Reinforcement Learning

From Fundamentals to Advanced Policy Methods

Author: Eng. Carlos Andrés Sierra, M.Sc.
cavirguezs@udistrital.edu.co

Full-time Adjunct Professor
Computer Engineering Program
School of Engineering
Universidad Distrital Francisco José de Caldas

2026-I

Outline

Outline

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

What is Reinforcement Learning?

Reinforcement Learning

- **Reinforcement Learning** is a computational approach to learning whereby an agent tries to maximize the notion of cumulative reward.
- The agent learns to achieve a goal in an uncertain, potentially complex environment.

- **Agent**: The learner or decision maker
- **Environment**: Everything the agent interacts with
- **Actions**: Set of all possible moves the agent can make
- **Rewards**: Feedback from environment for agent

What is Reinforcement Learning?

Reinforcement Learning

- **Reinforcement Learning** is a computational approach to learning whereby an agent tries to maximize the notion of cumulative reward.
- The agent learns to achieve a goal in an uncertain, potentially complex environment.

- **Agent**: The learner or decision maker
- **Environment**: Everything the agent interacts with
- **Actions**: Set of all possible moves the agent can make
- **Rewards**: Feedback from environment to agent

What is Reinforcement Learning?

Reinforcement Learning

- **Reinforcement Learning** is a computational approach to learning whereby an agent tries to maximize the notion of cumulative reward.
- The agent learns to achieve a goal in an uncertain, potentially complex environment.

- **Agent**: The learner or decision maker
- **Environment**: Everything the agent interacts with
- **Actions**: Set of all possible moves the agent can make
- **Rewards**: Feedback from environment to agent

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

What is Reinforcement Learning?

Reinforcement Learning

- **Reinforcement Learning** is a computational approach to learning whereby an agent tries to maximize the notion of cumulative reward.
- The agent learns to achieve a goal in an uncertain, potentially complex environment.

- **Agent**: The learner or decision maker
- **Environment**: Everything the agent interacts with
- **Actions**: Set of all possible moves the agent can make
- **Rewards**: Feedback from environment to agent

What is Reinforcement Learning?

Reinforcement Learning

- **Reinforcement Learning** is a computational approach to learning whereby an agent tries to maximize the notion of cumulative reward.
- The agent learns to achieve a goal in an uncertain, potentially complex environment.

- **Agent**: The learner or decision maker
- **Environment**: Everything the agent interacts with
- **Actions**: Set of all possible moves the agent can make
- **Rewards**: Feedback from environment to agent

RL vs. Other Learning Paradigms

Supervised Learning:

- Uses labeled data
- Direct feedback
- Static datasets

Unsupervised Learning:

- No labels
- Pattern discovery
- Structure finding

Reinforcement Learning:

- Trial-and-error learning
- Delayed rewards
- Sequential decision making
- Exploration vs. exploitation
- Dynamic environments

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

RL vs. Other Learning Paradigms

Supervised Learning:

- Uses labeled data
- Direct feedback
- Static datasets

Unsupervised Learning:

- No labels
- Pattern discovery
- Structure finding

Reinforcement Learning:

- Trial-and-error learning
- Delayed rewards
- Sequential decision making
- Exploration vs. exploitation
- Dynamic environments

Markov Decision Processes (MDPs)

An MDP is defined by the tuple (S, A, P, R, γ):

- **S**: Set of states
- **A**: Set of actions
- **P**: Transition probabilities $P(s'|s, a)$
- **R**: Reward function $R(s, a, s')$
- γ: Discount factor $[0, 1]$

Markov Property

The future is independent of the past given the present state.

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \dots) = P(s_{t+1}|s_t, a_t)$$

Markov Decision Processes (MDPs)

An MDP is defined by the tuple (S, A, P, R, γ):

- **S**: Set of states
- **A**: Set of actions
- **P**: Transition probabilities $P(s'|s, a)$
- **R**: Reward function $R(s, a, s')$
- γ: Discount factor $[0, 1]$

Markov Property

The future is independent of the past given the present state.

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \dots) = P(s_{t+1}|s_t, a_t)$$

Agent-Environment Interaction

The RL Loop:

1. Agent observes current state s_t
2. Agent selects action a_t based on policy $\pi(a|s)$
3. Environment transitions to new state s_{t+1}
4. Environment gives reward r_{t+1}
5. Agent updates its knowledge
6. Repeat...

Value Functions and Policies

Policy

A **policy** $\pi(a|s)$ is a mapping from states to probability distributions over actions.

Value Functions

- **State Value**: $V^{\pi}(s) = E[G_t|s_t = s]$
- **Action Value (Q-function)**: $Q^{\pi}(s, a) = E[G_t|s_t = s, a_t = a]$

Where $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ is the return.

Value Functions and Policies

Policy

A **policy** $\pi(a|s)$ is a mapping from states to probability distributions over actions.

Value Functions

- **State Value**: $V^\pi(s) = E[G_t|s_t = s]$
- **Action Value (Q-function)**: $Q^\pi(s,a) = E[G_t|s_t = s, a_t = a]$

Where $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ is the return.

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Practical Example: `Grid World Navigation`

- **Environment**: 4×4 grid with walls and goal
- **States**: Agent's position (x, y)
- **Actions**: Up, Down, Left, Right
- **Rewards**: -1 per step, $+100$ at goal, -100 in pit
- **Goal**: Find shortest path to reward

Practical Example: `Grid World Navigation`

- **Environment**: 4×4 grid with walls and goal
- **States**: Agent's position (x, y)
- **Actions**: Up, Down, Left, Right
- **Rewards**: -1 per step, $+100$ at goal, -100 in pit
- **Goal**: Find shortest path to reward

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Practical Example: Grid World Navigation

- **Environment**: 4×4 grid with walls and goal
- **States**: Agent's position (x, y)
- **Actions**: Up, Down, Left, Right
- **Rewards**: -1 per step, $+100$ at goal, -100 in pit
- **Goal**: Find shortest path to reward

Q-Learning Algorithm I

Definition

Q-Learning is a **model-free** reinforcement learning algorithm that learns the optimal action-value function $Q^*(s, a)$ directly from interactions with the environment.

Q-Learning Algorithm II

Q-Learning Update Rule

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

- α: Learning rate $[0, 1]$
- γ: Discount factor $[0, 1]$
- **Off-policy**: Learns optimal policy regardless of behavior policy
- **Model-free**: No need to know transition probabilities

Q-Learning Algorithm II

Q-Learning Update Rule

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

- α: Learning rate $[0, 1]$
- γ: Discount factor $[0, 1]$
- **Off-policy**: Learns optimal policy regardless of behavior policy
- **Model-free**: No need to know transition probabilities

Bellman Equation for Q-Values

Bellman Optimality Equation

$$Q^*(s,a) = E[r + \gamma \max_{a'} Q^*(s',a') | s,a]$$

- Q-Learning approximates this equation through temporal difference learning
- The TD error: $\delta = r + \gamma \max_a Q(s',a) - Q(s,a)$
- Converges to optimal Q-function under certain conditions

Bellman Equation for Q-Values

Bellman Optimality Equation

$$Q^*(s,a) = E[r + \gamma \max_{a'} Q^*(s',a')|s,a]$$

- Q-Learning approximates this equation through temporal difference learning
- The TD error: $\delta = r + \gamma \max_a Q(s',a) - Q(s,a)$
- Converges to optimal Q-function under certain conditions

Bellman Equation for Q-Values

Bellman Optimality Equation

$$Q^*(s,a) = E[r + \gamma \max_{a'} Q^*(s',a') | s,a]$$

- Q-Learning approximates this equation through temporal difference learning
- The TD error: $\delta = r + \gamma \max_a Q(s',a) - Q(s,a)$
- Converges to optimal Q-function under certain conditions

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Bellman Equation for Q-Values

Bellman Optimality Equation

$$Q^*(s,a) = E[r + \gamma \max_{a'} Q^*(s',a')|s,a]$$

- Q-Learning approximates this equation through temporal difference learning
- The TD error: $\delta = r + \gamma \max_a Q(s',a) - Q(s,a)$
- Converges to optimal Q-function under certain conditions

Exploration vs. Exploitation

The Exploration-Exploitation Dilemma

Should the agent choose the best known action (exploit) or try something new (explore)?

ϵ-Greedy Strategy

- With probability ϵ: choose random action (explore)
- With probability 1-ϵ: choose best known action (exploit)
- Often use decaying ϵ over time

Exploration vs. Exploitation

The Exploration-Exploitation Dilemma

Should the agent choose the best known action (exploit) or try something new (explore)?

ϵ-Greedy Strategy

- With probability ϵ: choose random action (explore)
- With probability 1-ϵ: choose best known action (exploit)
- Often use decaying ϵ over time

Practical Example: `Q-Learning in Grid World`

- Initialize $Q(s, a)$ arbitrarily (e.g., all zeros)
- For each episode:
  - Initialize state s
  - Repeat until episode ends:
    - Choose action a using ϵ-greedy policy
    - Take action a, observe reward r and new state s'
    - Update Q-value: $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_a Q(s', a) - Q(s, a)]$
    - Update state: $s \leftarrow s'$

Practical Example: `Q-Learning in Grid World`

- Initialize $Q(s, a)$ arbitrarily (e.g., all zeros)
- For each episode:
  - Initialize state s
  - Repeat until episode ends:
    - Choose action a using ϵ-greedy policy
    - Take action a, observe reward r and new state s'
    - Update Q-value: $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_a Q(s', a) - Q(s, a)]$
    - Update state: $s \leftarrow s'$

Outline

SARSA Algorithm

SARSA Update Rule

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

- **On-policy**: Updates based on the actual policy being followed
- **SARSA**: State-Action-Reward-State-Action sequence
- More conservative than Q-Learning
- Better for safety-critical applications

SARSA Algorithm

SARSA Update Rule

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

- **On-policy**: Updates based on the actual policy being followed
- **SARSA**: State-Action-Reward-State-Action sequence
- More conservative than Q-Learning
- Better for safety-critical applications

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

On-Policy vs. Off-Policy Learning

On-Policy (SARSA):

- Learns about policy currently following
- More conservative
- Accounts for exploration in updates
- Safer in risky environments

Off-Policy (Q-Learning):

- Learns about optimal policy
- More aggressive
- Ignores exploration in updates
- Faster convergence

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

SARSA Algorithm Mechanics

- Initialize $Q(s, a)$ arbitrarily (e.g., all zeros)
- For each episode:
  - Initialize state s
  - Choose action a using ϵ-greedy policy
  - Repeat until episode ends:
    - Take action a, observe reward r and new state s'
    - Choose next action a' using ϵ-greedy policy
    - Update Q-value: $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
    - Update state and action: $s \leftarrow s'$, $a \leftarrow a'$

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Practical Example: `Cliff Walking Problem`

- **Environment**: Agent walks along a cliff edge
- **Cliff**: Large negative reward (-100)
- **Goal**: Reach the end safely
- **SARSA**: Takes safer path (`away from cliff`)
- **Q-Learning**: Takes riskier optimal path (`near cliff`)

Practical Example: `Cliff Walking Problem`

- **Environment**: Agent walks along a cliff edge
- **Cliff**: Large negative reward (-100)
- **Goal**: Reach the end safely
- **SARSA**: Takes safer path (`away from cliff`)
- **Q-Learning**: Takes riskier optimal path (`near cliff`)

Practical Example: `Cliff Walking Problem`

- **Environment**: Agent walks along a cliff edge
- **Cliff**: Large negative reward (-100)
- **Goal**: Reach the end safely
- **SARSA**: Takes safer path (`away from cliff`)
- **Q-Learning**: Takes riskier optimal path (`near cliff`)

Function Approximation in RL

The Problem

- Large or continuous state spaces
- Cannot store Q-values for every state-action pair
- Need to generalize from seen to unseen states

Solution: Function Approximation

Use neural networks to approximate: $Q(s, a) \approx Q(s, a; \theta)$

Function Approximation in RL

The Problem

- Large or continuous state spaces
- Cannot store Q-values for every state-action pair
- Need to generalize from seen to unseen states

Solution: Function Approximation

Use neural networks to approximate: $Q(s,a) \approx Q(s,a;\theta)$

Deep Q Network (DQN) Architecture

Key Components

- **Deep Neural Network**: Approximates Q-function
- **Experience Replay**: Stores and samples past experiences
- **Target Network**: Stable Q-targets for training
- **Fixed Q-Targets**: Reduces correlation in updates

Loss Function

$$L(\theta) = E[(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2]$$

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Deep Q Network (DQN) Architecture

Key Components

- **Deep Neural Network**: Approximates Q-function
- **Experience Replay**: Stores and samples past experiences
- **Target Network**: Stable Q-targets for training
- **Fixed Q-Targets**: Reduces correlation in updates

Loss Function

$$L(\theta) = E[(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2]$$

Experience Replay Mechanism

- **Store**: Experiences (s, a, r, s') in replay buffer
- **Sample**: Random minibatch for training
- **Benefits**:
  - Breaks temporal correlations
  - Improves data efficiency
  - Enables batch learning

Experience Replay Mechanism

- **Store**: Experiences (s, a, r, s') in replay buffer
- **Sample**: Random minibatch for training
- **Benefits**:
  - Breaks temporal correlations
  - Improves data efficiency
  - Enables batch learning

Experience Replay Mechanism

- **Store**: Experiences (s, a, r, s') in replay buffer
- **Sample**: Random minibatch for training
- **Benefits**:
  - Breaks temporal correlations
  - Improves data efficiency
  - Enables batch learning

DQN for `Atari Breakout`: Practical Example

Atari Breakout Implementation

- **Input**: Raw pixels $(210 \times 160 \times 3) \rightarrow$ preprocessed $(84 \times 84 \times 4)$
- **Network**: `CNN` with 3 conv layers $+$ 2 dense layers
- **Actions**: 4 discrete actions (`NOOP`, `FIRE`, `RIGHT`, `LEFT`)

- **Preprocessing**: Grayscale, resize, frame stacking (4 frames)
- **Reward**: $+1$ for each brick destroyed, $+5$ for clearing level

DQN for `Atari Breakout`: Practical Example

Atari Breakout Implementation

- **Input**: Raw pixels $(210 \times 160 \times 3) \rightarrow$ preprocessed $(84 \times 84 \times 4)$
- **Network**: `CNN` with 3 conv layers $+$ 2 dense layers
- **Actions**: 4 discrete actions (`NOOP`, `FIRE`, `RIGHT`, `LEFT`)

- **Preprocessing**: Grayscale, resize, frame stacking (4 frames)
- **Reward**: $+1$ for each brick destroyed, $+5$ for clearing level
- **Strategy**: Agent learns to tunnel through bricks for maximum score
- **Performance**: Achieves superhuman performance (400 points)

DQN for `Atari Breakout`: Practical Example

Atari Breakout Implementation

- **Input**: Raw pixels $(210 \times 160 \times 3) \rightarrow$ preprocessed $(84 \times 84 \times 4)$
- **Network**: `CNN` with 3 conv layers $+$ 2 dense layers
- **Actions**: 4 discrete actions (`NOOP`, `FIRE`, `RIGHT`, `LEFT`)

- **Preprocessing**: Grayscale, resize, frame stacking (4 frames)
- **Reward**: $+1$ for each brick destroyed, $+5$ for clearing level
- **Strategy**: Agent learns to tunnel through bricks for maximum score
- **Performance**: Achieves superhuman performance (400 points)

DQN for `Atari Breakout`: Practical Example

Atari Breakout Implementation

- **Input**: Raw pixels $(210 \times 160 \times 3) \rightarrow$ preprocessed $(84 \times 84 \times 4)$
- **Network**: `CNN` with 3 conv layers $+$ 2 dense layers
- **Actions**: 4 discrete actions (`NOOP`, `FIRE`, `RIGHT`, `LEFT`)

- **Preprocessing**: Grayscale, resize, frame stacking (4 frames)
- **Reward**: $+1$ for each brick destroyed, $+5$ for clearing level
- **Strategy**: Agent learns to tunnel through bricks for maximum score
- **Performance**: Achieves superhuman performance (400 points)

DQN for `Atari Breakout`: Practical Example

Atari Breakout Implementation

- **Input**: Raw pixels $(210 \times 160 \times 3) \rightarrow$ preprocessed $(84 \times 84 \times 4)$
- **Network**: `CNN` with 3 conv layers $+ 2$ dense layers
- **Actions**: 4 discrete actions (`NOOP`, `FIRE`, `RIGHT`, `LEFT`)

- **Preprocessing**: Grayscale, resize, frame stacking (4 frames)
- **Reward**: $+1$ for each brick destroyed, $+5$ for clearing level
- **Strategy**: Agent learns to tunnel through bricks for maximum score
- **Performance**: Achieves superhuman performance (400 points)

Outline

Policy-Based vs. Value-Based Methods

Value-Based:

- Learn value functions
- Derive policy from values
- Discrete action spaces
- Q-Learning, SARSA

Policy-Based:

- Learn policy directly
- Parameterized policies
- Continuous actions
- Stochastic policies

Policy-Based vs. Value-Based Methods

Value-Based:

- Learn value functions
- Derive policy from values
- Discrete action spaces
- Q-Learning, SARSA

Policy-Based:

- Learn policy directly
- Parameterized policies
- Continuous actions
- Stochastic policies

Policy Gradient Theorem

Gradient of Expected Return

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a)]$$

- Î¸: Policy parameters
- **Intuition**: Move parameters in direction that increases probability of good actions
- **Key insight**: We can estimate gradients through sampling

Policy Gradient Theorem

Gradient of Expected Return

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a)]$$

- **Î¸**: Policy parameters
- **Intuition**: Move parameters in direction that increases probability of good actions
- **Key insight**: We can estimate gradients through sampling

REINFORCE Algorithm

REINFORCE Update

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a_t|s_t) G_t$$

- Uses Monte Carlo return G_t
- High variance but unbiased
- Often use baseline to reduce variance
- $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a_t|s_t)(G_t - b(s_t))$

REINFORCE Algorithm

REINFORCE Update

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a_t|s_t) G_t$$

- Uses Monte Carlo return G_t
- High variance but unbiased
- Often use baseline to reduce variance
- $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a_t|s_t)(G_t - b(s_t))$

Actor-Critic Architecture

Two Components

- **Actor**: Policy $\pi_\theta(a|s)$ — chooses actions
- **Critic**: Value function $V_\phi(s)$ — evaluates actions

Benefits

- Lower variance than REINFORCE
- Online learning (no need to wait for episode end)
- Uses TD error: $\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Actor-Critic Architecture

Two Components

- **Actor**: Policy $\pi_\theta(a|s)$ — chooses actions
- **Critic**: Value function $V_\phi(s)$ — evaluates actions

Benefits

- Lower variance than REINFORCE
- Online learning (no need to wait for episode end)
- Uses TD error: $\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$

Proximal Policy Optimization (PPO) Motivation

Problems with Basic Policy Gradients

- Large policy updates can be destructive
- Hard to choose appropriate step size
- Poor sample efficiency

PPO Solution

Constrain policy updates to stay in a trust region around current policy

Proximal Policy Optimization (PPO) Motivation

Problems with Basic Policy Gradients

- Large policy updates can be destructive
- Hard to choose appropriate step size
- Poor sample efficiency

PPO Solution

Constrain policy updates to stay in a trust region around current policy

PPO Clipped Objective

Clipped Surrogate Objective

$$L^{CLIP}(\theta) = E[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$$

- $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ (probability ratio)
- $\hat{A}_t$ is advantage estimate
- Clips ratio to $[1-\epsilon, 1+\epsilon]$
- Prevents destructively large policy updates

Practical Example: `PPO on LunarLander-v2`

LunarLander-v2 Environment

- **State**: Position, velocity, angle, angular velocity, leg contact (8D continuous)
- **Actions**: 4 discrete actions (`NOOP`, `LEFT`, `MAIN`, `RIGHT`)
- **Goal**: Land safely between flags ($+100$ to $+140$ points)

- **Reward Shaping**: Distance, speed, angle penalties; fuel costs
- **PPO Advantages**: Stable learning with sparse rewards
- **Policy Network**: MLP ($8 \to 128 \to 128 \to 4$)
- **Training**: Converges in $\sim 1000-3000$ episodes
- **Performance**: Solves environment (average reward > 200)

Practical Example: PPO on LunarLander-v2

LunarLander-v2 Environment

- **State**: Position, velocity, angle, angular velocity, leg contact ($8D$ continuous)
- **Actions**: 4 discrete actions (`NOOP`, `LEFT`, `MAIN`, `RIGHT`)
- **Goal**: Land safely between flags ($+100$ to $+140$ points)

- **Reward Shaping**: Distance, speed, angle penalties; fuel costs
- **PPO Advantages**: Stable learning with sparse rewards
- **Policy Network**: MLP ($8 \rightarrow 128 \rightarrow 128 \rightarrow 4$)
- **Training**: Converges in $1000 - 3000$ episodes
- **Performance**: Solves environment (average reward > 200)

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Practical Example: `PPO on LunarLander-v2`

LunarLander-v2 Environment

- **State**: Position, velocity, angle, angular velocity, leg contact (8D continuous)
- **Actions**: 4 discrete actions (`NOOP`, `LEFT`, `MAIN`, `RIGHT`)
- **Goal**: Land safely between flags ($+100$ to $+140$ points)

- **Reward Shaping**: Distance, speed, angle penalties; fuel costs
- **PPO Advantages**: Stable learning with sparse rewards
- **Policy Network**: MLP ($8 \to 128 \to 128 \to 4$)
- **Training**: Converges in $1000-3000$ episodes
- **Performance**: Solves environment (average reward > 200)

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Practical Example: `PPO on LunarLander-v2`

LunarLander-v2 Environment

- **State**: Position, velocity, angle, angular velocity, leg contact (8D continuous)
- **Actions**: 4 discrete actions (`NOOP`, `LEFT`, `MAIN`, `RIGHT`)
- **Goal**: Land safely between flags ($+100$ to $+140$ points)

- **Reward Shaping**: Distance, speed, angle penalties; fuel costs
- **PPO Advantages**: Stable learning with sparse rewards
- **Policy Network**: MLP ($8 \rightarrow 128 \rightarrow 128 \rightarrow 4$)
- **Training**: Converges in $1000 - 3000$ episodes
- **Performance**: Solves environment (average reward > 200)

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Outline

Trust Region Concept

Trust Region Optimization

- Optimize within a trusted region of current policy
- Ensures monotonic improvement
- Prevents policy collapse

Trust Region Policy Optimization (TRPO) Constraint

$$E[KL(\pi_{\theta_{old}}(\cdot|s), \pi_{\theta}(\cdot|s))] \leq \delta$$

Trust Region Concept

Trust Region Optimization

- Optimize within a trusted region of current policy
- Ensures monotonic improvement
- Prevents policy collapse

Trust Region Policy Optimization (TRPO) Constraint

$$E[KL(\pi_{\theta_{old}}(\cdot|s), \pi_{\theta}(\cdot|s))] \leq \delta$$

TRPO Mathematical Formulation

Constrained Optimization Problem

$$\text{maximize} \quad L(\theta) \tag{1}$$

$$\text{subject to} \quad \bar{D}_{KL}(\theta_{old}, \theta) \leq \delta \tag{2}$$

- Solved using conjugate gradient method
- More principled than PPO but computationally expensive
- Guarantees monotonic improvement

TRPO Mathematical Formulation

Constrained Optimization Problem

$$\text{maximize} \quad L(\theta) \tag{1}$$

$$\text{subject to} \quad \bar{D}_{KL}(\theta_{old}, \theta) \leq \delta \tag{2}$$

- Solved using conjugate gradient method
- More principled than PPO but computationally expensive
- Guarantees monotonic improvement

Natural Policy Gradients

Natural Policy Gradient

- Uses Fisher information matrix to scale gradients
- Accounts for geometry of policy space
- Update: $\theta \leftarrow \theta + \alpha F^{-1} \nabla_\theta J(\theta)$

- More stable updates than vanilla policy gradients
- Related to TRPO as it also considers trust regions
- Can be used in large action spaces

Natural Policy Gradients

Natural Policy Gradient

- Uses Fisher information matrix to scale gradients
- Accounts for geometry of policy space
- Update: $\theta \leftarrow \theta + \alpha F^{-1} \nabla_\theta J(\theta)$

- More stable updates than vanilla policy gradients
- Related to TRPO as it also considers trust regions
- Can be used in large action spaces

Practical Example: `TRPO in Complex Continuous Control`

Complex Continuous Control Tasks

- High-dimensional state and action spaces
- Requires precise control and coordination
- Examples: *robotic manipulation*, *locomotion*

- **TRPO Advantages**: Handles complex dynamics and constraints
- **Policy Network**: Deep neural networks for representation
- **Training**: Sample-efficient with trust region updates
- **Performance**: Achieves state-of-the-art results

Practical Example: TRPO in Complex Continuous Control

Complex Continuous Control Tasks

- High-dimensional state and action spaces
- Requires precise control and coordination
- Examples: *robotic manipulation*, *locomotion*

- **TRPO Advantages**: Handles complex dynamics and constraints
- **Policy Network**: Deep neural networks for representation
- **Training**: Sample-efficient with trust region updates
- **Performance**: Achieves state-of-the-art results

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Practical Example: TRPO in Complex Continuous Control

Complex Continuous Control Tasks

- High-dimensional state and action spaces
- Requires precise control and coordination
- Examples: *robotic manipulation*, *locomotion*

- **TRPO Advantages**: Handles complex dynamics and constraints
- **Policy Network**: Deep neural networks for representation
- **Training**: Sample-efficient with trust region updates
- **Performance**: Achieves state-of-the-art results

Practical Example: TRPO in Complex Continuous Control

Complex Continuous Control Tasks

- High-dimensional state and action spaces
- Requires precise control and coordination
- Examples: *robotic manipulation*, *locomotion*

- **TRPO Advantages**: Handles complex dynamics and constraints
- **Policy Network**: Deep neural networks for representation
- **Training**: Sample-efficient with trust region updates
- **Performance**: Achieves state-of-the-art results

Multi-Agent Environments

Challenges

- Non-stationary environments
- Other agents are learning simultaneously
- Coordination and competition
- Communication between agents

- **Cooperative:** Agents share common goal
- **Competitive:** Zero-sum games
- **Mixed:** Combination of cooperation and competition

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Multi-Agent Environments

Challenges

- Non-stationary environments
- Other agents are learning simultaneously
- Coordination and competition
- Communication between agents

- **Cooperative**: Agents share common goal
- **Competitive**: Zero-sum games
- **Mixed**: Combination of cooperation and competition

Multi-Agent Environments

Challenges

- Non-stationary environments
- Other agents are learning simultaneously
- Coordination and competition
- Communication between agents

- **Cooperative**: Agents share common goal
- **Competitive**: Zero-sum games
- **Mixed**: Combination of cooperation and competition

Multi-Agent Environments

Challenges

- Non-stationary environments
- Other agents are learning simultaneously
- Coordination and competition
- Communication between agents

- **Cooperative**: Agents share common goal
- **Competitive**: Zero-sum games
- **Mixed**: Combination of cooperation and competition

Nash Equilibrium in RL

Nash Equilibrium

A strategy profile where no agent can improve by unilaterally changing their strategy.

- Difficult to compute in practice
- May not always exist or be unique
- Agents may not converge to equilibrium
- Alternative: Find correlated equilibria

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Nash Equilibrium in RL

Nash Equilibrium

A strategy profile where no agent can improve by unilaterally changing their strategy.

- Difficult to compute in practice
- May not always exist or be unique
- Agents may not converge to equilibrium
- Alternative: Find correlated equilibria

Nash Equilibrium in RL

Nash Equilibrium

A strategy profile where no agent can improve by unilaterally changing their strategy.

- Difficult to compute in practice
- May not always exist or be unique
- Agents may not converge to equilibrium
- Alternative: Find correlated equilibria

Centralized Training, Decentralized Execution

CTDE Paradigm

- **Training**: Use global information and centralized critic
- **Execution**: Each agent acts with only local observations

- Addresses partial observability
- Enables coordination during training
- Maintains decentralized execution
- Examples: `MADDPG`, `COMA`, `QMIX`

Centralized Training, Decentralized Execution

CTDE Paradigm

- **Training**: Use global information and centralized critic
- **Execution**: Each agent acts with only local observations

- Addresses partial observability
- Enables coordination during training
- Maintains decentralized execution
- Examples: `MADDPG`, `COMA`, `QMIX`

Centralized Training, Decentralized Execution

CTDE Paradigm

- **Training**: Use global information and centralized critic
- **Execution**: Each agent acts with only local observations

- Addresses partial observability
- Enables coordination during training
- Maintains decentralized execution
- Examples: `MADDPG`, `COMA`, `QMIX`

Communication in Multi-Agent Systems

Communication Protocols

- **Explicit**: Agents exchange messages
- **Implicit**: Agents infer others' intentions from actions

- Communication can enhance coordination and performance
- Challenges: Bandwidth constraints, noise, security
- Research area: Learning communication protocols end-to-end

Communication in Multi-Agent Systems

Communication Protocols

- **Explicit**: Agents exchange messages
- **Implicit**: Agents infer others' intentions from actions

- Communication can enhance coordination and performance
- Challenges: Bandwidth constraints, noise, security
- Research area: Learning communication protocols end-to-end

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Communication in Multi-Agent Systems

Communication Protocols

- **Explicit**: Agents exchange messages
- **Implicit**: Agents infer others' intentions from actions

- Communication can enhance coordination and performance
- Challenges: Bandwidth constraints, noise, security
- Research area: Learning communication protocols end-to-end

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

Communication in Multi-Agent Systems

Communication Protocols

- **Explicit**: Agents exchange messages
- **Implicit**: Agents infer others' intentions from actions

- Communication can enhance coordination and performance
- Challenges: Bandwidth constraints, noise, security
- Research area: Learning communication protocols end-to-end

Outline

Thanks!

Questions?

Repo: *https://github.com/EngAndres/ud-public/tree/main/courses/reinforcement-learning*

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS