Reminders

- TA's announcement on course project report 1
- PRA4 due 3/14
- Course project progress report 2 due 3/26
- Come to OH for course project discussion!

Artificial Intelligence Methods for Social Good

Lecture 15:

Case Study: RL for Bike Repositioning

17-537 (9-unit) and 17-737 (12-unit)

Fei Fang

feifang@cmu.edu

Outline

- Deep Reinforcement Learning
  - Deep Q Learning for Ranger Patrol Planning
- A Spatio-Temporal Reinforcement Learning Algorithm for Bike Repositioning

Learning Objectives

- Briefly describe
  - Deep Q Learning
- For the adaptive ranger patrol problem, understand
  - Method used to solve the problem
- For the bike repositioning problem, understand
  - Motivation
  - Problem to be solved
  - Method
  - Evaluation

Recap: Q-Learning

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

- Q-Learning
  - Maintain a Q table
  - Start with some random guess of optimal Q values, $\hat{Q}^*(s,a)$
  - Agent interact with the environment following some policy π (no need to be optimal)
  - In one step of a trial: agent is in state s, take action $a = \pi(s)$, get reward r, end up in state s'
  - Update the estimated optimal Q value at (s,a) with

$$\hat{Q}^*(s,a) \leftarrow (1-\alpha)\hat{Q}^*(s,a) + \alpha(r + \gamma \max_{a'} \hat{Q}^*(s',a'))$$

Recap: Policy Gradient

$$J(\theta) = \mathbb{E}[V^{\pi}(s)]$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{s,a\sim\pi_\theta}[Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(s,a)]$$

- Estimate gradient through sampling
  - Sample possible histories
  - Compute gradient as average value of $Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(s,a)$
  - How to compute $Q^{\pi_\theta}(s,a)$?
    - REINFORCE: Directly use discounted reward from sampled history

Deep Q-learning playing Atari

Learn to Play Atari Games

https://youtu.be/V1eYniJ0Rnk?t=20

Deep Q-Networks

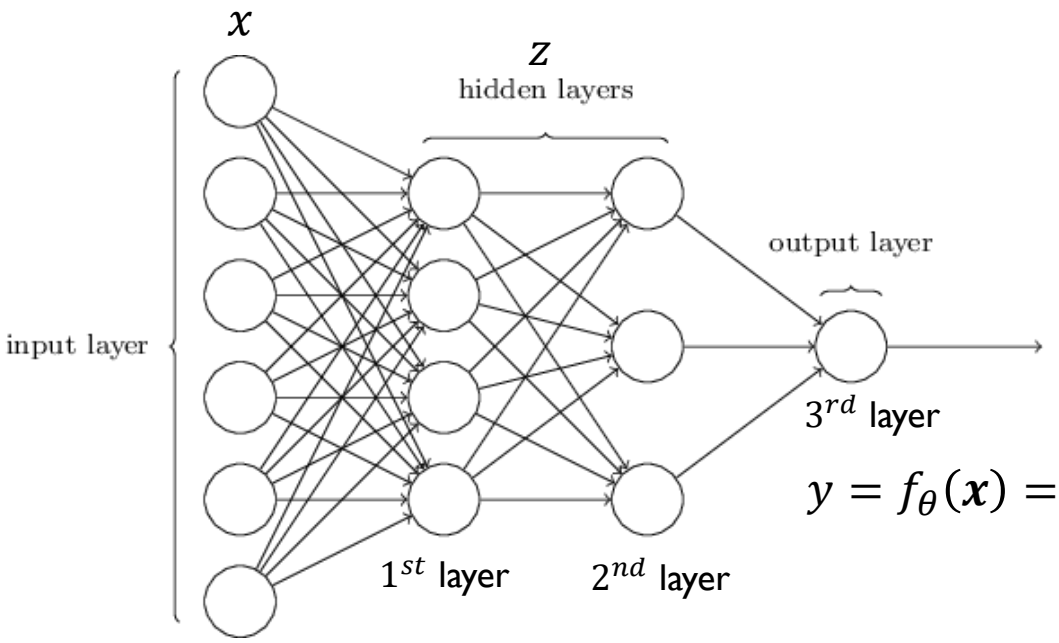
- When there are too many states, we cannot use a tabular approach to store and update the Q values based on the update rule

$$\hat{Q}^*(s,a) \leftarrow (1-\alpha)\hat{Q}^*(s,a) + \alpha(r + \gamma \max_{a'} \hat{Q}^*(s',a'))$$

- Deep-Q networks
  - Use a neural network to approximate the Q-value function

Recall: Neural Networks

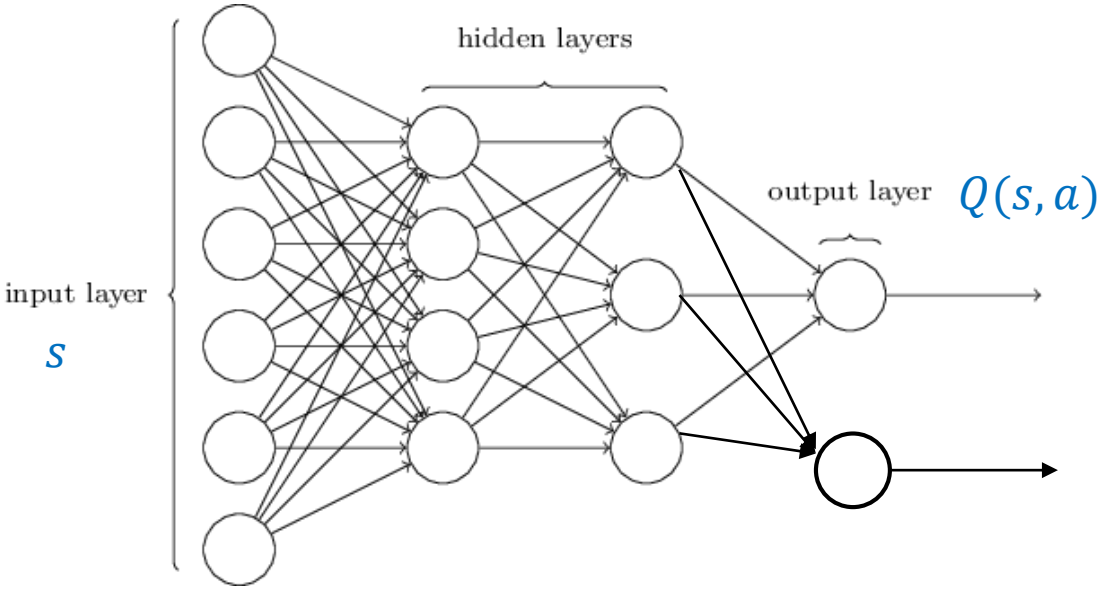
- Feedforward Neural Network
  - $\boldsymbol{x} \in \mathbb{R}^n \rightarrow$ layers of units $\rightarrow y = f_\theta(\boldsymbol{x}) \in \mathbb{R}$ or $\in \mathbb{R}^M$



$$y = f_\theta(\boldsymbol{x}) = f^{(3)}\left(f^{(2)}\left(f^{(1)}(\boldsymbol{x})\right)\right)$$

Deep-Q networks

- Input: a vector that describes the state
- Output: a vector that contains the estimated Q-value for each valid action
- Can use convolutional layers as well



Train the Q-Network

- Iteratively update network parameters θ through a gradient step towards minimizing loss function

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'}[\left(\hat{Q}(s,a|\theta) - y\right)^2]$$

y is the target value, which is computed as

$$y = r + \gamma \max_{a'} \bar{Q}(s', a')$$

$\bar{Q}(s', a')$ is a target Q function whose parameters are periodically updated with the most recent θ

Deep Q-Learning with Experience Replay

- Experience replay: Store the agent's experience at each time step $e_t = (s_t, a_t, r_t, s_{t+1})$ in a data-set $\mathcal{D} = e_1, e_2, \ldots, e_N$, pooled over many episodes into a replay memory
- In each iteration, randomly sample experiences from $\mathcal{D}$, use them to update parameters in the Q network
- After parameter update, choose an action based on ϵ −Greedy

Deep Q-Learning with Experience Replay for Atari Games

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

DQN In Practice

- Train in the Pong game environment using already implemented DQN code

```
python -m baselines.run --alg=deepq --env=PongNoFrameskip-v4
```

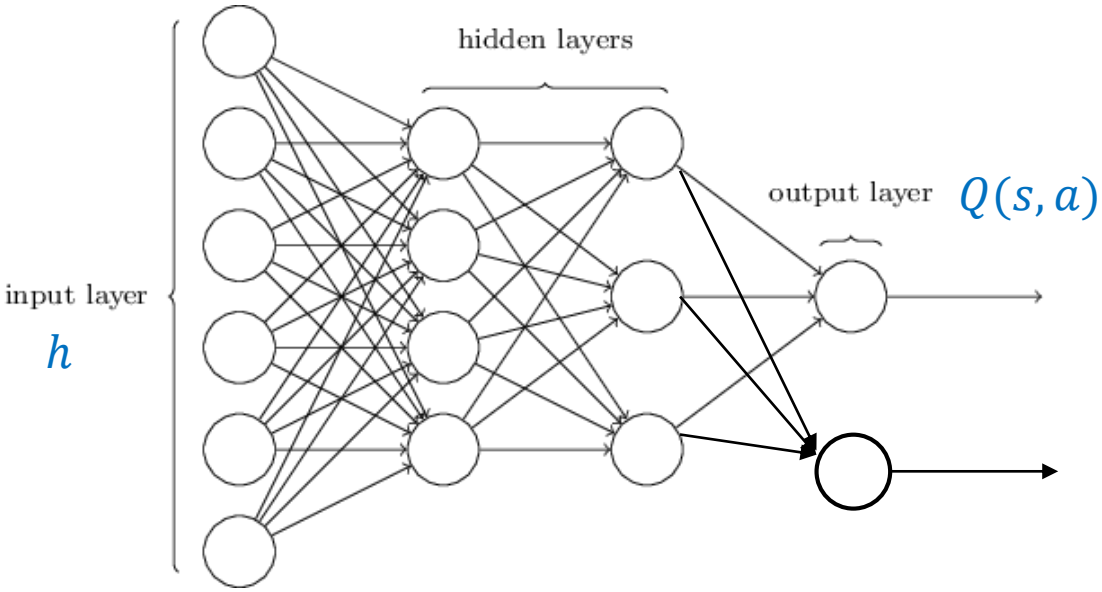
With OpenAI Baselines

Set different hyperparameters

```
def atari():
    return dict(
        network='conv_only',
        lr=1e-4,
        buffer_size=10000,
        exploration_fraction=0.1,
        exploration_final_eps=0.01,
        train_freq=4,
        learning_starts=10000,
        target_network_update_freq=1000,
        gamma=0.99,
        prioritized_replay=True,
        prioritized_replay_alpha=0.6,
        checkpoint_freq=10000,
        checkpoint_path=None,
        dueling=True
    )
```

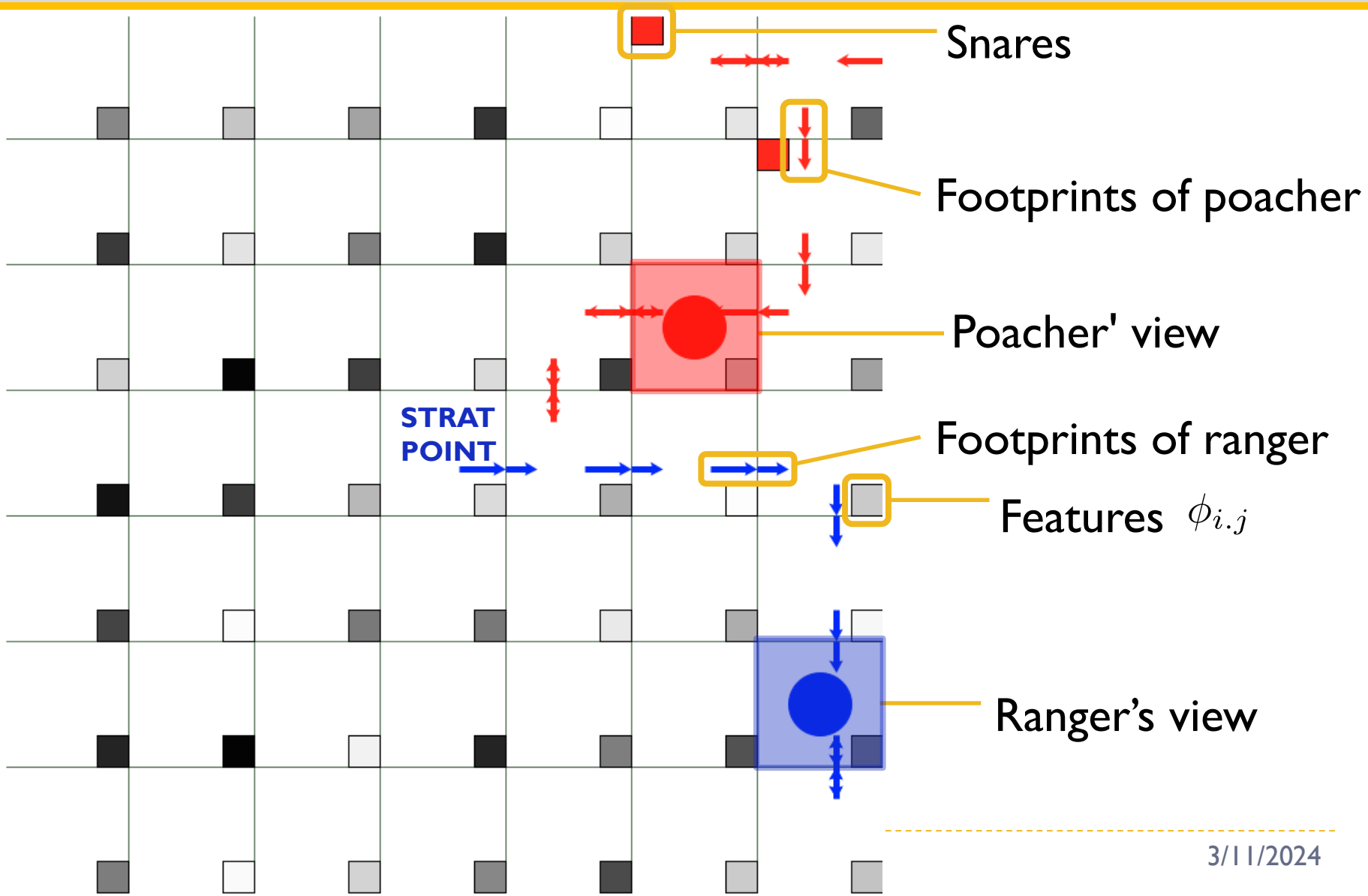
Deep Q-Learning for Partially Observable Problems

- Input: a vector that describes the history (observation + actions)
- Output: a vector that contains the estimated Q-value for each valid action

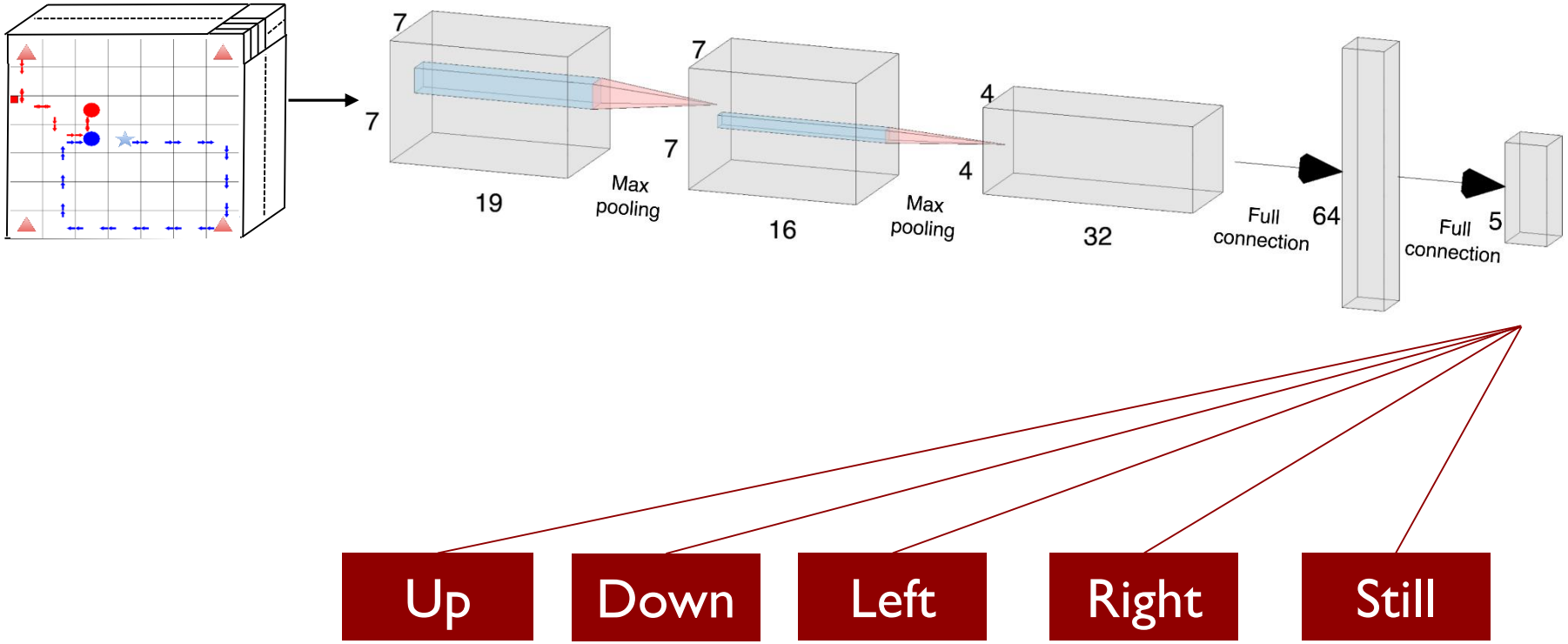


Conceptually, can be viewed as constructing a new MDP with state defined by the history

Ranger vs Heuristic Poacher

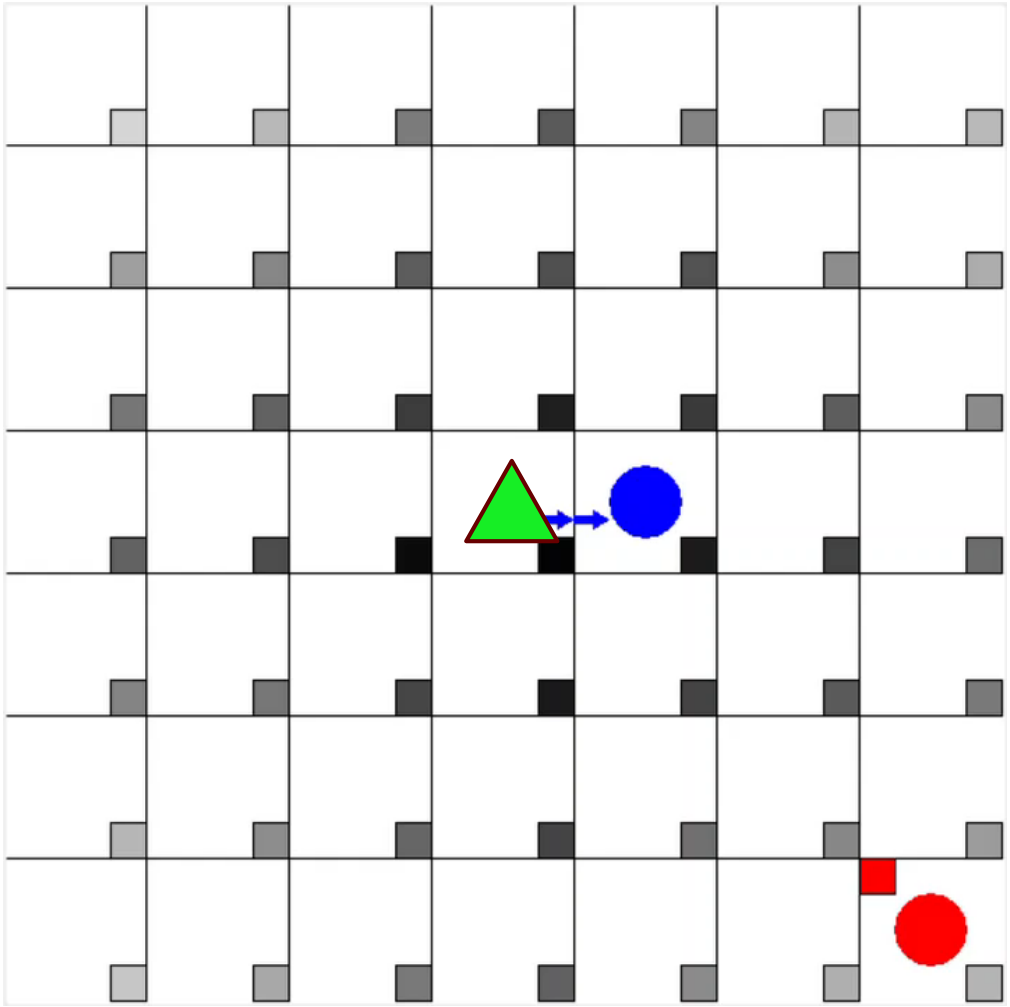


DQN Ranger Trained Against Heuristic Poacher



- Q Network: Game state → Q-value

DQN Defender Trained Against Heuristic Attacker



Attacker

Snares

Defender

Patrol Post

Policy Gradient with a Q-Network

- Recall in PG, we want to update θ towards the direction

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a)\, Q^{\pi_\theta}(s,a)]$$

- In REINFORCE, we use a sample return to estimate Q
- We can also train a Q-network to approximate Q
- Various actor-critic algorithms
  - "Critic" estimates the value function (e.g., Q value, V value)
  - "Actor" updates the policy distribution in the direction suggested by the Critic

Outline

- Deep Reinforcement Learning
  - Deep Q Learning for Ranger Patrol Planning
- A Spatio-Temporal Reinforcement Learning Algorithm for Bike Repositioning

Bike Repositioning Problem

- Bikesharing is popular and convenient

https://www.post-gazette.com/news/transportation/2018/12/02/Healthy-Ride-Pittsburgh-bike-sharing-program-adding-stations-new-neighborhoods/stories/201812020119

Bike Repositioning Problem

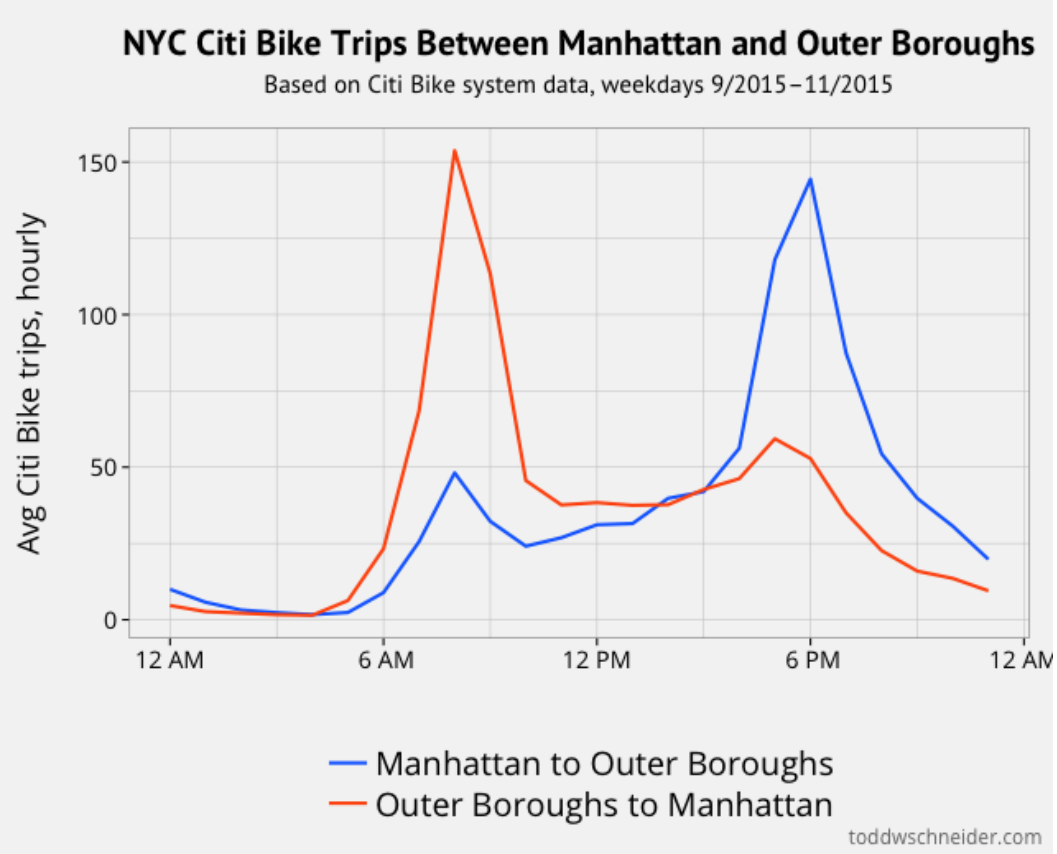
- Bikesharing is popular and convenient, but you may fail to rent/return

https://ggwash.org/view/38849/empty-bikeshare-stations-dont-always-mean-long-waits

https://www.treehugger.com/bikes/vancouvers-mobi-bike-share-system-just-weird.html

Bike Repositioning Problem

- Imbalanced demand over space and time



Video: https://toddwschneider.com/posts/a-tale-of-twenty-two-million-citi-bikes-analyzing-the-nyc-bike-share-system/

Bike Repositioning Problem

- Need to reposition them

https://farm9.static.flickr.com/8103/8588117819_8eaf60bd39_b.jpg

https://bikeportland.org/2016/09/07/portland-now-using-pedal-powered-trikes-to-help-rebalance-bike-share-stations-191007

Bike Repositioning Problem

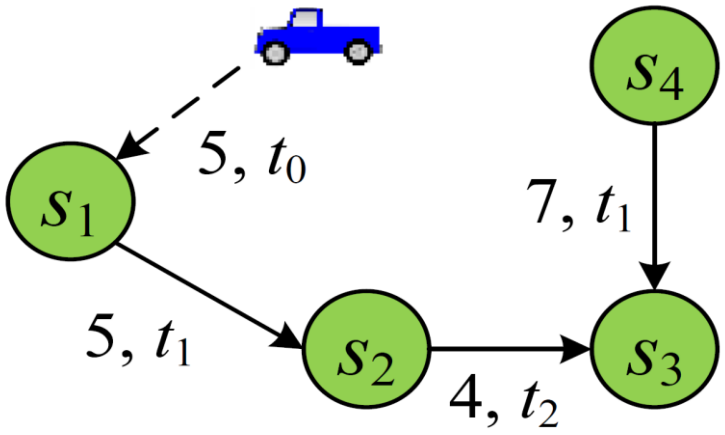
- What is optimal repositioning? Various formulations:
  - Minimize the customer loss in a long period given a fixed repositioning budget
  - Reduce repositioning cost while ensuring a level of customer satisfaction
  - Maximize profit minus repositioning cost

Bike Repositioning Problem

- How to achieve optimal repositioning?
- Redistribute the bikes after an unbalance is observed?
  - Might be too late
- Predict bike usage for the next time period and reposition greedily based on that?
  - Might still be too short-sighted

Challenges in Bike Repositioning

- Challenge 1: Repositioning now can impact the future

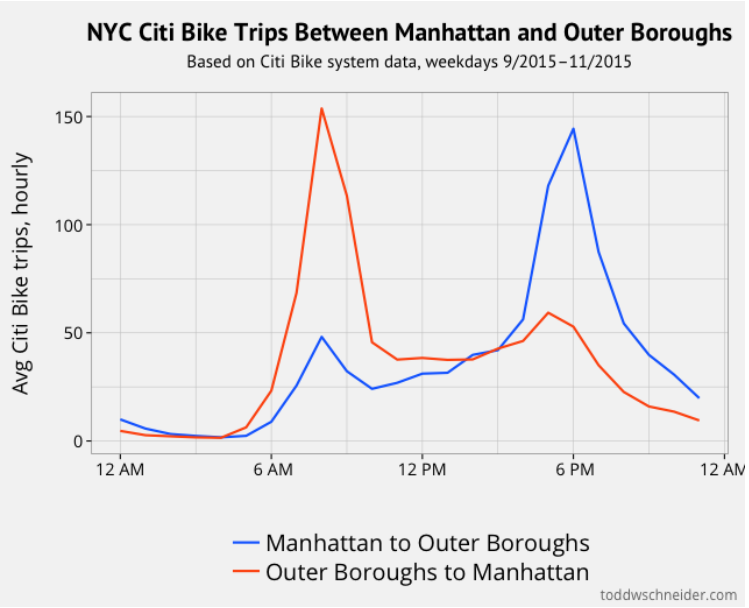


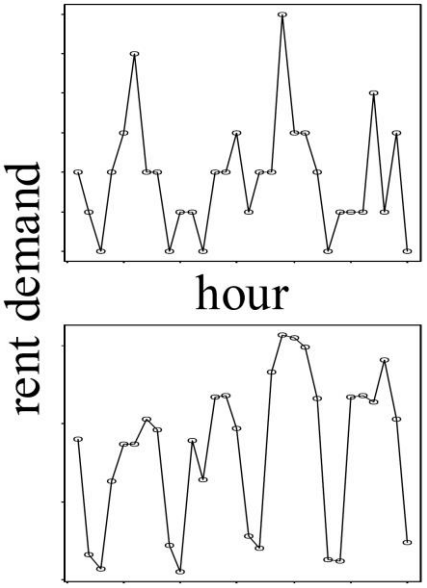
Green circle: empty-dock station
$5, t_1$: 5 bikes will be rented from s_1 and returned to s_2 at time period $t_1 \in \mathbb{Z}$
Dashed arrow: how a trike repositions
$t_0 < t_1 < t_2$ and move between nodes takes 1 time step

Poll 1: Assuming no bikes are available at any dock before t_0, how many more customers the platform is able to serve in total due to the repositioning?
A: 5; B: 4; C: 9; D: None of the above; E: I don't know

Challenges in Bike Repositioning

- Challenge 2. It is a complex system
  - Large-scale: tens of trikes repositioning among hundreds of stations in a system simultaneously
  - Dynamic: demand changes over space and time with large fluctuation





Challenges in Bike Repositioning

- Challenge 3: Uncertainties in practice

http://growthspurtcoaching.com/on-what-to-do-when-someone-else-falls-down/
https://www.npr.org/sections/parallels/2018/03/07/591140541/a-push-to-modernize-philippine-transport-threatens-the-beloved-jeepney

Bike Repositioning as an RL problem

The planner is the agent

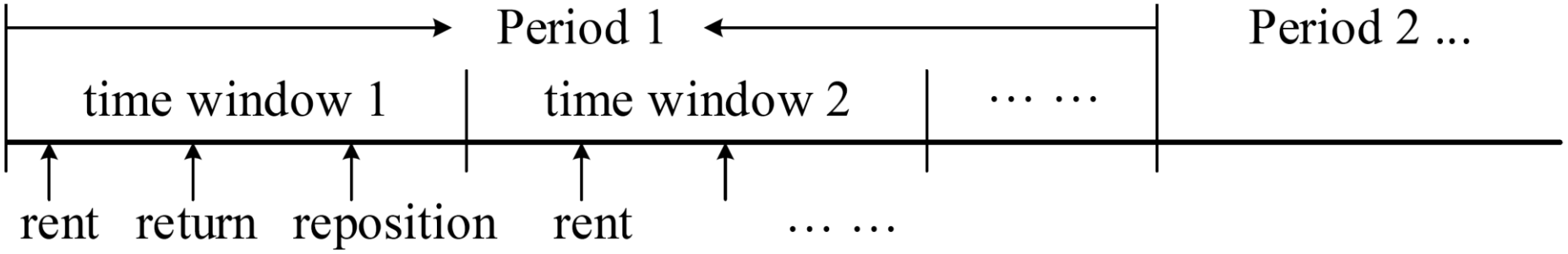
- Discussion: Formulate the problem as an POMDP
  - State S (unknown to agent):
  - Action A:
  - Observation Ω:
  - Observation probabilities $O(o|s_t)$ (unknown)
  - Transition function (unknown) $T(s_t, a_t, s_{t+1})$
  - R: reward function r_t = the negative customer loss

Bike Repositioning as an RL problem

- Action (for one trike): where a trike should go to pick up or unload how many bikes
- Observations
  - Current bike and dock availability at each region
  - Predicted rent and return demands in all future periods
  - Location of all the trikes
  - What repositioning task the trikes are completing
  - Current number of bikes on each trike
  - Expected arrival times of the trikes
  - …

Bike Repositioning as an RL problem

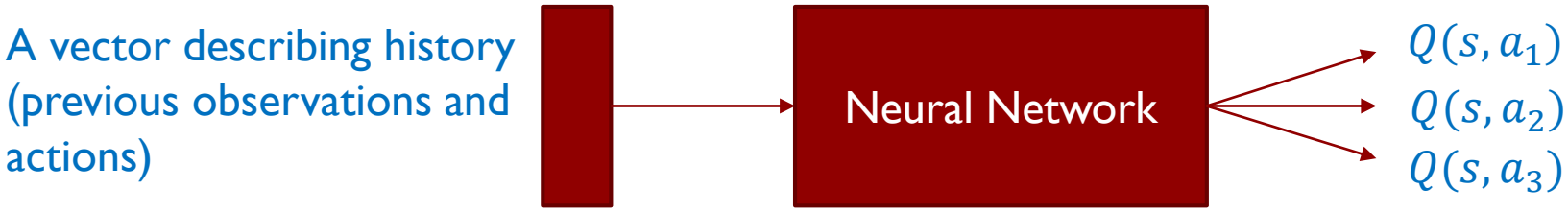
- An RL problem because we do not have explicit model of T or R
- Can't afford trial-and-error learning in the real world
- Build a system simulator
  - Simulate the (1) rent process, (2) return process, (3) reposition process (with noise)



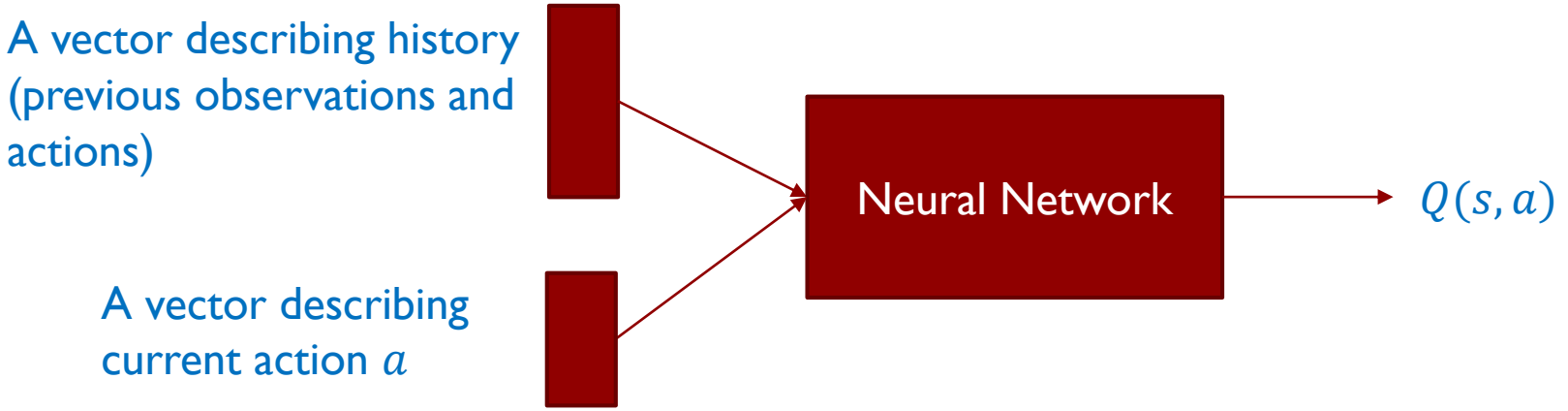
Let's Start Simple

- Assume we only have a small number of stations
- Assume we only control one trike
- There are other trikes are in the system, but we do not control them (can be treated as part of the environment)
- Apply deep Q-Learning to the problem

Q Network for Bike Repositioning

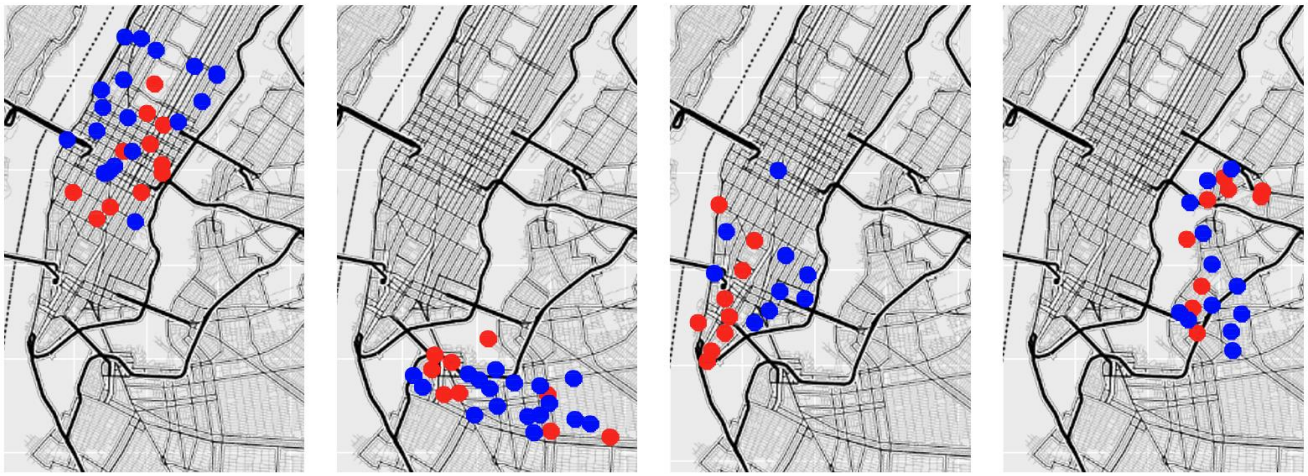


- Too many actions!
- Instead of having a vector output, build a network that also takes action as input, and output a value



How to Scale Up?

- Problem is much more challenging with more trikes under control and more stations
- Learn an optimal inner-cluster reposition policy
  - Group the stations into clusters
  - Only consider repositioning within each cluster



C_1 C_2 C_3 C_4

Blue: starved
Red: jammed

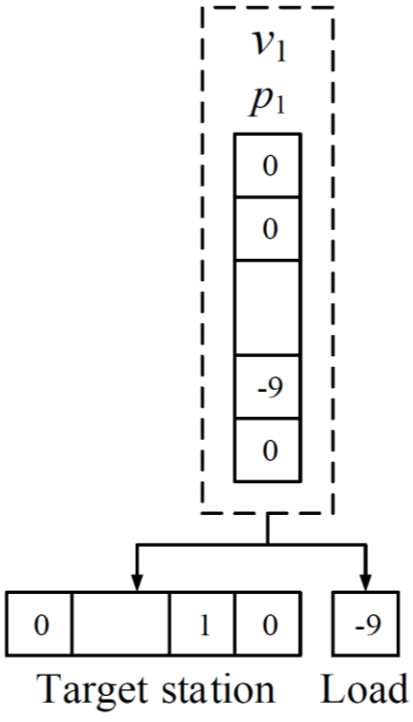
How to Scale Up?

- Multiple agents with the same policy
  - Each trike is a single agent and treat all other trikes as part of the environment
  - All the trikes share the same Q-network

Representation of Action

- For the trike under our control (w.l.o.g., assume it is trike 1): a vector describing where the trike should go to pick up or unload how many bikes

- Convert action to a one-hot vector and a scalar ($n + 1$ numbers in total)



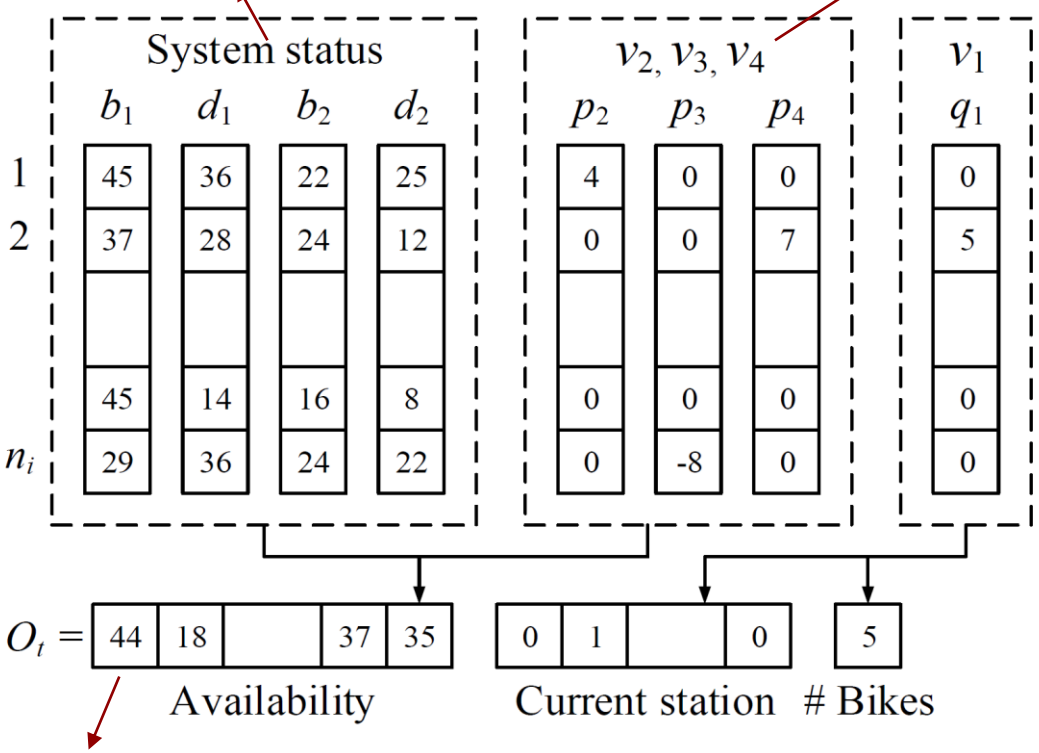
Representation of State/History

- $s_t = (O_{t-L_i}, a_{t-L_i}, \dots, O_{t-1}, a_{t-1}, O_t, t)$

Current bike and dock availability

Predicted rent and return demand

Status of other trikes

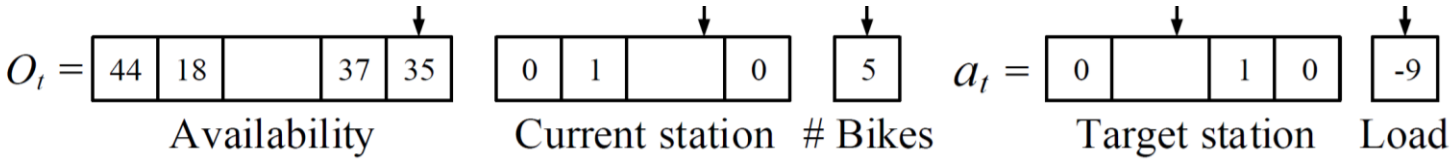


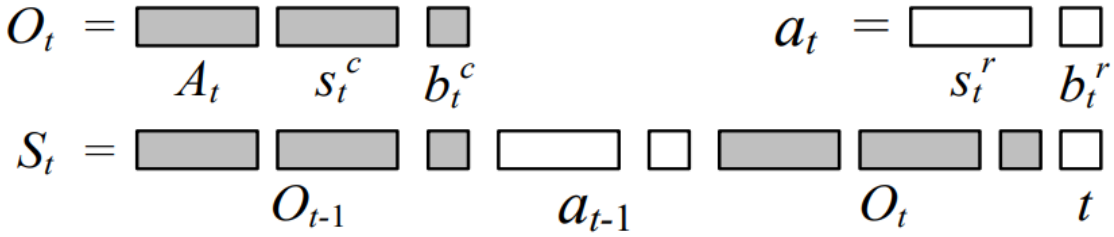
Status of this trike v_1 is at region s_2 with 5 bikes on it

Predicted bike availability in the coming period

Representation of State/History

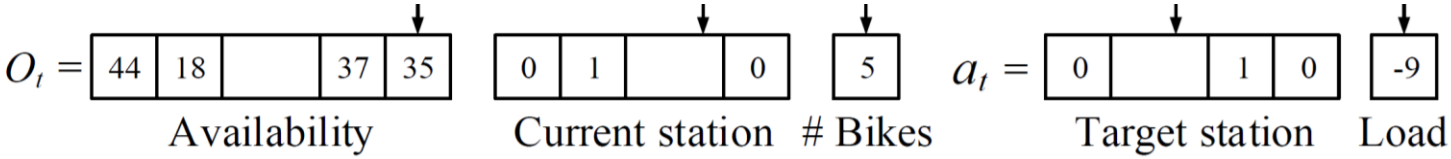
- $s_t = (O_{t-L_i}, a_{t-L_i}, \dots, O_{t-1}, a_{t-1}, O_t, t)$



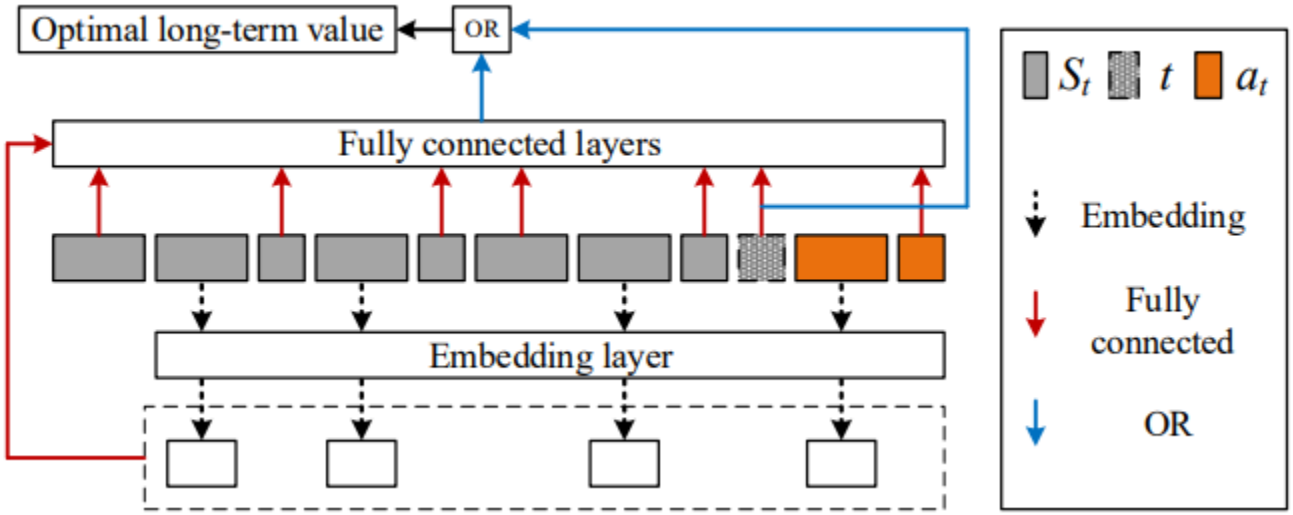


Build the Q-Network

- Recall



- Network structure



Evaluation

- Conduct experiments on real-world datasets from Citi Bike in Apr. - Oct. 2016
- Compare with baselines (No Reposition, Greedy Reposition, Prediction based Random Reposition, Optimization-based Reposition)
- Evaluation Metric: total customer loss in long period, including the ones failed to rent and the ones failed to return in morning rush hours

Customer Loss		NR	PR	GR	PGR	OR	STRL
Cluster 1	C_1	267	178	200	157	190	113
	C_3	286	229	256	238	237	178
	$C_1 + C_3$	553	407	456	395	427	291

Discussion

- Can you think of any other social good problem or problems you see in daily life that can be modeled as an RL problem and can potentially be solved using Deep Q-learning?

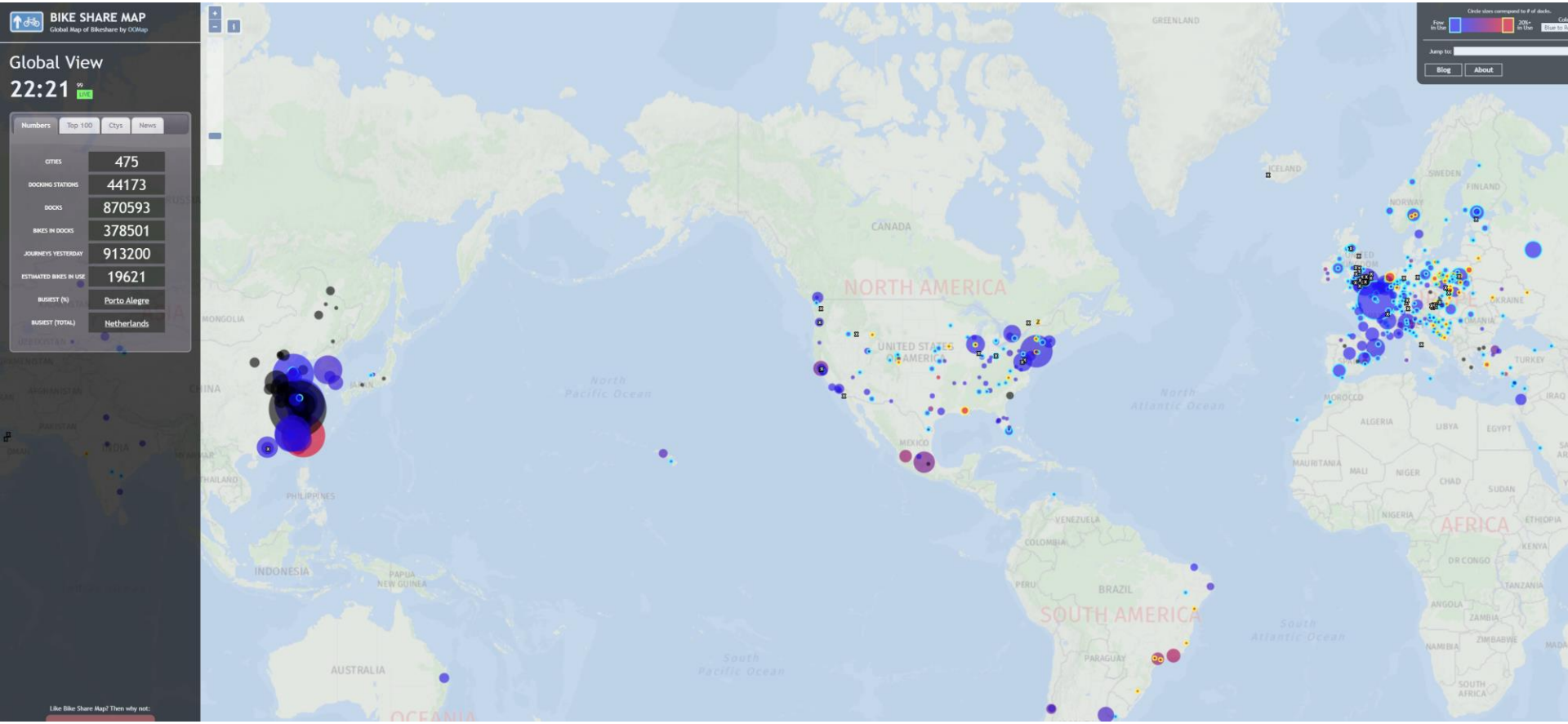
Other Resources and References

- Dynamic Bike Reposition: A Spatio-Temporal Reinforcement Learning Approach
- A Dynamic Approach to Rebalancing Bike-Sharing Systems
- Station Site Optimization in Bike Sharing Systems
- A Tale of Twenty-Two Million Citi Bike Rides: Analyzing the NYC Bike Share System
- OpenAI Gym: https://gym.openai.com/
- RL Lib: https://docs.ray.io/en/latest/rllib/index.html

Backup Slides

Bike Repositioning Problem

- Bikesharing is popular and convenient

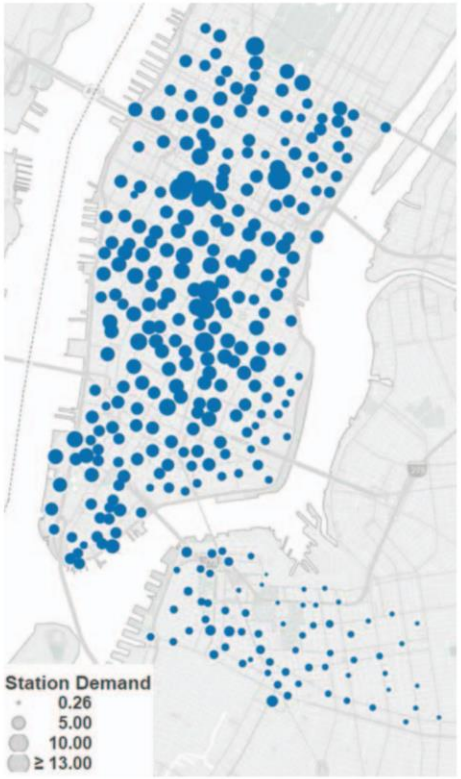


https://bikesharemap.com/?city=washingtondc#/3.3219280948873626/-126.6562/33.5066/

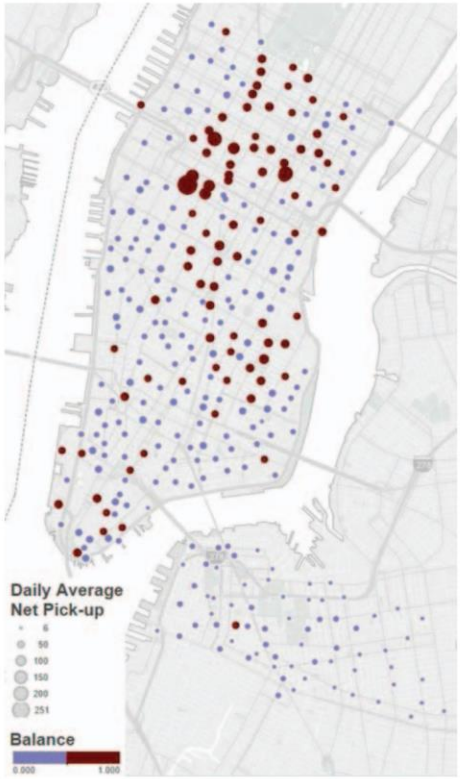
Bike Repositioning Problem

- Imbalanced demand over space

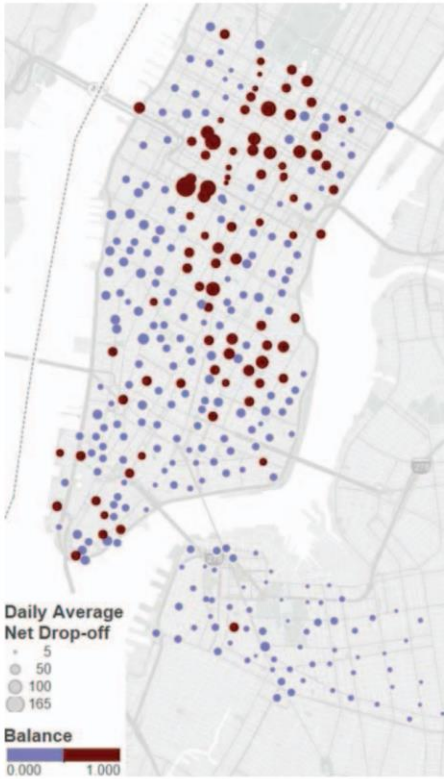
(a) Station Voronoi Region



(b) Demand Distribution

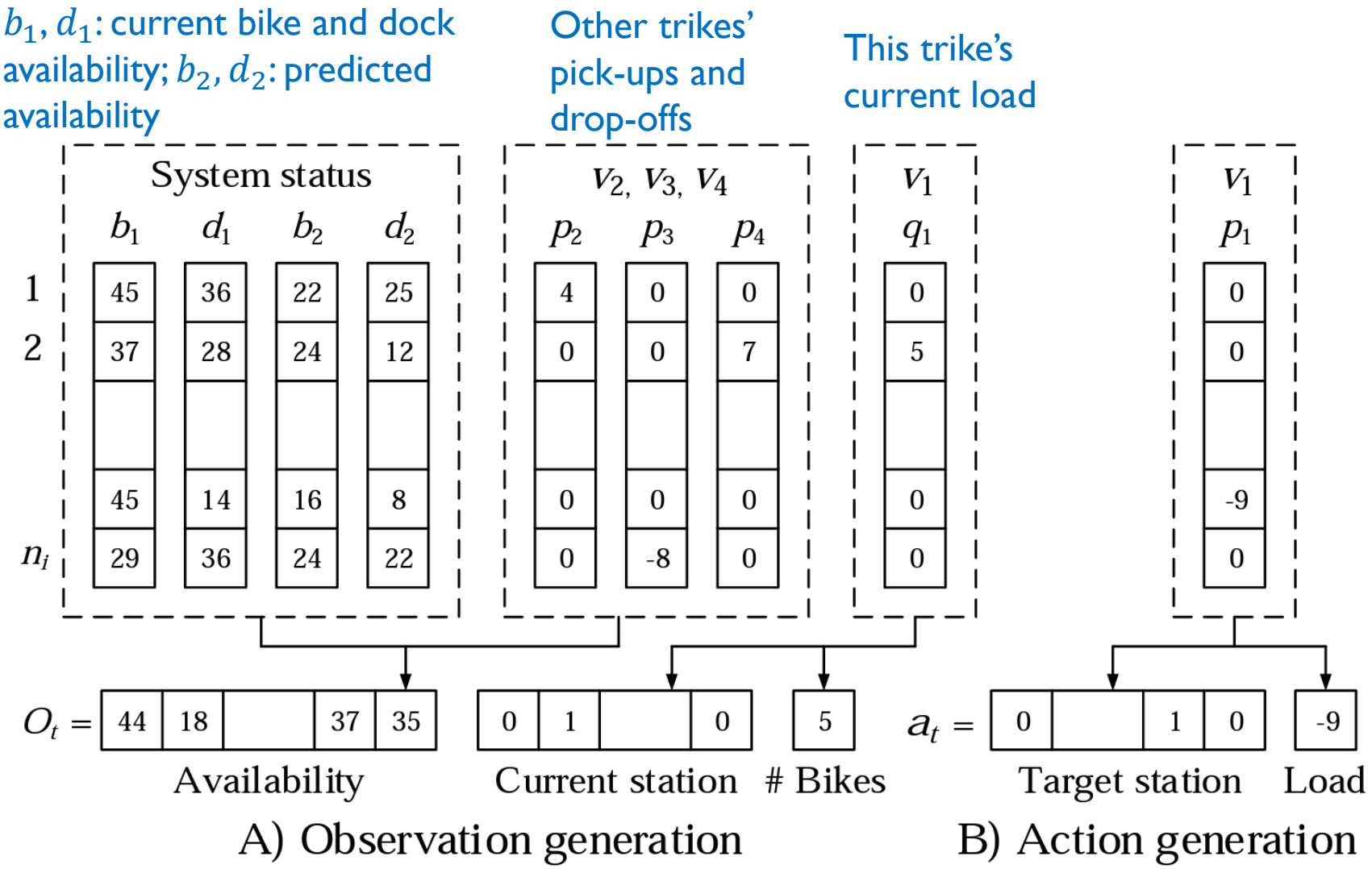


(c) Net-pick-up Distribution



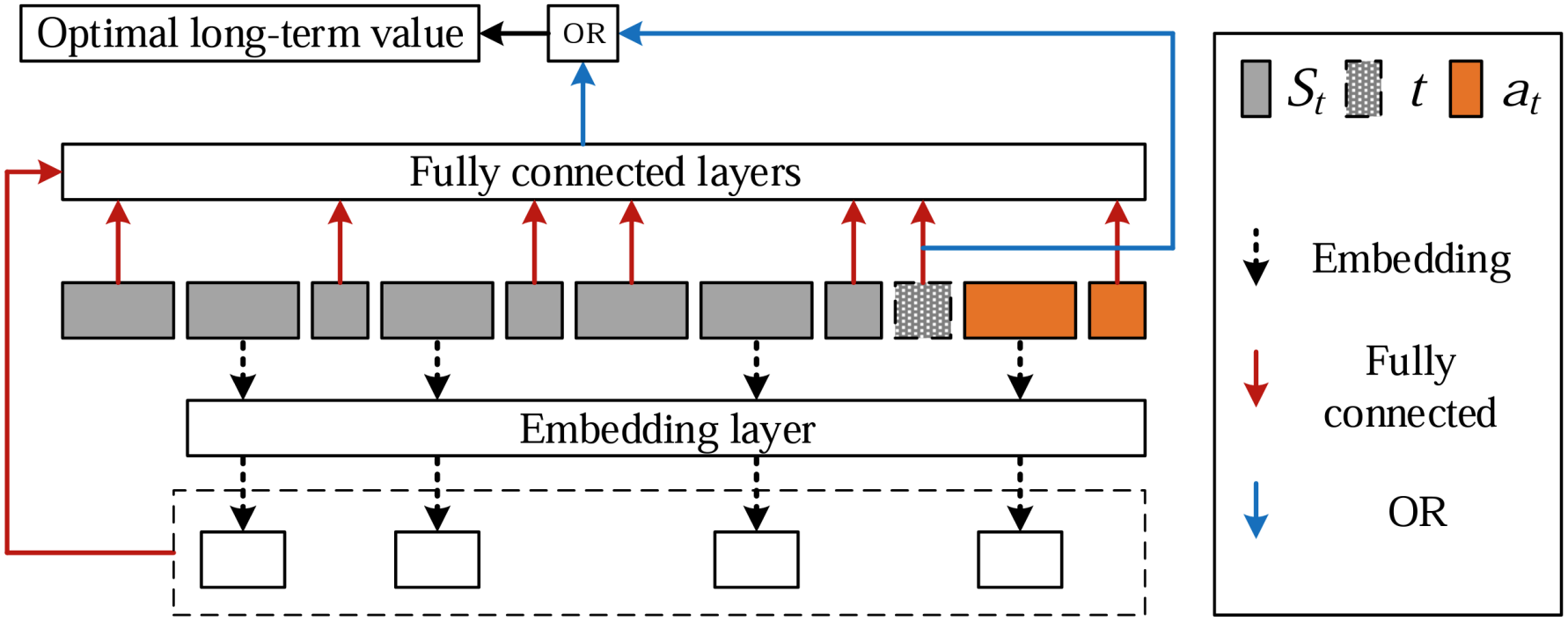
(d) Net-drop-off Distribution

Observation and Action Representation (one step)



Value Network Architecture

- Optimal long-term value network



Poll 2

- Let $L_i = 1$. Let n_i be the number of regions. How many numbers are needed to represent a state with this representation $s_t = (O_{t-L_i}, a_{t-L_i}, \dots, O_{t-1}, a_{t-1}, O_t, t)$?
  - A: $9n_i$
  - B: $8n_i + 1$
  - C: $3n_i + 2$
  - D: $3n_i + 3$
  - F: None of the above
  - G: I don't know