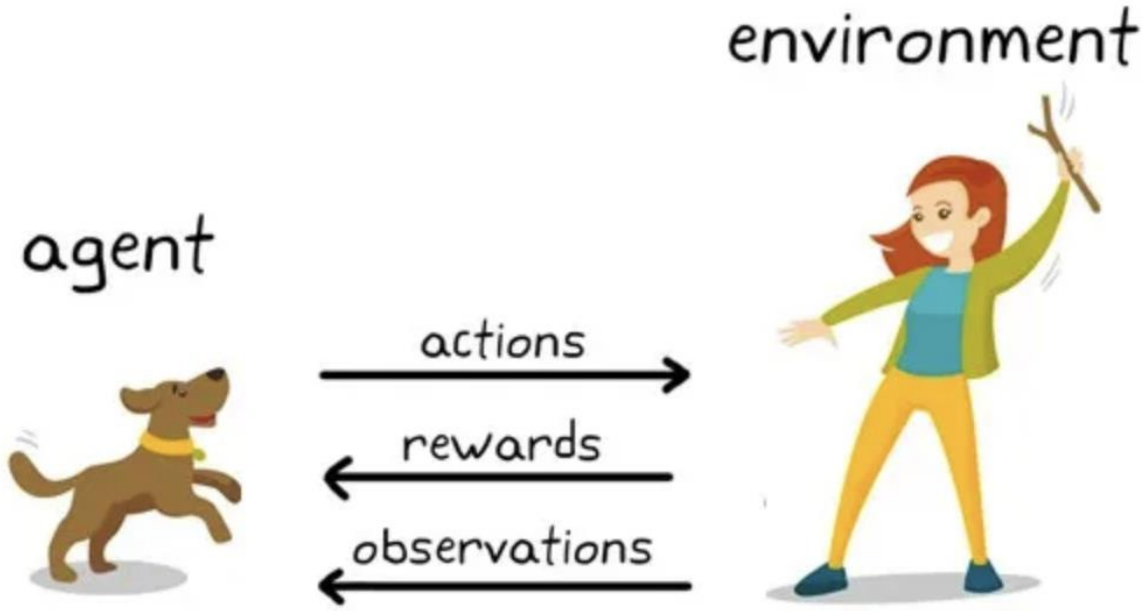
Lab 3

Reinforcement Learning

Tutor: cherl@uow.edu.au

Reinforcement Learning (RL)

- RL is a unique branch of machine learning (ML) where an agent learns to make decisions through trial and error.
- The agent observes and acts within an environment, receiving
  - Rewards for good decisions
  - Penalties for bad decisions
- Agent's goal: Device a strategy that maximizes positive feedback over time.
- The process is iterative and based on trial and error.



An Analogy of Reinforcement Learning

Tutor: cherl@uow.edu.au

RL vs other types of ML

	Supervised learning	Unsupervised learning	Reinforcement learning
Data type	Labeled training data e.g., transaction records	Unlabeled training data e.g., images, data	Interaction data (states, actions, rewards) collected through environment interactions.
Goal	Predict outcomes based on input data e.g., spam detection; stock price prediction	Uncover underlying patterns e.g., 1) extract features from images; 2) extract pattern of behaviour (if a customer buys bread, he/she is likely to buy butter)	Makes decisions that maximize reward from the environment
Types of problem to solve	Classification, Regression	Classification, Recommendation system, Market basket analysis	Decision making tasks

Reinforcement learning

In reinforcement learning, there are two broad approaches:

1. **Model-based methods**:
   - Where you explicitly learn or use a model of the environment (the P and R functions)
     - P: Transition probability of landing in s' from s by taking action a
     - R: Immediate reward for taking action a while in state s
   - Both P and R are known beforehand or from experience, and together they describe the environment.
   - Model-based method calculate the values of Q(s, a) and V(s) using P and R.
2. **Model-free methods**:
   - Where you learn values or policies directly without explicitly modelling the environment.
   - Hence, there are no explicit P and R functions.
   - Model-free method updates Q(s, a) from samples of experience (Q-learning) or while following a specific policy (SARSA).

Naming convention

1. Q(s, a): Action value function
   - You are in state s.
   - You want to know how good is it to take action “a” from state “s”.
   - Q(s, a) = the expected return when taking action “a” in state “s” and then following the optimal policy thereafter.
   - Hence, **Q(s, a) tells you which action to choose while you are in state s**.
   - $Q(s, a) = r(s, a) + \gamma \sum_{s}, P(s'|s, a) \cdot V(s')$ where:
     - r(s,a): immediate reward for taking action a in state s
     - V(s’): expected value of the next state s’.
2. V(s): State value function
   - Represents the expected return of following the optimal policy **starting from state s**.
   - How good is it to be in state “s”?
   - V(s) = max_a Q(s, a)
   - V(s) is the max. return considering all actions taken in state “s”.

Bellman equation for MDP

Bellman Equation $$V(s) = \max_a \left(R(s,a) + \gamma \sum_{s'} P(s,a,s')V(s') \right)$$

- V(s): value when starting at state “s”.
- R(s, a): Reward for taking action ”a” in state “s”.
- P(s, a, s’): Probability of landing in state s’ by taking action “a” in state “s”.
- V(s’): Expected value of next state s’.

Q Learning (Model free learning)

$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a)]$

1. You start in state s
2. You choose and take action a
3. You have an immediate reward r for taking action a
4. This action a leads you to a new state s' (as determined by the environment dynamics e.g., robot moves in grid or in TravelMDP problem, either walk or take bus etc.)
5. Assuming that you're in s', you consider all possible actions a' you could take from this new state s'
6. You find which action a' would give the maximum Q-value: max_a' Q(s',a')
7. This maximum Q value must be discounted by $\boldsymbol{\gamma}$ since it is in the future.
8. You update Q(s,a) by moving it partially toward this target (controlled by **learning rate α**)
9. Q-learning is called "off-policy" because you're updating Q(s,a) using the maximum possible future value, regardless of what action you might actually take next.

Example: Apply Q learning in a grid-world project

Consider a 3×3 grid world where an agent can move in four directions: up, down, left, and right.

States

- 9 possible positions (states):
  - (0,0), (0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1), (2,2)

Actions

- 4 possible actions: Up, Down, Left, Right

Q-table size

- (N^2) * (number of actions)

Q-Table

State	Up	Down	Left	Right
0,0	0	0.2	0	0.1
0,1	0	0.15	0.1	0.3
0,2	0	0.4	0.3	0
1,0	0.2	0	0	0.5
1,1	0.15	0.3	0.5	0.2
1,2	0.4	0	0.2	0
2,0	0	0	0.5	0
2,1	0	0	0.2	0.6
2,2	0	0	0.6	0

E.g., The algo has learnt that in state (1,2), taking action Up will generate a long-term reward of 0.4.

Steps for Q-learning

- Initialize all values in the table to zero.
- Update values in the table using the Q-learning update rule:
- $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \cdot \max Q(s',a') - Q(s,a)]$
- Problem setup:
  - r = 0 when agent moves.
  - $\alpha=0.1$
  - $\gamma=0.9$
- At state (0,0), takes action Right, r=0, and moves to state (0,1)
- Current Q(0,0, Right) = 0.1
- max Q(0,1, a') = max(0.0, 0.15, 0.1, **0.3**) = **0.3**
- New Q(0,0, Right) = Current Q(0,0,Right) + $\alpha[r + \gamma \cdot \max Q(0,1,a') - \text{Current } Q(0,0,\text{Right})]$
- $0.1 + 0.1 \times [0 + 0.9 \times 0.3 - 0.1] = 0.1 + 0.1 \times [0.27 - 0.1] = 0.1 + 0.1 \times 0.17 = 0.1 + 0.017 = 0.117$
- Hence, (0,0, Right) will be updated to 0.117.

Lab 3: Travel problem

Description:

- Street with blocks numbered 1 to n.
- Walking from s to s+1 takes 1 minute (hint: award -1.0).
- Taking a magic bus from s to 2s takes 2 minutes (hint: award -2.0).
- The magic bus fails with probability p, which means it will leave you in the same state but still costs 2 minutes.

Example:

1. Walking from block 2 to block 3 takes 1 minute.
2. In block 2, if you choose to take magic bus, there will be probability p you will stay in the same block and with probability 1-p, you will arrive at block 4, both of which take 2 minutes.

Tutor: cherl@uow.edu.au

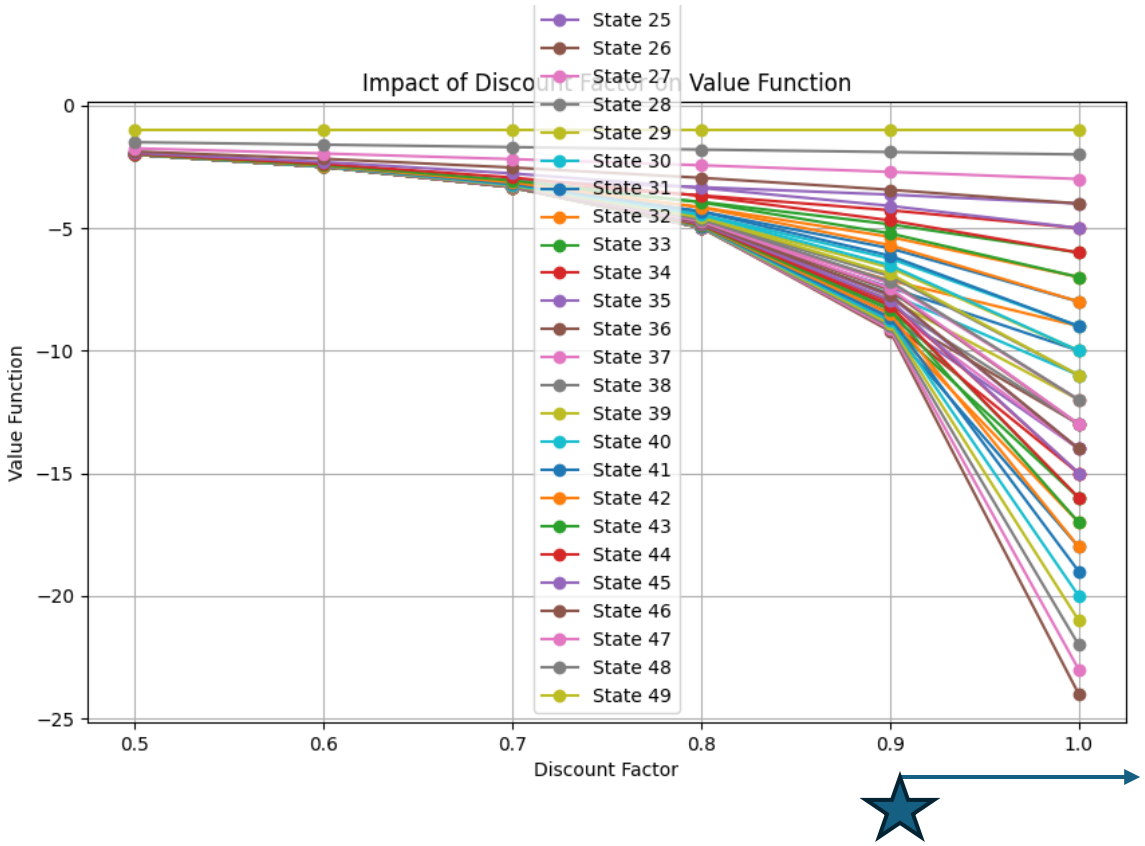
Lab 3: Travel problem

- Walk through solving the Bellman equation using Excel calculator.
- Run the program provided in Colab. https://colab.research.google.com/drive/1Z2NC6tkKJQ8fAMmmiMwoigMdODRczf-3?usp=sharing
- Observe the impact on the value functions using the graphs generated. Change the parameters to explore the model.

Tutor: cherl@uow.edu.au

Impact of discount factor on value function

- Value function decreases vs discount factor approaches 1.
- Higher discount factor means future rewards or penalties are valued more.
- This leads to longer path to the goal.

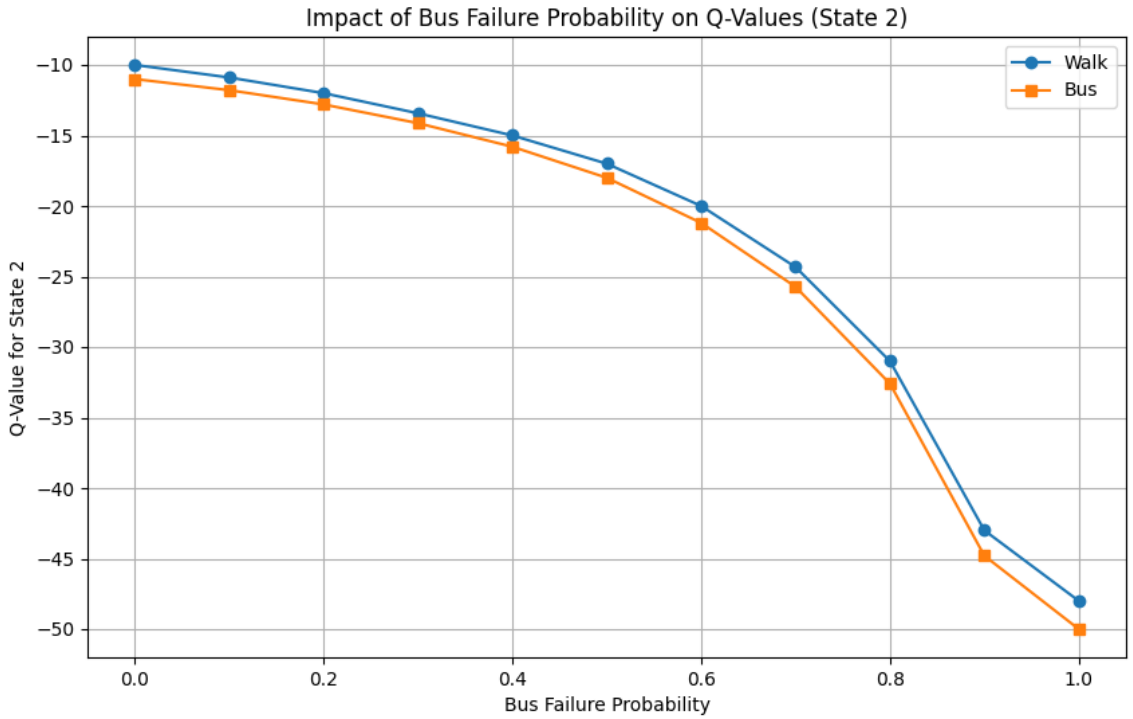


Agent changes in behaviour significantly when discount factor >=0.9

Tutor: cherl@uow.edu.au

Impact of bus failure probability on Q-values

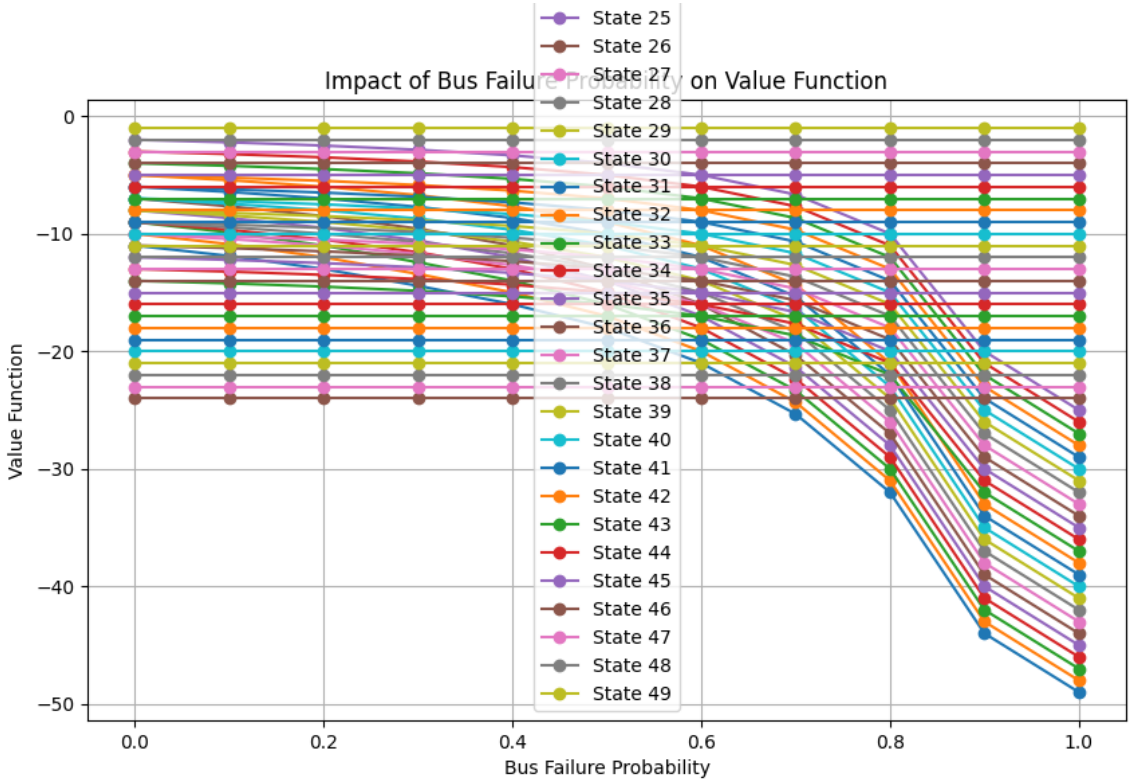
- Pick a state to analyze e.g. State 2
- Q values between bus and walk are quite close between p=0 to 0.85.
- That means the 2 modes of action (walk, bus) are quite “competitive” with each other.
- Beyond 0.85, walking is clearly better than taking bus.



Tutor: cherl@uow.edu.au

Impact of bus failure probability on value function

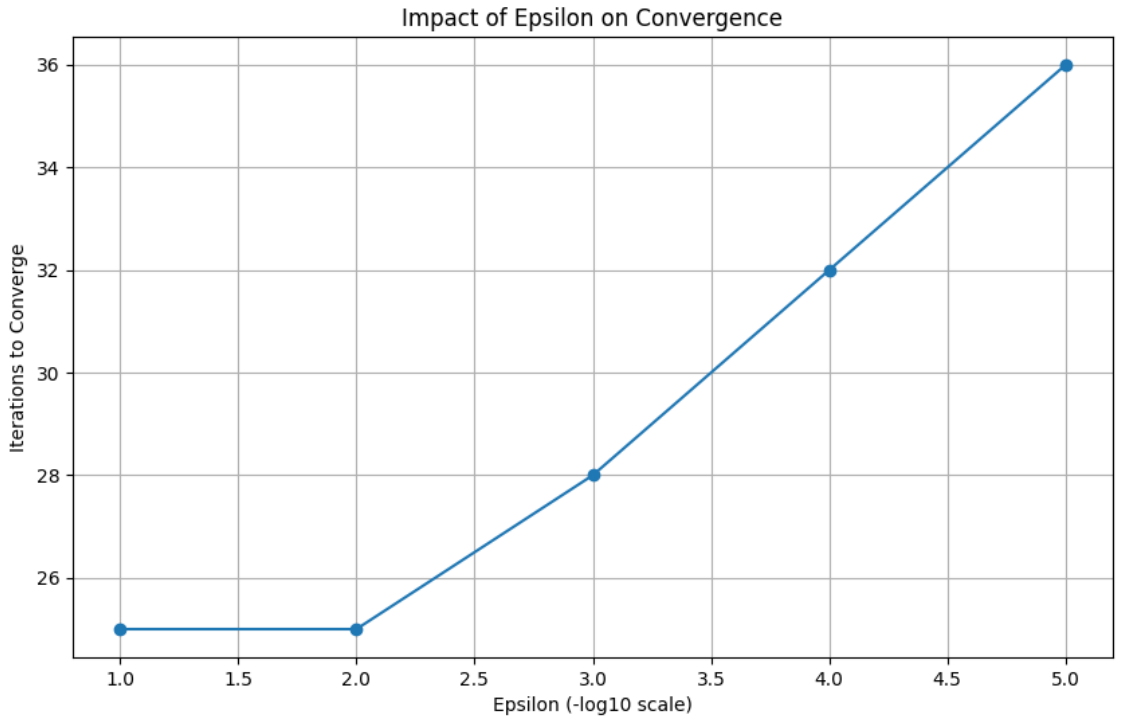
- Bus failure probability seems to have higher impact on the states that are further away from the goal state.
- That means smaller state numbers have more negative values than the higher state numbers when bus failure problem is high e.g., > 0.8
- Horizontal lines interpretation: Possible for states that have never used bus either because they are too close to the goal state.



Tutor: cherl@uow.edu.au

Impact of Epsilon on convergence

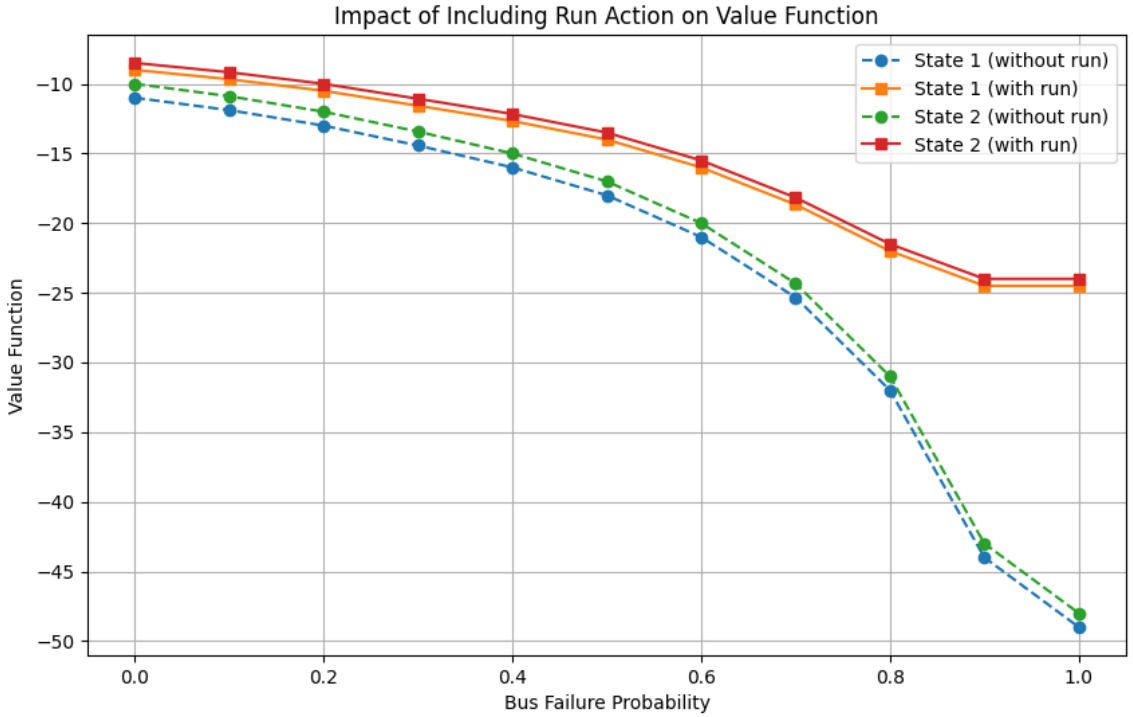
- The plot's x-axis is $-\log_{10}(\text{epsilon})$
- Hence, $x = -\log_{10}(\text{epsilon}) \Rightarrow \log_{10}(\text{epsilon}) = -x$
- $10^{-x} = \text{epsilon}$
- For example, when $x = 1.5$, $\text{epsilon} = 10^{(-1.5)} = 0.0316$.
- So, the **higher the x values, the smaller the Epsilon**.
- The smaller the epsilon (stricter criteria), the higher the number of iterations.
- Note: Epsion is the maximum allowed change in V(s) between consecutive iterations.



Tutor: cherl@uow.edu.au

Impact of Run on value function

- Having additional option such a Run has an overall positive impact on the value function (less negative).
- This observation is consistent across the different values of bus failure probability.
- Having more action choices should be considered especially for states close to the goal state.



Tutor: cherl@uow.edu.au