Week 11: Adversarial Reinforcement Learning

COMP90073
Security Analytics

Yi Han, CIS

Semester 2, 2021

Overview

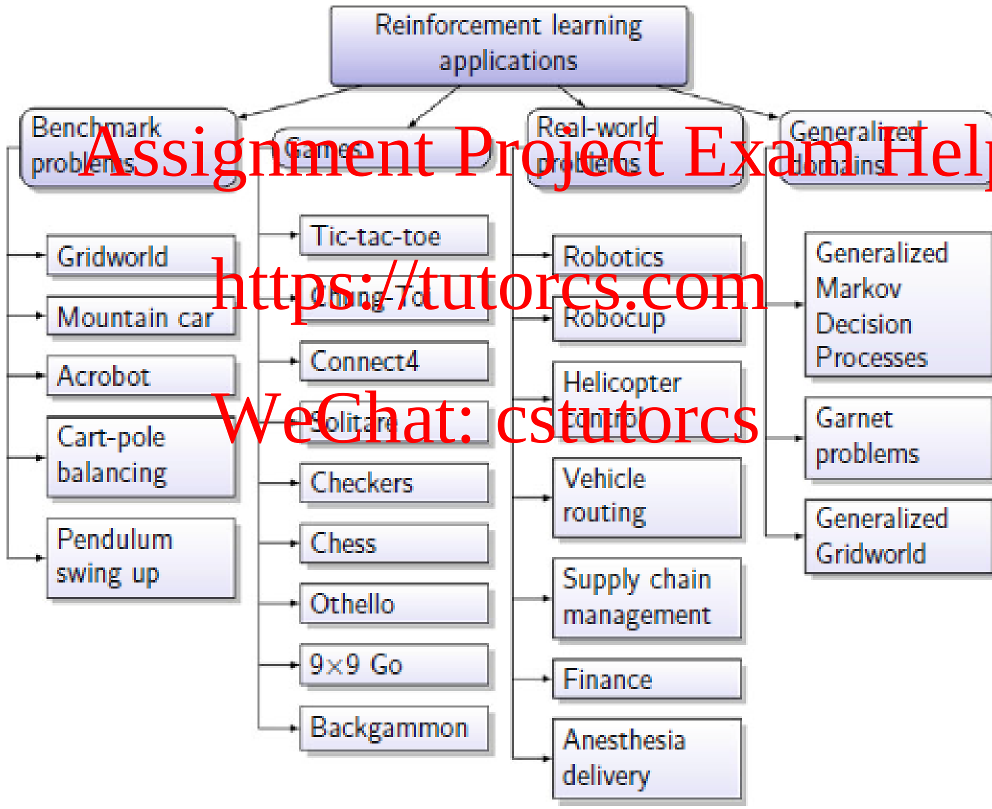
- Background on reinforcement learning
  - Introduction
  - Q-learning
  - Application in defending against DDoS attacks
- Adversarial attacks against RL models
  - Test time attack
  - Training time attack
- Defence

Overview

- Background on reinforcement learning
  - Introduction
  - Q-learning
  - Application in defending against DDoS attacks
- Adversarial attacks against RL models
  - Test time attack
  - Training time attack
- Defence

Background on reinforcement learning

- Application



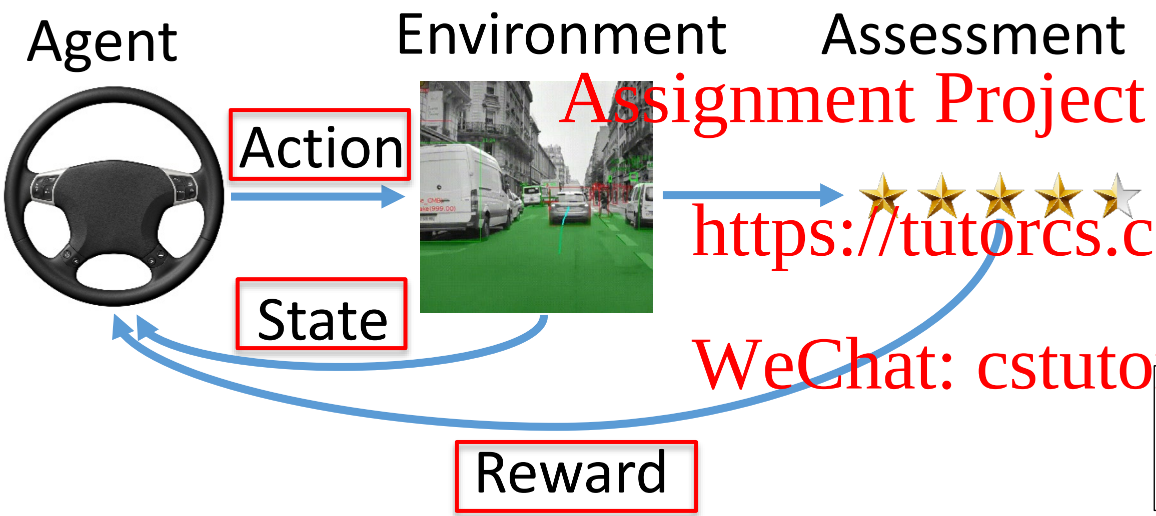
https://www.youtub

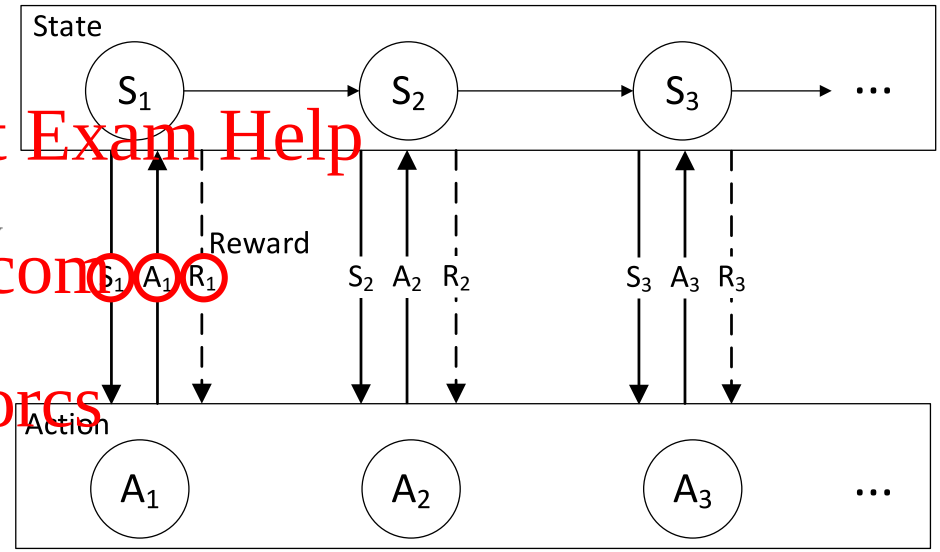
ps/autopilot-self-driving-
ng

https://www.myrealfacts.com/2019/05/applications-of-reinforcement-learning.html

Background on reinforcement learning

- Introduction

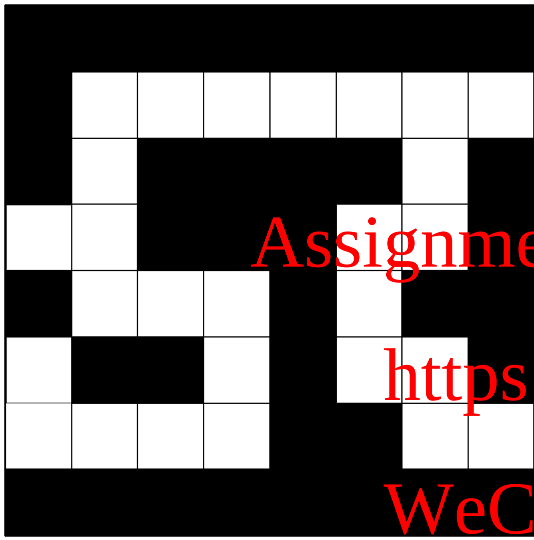


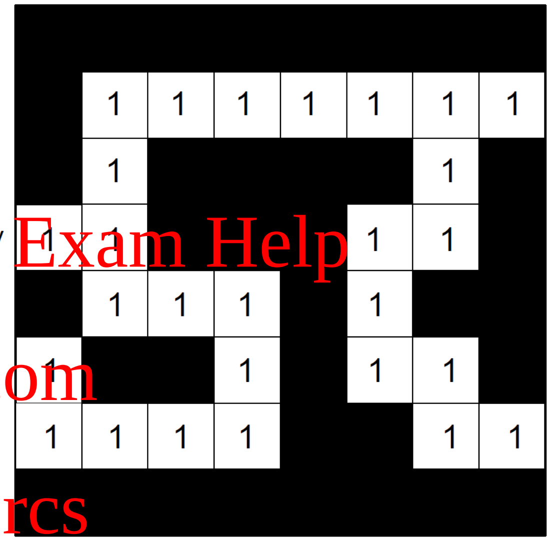


Maximise the discounted cumulative rewards over the long run: $R_t = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau , \gamma \in (0, 1]$

Background on reinforcement learning

- State

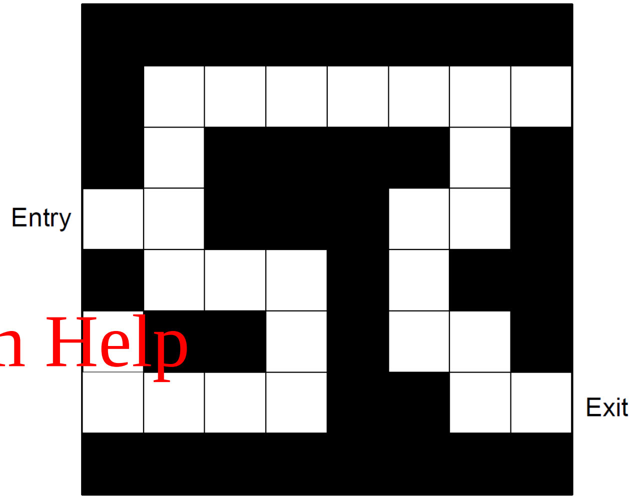




$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 2 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 2 & 0 & 0 & 0 & 0 & 1 & 0 \\ 2 & 2 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Background on reinforcement learning

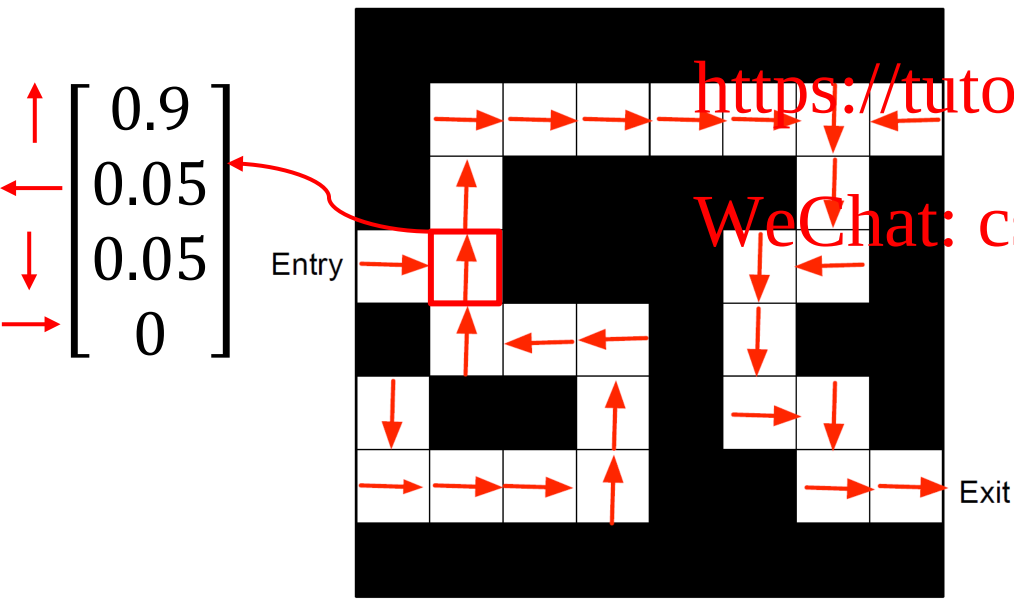
- Action
  - Up
  - Left
  - Down
  - Right

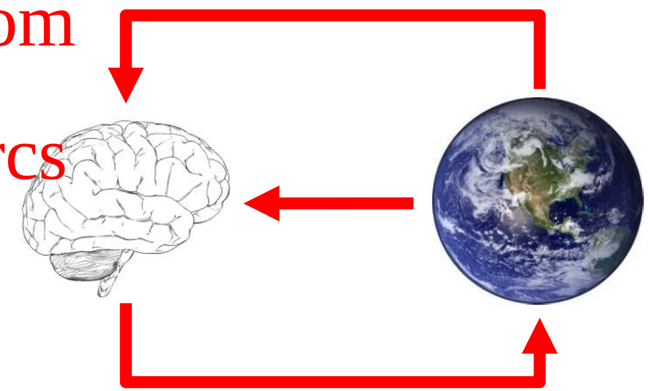


- Reward: an immediate feedback on whether an action is good
  - In the range of [-1, 1]
  - 1: reach the exit
  - -0.8: move to a blocked cell
  - -0.3: move to a visited cell
  - -0.05: move to an adjacent cell

Background on reinforcement learning

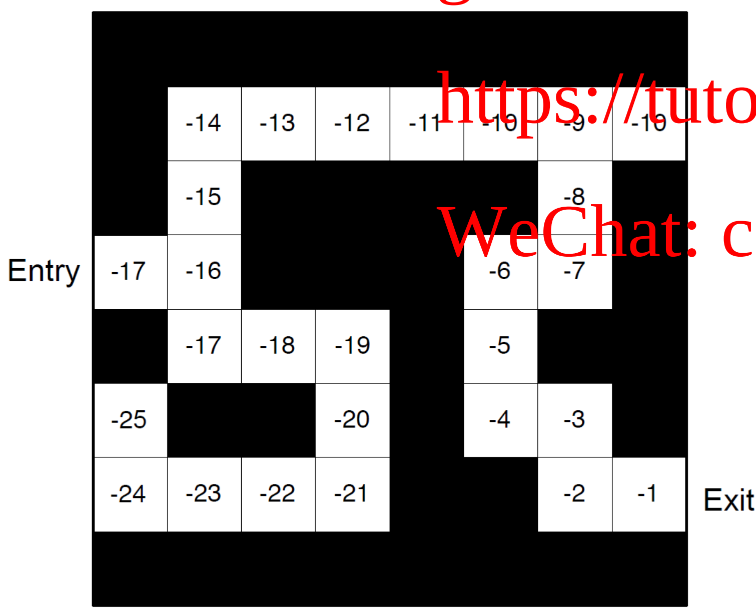
- Policy (π): a mapping from state to action, i.e. $a = \pi(s)$, it tells the agent what to do in a given state

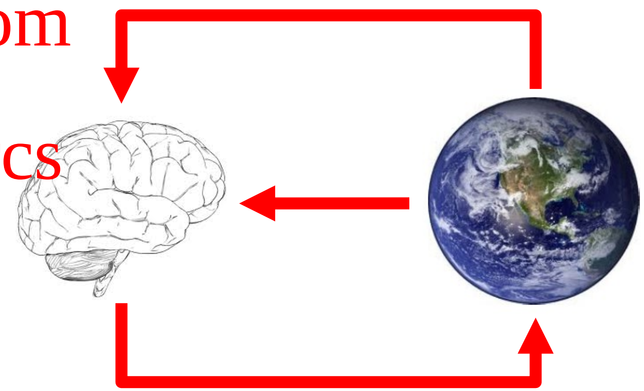




Background on reinforcement learning

- Value function: the future, long term reward of a state
  - Value of state s under policy π: $V^{\pi}(s) = \mathbb{E}\left[\sum_{i=1}^{T} \gamma^{i-1} r_i | S_t = s\right]$
  - Conditional on some policy π
  - Expected value of following policy π starting from state s

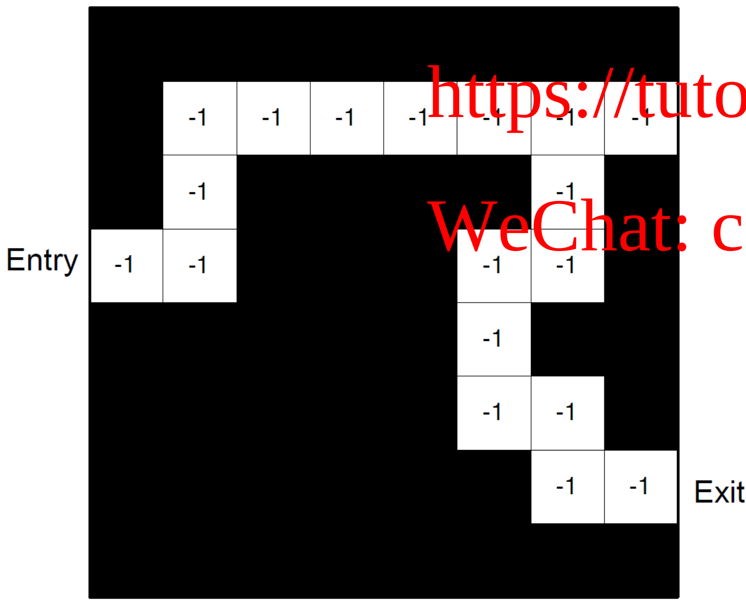


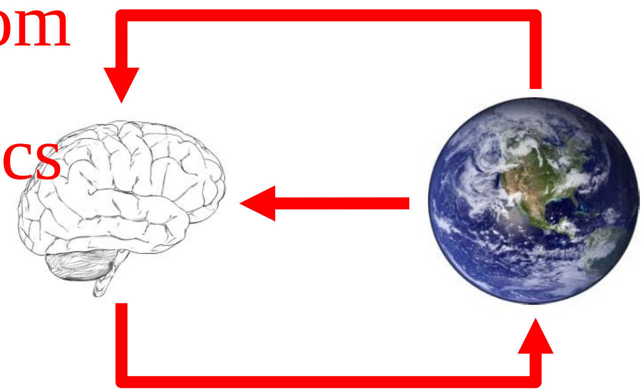


THE UNIVERSITY OF MELBOURNE

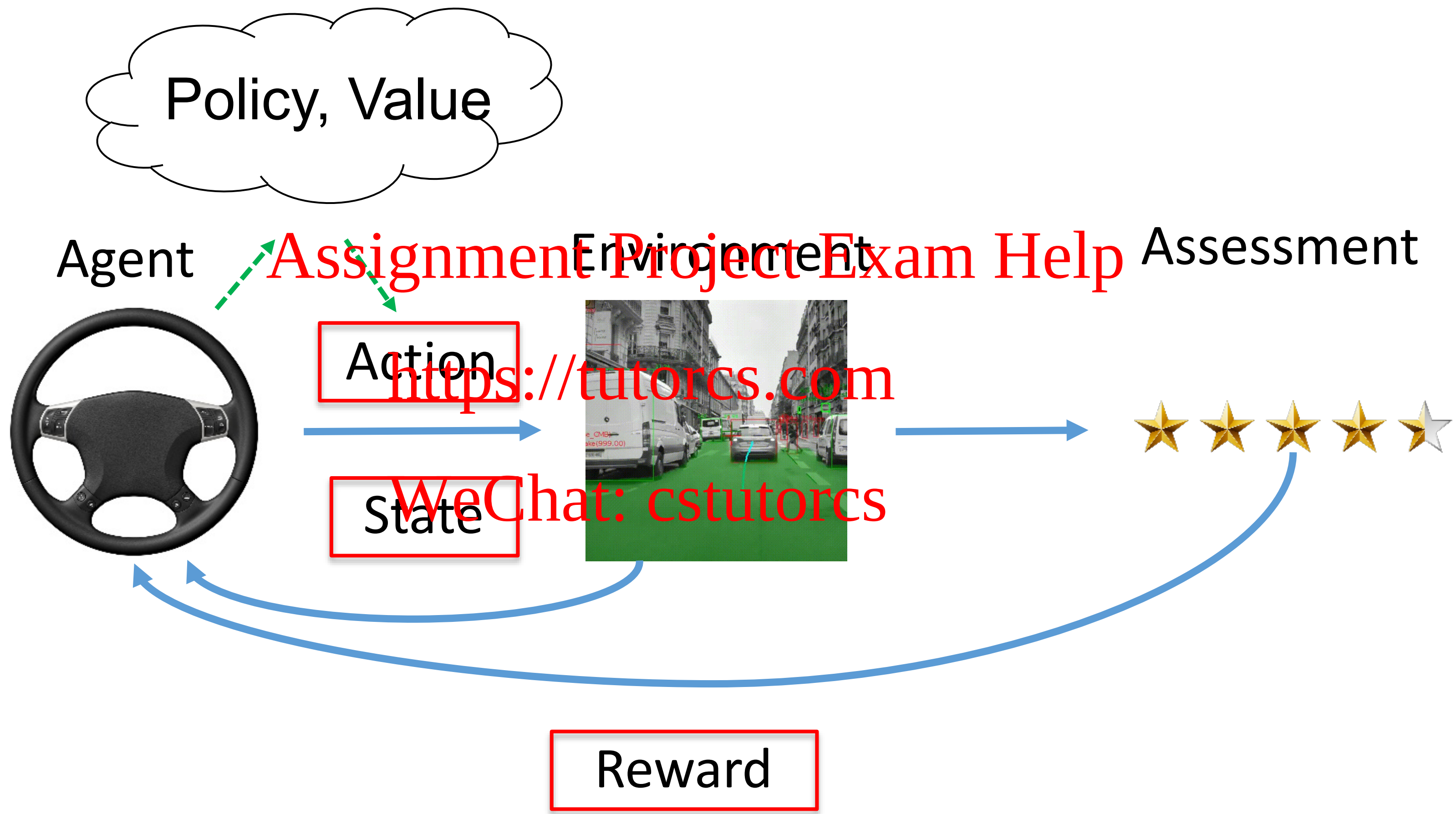
Background on reinforcement learning

- Model of the environment: mimic the behaviour of the environment, e.g., given a state & action, what the next state & reward might be.



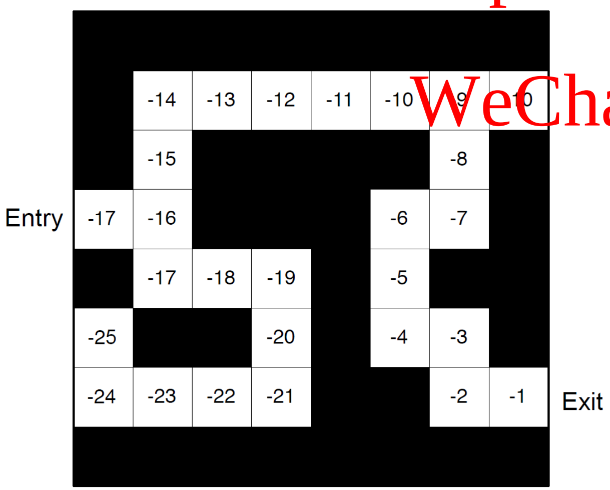


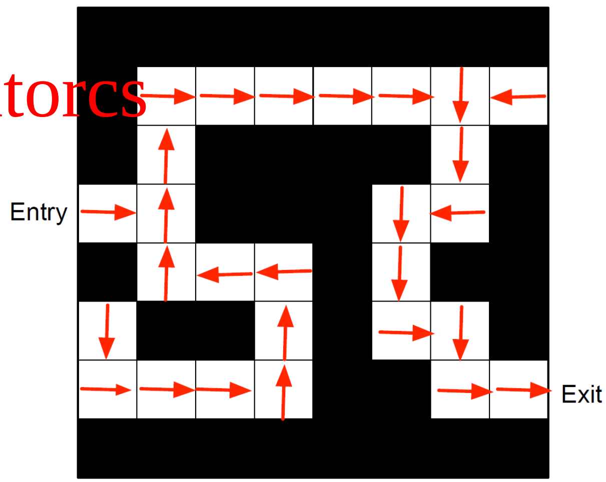
Background on reinforcement learning



Background on reinforcement learning

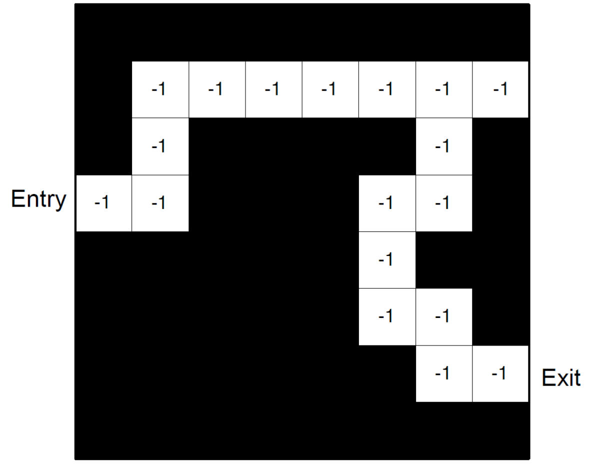
- Classification
  - Value-based algorithm estimates the value function
  - Policy-based algorithm learns the policy directly
  - Actor-critic: critic updates action-value function, actor updates policy





Background on reinforcement learning

- Classification
  - Model free algorithm directly learns the policy and/or the value function
  - Model based algorithm first builds up how the environment works



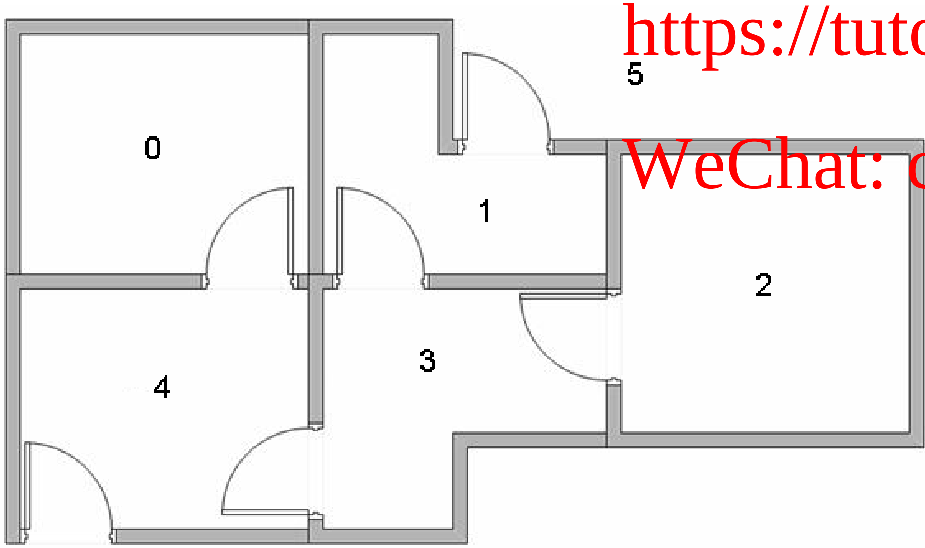
Overview

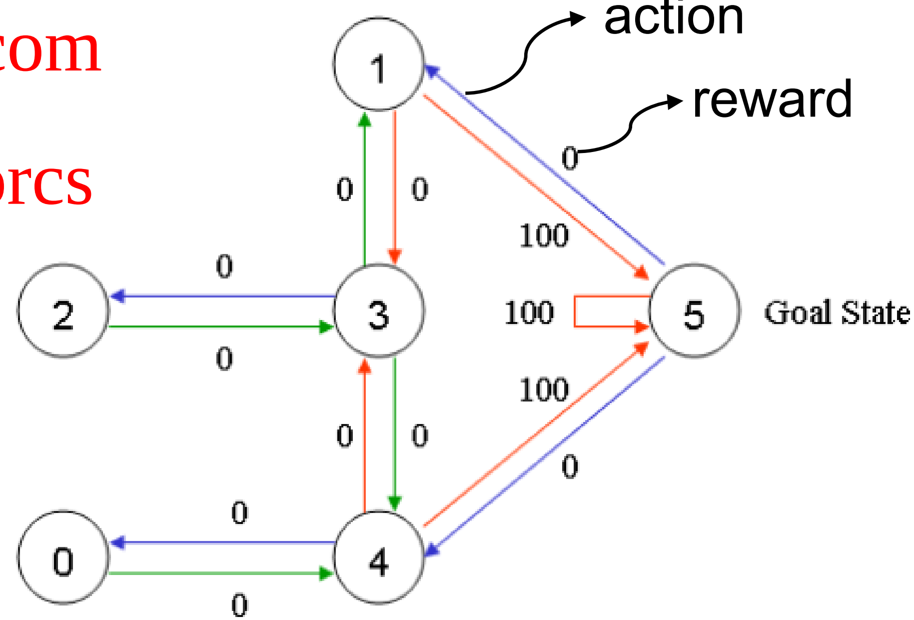
- Background on reinforcement learning
  - Introduction
  - Q-learning
  - Application in defending against DDoS attacks
- Adversarial attacks against RL models
  - Test time attack
  - Training time attack
- Defence

Background on reinforcement learning

- Q-learning: estimate action-value function $Q(s,a)$
  - Expected value of taking action a in state s and then following policy π:

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{i=1}^{T} \gamma^{i-1} r_i \mid S_t = s, A_t = a\right]$$





http://mnemstudio.org/path-finding-q-learning-tutorial.htm

Background on reinforcement learning

- Q-learning

Action

$$
R = \begin{array}{c|cccccc}
\text{State} & 0 & 1 & 2 & 3 & 4 & 5 \\
\hline
0 & -1 & -1 & -1 & -1 & 0 & -1 \\
1 & -1 & -1 & -1 & 0 & -1 & \boxed{100} \\
2 & -1 & -1 & -1 & 0 & -1 & -1 \\
3 & -1 & \boxed{0} & 0 & -1 & 0 & -1 \\
4 & 0 & -1 & -1 & 0 & -1 & 100 \\
5 & -1 & 0 & -1 & -1 & 0 & 100
\end{array}
$$

$$
Q = \begin{array}{c|cccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 \\
\hline
0 & 0 & 0 & 0 & 0 & 400 & 0 \\
1 & 0 & 0 & 0 & 320 & 0 & 500 \\
2 & 0 & 0 & 0 & 320 & 0 & 0 \\
3 & 0 & 400 & 256 & 0 & 400 & 0 \\
4 & 320 & 0 & 0 & 320 & 0 & 500 \\
5 & 0 & 400 & 0 & 0 & 400 & 500
\end{array}
$$

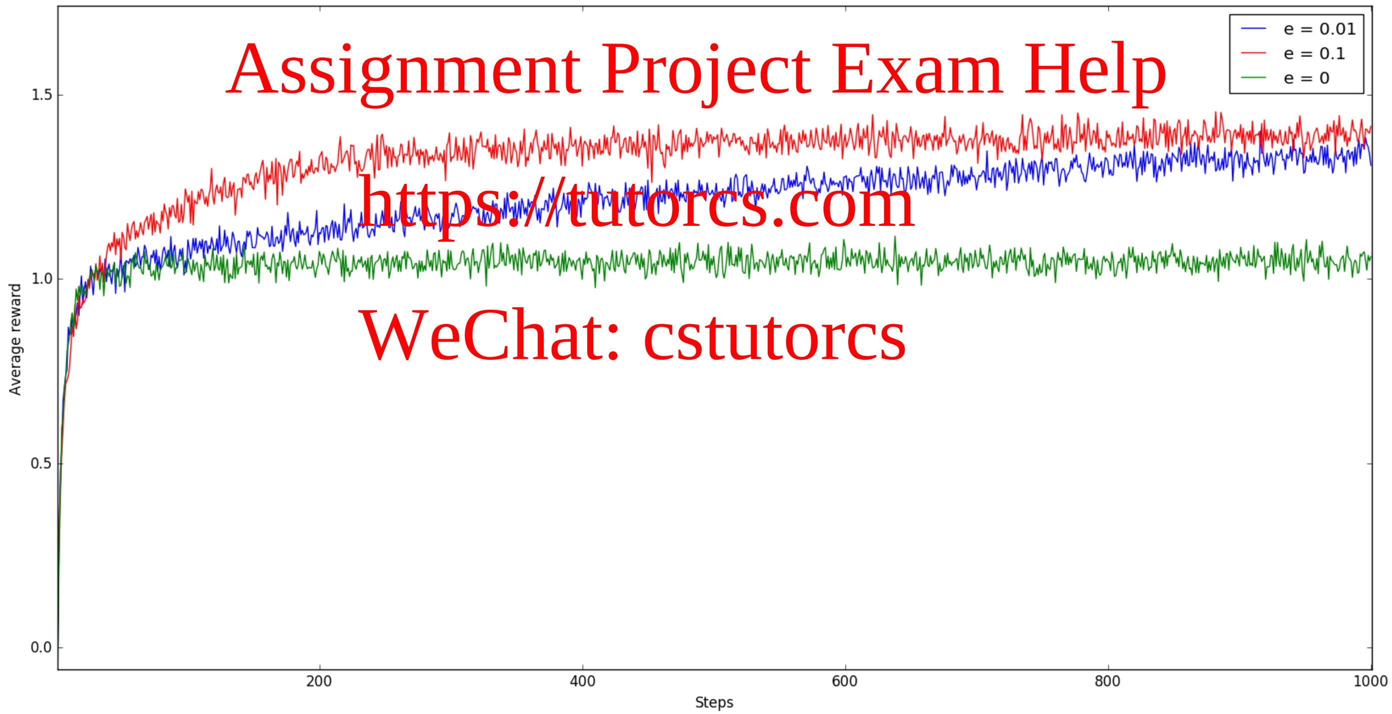
100+0.8*max(0,0,0)

0+0.8*max(0,100)

$$Q(s_t, a_t) \leftarrow \underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}$$

Background on reinforcement learning

- Exploitation vs. Exploration
  - ε-greedy



Background on reinforcement learning

- The tabular version does not scale with the action/state space
- Classical Q Network [1]
  - Function approximation
  - Approximate $Q(s,a)$ via a neural network: $Q(s,a) \approx Q^*(s,a,\theta)$
  - $L(\theta) = \mathbb{E}\left[\left(\underbrace{r + \gamma \boxed{\max_{a'} Q(s', a'; \theta)}}_{\text{Target Q}} - Q(s, a; \theta)\right)^2\right]$
  - Unstable

Background on reinforcement learning

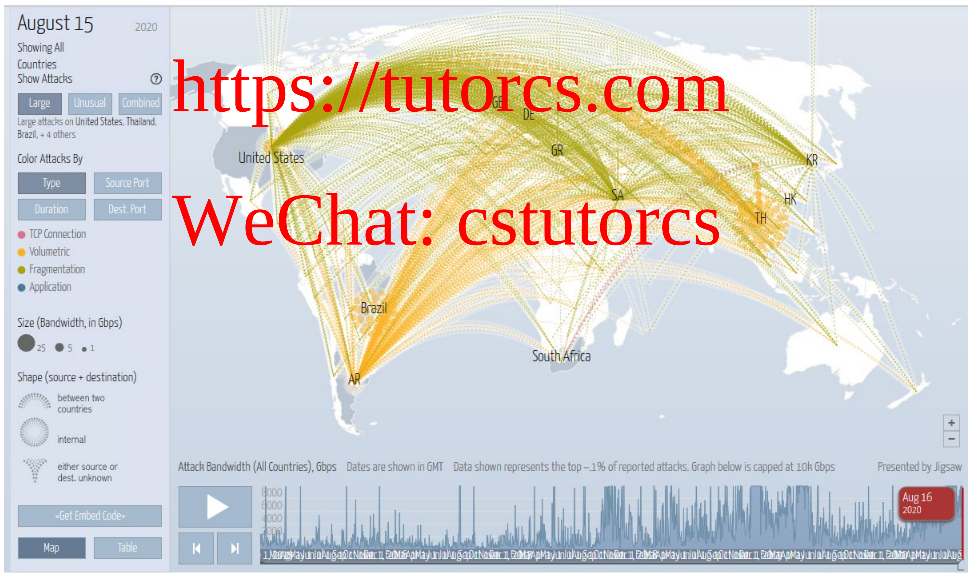
- Deep Q Network (DQN) [2]
  - Experience replay: draw randomly from a buffer of (*s*, *a*, *s*', *r*)
  - $Q(s,a) \leftarrow r + \gamma max_{a'} Q(s',a',\theta^-)$ (*s*', *a*', *s*'', *r*')
  - $L(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta^-) - Q(s,a;\theta)\right)^2\right]$
  - Reward clipped to [-1, 1]
- Double DQN (DDQN) [3]
  - Separate action selection from action evaluation
  - $Q_1(s,a) \leftarrow r + \gamma Q_2(s', argmax_{a'} Q_1(s',a'))$
  - $L(\theta) = \mathbb{E}\left[\left(r + \gamma\, Q_2(s', \arg\max_{a'} Q_1(s',a')) - Q_1(s,a;\theta)\right)^2\right]$

Overview

- Background on reinforcement learning
  - Introduction
  - Q-learning
  - Application in defending against DDoS attacks
- Adversarial attacks against RL models
  - Test time attack
  - Training time attack
- Defence

Background on reinforcement learning

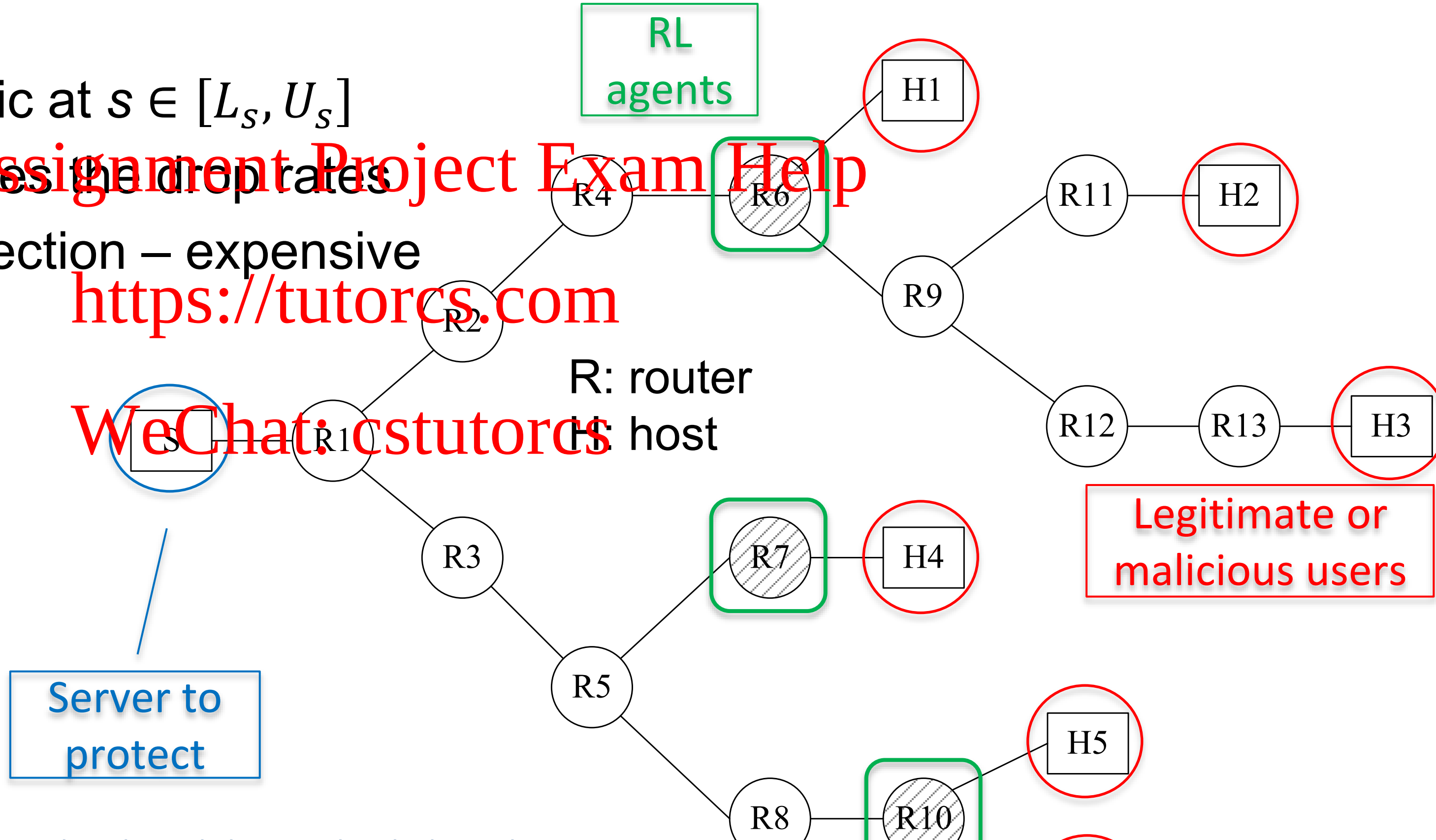
- Distributed Denial-of-Service (DDoS) attacks still occur almost every hour globally
  - http://www.digitalattackmap.com/
  - Statistics are gathered by Arbor’s Active Threat Level Analysis System from 330+ ISP customers with 130Tbps of global traffic



- Can RL be applied to throttle flooding DDoS attacks?

Background on reinforcement learning

- Problem setup [5]
  - A mixed set of legitimate users & attackers
  - Aggregated traffic at $s \in [L_s, U_s]$
  - RL agents decides the drop rates
  - No anomaly detection – expensive

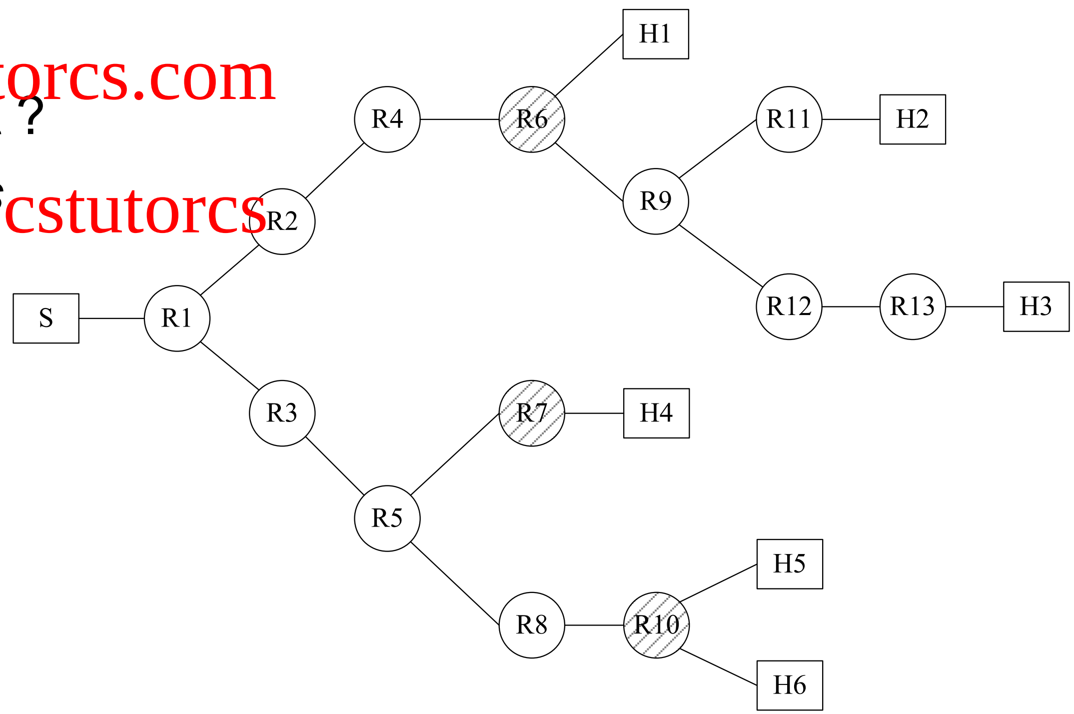


Kleanthis Malialis, Daniel Kudenko, Multiagent Router Throttling: Decentralized Coordinated Response Against DDoS Attacks, In Proc. of AAAI 2013

THE UNIVERSITY OF MELBOURNE

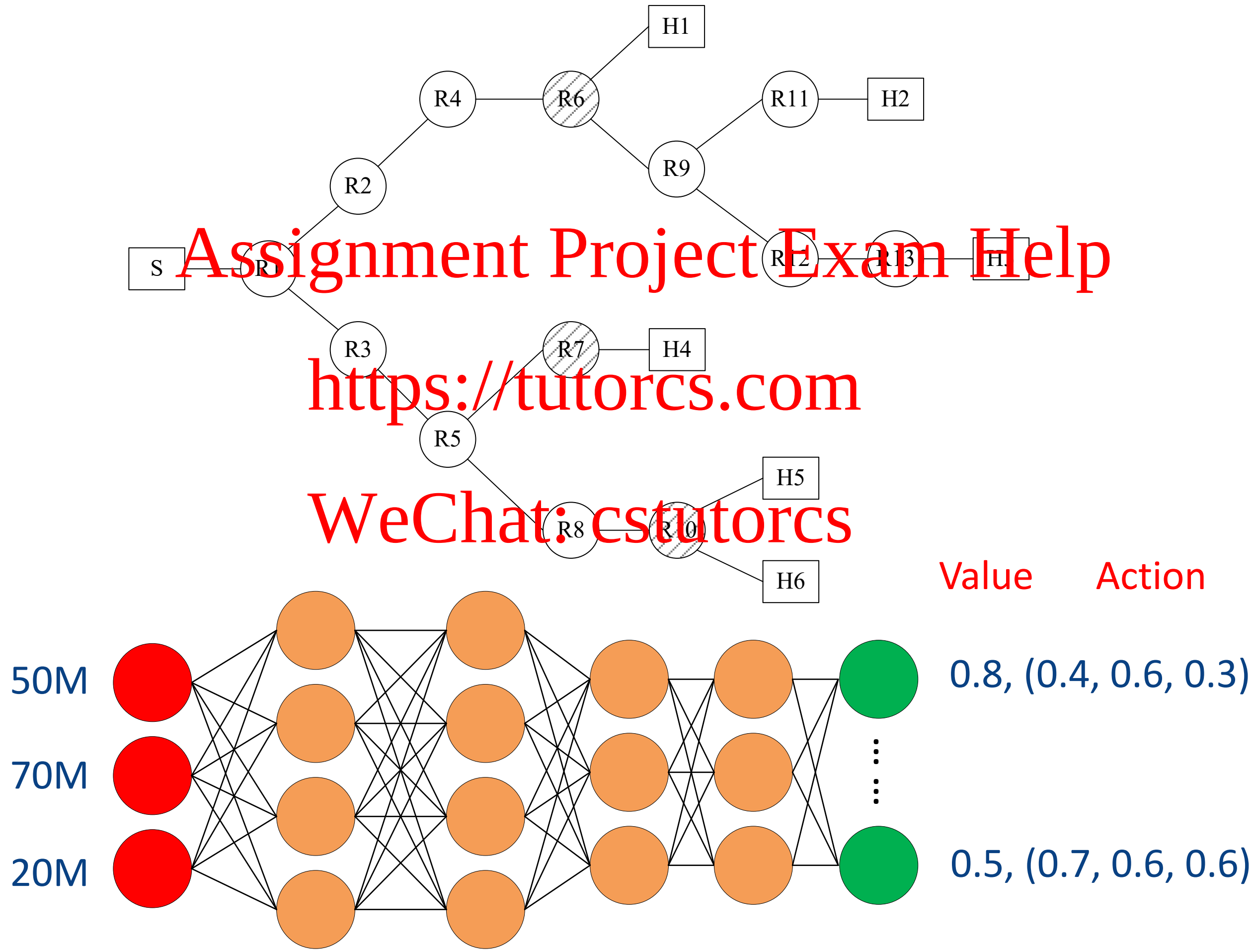
Background on reinforcement learning

- RL problem formalisation
  - State space
    - Aggregated traffic arrived at the router over the last T seconds
  - Action set
    - Percentage of traffic to drop: 0, 10%, 20%, 30%, … 90%
  - Reward
    - Aggregated traffic at $s > U_s$?
    - Legitimate traffic reached s



Background on reinforcement learning

- Training (DDQN)

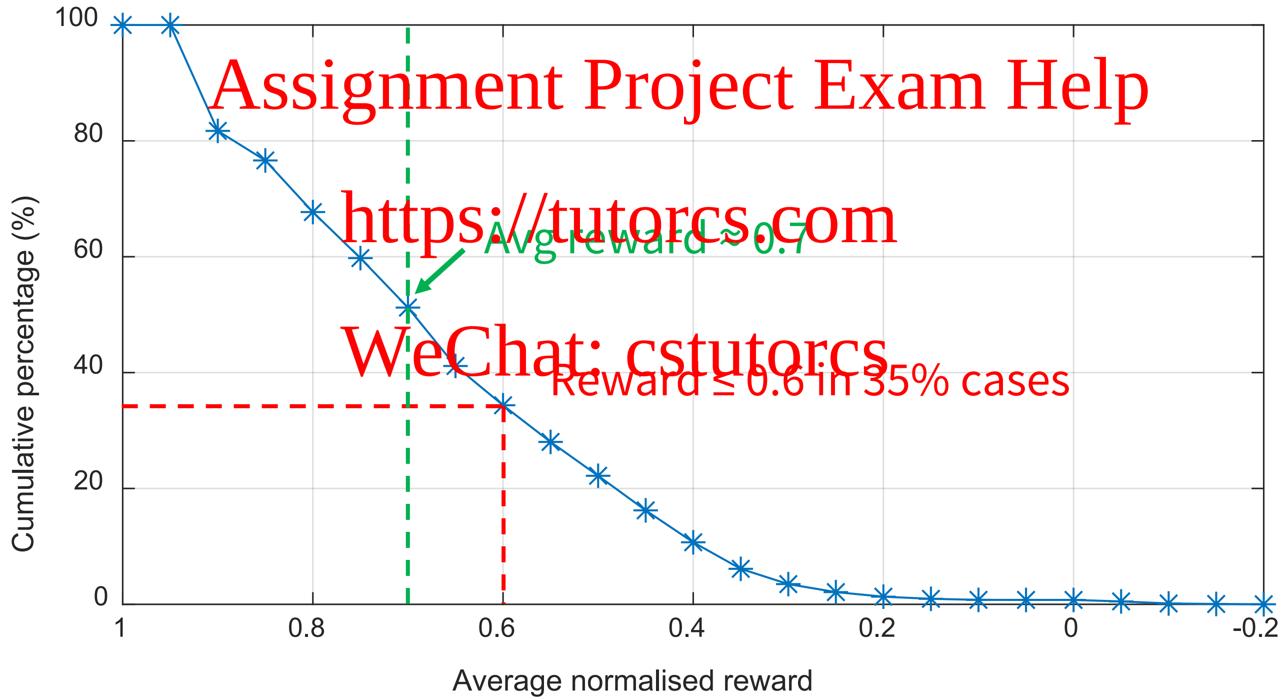


Background on reinforcement learning

$$L(\theta) = \mathbb{E}\left[\left(r + \gamma\, Q_2(s', \arg\max_{a'} Q_1(s', a')) - Q_1(s, a; \theta)\right)^2\right]$$

Background on reinforcement learning

- Test
  - 10000 cases (may not be seen in training)

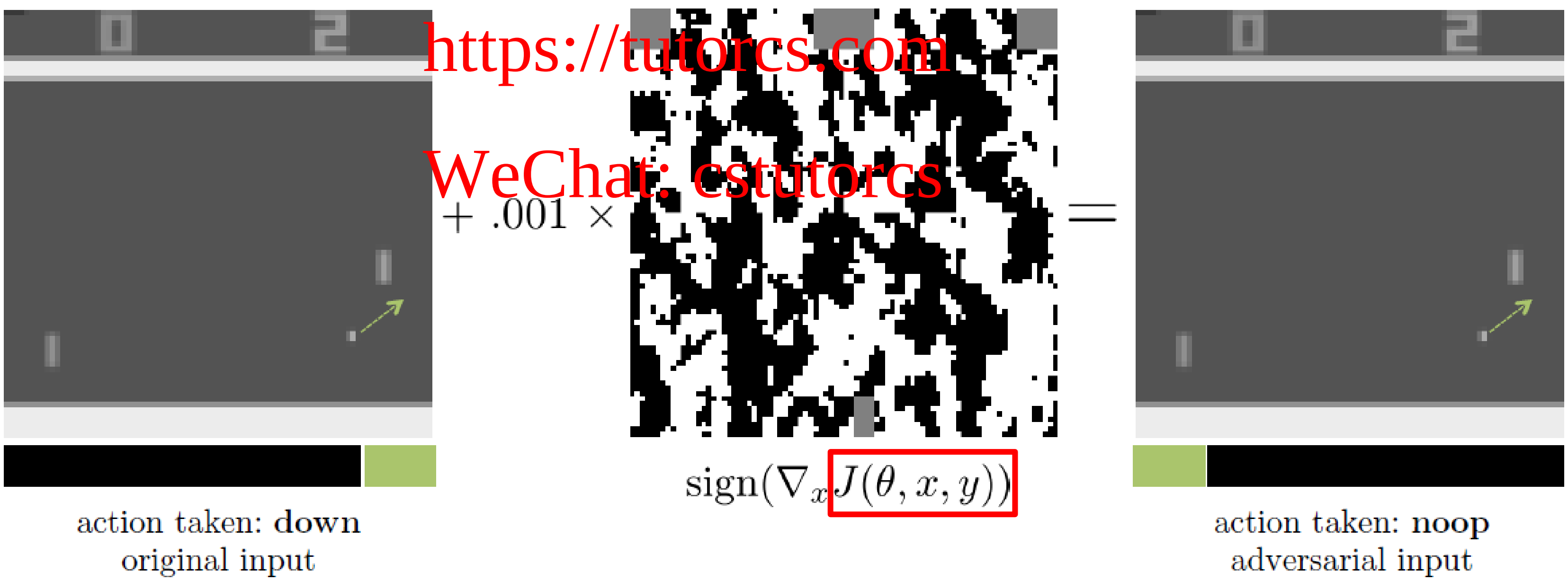


Overview

- Background on reinforcement learning
  - Introduction
  - Q-learning
  - Application in defending against DDoS attacks
- Adversarial attacks against RL models
  - Test time attack
  - Training time attack
- Defence

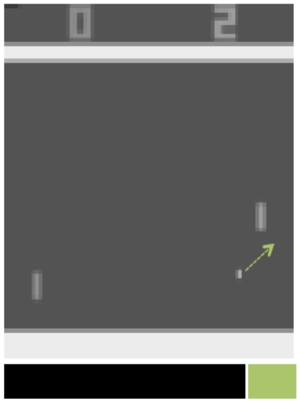
Adversarial attacks against RL models

- Test time attacks
  - Manipulate the environment observed by the agent [5]
  - Without attack: $\ldots, s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, \ldots$
  - With attack: $\ldots, s_t + \delta_t, a'_t, r'_t, s_{t+1} + \delta_{t+1}, a'_{t+1}, r'_{t+1}, s_{t+2} + \delta_{t+2}, \ldots$



Adversarial attacks against RL models

- $J(\theta, x, y)$
  - *y*: softmax of the Q-value, i.e., prob. of taking an action



$\rightarrow$ RL agent (θ) $\rightarrow$

$$\begin{pmatrix} 1.5 \\ 2.2 \\ \ldots \\ 0.4 \\ 3.6 \end{pmatrix} \xrightarrow{\text{Softmax}} \begin{pmatrix} 0.07 \\ 0.15 \\ \ldots \\ 0.02 \\ 0.61 \end{pmatrix}$$

$$\pi_i = \frac{e^{Q(s,a_i)}}{\sum_{a_j} e^{Q(s,a_j)}}$$

Q-value for each action

Prob.

  - *J*: cross-entropy loss between *y* and the distribution that places all weight on the action with the highest Q-value

$$y \begin{pmatrix} 0.07 \\ 0.15 \\ \ldots \\ 0.02 \\ \boxed{0.61} \end{pmatrix} \longleftrightarrow \begin{pmatrix} 0 \\ 0 \\ \ldots \\ 0 \\ \boxed{1} \end{pmatrix}$$

similar to the one-hot vector in supervised learning

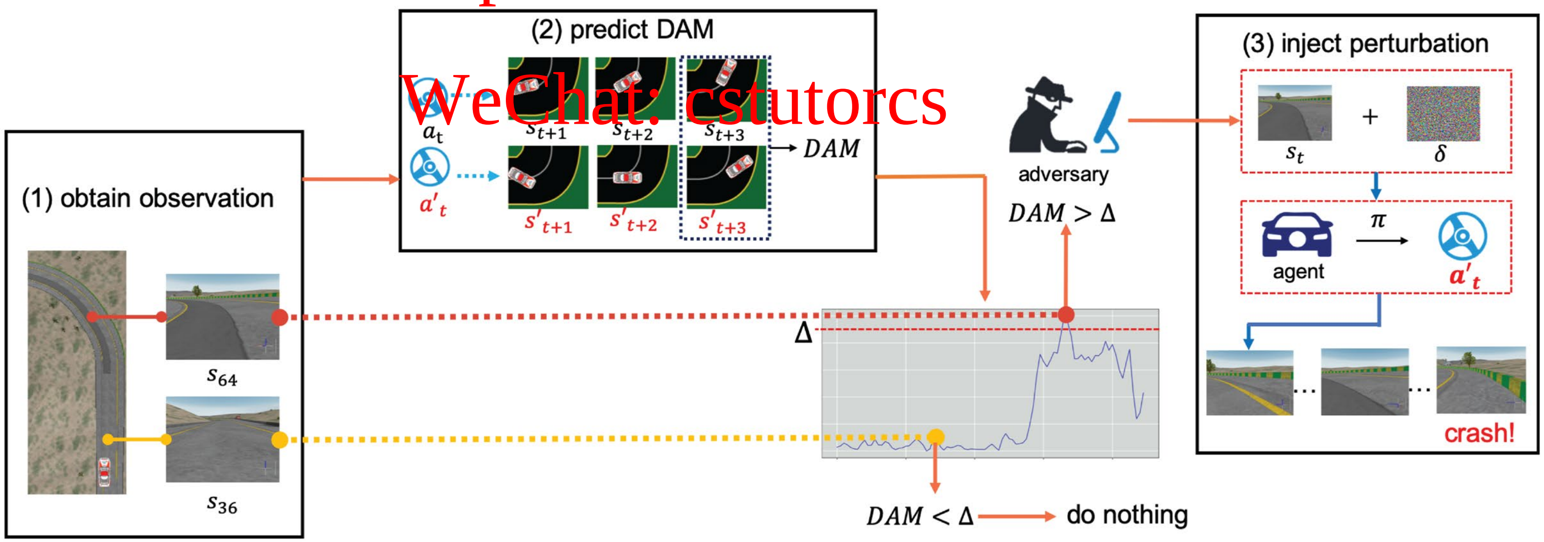
Adversarial attacks against RL models

- Timing of the attack
  - Heuristic method [6]: launch the attack only when

$$c(s) = \max_a \frac{e^{\frac{Q(s,a)}{T}}}{\sum_{a_k} e^{\frac{Q(s,a_k)}{T}}} - \min_a \frac{e^{\frac{Q(s,a)}{T}}}{\sum_{a_k} e^{\frac{Q(s,a_k)}{T}}} > \beta$$

Adversarial attacks against RL models

- Timing of the attack [8]
  - "Brute-force" search
    - Consider all possible *N* consecutive perturbations
    - Evaluate the attack damage at step $t + M$ ($M \geq N$)
    - $s_t, a_t, s_{t+1}, a_{t+1}, \dots, s_{t+N-1}, \dots, s_{t+M}$



Adversarial attacks against RL models

- Timing of the attack [8]
  - "Brute-force" search
    - Train a prediction model: $(s_t, a_t) \rightarrow s_{t+1}$
    - Predict the subsequent states and actions, $\{(s_t, a_t), (s_{t+1}, a_{t+1}), \dots (s_{t+M}, a_{t+M})\}$
    - Assess the potential damage of all possible strategies
      - Danger Awareness Metric (DAM):

        $DAM = |T(s'_{t+M}) - T(s_{t+M})|$

        T: domain-specific definition, e.g., distance between the car and the centre of the road

Adversarial attacks against RL models

- Timing of the attack [8]
  - Train an antagonist model
    - Learn the optimal attack strategy automatically without any domain knowledge
    - Maintain a policy: $s_t \rightarrow (p_t, a'_t)$
      - If $p_t > 0.5$, add the perturbation to trigger a'_t
      - Take the original action a_t
    - Reward: negative of the agent's reward

Adversarial attacks against RL models

- Black-box attack [9]
  - Train a proxy model that learns a task that is *related* to the target agent's policy
  - S threat model
    - Only have access to the states
    - Approximate an expectation of the state transition function
    - $psychic(s_t, \theta_P) \approx \mathbb{E}_{\pi_T}[P(s_{t+1}|s_t)]$
  - SR threat model
    - Have access to the states and reward
    - Estimate the value *V* of a given state under the policy π_T
    - $assessor(s_t, \theta_A) \approx \mathbb{E}_{\pi_T}\left[\sum_{k=0}^{\infty} \gamma_t{}^{(k)}\, r_{t+k+1}\right] = V^{\pi_T}(s_t)$

Adversarial attacks against RL models

- Black-box attack [9]
  - SA threat model
    - Have access to states and actions
    - Approximate the target's policy π_T
    - $imitator(s_t, \theta_I) \approx \pi_T(s_t)$
  - SRA threat model
    - Have access to states, actions and rewards
    - Action-conditioned psychic (AC-psychic): $AC\text{−}psychic(s_t, \theta_P) \approx \mathbb{E}_{\pi_T}[P(s_{t+1}|s_t, a_t)]$
    - Combine $assessor$ and $AC\text{−}psychic$ to decide whether to perturb the state

Adversarial attacks against RL models

- Black-box attack [9]
  - SRA threat model

Algorithm 1: Strategically-timed snooping attack

Input: Trained assessor, trained AC-psychic, trained proxy $\mathcal{M}_\kappa$, trained target agent $\mathcal{T}$, β

$\mathcal{K} \in \{S, SR, SA\}$

for t = *1*, T **do**
    Initialize empty list q;
    foreach $a \in \mathcal{A}$ **do**
        Predict s^H_{t+1} with *AC-psychic*(s_t, a)
        Estimate V^H with *assessor*(s^H_{t+1})
        Append V^H to q;
    end
    $c(s_t) = \max\left[\mathrm{Softmax}(q)\right] - \min\left[\mathrm{Softmax}(q)\right]$
    if $c(s_t) \geq \beta$ **then**
        Perturb s_t using $\nabla_x J_{\mathcal{M}_\kappa}$
    end
    Feed s_t to target $\mathcal{T}$ for action decision;
end

Adversarial attacks against RL models

- Black-box attack [9]
  - Surrogate: assume the adversary has access to the target agent's environment and can train an identical model

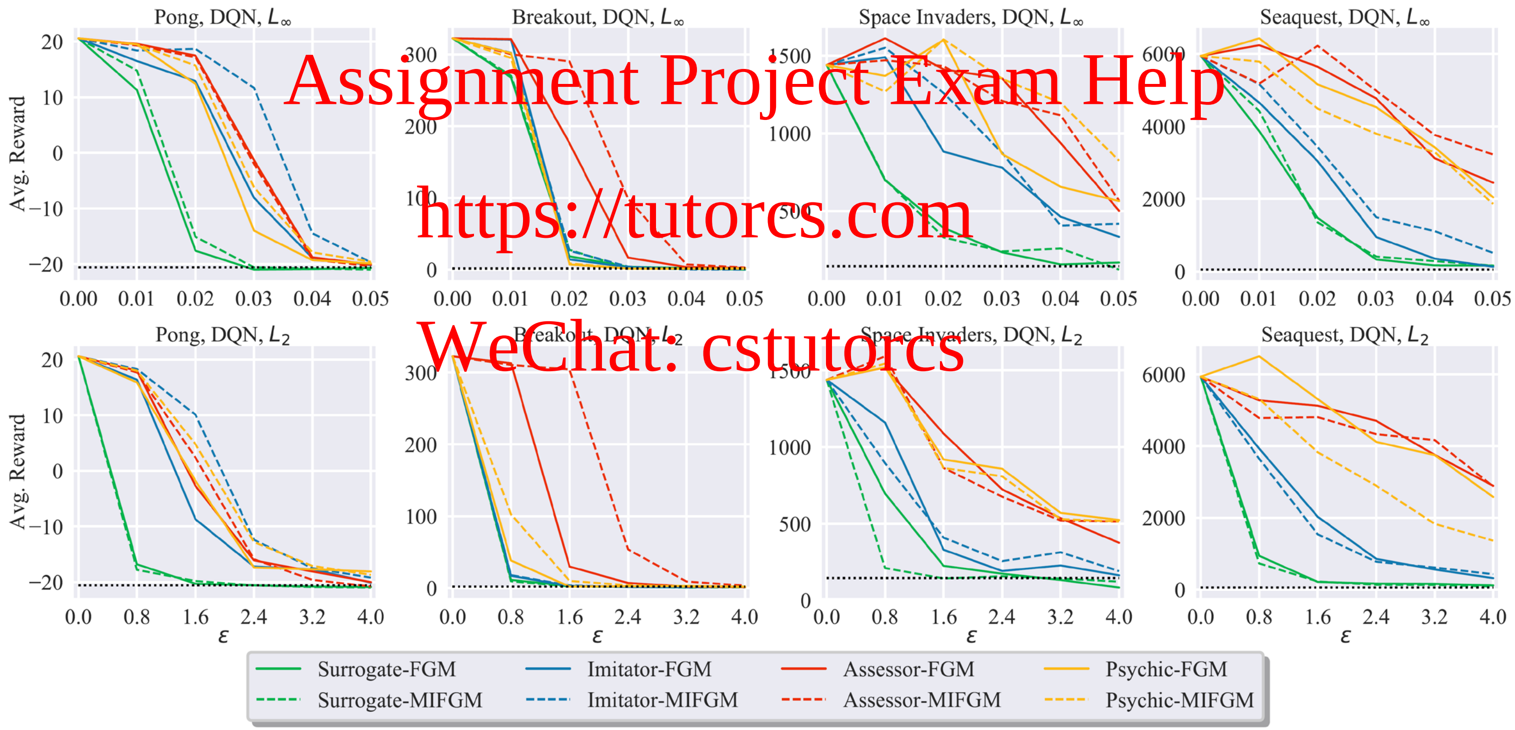


Figure 5: Performance reduction of DQN agents due to L_∞ and L_2 bounded perturbations. The black dotted line represents a random-guess policy.

Overview

- Background on reinforcement learning
  - Introduction
  - Q-learning
  - Application in defending against DDoS attacks
- Adversarial attacks against RL models
  - Test time attack
  - Training time attack
- Defence

Adversarial attacks against RL models

- Training time attack
  - Without attack: ..., $(s_t, a_t, s_{t+1}, r_t), (s_{t+1}, a_{t+1}, s_{t+2}, r_{t+1})$, ...
  - With attack: ..., $(s_t, a_t, s_{t+1} + \delta_{t+1}, r_t'), (s_{t+1} + \delta_{t+1}, a_{t+1}', s_{t+2} + \delta_{t+2}, r_{t+1}')$, ...
  - Purpose: generate δ_{t+1} so that the agent will not take the next action a_{t+1}
  - Cross entropy loss: $J = -\sum_i p_i log \pi_i$
    - $\pi_i = \frac{e^{Q(s,a_i)}}{\sum_{a_j} e^{Q(s,a_j)}}$ → prob. of taking action a_i
    - $p_i = \begin{cases} 1, \text{if } a_i = a_{t+1} \\ 0, \text{otherwise} \end{cases}$
    - Maximise $J = -log\pi_{t+1}$ → minimise the prob. of taking a_{t+1}
  - $\delta = \alpha \cdot Clip_\epsilon\left(\frac{\partial J}{\partial s}\right)$

What about targeted attacks?

Overview

- Background on reinforcement learning
  - Introduction
  - Q-learning
  - Application in defending against DDoS attacks
- Adversarial attacks against RL models
  - Test time attack
  - Training time attack
- Defence

Defence

- Adversarial training [7]
  - Calculate δ using the attacker's strategy: $(s_t, a_t, s_{t+1} + \delta_{t+1}, r'_t)$
  - $a'_{t+1} = \arg\max_a Q(s_{t+1} + \delta_{t+1}, a)$
  - Generate experience $(s_{t+1}, a'_{t+1}, s'_{t+2}, r'_{t+1})$ for the agent to train on

Untampered state

Potentially non-optimal action
→ explore more

Summary

- Reinforcement learning
  - State, action, reward
  - Value function, policy, model
  - Q-learning → Q-network → DQN → DDQN
- Adversarial reinforcement learning
  - Test time attack
    - Timing of the attack
    - Black-box attack
  - Training time attack
- Defence – adversarial training

References

- [1] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, First. Cambridge, MA, USA: MIT Press, 1998.
- [2] V. Mnih *et al.*, “Playing Atari with Deep Reinforcement Learning,” *CoRR*, vol. abs/1312.5602, 2013.
- [3] H. V. Hasselt, A. Guez, and D. Silver, “Deep Reinforcement Learning with Double Q-learning,” *eprint arXiv:1509.06461*, Sep. 2015.
- [4] K. Malialis and D. Kudenko, “Multiagent Router Throttling: Decentralized Coordinated Response Against DDoS Attacks,” in Proc. of the 27th AAAI Conference on Artificial Intelligence, Washington, 2013, pp. 1551–1556.
- **[5] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, “Adversarial Attacks on Neural Network Policies,” eprint arXiv:1702.02284, 2017.**
- [6] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, “Tactics of Adversarial Attack on Deep Reinforcement Learning Agents,” *eprint arXiv:1703.06748*, Mar. 2017.
- **[7] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, “Robust Deep Reinforcement Learning with Adversarial Attacks,” *arXiv:1712.03632 [cs]*, Dec. 2017.**

References

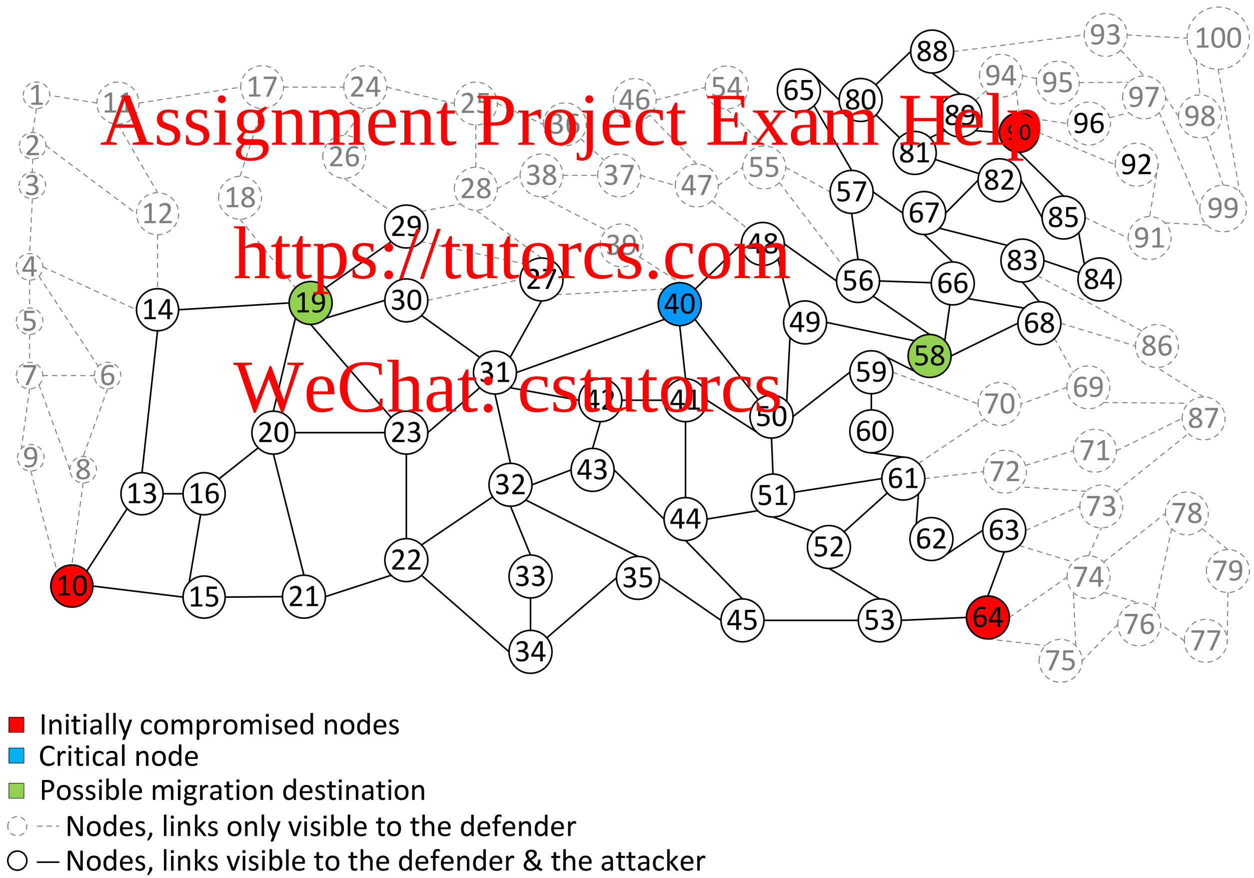
- [8] Jianwen Sun and Tianwei Zhang and Xiaofei Xie and Lei Ma and Yan Zheng and Kangjie Chen and Yang Liu, “Stealthy and Efficient Adversarial Attacks against Deep Reinforcement Learning,” AAAI 2020: 5883-5891
- [9] Matthew Inkawhich, Yiran Chen, and Hai Li. 2020. Snooping Attacks on Deep Reinforcement Learning. In Proceedings of the 19th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS '20). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 557–565.

THE UNIVERSITY OF MELBOURNE

Adversarial Reinforcement Learning in Autonomous Cyber Defence

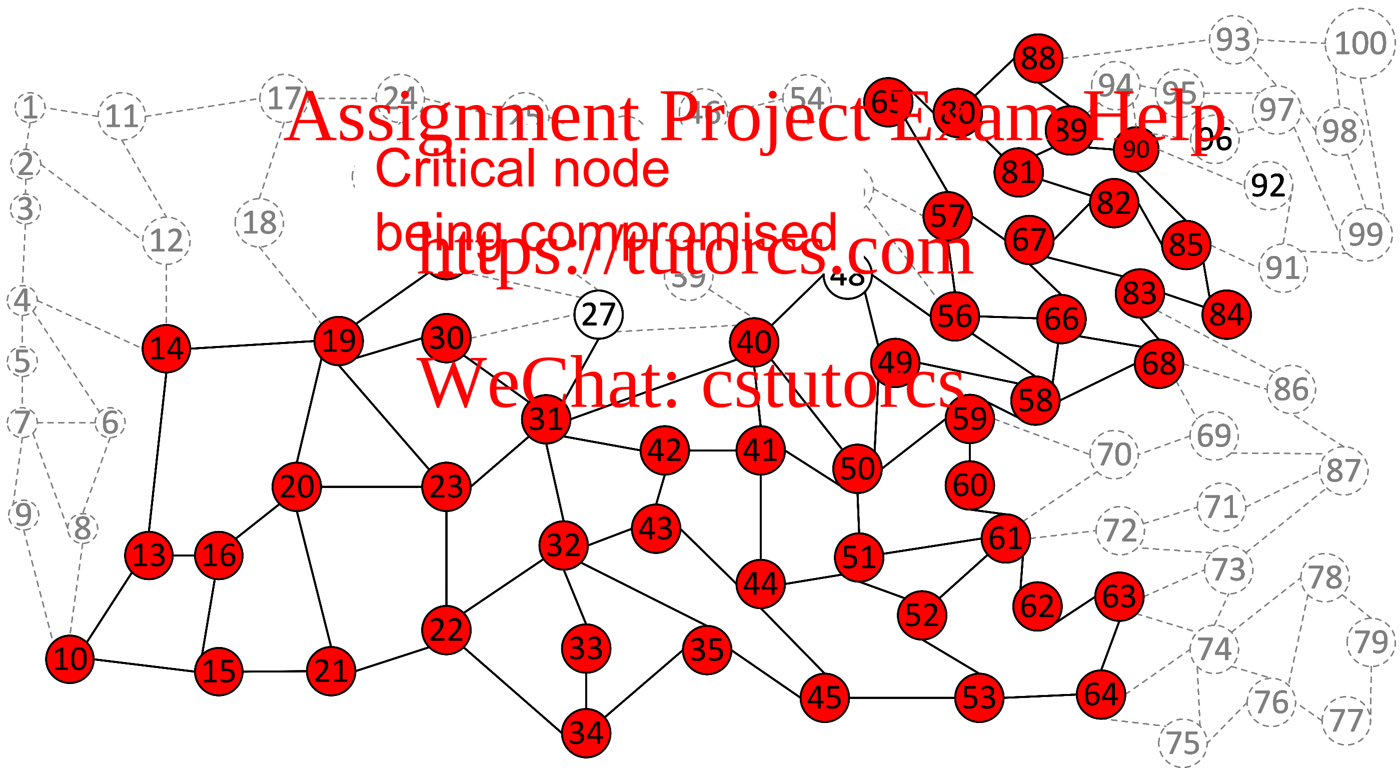
Adversarial attacks against RL models

- Attacker: propagates through the network to compromise the critical server
- The defender applies RL to prevent the critical server from compromise, and preserve as many nodes as possible



Adversarial attacks against RL models

- Attacker: partial observability of the network topology

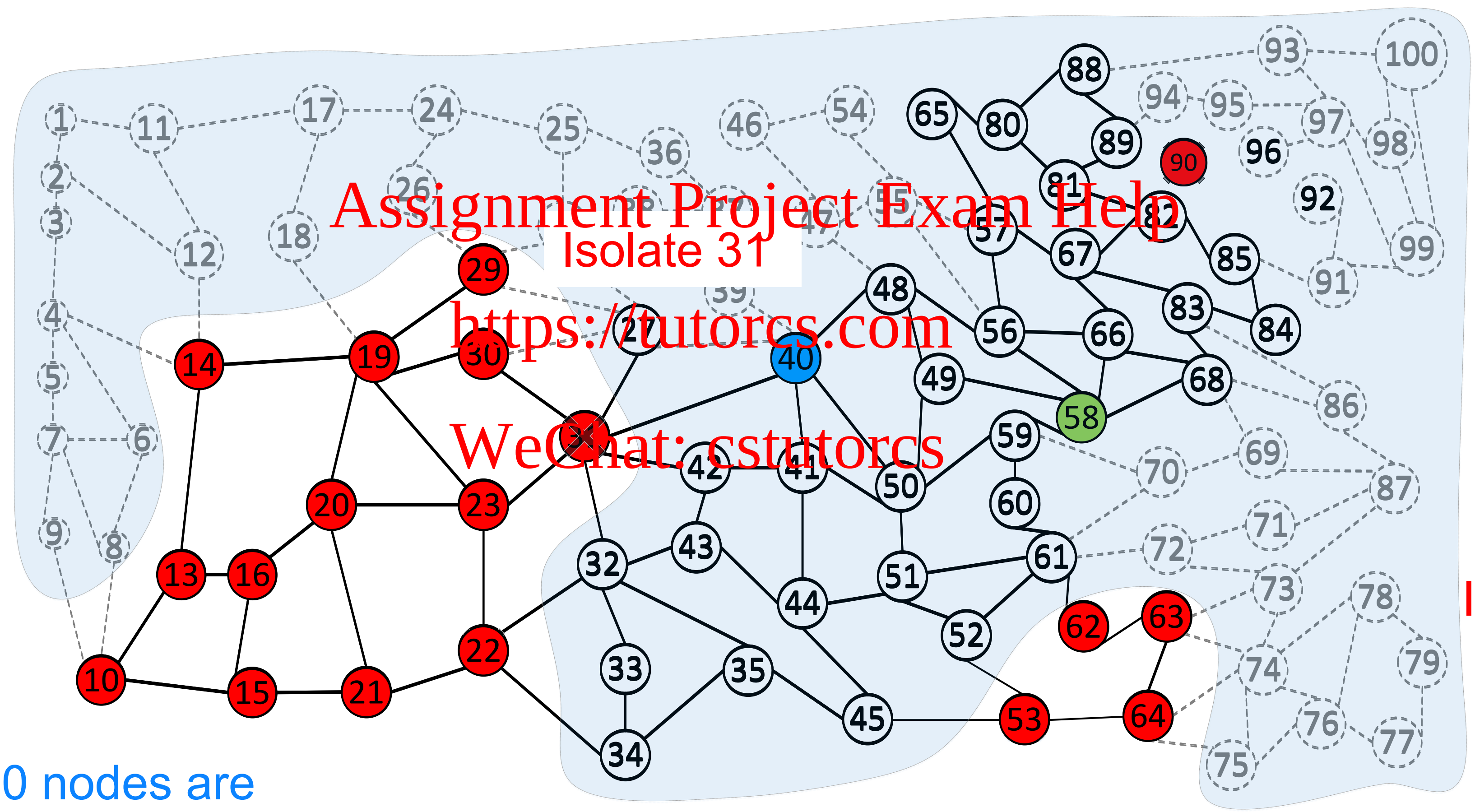


Adversarial attacks against RL models

- Problem definition
  - State: $[\underbrace{0, 0, \ldots, 0}_{\text{State of each node, 0: uncompromised, 1: compromised}}, \overbrace{0, 0, \ldots, 0}^{\text{State of each link, 0: on, 1: off}}]$
  - Action:
    - Action 0~*N*-1: isolate & patch a node $i \in [0, N-1]$
    - Action *N*~2*N*-1: reconnect a node $i \in [0, N-1]$
    - Action 2*N*~2*N*+*M*-1: migrate the critical node to one of the *M* destinations
    - Action 2*N*+*M*: take no action
  - Reward:
    - -1: (1) critical node is compromised or isolated, (2) invalid action
    - Proportional to number of uncompromised nodes that can still reach the critical node
  - Attacker can only compromised a node *x* if there is a visible link between *x* and any compromised node

Adversarial attacks against RL models

- Without training-time attacks



Isolate 90

Isolate 62

82/100 nodes are preserved

Isolate 22

Isolate 53

Adversarial attacks against RL models

- Training-time attack: manipulate states to prevent agents from taking optimal actions
  - $(s_t, a_t, s_{t+1}, r_t) \rightarrow (s_t, a_t, s_{t+1} + \delta_{t+1}, r_t')$
  - Binary state → cannot use gradient-descent based method
  - δ : false positives & false negatives
  - The attacker cannot manipulate the states of all the observable nodes
    - L_{FP}: nodes that can be perturbed as false positive
    - L_{FN}: nodes that can be perturbed as false negative
  - $\min Q(s_{t+1} + \delta_{t+1}, a_{t+1})$, a_{t+1}: the optimal action for s_{t+1} that has been learned so far
    - Loop through L_{FP} (L_{FN}) and flip the state of one node per time
    - Rank all nodes based on ΔQ (decrease of Q-value by flipping state)
    - Flip the states of the top K nodes

Adversarial attacks against RL models

Algorithm 1: Causative attack against DDQN via state perturbation

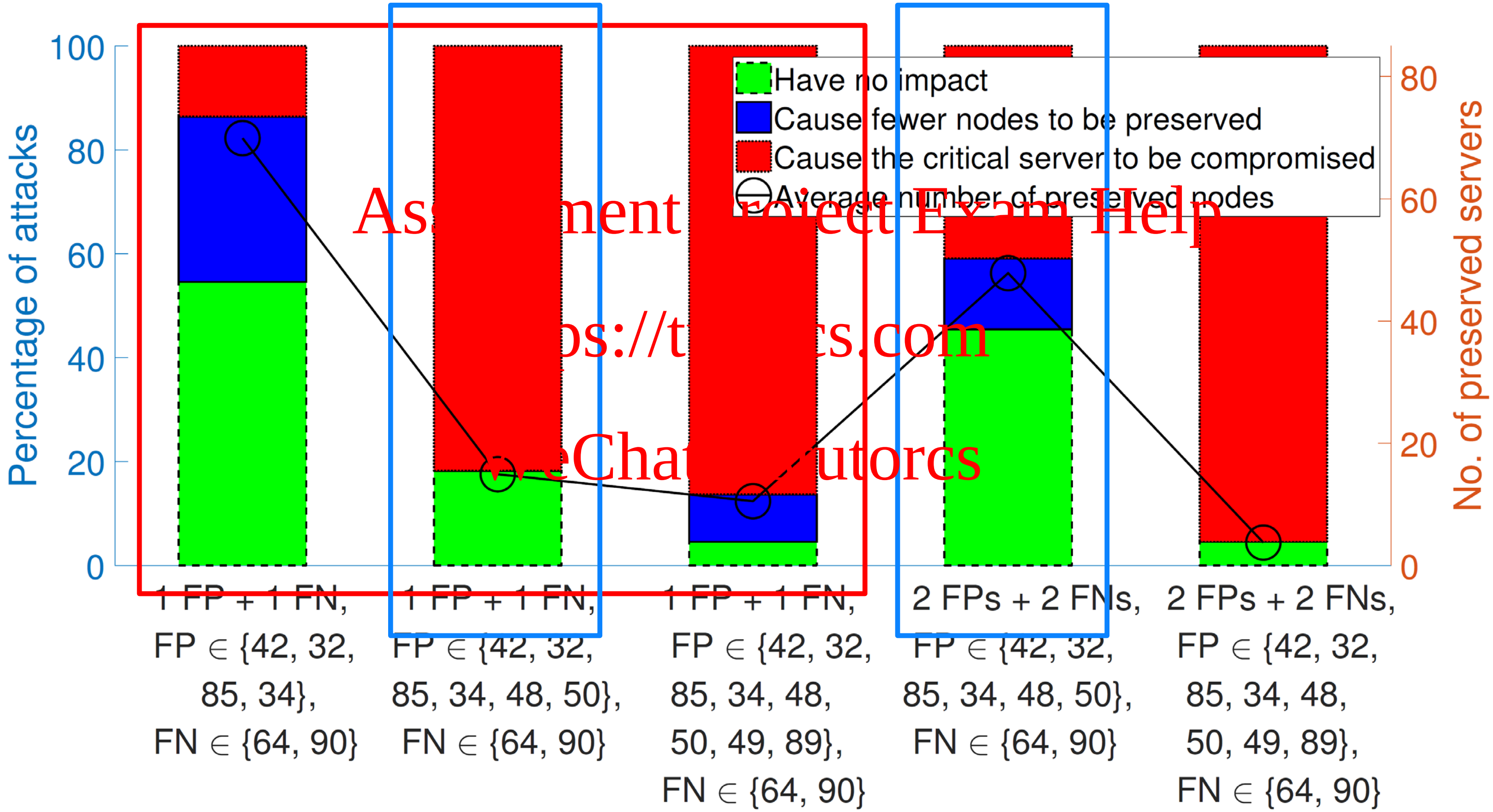
Input : The original experience, (s, a, s', r);
The list of observable nodes, N_O;
The list of nodes that can be perturbed as false positive (false negative) by the attacker, L_{FP} (L_{FN});
The main DQN, Q;
Limit on the number of FPs and FNs per time, LIMIT

Output: The tampered experience $(s, a, s' + \delta, r')$

$FN = FP = \{\}$;
$minQ_{FN} = minQ_{FP} = \{\}$;
$a' = \text{argmax}_{a^*} Q(s', a^*)$;
for *node* n *in* N_O **do**
  if n *is compromised and* n *in* L_{FN} **then**
    mark n as uncompromised;
    if $Q(s' + \delta, a') <$ *any value in* $minQ_{FN}$ **then**
      // δ represents the FP and/or FN readings
      insert n and $Q(s' + \delta, a')$ into appropriate positions in FN and $minQ_{FN}$;
      if $|FN| > LIMIT$ **then**
        remove extra nodes from FN and $minQ_{FN}$;
    restore n as compromised;
  else if n *is uncompromised and* n *in* L_{FP} **then**
    mark n as compromised;
    if $Q(s' + \delta, a') <$ *any value in* $minQ_{FP}$ **then**
      insert n and $Q(s' + \delta, a')$ into appropriate positions in FP and $minQ_{FP}$;
      if $|FP| > LIMIT$ **then**
        remove extra nodes from FP and $minQ_{FP}$;
    restore n as uncompromised;
Change nodes in FN to uncompromised;
Change nodes in FP to compromised;
return $(s, a, s' + \delta, r')$

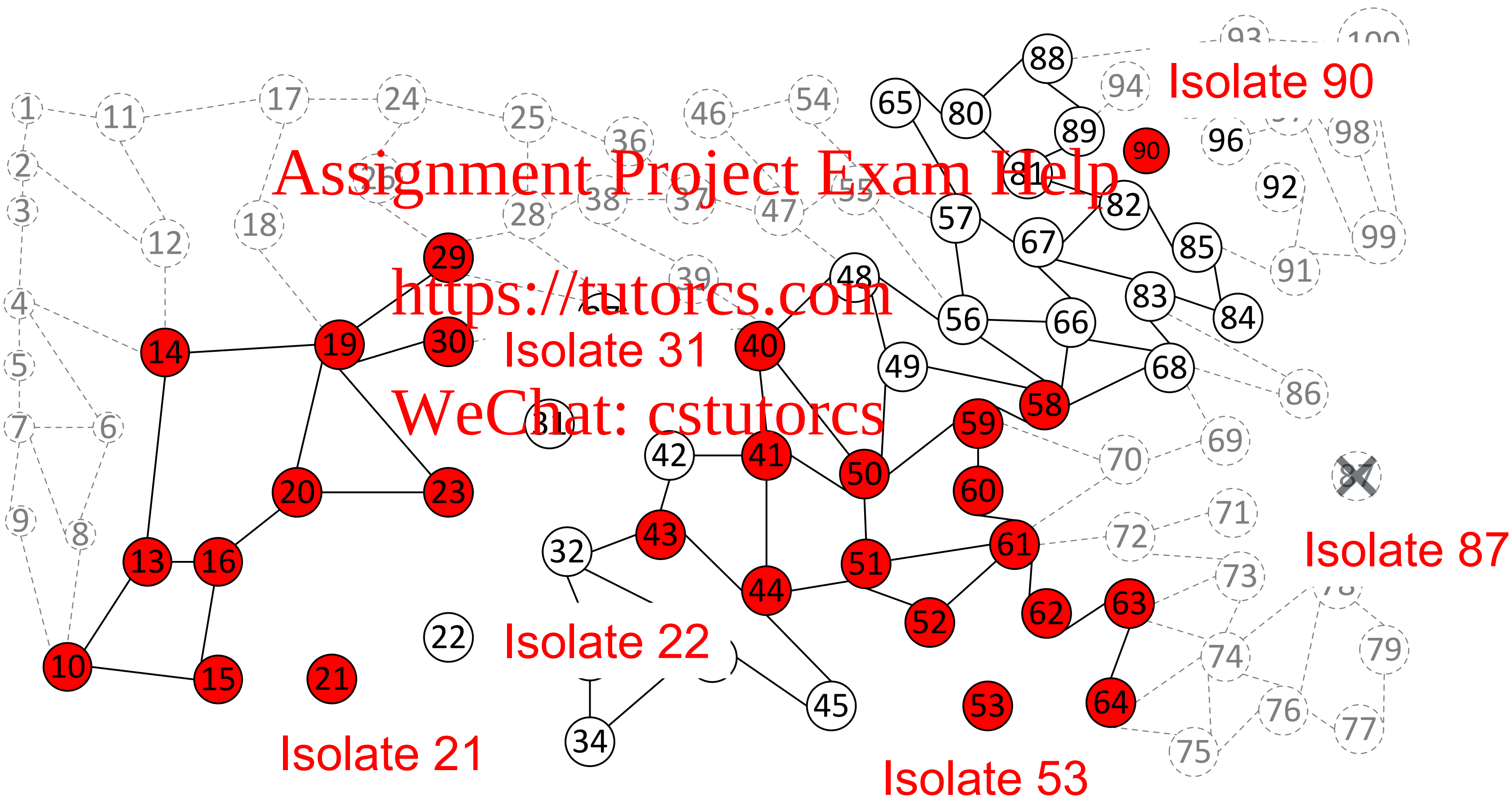
Adversarial attacks against RL models

- Result



Adversarial attacks against RL models

- After training-time attacks



Inversion defence method

- Aim to revert the perturbation/false readings

Attacker	Defender
$s_{t+1} \rightarrow s_{t+1} + \delta_{t+1}$ minimise $Q(s_{t+1} + \delta_{t+1}, a_{t+1})$	$s_{t+1} + \delta_{t+1} \rightarrow s_{t+1} + \delta_{t+1} + \delta'_{t+1}$ maximise $Q(s_{t+1} + \delta_{t+1} + \delta'_{t+1}, a_{t+1})$
Loop through L_{FP} and L_{FN}	Loop through all nodes
Flip K nodes	Flip K' nodes

- Effective even if $K' \neq K$
- Minimum impact on normal training process (i.e., $K = 0$, $K' > 0$)

Inversion defence method

- Before & after the defence method is applied