# REINFORCEMENT LEARNING

Lecture 5 : Deep Reinforcement Learning

Ibrahim Sammour

January | 2024

# SARSA

- Short for (state, action, reward, state, action)
- SARSA is an on-policy algorithm, meaning it learns from the policy it is currently following.
- Suitable for scenarios where exploration is crucial, such as in online learning or when the environment is unknown.
- Performs policy updates during the learning process.

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$$

# Q-Learning

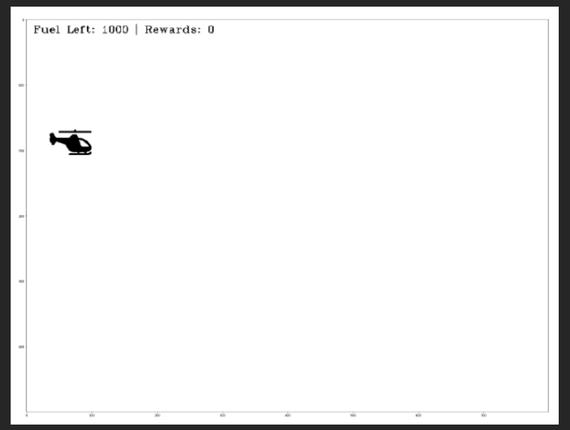
- Q-learning is an off-policy algorithm, meaning it learns the value of the optimal policy regardless of the policy being followed.
- Suitable for scenarios where it's essential to learn an optimal policy for later exploitation.

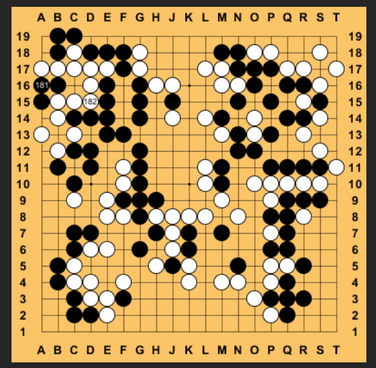
$$Q(s,a) = Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

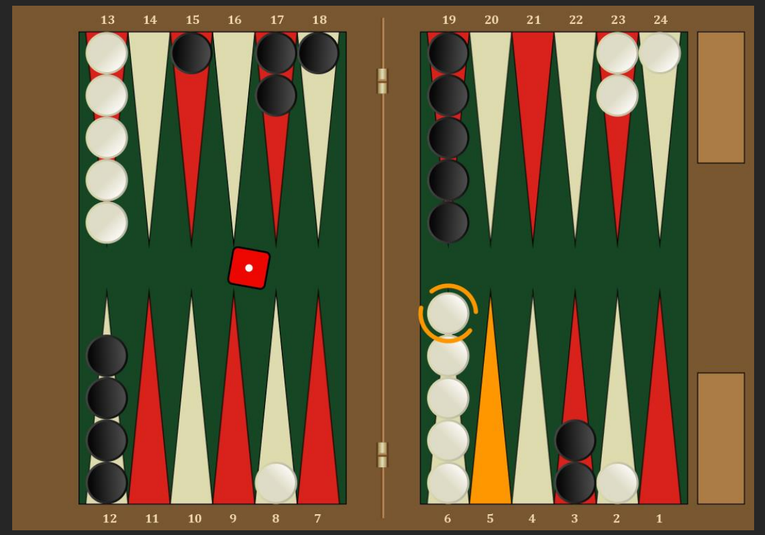
# Large-Scale Reinforcement Learning

Reinforcement learning can be used to solve large problems

- Backgammon: $10^{20}$ states
- Computer Go: $10^{170}$ states
- Helicopter: continuous state space



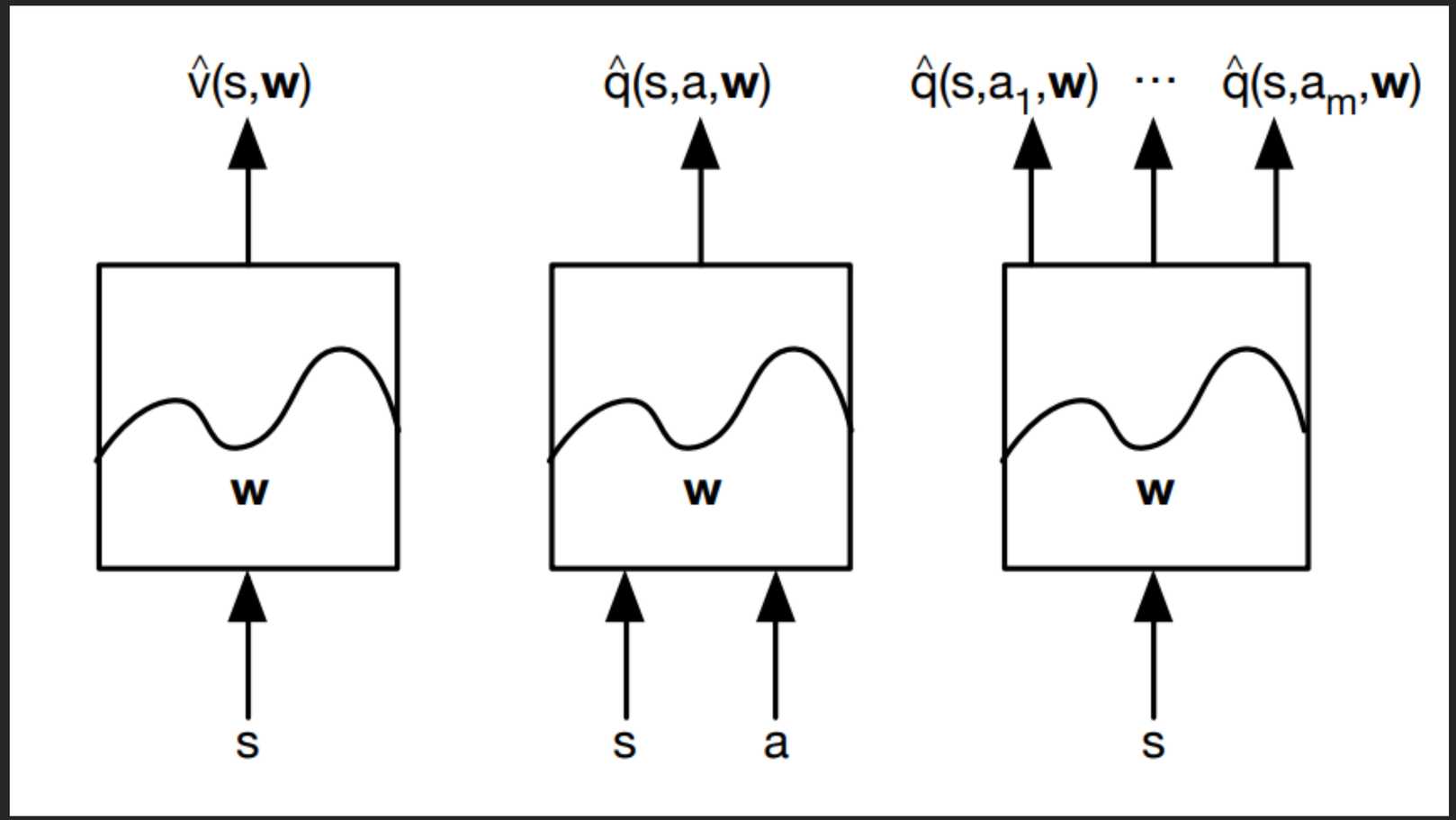
# Value Function Approximation

- So far we have represented value function by a lookup table
  - Every state s has an entry V(s)
  - Every state-action pair s, a has an entry Q(s, a)

- Problem with large spaces:
  - There are too many states and/or actions to store in memory
  - It is too slow to learn the value of each state individually

- Solution for large spaces:
  - Estimate value function with function approximation
    - $\hat{v}(s, w) \approx v_\pi(s)$
    - $\hat{q}(s, a, w) \approx q_\pi(s, a)$
  - Generalize from seen states to unseen states
  - Update parameter w using MC or TD learning

# Types of Value Function Approximation



# Types of Value Function Approximation

- There are many function approximators
  - Linear combinations of features
  - Neural network

# Gradient Descent

- Let J(w) be a differentiable function of parameter vector w
- The gradient of J(w) with respect to w is as follows:
- $\nabla_w J(w) = \begin{bmatrix} \frac{\partial \mathrm{J(w)}}{\partial w_1} \\ \vdots \\ \frac{\partial \mathrm{J(w)}}{\partial w_n} \end{bmatrix}$
- To minimize J(w)
  - We adjust w in direction of the gradient
  - $\Delta w = -\alpha \nabla_w J(w)$
  - where α is a step-size parameter

# Value Function Approximation by Stochastic Gradient Descent

- Goal: find parameter vector w to minimize mean-squared error between approximate value function $\hat{v}(S, w)$ and true value function $v_\pi(S)$

- $J(w) = E_\pi\left[\left(v_\pi(S) - \hat{v}(S, w)\right)^2\right]$

- Gradient decent finds a minimum
- $\Delta w = -\alpha \nabla_w J(w)$
- $\quad = 2\alpha E_\pi\left[\left(v_\pi(S) - \hat{v}(S, w)\right)\nabla_w \hat{v}(S, w)\right]$
- Stochastic gradient descent samples the gradient
  - $\Delta w = \alpha\left(v_\pi(S) - \hat{v}(S, w)\right)\nabla_w \hat{v}(S, w)$
- Expected update is equal to full gradient update

# Feature Vectors

- Represent state by a feature vector
  - $x(\mathrm{S}) = \begin{bmatrix} x_1(S) \\ \vdots \\ x_n(S) \end{bmatrix}$
- For example
  - Distance of robot from landmarks

# Linear Value Function Approximation

- Represent value function by a linear combination of features
  - $\hat{v}(S, w) = x(S)^T w$
- Objective function is quadratic in parameters w
  - J(w) = $E_\pi[(v_\pi(S) - x(S)^T w)^2]$
- Stochastic gradient descent converges on global optimum
- Update rule is as follows:
  - $\nabla_w \hat{v}(s, w) = x(S)$
  - $\Delta w = -\alpha\big(v_\pi(S) - \hat{v}(S\, w)\big)x(S)$
  - Update = learning rate * prediction error * feature value

# Table Lookup Features

- Table lookup is a special case of linear value function approximation
- Using table lookup features
  - $x^{table}(S) = \begin{bmatrix} l(S = s_1) \\ \vdots \\ l(S = s_n) \end{bmatrix}$
- Parameter vector w gives value of each individual state
  - $\hat{v}(S, w) = \begin{bmatrix} l(S = s_1) \\ \vdots \\ l(S = s_n) \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}$

# Incremental Prediction Algorithms

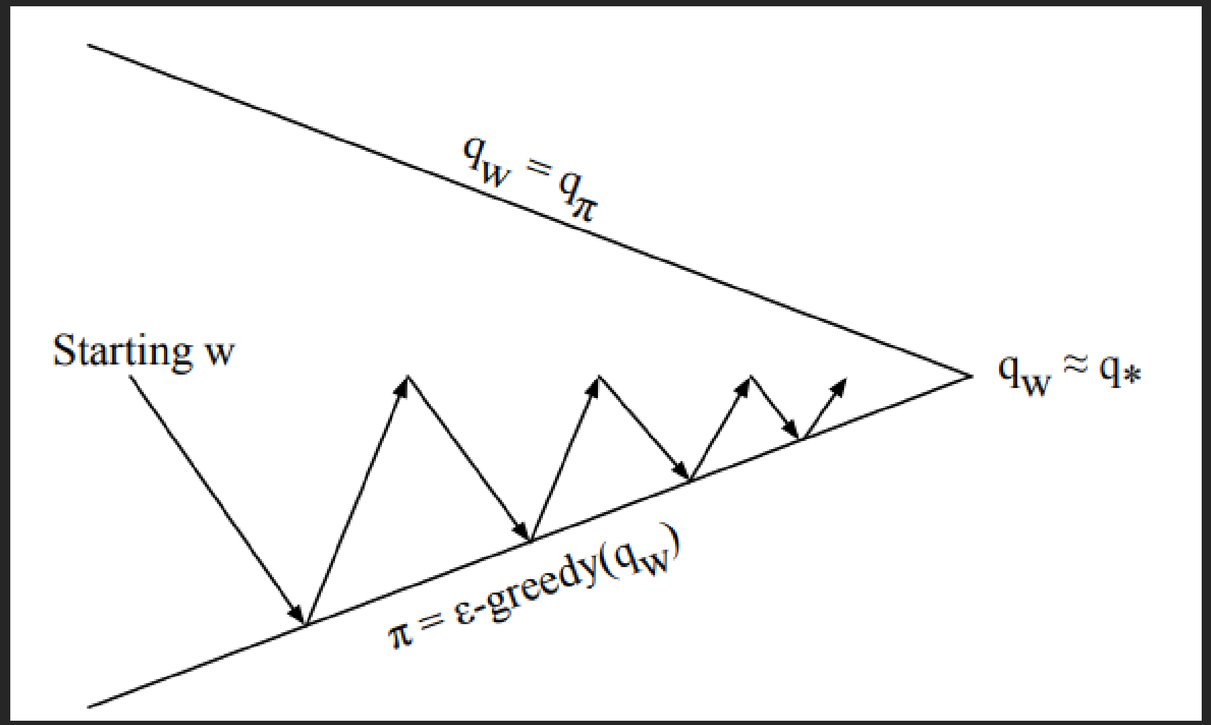
- Have assumed true value function $v_\pi(s)$ given by supervisor
- But in RL there is no supervisor, only rewards
- In practice, we substitute a target for $v_\pi(s)$
  - For MC, the target is the return $\bar{\mathbf{r}}_t$
    - $\Delta w = \alpha\left(\bar{\mathbf{r}}_t - \hat{v}(s,w)\right)\nabla_w \hat{v}(s,w)$
  - For TD(0), the target is the TD target $r_t + \gamma\hat{v}(s_{t+1},w)$
    - $\Delta w = \alpha\left(r_t + \gamma\hat{v}(s_{t+1},w) - \hat{v}(s,w)\right)\nabla_w \hat{v}(s,w)$

# Monte-Carlo with Value Function Approximation

- Return $G_t$ is an unbiased, noisy sample of true value $v_\pi(S_t)$
- Can therefore apply supervised learning to “training data”
  - $\langle S_1, G_1 \rangle, \langle S_2, G_2 \rangle, \cdots, \langle S_T, G_T \rangle$
- For example, using linear Monte-Carlo policy evaluation
  - $\Delta w = \alpha\big(G_t - \hat{v}(S, w)\big)\nabla_w \hat{v}(S, w)$
    $= \alpha\big(G_t - \hat{v}(S, w)\big)x(S_t)$
- When using non-linear value function approximation, Monte-Carlo evaluation converges to a local optimum

# TD Learning with Value Function Approximation

- Policy Evaluation: Approximate policy evaluation, $\hat{q}(\cdot,\cdot,w) \approx q_\pi$
- Policy Improvement: $\varepsilon$-greedy policy improvement



# Action-Value Function Approximation

- Approximate the action-value function
  - $\hat{q}(S, A, w) \approx q_\pi(S, A)$
- Minimize mean-squared error between approximate action-value function $\hat{q}(S, A, w)$ and true action-value function $q_\pi(S, A)$
  - $J(W) = E_\pi\left[\left(q_\pi(S, A) - \hat{q}(S, A, q)\right)^2\right]$
- Use stochastic gradient descent to find a local minimum
  - $-\nabla_w J(w) = \left(q_\pi(S, A) - \hat{q}(S, A, q)\right)\nabla_w \hat{q}(S, A, q)$
  - $\Delta w = 2\alpha\left(q_\pi(S, A) - \hat{q}(S, A, q)\right)\nabla_w \hat{q}(S, A, q)$

# Linear Action-Value Function Approximation

- Represent state and action by a feature vector
  - $x(\mathrm{S},\mathrm{A}) = \begin{bmatrix} x_1(S,A) \\ \vdots \\ x_n(S,A) \end{bmatrix}$
- Represent action-value function by linear combination of features
  - $\hat{q}(S,A,w) = x(S,A)^T w$
- Stochastic gradient descent update
  - $\nabla_w \hat{q}(S,A,w) = x(S,A)$
  - $\Delta w = \alpha\big(q_\pi(S,A) - \hat{q}(S,A,q)\big)x(S,A)$