A PRIMAL-DUAL VIEW OF REINFORCEMENT LEARNING

Gergely Neu
Univ. Pompeu Fabra
A PRIMAL-DUAL VIEW OF REINFORCEMENT LEARNING

Gergely Neu
Univ. Pompeu Fabra
A PRIMAL-DUAL VIEW OF REINFORCEMENT LEARNING

GERGELY NEU
UNIV. POMPEU FABRA
WHAT IS REINFORCEMENT LEARNING?

Agent

In state $s$, take action $a$

Environment

Reward $r$, new state $s'$

- maximize reward
- in a reactive environment
- under partial feedback
RL EXAMPLE 0.
RL EXAMPLE 0.
RL EXAMPLE 0.
RL EXAMPLE 0.
RL EXAMPLE 0.
RL EXAMPLE 0.

- State
- Actions
- Next State
- Reward
RL EXAMPLE 0.

State → Actions → Next State

Partial observability
RL EXAMPLE 0.

state → actions → next state

Partial observability

Reward?
WHY SHOULD I CARE?
WHY SHOULD I CARE?
WHY SHOULD I CARE?

Breakthrough in Atari game playing

- State: pixels on screen
- Actions: joystick
- State transitions: game dynamics
- Reward: score in game
WHY SHOULD I CARE?

Breakthrough in Go
WHY SHOULD I CARE?

Breakthrough in Go

- State: stones currently on board
- Actions: place stone on board
- State transitions: own move + adversary’s move
- Reward: +1 for winning the game
WHY SHOULD I CARE?

Breakthrough in Go

Autonomous driving
WHY SHOULD I CARE?

Breakthrough in Go

• State: stones currently on board
• Actions: place stone on board
• State transitions: own move + adversary's move
• Reward: +1 for winning the game

Autonomous driving

• State: road conditions, other vehicles, obstacles,…
• Actions: turn left/right, accelerate/brake,…
• State transitions: depending on state+action+randomness
• Reward: +100 for reaching destination, -100 for accidents,…
RECOMMENDED READING

  • For an enjoyable (but not very rigorous) introduction

• Dimitri Bertsekas (2012): “Dynamic Programming and Optimal Control”
  • For a rigorous treatment of the basics

• Csaba Szepesvári (2012): “Algorithms for RL”
  • For a rigorous description of basic RL algorithms
THIS SHORT COURSE: A PRIMAL-DUAL VIEW

• Markov decision processes
  • Value functions and optimal policies

• Primal view: Dynamic programming
  • Policy evaluation, value and policy iteration
  • Value-function-based methods
    • Temporal differences, Q-learning, LSTD, deep Q networks,…
THIS SHORT COURSE: A PRIMAL-DUAL VIEW

• Markov decision processes
  • Value functions and optimal policies

• Primal view: Dynamic programming
  • Policy evaluation, value and policy iteration
  • Value-function-based methods
    • Temporal differences, Q-learning, LSTD, deep Q networks,…

• Dual view: Linear programming
  • LP duality in MDPs
  • Direct policy optimization methods
    • Policy gradients, REPS, TRPO,…
THIS SHORT COURSE: A PRIMAL-DUAL VIEW

- Markov decision processes
  - Value functions and optimal policies

Primal view: Dynamic programming
- Policy evaluation, value and policy iteration
- Value-function-based methods
  - Temporal differences, Q-learning, LSTD, deep Q networks,…

Dual view: Linear programming
- LP duality in MDPs
- Direct policy optimization methods
  - Policy gradients, REPS, TRPO,…
THIS SHORT COURSE:
A PRIMAL-DUAL VIEW

• Markov decision processes
  • Value functions and optimal policies

• Primal view: Dynamic programming
  • Policy evaluation, value and policy iteration
  • Value-function-based methods
    • Temporal differences, Q-learning, LSTD, deep Q networks,…

• Dual view: Linear programming
  • LP duality in MDPs
  • Direct policy optimization methods
    • Policy gradients, REPS, TRPO,…
A Markov Decision Process (MDP) is characterized by
- $X$: a set of states
- $A$: a set of actions, possibly different in each state
- $P: X \times A \times X \rightarrow [0,1]$: a transition function with $P(\cdot | x, a)$ being the distribution of the next state given previous state $x$ and action $a$:
  $$P[x_{t+1} = x'| x_t = x, a_t = a] = P(x'| x, a)$$
- $r: X \times A \rightarrow [0,1]$: a reward function
A Markov Decision Process (MDP) is characterized by \((X, A, P, r)\)

- **\(X\):** a set of states
- **\(A\):** a set of actions, possibly different in each state
- **\(P\):** \(X \times A \times X \rightarrow [0,1]\): a transition function with \(P(\cdot | x, a)\) being the distribution of the next state given previous state \(x\) and action \(a\):
  \[
P[x_{t+1} = x'|x_t = x, a_t = a] = P(x'|x, a)
\]
- **\(r\):** \(X \times A \rightarrow [0,1]\): a reward function
A Markov Decision Process (MDP) is characterized by $(X, A, P, r)$

Interaction in an MDP: in each round $t = 1, 2, \ldots$

- Agent observes state $x_t$ and selects action $a_t$
- Environment moves to state $x_{t+1} \sim P(\cdot | x_t, a_t)$
- Agent receives reward $r_t$ such that $\mathbb{E}[r_t | x_t, a_t] = r(x_t, a_t)$
A Markov Decision Process (MDP) is characterized by $(X, A, P, r)$

Interaction in an MDP: in each round $t = 1, 2, ...$

- Agent observes state $x_t$ and selects action $a_t$
- Environment moves to state $x_{t+1} \sim P(\cdot | x_t, a_t)$
- Agent receives reward $r_t$ such that $\mathbb{E}[r_t | x_t, a_t] = r(x_t, a_t)$

**GOAL:**
maximize “total rewards”!
NOTIONS OF “TOTAL REWARD”

Episodic MDPs:
- There is a terminal state $x^*$
- **GOAL:** maximize total reward until final round $T$ when $x^*$ is reached:

$$ R^* = \mathbb{E}[\sum_{t=0}^{T} r_t] $$
NOTIONS OF “TOTAL REWARD”

Episodic MDPs:
- There is a terminal state $x^*$
- **GOAL:** maximize total reward until final round $T$
  when $x^*$ is reached:

$$ R^* = \mathbf{E}[\sum_{t=0}^{T} r_t] $$

Discounted MDPs:
- No terminal state
- Discount factor $\gamma \in (0,1)$
- **GOAL:** maximize total discounted reward

$$ R_\gamma = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t r_t] $$
NOTIONS OF “TOTAL REWARD”

Episodic MDPs:
- There is a terminal state $x^*$
- **GOAL:** maximize total reward until final round $T$ when $x^*$ is reached:
  \[ R^* = \mathbb{E}[\sum_{t=0}^{T} r_t] \]

Discounted MDPs:
- No terminal state
- Discount factor $\gamma \in (0,1)$
- **GOAL:** maximize total discounted reward
  \[ R_\gamma = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t] \]
NOTIONS OF “TOTAL REWARD”

Episodic MDPs:
- There is a terminal state $x^*$
- **GOAL:** maximize total reward until final round $T$ when $x^*$ is reached:
  $$R^* = \mathbb{E}[\sum_{t=0}^{T} r_t]$$

Discounted MDPs:
- No terminal state
- Discount factor $\gamma \in (0,1)$
- **GOAL:** maximize total discounted reward
  $$R_\gamma = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$$

+ other notions:
  - long-term average reward (part 2?)
  - total reward up to fixed horizon
  - ...

+ we will assume that $X$ and $A$ are finite
POLICIES AND TRAJECTORY DISTRIBUTIONS

Policy: mapping from histories to actions

\[ \pi : x_1, a_1, x_2, a_2, \ldots, x_t \mapsto a_t \]
**POLICIES AND TRAJECTORY DISTRIBUTIONS**

**Policy:** mapping from histories to actions

\[ \pi : x_1, a_1, x_2, a_2, \ldots, x_t \mapsto a_t \]

**Stationary policy:** mapping from states to actions
(no dependence on history or \( t \))

\[ \pi : x \mapsto a \]
**Policies and Trajectory Distributions**

**Policy:** mapping from histories to actions

\[ \pi: x_1, a_1, x_2, a_2, \ldots, x_t \mapsto a_t \]

**Stationary policy:** mapping from states to actions
(no dependence on history or \( t \))

\[ \pi: x \mapsto a \]

Let \( \tau = (x_1, a_1, x_2, a_2, \ldots) \) be a trajectory generated by running \( \pi \) in the MDP \( \tau \sim (\pi, P) \):

- \( a_t = \pi(x_t, a_{t-1}, x_{t-1}, \ldots, x_1) \)
- \( x_{t+1} \sim P(\cdot | x_t, a_t) \)
Policies and Trajectory Distributions

**Policy:** mapping from histories to actions
\[ \pi: x_1, a_1, x_2, a_2, \ldots, x_t \mapsto a_t \]

**Stationary policy:** mapping from states to actions
(no dependence on history or \( t \))
\[ \pi: x \mapsto a \]

Let \( \tau = (x_1, a_1, x_2, a_2, \ldots) \) be a trajectory generated by running \( \pi \) in the MDP \( \tau \sim (\pi, P) \):
- \( a_t = \pi(x_t, a_{t-1}, x_{t-1}, \ldots, x_1) \)
- \( x_{t+1} \sim P(\cdot | x_t, a_t) \)

Expectation under this distribution: \( \mathbb{E}_\pi[\cdot] \)
Optimal policy $\pi^*$: a policy that maximizes

$$E_\pi[R_\gamma] = E_\pi \left[ \sum_{t=0}^{\infty} \gamma^t r_t \right]$$
Optimal policy $\pi^*$: a policy that maximizes

$$E_\pi[R_\gamma] = E_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

Theorem
There exists a deterministic optimal policy $\pi^*$ such that

$$\pi^*(x_1, a_1, \ldots, x_t) = \pi^*(x_t)$$
Optimal policy $\pi^*$: a policy that maximizes

$$E_\pi[R_\gamma] = E_\pi \left[ \sum_{t=0}^{\infty} \gamma^t r_t \right]$$

**Theorem**
There exists a deterministic optimal policy $\pi^*$ such that

$$\pi^*(x_1, a_1, \ldots, x_t) = \pi^*(x_t)$$

**Consequence:** it’s enough to study stationary policies

$$\pi: x \mapsto a$$
DEFINING OPTIMALITY

**Theorem**
There exists a deterministic optimal policy $\pi^*$ such that
$$\pi^*(x_1, a_1, \ldots, x_t) = \pi^*(x_t)$$

**Consequence:** it’s enough to study stationary policies $\pi: x \mapsto a$

**Intuitive “proof”:** Future transitions $x_{t+1} \sim P(\cdot | x_t, a_t)$ do not depend on the previous states $x_1, x_2, \ldots$
DEFINING OPTIMALITY

Theorem
There exists a deterministic optimal policy $\pi^*$ such that
$$\pi^*(x_1, a_1, ..., x_t) = \pi^*(x_t)$$

Consequence: it’s enough to study stationary policies
$$\pi: x \mapsto a$$

Intuitive “proof”: Future transitions $x_{t+1} \sim P(\cdot | x_t, a_t)$ do not depend on the previous states $x_1, x_2, ...$

= “Markov property”
THIS SHORT COURSE: A PRIMAL-DUAL VIEW

- Markov decision processes
- Value functions and optimal policies

**Primal view: Dynamic programming**
- Policy evaluation, value and policy iteration
- Value-function-based methods
  - Temporal differences, Q-learning, LSTD, deep Q networks,…

**Dual view: Linear programming**
- LP duality in MDPs
- Direct policy optimization methods
  - Policy gradients, REPS, TRPO,…
Value function: evaluates policy $\pi$ starting from state $x$:

$$V^\pi(x) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t r_t \mid x_0 = x \right]$$
VALUE FUNCTIONS

**Value function:** evaluates policy $\pi$ starting from state $x$:

$$V^\pi(x) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t r_t \mid x_0 = x \right]$$

**Action-value function:** evaluates policy $\pi$ starting from state $x$ and action $a$:

$$Q^\pi(x, a) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t r_t \mid x_0 = x, a_0 = a \right]$$
VALUE FUNCTIONS

Value function: evaluates policy \( \pi \) starting from state \( x \):
\[
V^\pi(x) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t r_t \mid x_0 = x \right]
\]

Action-value function: evaluates policy \( \pi \) starting from state \( x \) and action \( a \):
\[
Q^\pi(x, a) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t r_t \mid x_0 = x, a_0 = a \right]
\]

“Optimal policy \( \pi^* \) 

\[= \text{arg max}_\pi V^\pi(x_0) \]"
Theorem
There exists a policy $\pi^*$ that satisfies
$$V^{\pi^*}(x) = \max_{\pi} V^{\pi}(x) \quad (\forall x)$$
Theorem
There exists a policy \( \pi^* \) that satisfies
\[
V^{\pi^*}(x) = \max_{\pi} V^\pi(x) \quad (\forall x)
\]
VALUE FUNCTIONS
AND THE OPTIMAL POLICY

Theorem
There exists a policy $\pi^*$ that satisfies

$$V^{\pi^*}(x) = \max_{\pi} V^{\pi}(x) \quad (\forall x)$$

Optimal policy: a policy $\pi^*$ that satisfies the above
**Theorem**

There exists a policy $\pi^*$ that satisfies

$$V^{\pi^*}(x) = \max_{\pi} V^{\pi}(x) \quad (\forall x)$$

**Optimal policy:** a policy $\pi^*$ that satisfies the above

**The optimal value function:**

$$V^* = V^{\pi^*}$$
**Theorem**

The value function of a stationary policy $\pi$ satisfies the system of equations ($\forall x \in X$)

$$V^\pi(x) = r(x, \pi(x)) + \gamma \sum_y P(y|x, \pi(x)) V^\pi(y)$$
**Theorem**

The value function of a stationary policy $\pi$ satisfies the system of equations ($\forall x \in X$)

$$V^\pi(x) = r(x, \pi(x)) + \gamma \sum_y P(y|x, \pi(x)) V^\pi(y)$$

**Proof:**

$$V^\pi(x) = E_\pi[\sum_{t=0}^\infty \gamma^t r(x_t, a_t) | x_0 = x]$$
The Bellman Equations

**Theorem**

The value function of a stationary policy $\pi$ satisfies the system of equations $(\forall x \in X)$

$$V^\pi(x) = r(x, \pi(x)) + \gamma \sum_y P(y|x, \pi(x)) V^\pi(y)$$

**Proof:**

$$V^\pi(x) = E_\pi[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t) | x_0 = x]$$

$$= r(x, \pi(x)) + E_\pi[\sum_{t=1}^{\infty} \gamma^t r(x_t, a_t) | x_0 = x]$$
THE BELLMAN EQUATIONS

Theorem

The value function of a stationary policy \( \pi \) satisfies the system of equations (\( \forall x \in X \))

\[
V^\pi(x) = r(x, \pi(x)) + \gamma \sum_y P(y|x, \pi(x)) V^\pi(y)
\]

Proof:

\[
V^\pi(x) = E_{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t r(x_t, a_t) \mid x_0 = x \right]
\]

\[
= r(x, \pi(x)) + E_{\pi} \left[ \sum_{t=1}^{\infty} \gamma^t r(x_t, a_t) \mid x_0 = x \right]
\]

\[
= r(x, \pi(x)) + \gamma \sum_y P(y|x, \pi(x)) E_{\pi} \left[ \sum_{t=1}^{\infty} \gamma^{t-1} r(x_t, a_t) \mid x_1 = y \right]
\]
THE BELLMAN EQUATIONS

Theorem
The value function of a stationary policy $\pi$ satisfies the system of equations ($\forall x \in X$)

$$V^\pi(x) = r(x, \pi(x)) + \gamma \sum_y P(y|x, \pi(x)) V^\pi(y)$$

Proof:

$$V^\pi(x) = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t r(x_t, a_t) | x_0 = x]$$

$$= r(x, \pi(x)) + \mathbb{E}_\pi[\sum_{t=1}^\infty \gamma^t r(x_t, a_t) | x_0 = x]$$

$$= r(x, \pi(x)) + \gamma \sum_y P(y|x, \pi(x)) \mathbb{E}_\pi[\sum_{t=1}^\infty \gamma^{t-1} r(x_t, a_t) | x_1 = y]$$

$$= r(x, \pi(x)) + \gamma \sum_y P(y|x, \pi(x)) V^\pi(y) \quad \square$$
The optimal value function satisfies the system of equations

$$V^*(x) = \max_a \left\{ r(x, a) + \gamma \sum_y P(y|x, a) V^*(y) \right\}$$
THE BELLMAN OPTIMALITY EQUATIONS

Theorem
The optimal value function satisfies the system of equations

\[ V^*(x) = \max_a \left\{ r(x, a) + \gamma \sum_y P(y|x, a) V^*(y) \right\} \]

Theorem
An optimal policy \( \pi^* \) satisfies

\[ \pi^*(x) \in \arg \max_a \left\{ r(x, a) + \gamma \sum_y P(y|x, a) V^*(y) \right\} \]
Theorem
The optimal action-value function satisfies

\[ Q^*(x, a) = r(x, a) + \gamma \sum_y P(y|x, a) \max_b Q^*(y, b) \]
Theorem
The optimal action-value function satisfies

\[ Q^*(x, a) = r(x, a) + \gamma \sum_y P(y|x, a) \max_b Q^*(y, b) \]

Theorem
An optimal policy \( \pi^* \) satisfies

\[ \pi^*(x) \in \arg \max_a Q^*(x, a) \]
Theorem

The optimal action-value function satisfies

\[ Q^*(x, a) = r(x, a) + \gamma \sum_y P(y|x, a) \max_b Q^*(y, b) \]

Theorem

An optimal policy \( \pi^* \) satisfies

\[ \pi^*(x) \in \arg \max_a Q^*(x, a) \]

= greedy with respect to \( Q^* \)
SHORT SUMMARY SO FAR

So far, we have characterized
- The value functions of a given policy
- The optimal policy through value functions
- The optimal value functions
- The optimal policy through the optimal value functions
So far, we have characterized
- The value functions of a given policy
- The optimal policy through value functions
- The optimal value functions
- The optimal policy through the optimal value functions

BUT HOW DO WE FIND THE OPTIMAL VALUE FUNCTION??

… also, is there a way to clean up this mess? See part 2!
EASY ANSWER FOR FINITE-HORIZON PROBLEMS

Bae: Come over
Dijkstra: But there are so many routes to take and I don’t know which one’s the fastest
Bae: My parents aren’t home
Dijkstra:

Dijkstra's algorithm

Graph search algorithm

Not to be confused with Dykstra’s projection algorithm.

Dijkstra’s algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later.\[1\][2]

The algorithm exists in many variants; Dijkstra’s original variant found the shortest path between two nodes,\[2\] but a more common variant fixes a single node as the “source” node and finds shortest paths from the source to all other nodes in the graph, producing a shortest-path tree.
THIS SHORT COURSE: A PRIMAL-DUAL VIEW

- Markov decision processes
  - Value functions and optimal policies

- Primal view: Dynamic programming
  - Policy evaluation, value and policy iteration
  - Value-function-based methods
    - Temporal differences, Q-learning, LSTD, deep Q networks,…

- Dual view: Linear programming
  - LP duality in MDPs
  - Direct policy optimization methods
    - Policy gradients, REPS, TRPO,…
Dynamic programming

= computing value functions through repeated use of the “Bellman operators”
Bellman operator $T^\pi$: maps a function $V \in \mathbb{R}^X$ to another function $T^\pi V \in \mathbb{R}^X$:

$$(T^\pi V)(x) = r(x, \pi(x)) + \gamma \sum_y P(y|x, \pi(x))V(y)$$
THE BELLMAN OPERATOR

Bellman operator $T^\pi$:
maps a function $V \in \mathbb{R}^X$ to another function $T^\pi V \in \mathbb{R}^X$:

$$(T^\pi V)(x) = r(x, \pi(x)) + \gamma \sum_y P(y|x, \pi(x)) V(y)$$

r.h.s. of BE
**THE BELLMAN OPERATOR**

Bellman operator $T^\pi$:
maps a function $V \in \mathbb{R}^X$ to another function $T^\pi V \in \mathbb{R}^X$

$$(T^\pi V)(x) = r(x, \pi(x)) + \gamma \sum_y P(y|x, \pi(x))V(y)$$

**The Bellman Equations:**

$$V^\pi(x) = r(x, \pi(x)) + \gamma \sum_y P(y|x, \pi(x)) V^\pi(y)$$
The Bellman Operator

Bellman operator $T^\pi$: maps a function $V \in \mathbb{R}^X$ to another function $T^\pi V \in \mathbb{R}^X$

$(T^\pi V)(x) = r(x, \pi(x)) + \gamma \sum_y P(y|x, \pi(x))V(y)$

The Bellman Equations:

$V^\pi = T^\pi V^\pi$
Bellman operator $T^\pi$: maps a function $V \in \mathbb{R}^X$ to another function $T^\pi V \in \mathbb{R}^X$

$$(T^\pi V)(x) = r(x, \pi(x)) + \gamma \sum_y P(y|x, \pi(x))V(y)$$

$V^\pi$ is the fixed point of $T^\pi$

The Bellman Equations:

$$V^\pi = T^\pi V^\pi$$
POLICY EVALUATION USING THE BELLMAN OPERATOR

Idea: repeated application of $T^\pi$ on any function $V_0$ should converge to $V^\pi$ ...
POLICY EVALUATION USING THE BELLMAN OPERATOR

Idea: repeated application of $T^\pi$ on any function $V_0$ should converge to $V^\pi$...

...and it works!!

Power iteration

Input: arbitrary $V_0: X \to \mathbb{R}$ and $\pi$
For $k = 1, 2, \ldots$, compute
\[ V_{k+1} = T^\pi V_k \]
Idea: repeated application of $T^\pi$ on any function $V_0$ should converge to $V^\pi$ ...

...and it works!!

Power iteration

Input: arbitrary $V_0: X \rightarrow \mathbb{R}$ and $\pi$

For $k = 1, 2, \ldots$, compute

$$V_{k+1} = T^\pi V_k$$

Theorem: $\lim_{k \rightarrow \infty} V_k = V^\pi$
CONVERGENCE OF POWER ITERATION: PROOF SKETCH

- Power iteration can be written as the linear recursion

\[ V_{k+1} = r + \gamma P^\pi V_k \]
CONVERGENCE OF POWER ITERATION:
PROOF SKETCH

- Power iteration can be written as the linear recursion

\[ V_{k+1} = r + \gamma P^\pi V_k = r + \gamma P^\pi (r + \gamma P^\pi V_{k-1}) \]
CONVERGENCE OF POWER ITERATION: PROOF SKETCH

- Power iteration can be written as the linear recursion

\[ V_{k+1} = r + \gamma P^\pi V_k = r + \gamma P^\pi (r + \gamma P^\pi V_{k-1}) \]
\[ = r + \gamma P^\pi r + (\gamma P^\pi)^2 r + \cdots + (\gamma P^\pi)^k r \]
CONVERGENCE OF POWER ITERATION: PROOF SKETCH

- Power iteration can be written as the linear recursion

\[ V_{k+1} = r + \gamma P^\pi V_k = r + \gamma P^\pi (r + \gamma P^\pi V_{k-1}) \]
\[ = r + \gamma P^\pi r + (\gamma P^\pi)^2 r + \cdots + (\gamma P^\pi)^k r \]
\[ = \sum_{t=0}^{k} (\gamma P^\pi)^k r \]
CONVERGENCE OF POWER ITERATION: PROOF SKETCH

- Power iteration can be written as the linear recursion

\[ V_{k+1} = r + \gamma P^\pi V_k = r + \gamma P^\pi (r + \gamma P^\pi V_{k-1}) \]

\[ = r + \gamma P^\pi r + (\gamma P^\pi)^2 r + \cdots + (\gamma P^\pi)^k r \]

\[ = \sum_{t=0}^{k} (\gamma P^\pi)^t r \]

\[ = (I - \gamma P^\pi)^{-1} \cdot (I - (\gamma P^\pi)^k)r \]

Geometric sum! (von Neumann series)
CONVERGENCE OF POWER ITERATION: PROOF SKETCH

- Power iteration can be written as the linear recursion
  \[ V_{k+1} = r + \gamma P^\pi V_k = r + \gamma P^\pi (r + \gamma P^\pi V_{k-1}) \]
  \[ = r + \gamma P^\pi r + (\gamma P^\pi)^2 r + \cdots + (\gamma P^\pi)^k r \]
  \[ = \sum_{t=0}^{k} (\gamma P^\pi)^t r \]
  \[ = (I - \gamma P^\pi)^{-1} \cdot (I - (\gamma P^\pi)^k)r \]
  \[ \rightarrow (I - \gamma P^\pi)^{-1} r \quad (k \rightarrow \infty) \]

Geometric sum! (von Neumann series)
CONVERGENCE OF POWER ITERATION: PROOF SKETCH

- Power iteration can be written as the linear recursion

\[
V_{k+1} = r + \gamma P^\pi V_k = r + \gamma P^\pi (r + \gamma P^\pi V_{k-1}) \\
= r + \gamma P^\pi r + (\gamma P^\pi)^2 r + \cdots + (\gamma P^\pi)^k r \\
= \sum_{t=0}^{k} (\gamma P^\pi)^k r \\
= (I - \gamma P^\pi)^{-1} \cdot (I - (\gamma P^\pi)^k)r \\
\rightarrow (I - \gamma P^\pi)^{-1}r \\
(k \rightarrow \infty)
\]

- The value function \( V^\pi \) satisfies

\[
V^\pi = r + \gamma P^\pi V^\pi \iff V^\pi = (I - \gamma P^\pi)^{-1}r
\]
POWER ITERATION IN ACTION

Gridworld MDP
Power Iteration in Action

Gridworld MDP

- **State:** location on the grid
- **Actions:** try to move in one of 8 directions or stay put
- **Transition probabilities:**
  - move successfully w.p. $p = 0.5$
  - otherwise move in neighboring direction
**POWER ITERATION IN ACTION**

**Gridworld MDP**

- **State:** location on the grid
- **Actions:** try to move in one of 8 directions or stay put
- **Transition probabilities:**
  - move successfully w.p. $p = 0.5$
  - otherwise move in neighboring direction
POWER ITERATION IN ACTION

Uniform policy:

\[ \pi(a|x) = \frac{1}{9} \]

for all actions \( a \in \{1, 2, \ldots, 9\} \)
Power Iteration in Action

Uniform policy:

\[ \pi(a|x) = \frac{1}{9} \]

for all actions \( a \in \{1, 2, \ldots, 9\} \)
POWER ITERATION IN ACTION

Uniform policy:

$$\pi(a|x) = \frac{1}{9}$$

for all actions $$a \in \{1, 2, \ldots, 9\}$$
POWER ITERATION IN ACTION

Uniform policy:

\[ \pi(a|x) = \frac{1}{9} \]

for all actions \( a \in \{1, 2, \ldots, 9\} \)
POWER ITERATION IN ACTION

Uniform policy:

\[ \pi(a|x) = \frac{1}{9} \]

for all actions \( a \in \{1,2, \ldots, 9\} \)
“Upwards” policy:
\[ \pi(\text{up}|x) = 1 \]
POWER ITERATION IN ACTION

“Upwards” policy:
\[ \pi(\text{up}|x) = 1 \]
POWER ITERATION IN ACTION

“Upwards” policy:
\[ \pi(\text{up}|x) = 1 \]
POWER ITERATION IN ACTION

Vhat_{up}, iteration 10

“Upwards” policy:
\[ \pi(up|x) = 1 \]
The Bellman Optimality Operator

Bellman optimality operator $T^*$: maps a function $V \in \mathbb{R}^X$ to another function $T^*V \in \mathbb{R}^X$:

$$(T^*V)(x) = \max_a \{ r(x, a) + \gamma \sum_y P(y|x, a)V(y) \}$$
Bellman optimality operator $T^*$: 
maps a function $V \in \mathbb{R}^X$ to another function $T^*V \in \mathbb{R}^X$: 
$$(T^*V)(x) = \max_{a} \{ r(x, a) + \gamma \sum_y P(y|x, a)V(y) \}$$

r.h.s. of BOE
Bellman optimality operator $T^*$: maps a function $V \in \mathbb{R}^X$ to another function $T^*V \in \mathbb{R}^X$: 

$$(T^*V)(x) = \max_a \{r(x, a) + \gamma \sum_y P(y|x, a)V(y)\}$$

The Bellman Optimality Equations:

$$V^*(x) = \max_a \{r(x, a) + \gamma \sum_y P(y|x, a)V^*(y)\}$$
The Bellman Optimality Operator

Bellman optimality operator $T^*$: maps a function $V \in \mathbb{R}^X$ to another function $T^*V \in \mathbb{R}^X$:

$$(T^*V)(x) = \max_a \{ r(x, a) + \gamma \sum_y P(y|x, a)V(y) \}$$

$V^*$ is the fixed point of $T^*$

The Bellman Optimality Equations:

$$V^* = T^*V^*$$

r.h.s. of BOE
VALUE ITERATION

Idea: repeated application of $T^*$ on any function $V_0$ should converge to $V^*$ ...

...and it works!!
VALUE ITERATION

Idea: repeated application of $T^*$ on any function $V_0$ should converge to $V^*$...

...and it works!!

Value iteration

Input: arbitrary function $V_0: X \rightarrow \mathbb{R}$

For $k = 1, 2, \ldots$, compute

$$V_{k+1} = T^*V_k$$
**VALUE ITERATION**

**Idea:** repeated application of $T^*$ on any function $V_0$ should converge to $V^*$ ...

...and it works!!

**Value iteration**

**Input:** arbitrary function $V_0: X \to \mathbb{R}$

For $k = 1, 2, \ldots$, compute

$$V_{k+1} = T^*V_k$$

**Theorem:** $\lim_{k \to \infty} V_k = V^*$
Key idea: $T^*$ is a contraction

- for any two functions $V$ and $V'$, we have
  $$\|T^*V - T^*V'\|_\infty \leq \gamma \|V - V'\|_\infty$$
THE CONVERGENCE OF VALUE ITERATION: PROOF SKETCH

Key idea: $T^*$ is a contraction

- for any two functions $V$ and $V'$, we have
  \[ \|T^*V - T^*V'\|_\infty \leq \gamma \|V - V'\|_\infty \]

- repeated application gives
  \[ \|V_{k+1} - V^*\|_\infty = \|T^*V_k - T^*V^*\|_\infty \]
THE CONVERGENCE OF VALUE ITERATION: PROOF SKETCH

Key idea: $T^*$ is a contraction

- for any two functions $V$ and $V'$, we have
  $$\|T^*V - T^*V'\|_\infty \leq \gamma \|V - V'\|_\infty$$
- repeated application gives
  $$\|V_{k+1} - V^*\|_\infty = \|T^*V_k - T^*V^*\|_\infty \leq \gamma \|V_k - V^*\|_\infty$$
THE CONVERGENCE OF VALUE ITERATION: PROOF SKETCH

Key idea: \( T^* \) is a contraction

- for any two functions \( V \) and \( V' \), we have
  \[
  \|T^*V - T^*V'\|_\infty \leq \gamma \|V - V'\|_\infty
  \]

- repeated application gives
  \[
  \|V_{k+1} - V^*\|_\infty = \|T^*V_k - T^*V^*\|_\infty
  \leq \gamma \|V_k - V^*\|_\infty
  \leq \gamma^2 \|V_{k-1} - V^*\|_\infty
  \]
THE CONVERGENCE OF VALUE ITERATION: PROOF SKETCH

Key idea: \( T^* \) is a contraction

- for any two functions \( V \) and \( V' \), we have
  \[
  \|T^*V - T^*V'\|_\infty \leq \gamma \|V - V'\|_\infty
  \]

- repeated application gives
  \[
  \|V_{k+1} - V^*\|_\infty = \|T^*V_k - T^*V^*\|_\infty \\
  \leq \gamma \|V_k - V^*\|_\infty \\
  \leq \gamma^2 \|V_{k-1} - V^*\|_\infty \\
  \leq \cdots \leq \gamma^k \|V_0 - V^*\|_\infty
  \]
The Convergence of Value Iteration: Proof Sketch

**Key idea:** $T^*$ is a contraction

- for any two functions $V$ and $V'$, we have
  $$\|T^*V - T^*V'\|_\infty \leq \gamma \|V - V'\|_\infty$$

- repeated application gives
  $$\|V_{k+1} - V^*\|_\infty = \|T^*V_k - T^*V^*\|_\infty \leq \gamma \|V_k - V^*\|_\infty \leq \gamma^2 \|V_{k-1} - V^*\|_\infty \leq \cdots \leq \gamma^k \|V_0 - V^*\|_\infty$$

- thus
  $$\lim_{k \to \infty} \|V_{k+1} - V^*\|_\infty = 0$$
VALUE ITERATION IN ACTION

Gridworld MDP

- **State**: location on the grid
- **Actions**: try to move in one of 8 directions or stay put
- **Transition probabilities**:
  - move successfully w.p. $p = 0.5$
  - otherwise move in neighboring direction

Reward: +100
Reward: +500
VALUE ITERATION IN ACTION

$V_{\text{hat}}^{\text{opt}}$, iteration 0
VALUE ITERATION IN ACTION

$V_{\hat{\pi}^{opt}}$, iteration 1
VALUE ITERATION IN ACTION

$V_{\text{opt}}$, iteration 5
VALUE ITERATION IN ACTION

$V_{\hat{\text{opt}}, \text{iteration } 10}$
VALUE ITERATION IN ACTION

$V_{opt}^{\text{hat}}$, iteration 20
VALUE ITERATION IN ACTION

$V_{\text{hat}_{\text{opt}}}$, iteration 50
VALUE ITERATION IN ACTION

\[ V_{\hat{\pi}_{opt}} \text{, iteration 100} \]
VALUE ITERATION IN ACTION

$V_{\text{hat}}_{\text{opt}}$, iteration 500
VALUE ITERATION IN ACTION
**Greedy policy** with respect to $V$:

$$(GV)(x) = \arg \max_a \{ r(x, a) + \sum_y P(y|x, a)V(x) \}$$
**Greedy policy** with respect to $V$:

$$(GV)(x) = \arg \max_a \{r(x, a) + \sum_y P(y|x, a)V(x)\}$$

Recall: $\pi^* = GV^*$
**Policy Iteration**

**Greedy policy** with respect to $V$:

$$(GV)(x) = \arg \max_a \left\{ r(x, a) + \sum_y P(y|x, a)V(x) \right\}$$

**Policy Iteration**

**Input:** arbitrary function $V_0: X \rightarrow \mathbb{R}$

For $k = 0, 1, \ldots$, compute

$$\pi_k = G(V_k), \quad V_{k+1} = V^{\pi_k}$$

Recall: $\pi^* = GV^*$
**Policy Iteration**

*Input:* arbitrary function $V_0: X \to \mathbb{R}$

For $k = 0, 1, \ldots$, compute

$$\pi_k = G(V_k), \quad V_{k+1} = V^{\pi_k}$$

**Theorem:** $\lim_{k \to \infty} V_k = V^*$

---

**Greedy policy** with respect to $V$:

$$(GV)(x) = \arg \max_a \{ r(x, a) + \sum_y P(y|x, a)V(x) \}$$

Recall: $\pi^* = GV^*$
Key idea: $T^*$ is a contraction

- for any two functions $V$ and $V'$, we have
  \[ \|T^*V - T^*V'\|_\infty \leq \gamma \|V - V'\|_\infty \]

- repeated application gives
  \[
  \|V_{k+1} - V^*\|_\infty = \|T^*V_k - T^*V^*\|_\infty \\
  \leq \gamma \|V_k - V^*\|_\infty \\
  \leq \gamma^2 \|V_{k-1} - V^*\|_\infty \\
  \leq \cdots \leq \gamma^k \|V_0 - V^*\|_\infty
  \]

- thus
  \[ \lim_{k \to \infty} \|V_{k+1} - V^*\|_\infty = 0 \]
THE CONVERGENCE OF VALUE ITERATION: PROOF SKETCH

Key idea: \( T^* \) is a contraction

- for any two functions \( V \) and \( V' \), we have
  \[
  \| T^* V - T^* V' \|_\infty \leq \gamma \| V - V' \|_\infty
  \]

- repeated application gives
  \[
  \| V_{k+1} - V^* \|_\infty = \| T^* V_k - T^* V^* \|_\infty
  \leq \gamma \| V_k - V^* \|_\infty
  \leq \gamma^2 \| V_{k-1} - V^* \|_\infty
  \leq \cdots \leq \gamma^k \| V_0 - V^* \|_\infty
  \]

- thus
  \[
  \lim_{k \to \infty} \| V_{k+1} - V^* \|_\infty = 0
  \]

Just replace \( T^* \) with the operator
\[
B^*: V \mapsto (T^G(V))_\infty
\]
THIS SHORT COURSE:
A PRIMAL-DUAL VIEW

- Markov decision processes
  - Value functions and optimal policies

- Primal view: Dynamic programming
  - Policy evaluation, value and policy iteration
    - Value-function-based methods
      - Temporal differences, Q-learning, LSTD, deep Q networks,…

- Dual view: Linear programming
  - LP duality in MDPs
  - Direct policy optimization methods
    - Policy gradients, REPS, TRPO,…
Policy iteration:

\[ V_k \]
Policy iteration:

\[ \pi_k = GV_k \]
FROM DYNAMIC PROGRAMMING TO VALUE-BASED REINFORCEMENT LEARNING

Policy iteration:

\[ \pi_k = G V_k \]

\[ V_{k+1} = V^{\pi_k} \]
FROM DYNAMIC PROGRAMMING TO VALUE-BASED REINFORCEMENT LEARNING

Policy iteration:

\[
\begin{align*}
\text{improve policy} & : \pi_k = GV_k \\
\text{evaluate policy} & : V_{k+1} = V^{\pi_k}
\end{align*}
\]

Approximate policy iteration:

\[
\begin{align*}
\text{improve policy} & : \pi_k \approx G\hat{V}_k \\
\text{evaluate policy} & : \hat{V}_{k+1} \approx V^{\pi_k}
\end{align*}
\]
FROM DYNAMIC PROGRAMMING TO VALUE-BASED REINFORCEMENT LEARNING

Fundamental RL tasks:
- Policy evaluation
- Policy improvement

\[ \pi_k = GV_k \]
\[ V_{k+1} = V^{\pi_k} \]

Approximate policy iteration:
\[ \hat{\pi}_k \approx G\hat{V}_k \]
\[ \hat{V}_{k+1} \approx V^{\pi_k} \]
FROM DYNAMIC PROGRAMMING TO VALUE-BASED REINFORCEMENT LEARNING

**Fundamental RL tasks:**
- Policy evaluation
- Policy improvement

**Challenges in RL:**
- Unknown transition and reward functions ⇒ have to learn from sample access only
- State/action space can be large ⇒ $V^*$ and $\pi^*$ cannot be stored in memory

Approximate policy iteration:

\[
\hat{V}_k \approx G\hat{V}_k
\]

improve policy

\[
\pi_k \approx \hat{V}_k
\]

evaluate policy

\[
\hat{V}_{k+1} \approx V^{\pi_k}
\]
FROM DYNAMIC PROGRAMMING TO VALUE-BASED REINFORCEMENT LEARNING

Fundamental RL tasks:
• Policy evaluation
• Policy improvement

Challenges in RL:
• Unknown transition and reward functions ⇒ have to learn from sample access only
• State/action space can be large ⇒ $V^*$ and $\pi^*$ cannot be stored in memory

Approximate policy iteration:

$\pi_k \approx G \hat{V}_k$

$\hat{V}_{k+1} \approx V^{\pi_k}$
Unknown transition and reward functions ⇒ have to learn from sample access only
Levels of Sample Access

- Full knowledge of $P$ ⇒ Planning (not RL)
- Unknown transition and reward functions ⇒ have to learn from sample access only
Unknown transition and reward functions $\Rightarrow$ have to learn from **sample access only**

LEVELS OF SAMPLE ACCESS

```
Generative model:
Full sample access to $P(\cdot | x, a)$ for any $(x, a)$
```

```
Full knowledge of $P$
$\Rightarrow$ Planning (not RL)
```
Levels of Sample Access

Unknown transition and reward functions ⇒ have to learn from *sample access only*

Full knowledge of $P$ ⇒ Planning (not RL)

Generative model:
Full sample access to $P(\cdot | x, a)$ for any $(x, a)$

Samples from full trajectories + reset action or save states
Unknown transition and reward functions ⇒ have to learn from sample access only

LEVELS OF SAMPLE ACCESS

Samples from a single trajectory ⇒ online RL

Samples from full trajectories + reset action or save states

Generative model: Full sample access to $P(\cdot | x, a)$ for any $(x, a)$

Full knowledge of $P$ ⇒ Planning (not RL)
Unknown transition and reward functions ⇒ have to learn from sample access only

LEVELS OF SAMPLE ACCESS

Samples from a single trajectory ⇒ online RL

Samples from full trajectories + reset action or save states

Generative model:
Full sample access to $P(\cdot |x, a)$ for any $(x, a)$

Full knowledge of $P$ ⇒ Planning (not RL)
State/action space can be large
⇒ $V^*$ and $\pi^*$ cannot be stored in memory

DEALING WITH LARGE STATE SPACES

Idea: approximate $V^*$ and/or $\pi^*$ in a computationally tractable way!
State/action space can be large
⇒ $V^*$ and $\pi^*$ cannot be stored in memory

DEALING WITH LARGE STATE SPACES

Idea: approximate $V^*$ and/or $\pi^*$ in a computationally tractable way!

Approximating $V^*$:
linear function approximation

- Define a set of $d$ features:
  $\phi_i: X \rightarrow \mathbb{R}$
- Parametrize value functions as
  $V_{\theta}(x) = \theta^T \phi(x)$
- Learning $V^* \Leftrightarrow$ Learning a good $\theta^*$
  $V_{\theta^*} \approx V^*$
State/action space can be large  
⇒ \( V^* \) and \( \pi^* \) cannot be stored in memory

DEALING WITH LARGE STATE SPACES

Idea: approximate \( V^* \) and/or \( \pi^* \) in a computationally tractable way!

Approximating \( V^* \):  
linear function approximation

- Define a set of \( d \) features:  
  \( \phi_i: X \rightarrow \mathbb{R} \)
- Parametrize value functions as  
  \( V_\theta(x) = \theta^T \phi(x) \)
- Learning \( V^* \) ⇔ Learning a good \( \theta^* \)  
  \( V_{\theta^*} \approx V^* \)

Approximating \( \pi^* \):  
parametrized policies

- Define a set of \( d \) features:  
  \( \phi_i: X \times A \rightarrow \mathbb{R} \)
- Parametrize (stochastic) policies as  
  \( \pi_\theta(a|x) \propto \exp(\theta^T \phi(x)) \)
- Learning \( \pi^* \) ⇔ Learning a good \( \theta^* \)  
  \( \pi_{\theta^*} \approx \pi^* \)
State/action space can be large
⇒ \( V^* \) and \( \pi^* \) cannot be stored in memory

DEALING WITH LARGE STATE SPACES

Idea: approximate \( V^* \) and/or \( \pi^* \) in a computationally tractable way!

Approximating \( V^* \): linear function approximation
- Define a set of \( d \) features:
  \[
  \phi_i: X \rightarrow \mathbb{R}
  \]
- Parametrize value functions as
  \[
  V_\theta(x) = \theta^T \phi(x)
  \]
- Learning \( V^* \) ⇔ Learning a good \( \theta^* \)
  \[
  V_{\theta^*} \approx V^*
  \]

Approximating \( \pi^* \): parametrized policies
- Define a set of \( d \) features:
  \[
  \phi_i: X \times A \rightarrow \mathbb{R}
  \]
- Parametrize (stochastic) policies as
  \[
  \pi_\theta(a|x) \propto \exp(\theta^T \phi(x))
  \]
- Learning \( \pi^* \) ⇔ Learning a good \( \theta^* \)
  \[
  \pi_{\theta^*} \approx \pi^*
  \]
FEATURE MAP EXAMPLE
FEATURE MAP EXAMPLE
FEATURE MAP EXAMPLE

“coarse coding”
≈
indicator features
\( \phi_i(x) = 1\{x \in X_i\} \)
“PROST” FEATURES FOR ATARI GAMES

High-dimensional observations: 192×160 pixels
“PROST” FEATURES FOR ATARI GAMES

High-dimensional observations: 192×160 pixels
“PROST” FEATURES FOR ATARI GAMES

High-dimensional observations: 192×160 pixels

Low-dimensional observations: 14×16 patches
METHODS FOR POLICY EVALUATION

\[ \pi_k \approx G \hat{V}_k \]

improve policy

\[ \hat{V}_{k+1} \approx V^{\pi_k} \]

evaluate policy

policy evaluation
A GENTLE START: MONTE CARLO

Observe:

Policy evaluation = estimating $V^\pi$:

$V^\pi(x) = E_\pi[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t) \mid x_0 = x]$
A GENTLE START: MONTE CARLO

Observe:
Policy evaluation = estimating $V^\pi$:
$V^\pi(x) = E_\pi[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t) | x_0 = x]$

Idea:
approximate $E_\pi[\cdot]$ by sample averages!

- Simulate $N$ trajectories using policy $\pi$
- For every state $x$ that appears in the trajectories, let
  $\hat{V}_N(x) = \text{avg}(R_{1:N}(x))$
A GENTLE START: MONTE CARLO

Idea:
approximate $E_\pi[\cdot]$ by sample averages!

- Simulate $N$ trajectories using policy $\pi$
- For every state $x$ that appears in the trajectories, let
  \[ \hat{V}_N(x) = \text{avg}(R_{1:N}(x)) \]
A GENTLE START: MONTE CARLO

Idea:
approximate $E_{\pi}[\cdot]$ by sample averages!

- Simulate $N$ trajectories using policy $\pi$
- For every state $x$ that appears in the trajectories, let
  $$\hat{V}_N(x) = \text{avg}(R_{1:N}(x))$$

Collection of discounted returns $\sum_{t=0}^{T'} \gamma^t r_t$ after first visit to $x$
A GENTLE START: MONTE CARLO

Idea:
approximate $E_\pi[\cdot]$ by sample averages!

- Simulate $N$ trajectories using policy $\pi$
- For every state $x$ that appears in the trajectories, let
  \[ \hat{V}_N(x) = \text{avg}(R_{1:N}(x)) \]

Average of i.i.d. random variables:
\[ \lim_{N \to \infty} \hat{V}_N = V_\pi \]

Collection of discounted returns $\sum_{t=0}^{T'} \gamma^t r_t$ after first visit to $x$
Monte Carlo policy evaluation

Input:

\( N \) trajectories \( \sim \pi \), feature map \( \phi: X \to \mathbb{R}^d \)

Output:

\[
\hat{V}_N = \arg\min_{\theta \in \mathbb{R}^d} \mathbb{E}_x \left[ (\theta^T \phi(x) - R_{1:N}(x))^2 \right]
\]
Monte Carlo policy evaluation

Input:
\( N \) trajectories \( \sim \pi \), feature map \( \phi: X \rightarrow \mathbb{R}^d \)

Output:
\[
\hat{V}_N = \arg \min_{\theta \in \mathbb{R}^d} \mathbb{E}_x \left[ (\theta^T \phi(x) - R_{1:N}(x))^2 \right]
\]

Least-squares fit of discounted returns
PROPERTIES OF MONTE CARLO

😊 Value estimates converge to true values 😊

😊 Doesn’t need prior knowledge of $P$ or $r$ 😊
PROPERTIES OF MONTE CARLO

😊 Value estimates converge to true values 😊

😊 Doesn’t need prior knowledge of $P$ or $r$ 😊

😖 Doesn’t make use of the Bellman equations 😖
A BETTER OBJECTIVE?

Idea: construct an objective that uses the Bellman equations

\[ V^{\pi} \approx T^{\pi} V^{\pi} \]
A BETTER OBJECTIVE?

**Idea:** construct an objective that uses the Bellman equations

\[ V^\pi \approx T^\pi V^\pi \]

**The Bellman error**

\[ L(V) = \mathbb{E}_{x \sim \mu} \left[ (T^\pi V(x) - V(x))^2 \right] \]
Idea: use stochastic approximation to reduce instantaneous Bellman error

$$\Delta_t = \left( T^\pi \hat{V}_t(x_t) - \hat{V}_t(x_t) \right)^2$$
**TEMPORAL DIFFERENCE LEARNING**

**Idea:** use stochastic approximation to reduce instantaneous Bellman error

$$\Delta_t = \left(T^\pi \hat{V}_t(x_t) - \hat{V}_t(x_t)\right)^2$$

**TD(0)**

**Input:** arbitrary function $\hat{V}_0: X \rightarrow \mathbb{R}$

For $t = 0, 1, \ldots$

$$\delta_t = r_t + \gamma \hat{V}_t(x_{t+1}) - \hat{V}_t(x_t)$$

$$\hat{V}_{t+1}(x_t) = \hat{V}_t(x_t) + \alpha_t \delta_t$$
TEMPORAL DIFFERENCE LEARNING

**TD(0)**

**Input:** arbitrary function $\hat{V}_0: \mathcal{X} \rightarrow \mathbb{R}$

For $t = 0,1,\ldots,$

$$\delta_t = r_t + \gamma \hat{V}_t(x_{t+1}) - \hat{V}_t(x_t)$$

$$\hat{V}_{t+1}(x_t) = \hat{V}_t(x_t) + \alpha_t \delta_t$$

Converges if **step-sizes** satisfy

$$\sum_{t=0}^{\infty} \alpha_t = \infty \quad \text{and} \quad \sum_{t=0}^{\infty} \alpha_t^2 < \infty$$

(e.g., $\alpha_t = c/t$ does the job)
**TEMPORAL DIFFERENCE LEARNING**

**TD(0)**

**Input:** arbitrary function $\hat{V}_0: X \rightarrow \mathbb{R}$

For $t = 0, 1, \ldots$,

\[
\delta_t = r_t + \gamma \hat{V}_t(x_{t+1}) - \hat{V}_t(x_t)
\]

\[
\hat{V}_{t+1}(x_t) = \hat{V}_t(x_t) + \alpha_t \delta_t
\]

Converges if step-sizes satisfy

\[
\sum_{t=0}^{\infty} \alpha_t = \infty \quad \text{and} \quad \sum_{t=0}^{\infty} \alpha_t^2 < \infty
\]

(e.g., $\alpha_t = c/t$ does the job)

**In equilibrium,**

\[
E[r_t + \gamma \hat{V}_t(x_{t+1}) - \hat{V}_t(x_t)] = 0
\]
TD(0) WITH
LINEAR FUNCTION APPROXIMATION

Let $\phi: X \to \mathbb{R}^d$ be a feature vector
TD(0) WITH LINEAR FUNCTION APPROXIMATION

Let $\phi: X \to \mathbb{R}^d$ be a feature vector

Approximating $V^\pi(x) \approx \theta^\top \phi(x)$ by TD(0):

**TD(0) with LFA**

**Input:** arbitrary param. vector $\theta_0 \in \mathbb{R}^d$

For $t = 0,1, \ldots$,

\[
\delta_t = r_t + \gamma \theta_t^\top \phi(x_{t+1}) - \theta_t^\top \phi(x_t)
\]

\[
\theta_{t+1} = \theta_t + \alpha_t \delta_t \phi(x_t)
\]
Let $\phi: X \to \mathbb{R}^d$ be a feature vector

Approximating $V^\pi(x) \approx \theta^\top \phi(x)$ by TD(0):

**TD(0) with LFA**

**Input:** arbitrary param. vector $\theta_0 \in \mathbb{R}^d$

For $t = 0, 1, \ldots$,

\[\delta_t = r_t + \gamma \theta_t^\top \phi(x_{t+1}) - \theta_t^\top \phi(x_t)\]

\[\theta_{t+1} = \theta_t + \alpha_t \delta_t \phi(x_t)\]

This still converges to $V^\pi$!!!
Let $V_\theta : X \to R$ be a parametric class of functions (e.g., deep neural network)

Approximating $V^\pi(x) \approx V_\theta(x)$ by TD(0):

**TD(0) with general FA**

**Input:** arbitrary param. vector $\theta_0 \in \mathbb{R}^d$

For $t = 0, 1, \ldots$,

\[
\delta_t = r_t + \gamma V_{\theta_t}(x_{t+1}) - V_{\theta_t}(x_t)
\]

\[
\theta_{t+1} = \theta_t + \alpha_t \delta_t \nabla_{\theta} V_{\theta_t}(x_t)
\]
Let $V_\theta : X \to \mathbb{R}$ be a parametric class of functions (e.g., deep neural network).

Approximating $V^\pi(x) \approx V_\theta(x)$ by TD(0):

**TD(0) with general FA**

**Input:** arbitrary param. vector $\theta_0 \in \mathbb{R}^d$

For $t = 0, 1, \ldots$,

$$
\delta_t = r_t + \gamma V_{\theta_t}(x_{t+1}) - V_{\theta_t}(x_t)
$$

$$
\theta_{t+1} = \theta_t + \alpha_t \delta_t \nabla_{\theta} V_{\theta_t}(x_t)
$$

Not much is known about convergence 😞
PROPERTIES OF TD(0)

😊 Value estimates converge to true values 😊

😊 Doesn’t need prior knowledge of $P$ or $r$ 😊

😊 Based on the concept of Bellman error 😊
PROPERTIES OF TD(0)

😊 Value estimates converge to true values 😊

😊 Doesn’t need prior knowledge of $P$ or $r$ 😊

😊 Based on the concept of Bellman error 😊

= “bootstrapping”
WHERE DOES TD(0) CONVERGE TO?

**TD(0) with LFA**

**Input:** arbitrary param. vector $\theta_0 \in \mathbb{R}^d$

For $t = 0, 1, ...$,

$$\delta_t(\theta) = r_t + \gamma \theta^\top \phi(x_{t+1}) - \theta^\top \phi(x_t)$$

$$\theta_{t+1} = \theta_t + \alpha_t \delta_t(\theta_t) \phi(x_t)$$
WHERE DOES TD(0) CONVERGE TO?

**TD(0) with LFA**

**Input:** arbitrary param. vector $\theta_0 \in \mathbb{R}^d$

For $t = 0,1, \ldots$, 

$$
\delta_t(\theta) = r_t + \gamma \theta^T \phi(x_{t+1}) - \theta^T \phi(x_t)
$$

$$
\theta_{t+1} = \theta_t + \alpha_t \delta_t(\theta_t) \phi(x_t)
$$

In the limit, TD(0) finds a $\theta^*$ such that 

$$
\mathbb{E}[\delta_t(\theta^*) \phi(x_t)] = 0
$$
WHERE DOES TD(0) CONVERGE TO?

Idea: given a finite trajectory, approximate the TD fixed point by solving

\[ E[\delta_t(\theta)\phi(x_t)] \approx \frac{1}{T} \sum_{t=1}^{T} \delta_t(\theta)\phi(x_t) = 0 \]
WHERE DOES TD(0) CONVERGE TO?

**Idea:** given a finite trajectory, approximate the TD fixed point by solving

\[
\mathbb{E}[\delta_t(\theta)\phi(x_t)] \approx \frac{1}{T} \sum_{t=1}^{T} \delta_t(\theta)\phi(x_t) = 0
\]

Equivalently:

\[
\frac{1}{T} \sum_{t=1}^{T} \phi(x_t)(\phi(x_t) - \gamma \phi(x_{t+1}))^\top \theta \overset{=} \frac{1}{T} \sum_{t=1}^{T} r_t \phi(x_t)
\]
WHERE DOES TD(0) CONVERGE TO?

Idea:
given a finite trajectory, approximate the TD fixed point by solving

$$E_{\delta \theta \phi x t} \approx 1$$

$$T t = 1$$

Equivalently:

$$1 \frac{T}{T} \sum_{t=1}^{T} \phi(x_t)\left(\phi(x_t) - \gamma \phi(x_{t+1})\right)^{\top} \theta = 1 \frac{T}{T} \sum_{t=1}^{T} r_t \phi(x_t)$$

This is a linear system

$$A_T \theta = b_T$$

Solution:
WHERE DOES TD(0) CONVERGE TO?

Idea:
given a finite trajectory, approximate
the TD fixed point by solving

\[ E \delta_t \theta = \frac{1}{T} \]

\[ A_T \theta = b_T \]

Solution: \( \theta_T = A_T^{-1} b_T \)

This is a linear system

Equivalently:

\[
\frac{1}{T} \sum_{t=1}^{T} \phi(x_t)(\phi(x_t) - \gamma \phi(x_{t+1}))^T \theta = \frac{1}{T} \sum_{t=1}^{T} r_t \phi(x_t)
\]
LEAST-SQUARES TEMPORAL DIFFERENCE LEARNING (LSTD)

**LSTD(0)**

Input: trajectory \((x_t, a_t, r_t)^T_{t=1}\)

\[
\begin{align*}
\theta_T &= A_T^{-1} b_T \\
\hat{V}_T &= \theta_T^T \phi
\end{align*}
\]
LEAST-SQUARES TEMPORAL DIFFERENCE LEARNING (LSTD)

LSTD(0)

Input: trajectory \((x_t, a_t, r_t)^T_{t=1}\)

\[ \theta_T = A_T^{-1} b_T \]
\[ \hat{V}_T = \theta_T^\top \phi \]

😊 converges to same \(\theta^*\) as TD(0) 😊

😊 no need to set step sizes \(\alpha_t\) 😊
LEAST-SQUARES TEMPORAL DIFFERENCE LEARNING (LSTD)

**LSTD(0)**

**Input:** trajectory $(x_t, a_t, r_t)^T_{t=1}$

\[
\theta_T = A_T^{-1} b_T \\
\hat{V}_T = \theta_T^T \phi
\]

- 🎉 converges to same $\theta^*$ as TD(0) 🎉
- 🎉 no need to set step sizes $\alpha_t$ 🎉
- 😞 computational complexity: $O(Td^2 + d^3)$ ☹️
- 😞 $A_T^{-1}$ may not exist for small $T$ ☹️

TD(0): $O(Td)$
In the limit $T \to \infty$, LSTD(0) and TD(0) both minimize the projected Bellman error

$$L(V) = \mathbb{E}_{x \sim \mu} \left[ \left( \Pi_\phi [T^\pi V(x)] - V(x) \right)^2 \right]$$
THE CONVERGENCE OF TD(0) AND LSTD(0)

**Theorem**

In the limit $T \to \infty$, LSTD(0) and TD(0) both minimize the projected Bellman error

$$L(V) = \mathbb{E}_{x \sim \mu} \left[ \left( \Pi \phi [T^\pi V(x)] - V(x) \right)^2 \right]$$

Projection onto span of features
From Policy Evaluation
Policy Improvement

\[ \pi_k \approx G \hat{V}_k \]

\[ \hat{V}_{k+1} \approx V^{\pi_k} \]

So far: policy evaluation
FROM POLICY EVALUATION

POLICY IMPROVEMENT

\[ \hat{V}_k \approx G \hat{V}_k \]

\[ \hat{V}_{k+1} \approx V^{\pi_k} \]

evaluate policy

improve policy

now for the real deal:

policy eval + improvement
OFF-POLICY CONTROL: Q-LEARNING

Idea: Let’s try to
• directly learn about $Q^*$, and
• improve the policy on the fly!
OFF-POLICY CONTROL: Q-LEARNING

Idea: Let’s try to

- directly learn about $Q^*$, and
- improve the policy on the fly!

• Compute $\varepsilon$-greedy policy w.r.t. $\hat{Q}_t$:
  
  $$\pi_t(x) = \begin{cases} 
  \arg \max \hat{Q}_t(x, a), & \text{w. p. } 1 - \varepsilon \\
  \text{uniform random action}, & \text{w. p. } \varepsilon 
  \end{cases}$$

• Improve estimated $\hat{Q}_{t+1}$ by reducing Bellman error
  
  $$\Delta_t = \left( \mathbb{E} \left[ r_t + \gamma \max_a \hat{Q}_t(x_{t+1}, a) \right] - \hat{Q}_t(x_t, a_t) \right)$$
**OFF-POLICY CONTROL: Q-LEARNING**

**Idea:** Let’s try to

- directly learn about $Q^*$, and
- improve the policy on the fly!

- Compute $\varepsilon$-greedy policy w.r.t. $\hat{Q}_t$:
  $$\pi_t(x) = \begin{cases} 
  \text{arg max } \hat{Q}_t(x, a), & \text{w. p. } 1 - \varepsilon \\
  \text{uniform random action}, & \text{w. p. } \varepsilon
  \end{cases}$$

- Improve estimated $\hat{Q}_{t+1}$ by reducing Bellman error
  $$\Delta_t = \left( \mathbb{E} [r_t + \gamma \max_a \hat{Q}_t(x_{t+1}, a)] - \hat{Q}_t(x_t, a_t) \right)^2$$
**OFF-POLICY CONTROL: Q-LEARNING**

**Q-learning**

**Input:** arbitrary \( \hat{Q}_0 : X \times A \to \mathbb{R} \)

For \( t = 0, 1, \ldots \),

- Choose action \( a_t \sim \varepsilon\text{-greedy w.r.t. } \hat{Q}_t \)
- Observe \( r_t, x_{t+1} \)
- Compute

\[
\delta_t = r_t + \gamma \max_a \hat{Q}_t(x_{t+1}, a) - \hat{Q}_t(x_t, a_t)
\]

\[
\hat{Q}_{t+1}(x_t, a_t) = \hat{Q}_t(x_t, a_t) + \alpha_t \delta_t
\]
**ON-POLICY CONTROL: ** **SARSA**

**SARSA**

**Input:** arbitrary $\hat{Q}_0 : X \times A \rightarrow \mathbb{R}$

For $t = 0,1,\ldots$, 

- Choose action $a_t \sim \varepsilon$-greedy w.r.t. $Q_t$
- Observe $r_t, x_{t+1}, a'_{t+1}$
- Compute
  
  $\delta_t = r_t + \gamma \hat{Q}_t(x_{t+1}, a'_{t+1}) - \hat{Q}_t(x_t, a_t)$
  
  $\hat{Q}_{t+1}(x_t, a_t) = \hat{Q}_t(x_t, a_t) + \alpha_t \delta_t$
ON-POLICY CONTROL: SARSA

**SARSA**

**Input:** arbitrary $\hat{Q}_0 : X \times A \to R$

For $t = 0, 1, \ldots$,

- Choose action $a_t \sim \epsilon$-greedy w.r.t. $\hat{Q}_t$
- Observe $r_t, x_{t+1}, a'_{t+1}$
- Compute

$$\delta_t = r_t + \gamma \hat{Q}_t(x_{t+1}, a'_{t+1}) - \hat{Q}_t(x_t, a_t)$$

$$\hat{Q}_{t+1}(x_t, a_t) = \hat{Q}_t(x_t, a_t) + \alpha_t \delta_t$$

$a'_{t+1} \sim \epsilon$-greedy: on-policy
ON-POLICY CONTROL: **SARSA**

**SARSA**

**Input:** arbitrary $\hat{Q}_0: X \times A \rightarrow R$

For $t = 0,1,\ldots$,

- Choose action $a_t \sim \varepsilon$-greedy w.r.t. $\hat{Q}_t$
- Observe $r_t, x_{t+1}, a'_{t+1}$
- Compute

  \[
  \delta_t = r_t + \gamma \hat{Q}_t(x_{t+1}, a'_{t+1}) - \hat{Q}_t(x_t, a_t)
  \]
  \[
  \hat{Q}_{t+1}(x_t, a_t) = \hat{Q}_t(x_t, a_t) + \alpha_t \delta_t
  \]

**SARSA** = $(s_t, a_t, r_t, s_{t+1}, a'_{t+1})$
ON-POLICY CONTROL: SARSA

SARSA ~ XARXA

Input: arbitrary $\hat{Q}_0: X \times A \to R$

For $t = 0, 1, \ldots$,

- Choose action $a_t \sim \epsilon$-greedy w.r.t. $\hat{Q}_t$
- Observe $r_t, x_{t+1}, a'_{t+1}$
- Compute
  \[ \delta_t = r_t + \gamma \hat{Q}_t(x_{t+1}, a'_{t+1}) - \hat{Q}_t(x_t, a_t) \]
  \[ \hat{Q}_{t+1}(x_t, a_t) = \hat{Q}_t(x_t, a_t) + \alpha_t \delta_t \]

SARSA = $(s_t, a_t, r_t, s_{t+1}, a'_{t+1})$
Q-LEARNING VS. SARSA WITH FUNCTION APPROXIMATION

Both algorithms can be adapted to linear and non-linear FA by using the update rule
\[ \theta_{t+1} = \theta_t + \alpha_t \delta_t \nabla_{\theta} Q_\theta(x_t, a_t) \]
Q-LEARNING VS. SARSA WITH FUNCTION APPROXIMATION

Both algorithms can be adapted to linear and non-linear FA by using the update rule

$$\theta_{t+1} = \theta_t + \alpha_t \delta_t \nabla_\theta Q_\theta(x_t, a_t)$$

- SARSA guarantees bounded error and tends to behave well in practice (may not find optimal policy though)
Q-LEARNING VS. SARSA
WITH FUNCTION APPROXIMATION

Both algorithms can be adapted to linear and non-linear FA by using the update rule

$$\theta_{t+1} = \theta_t + \alpha_t \delta_t \nabla_\theta Q_\theta(x_t, a_t)$$

- SARSA guarantees bounded error and tends to behave well in practice (may not find optimal policy though)
- Q-learning may diverge catastrophically
Q-LEARNING VS. SARSA
WITH FUNCTION APPROXIMATION

Both algorithms can be adapted to linear and non-linear FA by using the update rule

\[ \theta_{t+1} = \theta_t + \alpha_t \delta_t \nabla_{\theta} Q_{\theta}(x_t, a_t) \]

- SARSA guarantees bounded error and tends to behave well in practice (may not find optimal policy though)
- Q-learning may diverge catastrophically
  - Proposed fixes: gradient TD algorithms, emphatic TD algorithms, double Q-learning, soft Q-learning, G-learning,…
Q-LEARNING VS. SARSA WITH FUNCTION APPROXIMATION

Both algorithms can be adapted to linear and non-linear FA by using the update rule

\[ \theta_{t+1} = \theta_t + \alpha_t \delta_t \nabla_{\theta} Q_\theta(x_t, a_t) \]

- SARSA guarantees bounded error and tends to behave well in practice (may not find optimal policy though)
- **Q-learning may diverge catastrophically**
  - Proposed fixes: gradient TD algorithms, emphatic TD algorithms, double Q-learning, soft Q-learning, G-learning,…
  - Practical solution: tune it until it works
DIVERGENCE OF OFF-POLICY TD LEARNING

The “deadly triad”:
- Function approximation
- Bootstrapping
- Off-policy learning
DIVERGENCE OF OFF-POLICY TD LEARNING

The “deadly triad”:
- Function approximation
- Bootstrapping
- Off-policy learning

BUT

Divergence is typically not too extreme when behavior policy is close to evaluation policy and FA is linear
and now

the moment you all have been waiting for
DEEP
REINFORCEMENT
LEARNING
THE PROMISE OF DEEP REINFORCEMENT LEARNING

Parametrize $Q$-function/policy by a deep net

$(x, a) \rightarrow Q_{\theta}(x, a) \rightarrow \pi_{\theta}(a|x)$
Parametrize $Q$-function/policy by a deep net

INPUT: $(x, a)$

$Q_\theta(x, a)$

$\pi_\theta(a|x)$

Hope: Take advantage of representation power!
THE PROMISE OF DEEP REINFORCEMENT LEARNING

Parametrize \( Q \)-function/policy by a deep net

Hope: Take advantage of representation power!

Challenge: Existing RL methods difficult to generalize
LEAST-SQUARES TEMPORAL DIFFERENCE LEARNING (LSTD)

**LSTD(0)**

**Input:** trajectory \((x_t, a_t, r_t)^T_{t=1}\)

\[\theta_T = A_T^{-1} b_T\]

\[\hat{V}_T = \theta_T^T \phi\]

Idea not directly applicable to non-linear function approximation!
Can we optimize Bellman error

$$L(\theta) = \mathbb{E}_{x \sim \mu} \left[ (T^\pi V_\theta (x) - V_\theta (x))^2 \right]$$

by stochastic gradient descent????
Can we optimize Bellman error

\[ L(\theta) = \mathbb{E}_{x \sim \mu} \left[ (T^\pi V_\theta(x) - V_\theta(x))^2 \right] \]

by stochastic gradient descent? NO!!

Bellman error involves a double expectation:

\[ L(\theta) = \mathbb{E}_X [\ell(\theta; X, \mathbb{E}_Y[Y|X])] \]

can’t get unbiased gradients!
Can we optimize Bellman error $L(\theta) = \mathbb{E}_{x \sim \mu} \left[ \ell(\theta; x, \mathbb{E}_y[y|x]) \right]$ by stochastic gradient descent???

The infamous “double sampling” issue of RL can’t get unbiased gradients!
TACKLING DOUBLE SAMPLING

• Saddle-point optimization:
  \[ \min_{\theta} \mathbb{E}[f(\theta; X, \mathbb{E}[Y|X])^2] \]
TACKLING DOUBLE SAMPLING

• Saddle-point optimization:

\[
\min_{\theta} \mathbb{E}[f(\theta; X, \mathbb{E}[Y|X])^2] = \\
\min_{\theta} \max_{z} \mathbb{E}[z(X, Y) \cdot f(\theta; X, \mathbb{E}[Y|X])] - \mathbb{E}[z^2(X, Y)]
\]
**TACKLING DOUBLE SAMPLING**

• Saddle-point optimization:

\[
\min_{\theta} \mathbb{E}[f(\theta; X, \mathbb{E}[Y|X])^2] = \\
\min_{\theta} \max_{z} \mathbb{E}[z(X, Y) \cdot f(\theta; X, \mathbb{E}[Y|X])] - \mathbb{E}[z^2(X, Y)]
\]

⇒ “modified Bellman residual” (Antos et al. 2008), “Gradient TD” methods (Sutton et al. 2009), SBEED (Dai et al., 2018)
Saddle-point optimization:

\[
\min_{\theta} \mathbb{E}[f(\theta; X, \mathbb{E}[Y|X])^2] = \\
\min_{\theta} \max_z \mathbb{E}[z(X, Y) \cdot f(\theta; X, \mathbb{E}[Y|X])] - \mathbb{E}[z^2(X, Y)]
\]

⇒ “modified Bellman residual” (Antos et al. 2008), “Gradient TD” methods (Sutton et al. 2009), SBEED (Dai et al., 2018)

Iterative optimization schemes
TACKLING DOUBLE SAMPLING

• Saddle-point optimization:
  \[
  \min_{\theta} \mathbb{E}[f(\theta; X, \mathbb{E}[Y|X])^2] = \\
  \min_{\theta} \max_z \mathbb{E}[z(X, Y) \cdot f(\theta; X, \mathbb{E}[Y|X])] - \mathbb{E}[z^2(X, Y)]
  \]
  ⇒ “modified Bellman residual” (Antos et al. 2008), “Gradient TD” methods (Sutton et al. 2009), SBEED (Dai et al., 2018)

• Iterative optimization schemes
**Fitted Policy Evaluation**

**Idea:** compute sequence of value functions by minimizing

$$L_n(\hat{V}; \hat{V}_k) = \frac{1}{n} \sum_{t=1}^{n} \left( r_t + \hat{V}_k(x_{t+1}) - \hat{V}(x_t) \right)^2$$
Fitted Policy Evaluation

Idea: compute sequence of value functions by minimizing

$$L_n(\hat{V}; \hat{V}_k) = \frac{1}{n} \sum_{t=1}^{n} \left( r_t + \hat{V}_k(x_{t+1}) - \hat{V}(x_t) \right)^2$$

This can be finally treated as a regression problem & solved by SGD!
Fitted Policy Iteration

\[ \hat{V}_k \approx G \hat{V}_k \]

\[ \hat{V}_{k+1} \approx V^{\pi_k} \]

\[ \pi_k \approx G \hat{V}_k \]

\[ \epsilon\text{-Greedy policy update} \]

Fitted policy evaluation
Fitted Policy Iteration

\[ \pi_k \approx G \hat{V}_k \]

\[ \hat{V}_{k+1} \approx V^{\pi_k} \]

\( \varepsilon \)-Greedy policy update

Computing policy needs model of \( P \) … better use Q-functions!

Fitted policy evaluation
**Fitted Value Iteration**

Idea: compute sequence of $Q$-value functions by minimizing

$$L_n(\hat{Q}; \hat{Q}_k) = \frac{1}{n} \sum_{t=1}^{n} \left( r_t + \max_a \hat{Q}_k(x_{t+1}, a) - \hat{Q}(x_t, a_t) \right)^2$$

Target

Free variable
Fitted value iteration

Input: function space $F$, $\hat{Q}_0 \in F$

For $k = 0,1, \ldots$, 

• $\pi_k = G_{\varepsilon} \hat{Q}_k$
• generate trajectory $\left(x_t, a_t, r_t\right)_{t=1}^n \sim \pi_k$
• compute $\hat{Q}_{k+1} = \text{argmin}_{\hat{Q} \in F} L_n(\hat{Q}; \hat{Q}_k)$
Fitted value iteration

Input: function space $F$, $\hat{Q}_0 \in F$

For $k = 0, 1, \ldots$,

- $\pi_k = G_\varepsilon \hat{Q}_k$
- generate trajectory
  $$(x_t, a_t, r_t)_{t=1}^n \sim \pi_k$$
- compute
  $$\hat{Q}_{k+1} = \arg\min_{\hat{Q} \in F} L_n(\hat{Q}; \hat{Q}_k)$$

[Note: Computing policy is trivial!]

**Fitted Value Iteration**
**Fitted Value Iteration**

**Input:** function space $F$, $\hat{Q}_0 \in F$

For $k = 0, 1, \ldots$,
- $\pi_k = G_\varepsilon \hat{Q}_k$
- generate trajectory
  \[
  (x_t, a_t, r_t)_{t=1}^n \sim \pi_k
  \]
- compute
  \[
  \hat{Q}_{k+1} = \arg\min_{\hat{Q} \in F} L_n(\hat{Q}; \hat{Q}_k)
  \]

Convergence can be guaranteed! under very technical assumptions…

Computing policy is trivial!
DEEP Q NETWORKS

Parametrize \( Q \)-function by a deep neural net:

\[
Q_\theta(x, a) = \mathcal{N}(\theta(x, a))
\]
DEEP Q NETWORKS

Minimize the loss

\[
E_{(X,A,R,X') \sim D} \left[ \left( R + \gamma \max_b Q_{\theta_k}(X', b) - Q_{\theta}(X, A) \right)^2 \right]
\]
DEEP Q NETWORKS

Minimize the loss

\[
E_{(X,A,R,X')\sim D} \left[ \left( R + \gamma \max_b Q_{\theta_k}(X', b) - Q_\theta(X, A) \right)^2 \right]
\]

+ training tricks:
- Store transitions \((x, a, r, x')\) in replay buffer \(D\) to break dependence on recent samples
- Compute small updates by mini-batch stochastic gradient descent
- Use an older parameter vector \(\theta_{k-m}\) in target to avoid oscillations
- …
DEEP Q NETWORKS FOR PLAYING ATARI

Superhuman performance!!
BUT results very difficult to reproduce as the system is very unstable…
THIS SHORT COURSE: A PRIMAL-DUAL VIEW

- Markov decision processes
  - Value functions and optimal policies

- Primal view: Dynamic programming
  - Policy evaluation, value and policy iteration
  - Value-function-based methods
    - Temporal differences, Q-learning, LSTD, deep Q networks,…

- Dual view: Linear programming
  - LP duality in MDPs
  - Direct policy optimization methods
    - Policy gradients, REPS, TRPO,…
THIS SHORT COURSE: 
A PRIMAL-DUAL VIEW

- Markov decision processes
  - Value functions and optimal policies

- Primal view: Dynamic programming
  - Policy evaluation
  - Value- and policy iteration
  - Value-function-based methods
    - Temporal differences, Q-learning, LSTD, deep Q networks,…

- Dual view: Linear programming
  - LP duality in MDPs
  - Direct policy optimization methods
    - Policy gradients, REPS, TRPO,…

But first: some more notation 😊
**Policies and Trajectory Distributions**

**Policy:** mapping from histories to actions

\[ \pi : x_1, a_1, x_2, a_2, \ldots, x_t \mapsto a_t \]

**Stationary policy:** mapping from states to actions

(no dependence on history or \( t \))

\[ \pi : x \mapsto a \]

Let \( \tau = (x_1, a_1, x_2, a_2, \ldots) \) be a **trajectory** generated by running \( \pi \) in the MDP \( \tau \sim (\pi, P) \):

- \( a_t = \pi(x_t, a_{t-1}, x_{t-1}, \ldots, x_1) \)
- \( x_{t+1} \sim P(\cdot \mid x_t, a_t) \)

Expectation under this distribution: \( E_\pi[\cdot] \)
**Policies and Trajectory Distributions**

**Stationary stochastic policy**: mapping from states to action distributions

\( \pi: A \times X \rightarrow [0,1] \)

where

\[ \pi(a|x) = P[a_t = a|x_t = x] \]

Let \( \tau = (x_1, a_1, x_2, a_2, \ldots) \) be a trajectory generated by running \( \pi \) in the MDP \( \tau \sim (\pi, P) \):

- \( a_t = \pi(x_t, a_{t-1}, x_{t-1}, \ldots, x_1) \)
- \( x_{t+1} \sim P(\cdot | x_t, a_t) \)

Expectation under this distribution: \( E_{\pi}[\cdot] \)
Policies and Trajectory Distributions

**Stationary stochastic policy:** mapping from states to action distributions

\[ \pi : A \times X \to [0,1] \]

where

\[ \pi(a|x) = P[a_t = a|x_t = x] \]

Let \( \tau = (x_1, a_1, x_2, a_2, \ldots) \) be a trajectory generated by running \( \pi \) in the MDP \( \tau \sim (\pi, P) \):

- \( a_t \sim \pi(\cdot | x_t) \)
- \( x_{t+1} \sim P(\cdot | x_t, a_t) \)

Expectation under this distribution: \( E_\pi[\cdot] \)
Discounted MDPs:
- No terminal state
- Discount factor $\gamma \in (0,1)$
- **GOAL:** maximize total discounted reward

$$R_\gamma = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r_t \right]$$
ANOTHER PERSPECTIVE ON DISCOUNTED REWARDS

Discounted MDPs:
- No terminal state, initial state $x_0 \sim \mu_0$
- Discount factor $\gamma \in (0,1)$
- GOAL: maximize total discounted reward

$$R_\gamma = \mathbb{E}[^{\infty}_{t=0} \gamma^t r_t]$$
ANOTHER PERSPECTIVE ON DISCOUNTED REWARDS

Discounted MDPs:
- No terminal state, initial state $x_0 \sim \mu_0$
- Discount factor $\gamma \in (0,1)$
- **GOAL:** maximize total discounted reward

$$R_\gamma = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r_t \right]$$

Observe: the discounted reward of a policy is

$$R_\gamma^\pi = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t r(x_t, a_t) \right]$$
ANOTHER PERSPECTIVE ON DISCOUNTED REWARDS

Discounted MDPs:

- No terminal state, initial state $x_0 \sim \mu_0$
- Discount factor $\gamma \in (0,1)$
- GOAL: maximize total discounted reward

$$R_\gamma = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$$

Observe: the discounted reward of a policy is

$$R_\gamma^{\pi} = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t)]$$
$$= \mathbb{E}_{\pi}\left[\sum_{x,a} \sum_{t=0}^{\infty} \gamma^t 1_{\{x_t=x, a_t=a\}} r(x, a)\right]$$
ANOTHER PERSPECTIVE ON DISCOUNTED REWARDS

Discounted MDPs:
- No terminal state, initial state $x_0 \sim \mu_0$
- Discount factor $\gamma \in (0,1)$
- GOAL: maximize total discounted reward

$$R_\gamma = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$$

Observe: the discounted reward of a policy is

$$R_\gamma^\pi = \mathbb{E}_\pi [\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t)]$$
$$= \mathbb{E}_\pi [\sum_{x,a} \sum_{t=0}^{\infty} \gamma^t 1_{\{x_t=x, a_t=a\}} r(x, a)]$$
$$= \sum_{x,a} \sum_{t=0}^{\infty} \gamma^t P_\pi [x_t = x, a_t = a] r(x, a)$$
ANOTHER PERSPECTIVE ON DISCOUNTED REWARDS

Discounted MDPs:
- No terminal state, initial state $x_0 \sim \mu_0$
- Discount factor $\gamma \in (0,1)$
- **GOAL:** maximize total discounted reward
  \[ R_\gamma = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t] \]

**Observe:** the discounted reward of a policy is
\[
R^\pi_\gamma = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t)]
= \mathbb{E}_\pi[\sum_{x,a} \sum_{t=0}^{\infty} \gamma^t 1_{\{x_t=x, a_t=a\}} r(x, a)]
= \sum_{x,a} \sum_{t=0}^{\infty} \gamma^t \mathbb{P}_\pi[x_t = x, a_t = a] r(x, a)
= \sum_{x,a} \mu_\pi(x, a)r(x, a)\]
ANOTHER PERSPECTIVE ON DISCOUNTED REWARDS

Discounted MDPs:
- No terminal state, initial state $x_0 \sim \mu_0$
- Discount factor $\gamma \in (0,1)$
- GOAL: maximize total discounted reward

$$R_\gamma = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$$

Observe: the discounted reward of a policy is

$$R_\gamma^\pi = \mathbb{E}_\pi [\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t)]$$
$$= \mathbb{E}_\pi [\sum_{x,a} \sum_{t=0}^{\infty} \gamma^t 1_{\{x_t=x, a_t=a\}} r(x, a)]$$
$$= \sum_{x,a} \sum_{t=0}^{\infty} \gamma^t P_\pi [x_t = x, a_t = a] r(x, a)$$
$$= \sum_{x,a} \mu_\pi (x, a) r(x, a) = \langle \mu_\pi, r \rangle$$
ANOTHER PERSPECTIVE ON DISCOUNTED REWARDS

Discounted MDPs:

- No terminal state, initial state $x_0 \sim \mu_0$
- Discount factor $\gamma \in (0,1)$
- GOAL: maximize total discounted reward

$$R_\gamma = \mathbf{E} \left[ \sum_{t=0}^{\infty} \gamma^t r_t \right]$$

Observe: the discounted reward of a policy is

$$R_\gamma^\pi = \langle \mu_\pi, r \rangle$$

$\mu_\pi = \text{the discounted occupancy measure}$ induced by policy $\pi$:

$$\mu_\pi(x, a) = \sum_{t=0}^{\infty} \gamma^t P_\pi[x_t = x, a_t = a]$$
ANOTHER PERSPECTIVE ON DISCOUNTED REWARDS

Discounted MDPs:

- No terminal state, initial state $x_0 \sim \mu_0$
- Discount factor $\gamma \in (0, 1)$
- GOAL: maximize total discounted reward $R_\gamma = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$

Observe: the discounted reward of a policy is $R_\gamma^\pi = \langle \mu_\pi, r \rangle$

$\mu_\pi = \text{the discounted occupancy measure induced by policy } \pi$

$\mu_\pi(x, a) = \sum_{t=0}^{\infty} \gamma^t P_\pi[x_t = x, a_t = a]$
Theorem
A function $\mu$ is a discounted occupancy measure of some (stationary stochastic) policy $\pi$ if and only if it satisfies

$$
\sum_{a'} \mu(x', a') = (1 - \gamma) \sum_{a'} \mu_0(x', a') + \gamma \sum_{x,a} P(x'|x,a)\mu(x,a)
$$

and $\sum_{x,a} \mu(x,a) = 1/(1 - \gamma)$. 
Towards a linear-program formulation

Theorem
A function $\mu$ is a discounted occupancy measure of some (stationary stochastic) policy $\pi$ if and only if it satisfies

$$\sum_{a'} \mu(x', a') = (1 - \gamma) \sum_{a'} \mu_0(x', a') + \gamma \sum_{x,a} P(x'|x, a)\mu(x, a)$$

and $\sum_{x,a} \mu(x, a) = 1 / (1 - \gamma)$.

Linear constraints!
Define $\Delta = \text{the set of occupancy measures } \mu$. 
OPTIMIZATION IN MDPS AS A LINEAR PROGRAM

\[
\text{LP} \\
R_\gamma^* = \max_{\mu \in \Delta} \langle \mu, r \rangle
\]
OPTIMIZATION IN MDPs AS A LINEAR PROGRAM

\[
\text{LP} \\
R^*_\gamma = \max_{\mu \in \Delta} \langle \mu, r \rangle
\]

\[
\text{LP}' \\
R^*_\gamma = \min_{\forall x, \forall V(\in \mathbb{R})} \langle \mu_0, V \rangle \\
\text{s.t. } V(x) \geq r(x, a) + \gamma \sum_y P(y|x, a)V(y) \quad (\forall x, a)
\]
Optimization in MDPs as a Linear Program

**Dual LP**

\[ R^*_\gamma = \max_{\mu \in \Delta} \langle \mu, r \rangle \]

**Primal LP**

\[ R^*_\gamma = \min_{V \in \mathbb{R}^x} \langle \mu_0, V \rangle \]

s.t. \[ V(x) \geq r(x, a) + \gamma \sum_y P(y|x, a)V(y) \quad (\forall x, a) \]

*names are due to tradition*
Dual LP
\[ R_\gamma^* = \max_{\mu \in \Delta} \langle \mu, r \rangle \]

Primal LP \equiv \text{The Bellman opt. equations}
\[ V^*(x) = \max_a \{ r(x, a) + \gamma \sum_y P(y|x, a)V^*(y) \} \]

Assuming \( \mu_0 > 0 \)

*names are due to tradition
Optimization in MDPs as a Linear Program

Dual LP

\[ R^*_\gamma = \max_{\mu \in \Delta} \langle \mu, r \rangle \]

Primal LP \equiv The Bellman opt. equations

\[ V^*(x) = \max_a \{ r(x, a) + \gamma \sum_y P(y|x, a)V^*(y) \} \]

Assuming \( \mu_0 > 0 \)

*Names are due to tradition
Theorem
There exists a basic solution $\mu^* \in \Delta$ to the dual LP.
Theorem
There exists a basic solution $\mu^* \in \Delta$ to the dual LP.

“Proof”:
objective $\langle \mu, r \rangle$ is bounded on nonempty $\Delta$

$\Rightarrow$

there exists optimal solution $\mu^* \in \Delta$

$\Rightarrow$

there exists basic solution $\mu^* \in \Delta$
Theorem
There exists a basic solution $\mu^* \in \Delta$ to the dual LP.

“Proof”:

objective $\langle \mu, r \rangle$ is bounded on nonempty $\Delta$

$\Rightarrow$

there exists optimal solution $\mu^* \in \Delta$

$\Rightarrow$

there exists basic solution $\mu^* \in \Delta$

A “corner” of $\Delta$
Question: how do we extract a policy from a feasible $\mu \in \Delta$?
Question: how do we extract a policy from a feasible \( \mu \in \Delta \)?

**Corollary**

Assume that \( \mu_0(x) > 0 \) for all \( x \in X \). Then, for any occupancy measure \( \mu \in \Delta \), there exists a unique policy \( \pi \) such that \( \mu = \mu_\pi \), given by

\[
\pi(a|x) = \frac{\mu(x,a)}{\sum_b \mu(x,b)}.
\]
**Question:** how do we extract a policy from a feasible $\mu \in \Delta$?

**Corollary**
Assume that $\mu_0(x) > 0$ for all $x \in X$. Then, for any occupancy measure $\mu \in \Delta$, there exists a unique policy $\pi$ such that $\mu = \mu_\pi$, given by

$$
\pi(a|x) = \frac{\mu(x,a)}{\sum_b \mu(x,b)}.
$$

Well-defined since $\sum_b \mu(x,b) > 0$ by assumption.
Question: how do we extract a policy from a feasible $\mu \in \Delta$?

**Corollary**
Assume that $\mu_0(x) > 0$ for all $x \in X$. Then, for any occupancy measure $\mu \in \Delta$, there exists a unique policy $\pi$ such that $\mu = \mu_\pi$, given by

$$\pi(a|x) = \frac{\mu(x,a)}{\sum_b \mu(x,b)}.$$  

Basic solutions $\iff$ Deterministic policies

Well-defined since $\sum_b \mu(x,b) > 0$ by assumption
“Why don’t they teach this in school?!?”

• Needs some strange conditions that DP theory does not $(\mu_0 > 0$ for existence results and for optimal policy)
  • Temporal aspect is rather abstract
  • Less intuitive for control theorists and computational neuroscience folks (classic RL crowd)
LINEAR PROGRAMMING FOR MDPS

“Why don’t they teach this in school?!?”
• Needs some strange conditions that DP theory does not ($\mu_0 > 0$ for existence results and for optimal policy)
  • Temporal aspect is rather abstract
  • Less intuitive for control theorists and computational neuroscience folks (classic RL crowd)

Advantages
• Defining optimality is very simple (no value functions, no fixed points, etc.)
• Easily comprehensible with an optimization background (single numerical objective)
• Powerful tool for developing algorithms
“Why don’t they teach this in school?!?”

• Needs some strange conditions that DP theory does not ($\mu_0 > 0$ for existence results and for optimal policy)
  • Temporal aspect is rather abstract
  • Less intuitive for control theorists and computational neuroscience folks (classic RL crowd)

Advantages

• Defining optimality is very simple
  (no value functions, no fixed points, etc.)
• Easily comprehensible with an optimization background
  (single numerical objective)
• **Powerful tool for developing algorithms**
THIS SHORT COURSE: A PRIMAL-DUAL VIEW

• Markov decision processes
  • Value functions and optimal policies

• Primal view: Dynamic programming
  • Policy evaluation, value and policy iteration
  • Value-function-based methods
    • Temporal differences, Q-learning, LSTD, deep Q networks,…

• Dual view: Linear programming
  • LP duality in MDPs
  • Direct policy optimization methods
    • Policy gradients, REPS, TRPO,…
DIRECT POLICY OPTIMIZATION

Idea: derive algorithms by thinking of $\mu \in \Delta$ as the decision variable!
DIRECT POLICY OPTIMIZATION

Idea: derive algorithms by thinking of $\mu \in \Delta$ as the decision variable!

Examples

- Policy gradient methods
  $= \text{gradient descent on } - R_\gamma^\pi$

- Relative Entropy Policy Search (REPS)
  $= \text{mirror descent on } - R_\gamma^\pi$

- Trust-region policy optimization (TRPO)
  $= \text{mirror descent on (a surrogate of) } - R_\gamma^\pi$
DIRECT POLICY OPTIMIZATION

Idea: derive algorithms by thinking of $\mu \in \Delta$ as the decision variable!

Examples

- Policy gradient methods
  $= \text{gradient descent on } -R_\gamma^{\pi}$

- Relative Entropy Policy Search (REPS)
  $= \text{mirror descent on } -R_\gamma^{\pi}$

- Trust-region policy optimization (TRPO)
  $= \text{mirror descent on (a surrogate of) } -R_\gamma^{\pi}$
POLICY GRADIENT METHODS

- Construct mapping $\theta \mapsto \pi_\theta$
POLICY GRADIENT METHODS

Parameter space $\Theta$

- Construct mapping
  \[ \theta \mapsto \pi_{\theta} \]
POLICY GRADIENT METHODS

Parameter space $\Theta$

- Construct mapping $\theta \mapsto \pi_\theta$
- Define objective function:
  $$\rho(\theta) = R^\pi_\gamma$$

$(x, a) \rightarrow \pi_\theta(a|x)$
POLICY GRADIENT METHODS

Parameter space $\Theta$

- Construct mapping $\theta \mapsto \pi_\theta$
- Define objective function: $\rho(\theta) = R^\pi_\theta$
- Update parameters by gradient ascent:
  $\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta \rho(\theta_k)$
POLICY GRADIENT METHODS

- Construct mapping \( \theta \mapsto \pi_\theta \)
- Define objective function: \( \rho(\theta) = R^\pi_\theta \)
- Update parameters by gradient ascent:
  \[ \theta_{k+1} = \theta_k + \alpha_k \nabla_\theta \rho(\theta_k) \]

\( (x, a) \mapsto \pi_\theta(a|x) \)

... and hope for convergence
POLICY GRADIENT METHODS

Parameter space $\Theta$

How can we estimate the gradients?

- Construct mapping $\theta \mapsto \pi_\theta$
- Define objective function:
  $$\rho(\theta) = R^\pi_\theta$$
- Update parameters by gradient ascent:
  $$\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta \rho(\theta_k)$$

... and hope for convergence
**The Policy Gradient Theorem**

\[ \nabla_\theta \rho(\theta) = \sum_x \mu_\theta(x) \sum_a \nabla_\theta \pi_\theta(a|x) Q^{\pi_\theta}(x, a) \]
THE POLICY GRADIENT THEOREM

**Theorem**

\[ \nabla_{\theta} \rho(\theta) = \sum_{x} \mu_{\theta}(x) \sum_{a} \nabla_{\theta} \pi_{\theta}(a|x) Q^{\pi_{\theta}}(x, a) \]

**Corollary**

Assuming that \( \pi_{\theta}(a|x) > 0 \) for all \( x, a \),

\[ \nabla_{\theta} \rho(\theta) = \sum_{x,a} \mu_{\theta}(x)\pi_{\theta}(a|x) \left( \nabla_{\theta} \log \pi_{\theta}(a|x) Q^{\pi_{\theta}}(x, a) \right) \]
Theorem

\[ \nabla_\theta \rho(\theta) = \sum_x \mu_\theta(x) \sum_a \nabla_\theta \pi_\theta(a|x) Q^{\pi_\theta}(x, a) \]

Corollary

Assuming that \( \pi_\theta(a|x) > 0 \) for all \( x, a \),

\[ \nabla_\theta \rho(\theta) = \mathbb{E}_{(\tilde{x}, \tilde{a}) \sim \mu_\theta \pi_\theta} \left[ \nabla_\theta \log \pi_\theta(\tilde{a}|\tilde{x}) Q^{\pi_\theta}(\tilde{x}, \tilde{a}) \right] \]
**The Policy Gradient Theorem**

**Theorem**

\[ \nabla_\theta \rho(\theta) = \sum_x \mu_\theta(x) \sum_a \nabla_\theta \pi_\theta(a|x) Q^{\pi_\theta}(x, a) \]

Gradient can be written as an expectation!!!!

**Corollary**

Assuming that \( \pi_\theta(a|x) > 0 \) for all \( x, a \),

\[ \nabla_\theta \rho(\theta) = \mathbb{E}_{(\tilde{x}, \tilde{a}) \sim \mu_\theta \pi_\theta} [\nabla_\theta \log \pi_\theta(\tilde{a}|\tilde{x}) Q^{\pi_\theta}(\tilde{x}, \tilde{a})] \]
REINFORCE: A STOCHASTIC POLICY GRADIENT ALGORITHM

Idea: replace expectation by a sample mean ⇒ stochastic gradient algorithm
**REINFORCE: A STOCHASTIC POLICY GRADIENT ALGORITHM**

**Idea:** replace expectation by a sample mean \(\Rightarrow\) stochastic gradient algorithm

---

**REINFORCE**

**Input:** arbitrary initial \(\theta_0\)

For \(k = 0,1, \ldots\)

- Obtain sample trajectory \((x_t, a_t, r_t)^{T}_{t=1} \sim \pi_{\theta_k}\)
- Estimate \(\hat{Q}_k \approx Q_{\pi_{\theta_k}}\) by Monte Carlo
- Estimate \(g_k \approx \nabla_{\theta} \rho(\theta_k)\) by the average of
  \[
g_{k,t} = \nabla_{\theta} \log \pi_{\theta_k}(a_t | x_t) \hat{Q}_k(x_t, a_t)
  \]
- Update \(\theta_{k+1} = \theta_k + \alpha_k g_k\)
**REINFORCE: A STOCHASTIC POLICY GRADIENT ALGORITHM**

**Input:** arbitrary initial $\theta_0$

For $k = 0, 1, \ldots$

- Obtain sample trajectory $(x_t, a_t, r_t)^T_{t=1} \sim \pi_{\theta_k}$
- Estimate $\hat{Q}_k \approx Q^{\pi_{\theta_k}}$ by Monte Carlo
- Estimate $g_k \approx \nabla_{\theta} \rho(\theta_k)$ by the average of $g_{k,t} = \nabla_{\theta} \log \pi_{\theta_k}(a_t|x_t) \hat{Q}_k(x_t, a_t)$
- Update $\theta_{k+1} = \theta_k + \alpha_k g_k$

Idea: replace expectation by a sample mean $\Rightarrow$ stochastic gradient algorithm

$\mathbb{E}[g_k] = \nabla_{\theta} \rho(\theta_k)$
REINFORCE AS DIRECT POLICY SEARCH

Policy gradient update

\[ \pi_k \approx G \hat{V}_k \]

improve policy

\[ \hat{V}_{k+1} \approx V_{\pi_k} \]

evaluate policy

Monte Carlo evaluation
REINFORCE AS DIRECT POLICY SEARCH

Policy gradient update

\[ \pi_k \approx G \hat{V}_k \]

improve policy

\[ \hat{V}_k \]

evaluate policy

\[ \hat{V}_{k+1} \approx V^{\pi_k} \]

Monte Carlo evaluation

😊 direct method: no explicit approximation of \( V^\pi \)

😊 converges to local optimum

😊 less aggressive updates

😮 large variance of \( g_k \)
**Actor-Critic Methods**

Typical actor: policy gradient updates

- $\pi_k \approx G \hat{V}_k$
- $\hat{V}_{k+1} \approx V^{\pi_k}$

**Critic:**
- Monte Carlo $\Rightarrow$ REINFORCE
- TD($\lambda$)
- LSTD($\lambda$)
- DQN, …
A typical Deep RL architecture: A3C

Parametrize policy by a deep neural net

\((x, a)\) → \(\pi_\theta(a|x)\)
A TYPICAL DEEP RL ARCHITECTURE: A3C

Parametrize policy by a deep neural net

+ another neural net to estimate $V^{\pi_\theta}$ and to estimate $Q^{\pi_\theta}$ by “bootstrapped” Monte Carlo
+ asynchronous updates
+ entropy-regularization of the objective
+ …
Policy Gradients: The Final Answer?

Policy gradient update

$$\theta_{t+1} = \arg \max_{\theta} \left\{ \langle \theta, \nabla \rho(\theta_t) \rangle - \frac{1}{\alpha_t} \| \theta - \theta_t \|_2^2 \right\}$$
Policy gradient update

\[ \theta_{t+1} = \arg \max_{\theta} \left\{ \langle \theta, \nabla \rho(\theta_t) \rangle - \frac{1}{\alpha_t} \| \theta - \theta_t \|_2^2 \right\} \]

Issue #1:
Euclidean norm may be unnatural way to measure distance between \( \mu_\theta \) and \( \mu_{\theta_t} \)?
Policy gradients: the final answer?

Policy gradient update

\[ \theta_{t+1} = \arg \max_{\theta} \left\{ \langle \theta, \nabla \rho(\theta_t) \rangle - \frac{1}{\alpha_t} \| \theta - \theta_t \|^2 \right\} \]

Issue #1: Euclidean norm may be unnatural way to measure distance between \( \mu_{\theta} \) and \( \mu_{\theta_t} \)?

Issue #2: Linearizing \( \rho \) at \( \theta_t \) may lead to instability?
POLICY GRADIENTS: THE FINAL ANSWER?

Policy gradient update

$$\theta_{t+1} = \arg \max_{\theta} \left\{ \langle \theta, \nabla \rho(\theta_t) \rangle - \frac{1}{\alpha_t} \| \theta - \theta_t \|_2^2 \right\}$$

**Issue #1:**
Euclidean norm may be unnatural way to measure distance between $\mu_\theta$ and $\mu_{\theta_t}$?

**Issue #2:**
Linearizing $\rho$ at $\theta_t$ may lead to instability?

**Issue #3:**
Policy gradient estimator has huge variance 😞
A BETTER APPROACH: SMOOTHED LINEAR PROGRAMS

Dual LP

\[ R_γ^* = \max_{\mu \in \Delta} \langle \mu, r \rangle \]
A BETTER APPROACH:
SMOOTHED LINEAR PROGRAMS

Dual convex program

\[ \tilde{R}^*_γ = \max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle + \frac{1}{\eta} \Phi(\mu) \right\} \]
A BETTER APPROACH: SMOOTHED LINEAR PROGRAMS

Dual convex program

\[ \tilde{R}_γ^* = \max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle + \frac{1}{\eta} \Phi(\mu) \right\} \]

\( \Phi \): strongly convex function of \( \mu \):

- smooth optimum

\[ \mu^* = \arg \max_{\mu} \left\{ \langle \mu, r \rangle + \frac{1}{\eta} \Phi(\mu) \right\} = \frac{1}{\eta} \nabla r \Phi^*(\eta r) \]

- regularization effect \( \Rightarrow \) better generalization?
BETTER PROXIMAL REGULARIZATION: MIRROR DESCENT

Policy gradient update

\[ \theta_{t+1} = \arg \max_\theta \left\{ \langle \theta, \nabla \rho(\theta_t) \rangle - \frac{1}{\alpha_t} \| \theta - \theta_t \|_2^2 \right\} \]
Better proximal regularization: mirror descent

Policy gradient update

$$\theta_{t+1} = \arg \max_{\theta} \left\{ \langle \theta, \nabla \rho(\theta_t) \rangle - \frac{1}{\alpha_t} \| \theta - \theta_t \|_2^2 \right\}$$

Mirror descent update

$$\mu_{t+1} = \arg \max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta_t} D(\mu | \mu_t) \right\}$$
BETTER PROXIMAL REGULARIZATION: MIRROR DESCENT

Policy gradient update
\[ \theta_{t+1} = \arg\max_{\theta} \mathbb{E}_{\theta_t} \left\{ \log P_{\theta}(r) - \frac{1}{\alpha_t} \|\theta - \theta_t\|_2^2 \right\} \]

Mirror descent update
\[ \mu_{t+1} = \arg\max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta_t} D(\mu | \mu_t) \right\} \]

Proximal regularization through Bregman divergence \( D(\mu | \mu') \) (strongly convex in \( \mu \))

No need for local linearization
DIRECT POLICY OPTIMIZATION

Idea: derive algorithms by thinking of 
\( \mu \in \Delta \) as the decision variable!

Examples

- Policy gradient methods 
  = gradient descent on 
  \(- R_\gamma^{\pi}\)

- Relative Entropy Policy Search (REPS) 
  = mirror descent on \(- R_\gamma^{\pi}\)

- Trust-region policy optimization (TRPO) 
  = mirror descent on (a surrogate of) \(- R_\gamma^{\pi}\)
Mirror descent update

\[ \mu_{t+1} = \arg \max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta_t} D(\mu | \mu_t) \right\} \]

\[ D(\mu | \mu') = \sum_{x,a} \mu(x, a) \log \frac{\mu(x,a)}{\mu'(x,a)} \]
Mirror descent update:

\[ \mu_{t+1} = \arg \max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta_t} D(\mu || \mu_t) \right\} \]

Closed-form “policy update”: 

\[ \mu_{t+1}(x,a) = \mu_t(x,a) e^{\eta_t \left( r(x,a) + \gamma \mathbb{E}_{y|x,a} [\bar{V}_t(y)] - \bar{V}_t(x) \right)} \]
RELATIVE ENTROPY POLICY SEARCH
(REPS, PETERS ET AL., 2010)

Mirror descent update

\[
\mu_{t+1} = \arg\max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta_t} D(\mu | \mu_t) \right\}
\]

\[
D(\mu | \mu') = \sum_{x,a} \mu(x,a) \log \frac{\mu(x,a)}{\mu'(x,a)}
\]

Closed-form “policy update”:

\[
\mu_{t+1}(x, a) = \mu_t(x, a) e^{\eta_t \left( r(x,a) + \gamma E_y | x, a [ \tilde{V}_t(y) ] - \tilde{V}_t(x) \right)}
\]

“Value function”

\[
\tilde{V}_t = ???
\]
The REPS value function $\tilde{V}_t$ is given as the minimizer of the loss function

$$\tilde{L}(V) = \log \mathbb{E}_{x \sim \mu_t} \left[ e^{\eta_t (T \pi V(x) - V(x))} \right]$$
THE REPS VALUE FUNCTION

**Theorem**

The REPS value function $\tilde{V}_t$ is given as the minimizer of the loss function

$$\tilde{L}(V) = \log E_{x \sim \mu_t} [e^{\eta_t (T\pi V(x) - V(x))}]$$

**Proof**: Lagrangian duality.
The REPS Value Function

Theorem
The REPS value function $\tilde{V}_t$ is given as the minimizer of the loss function

$$\tilde{L}(V) = \log \mathbb{E}_{x \sim \mu_t} \left[ e^{\eta_t (T^\pi V(x) - V(x))} \right]$$

“Proof”: Lagrangian duality.

A natural competitor for the Bellman error

$$L(V) = \mathbb{E}_{x \sim \mu} \left[ (T^\pi V(x) - V(x))^2 \right]$$

Stay tuned for “deep REPS” results 😊
DIRECT POLICY OPTIMIZATION

Idea: derive algorithms by thinking of $\mu \in \Delta$ as the decision variable!

Examples

- Policy gradient methods
  $= \text{gradient descent on } -R_{\gamma}^{\pi}$

- Relative Entropy Policy Search (REPS)
  $= \text{mirror descent on } -R_{\gamma}^{\pi}$

- Trust-region policy optimization (TRPO)
  $= \text{mirror descent on (a surrogate of) } -R_{\gamma}^{\pi}$
The Bellman opt. equations

\[ V^*(x) = \max_a \{ r(x, a) + \gamma \sum_y P(y|x, a)V^*(y) \} \]
The regularized Bellman opt. equations

\[ V^*(x) = \text{softmax}_a^{\eta} \left\{ r(x, a) + \gamma \sum_y P(y|x, a)V^*(y) \right\} \]
The regularized Bellman opt. equations

\[ V^*(x) = \underset{a}{\text{softmax}}^\eta \{ r(x, a) + \gamma \sum_y P(y|x, a)V^*(y) \} \]

Used almost exclusively since \( \sim \) late 2016

- Better optimization properties:
  smooth gradients, less sensitive to errors
- Better exploration:
  optimal policy naturally stochastic, no need for \( \varepsilon \) – greedy trick
THE REGULARIZED BELLMAN EQUATIONS

The regularized Bellman opt. equations

\[ V^*(x) = \text{softmax}_a^{\eta} \{ r(x, a) + \gamma \sum_y P(y|x, a)V^*(y) \} \]

Is there a natural “dual” explanation?

Used almost exclusively since ~late 2016

• Better optimization properties:
  smooth gradients, less sensitive to errors

• Better exploration:
  optimal policy naturally stochastic, no need for \( \varepsilon \) –greedy trick
The regularized Bellman opt. equations

\[ V^*(x) = \text{softmax}_a^\eta \left\{ r(x, a) + \gamma \sum_y P(y|x, a)V^*(y) \right\} \]

?? Dual convex program ??

\[ \tilde{R}_\gamma^* = \max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta} \Phi(\mu) \right\} \]
The two formulations are connected by Lagrangian duality with the choice

\[
\Phi(\mu) = \sum_{x,a} \mu(x,a) \log \frac{\mu(x,a)}{\sum_b \mu(x,b)}
\]

\[
= \sum_x \mu(x) \sum_a \pi_\mu(a|x) \log \pi_\mu(a|x)
\]
Theorem (Neu et al., 2017)
The two formulations are connected by Lagrangian duality with the choice

\[ \Phi(\mu) = \sum_{x,a} \mu(x, a) \log \frac{\mu(x, a)}{\sum_b \mu(x, b)} \]

\[ = \sum_x \mu(x) \sum_a \pi_{\mu}(a|x) \log \pi_{\mu}(a|x) \]

The conditional entropy of \( A|X \) under \( \mu \)
Theorem (Neu et al., 2017)
The two formulations are connected by Lagrangian duality with the choice

$$\Phi(\mu) = \sum_{x,a} \mu(x,a) \log \frac{\mu(x,a)}{\sum_b \mu(x,b)}$$

$$= \sum_x \mu(x) \sum_a \pi_\mu(a|x) \log \pi_\mu(a|x)$$

The conditional entropy of $A|X$ under $\mu$

A convex function of $\mu$!
DUALITY THEORY FOR
THE REGULARIZED BELLMAN EQUATIONS

The regularized Bellman opt. equations
\[ V^*(x) = \text{softmax}_a^{\eta} \{ r(x, a) + \gamma \sum_y P(y|x, a)V^*(y) \} \]

Dual convex program
\[ \tilde{R}_\gamma^* = \max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta} \Phi(\mu) \right\} \]
Mirror descent with conditional entropy (Neu et al., 2017)

Mirror descent update

$$\mu_{t+1} = \arg \max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta_t} D_\Phi(\mu | \mu_t) \right\}$$

$$D_\Phi(\mu | \mu_t) = \sum_{x,a} \mu(x, a) \log \frac{\pi_\mu(a|x)}{\pi_t(x,a)}$$
MIRROR DESCENT WITH CONDITIONAL ENTROPY (NEU ET AL., 2017)

Mirror descent update

$$\mu_{t+1} = \arg \max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta_t} D_\Phi(\mu|\mu_t) \right\}$$

$$D_\Phi(\mu|\mu_t) = \sum_{x,a} \mu(x,a) \log \frac{\pi_\mu(a|x)}{\pi_t(x,a)}$$

Closed-form policy update:

$$\pi_{t+1}(a|x) = \pi_t(a|x) e^{\eta_t \left[ r(x,a) + \gamma E_{y|x,a} [\bar{V}_t(y)] - \bar{V}_t(x) \right]}$$
**Mirror Descent with Conditional Entropy (Neu et al., 2017)**

**Mirror descent update**

\[
\mu_{t+1} = \arg \max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta_t} D_{\Phi}(\mu | \mu_t) \right\}
\]

\[
D_{\Phi}(\mu | \mu_t) = \sum_{x,a} \mu(x, a) \log \frac{\pi_{\mu}(a|x)}{\pi_{t}(x,a)}
\]

**Closed-form policy update:**

\[
\pi_{t+1}(a|x) = \pi_t(a|x)e^{\eta_t(r(x,a)+\gamma E_{y|x,a}[\tilde{V}_t(y)]-\tilde{V}_t(x))}
\]

Value function \(\tilde{V}_t = \) solution to proximally regularized BOE
TRUST-REGION POLICY OPTIMIZATION (TRPO, SCHULMAN ET AL., 2015)

Mirror descent update

\[ \mu_{t+1} = \arg \max_{\mu \in \Delta} \left\{ \langle \mu, r \rangle - \frac{1}{\eta_t} D_\Phi(\mu|\mu_t) \right\} \]

\[ D_\Phi(\mu|\mu_t) = \sum_{x,a} \mu(x,a) \log \frac{\pi_\mu(a|x)}{\pi_t(x,a)} \]
Mirror descent update

\[ \mu_{t+1} = \arg \max_{\mu \in \Delta} \left\{ \langle \mu, \tilde{Q}_t - \tilde{V}_t \rangle - \frac{1}{\eta_t} D_{\Phi}(\mu|\mu_t) \right\} \]

\[ D_{\Phi}(\mu|\mu_t) = \sum_x \mu_t(x) \sum_a \pi_\mu(a|x) \log \frac{\pi_\mu(a|x)}{\pi_t(x,a)} \]
TRUST-REGION POLICY OPTIMIZATION (TRPO, SCHULMAN ET AL., 2015)

Mirror descent update

$$
\mu_{t+1} = \arg \max_{\mu \in \Delta} \left\{ \langle \mu, \tilde{Q}_t - \tilde{V}_t \rangle - \frac{1}{\eta_t} D_{\Phi}(\mu|\mu_t) \right\}
$$

$$
D_{\Phi}(\mu|\mu_t) = \sum_x \mu_t(x) \sum_a \pi_\mu(a|x) \log \frac{\pi_\mu(a|x)}{\pi_t(x,a)}
$$

Dense surrogate for $\langle \mu, r \rangle$
(works because $\langle \mu, r \rangle = \langle \mu, \tilde{Q}_t - \tilde{V}_t \rangle$ when $\mu \in \Delta$)

$$
\mu_t \approx \mu_{t+1}, \text{ but } \mu_t \text{ can be sampled from}
$$
Theorem (Neu et al., 2017)
TRPO is equivalent to the MDP-E algorithm of Even-Dar, Kakade and Mansour (2006)

$$\lim_{t \to \infty} \langle \mu_t, r \rangle = \langle \mu^*, r \rangle$$
TRUST-REGION POLICY OPTIMIZATION (TRPO, SCHULMAN ET AL., 2015)

Theorem (Neu et al., 2017)
TRPO is equivalent to the MDP-E algorithm of Even-Dar, Kakade and Mansour (2006)

\[ \lim_{t \to \infty} \langle \mu_t, r \rangle = \langle \mu^*, r \rangle \]

+ more tricks:
  - Another surrogate for \( \mu \)
  - Truncation of objective
  - Constraint vs. penalty
  - Mini-batch SGD
  - …
TRUST-REGION POLICY OPTIMIZATION (TRPO, SCHULMAN ET AL., 2015)

**Theorem** (Neu et al., 2017)
TRPO is equivalent to the MDP-E algorithm of Even-Dar, Kakade and Mansour (2006)

\[ \lim_{t \to \infty} \langle \mu_t, r \rangle = \langle \mu^*, r \rangle \]

+ more tricks:
  - Another surrogate for \( \mu \)
  - Truncation of objective
  - Constraint vs. penalty
  - Mini-batch SGD
  - …

Literally the most broadly used deep RL algorithm!
(but reading the original paper is not recommended…)

\[
Q(\theta | x, a) = \pi (a | x) \]

\[
\pi (a | x) = \frac{e^{\theta a}}{\sum_{a'} e^{\theta a'}}
\]

\[
\phi (x) = \sum_{a} \pi (a | x) \phi (x, a)
\]
BEYOND LINEAR PROGRAMMING: SADDLE-POINT OPTIMIZATION

Dual LP

\[ R^*_\gamma = \max_{\mu \in \Delta} \langle \mu, r \rangle \]

Primal LP

\[ R^*_\gamma = \min_{V \in \mathbb{R}^x} \langle \mu_0, V \rangle \]

s.t. \( V(x) \geq r(x, a) + \gamma \sum_y P(y|x, a) V(y) \) (\( \forall x, a \))
Bellman saddle point

$$\min_V \max_{\mu \in \Delta} \left\{ \langle \mu, r + \gamma PV - V \rangle + (1 - \gamma) \langle \mu_0, V \rangle \right\}$$
Bellman saddle point

\[
\min_{V} \max_{\mu \in \Delta} \{ \langle \mu, \mathbf{r} + \gamma \mathbf{P} V - V \rangle + (1 - \gamma) \langle \mu_0, V \rangle \}
\]

\[ \approx \text{the Lagrangian of the two LPs} \]

\[ \Rightarrow \]

solution exists & optimal policy can be extracted under same conditions
**PRIMAL-DUAL \( \pi \)-LEARNING**

(WANG ET AL., 2017-)

Bellman saddle point

\[
\min_{V} \max_{\mu \in \Delta} \{ \langle \mu, r + \gamma PV - V \rangle + (1 - \gamma) \langle \mu_0, V \rangle \}
\]
**PRIMAL-DUAL π-LEARNING**
*(WANG ET AL., 2017-)*

**Bellman saddle point**

$$\min_V \max_{\mu \in \Delta} \{ \langle \mu, r + \gamma PV - V \rangle + (1 - \gamma) \langle \mu_0, V \rangle \}$$

**Value update:**

$$\tilde{V}_{t+1} = \tilde{V}_t + \alpha_t (\mu_t - \gamma \mu_t P)$$

**Policy update:**

$$\mu_{t+1}(x, a) = \mu_t(x, a) e^{\eta_t \left( r(x,a) + \gamma E_{y|x,a}[\tilde{V}_t(y)] - \tilde{V}_t(x) \right)}$$
PRIMAL-DUAL $\pi$-LEARNING (WANG ET AL., 2017-)

Bellman saddle point

$$\min_{\hat{V}} \max_{\mu \in \Delta} \{\langle \mu, r + \gamma \hat{V} - V \rangle + (1 - \gamma)\langle \mu_0, V \rangle \}$$

Value update:

$$\hat{V}_{t+1} = \hat{V}_t + \alpha_t (\mu_t - \gamma \mu_t P)$$

Policy update:

$$\mu_{t+1}(x, a) = \mu_t(x, a) e^{\eta_t (r(x,a) + \gamma E_y|x,a[\hat{V}_t(y)] - \hat{V}_t(x))}$$

Gradient step in primal

Exponentiated gradient step in dual
PRIMAL-DUAL $\pi$-LEARNING (WANG ET AL., 2017-)

Bellman saddle point

$$\min_V \max_{\mu \in \Delta} \{ \langle \mu, r + \gamma PV - V \rangle + (1 - \gamma)\langle \mu_0, V \rangle \}$$

$\approx$ incremental REPS

state-of-the-art sample complexity results for discounted & undiscounted MDPs!

$$\mu_{t+1}(x, a) = \mu_t(x, a) e^{\nabla_t^P (V_t - V_t^P)} = \mu_t(x, a) e^{\nabla_t^P [\tilde{V}_t(y) - \tilde{V}_t(x)]}$$
HIS SHORT COURSE: A PRIMAL-DUAL VIEW

- Markov decision processes
  - Value functions and optimal policies

- Primal view: Dynamic programming
  - Policy evaluation, value and policy iteration
  - Value-function-based methods
    - Temporal differences, Q-learning, LSTD, deep Q networks,…

- Dual view: Linear programming
  - LP duality in MDPs
  - Direct policy optimization methods
    - Policy gradients, REPS, TRPO,…
THIS SHORT COURSE: A PRIMAL-DUAL VIEW

- Markov decision processes
  - Value functions and optimal policies

- Primal view: Dynamic programming
  - Policy evaluation, value and policy iteration
  - Value-function-based methods
    - Temporal differences, Q-learning, LSTD, deep Q networks,…

- Dual view: Linear programming
  - LP duality in MDPs
  - Direct policy optimization methods
    - Policy gradients, REPS, TRPO,…

what else?
EXPLORATION VS. EXPLOITATION

- state
- actions
- next state
- reward?
EXPLORATION VS. EXPLOITATION

- Multi-armed bandits
- Exploration bonuses
- Thompson sampling
- Monte Carlo tree search
- ...
EXPLORATION VS. EXPLOITATION

Still no practical algorithms!

• Multi-armed bandits
• Exploration bonuses
• Thompson sampling
• Monte Carlo tree search
  • …
CONCLUSION

RL is an insanely popular field with

- huge recent successes
- some beautiful fundamental theory
- unique algorithmic ideas
CONCLUSION

RL is an insanely popular field with
- huge recent successes
- some beautiful fundamental theory
- unique algorithmic ideas

BUT still fundamental challenges in
- understanding efficient exploration
- stability of algorithms
- generalizability of successes
CONCLUSION

Come and work on RL theory ;)

BUT still fundamental challenges in
  • understanding efficient exploration
  • stability of algorithms
  • generalizability of successes
CONCLUSION

Come and work on RL theory ;)

Thanks!!!

+ also come see PARADISE LOST tonight!