UNIVERSITY OF TORONTO | Engineering

Value Function-based Method

- Temporal Difference
- SARSA and Q-Learning
- n-step Temporal Difference

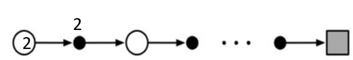
39

Learning Algorithm to MDP

- Estimation by Sampling

$R_{t+1} + \gamma R_{t+2} + \cdots | S_t = 2, A_t = 2$

$$\begin{bmatrix} q(1,1) & q(1,2) & \cdots & q(1,|A|) \\ q(2,1) & \boxed{q(2,2)} & & q(2,|A|) \\ \vdots & \vdots & \ddots & \vdots \\ q(|S|,1) & q(|S|,2) & \cdots & \boxed{q(|S|,|A|)} \end{bmatrix}$$



$q(2,2) = R_{t+1} + R_{t+2} + \cdots$

$R_{t+1} + \gamma R_{t+2} + \cdots | S_t = |S|, A_t = |A|$

UNIVERSITY OF TORONTO | Engineering

40

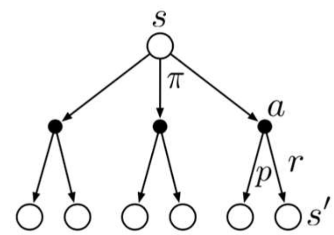
Bootstrapping

- Simulation only without bootstrapping

$$v_{k+1}(s) = \frac{1}{c_s}\sum v_t(s)$$

where $v_t(s) = \{R_t(s) + \gamma R_{t+1}(S_{t+1}) + \cdots\}$

- One simulation with max bootstrapping



$$v_{k+1}(s) = R_{t+1} + \gamma \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) v_k(s')$$

- There should be something in between → **Temporal Difference** (TD)

Temporal Difference Learning

- Temporal Difference (Error)

$$\delta_{t+1} = \underbrace{\underbrace{R_{t+1}}_{\text{Observation}} + \underbrace{\gamma V(S_{t+1})}_{\text{Bootstrapping}}}_{\text{New estimate}} - \underbrace{V(S_t)}_{\text{Original estimate}}$$

  - New estimate = observation + bootstrapped value function
  - Old estimate = original value function
  - Hence, TD is the error between new estimate and old estimate

Monte Carlo to Temporal Difference

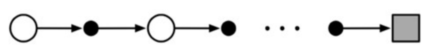
- The same learning equation $V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big)$

- Return in Monte Carlo

  $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \cdots$

  Then the learning equation becomes

  $V(S_t) \leftarrow V(S_t) + \alpha\big(R_{t+1} + \gamma R_{t+2} \cdots - V(S_t)\big)$

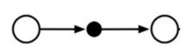


- Return in Temporal Difference

  $G_t = R_{t+1} + \gamma V(S_{t+1})$

  Then the learning equation becomes

  $V(S_t) \leftarrow V(S_t) + \alpha\big(\underbrace{R_t + \gamma V(S_{t+1}) - V(S_t)}_{\text{Temporal Difference}}\big)$



43

Temporal Difference Learning

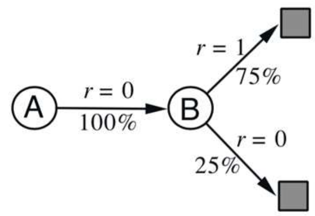
- Comparison between TD and MC

TD	MC
Learning upon transition	Learning at the end of an episode
Learning from incomplete episode	Learning from complete episode
Can be used in continuing task	Can be used only with episodic task
Biased but small variance	Unbiased but high variance
Sensitive to the initial values	Insensitive to the initial values
More efficient	Easy to use

44

Example 6.4: You are the Predictor Place yourself now in the role of the predictor of returns for an unknown Markov reward process. Suppose you observe the following eight episodes:

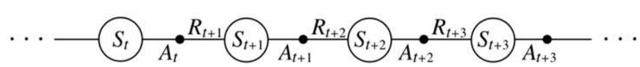
A, 0, B, 0	B, 1
B, 1	B, 1
B, 1	B, 1
B, 1	B, 0



- Optimal estimation for V(B)=?
  - 6 out of 8 times in B, the process terminated with an immediate reward 1
  - 2 out of 8 times in B, the process terminated with an immediate reward 0
  - Hence, on average V(B)=3/4
- Optimal estimation for V(A)=?
  - By TD(0), After A, immediate reward = 0 + bootstrap B (=3/4) = 3/4
  - By MC, single episode starting at A has a total return of 0

SARSA

- Given an episode following a behavior policy π



- Learning Equations
  - Monte Carlo
    $V(S_t) \leftarrow V(s_t) + \alpha\big(G_t - V(S_t)\big)$
  - With Temporal Difference
    $V(S_t) \leftarrow V(s_t) + \alpha\big(R_t + \gamma V(S_{t+1}) - V(S_t)\big)$
  - Learning Q rather than V

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big].$$

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

- Q: Where is the policy improved?

UNIVERSITY OF TORONTO | Engineering

47

On-policy vs. Off-policy Learning

- **On-policy**
  - Learn a policy with data collected by the policy
  - Estimate mean of a distribution with samples from the distribution
  - Stable but sample inefficient
  - Example: SARSA

- **Off-policy**
  - Learn a policy with data collected by different policies
  - Estimate mean of a distribution with sample from different distribution
  - Unstable but sample efficient
  - Example: Q-Learning

UNIVERSITY OF TORONTO | Engineering

48

Q-Learning

- From SARSA to Q-Learning

SARSA: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big].$

Optimistic boostrapping

- Value iteration algorithm
- Off-policy
- Optimization bias

Q-Learning: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big].$

- Why the new equation makes the algorithm **off-policy**?
  - What is the target and the behavior policy in Q-learning equation?

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
        $S \leftarrow S'$
    until S is terminal

- Identify behavior policy and target policy

Function Approximation

- Approximation of State and Action
- Features and Approximation
- Deep Q-Learning and other Extensions

51

Learning Algorithm to MDP

- Estimation by Sampling

$R_{t+1} + \gamma R_{t+2} + \cdots | S_t = 2, A_t = 2$

$$\begin{bmatrix} q(1,1) & q(1,2) & \cdots & q(1,|A|) \\ q(2,1) & q(2,2) & & q(2,|A|) \\ \vdots & \vdots & \ddots & \vdots \\ q(|S|,1) & q(|S|,2) & \cdots & q(|S|,|A|) \end{bmatrix}$$

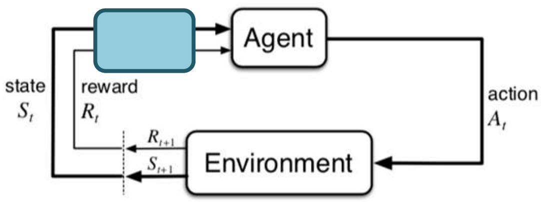
$q(2,2) = R_{t+1} + R_{t+2} + \cdots$

$R_{t+1} + \gamma R_{t+2} + \cdots | S_t = |S|, A_t = |A|$

52

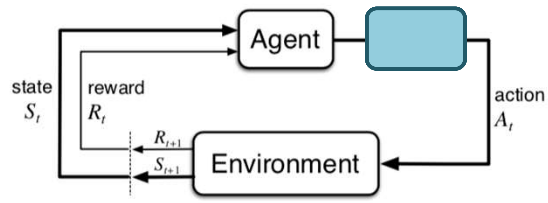
Approximation

- RL + Supervised Learning



  - Linear combination of basis functions
  - Least square fit
  - Non-linear approximation: DL

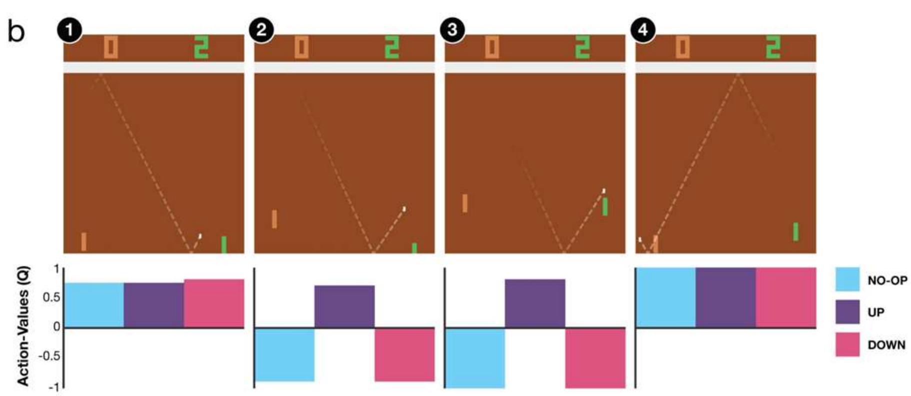
- Policy Learning via Gradient Descent



  - The policy gradient theorem
  - REINFORCE, DPG, DDPG, TRPO, PPO, etc.

53

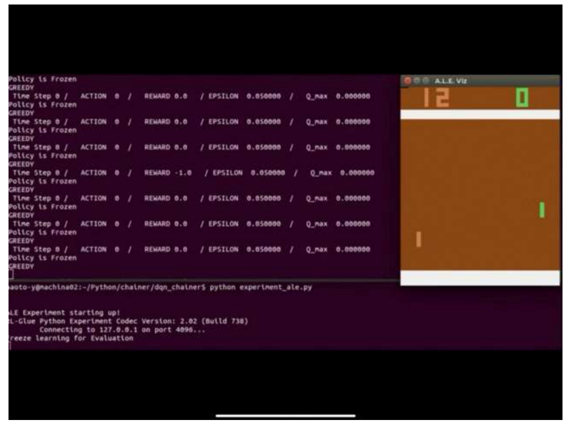
Action-Value Function



54

Deep Q-Net with Experience Replay

Algorithm 1: deep Q-learning with experience replay.
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
For episode $= 1, M$ **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For



UNIVERSITY OF TORONTO Engineering

55

Function Approximation

- Motivation
  - Curse of dimensionality
  - Generalization
- Example: Tic-Tac-Toe
  - State space
    - 3X3 board
    - Each position can be "empty", "O" or "X"
    - There are 3^9 $(= 19{,}683)$ states
  - State space can be compressed using features

1	2	3
4	5	6
7	8	9

		X
X	O	O
O	X	O

$\phi_1(s) = 1$ if "X" is at the centre; 0 otherwise $\phi_1(s) = 0$

$\phi_2(s) =$ # of corner cells with "X". $\phi_2(s) = 1$

$\phi_3(s) =$ # of instances of adjacent cells with "X". $\phi_3(s) = 0$

UNIVERSITY OF TORONTO Engineering

56

- Linear approximation given the features

$$\tilde{V}_t(s) = \sum_{f \in F} \theta_f \phi_f(s)$$

		X
X	O	O
O	X	O

$$\tilde{V}_t(s) = \theta_1\phi_1(s) + \theta_2\phi_2(s) + \theta_3\phi_3(s)$$
$$= \theta_1 \times 0 \;+\; \theta_2 \times 1 \;+\; \theta_3 \times 0$$
$$= \theta_2$$

Dimensionality = 9

Dimensionality = 3

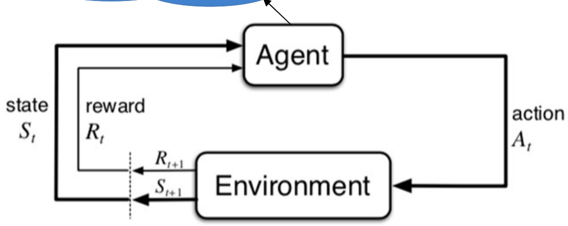
UNIVERSITY OF TORONTO Engineering

57

RL with Function Approximation

$\theta_{t+1} \leftarrow \theta_t - \alpha \cdot \nabla\vec{e}$, where $Q(s,a;\theta_{t+1})$

$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\delta_t$



Approximation Error:

$$e(\theta) = \sum_{s \in S} \{V_\pi(s) - \tilde{V}(s;\theta)\}^2$$
$$= \sum_{s \in S} \mu(s)\{V_\pi(s) - \tilde{V}(s;\theta)\}^2$$
$$= \sum_{s \in S} \mu(s) \sum_a \pi(a|s)\left\{Q_\pi(s,a) - \tilde{Q}(s,a;\theta)\right\}^2$$

UNIVERSITY OF TORONTO Engineering

58

RL with Function Approximation

- General Algorithm Structure

Input Differentiable parameterized function $f(s,a;\theta)$ for $Q(s,a)$: $S \times A \times \mathbb{R}^d \to \mathbb{R}$, step size α, small $\epsilon > 0$

Initialize Initialize the value function parameter $\theta \in \mathbb{R}^d$

Loop for each episode

$S, A \leftarrow$ initial state and action of episode by ϵ-greedy

Loop for each step of episode

Take action A and observe R, S'

If S' is terminal, exit the loop

Choose A' via ϵ-greedy based on $f(S',\cdot;\theta)$

$$\boldsymbol{\theta \leftarrow \theta + \alpha\{R + \gamma f(S', A'; \theta) - f(S, A; \theta)\} \cdot \nabla f(S, A; \theta)}$$

$S \leftarrow S'; A \leftarrow A'$

End of Loop

End of Loop

$$L(\theta) = \mathbb{E}_{S,A,S',A\sim\rho}\{R + \gamma f(S',A';\theta) - f(S,A;\theta)\}^2$$

$$\nabla_\theta L(\theta) = \mathbb{E}_{S,A,S',A\sim\rho}\{R + \gamma f(S',A';\theta) - f(S,A;\theta)\}\nabla_\theta f(S,A;\theta)$$

UNIVERSITY OF TORONTO | Engineering

59

Deep Q-Learning

- The Same Old Q Learning Equation

$$Q^*(s,a) = \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s',a') \middle| s,a\right]$$

- Fit a Function $Q(s,a;\theta_i)$ to Samples y_i for $Q(s,a)$
  - Sample

$$y_i \;=\; \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max\nolimits_{a'} Q(s',a';\theta_{i-1}) | s,a\right]$$

  - Loss for fitting

$$L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[(y_i - Q(s,a;\theta_i))^2\right],$$

  - Gradient to improve the fit

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot);s'\sim\mathcal{E}}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i)\right)\nabla_{\theta_i} Q(s,a;\theta_i)\right]$$

UNIVERSITY OF TORONTO | Engineering

60

Algorithm 1: deep Q-learning with experience replay.
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
For episode $= 1, M$ **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \mathrm{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}\left(\phi_{j+1}, a'; \theta^-\right) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q\left(\phi_j, a_j; \theta\right)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

ϵ-greedy

Response from Env.

TD-based Sample

Improve the fit (regression)

Update the target net

UNIVERSITY OF TORONTO | Engineering

61

Deep Q-Net

- Challenges
  - Catching dynamics
  - Samples are not independent
  - Poor stability
- Remedies
  - Use features carrying historical information
    Inputs include the last 4 consecutive screenshots
  - Experience replay
    Experiences for training are sampled according to uniform distribution from buffer
  - Delayed learning via two networks
    The current and the target networks

UNIVERSITY OF TORONTO | Engineering

62

Extensions of DQN

- Double DQN

$$y_i^{DDQN} = r + \gamma Q(s', \arg\max_{a'} Q(s', a'; \theta_i); \theta^-)$$

  - This is to mitigate the optimality bias

- Prioritized experience replay
  - Which experience to sample from the buffer?
  - Stochastic sampling driven by temporal difference

$$\delta_j = R_j + \gamma_j Q_{\text{target}}\left(S_j, \arg\max_a Q(S_j, a)\right) - Q(S_{j-1}, A_{j-1})$$

$$p_j \leftarrow |\delta_j|$$

$$\text{Sample transition } j \sim P(j) = p_j^\alpha / \sum_i p_i^\alpha$$

UNIVERSITY OF TORONTO | Engineering

63

UNIVERSITY OF TORONTO | Engineering

Policy-based Method

- Policy Gradient Theorem
- REINFORCE: a Monte Carlo Policy Gradient Algorithm
- Actor-Critic Methods

64

Value Function-based vs. Policy-based Learning

- RL so far has been Value Function-based
  - Learning (action) value function
  - Acting optimally given the value function
- An Alternative is Policy-based Learning
  - Optimize policy directly using gradient
  - Good for large action space (even continuous action space)
  - Algorithms are better behaving thanks to gradient-descent optimization
  - Mostly converges to local optimal policy

UNIVERSITY OF TORONTO | Engineering

65

3 Steps in Policy Gradient

- Step1: Policy Parameterization

  Let $\theta \in \mathbb{R}^d$ be the policy's parameter vector, then policy is

$$\pi(a|s,\theta) = Pr\{A_t = a|S_t = s, \theta_t = \theta\}$$

- Step 2: Performance Parameterization

$$J(\theta) \triangleq v_{\pi_\theta}(s_0), \qquad \forall\, s_0$$

- Step 3: Performance Improvement using Stochastic Gradient

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \widehat{\nabla J}(\theta_t)$$

UNIVERSITY OF TORONTO | Engineering

66

Policy Gradient Theorem

- Can we actually compute the gradient?

$$
\begin{tikzcd}
 & \pi_\theta(a|s) \arrow[r] & \mu(s) \arrow[dl] \\
\theta \arrow[ur] \arrow[r] & J(\theta) \arrow[r] & \nabla J(\theta)
\end{tikzcd}
$$

- Answer = Yes by Policy Gradient Theorem

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$

UNIVERSITY OF TORONTO | Engineering

67

- Policy Gradient Theorem

$$
\begin{aligned}
\nabla v_\pi(s) &= \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right], \quad \text{for all } s \in \mathcal{S} && \text{(Exercise 3.18)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \Big] && \text{(product rule of calculus)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r|s,a)\big(r + v_\pi(s')\big) \Big] \\
& && \text{(Exercise 3.19 and Equation 3.2)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \nabla v_\pi(s') \Big] && \text{(Eq. 3.4)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) && \text{(unrolling)} \\
&\qquad \sum_{a'} \big[\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \big] \Big] \\
&= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla \pi(a|x) q_\pi(x,a),
\end{aligned}
$$

UNIVERSITY OF TORONTO | Engineering

68

- Policy Gradient Theorem

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \nabla v_\pi(s_0) \\
&= \sum_s \left(\sum_{k=0}^{\infty} \Pr(s_0 \to s, k, \pi) \right) \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&= \sum_s \eta(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&= \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&= \sum_{s'} \eta(s') \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&\propto \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a)
\end{aligned}
$$

UNIVERSITY OF TORONTO | Engineering

69

- Policy Gradient Theorem

$$
\begin{aligned}
\nabla v_\pi(s) &= \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right], \quad \text{for all } s \in \mathcal{S} && \text{(Exercise 3.18)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \Big] && \text{(product rule of calculus)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r|s,a)\big(r + v_\pi(s')\big) \Big] \\
& && \text{(Exercise 3.19 and Equation 3.2)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \nabla v_\pi(s') \Big] && \text{(Eq. 3.4)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) && \text{(unrolling)} \\
&\qquad \sum_{a'} \big[\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \big] \Big] \\
&= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla \pi(a|x) q_\pi(x,a),
\end{aligned}
$$

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \nabla v_\pi(s_0) \\
&= \sum_s \left(\sum_{k=0}^{\infty} \Pr(s_0 \to s, k, \pi) \right) \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&= \sum_s \eta(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&= \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&= \sum_{s'} \eta(s') \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&\propto \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a)
\end{aligned}
$$

UNIVERSITY OF TORONTO | Engineering

70

Monte Carlo Policy Gradient Method (REINFORCE)

From **policy gradient theorem** to **REINFORCE**,

$$\begin{aligned}
\nabla J(\boldsymbol{\theta}) &\propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta}) \\
&= \mathbb{E}_\pi\left[\sum_a q_\pi(S_t,a)\nabla\pi(a|S_t,\boldsymbol{\theta})\right]. \qquad (13.6) \\
&= \mathbb{E}_\pi\left[\sum_a \pi(a|S_t,\boldsymbol{\theta}) q_\pi(S_t,a) \frac{\nabla\pi(a|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})}\right] \\
&= \mathbb{E}_\pi\left[q_\pi(S_t,A_t)\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right] \qquad \text{(replacing } a \text{ by the sample } A_t \sim \pi) \\
&= \mathbb{E}_\pi\left[G_t\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right], \qquad \text{(because } \mathbb{E}_\pi[G_t|S_t,A_t] = q_\pi(S_t,A_t))
\end{aligned}$$

where G_t is the usual return, which is estimated by Monte Carlo here.

UNIVERSITY OF TORONTO | Engineering

71

- Policy Grad Theorem to REINFORCE

$$\nabla J(\theta) = E_\pi\left[G_t \frac{\nabla\pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}\right] \longrightarrow \widehat{\nabla J}(\theta) = G_t\frac{\nabla\pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}$$

  - Learning equation

$$\begin{aligned}
\theta_{t+1} &\triangleq \theta_t + \alpha G_t \frac{\nabla\pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)} \\
&= \theta_t + \alpha \cdot G_t \cdot \boldsymbol{\nabla \ln \pi(A_t|S_t,\theta_t)}
\end{aligned}$$

Log Derivative Trick

Recall $\nabla \ln x = \frac{\nabla x}{x}$

Hence, we have

$$\nabla \ln \pi(A_t|S_t,\theta_t) = \frac{\nabla\pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}$$

Therefore,

$$\theta_{t+1} \triangleq \theta_t + \alpha G_t \nabla \ln \pi(A_t|S_t,\theta_t)$$

UNIVERSITY OF TORONTO | Engineering

72

REINFORCE

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\gamma^t G \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$

Actor-Critic Methods

- Advantage Function $A(S_t)$

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$

$$\sum_a b(s) \nabla \pi(a|s,\boldsymbol{\theta}) \;=\; b(s)\nabla \sum_a \pi(a|s,\boldsymbol{\theta}) \;=\; b(s)\nabla 1 \;=\; 0.$$

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a \Big(q_\pi(s,a) - b(s)\Big) \nabla \pi(a|s,\boldsymbol{\theta})$$

- Learning Equation

$V(S_t)$

$$\widehat{\nabla J}(\theta) = \big(G_t - b(S_t)\big) \cdot \nabla \ln \pi(A_t|S_t,\theta_t) = \boldsymbol{A}(\boldsymbol{S_t}) \cdot \nabla \ln \pi(A_t|S_t,\theta_t)$$

Monte Carlo Actor-Critic Algorithm

REINFORCE with Baseline (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Algorithm parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
    $\delta \leftarrow G - \hat{v}(S_t,\mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t,\mathbf{w})$ ← value learning
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$ ← policy grad

UNIVERSITY OF TORONTO | Engineering

75

1-Step Actor-Critic Method

One-step Actor–Critic (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
  Initialize S (first state of episode)
  $I \leftarrow 1$
  Loop while S is not terminal (for each time step):
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

UNIVERSITY OF TORONTO | Engineering

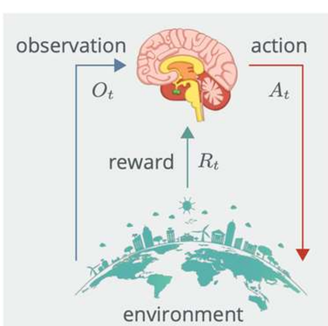
76

Engineering

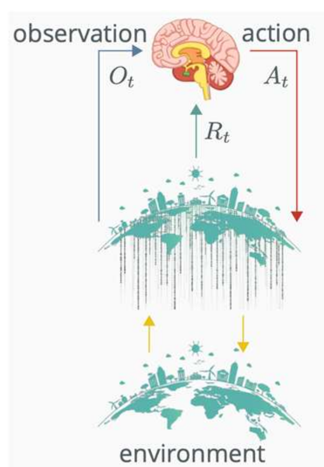
Model-based Reinforcement Learning

77

Model-Free vs. Model-Based RL



vs.



Engineering

78

Model-Free vs. Model-Based RL

- Model-free learns policy directly without learning any model
- Model-based learns models first, which is then be used to find a policy
- Why model-based?
  1. More sample efficient
  2. Efficient exploration
  3. Avoid trial-and-error on a real physical system
- Challenges of model-based RL?
  1. Learning models is more challenging than policy
  2. Requires more assumptions than model-free
  3. Optimization exploits the mis-fit in learned model (objective mismatch)
  4. Errors accumulate during unrolling

UNIVERSITY OF TORONTO | Engineering

79

Model-based Reinforcement Learning

- Models
  - Anything to predict how the environment will respond
  - Specifically, they involve state transition $T(s'|s,a)$ and reward $R(s,a)$
- General Structure
  1. Act on the environment | state
  2. Observe feedbacks
  3. Learn the model(s)
  4. Plan using the model
  5. Update the value function and/or policy

UNIVERSITY OF TORONTO | Engineering

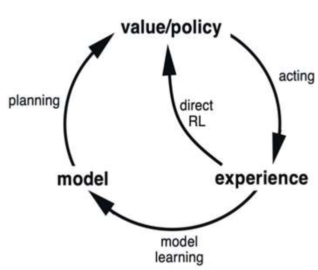
80

Model-based Reinforcement Learning

- How to learn models?
  - Supervised learning given experience tuples (s, a, s', r)
  - $s, a \to r$ is a regression problem $\quad R_{t+1} = \mathcal{R}_\eta(R_{t+1} \mid S_t, A_t)$
  - $s, a \to s'$ is a density estimation problem $\quad S_{t+1} \sim \mathcal{P}_\eta(S_{t+1} \mid S_t, A_t)$

- Types of Models
  1. Lookup table
  2. Linear model
  3. Linear Gaussian model
  4. Gaussian process
  5. Deep belief network

Dyna-Q



- Assuming deterministic environment

Tabular Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Loop forever:
  (a) $S \leftarrow$ current (nonterminal) state
  (b) $A \leftarrow \varepsilon$-greedy(S, Q)
  (c) Take action A; observe resultant reward, R, and state, S'
  (d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
  (e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
  (f) Loop repeat n times:
    $S \leftarrow$ random previously observed state
    $A \leftarrow$ random action previously taken in S
    $R, S' \leftarrow Model(S,A)$
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

Dyna-Q with Stochastic Environment

- Lookup Table Model
  - After taking action a from state s at time T
    - Transition dynamics (using table of $|S| \times |A| \times |S|$)

$$\hat{\mathcal{P}}^a_{s,s'} = \frac{1}{N(s,a)} \sum_{t=1}^{T} \mathbf{1}(S_t, A_t, S_{t+1} = s, a, s')$$

    - Reward (using table of $|S| \times |A|$)

$$\hat{\mathcal{R}}^a_s = \frac{1}{N(s,a)} \sum_{t=1}^{T} \mathbf{1}(S_t, A_t = s, a) R_t$$

where $\mathbf{1}(\cdot)$ is the indicator function

$N(s,a)$ is a visitation counter (i.e., $N(s,a) = \sum_t \mathbf{1}(S_t = s, A_t = a)$)

UNIVERSITY OF TORONTO | Engineering

83

Dyna-Q with Stochastic Environment

- Sampling experience from two sources
  1. Sample experience from the model

$$S' \sim \mathcal{P}_\eta(S' \mid S, A)$$
$$R = \mathcal{R}_\eta(R \mid S, A)$$

  2. Sample experience from the environment

$$S' \sim \mathcal{P}^a_{s,s'}$$
$$R = \mathcal{R}^a_s$$

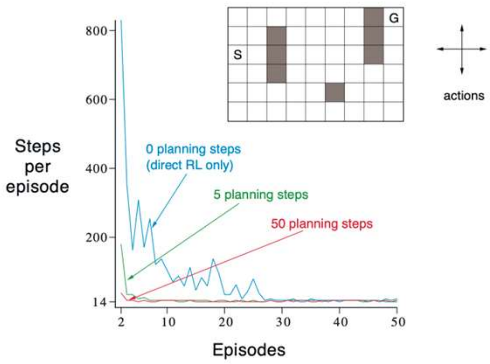
- Given samples, model-free RL can be applied

UNIVERSITY OF TORONTO | Engineering

84

Dyna-Q with varying planning horizon

- Performance comparison w.r.t. planning steps (n)

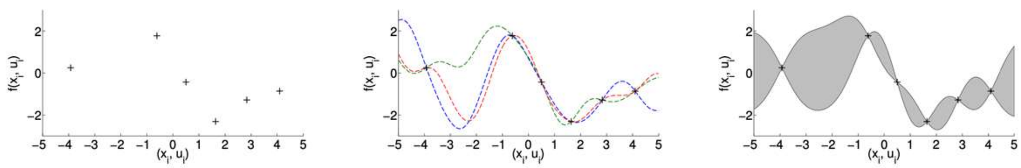


  - Longer planning results in a better performance

85

PILCO (Probabilistic Inference for Learning COntrol)

- Use Gaussian process to learn dynamics model



  - x_i is the current state, u_i the control and $f(x_i, u_i)$ the change in state given x_i and u_i

- Gaussian Process
  - A probability distribution over functions $y(x)$ such that $y(x_1), y(x_2), \dots, y(x_N)$ at an arbitrary set of points $x_1, x_2, \dots, x_N$ are jointly Gaussian.
  - More generally, a stochastic process $y(x)$ is specified by the joint probability distribution for any finite set of values $y(x_1), y(x_2), \dots, y(x_N)$.

86

PILCO

Algorithm 1 PILCO

1: *Define* policy's functional form: $\pi : z_t \times \psi \rightarrow u_t$.
2: *Initialise* policy parameters ψ randomly.
3: **repeat**
4: *Execute* system, record data.
5: *Learn* dynamics model.
6: *Predict* system trajectories from $p(X_0)$ to $p(X_T)$.
7: *Evaluate* policy:
$$J(\psi) = \sum_{t=0}^{T} \gamma^t \mathbb{E}_X[\text{cost}(X_t)|\psi].$$
8: *Optimise* policy:
$$\psi \leftarrow \arg\min_{\psi} J(\psi).$$
9: **until** policy parameters ψ converge

Gaussian process

$$J^{\pi}(\theta) = \sum_{t=0}^{T} \mathbb{E}_{x_t}[c(x_t)]$$

where $x_t = x_{t-1} + \Delta_t + \epsilon$

$p(\Delta_t)$ given input distribution $p(x_{t-1}, u_{t-1})$

Gradient descent using

$$\frac{\mathrm{d}\mathcal{E}_t}{\mathrm{d}\theta} = \frac{\mathrm{d}\mathcal{E}_t}{\mathrm{d}p(\mathbf{x}_t)} \frac{\mathrm{d}p(\mathbf{x}_t)}{\mathrm{d}\theta} := \frac{\partial \mathcal{E}_t}{\partial \mu_t} \frac{\mathrm{d}\mu_t}{\mathrm{d}\theta} + \frac{\partial \mathcal{E}_t}{\partial \Sigma_t} \frac{\mathrm{d}\Sigma_t}{\mathrm{d}\theta}$$

UNIVERSITY OF TORONTO | Engineering

87

UNIVERSITY OF TORONTO | Engineering

MARL – Part 1

- Game theory
- Stochastic game

88

Multi-agent Reinforcement Learning

- From RL to MARL
  - The only change = there are **multiple agents** making sequential decisions
  - The evolution of the environment and the rewards are determined by **joint actions**
  - Agents are interacting with not only the environment but also other agents
- MARL = Game + RL
  - In game, equilibrium is the solution as a result of mutual best responses
  - In RL, state is evolving as a result of actions and intrinsic uncertainty
- In MARL, agents will have to optimize **the long-term return** in anticipation of **future games**

UNIVERSITY OF TORONTO | Engineering

89

Game Theory

Prisoner's Dilemma

Two suspects are being separately questioned and invited to confess. Player 1 is told “If the other suspect does not confess, then you can cut a very good deal for yourself by confessing. But if the other does, then you would do well to confess, too; Otherwise, the court will be especially tough on you. So, you should confess no matter what the other does.”

		Player 2	
		c	d
Player 1	C	10 Years 10 Years	25 Years 1 Year
	D	1 Year 25 Year	3 Years 3 Years

		Player 2	
		c	d
Player 1	C	(1, 1)	(3, 0)
	D	(0, 3)	(2, 2)

UNIVERSITY OF TORONTO | Engineering

90

Solution Concepts

- **Nash Equilibrium**

  A configuration of strategies, one for each player, such that uni-lateral deviation would not improve the payoff.

Prisoner's Dilemma		Player 2	
		c	d
Player 1	C	(1, 1)	(3, 0)
	D	(0, 3)	(2, 2)

UNIVERSITY OF TORONTO | Engineering

91

Equilibrium in a Duopoly

Price Competition

- Two pizza stores, UT Pizza and York Pizza, split the market depending on their prices.
- The cost of making a pizza is $3 for each store.
- A recent market survey reveals the following weekly demands for the two stores:

$$Q_U = 12 - P_U + 0.5P_Y$$
$$Q_Y = 12 - P_Y + 0.5P_U$$

where $P_i \in [0, \infty]$ and demands are given in 1000s.

UNIVERSITY OF TORONTO | Engineering

92

- Duopoly (Continued) $Q_U = 12 - P_U + 0.5P_Y$ $Q_Y = 12 - P_Y + 0.5P_U$

Let Y_i be profit per week for player i. Then

$$Y_U = (P_U - 3)Q_U = (P_U - 3)(12 - P_U + 0.5P_Y)$$
$$Y_Y = (P_Y - 3)Q_Y = (P_Y - 3)(12 - P_Y + 0.5P_U)$$

To maximize Y_U, P_u should satisfy $\frac{dY_U}{dP_U} = 0$.

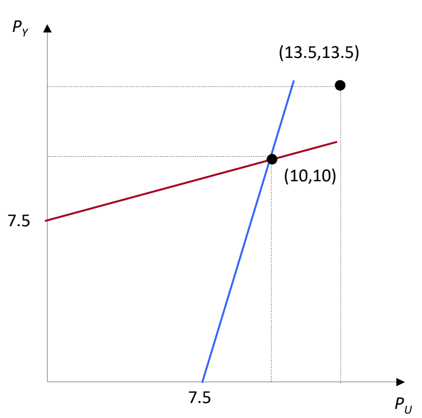
$$\frac{dY_U}{dP_U} = 12 - P_U + 0.5P_Y + 3 - P_U = 15 + 0.5P_Y - 2P_U = 0$$

Likewise,

$$\frac{dY_Y}{dP_Y} = 15 + 0.5P_U - 2P_Y = 0$$

UNIVERSITY OF TORONTO | Engineering

93



$$\frac{dY_U}{dP_U} = 15 + 0.5P_Y - 2P_U = 0 \Rightarrow P_U = \frac{1}{4}P_Y + 7.5$$: UT's best response

$$\frac{dY_Y}{dP_Y} = 15 + 0.5P_U - 2P_Y = 0 \Rightarrow P_Y = \frac{1}{4}P_U + 7.5$$: York's best response

Under Nash Equilibrium

$$\begin{matrix} P_U = 10 \\ P_Y = 10 \end{matrix} \Rightarrow \begin{matrix} Y_U = (P_U - 3)(12 - P_U + 0.5P_Y) = \$49{,}000 \\ Y_Y = (P_Y - 3)(12 - P_Y + 0.5P_U) = \$49{,}000 \end{matrix}$$

UNIVERSITY OF TORONTO | Engineering

94

Stochastic Games

- Sequential Decision Making

1. Markov decision processes
   - one decision maker
   - multiple states

2. Repeated games
   - multiple decision makers
   - one state (e.g., one normal form game)

3. **Stochastic games (Markov games)**
   - multiple decision makers
   - multiple states (e.g., multiple normal form games)

1+2=3

A Stochastic Game
- In each state, there is a normal form game
- After a round, the game randomly transitions into another state
- Transition probabilities depend on state and joint actions by all agents

UNIVERSITY OF TORONTO | Engineering

95

Formal Definition of Stochastic Games

SG is defined by a tuple $(n, S, A_1 ... A_n, T, R_1 ... R_n)$

n : the number of players,

S : the set of states,

A_i : the set of actions available to player i (and A is the joint action space $A_1 \times \cdots \times A_n$),

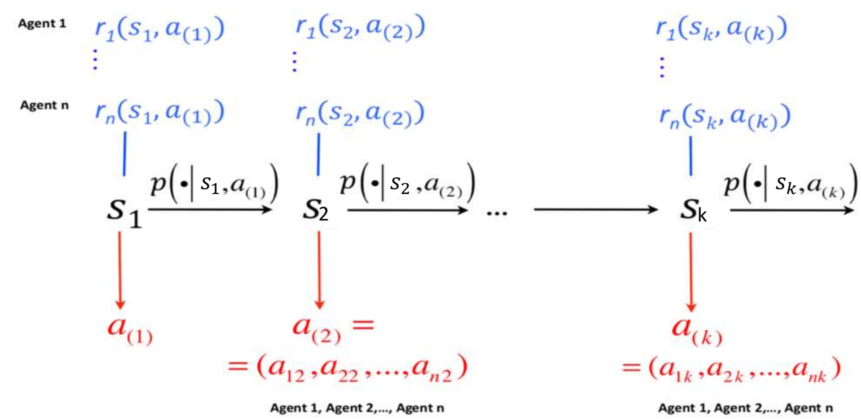
T : the transition function $S \times A \times S \to [0,1]$, and

R_i : the reward function of the i-th agent $S \times A \to R$.

UNIVERSITY OF TORONTO | Engineering

96

Formal Definition of Stochastic Games



- Example: Pollution Tax Game
  - Two firms contribute to the emission of certain pollutant

	Filtered	Polluted
Filtered	(4, 5)	(3, 8)
Polluted	(7, 4)	(6, 7)

- The government can detect only if the combined emission is high.
- Upon detecting high emission, tax (3) will be charged
- State space: S={no tax, tax=3}

$\Pr(0|0,F,F) = 1$ $\quad \Pr(0|0,F,P) = 0.5$
$\Pr(1|0,F,F) = 0$ $\quad \Pr(1|0,F,P) = 0.5$

$\Pr(0|0,P,F) = 0.5$ $\quad \Pr(0|0,P,P) = 0$
$\Pr(1|0,P,F) = 0.5$ $\quad \Pr(1|0,P,P) = 1$

	Filtered	Polluted
Filtered	(4, 5) / (1, 0)	(3, 8) / (.5, .5)
Polluted	(7, 4) / (.5, .5)	(6, 7) / (0, 1)

State = 0 (no tax)

$\xrightarrow{\Pr(1|0, a_1, a_2)}$

$\xleftarrow[\Pr(0|1, a_1, a_2)]{}$

	Filtered	Polluted
Filtered	(1, 2) / (1, 0)	(0, 5) / (.5, .5)
Polluted	(4, 1) / (.5, .5)	(3, 4) / (0, 1)

State = 1 (tax=3)