The City College of New York

CSC 59866-E: Senior Project I

AI Agents for Decision Making in the Real World

By Saptarashmi Bandyopadhyay
Email: sbandyopadhyay@ccny.cuny.edu, sbandyopadhyay@gc.cuny.edu
Assistant Professor of Computer Science
City College of New York and Graduate Center at the City University of New York

February 9, 2026 CSC 59866

Generalizability: AI Agents and Reward Modeling

Logistics

Hand-Written Research Paper Review 1 (Please submit via Brightspace/Email if you haven't).

Upcoming (Feb 11): Project Groups and Topics will be assigned **next class**.

Coding Assignment 1: Will be released today, due date will be pushed back.

- *Reminder:* Don't wait until Feb 17 to try running Google Colab for the first time!

Generalization

From Memorizing to Generalizing

Recap: In Lecture 4, we saw that a Q-Table can solve a fixed 10 x 10 grid perfectly.

The Problem: If we move the goal one square to the right, a tabular agent fails completely. It has zero **Generalizability**.

Definition: Generalizability is the ability of an agent to perform well in environments it has never seen, based on concepts it learned during training.

Much of AI (and broadly) ML research is driven by the need to generalize as much as possible!

Types of Generalization

- **I.I.D. (Independent and Identically Distributed):**
  - Test environment is "statistically similar" to training.
  - *Example:* Training on Mario Levels 1-10, Testing on Level 11 (similar physics, new layout).
- **O.O.D. (Out-of-Distribution) / Domain Shift:**
  - Test environment is fundamentally different.
  - *Example:* Training a self-driving car in sunny California, testing it in snowy New York.
  - This is where 90% of commercial AI agents fail.

Zero-Shot vs. Few-Shot Learning

- **Zero-Shot:** The agent must solve a task with *no* prior examples, just a description (e.g., "Find the red box").
- **Few-Shot:** The agent gets 1-5 demonstrations of the new task before it must perform.
- **Why it matters for your Project:** You cannot train on every possible scenario. Your agent *must* handle some level of uncertainty.

Reward Modeling

The Reward Hypothesis

"All goals can be described as the maximization of expected cumulative reward." — *Richard Sutton, creator of REINFORCE*

The Challenge: The agent will maximize the *number* you give it, not the *intent* you had.

Types of Rewards:

- **Dense Rewards:** +0.1 for every step closer to the goal. (Fast learning, high bias).
- **Sparse Rewards:** +100 only when the goal is achieved. (Slow learning, true objective)

Reward Hacking

Definition: When an agent finds a loophole to maximize reward without solving the task.

Video Game Example (CoastRunners):

- Agent was rewarded for hitting "turbos." It found a loop where it could hit turbos forever, crash, and repeat, never actually finishing the race.

Reward Shaping

How can we “nudge” the behavior without overengineering the reward?

- **Concept:** Adding "hints" to the reward function to guide the agent.
- **Potential-Based Shaping:** $R' = R + \gamma\Phi(s') - \Phi(s)$
  - Mathematically proven *not* to change the optimal policy (*Ng et. al. 1999*)
  - *Example:* $\Phi(s)$ = "Distance to Goal".
  - This helps the agent learn *faster* without changing *what* it learns.

Example: Infinite Loops

- **State S (Start):** The Agent starts here.
- **Path A (The Goal):** Takes **3 steps** to reach the Goal (G).
  - Reward at $G = +10$ (Terminal State).
  - Step cost $= 0$.
- **Path B (The Loop):** Takes **1 step** to reach a "Turbo Button" (T).
  - Reward at $T = +1$.
  - From T , the agent instantly respawns at T (Infinite Loop).
- **Discount Factor (γ):** 0.9 .

As a human, you know Path A is 'correct'. But mathematically, which path will the AI Agent choose?

Example: Infinite Loops

The **Cumulative Reward formula** is $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$

Step 1: Calculate Path A (The Goal)

- Time $t = 1$: Step 1 (Reward 0)
- Time $t = 2$: Step 2 (Reward 0)
- Time $t = 3$: Arrive at Goal (Reward +10)
- **Math:**

$$G_{Goal} = 0 + 0 + \gamma^2(10)$$

$$G_{Goal} = (0.9)^2 \times 10 = 0.81 \times 10 = \mathbf{8.1}$$

Example: Infinite Loops

The **Cumulative Reward formula** is $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$

Step 2: Calculate Path B (The Loop)

- Time $t = 1$: Hit Turbo (Reward +1) Time $t = 2$: Hit Turbo (Reward +1) ... Forever.
- **Math:** This is a Geometric Series $S = \frac{a}{1-r}$.

$$G_{Loop} = 1 + \gamma(1) + \gamma^2(1) + \dots$$

$$G_{Loop} = \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}$$

$$G_{Loop} = \frac{1}{1-0.9} = \frac{1}{0.1} = \mathbf{10.0}$$

Advanced Concepts in Generalization and Reward Modeling

Inverse Reinforcement Learning

Problem: How do you write a reward function for "Driving safely"? It's too complex (don't hit cars, obey laws, be polite, etc.).

Solution: Inverse Reinforcement Learning (IRL).

- Instead of designing the reward, we collect data of a **Human Expert** driving.
- The AI assumes the human is optimal and tries to mathematically calculate the reward function the human is using.
- *Application:* This is how Waymo/Tesla train behavioral layers.

Curriculum Learning

Concept: Don't start with the hardest task.

Method:

1. Train on a 3 x 3 grid with no obstacles.
2. Train on a 5 x 5 grid with static obstacles.
3. Train on a 10 x 10 grid with moving enemies.

Relevance: For your Senior Projects, if your agent isn't learning, **simplify the task**. Don't just tune hyperparameters.

Summary and Next Steps

Summary

Generalization: Train on variability to survive the real world.

Rewards: Be careful what you wish for.

- Sparse = Safe but Slow.
- Dense = Fast but Risky (Hacking).

Design Principle: Always visualize *what* your agent is actually doing, not just the reward curve. High reward != Good behavior.

Next Steps

Look out for Coding Assignment 1 being released soon!

Project Assignment: I will assign the **10 Research Topics** and **Groups** this week! Attendance next class is highly recommended for questions about the project and groups.

Questions?