CS 540 Introduction to Artificial Intelligence
Reinforcement Learning II

University of Wisconsin-Madison
Spring 2026 Sections 1 & 2

Announcements

- **Homework**:
  - HW9 due **tomorrow April 22nd at 11:59 PM**
  - HW10 will also be released tomorrow

- Class roadmap:

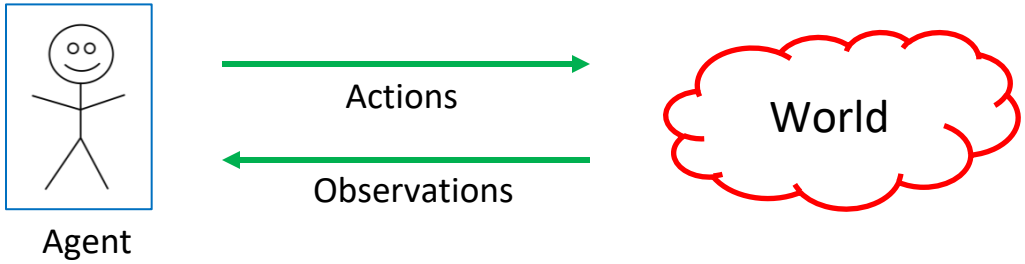
Reinforcement Learning II
Advanced Search
Ethics and Trust in AI

Outline

- Review of reinforcement learning setting.
  - MDPs, value functions
- Bellman equations and dynamic programming
- From dynamic programming to Q-learning
- Deep Q-Learning and Deep Q- Networks

Back to Our General Model

We have an **agent** **interacting** with the **world**



- Agent receives a reward based on state of the world
  - **Goal**: maximize reward / utility **($$$)**
  - Note: **data** consists of actions & observations
    - Compare to unsupervised learning and supervised learning

Markov Decision Process (MDP)

The formal mathematical model:

- **State set** S. Initial state s_0. **Action set** A
- **State transition model**: $P(s_{t+1}|s_t, a_t)$
  - Markov assumption: transition probability only depends on s_t and a_t, and not previous actions or states.
- **Reward function: *r*(s_t)**
- **Policy**: $\pi(s) : S \rightarrow A$, action to take at a particular state.

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \dots$$

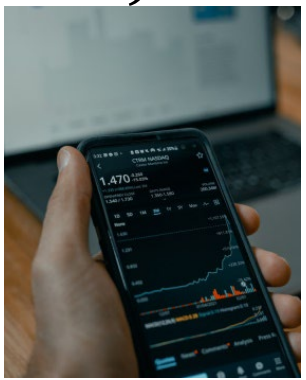
Defining the Optimal Policy

For policy π, **expected utility** over all possible state sequences from s_0 produced by following that policy:

Utility of sequence

$$V^{\pi}(s_0) = \sum_{\text{sequences starting from } s_0} P(\text{sequence})U(\text{sequence})$$

Probability of sequence when following π



Called the **value function** (for π, s_0)

Discounting Rewards

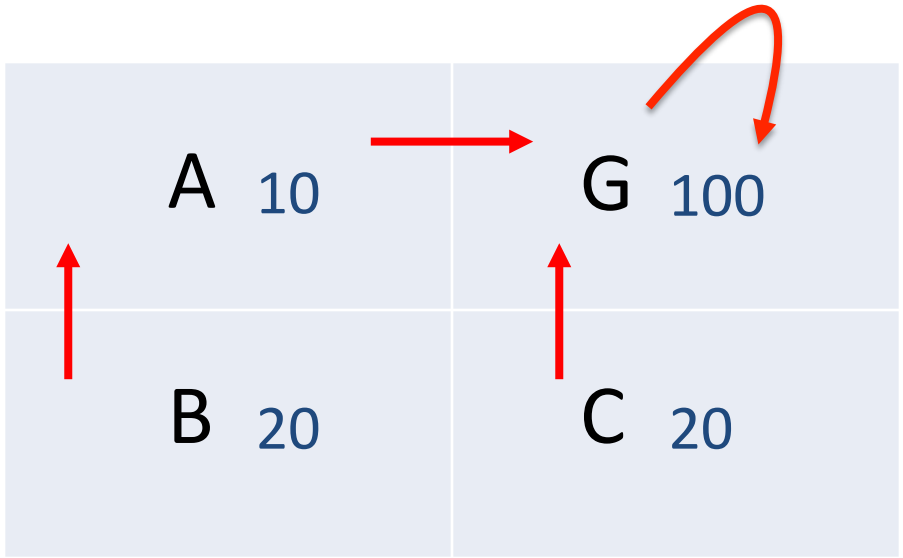
One issue: these are infinite series. **Convergence**?

- Solution

$$U(s_0, s_1 \ldots) = r(s_0) + \gamma r(s_1) + \gamma^2 r(s_2) + \ldots = \sum_{t \geq 0} \gamma^t r(s_t)$$

- Discount factor γ between 0 and 1
  - Set according to how important **present** is VS **future**
  - Note: has to be less than 1 for convergence

Example



Deterministic transitions; $\gamma = 0.8$; policy shown with red arrows.

Values and Policies

- Now that $V^{\pi}(s_0)$ is defined what a should we take?
  - First, set V*(s) to be expected utility for **optimal** policy from s
  - What's the expected utility of an action?
    - Specifically, action a in state s?

$$\sum_{s'} P(s'|s,a)V^*(s')$$

All the states we could go to

Transition probability

Expected rewards

Obtaining the Optimal Policy

Assume, we know the expected utility of an action.

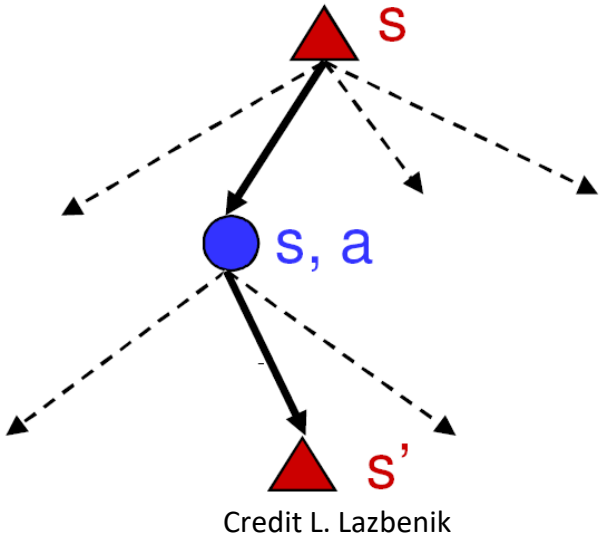
- So, to get the optimal policy, compute

$$\pi^*(s) = \text{argmax}_a \sum_{s'} P(s'|s,a)V^*(s')$$

All the states we could go to

Transition probability

Expected rewards



Credit L. Lazbenik

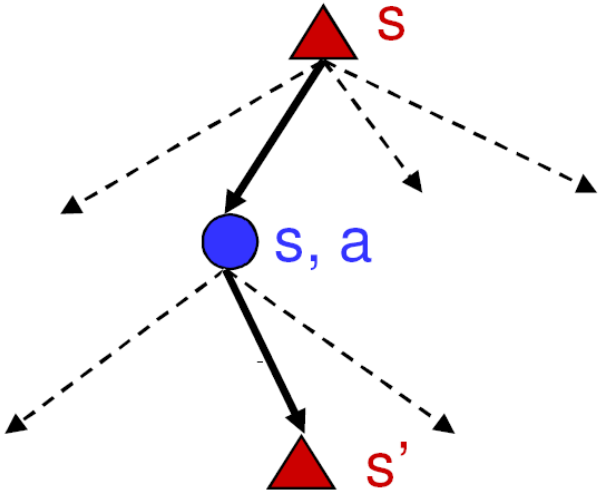
Bellman Equations

Let's walk over one step for the value function:

$$V^*(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s,a)V^*(s')$$

Current state reward

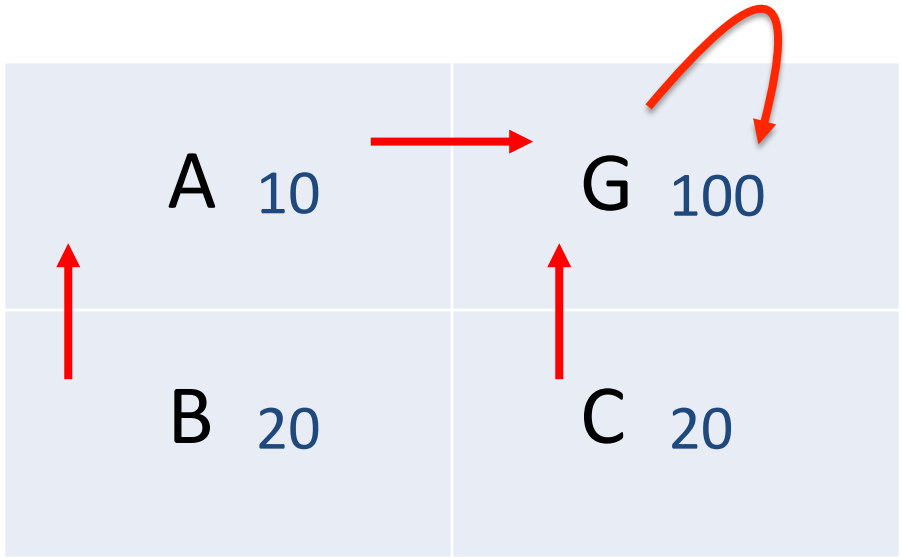
Discounted expected future **rewards**



Credit L. Lazbenik

Richard Bellman: Inventor of dynamic programming.

Example



Deterministic transitions; $\gamma = 0.8$; policy shown with red arrows.

Value Iteration

Q: how do we find $V^*(s)$?

- Why do we want it? Can use it to get the best policy
- Know: reward $r(s)$, transition probability $P(s'|s,a)$
- Also know $V^*(s)$ satisfies Bellman equation:

$$V^*(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s,a) V^*(s')$$

A: Use the property. Start with $V_0(s)=0$. Then, update

$$V_{i+1}(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s,a) V_i(s')$$

Break & Quiz

Q 1.1 Consider an MDP with 2 states {*A, B*} and 2 actions: **"stay"** at current state and "**move**" to other state. Let **r** be the reward function such that **r**(*A*) = 1, **r**(*B*) = 0. Let γ be the discounting factor. Let π: π(*A*) = π(*B*) = **move** (i.e., an "always move" policy). What is the value function $V^{\pi}(A)$?

- A. 0
- B. 1 / (1 -γ)
- C. 1 / (1 -γ^2)
- D. 1

Break & Quiz

Q 1.1 Consider an MDP with 2 states {*A, B*} and 2 actions: **"stay"** at current state and "**move**" to other state. Let **r** be the reward function such that **r**(*A*) = 1, **r**(*B*) = 0. Let γ be the discounting factor. Let π: π(*A*) = π(*B*) = **move** (i.e., an "always move" policy). What is the value function $V^{\pi}(A)$?

- A. 0
- B. $1/(1-\gamma)$
- **C. $1/(1-\gamma^2)$**
- D. 1

Break & Quiz

Q 1.1 Consider an MDP with 2 states {*A, B*} and 2 actions: **"stay"** at current state and "**move**" to other state. Let **r** be the reward function such that **r**(*A*) = 1, **r**(*B*) = 0. Let γ be the discounting factor. Let π: π(*A*) = π(*B*) = **move** (i.e., an "always move" policy). What is the value function $V^{\pi}(A)$?

- A. 0
- B. 1/(1-γ)
- **C. 1/(1-γ^2)** **(**States: A,B,A,B,… rewards 1,0, γ^2,0, γ^4,0, …)
- D. 1

Break & Quiz

Q 1.2 Supposed you have the following information about an environment:

1. The discount factor is 0.8
2. The reward in s_1 taking action α_1 is 3
3. The transition probabilities are: $P(s_2|s_1, \alpha_1) = 0.6$ and $P(s_3|s_1, \alpha_1) = 0.4$ Currently, $V(s_2) = 10$ and $V(s_3) = 6$

Remember the update for value iteration is $$V_{i+1}(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s, a) V_i(s')$$

Assuming the maximizing action taken from s_1 is α_1, what is the value for state s_1 (what is V (s_1))? Choose the closest option.

A. 8

B. 10

C. 12

D. 14

Break & Quiz

Q 1.2 Supposed you have the following information about an environment:

1. The discount factor is 0.8
2. The reward in s_1 taking action α_1 is 3
3. The transition probabilities are: $P(s_2|s_1, \alpha_1) = 0.6$ and $P(s_3|s_1, \alpha_1) = 0.4$ Currently, $V(s_2) = 10$ and $V(s_3) = 6$

Remember the update for value iteration is $$V_{i+1}(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s, a) V_i(s')$$

Assuming the maximizing action taken from s_1 is α_1, what is the value for state s_1 (what is V (s_1))? Choose the closest option.

A. 8

B. 10

C. 12

D. 14

Break & Quiz

Q 1.2 Supposed you have the following information about an environment:

1. The discount factor is 0.8
2. The reward in s_1 taking action α_1 is 3
3. The transition probabilities are: $P(s_2|s_1, \alpha_1) = 0.6$ and $P(s_3|s_1, \alpha_1) = 0.4$ Currently, $V(s_2) = 10$ and $V(s_3) = 6$

Remember the update for value iteration is $$V_{i+1}(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s, a) V_i(s')$$

Assuming the maximizing action taken from s_1 is α_1, what is the value for state s_1 (what is V (s_1))? Choose the closest option.

A. 8

B. 10

V (s1) = 3 + 0.8(0.6 × 10 + 0.4 × 6) = 9.72 = 10

C. 12

D. 14

Q-Learning

- Our **next** reinforcement learning algorithm.
- Does not require knowing *r* or *P*. Learn from data of the form:$\{(s_t, a_t, r_t, s_{t+1})\}$.
- Learns an action-value function *Q**(*s*,*a*) that tells us the expected value of taking *a* in state *s*.
  - Note: $V^*(s) = \underset{a}{max} Q^*(s, a)$.
- Optimal policy is formed as $\pi^*(s) = \arg\underset{a}{max} Q^*(s, a)$

The Q*(s,a) function

- Starting from state s, perform (perhaps suboptimal) action a. THEN follow the optimal policy

$$Q^*(s,a) = r(s) + \gamma \sum_{s'} P(s'|s,a) V^*(s')$$

- Equivalent to

$$Q^*(s,a) = r(s) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q^*(s',a')$$

Q-Learning Iteration

How do we get Q(*s*,*a*)?

- Iterative procedure

Current Q Value

Reward for taking that action at that state

Current Q Value

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_b Q(s_{t+1}, b) - Q(s_t, a_t) \right]$$

New Q Value for that state and action

Learning rate

Maximum expected future reward

$$Q(s_t, a_t) \leftarrow (1 - \alpha) Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_b Q(s_{t+1}, b) \right]$$

Idea: combine old value and new estimate of future value.

Note: We are using a policy to take actions; based on the estimated Q!

Q-Learning

Estimate Q*(*s*,*a*) from data $\{(s_t, a_t, r_t, s_{t+1})\}$:

1. Initialize Q(.,.) arbitrarily (eg all zeros)
    1. Except terminal states $Q(s_{terminal},.)=0$
2. Iterate over data until Q(.,.) converges:

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_b Q(s_{t+1}, b))$$

Example

A -10	G 10
S	B

discount factor $\gamma = 0.9$

learning rate $a = 0.1$

Example

A -10	G 10
S	B

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_b Q(s_{t+1}, b))$$

discount factor $\gamma = 0.9$

learning rate $a = 0.1$

	left	right	up	down
S			-1	
A				
B				
G				

$$Q(S, up) = (1-0.1) * 0 + 0.1(-10) = -1$$

Example

A -10	G 10
S	B

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_b Q(s_{t+1}, b))$$

discount factor $\gamma = 0.9$

learning rate $a = 0.1$

	left	right	up	down
S		0	-1	
A				
B				
G				

$$Q(S, right) = (1-0.1) * 0 + 0.1(0 + 0.9 * 0) = \ 0$$

Example

A^{-10}	G^{10}
S	B

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_b Q(s_{t+1}, b))$$

discount factor $\gamma = 0.9$

learning rate $a = 0.1$

	left	right	up	down
S		0	-1	
A				
B			1	
G				

$$Q(S, right) = (1-0.1) * 0 + 0.1(0 + 0.9 * 0) = \ 0$$

$$Q(B, up) = (1-0.1) * 0 + 0.1(10) = \ 1$$

Example

A -10	G 10
S	B

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_b Q(s_{t+1}, b))$$

discount factor $\gamma = 0.9$

learning rate $a = 0.1$

	left	right	up	down
S		0.09	-1	
A				
B			1.9	
G				

$$Q(S, right) = (1-0.1) * 0 + 0.1(0 + 0.9 * 1) = \ 0.09$$

$$Q(B, up) = (1-0.1) * 1 + 0.1(10) = \ 1.9$$

Exploration Vs. Exploitation

General question!

- **Exploration:** take an action with unknown consequences
  - **Pros**:
    - Get a more accurate model of the environment
    - Discover higher-reward states than the ones found so far
  - **Cons**:
    - When exploring, not maximizing your utility
    - Something bad might happen
- **Exploitation:** go with the best strategy found so far
  - **Pros**:
    - Maximize reward as reflected in the current utility estimates
    - Avoid bad stuff
  - **Cons**:
    - Might prevent you from discovering the true optimal strategy

Q-Learning: ε-Greedy Behavior Policy

Getting data with both **exploration and exploitation**

- With probability ε, take a random action; else the action with the highest (current) Q(*s*,*a*) value.

$$a = \begin{cases} \text{argmax}_{a \in A} Q(s, a) & \text{uniform}(0, 1) > \epsilon \\ \text{random } a \in A & \text{otherwise} \end{cases}$$

Q-learning Algorithm

Input: step size α, exploration probability ϵ

1. set $Q(s,a) = 0$ for all s, a.
2. For each episode:
3. Get initial state s.
4. While (s not a terminal state):
5. Perform $a = \epsilon$-greedy(Q, s), receive r, s'
6. $Q(s,a) = (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_b Q(s',b))$
7. $s \leftarrow s'$
8. End While
9. End For

Explore: take action to see what happens.

Update action-value based on result.

Q-Learning: SARSA

An alternative update rule:

- Just use the next action, no max over actions:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right]$$

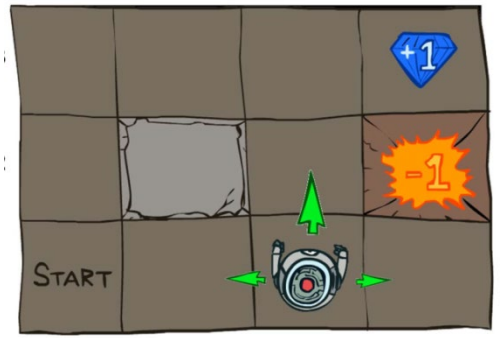
Learning rate

- Called state–action–reward–state–action (**SARSA**)
- Can use with epsilon-greedy policy

Q-Learning Details

Note: if we have a **terminal** state, the process ends

- An **episode**: a sequence of states ending at a terminal state
- Want to run on many episodes
- Slightly different Q-update for terminal states



Break & Quiz

Q 2.1 For Q learning to converge to the true Q function, we must

- A. Visit every state and try every action infinitely often.
- B. Perform at least 20,000 iterations.
- C. Re-start with different random initial table values.
- D. Prioritize exploitation over exploration.

Break & Quiz

Q 2.1 For Q learning to converge to the true Q function, we must

- **A. Visit every state and try every action infinitely often**
- B. Perform at least 20,000 iterations.
- C. Re-start with different random initial table values.
- D. Prioritize exploitation over exploration.

Break & Quiz

Q 2.1 For Q learning to converge to the true Q function, we must

- **A. Visit every state and try every action infinitely often.**
- B. Perform at least 20,000 iterations. (No: this is dependent on the particular problem, not a general constant).
- C. Re-start with different random initial table values. (No: this is not necessary in general).
- D. Prioritize exploitation over exploration. (No: insufficient exploration means potentially unupdated state action pairs).

Deep Q-Learning

Q-tables work great for small, discrete state spaces.

How do we get Q(*s*,*a*) with a large number of states?

- **Deep Q-learning** uses a **neural network** to approximate Q(s,a)
  - Let $Q_\theta : S \times A \to \mathbb{R}$ be the neural network with weights and biases denoted θ.
- Training is similar to supervised regression:
  - (s, a) as input and compute target $y = r(s) + \gamma \max_{a'} Q_\theta(s', a')$.
  - Note that output of the neural network is used in the target/label.
  - Loss function: $\mathcal{L}(\theta) = (y - Q_\theta(s, a))^2$

Problem with Deep Q-Learning

This doesn't work well! Training is **very unstable**

- Correlated samples

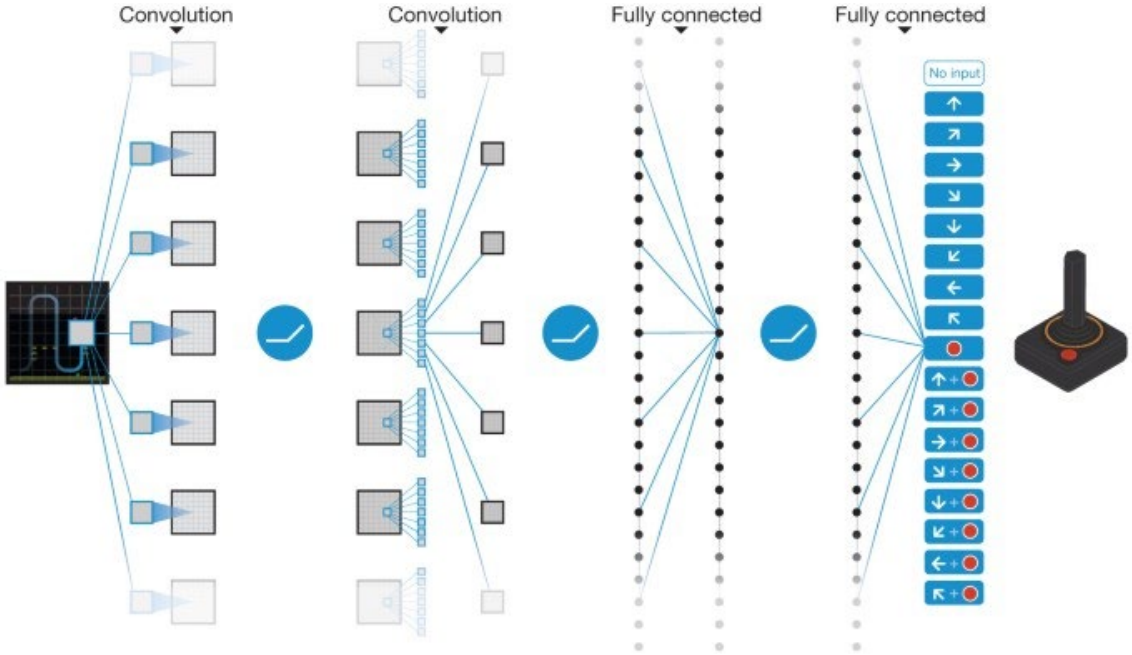
  — consecutive experiences are highly correlated

- Moving target

  — the target y changes every time we update θ

Deep Q-Networks (DQN)

Deep Q-Networks stabilized Deep Q-Learning.



Mnih et al, "Human-level control through deep reinforcement learning"

DQN Key Innovations

1. **Experience Replay**:
   - Store all transitions in a replay buffer D
   - At each training step, sample a random mini-batch from D
   - Use this mini-batch to compute the gradient update
2. Separate **target network (θ^-)** to compute targets:
   - **Online Network (θ)**
     - Updated every step via SGD
     - Used to select actions (ε-greedy)
   - **Target Network (θ^-)**
     - Frozen copy of θ
     - Updated: $\theta^- \leftarrow \theta$ every C steps

Summary of RL

- Reinforcement learning setup
- Mathematical formulation: MDP
- Value functions & the Bellman equation
- Value iteration
- Q-learning
- Deep Q-Learning and DQN

Suggested Readings

Textbook: Artificial Intelligence: A Modern Approach (4th edition). Stuart Russell and Peter Norvig. Pearson, 2020.

- Chapter 22