Reinforcement Learning
Architecture 10/09
ORCS 4529: Fall 2025

Aksel Kretsinger-Walters, adk2164
adk2164@columbia.edu
Due: 10/09/2025 11:59 PM ET

State

Discrete-Time Markov Decision Process (MDP)

Time steps are 1 minute long

Environment

- Static class - no state changes
- Embedded into l_2
- Graph:
  - Nodes - intersections
  - Edges - roads
- Nodes:
  - Only places rides can start/end
  - l_2 coordinates
  - bool is_charging_station
  - int num_available_chargers
- Edges:
  - Length
  - 2-way speed limit, possibly asymmetric
  - *potentially:* 2-way current speed, possibly asymmetric (this would be dynamic)

Fleet

- n=10? vehicles
- Per vehicle:
  - Vehicle ID

  - Location
  - Battery level (percentage)
  - Status (available, en route to pickup, with passenger, en route to charging station)

Demand

I'm a little unsure about whether this should be *explicitly* modeled in the state or "learned" by the agent. I'm learning towards the former for a few reasons

(a) It's relatively independent of the actions of the agent

(b) It can be estimated from historical data

(c) Both assumptions above are likely to be true in the real world

(d) It greatly simplifies the learning problem

If we do explicitly model it

- Location
- Expected demand (number of requests) in next time step

Active Rides

I think that this class should mostly serve as an enriched tuple between a vehicle and a ride request (described below)

- Matched Vehicle ID
- Request ID
- Status (en route to pickup, with passenger)
- Time to pickup (minutes)
- Duration of ride (minutes)
- Estimated remaining time (minutes)

Ride Requests

There's one part of this process that I believe is non-trivial. In the real world, the platform (e.g. Uber, Lyft) suggests a menu of prices to the rider, who then selects one. Subsequently, all (nearby) drivers are notified of the ride request, and one can accept it.

In our simulation, if the agent recommends prices, intuitively it would be a price that the agent would accept. So that transition probability is 1. However, you could make a case that riders should "bid" on rides, and the agent can (1) accept (2) reject (3) counter-offer. For the sake of simplicity, I would rather model the process as (1) customer requests a ride (2) the agent suggests a price (3) the customer either accepts or rejects the price based on some probabilistic model. This would allow the agent to learn to price rides optimally. It would also provide a bound on the maximum price the agent could set (since the agent would learn to just rise prices without bound if the customer always accepted).

- $n_v \sim \text{Poisson}(\Lambda)$ new requests per time step (this lives in Demand class)
- Each request:
  - Request ID
  - Pickup location
  - Dropoff location
  - Request time
  - Status (pending, assigned, completed, cancelled)