^\pi\|^2}{(1-\gamma)\sqrt{T}}. \tag{1}$$

Remark: The above guarantee also holds true for the average iterate $\bar{V}_T = \frac{1}{T}\sum_{t=1}^{T} V_t$.

SGD analysis via strong convexity

- We can achieve a faster rate if the distribution ρ has full support.
- Indeed, the loss function $\mathcal{L}(V)$ is $(\min_s \rho(s))$-strongly convex (recall the Hessian $\nabla^2 \mathcal{L}(V) = \mathrm{diag}(\rho)$).
    - Below, we will assume that $(\min_s \rho(s)) > 0$.

Recall: The strongly convex SGD bound for convex minimization[2]

Let us apply SGD to the problem $\min_{x \in \mathcal{X}} f(x)$, where $f : \mathcal{X} \to \mathbb{R}$ is a χ-strongly convex function:
$x_{t+1} = x_t - \eta_t g(x_t)$.

We assume that the stochastic gradient $g(x)$ is a random vector such that

- $\mathbb{E}\left[g(x)\right] = \nabla f(x)$ for all $x \in \mathcal{X}$.
- $\mathbb{E}\left[\|g(x)\|^2\right] \leq \sigma^2$ for all $x \in \mathcal{X}$.

If we use the step-size $\eta_t = 1/(\chi t)$, then it holds that

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} f(x_t) - f(x^\star)\right] \leq \frac{\sigma^2 \log(T)}{\chi T}.$$

[2]See EE-556 Math of Data Lecture 6 for further details.

Plug in to the SGD bound for strongly convex functions

The strongly-convex guarantee for the SGD algorithm for (MC): $\boxed{V_{t+1} = V_t - \eta g_t}$

Plugging into the SGD bound for strongly convex smooth functions with

- $\sigma^2 = \frac{1}{(1-\gamma)^2}$, $\chi = \min_s \rho(s)$, and $\eta_t = \frac{1}{\chi t}$,

we can obtain the following guarantee for SGD:

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|V^\pi - V_t\|_\rho^2] \leq \frac{\log T}{T \min_s \rho(s)(1-\gamma)^2}. \tag{2}$$

By Jensen's inequality, the above guarantee holds true for the average iterate $\bar{V}_T = \frac{1}{T}\sum_{t=1}^{T} V_t$ as well.

Question*: ○ What happens $(\min_s \rho(s)) = 0$? Which setting is better? This is a nuanced question!

Monte Carlo: The algorithm

Idea: ◦ Estimate $V^\pi(s)$ by the average of returns following all visits to s.

The Monte Carlo Algorithm [12, 18]

```
for $t = 1, \ldots, T$ do
    Collect an episode $\tau = \{s_0, a_0, r_0, s_1, \ldots, s_{\mathbb{T}}, a_{\mathbb{T}}, r_{\mathbb{T}}\}$ generated following $\pi$
    for each state $s_h$ do
        Compute return $G^\pi(s_t) = r_t + \gamma r_{t+1} + \cdots$
        Update $V_{t+1}(s_h) \leftarrow V_t(s_h) + \eta_t(G^\pi(s_h) - V_t(s_h))$
    end for
end for
```

Observations: ◦ This is **not** the SGD method with the step-size η_t (why?)

Monte Carlo: The algorithm

Idea: ◦ Estimate $V^\pi(s)$ by the average of returns following all visits to s.

The Monte Carlo Algorithm [12, 18]

```
for $t = 1, \ldots, T$ do
    Collect an episode $\tau = \{s_0, a_0, r_0, s_1, \ldots, s_\mathbb{T}, a_\mathbb{T}, r_\mathbb{T}\}$ generated following $\pi$
    for each state $s_h$ do
        Compute return $G^\pi(s_t) = r_t + \gamma r_{t+1} + \cdots$
        Update $V_{t+1}(s_h) \leftarrow V_t(s_h) + \eta_t(G^\pi(s_h) - V_t(s_h))$
    end for
end for
```

Observations:
- ◦ This is **not** the SGD method with the step-size η_t (why?) Due to the second loop!
- ◦ Notice that in this implementation the updates G^π are correlated in the second loop.
- ◦ The value estimates do not build on the other states even if they may be correlated.
- ◦ Learning can be slow when the episodes are long.
- ◦ The terminology of episode and trajectories are equivalent in the literature.

Monte Carlo with linear function approximation (LFA)

∘ We can parameterize the value function, i.e., $V_\theta = \Phi\theta$ with $\theta \in \mathbb{R}^d, \Phi \in \mathbb{R}^{|\mathcal{S}|\times d}$.

Monte Carlo optimization problem with LFA

For the case of linear function approximation, we look at the following optimization problem

$$\min_\theta \mathcal{L}(\theta) \tag{MC LFA}$$

with $\mathcal{L}(\theta) = \frac{1}{2}\left\|V^\pi - \Phi\theta\right\|_\rho^2$.

Observations: ∘ The gradient of the loss in $(\mathrm{MC\ LFA})$ is $\nabla\mathcal{L}(\theta) = \Phi^T \mathrm{diag}(\rho)(\Phi\theta - V^\pi)$.

∘ Similar to the SGD approach in (MC), we can have an unbiased estimator of the gradient.

SGD with linear function approximation

Let $s \sim \rho$ and let $\phi(s)$ denote the s^{th} row of Φ. Then, the SGD updates are as follows

$$\theta_{t+1} = \theta_t - \eta_t \phi(s)((\Phi\theta_t)(s) - G^\pi(s)),$$

where $\phi(s)((\Phi\theta_t)(s) - G^\pi(s))$ is an unbiased gradient estimate (why?) and η_t has to be chosen appropriately.

Monte Carlo analysis

- We can analyze MC with LFA via SGD.
- By using the SGD convergence for convex functions, we obtain the following guarantee.

SGD convergence bound with LFA

Let $\theta^\star$ be the minimizer of the loss function in $(\mathrm{MC\ LFA})$ and let us assume realizability, i.e., $V_{\theta^\star} = V^\pi$. Let us run Monte Carlo for T iterations with step size $\eta = \frac{\sqrt{d}(1-\gamma)}{\sqrt{t}}$. Then, it holds that

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|V_{\theta_t}(s) - V^\pi(s)\|_\rho^2] \leq \frac{2\sqrt{d}\log T}{(1-\gamma)\sqrt{T}}.$$

By Jensen's inequality, the above guarantee holds true for the average iterate $\bar{V}_T = \frac{1}{T}\sum_{t=1}^{T} V_{\theta_t}$ as well.

Derivation: ◦ The complete proof is in the appendix 13.

Monte Carlo cannot handle infinite horizon problems

- Unfortunately, we can make an update only after reaching the end of an episode.
  - Recall the definition $G^\pi(s_0) := \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$.
  - All of our unbiased gradient estimators used this quantity.
- If the problem has an infinite horizon, we cannot compute G!
- We need to artificially introduce a finite horizon truncating the trajectories.
- The truncation introduces some bias making the above analysis invalid.
- These problems can be overcome by the temporal difference method that we look at next.

Temporal difference (TD) learning

- Recall LFA: We parameterize the value function, i.e., $V_\theta(s) = \phi(s)^T\theta$.
- Recall the Bellman operator: $(\mathcal{T}^\pi V)(s) = \sum_a \pi(a|s)\left(r(s,a) + \gamma \sum_{s'} \mathsf{P}(s'|s,a)V(s')\right)$.
- Recall that V^π is a fixed point of $\mathcal{T}^\pi$: i.e., $V^\pi = \mathcal{T}^\pi V^\pi$.

Optimization program for TD learning

The TD learning solves the following convex program

$$\min_{\theta\in\mathbb{R}^d} \mathcal{L}(\theta), \tag{TD}$$

where the objective $\mathcal{L}(\theta) = \frac{1}{2}\|V_\theta - \mathcal{T}^\pi V_\theta\|^2_{\rho_\infty}$ measures the violation of the fixed-point condition.

- In TD, we consider ρ_∞ as the limit distribution below (in contrast to MC, where it is the initial distribution):

$$\rho_\infty(s) = \lim_{t\to\infty} \mathbb{P}_\pi[s_t = s].$$

- We can write the gradient of the loss function (TD) as

$$\nabla_\theta \mathcal{L}(\theta) = \Phi^T(I - \mathcal{T}^\pi)^T \mathrm{diag}(\rho_\infty)(I - \mathcal{T}^\pi)\Phi\theta. \tag{TD-Gradient}$$

Can we find an unbiased estimator, i.e. $\mathbb{E}[g_t] = \nabla_\theta \mathcal{L}(\theta)$?

- Challenge: We do not know the transition dynamics P in the model-free setting.
  - but we can sample from it!
- The TD-Gradient can be numerically computed via the following expression:

$$\begin{aligned}\nabla_\theta \mathcal{L}(\theta) &= \sum_{s\in\mathcal{S}} \rho_\infty(s) \left(V_\theta(s) - \sum_{a\in\mathcal{A}} \pi(a|s)(r(s,a) + \gamma \mathbb{E}_{s'\sim \mathsf{P}(\cdot|s,a)}[V_\theta(s')]) \right) \cdot \left(\nabla_\theta V_\theta(s) - \gamma \mathbb{E}_{s'\sim \mathsf{P}(\cdot|s,a)}[\nabla_\theta V_\theta(s')] \right) \\ &= \mathbb{E}_{s\sim\rho_\infty} \left[\left(V_\theta(s) - \sum_{a\in\mathcal{A}} \pi(a|s)(r(s,a) + \gamma \mathbb{E}_{s'\sim \mathsf{P}(\cdot|s,a)}[V_\theta(s')]) \right) \cdot \left(\nabla_\theta V_\theta(s) - \gamma \mathbb{E}_{s'\sim \mathsf{P}(\cdot|s,a)}[\nabla_\theta V_\theta(s')] \right) \right]\end{aligned}$$

- So to approximate it, we can sample $S \sim \rho, A \sim \pi(\cdot|S)$ and $S' \sim \mathsf{P}(\cdot|S,A)$, and propose the estimator

$$g_t = \left(V_\theta(S) - r(S,A) - \gamma V_\theta(S') \right) \cdot \left(\nabla_\theta V_\theta(S) - \gamma \nabla_\theta V_\theta(S') \right)$$

- But this is clearly biased because the expectation does not distribute over products:

$$\mathbb{E}_{S'\sim\mathsf{P}(\cdot|S,A)}[V_\theta(S')\nabla_\theta V_\theta(S')] \neq \mathbb{E}_{S'\sim\mathsf{P}(\cdot|S,A)}[V_\theta(S')]\mathbb{E}_{S'\sim\mathsf{P}(\cdot|S,A)}[\nabla_\theta V_\theta(S')].$$

Controlling the bias: the temporal difference (TD) method

◦ We will judiciously ignore the second term depending on θ, i.e.,

$$\Big(V_\theta(S) - r(S,A) - \gamma V_\theta(S')\Big) \cdot \Big(\nabla_\theta V_\theta(S) - \cancel{\gamma \nabla_\theta V_\theta(S')}\Big),$$

and we define

$$g_t = \Big(V_\theta(S) - r(S,A) - \gamma V_\theta(S')\Big) \cdot \left(\nabla_\theta V_\theta(S)\right).$$

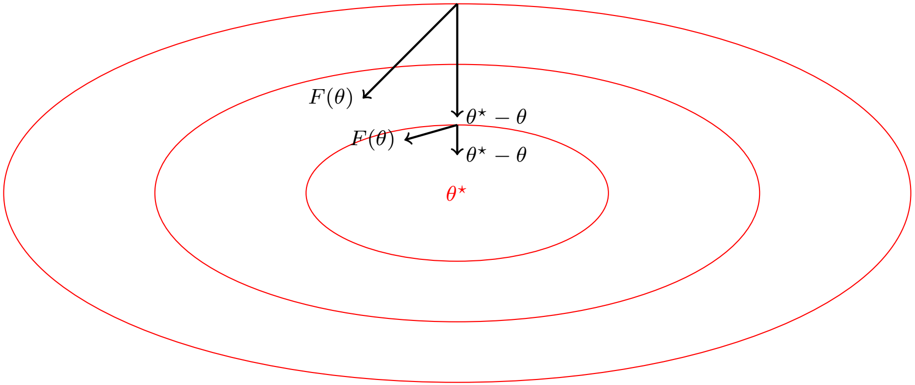
◦ This suggestion for g_t is an unbiased estimator for the following partial computation of TD-Gradient

$$\begin{aligned} F(\theta) &= \sum_{s\in\mathcal{S}} \rho_\infty(s) \left(V_\theta(s) - \sum_{a\in\mathcal{A}} \pi(a|s)(r(s,a) + \gamma \mathbb{E}_{s'\sim \mathsf{P}(\cdot|s,a)}[V_\theta(s')]) \right) \cdot \nabla_\theta V_\theta(s) \\ &= \mathbb{E}_{s\sim\rho_\infty}\left[\Big(V_\theta(s) - \sum_{a\in\mathcal{A}} \pi(a|s)(r(s,a) + \gamma \mathbb{E}_{s'\sim \mathsf{P}(\cdot|s,a)}[V_\theta(s')]) \Big) \cdot \nabla_\theta V_\theta(s) \right] \end{aligned}$$

◦ Is $F(\theta)$ a good update direction?

- Yes, it holds that $\langle F(\theta), \theta^\star - \theta\rangle \geq (1-\gamma)\left\|V_\theta - V_{\theta^\star}\right\|_\rho^2$, i.e., it is a *descent* direction (*cf.*, proof in [1]).
- This means that the expected direction of the update forms an acute angle with the vector pointing to $\theta^\star$.
- In addition, $F(\theta)$ satisfies $\|F(\theta)\|_2 \leq 2\left\|V_\theta - V_{\theta^\star}\right\|_\rho$ (*cf.*, proof in Lemma 4 of [1]).

A visualization of $F(\theta)$ and $\theta^\star - \theta$



- Notice that $F(\theta)$ is biased but forms an acute angle with $\theta^\star - \theta$.
- The reasor being that we show that their inner product is positive.
- The closer we get to $\theta^\star$, the less acute the angle can be.

Analysis of TD

∘ We consider the following iterative scheme

$$\theta_{t+1} = \theta_t - \eta_t F(\theta_t). \tag{TD Update}$$

Finite time bound for TD

Running T updates in (TD Update) with a step-size $\eta_t = \frac{1-\gamma}{4\sqrt{t}}$, we obtain

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|V_{\theta_t} - V_{\theta^\star}\|_\rho^2] \leq \frac{9d\log T}{2(1-\gamma)^2\sqrt{T}},$$

where $\theta^\star$ is the minimizer of the loss function in (TD).

Remarks:

∘ It is slightly worse than applying SGD to (MC) but is much easier to implement!

∘ The proof is in the appendix and simplifies the derivations in [1].

Temporal difference learning

Idea:

- Incrementally estimate $V^\pi(s)$ by the intermediate return plus estimated return at next state.
- In the linear case, $g_t = \big(V(s_t) - r_t - \gamma V(s_{t+1})\big)\nabla_\theta V_{\theta_t}(s_t)$ and the TD update is

$$\theta \leftarrow \theta - \eta_t g_t$$

TD Learning / TD(0)

```
for $t = 1, \ldots, T$ do
  for each step of an episode $\tau = \{s_0, a_0, r_0, s_1, \ldots, s_\mathbb{T}, a_\mathbb{T}, r_\mathbb{T}\}$ following $\pi$ do
    Compute update direction $g_t = \big(V(s_t) - r_t - \gamma V(s_{t+1})\big)\nabla_\theta V_{\theta_t}(s_t)$
    Update $\theta_{t+1} \leftarrow \theta_t - \eta_t g_t$
  end for
end for
```

Observations:

- Note that above implementation (and practitioners) do not sample from ρ_∞!
  - [1] proves that you only suffer a small additive bias in the convergence guarantee.
- Similar to MC: learn directly from episodes of experiences without the MDP knowledge.
- Unlike MC: learn from incomplete episodes, and applicable to non-terminating environment.

Convergence in the misspecified setting: Monte Carlo

- In the misspecified case, we have that $\min_\theta \|\Phi\theta - V^\pi\|_\rho = \epsilon_{\text{approx}} > 0$.
- If $\epsilon_{\text{approx}} = 0$ MC and TD converge to the same point.
- Otherwise, MC and TD converges to different points.

A property of the MC solution

For MC, the optimality condition $\nabla_\theta \mathcal{L}(\theta^\star_{\text{MC}}) = 0$ implies that

$$\Phi^T \text{diag}(\rho)\Phi\theta^\star_{\text{MC}} = \Phi^T \text{diag}(\rho) V^\pi,$$

which implies that $\theta^\star_{\text{MC}}$ is the projection of V^π in the feature span.

Derivation:

- From the stationarity of the solution, we have

$$\begin{aligned} &\nabla_\theta \mathcal{L}(\theta^\star_{\text{MC}}) = 0 \\ \implies &\Phi^T \text{diag}(\rho)(V_{\theta^\star_{\text{MC}}} - V^\pi) = 0 \\ \implies &\Phi^T \text{diag}(\rho)(\Phi\theta^\star_{\text{MC}} - V^\pi) = 0 \end{aligned}$$

Therefore, rearranging the terms gives $\Phi^T \text{diag}(\rho)\Phi\theta^\star_{\text{MC}} = \Phi^T \text{diag}(\rho) V^\pi$.

Convergence in the misspecified setting: TD

- For TD, we have that the stationary point $\theta^\star_{\mathrm{TD}}$ satisfies $F(\theta^\star_{\mathrm{TD}}) = 0$, i.e.,

$$\theta^\star_{\mathrm{TD}} = (\Phi^T \mathrm{diag}(\rho)\Phi)^{-1}\Phi^T \mathrm{diag}(\rho) T^\pi \Phi \theta^\star_{\mathrm{TD}}.$$

  - the derivation can be found in the next slide.
- Hence, $\theta^\star_{\mathrm{TD}}$ is not directly related to V^π but it is the fixed point of a projected Bellman equation.
- [20] has shown that

$$\left\| V_{\theta^\star_{\mathrm{TD}}} - V^\pi \right\|_\rho \leq \frac{1}{\sqrt{1-\gamma^2}} \left\| \Phi \theta^\star_{\mathrm{MC}} - V^\pi \right\|_\rho .$$

Remarks:

- $\frac{1}{\sqrt{1-\gamma^2}}$ is an inflation factor that TD pays w.r.t. the minimum possible approximation error.
- The minimum possible approximation error is achieved by the MC method.

*Stationarity condition in TD

◦ In the previous slides, we used the fact

$$\theta^\star_{\mathrm{TD}} = (\Phi^T \mathrm{diag}(\rho)\Phi)^{-1}\Phi^T \mathrm{diag}(\rho) T^\pi \Phi \theta^\star_{\mathrm{TD}}.$$

Derivation: ◦ The stationarity condition $F(\theta^\star_{\mathrm{TD}}) = 0$ can be written in vector form, i.e.

$$\Phi^T \mathrm{diag}(\rho)(I - \mathcal{T}^\pi)\Phi\theta^\star_{\mathrm{TD}} = 0.$$

◦ Then, we just need the following steps:

$$\begin{aligned} &\Phi^T \mathrm{diag}(\rho)(\Phi\theta^\star_{\mathrm{TD}} - \mathcal{T}^\pi \Phi\theta^\star_{\mathrm{TD}}) = 0 \\ \Longrightarrow\ &\Phi^T \mathrm{diag}(\rho)\Phi\theta^\star_{\mathrm{TD}} = \Phi^T \mathrm{diag}(\rho)\mathcal{T}^\pi \Phi\theta^\star_{\mathrm{TD}} \\ \Longrightarrow\ &\theta^\star_{\mathrm{TD}} = (\Phi^T \mathrm{diag}(\rho)\Phi)^{-1}\Phi^T \mathrm{diag}(\rho)\mathcal{T}^\pi \Phi\theta^\star_{\mathrm{TD}} \end{aligned}$$

State-Action-Reward-State-Action (SARSA) for Q-value estimation

- In VI or PI, we often require the evaluation of Q-function to compute the greedy policy or optimal policy.
- How do we estimate Q^π?

SARSA optimization problem

The algorithm SARSA solves the following program:

$$\min_Q \mathcal{L}(Q) := \frac{1}{2} \|Q - \mathcal{T}^\pi Q\|_\rho^2 ,$$

where $\mathcal{T}^\pi$ is in this case the Bellman operator for Q value functions, that is,

$$(\mathcal{T}^\pi Q)(s,a) = r(s,a) + \gamma \sum_{s',a'} \mathsf{P}(s'|s,a)\pi(a'|s')Q(s',a').$$

Remarks:
- This is essentially the analog of the fixed point the TD loss but for the Q-function.
- We again use ρ as the stationary distribution (in contrast to initial distribution).

SARSA: the algorithm

- If we solve the SARSA program with SGD we obtain the following algorithm.
- SARSA should be understood as the TD method applied to the action-value function.

SARSA [15]

```
for $t = 1, \dots, T$ do
  for each step of an episode $\tau = \{s_0, a_0, r_0, s_1, \dots, s_{\mathbb{T}}, a_{\mathbb{T}}, r_{\mathbb{T}}\}$ following $\pi$ do
    Update $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
  end for
end for
```

- With the same steps presented for the case of TD(0), one can prove convergence of SARSA.

Using model-free prediction for control

◦ We can look at MC and TD as approximate policy evaluation routines and use them in policy iteration.

Algorithm: Policy Iteration (PI) for solving MDPs

```
Start with an arbitrary policy guess $\pi_0$
for each iteration $t$ do
    (Step 1: Policy evaluation) Compute $V^{\pi_t}$:
        (Option 1) Iteratively apply policy value iteration, $V_t \leftarrow \mathcal{T}^{\pi_t} V_t$, until convergence. (Exact)
        (Option 2) Use the closed-form solution: $V^{\pi_t} = (I - \gamma P^{\pi_t})^{-1} R^{\pi_t}$. (Exact)
        (Option 3) Approximate $V^{\pi_t}$ with Monte Carlo. (Inexact)
        (Option 4) Approximate $V^{\pi_t}$ with Temporal Differences. (Inexact)
    (Step 2: Policy improvement) Update the current policy $\pi_t$ by the greedy policy
        $\pi_{t+1}(s) = \arg\max_{a \in \mathcal{A}} \left[ r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathrm{P}(s'|s,a) V^{\pi_t}(s') \right]$.    (3)
end for
```

Using model-free prediction for control

◦ The advantage of Options 3 and 4 is that they do not require knowledge of P.

◦ The disadvantage is that the value functions computed in these ways are *inexact*.

How the errors propagate in the PI analysis?

This question is answered by [13] that gives the bound

$$\sum_{s\in\mathcal{S}} \rho(s)(V^\star(s) - V^{\pi^k}(s)) \leq \mathcal{O}\left(\frac{1}{(1-\gamma)^2} \max_{1\leq j\leq k} \underbrace{\left\|V_{\theta_j} - V^{\pi_j}\right\|_\rho}_{\text{evaluation error}}\right) + \mathcal{O}(\gamma^k).$$

Remarks:

◦ The evaluation error can be controlled with the results derived before for MC or DP.

◦ The PI with these inexact options is also known as Least Square Policy Iteration [8].

Control while learning: Q-Learning

- We can use the same idea to estimate directly the optimal action-value function.

Q-Learning [22]

```
for $t = 1, \ldots, T$ do
  for each step of an episode $\tau = \{s_0, a_0, r_0, s_1, \ldots, s_{\mathbb{T}}, a_{\mathbb{T}}, r_{\mathbb{T}}\}$ following $\pi$ do
    Update $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t(r_t + \gamma \max_{b \in \mathcal{A}} Q(s_{t+1}, b) - Q(s_t, a_t))$
  end for
end for
```

- In Q-Learning, learning and control phases are not clearly separated.
- Running the notebooks, you will see that the choice of the policy π makes a big difference in practice.
- Actually also in theory but proving it is a bit too advanced for now.

Summary: Model-free prediction

Methods	**DP**	**MC**	**TD(0)**
Model knowledge	Need	No need	No need
Uses Bellman equation?	Yes	No	Yes
When to perform updates	After next step	After whole episode	After next step
Bias	-	Unbiased	Biased
Variance	-	Big	Small

Reference: [17]

Wrap Up

- PI and VI are dynamic programming methods applicable when the transition matrix is known.
- Monte Carlo methods are used to estimate value function when the transition matrix is unknown.
- Monte Carlo methods are an instance of stochastic approximation.
- TD is an application of dynamic programming when the transition matrix is unknown.
- The following week is about Linear Programming for RL!
- Next is Jupiter Notebook #1.

References I

[1] Jalaj Bhandari, Daniel Russo, and Raghav Singal.
A finite time analysis of temporal difference learning with linear function approximation.
In *Conference On Learning Theory*, pages 1691–1692. PMLR, 2018.
30, 32, 33

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio.
Generative Adversarial Networks.
ArXiv e-prints, June 2014.
11

[3] Caglar Gulcehre, Sergio Gómez Colmenarejo, Jakub Sygnowski, Thomas Paine, Konrad Zolna, Yutian Chen, Matthew Hoffman, Razvan Pascanu, Nando de Freitas, et al.
Addressing extrapolation error in deep offline reinforcement learning.
2020.
8

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.
Deep residual learning for image recognition.
pages 770–778, 2016.
11

[5] Sepp Hochreiter and Jürgen Schmidhuber.
Long short-term memory.
Neural computation, 9(8):1735–1780, 1997.
11

References II

[6] Arthur Jacot, Franck Gabriel, and Clément Hongler.
Neural tangent kernel: Convergence and generalization in neural networks.
In *Advances in neural information processing systems*, pages 8571–8580, 2018.
11

[7] Mykel J. Kochenderfer, Tim A. Wheeler, and Kyle H. Wray.
Algorithms for Decision Making.
MIT press, 2022.
7, 53, 54, 55

[8] Michail G Lagoudakis and Ronald Parr.
Least-squares policy iteration.
The Journal of Machine Learning Research, 4:1107–1149, 2003.
40

[9] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner.
Gradient-based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324, Nov 1998.
11

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton.
Deep learning.
Nature, 521(7553):436–444, 2015.
11

References III

[11] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu.
Offline reinforcement learning: Tutorial, review, and perspectives on open problems.
arXiv preprint arXiv:2005.01643, 2020.
9

[12] Donald Michie and Roger A Chambers.
Boxes: An experiment in adaptive control.
Machine intelligence, 2(2):137–152, 1968.
23, 24

[13] Bruno Scherrer, Victor Gabillon, Mohammad Ghavamzadeh, and Matthieu Geist.
Approximate modified policy iteration.
arXiv preprint arXiv:1205.3054, 2012.
40

[14] Bernhard Schölkopf and Alexander J. Smola.
Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.
MIT Press, Cambridge, MA, USA, 2001.
11

[15] Satinder Singh, Tommi Jaakkola, Michael L Littman, and Csaba Szepesvári.
Convergence results for single-step on-policy reinforcement-learning algorithms.
Machine learning, 38(3):287–308, 2000.
38

References IV

[16] Richard S. Sutton and A. G. Barto.
Reinforcement Learning: An Introduction.
MIT Press, Cambridge, MA, 1998.
60, 65

[17] Richard S Sutton and Andrew G Barto.
Reinforcement learning: An introduction.
MIT press, 2018.
42, 63, 66

[18] Richard S Sutton, Andrew G Barto, et al.
Introduction to reinforcement learning, volume 135.
MIT press Cambridge, 1998.
23, 24

[19] Emanuel Todorov, Tom Erez, and Yuval Tassa.
Mujoco: A physics engine for model-based control.
In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5033, 2012.
13

[20] John Tsitsiklis and Benjamin Van Roy.
Analysis of temporal-diffference learning with function approximation.
Advances in neural information processing systems, 9, 1996.
35

References V

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin.
Attention is all you need.
2017.
11

[22] Christopher JCH Watkins and Peter Dayan.
Q-learning.
Machine learning, 8(3-4):279–292, 1992.
41

Supplementary material

POMDPs

Partial observable Markov decision processes (POMDPs)

- $\mathcal{S}$ is the set of all possible states
- $\mathcal{A}$ is the set of all possible actions
- $\mathsf{P}(s'|s,a)$: $\mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition model
- Ω is the set of observations: $o \in \Omega$.
- O is a set of conditional observation probabilities: $\mathsf{O}(o|s',a)$.
- $r(s,a)$: $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function
- μ is the initial state distribution: $s_0 \sim \mu \in \Delta(\mathcal{S})$
- γ is the discount factor: $\gamma \in [0,1]$

MDP vs POMDP:

◦ POMDPs are flexible: *We do not have to have perfect information about the states.*

◦ POMDPs are closer to the real world.

Example: see a baby crying but do not know the true state (hungry, sleepy, etc).

◦ MDPs assume perfect knowledge of the states.

POMDPs

∘ When we do not observe the actual states, we construct the so-called *belief states* vector.

Definition (Belief states)

A belief state vector b_t is a distribution over states at time t that estimates the state distribution given the observation and the action history $h_t = \{o_0, a_0, \ldots, a_{t-1}, o_t\}$, i.e., $P(s_t = s|h_t)$:

$$b_t(s) := P(s_t = s|h_t).$$

Remarks: ∘ Via the Bayes rule, the belief states must satisfy:

$$\begin{aligned} \mathsf{P}(s_t = s|h_t) =& \frac{\mathsf{O}(o_t|s_t, a_{t-1}, h_{t-1})\mathsf{P}(s_t|a_{t-1}, h_{t-1})}{\mathsf{P}(o_t|a_{t-1}, h_{t-1})} \\ =& \frac{\mathsf{O}(o_t|s_t, a_{t-1}, h_{t-1}) \sum_{s_{t-1}} \mathsf{P}(s_t|s_{t-1}, a_{t-1})\mathsf{P}(s_{t-1}|h_{t-1})}{\sum_{s_t} \mathsf{O}(o_t|s_t, a_{t-1}, h_{t-1}) \sum_{s_{t-1}} \mathsf{P}(s_t|s_{t-1}, a_{t-1})\mathsf{P}(s_{t-1}|h_{t-1})}. \end{aligned}$$

∘ As a result, we have a recursion for the conditional probability $\mathsf{P}(s_t = s|h_t)$.

∘ We will represent this recursion via a "belief operator."

The belief operator

∘ We can concisely represent the recursion on $b_t(s)$ using the belief operator $U : \Delta(\mathcal{S}) \times \Omega \times \mathcal{A} \to \Delta(\mathcal{S})$:

$$b_{t+1}(s') = U(b_t; a, o)(s') = \frac{\mathsf{O}(o|s', a) \sum_{s \in S} \mathsf{P}(s'|s, a) b_t(s)}{\sum_{s'} \mathsf{O}(o|s', a) \sum_{s \in S} \mathsf{P}(s'|s, a) b_t(s)}.$$

Remarks:

∘ The expected (non-stationary) **reward** now also depends on our current belief state:

$$r_t(a) = \sum_{s \in S} r(a, s) b_t(s).$$

∘ We will focus more on MDPs and how to solve them optimally.

∘ Tools for MDPs translate readily to POMDPs once we have an estimate of $b_t(s)$.

Numerical example: Hex World

- Traverse a tile map to reach a goal state
- Each cell in the tile map represents a state; action is a move in any of the 6 directions
- Taking any action in certain cells gives a specified reward and transports to a **terminal state**

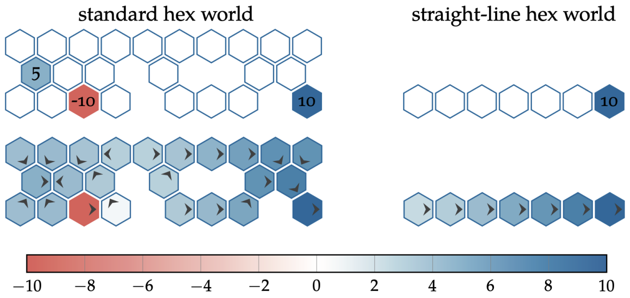


Figure: Top row shows the base problem setup and colors hexes with terminal rewards. Bottom row shows an optimal policy for each problem and colors the expected value. Arrows indicate the action to take in each state. [7]

Numerical example: Value iteration

◦ Initialized with the **east-moving** policy

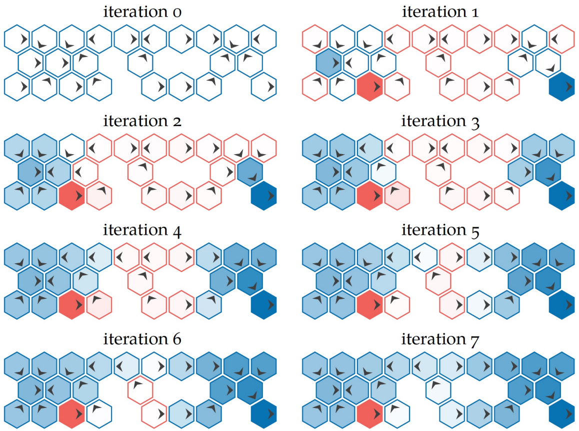


Figure: Value iteration for Hex World. [7]

Numerical example: Policy iteration

- Initialized with the **east-moving** policy
- An optimal policy is obtained (the algorithm converges) in four iterations

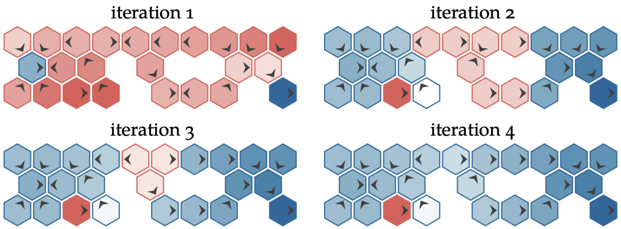


Figure: Policy iteration for Hex World. [7]

Proof for the Monte Carlo case.

Proof.

$$\begin{aligned}\|\theta_{t+1}-\theta^\star\|^2 &= \|\theta_t-\theta^\star\|^2 - \eta(\theta_t-\theta^\star)^T\phi(s)(V_{\theta_t}(s)-G(s)) + \eta^2\|\phi(s)(V_{\theta_t}(s)-G(s))\|^2\\ &= \|\theta_t-\theta^\star\|^2 - \eta(V_{\theta_t}(s)-V_{\theta^\star}(s))\cdot(V_{\theta_t}(s)-G(s)) + \eta^2\|\phi(s)(V_{\theta_t}(s)-G(s))\|^2\end{aligned}$$

Then, taking expectation and using that $\|\phi(s)(V_{\theta_t}(s)-G(s))\|^2 \le \frac{1}{(1-\gamma)^2}$,

$$\begin{aligned}\mathbb{E}[\|\theta_{t+1}-\theta^\star\|^2] &\le \mathbb{E}[\|\theta_t-\theta^\star\|^2] - \eta\mathbb{E}[(V_{\theta_t}(s)-V_{\theta^\star}(s))\cdot(V_{\theta_t}(s)-\mathbb{E}[G(s)|s])] + \frac{\eta^2}{(1-\gamma)^2}\\ &\le \mathbb{E}[\|\theta_t-\theta^\star\|^2] - \eta\mathbb{E}_{s\sim\rho}\mathbb{E}[(V_{\theta_t}(s)-V^\pi(s))^2] + \frac{\eta^2}{(1-\gamma)^2}\\ &\le \mathbb{E}[\|\theta_t-\theta^\star\|^2] - \eta\mathbb{E}[\|V_{\theta_t}(s)-V^\pi(s)\|_\rho^2] + \frac{\eta^2}{(1-\gamma)^2}\end{aligned}$$

Then, rearranging and dividing by η.

$$\mathbb{E}[\|V_{\theta_t}(s)-V^\pi(s)\|_\rho^2] \le \frac{\mathbb{E}[\|\theta_t-\theta^\star\|^2]-\mathbb{E}[\|\theta_{t+1}-\theta^\star\|^2]}{\eta} + \frac{\eta}{(1-\gamma)^2}$$

□

Proof for the Monte Carlo case (continued).

Proof.

Summing over $t \in [T]$, we obtain

$$\sum_{t=1}^{T} \mathbb{E}[\|V_{\theta_t}(s) - V^{\pi}(s)\|_{\rho}^2] \leq \frac{\mathbb{E}[\|\theta_1 - \theta^{\star}\|^2]}{\eta} + \frac{\eta}{(1-\gamma)^2} T$$
$$\leq \frac{d}{\eta} + \frac{\eta}{(1-\gamma)^2} T$$

Therefore, choosing $\eta = \frac{\sqrt{d}(1-\gamma)}{\sqrt{T}}$ and dividing by T.

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[\|V_{\theta_t}(s) - V^{\pi}(s)\|_{\rho}^2] \leq \frac{2\sqrt{d}}{(1-\gamma)\sqrt{T}}$$

Back to 22 □

Proof of Finite Time Bound for TD

Proof.

- For the analysis we restart from the same step we did for the case without variance but we take expectation.
- Recall that g_t equals $F(\theta)$ plus zero mean noise, i.e. $\mathbb{E}[g_t] = F(\theta)$.
- And it holds $\mathbb{E}\,\|g_t - F(\theta)\| \leq \sigma$.

$$
\begin{aligned}
\mathbb{E}[\|\theta_{t+1} - \theta^\star\|^2] &= \mathbb{E}[\|\theta_t - \theta^\star\|^2] - 2\mathbb{E}[\eta g_t^T(\theta_t - \theta^\star)] + \eta^2\mathbb{E}[\|g_t\|^2] \\
&\leq \mathbb{E}[\|\theta_t - \theta^\star\|^2] - 2\eta(1-\gamma)\mathbb{E}[\|V_{\theta_t} - V_{\theta^\star}\|_\rho^2] + 4\eta^2\mathbb{E}[\|V_{\theta_t} - V_{\theta^\star}\|_\rho^2] + \eta^2\mathbb{E}[\|g_t - F(\theta_t)\|^2] \\
&\leq \mathbb{E}[\|\theta_t - \theta^\star\|^2] - 2\eta(1-\gamma)\mathbb{E}[\|V_{\theta_t} - V_{\theta^\star}\|_\rho^2] + 4\eta^2\mathbb{E}[\|V_{\theta_t} - V_{\theta^\star}\|_\rho^2] + \eta^2\sigma^2 \\
&\leq (1 - \frac{(1-\gamma)^2\lambda_{\min}^2}{4})\mathbb{E}[\|\theta_t - \theta^\star\|^2] + \frac{\lambda_{\min}^2}{4} \\
&\leq \left(1 - \frac{(1-\gamma)^2\lambda_{\min}^2}{4}\right)^t \mathbb{E}[\|\theta_1 - \theta^\star\|^2] + \frac{\lambda_{\min}^2}{4}\sum_{\ell=0}^{\infty}\left(1 - \frac{(1-\gamma)^2\lambda_{\min}^2}{4}\right)^\ell \\
&\leq \left(1 - \frac{(1-\gamma)^2\lambda_{\min}^2}{4}\right)^t \mathbb{E}[\|\theta_1 - \theta^\star\|^2] + \frac{1}{(1-\gamma)^2}
\end{aligned}
$$

□

Proof of $\lambda_{\min}$ independent Finite Time bound.

Proof.

∘ Restarting from

$$\mathbb{E}[\|\theta_{t+1}-\theta^\star\|^2] \le \mathbb{E}[\|\theta_t-\theta^\star\|^2] - 2\eta(1-\gamma)\mathbb{E}[\|V_{\theta_t}-V_{\theta^\star}\|_\rho^2] + 4\eta^2\mathbb{E}[\|V_{\theta_t}-V_{\theta^\star}\|_\rho^2] + \eta^2\sigma^2$$

∘ We set $\eta \le \frac{1-\gamma}{4}$ and we can rearrange the term as follows

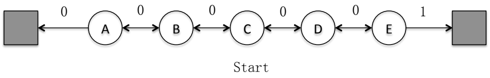
$$\mathbb{E}[\|V_{\theta_t}-V_{\theta^\star}\|_\rho^2] \le \frac{1}{\eta(1-\gamma)}\left(\mathbb{E}[\|\theta_t-\theta^\star\|^2] - \mathbb{E}[\|\theta_{t+1}-\theta^\star\|^2]\right) + \eta\frac{\sigma^2}{1-\gamma}$$

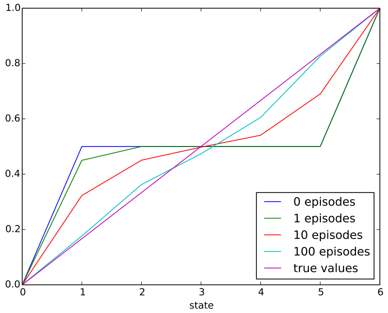
∘ Finally averaging over $t = 1, \ldots, T$, we obtain $\frac{1}{T}\sum_{t=1}^T \mathbb{E}[\|V_{\theta_t}-V_{\theta^\star}\|_\rho^2] \le \frac{1}{\eta(1-\gamma)}\mathbb{E}[\|\theta_1-\theta^\star\|^2] + \eta\frac{\sigma^2}{1-\gamma}$ ∘ Setting $\eta = \frac{1-\gamma}{4\sqrt{T}}$, we obtain

$$\frac{1}{T}\sum_{t=1}^T \mathbb{E}[\|V_{\theta_t}-V_{\theta^\star}\|_\rho^2] \le \frac{4}{\sqrt{T}}\mathbb{E}[\|\theta_1-\theta^\star\|^2] + \frac{1}{2(1-\gamma)^2\sqrt{T}} \le \frac{9d}{2(1-\gamma)^2\sqrt{T}}$$

□

Numerical example: Random walk





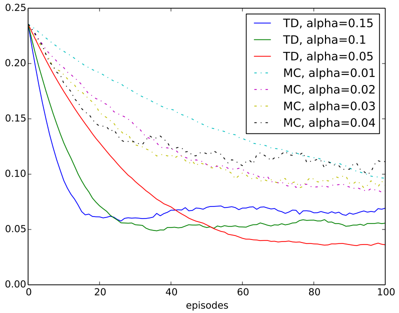


Figure: Left: values learned after various number of updates in a single run of TD(0). Right: the root mean-squared (RMS) error between the value functions learned and the true values. [16]

Bias-variance trade-off

- MC return is **unbiased**, but has **higher variance** since it relies on many random steps
- TD target is **biased**, but has **lower variance** since it only relies on the next step
- The MC error can be written as a sum of TD errors:

$$\begin{aligned} G_t - V(s_t) &= r_{t+1} + \gamma G_{t+1} - V(s_t) + \gamma V(s_{t+1}) - \gamma V(s_{t+1}) \\ &= \delta_t + \gamma(G_{t+1} - V(s_{t+1})) \\ &= \delta_t + \gamma \delta_{t+1} + \gamma^2 (G_{t+2} - V(s_{t+2})) \\ &= \cdots \\ &= \sum_{k=t}^{\infty} \gamma^{k-t} \delta_k \end{aligned}$$

Multiple-step TD learning

Definition (n-step return)

Let T be the termination time step in a given episode, $\gamma \in [0,1]$.

$$\begin{aligned}
G_t^{(1)} &= r_{t+1} + \gamma V(s_{t+1}) && \textit{TD(0)} \\
G_t^{(2)} &= r_{t+1} + \gamma r_{t+2} + \gamma^2 V(s_{t+2}) && \textit{(two-step return)} \\
G_t^{(n)} &= r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n}) && \textit{(n-step return)} \\
G_t^{(\infty)} &= r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{T-t-1} r_T && \textit{MC}
\end{aligned}$$

Note that $G_t^{(n)} = G_t^{(\infty)}$ if $t+n \geq T$.

Multiple-step TD learning

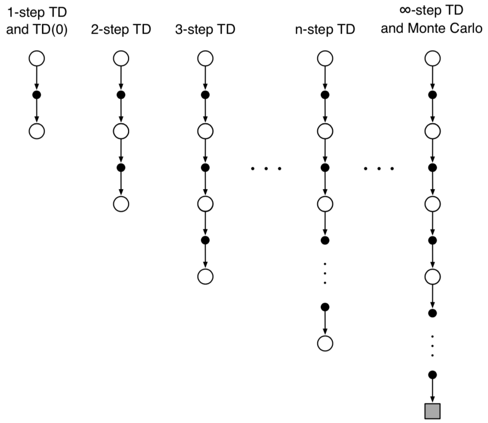


Figure: [17]

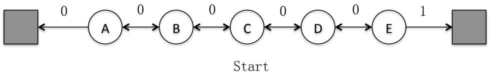
Multiple-step TD learning

Multi-step TD learning:

$$V(s_t) \leftarrow V(s_t) + \alpha_t \Big(\underbrace{G_t^{(n)} - V(s_t)}_{n\text{-step TD error}} \Big)$$

Observations:
- Unifies and combines TD(0) and MC: $n = 1$ recovers TD(0) and $n = \infty$ recovers MC.
- Trades-off bias and variance.
- However, we need to observe $r_{t+1}, \cdots, r_{t+n}$.

Numerical example: Longer random walk



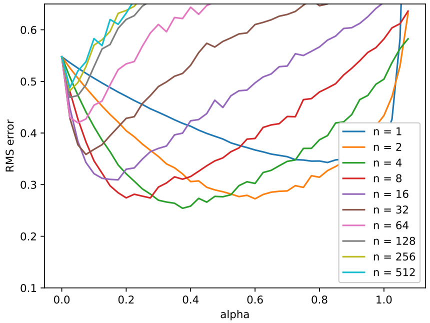


Figure: Performance of n-step TD methods as a function of α, for various values of n, on a 19-state random walk task. [16]

Further extension: TD(λ) with eligibility trace

λ-return (weighted average of all n-step returns)

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_t^{(n)}$$

TD(λ)

$$V(s_t) \leftarrow V(s_t) + \alpha\left[G_t^{\lambda} - V(s_t)\right]$$

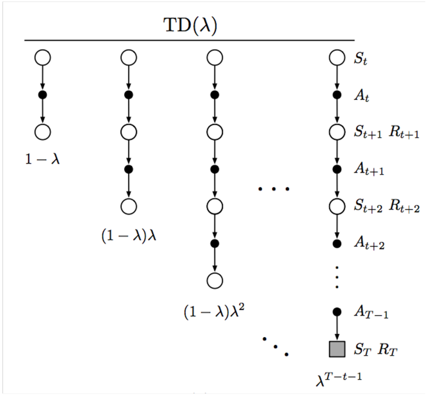


Figure: [17]

Further extension: TD(λ) with eligibility trace

TD(λ)

$$V(s_t) \leftarrow V(s_t) + \alpha \left[G_t^\lambda - V(s_t)\right]$$

Observations:

- $\lambda = 0$ reduces to TD(0); $\lambda = 1$ reduces to MC.
- Can be efficiently implemented:

$$\begin{aligned} V(s) &\leftarrow V(s) + \alpha \delta_t e_t(s) \\ e_t(s) &= \gamma \lambda e_{t-1}(s) + \mathbb{1}\{s_t = s\} \end{aligned}$$

- The term $e_t(s) = \sum_{k=0}^{t} \gamma^{t-k} \mathbb{1}\{s_t = s\}$ is called the eligibility trace.
- Converge faster than TD(0) when λ is appropriately chosen.