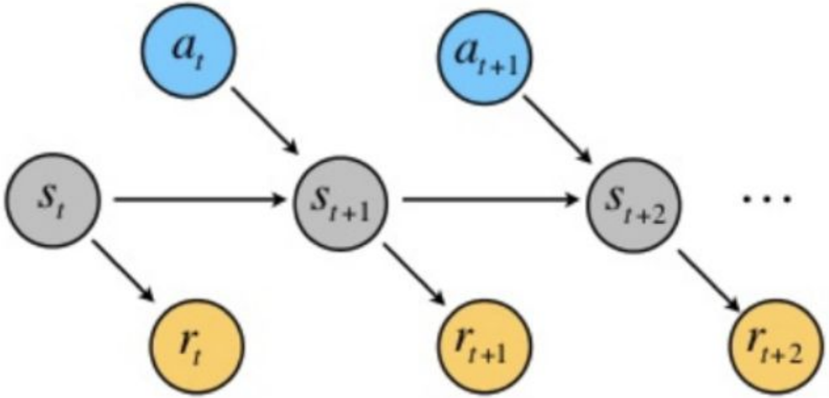
General Principles of Human and Machine Learning

Tutorial 3: Introduction to RL

RL Framework

- In RL, problems can be described using the MDP formality
  - MDP is a 4-tuple (S, A, P, R)
    - S: state space
    - A: action space
    - P: state transition probability
      - $P(S_{t+1}=s_{t+1}|S_t=s_t,A_t=a_t)$
    - R: state transition returns
      - $R(s_t,s_{t+1}) = r_t$
  - Markov Decision Process because of the Markov property
    - $P(S_{t+1}|S_t,\ldots,S_1,A_t) = P(S_{t+1}|S_t,A_t)$



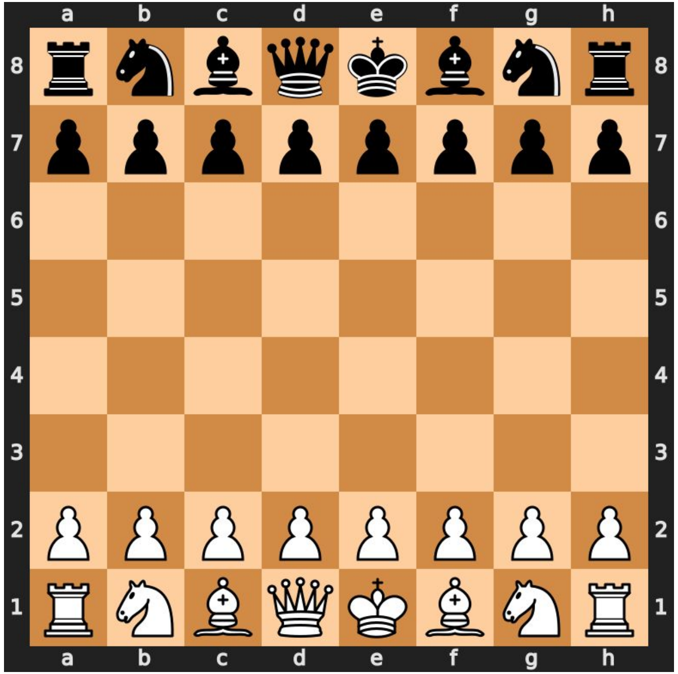
Andrey Markov

Tutorial Questions

Devise three example tasks of your own that fit into the reinforcement learning framework, identifying for each its states, actions, and rewards. Make the three examples as different from each other as possible. The framework is abstract and flexible and can be applied in many different ways. Stretch its limits in some way in at least one of your examples.

Example 1: Chess

- S: board position of the pieces
- S_0 = {Black rook: a8, Black knight: b8, …., White rook: h1}



Example 1: Chess

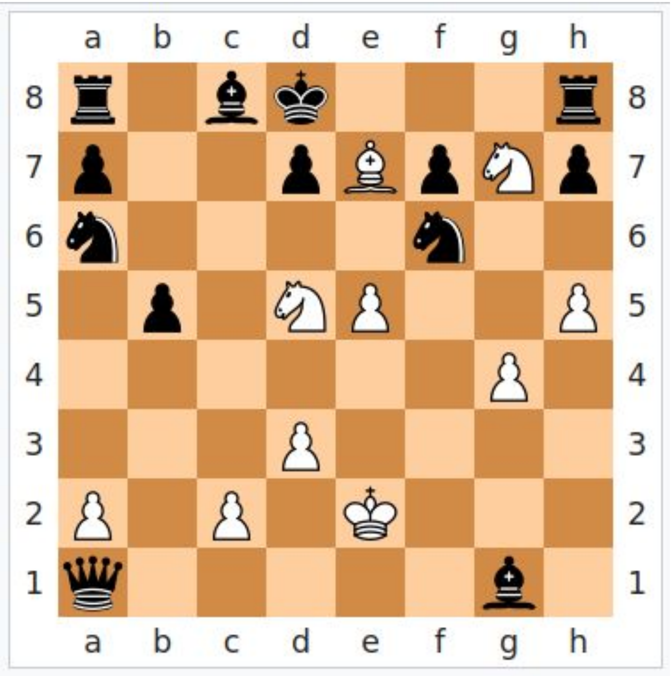
- A: all the legal moves (whites) given the current board configuration
  - For example, on the board to the right:
    - White pawn: e2 →e4

Example 1: Chess

- P: the probability of each board configuration given player's move and the current configuration.
  - For example, on the board to the right:
    - Given White pawn: e2 →e4
      - Black pawn: e7 →e5
      - But could have also played:
        - Black pawn: d7 →d5
        - Black pawn: e7 →e6
        - Black knight: b8 →c6
        - Etc
- $P(S_1$ = {Black rook: a8, Black knight: b8, …., Black pawn: e5,White pawn: e6, …, White rook: h1} | S_0 = {Black rook: a8, Black knight: b8, …., White rook: h1}, A_0 = White pawn: e2 →e4$) \in [0,1]$
- Note that the Markov property is preserved:
  - Previous board configurations don't affect the transitions:
    - All the necessary information is incorporated into the current board configuration

Example 1: Chess

- R:
  - Endgame:
    - Win: +1

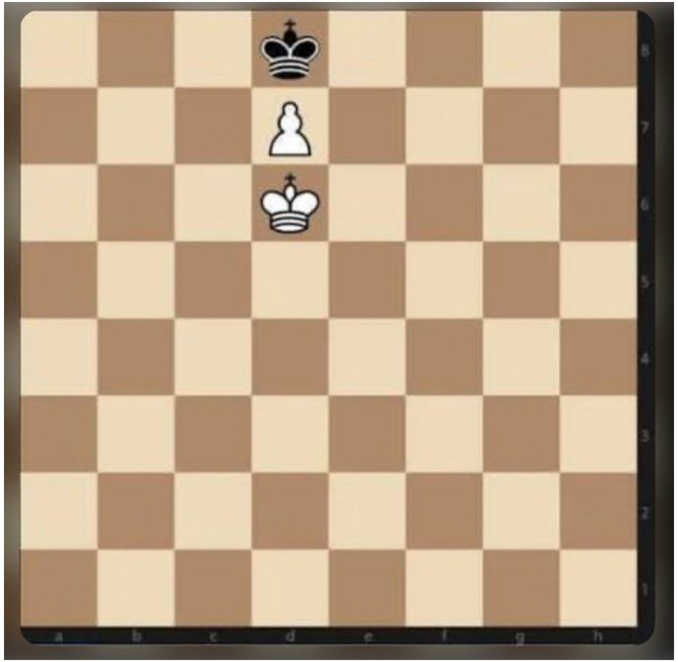


Example 1: Chess

- R:
  - Endgame:
    - Win: +1
    - Loss: -1

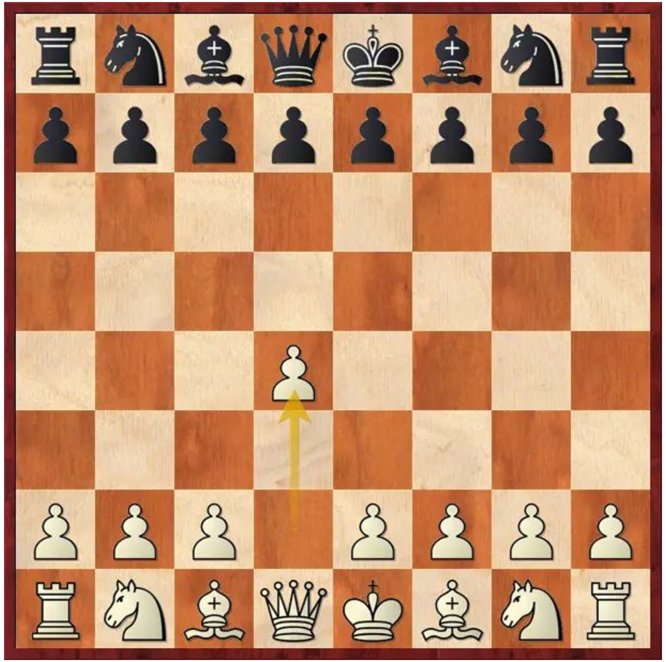
Example 1: Chess

- R:
  - Endgame:
    - Win: +1
    - Loss: -1
    - Draw: 0



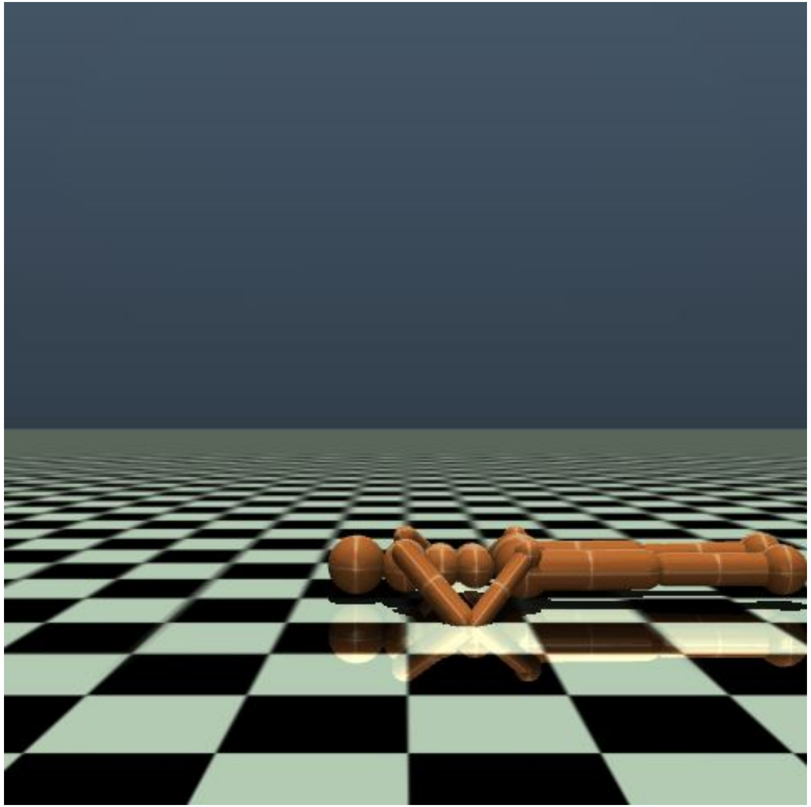
Example 1: Chess

- R:
  - Endgame:
    - Win: +1
    - Loss: -1
    - Draw: 0
  - Otherwise:
    - If a move doesn’t lead to an endgame situation: 0
      - This is a major problem in RL called sparsity of reward
      - It leads to the credit assignment problem



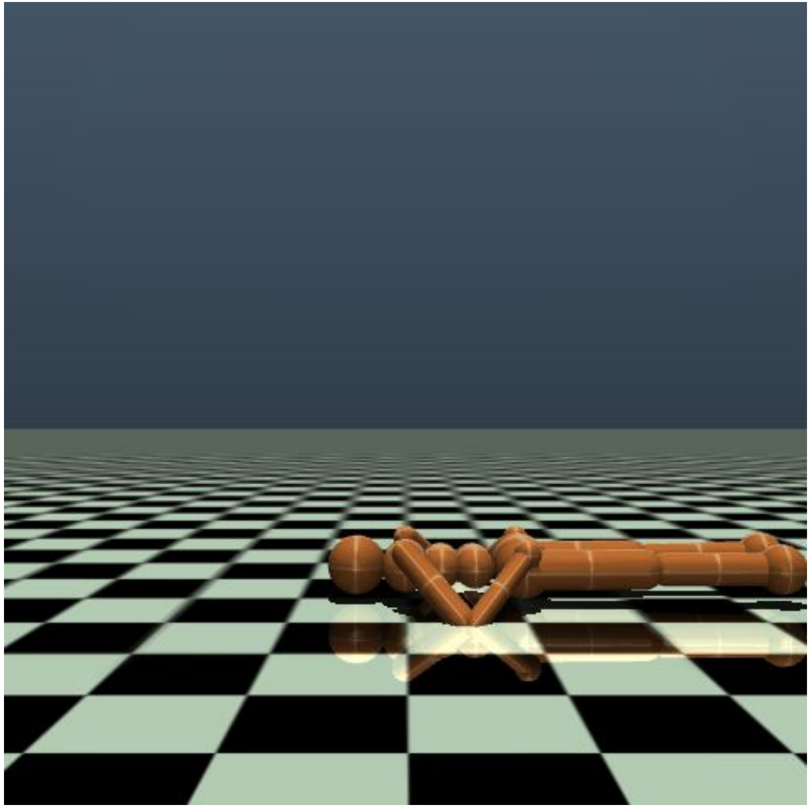
Example 2: OpenAI Gym (Humanoid Standup)

- S: The state space consists of positional values of different body parts of the Humanoid, followed by the velocities of those individual parts (their derivatives) with all the positions ordered before all the velocities.
    - 1: z-coordinate of the torso (centre)
    - 2: x-orientation of the torso (centre)
    - …
    - 45: angular velocity of the angle between left upper arm and left_lower_arm



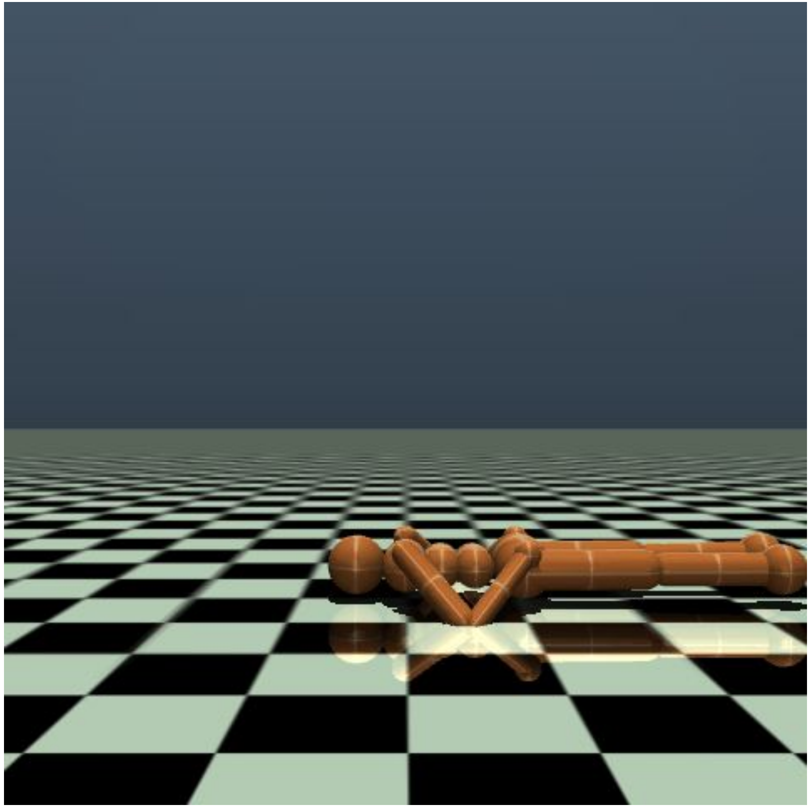
Example 2: OpenAI Gym (Humanoid Standup)

- A: The agent take a 17-element vector for actions representing the numerical torques applied at the hinge joints.
    - 0: Torque applied on the hinge in the y-coordinate of the abdomen
    - 1: Torque applied on the hinge in the z-coordinate of the abdomen
    - …
    - 16: Torque applied on the rotor between the left upper arm and left lower arm



Example 2: OpenAI Gym (Humanoid Standup)

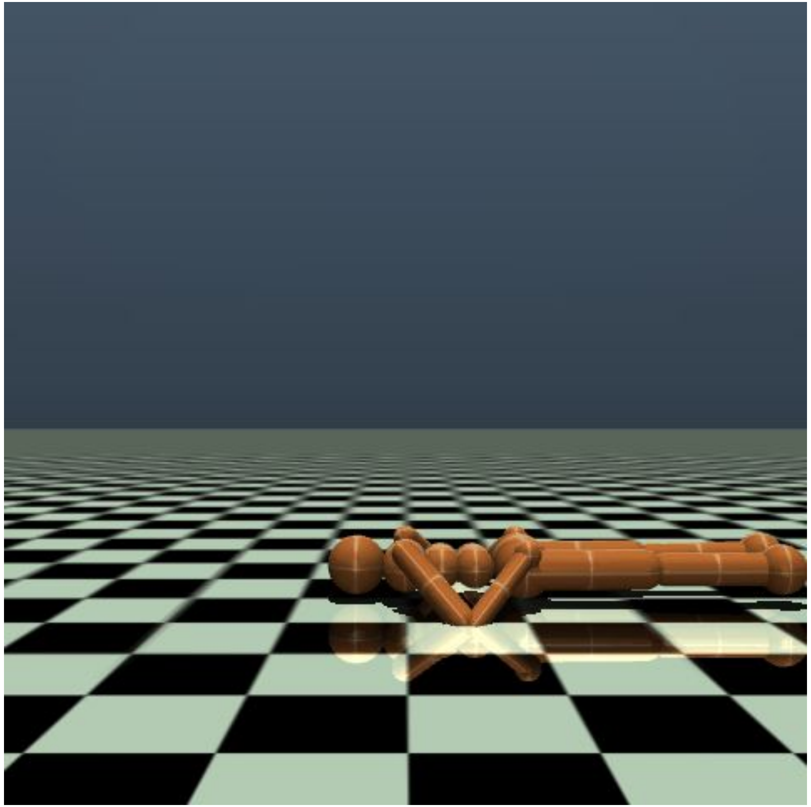
- P: the probability of the next positional values of different body parts given their velocities and applied torques at the joints
    - The environment is inherently noisy which makes the transitions non-deterministic



Example 2: OpenAI Gym (Humanoid Standup)

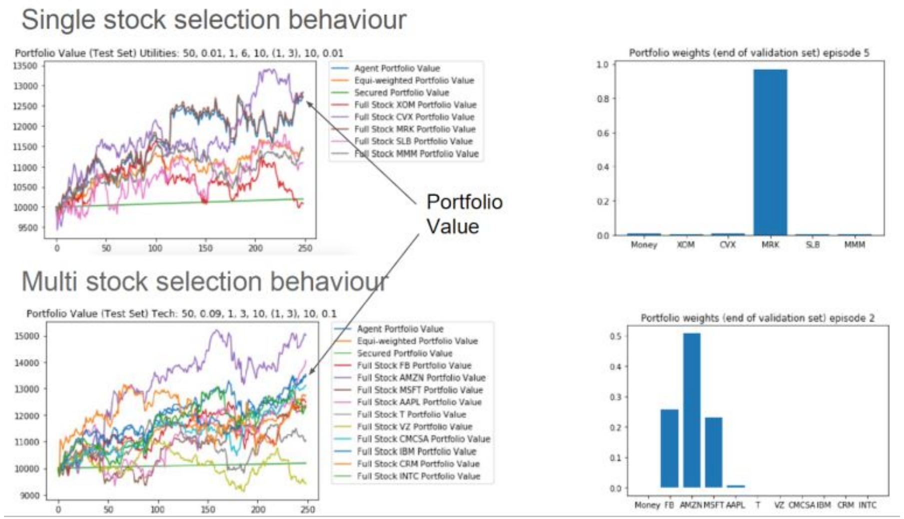
- R: a reward for moving upward
  - Technically there are other rewards as well, but it's irrelevant for our purposes

https://www.youtube.com/shorts/K-pzg5nw7us



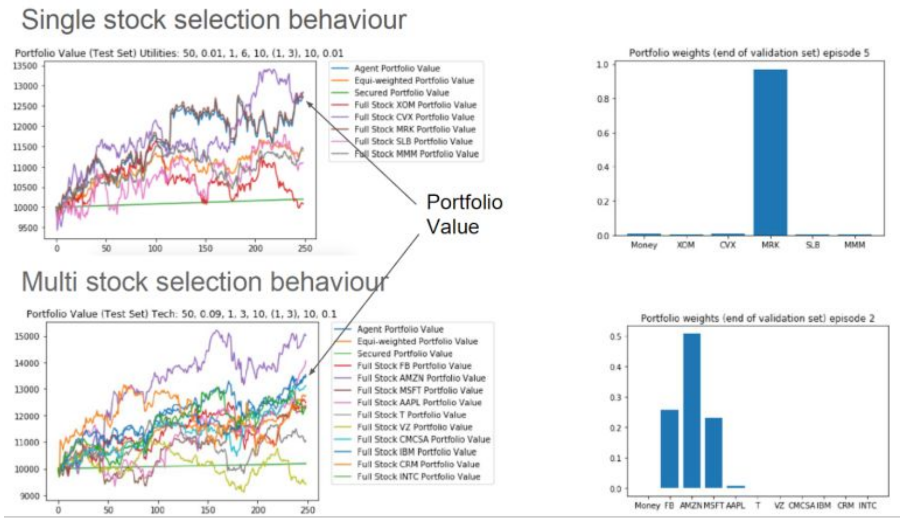
Example 3: Portfolio management (Finance)

- S: Number of each stock in the portfolio and their current value
  - Full Stock FB (Number: 100, Price: 235.79)
  - Full Stock AMZN (Number: 50, 112.18)
  - …



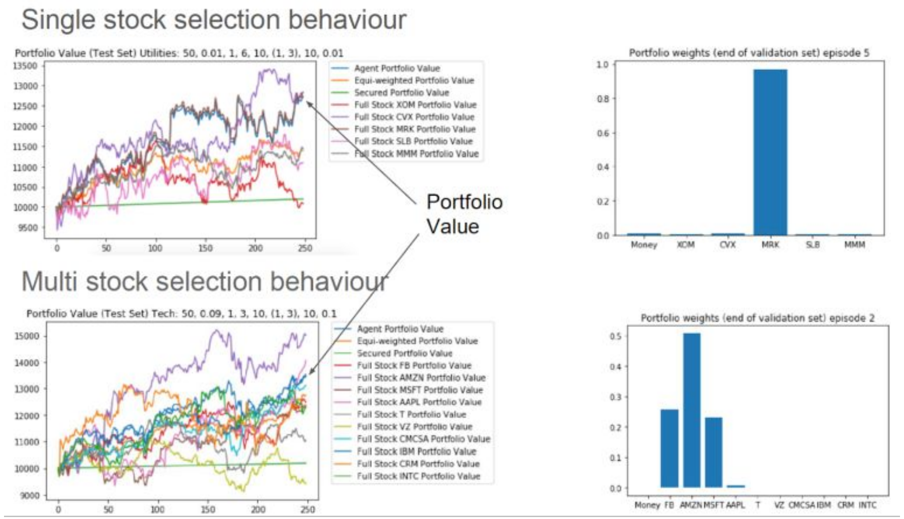
Example 3: Portfolio management (Finance)

- A: Buy #stocks available in the market, Sell #stocks in your portfolio:
  - Sell Full Stock FB (Number: 10)
  - Buy Full Stock BABA (Number: 20) 112.18)
  - etc



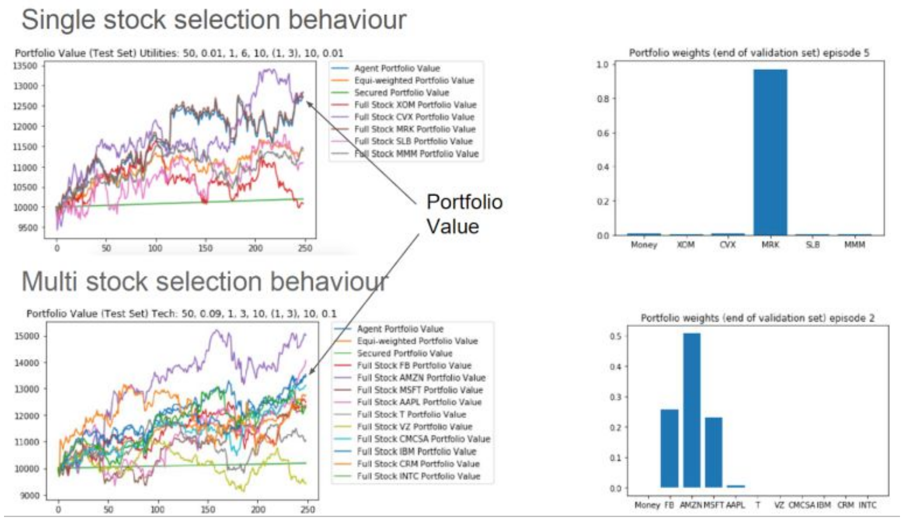
Example 3: Portfolio management (Finance)

- P: the probability of the next time step values of the stocks in the market and the number of stocks the agent has in the portfolio
    - The values are inherently stochastic because they depend on market forces that aren't fully predictable by the agent



Example 3: Portfolio management (Finance)

- R: the change in portfolio valuation



Tutorial Questions

Describe/program a 2, 3 or 4-armed bandit, where each option has a different reward distribution

- Programmers: describe each bandit based on a Gaussian distribution, with its own mean and variance
- If not a programmer: describe the rewards of each bandit based on coin flips or dice rolls
    - E.g., Rolling a D6 + 2, flipping a coin: heads = 1, tails = 3, etc…

Exercise 2: Sample actions from policy

assume:
$\eta = .9$
$\tau = .25$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0			
t=2					
t=3					
t=4					

Tutorial Questions

Describe/program a 2, 3 or 4-armed bandit, where each option has a different reward distribution

Python

```python
import numpy as np
import pandas as pd

# Parameters
K = 2  # number of options
meanVec = np.random.uniform(-10, 10, K)  #
Payoff means, sampled from uniform
distribution between -10 and 10
sigmaVec = np.ones(K)  # Variability of
payoffs (standard deviation), set to 1 for
all options

# Bandit generator function
def bandit_generator(k):
    # k can be an integer or a list of
integers representing arms to pull
    payoff =
np.random.normal(loc=meanVec[k],
scale=sigmaVec[k], size=len(k) if
hasattr(k, '__len__') else 1)
    return payoff

# Generate 25 random actions
action_sequence =
np.random.choice(range(K), size=25,
replace=True)

# Generate payoffs
payoffs =
bandit_generator(action_sequence)

# Create a dataframe of the actions and
payoffs
df = pd.DataFrame({'action':
action_sequence, 'payoff': payoffs})

print(df)
```

R

```r
K = 2 #number of options
meanVec <- runif(n=K, min = -10, max = 10)
#Payoff means, which we sample from uniform
distribution between -10 to 10
sigmaVec <- rep(1, K) #Variability of payoffs
(as a standard deviation), which we assume is
the same for all options and is set to 1.

banditGenerator <- function(k) {#k is an
integer or a length k vector of integers,
selecting one of the 1:K arms of the bandit
  payoff <- rnorm(n = length(k), mean =
meanVec[k], sd = sigmaVec[k])
  return (payoff)
}

# generate 25 random actions
actionSequence <- sample(1:K, size = 25,
replace = TRUE)
# generate payoffs
payoffs <- banditGenerator(actionSequence)

# create a dataframe of the actions and payoffs
df <- data.frame(action = actionSequence,

payoff = payoffs)
```

Tutorial Questions

Implement a Q-learning model

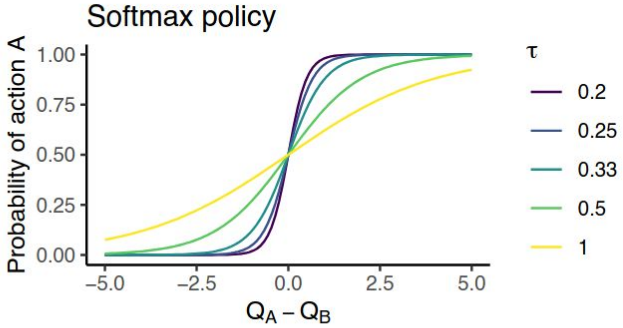
Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta \left[r - Q_t(a) \right]$$

Policy

temperature

$$P(a) \propto \exp(Q_t(a)/\tau) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$



Exercise 2: Sample actions from policy

assume:
$\eta = .9$
$\tau = .25$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A		
t=2					
t=3					
t=4					

Tutorial Questions

Implement a Q-learning model

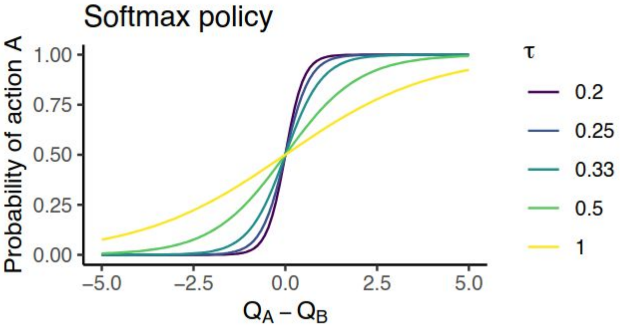
Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta \left[r - Q_t(a) \right]$$

Policy

temperature

$$P(a) \propto \exp(Q_t(a)/\tau) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$



Exercise 2: Sample actions from policy

assume:

$\eta = .9$

$\tau = .25$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A	4	
t=2					
t=3					
t=4					

Tutorial Questions

Implement a Q-learning model

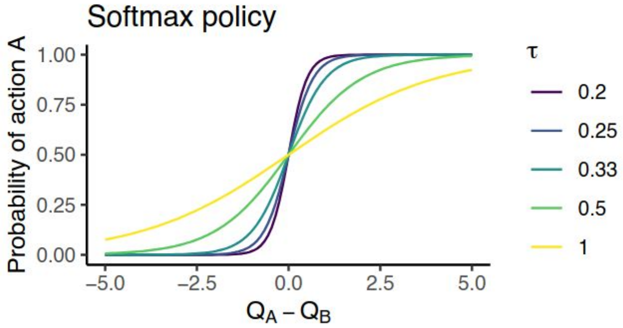
Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta \left[r - Q_t(a) \right]$$

Policy

temperature

$$P(a) \propto \exp(Q_t(a)/\tau) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$



Exercise 2: Sample actions from policy

assume:
$\eta = .9$
$\tau = .25$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A	4	4
t=2					
t=3					
t=4					

Tutorial Questions

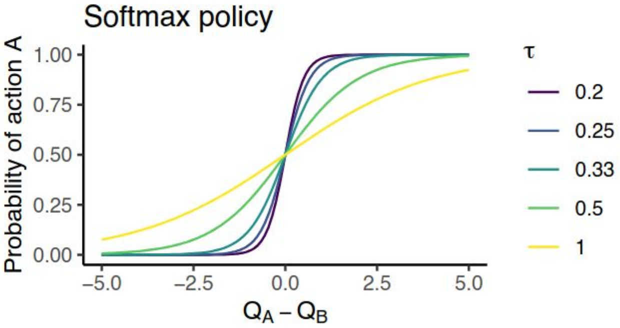
Implement a Q-learning model

Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta \left[r - Q_t(a) \right]$$

Policy

$$P(a) \propto \exp(Q_t(a)/\underbrace{\tau}_{\text{temperature}}) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$



Exercise 2: Sample actions from policy

assume:
$\eta = .9$
$\tau = .25$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A	4	4
t=2	3.6				
t=3					
t=4					

Tutorial Questions

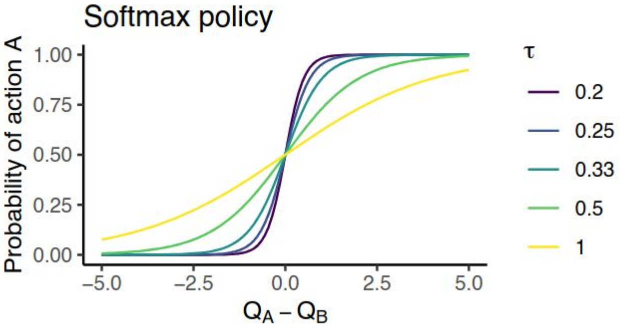
Implement a Q-learning model

Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta\left[r - Q_t(a)\right]$$

Policy

$$P(a) \propto \exp(Q_t(a)/\underset{\text{temperature}}{\tau}) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$



Exercise 2: Sample actions from policy

assume:
$\eta = .9$
$\tau = .25$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A	4	4
t=2	3.6	0			
t=3					
t=4					

Tutorial Questions

Implement a Q-learning model

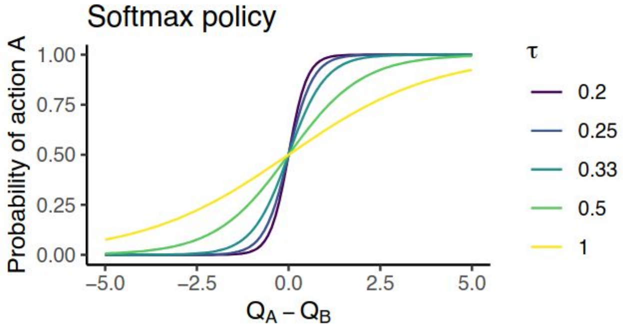
Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta\left[r - Q_t(a)\right]$$

Policy

temperature

$$P(a) \propto \exp(Q_t(a)/\tau) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$



Exercise 2: Sample actions from policy

assume:
$\eta = .9$
$\tau = .25$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A	4	4
t=2	3.6	0	A		
t=3					
t=4					

Tutorial Questions

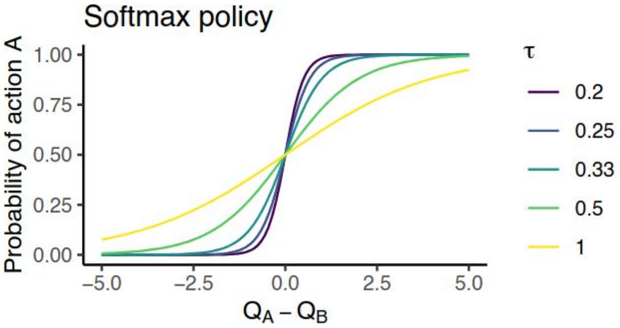
Implement a Q-learning model

Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta \left[r - Q_t(a) \right]$$

Policy

$$P(a) \propto \exp(Q_t(a)/\underbrace{\tau}_{\text{temperature}}) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$



Exercise 2: Sample actions from policy

assume:
$\eta = .9$
$\tau = .25$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A	4	4
t=2	3.6	0	A	8	
t=3					
t=4					

Tutorial Questions

Implement a Q-learning model

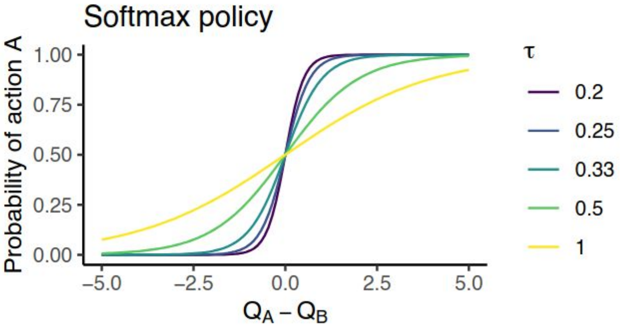
Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta\left[r - Q_t(a)\right]$$

Policy

temperature

$$P(a) \propto \exp(Q_t(a)/\tau) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$



Exercise 2: Sample actions from policy

assume:
$\eta = .9$
$\tau = .25$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A	4	4
t=2	3.6	0	A	8	4.4
t=3					
t=4					

Tutorial Questions

Implement a Q-learning model

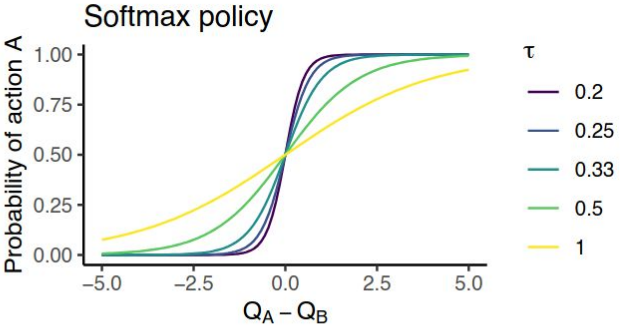
Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta\left[r - Q_t(a)\right]$$

Policy

temperature

$$P(a) \propto \exp(Q_t(a)/\tau) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$



Exercise 2: Sample actions from policy

assume:
$\eta = .9$
$\tau = .25$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A	4	4
t=2	3.6	0	A	8	4.4
t=3	7.56				
t=4					

Tutorial Questions

Implement a Q-learning model

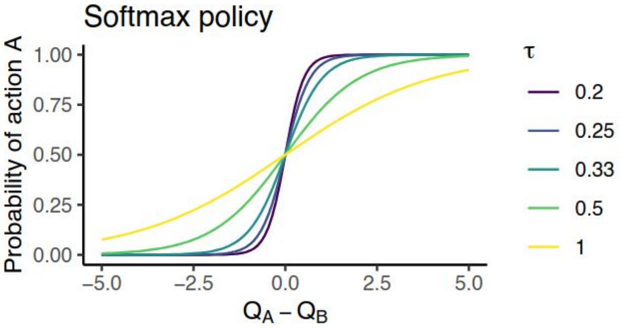
Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta \left[r - Q_t(a) \right]$$

Policy

temperature

$$P(a) \propto \exp(Q_t(a)/\tau) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$



Exercise 2: Sample actions from policy

assume:

$\eta = .9$

$\tau = .25$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A	4	4
t=2	3.6	0	A	8	4.4
t=3	7.56	0			
t=4					

Tutorial Questions

Implement a Q-learning model

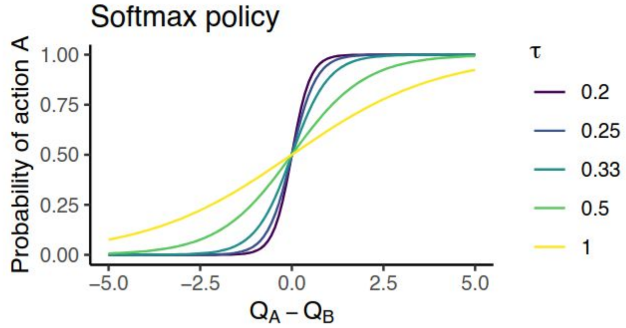
Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta \left[r - Q_t(a) \right]$$

Policy

temperature

$$P(a) \propto \exp(Q_t(a)/\tau) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$



Exercise 2: Sample actions from policy

assume:
$\eta = .9$
$\tau = .25$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A	4	4
t=2	3.6	0	A	8	4.4
t=3	7.56	0	A		
t=4					

Tutorial Questions

Implement a Q-learning model

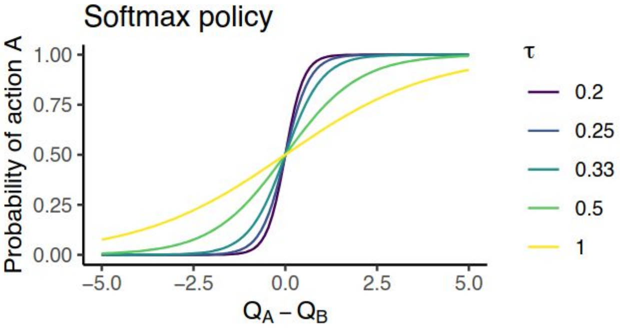
Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta \left[r - Q_t(a) \right]$$

Policy

temperature

$$P(a) \propto \exp(Q_t(a)/\tau) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$



Exercise 2: Sample actions from policy

assume:
$\eta = .9$
$\tau = .25$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A	4	4
t=2	3.6	0	A	8	4.4
t=3	7.56	0	A	5	
t=4					

Tutorial Questions

Implement a Q-learning model

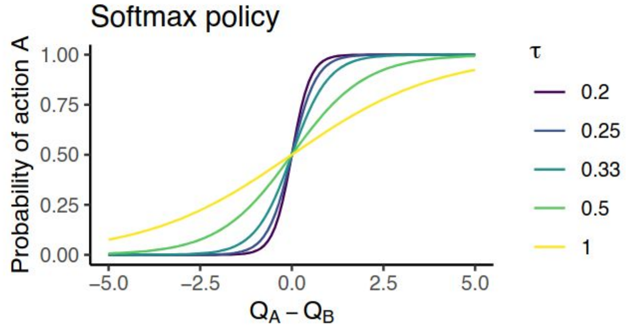
Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta \left[r - Q_t(a) \right]$$

Policy

temperature

$$P(a) \propto \exp(Q_t(a)/\tau) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$



Exercise 2: Sample actions from policy

assume:
$\eta = .9$
$\tau = .25$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A	4	4
t=2	3.6	0	A	8	4.4
t=3	7.56	0	A	5	-2.56
t=4					

Tutorial Questions

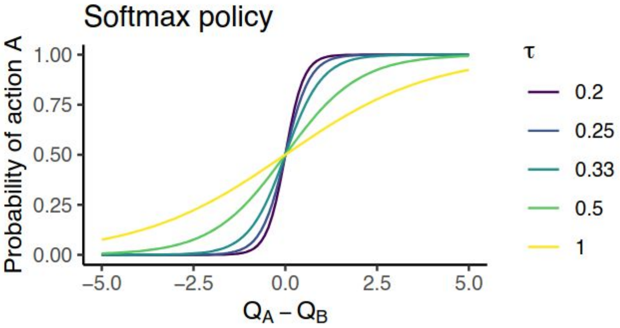
Implement a Q-learning model

Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta \left[r - Q_t(a) \right]$$

Policy

$$P(a) \propto \exp(Q_t(a)/\underbrace{\tau}_{\text{temperature}}) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$



Exercise 2: Sample actions from policy

assume:
$\eta = .9$
$\tau = .25$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A	4	4
t=2	3.6	0	A	8	4.4
t=3	7.56	0	A	5	-2.56
t=4	5.256				

Tutorial Questions

Implement a Q-learning model

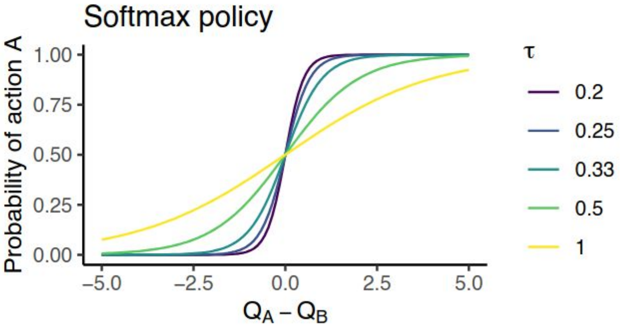
Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta \left[r - Q_t(a) \right]$$

Policy

temperature

$$P(a) \propto \exp(Q_t(a)/\tau) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$



Exercise 2: Sample actions from policy

A B

assume:
$\eta = .9$
$\tau = .25$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A	4	4
t=2	3.6	0	A	8	4.4
t=3	7.56	0	A	5	-2.56
t=4	5.256	0			

Tutorial Questions

Implement a Q-learning model

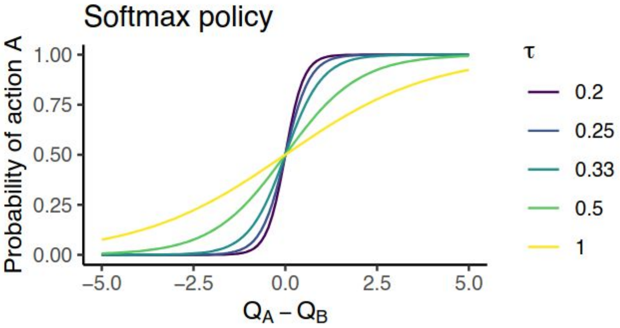
Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta \left[r - Q_t(a) \right]$$

Policy

temperature

$$P(a) \propto \exp(Q_t(a)/\tau) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$



Exercise 2: Sample actions from policy

assume:

$\eta = .9$

$\tau = .25$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A	4	4
t=2	3.6	0	A	8	4.4
t=3	7.56	0	A	5	-2.56
t=4	5.256	0	A		

Tutorial Questions

Implement a Q-learning model

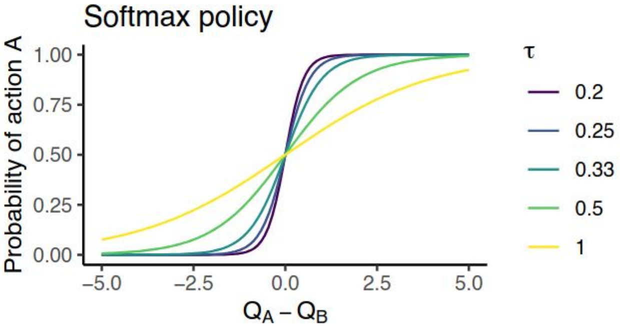
Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta \left[r - Q_t(a) \right]$$

Policy

temperature

$$P(a) \propto \exp(Q_t(a)/\tau) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$



Exercise 2: Sample actions from policy

assume:
$\eta = .9$
$\tau = .25$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A	4	4
t=2	3.6	0	A	8	4.4
t=3	7.56	0	A	5	-2.56
t=4	5.256	0	A	6	

Tutorial Questions

Implement a Q-learning model

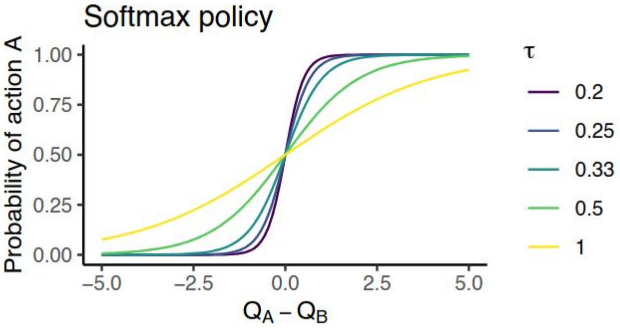
Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta \left[r - Q_t(a)\right]$$

Policy

temperature

$$P(a) \propto \exp(Q_t(a)/\tau) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$



Exercise 2: Sample actions from policy

assume:

$\eta = .9$

$\tau = .25$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A	4	4
t=2	3.6	0	A	8	4.4
t=3	7.56	0	A	5	-2.56
t=4	5.256	0	A	6	0.744

Tutorial Questions

Implement a Q-learning model

Python

```python
# Model parameters
K = 2  # number of arms
alpha = 0.9  # learning rate
tau = 1  # Softmax temperature
Qvec = np.zeros(K)  # Prior initialization of Q-values

# Softmax policy function
def softmax(Qvec, tau):
    p = np.exp(Qvec / tau)
    p /= np.sum(p)  # Normalize to sum to 1
    return p

# Simulate data
sim_data = []

for t in range(1, 51):  # Loop through 50 trials
    p = softmax(Qvec, tau)  # Compute softmax policy
    action = np.random.choice(K, p=p)  # Sample action based on probabilities
    reward = bandit_generator([action])[0]  # Generate reward from bandit
    Qvec[action] += alpha * (reward - Qvec[action])  # Update Q-values
     # Record trial data
    chosen = np.zeros(K)
    chosen[action] = 1  # 1 = chosen, 0 = not
    trial_data = {
        'trial': t,
        'Q_values': Qvec.copy(),
        'action': action,
        'chosen': list(chosen),
        'reward': reward
    }
```

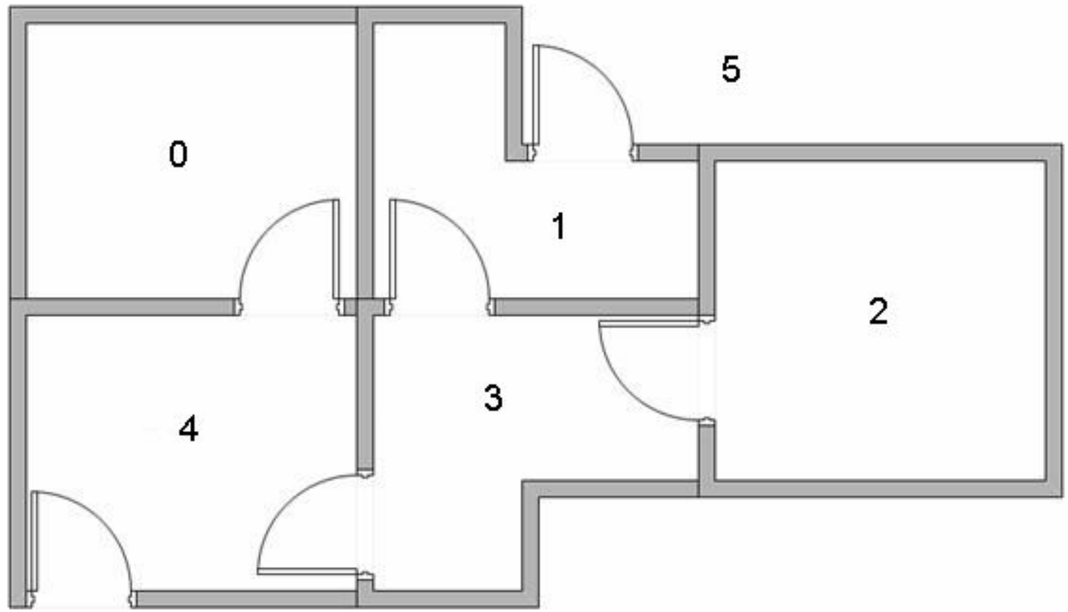
R

```r
#Model parameters
k <- 2 #number of arms
alpha <- .9 #learning rate
tau <- 1 #Softmax temperature
Qvec <-  rep(0,k) #prior initialization of Q-values

#Softmax policy

softmax <- function(Qvec, tau){

  p <- exp(Qvec/tau)

  p <- p/sum(p) #normalize to sum to 1

  return(p)

}

#Now simulate data

simDF <- data.frame()

for (t in 1:50){ #loop through trials

      p <- softmax(Qvec,tau) #compute softmax policy

      action <- sample(1:k,size = 1, prob=p) #sample action

      reward <- banditGenerator(action) #generate reward

      Qvec[action] <- Qvec[action] + alpha*(reward - Qvec[action]) #update q-values

      chosen <- rep(0, k) #create an index for the chosen option

      chosen[action]<- 1 #1 = chosen, 0 = not

      trialDF <- data.frame(trial = t, Q = Qvec, action = 1:k, chosen = chosen, reward =
```

Tutorial Questions

Let's consider a situation in which a robot is placed inside a building that has a floorplan like that shown in the following image.

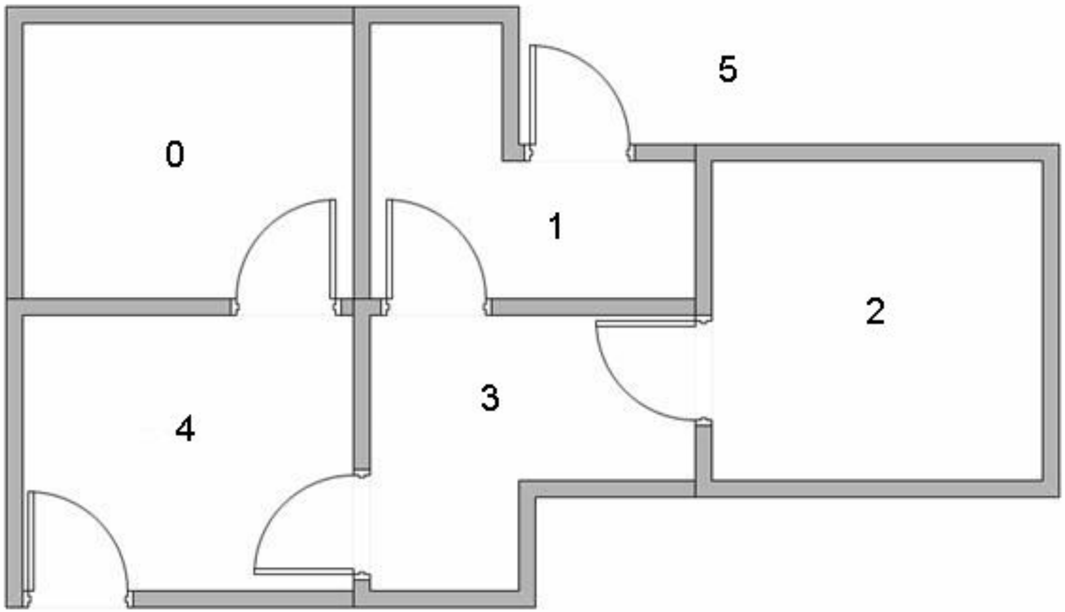
We can characterize this space as an MDP, where each state represents one room in the building (or outside, e.g., room 5) and where the agent can transition between rooms by moving either north, south, east, or west. The agent cannot stay in the same state from time step to time step, except once it is outside.



Tutorial Questions

Let's consider a situation in which a robot is placed inside a building that has a floorplan like that shown in the following image.

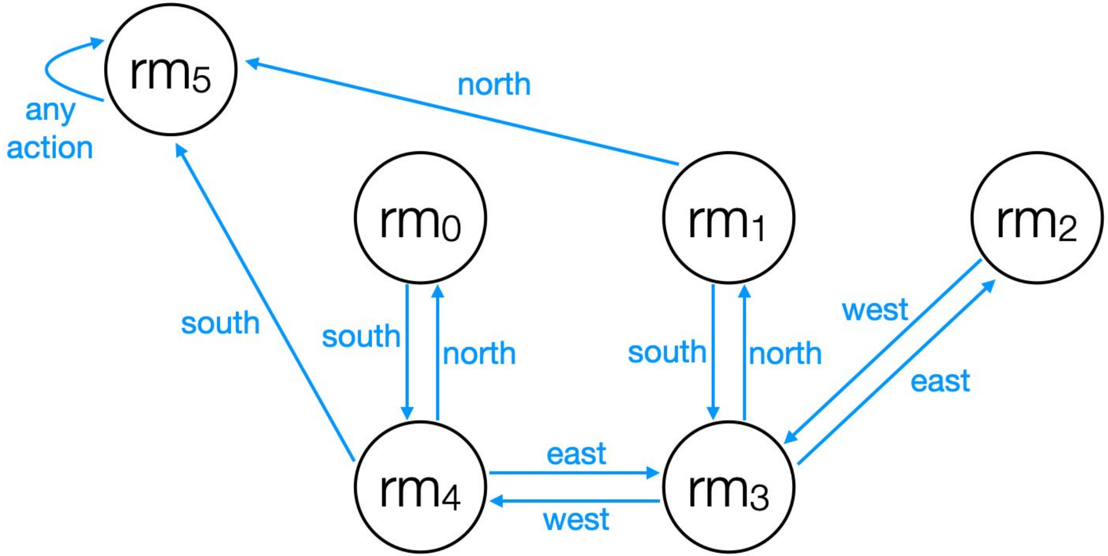
Draw a graph with nodes corresponding to states and edges to - state transitions.



Tutorial Questions

Let's consider a situation in which a robot is placed inside a building that has a floorplan like that shown in the following image.

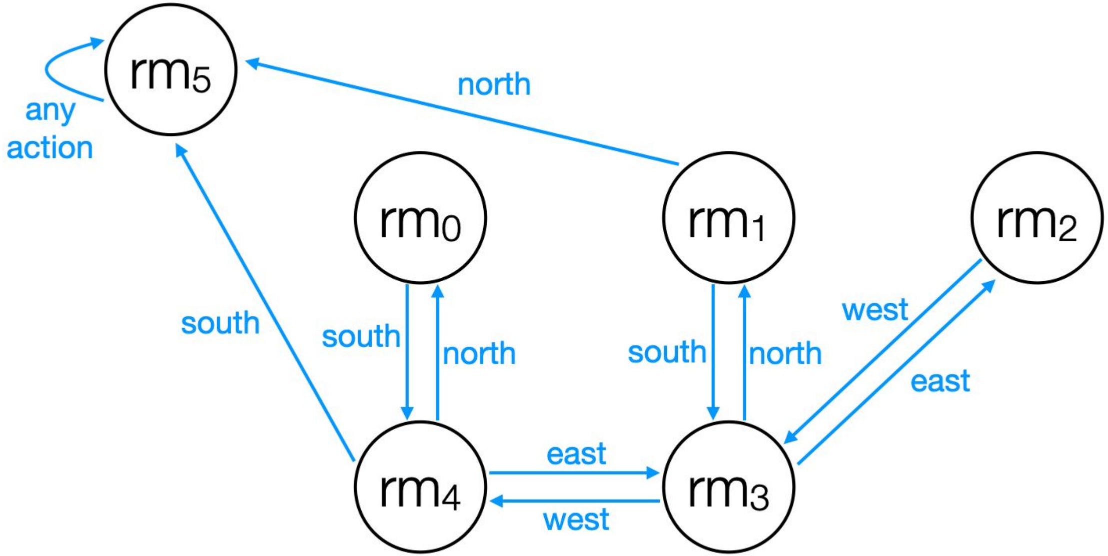
Draw a graph with nodes corresponding to states and edges to - state transitions.



Tutorial Questions

The agent receives a reward of 100 when it transitions outside from either room 1 or room 4. Additionally, the agent continues reaping rewards once it's already outside every time step.

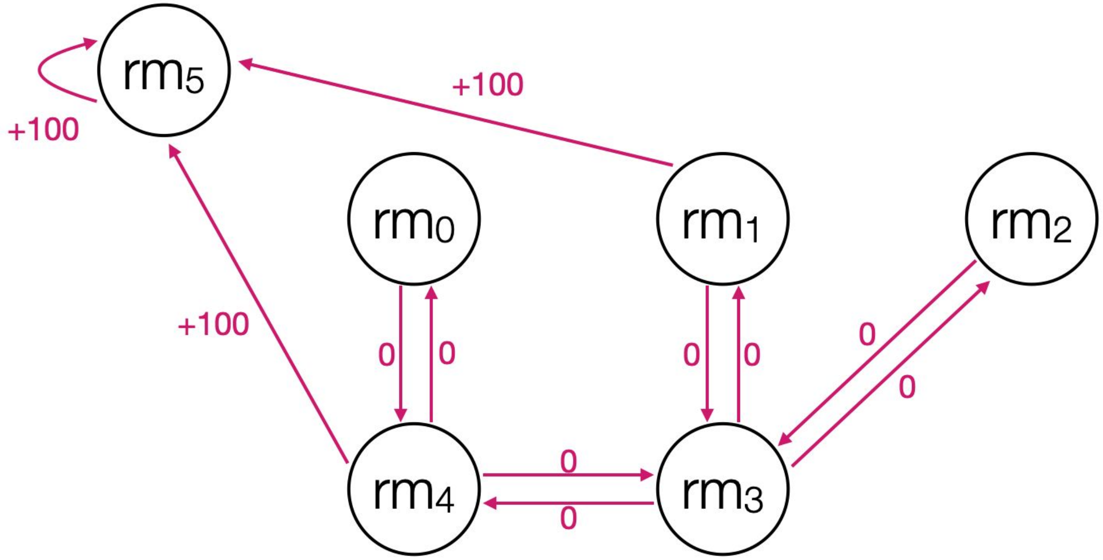
Add the rewards to the transitions



Tutorial Questions

The agent receives a reward of 100 when it transitions outside from either room 1 or room 4. Additionally, the agent continues reaping rewards once it's already outside every time step.

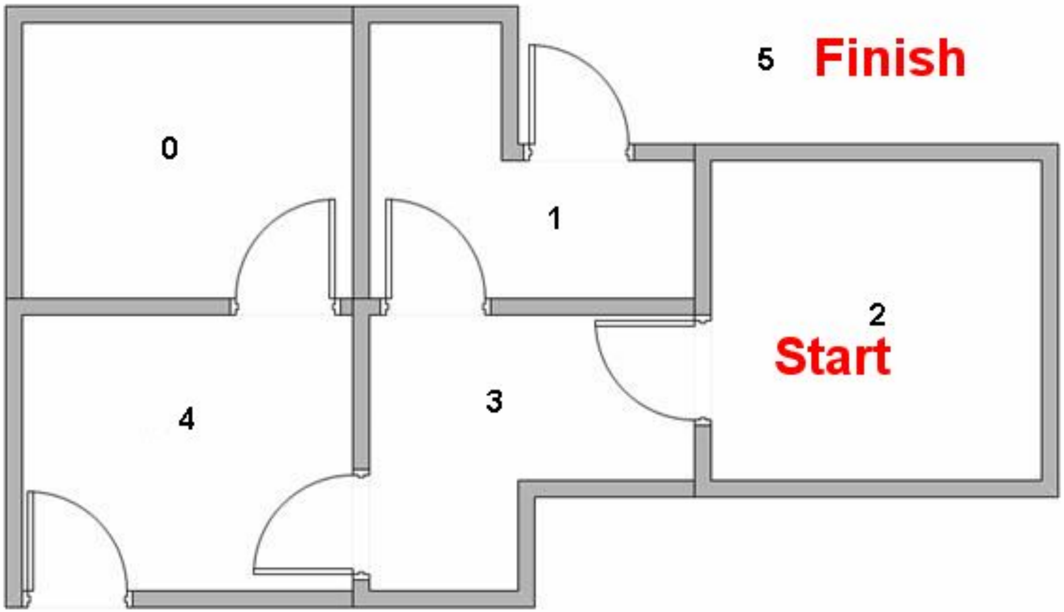
Add the rewards to the transitions



Tutorial Questions

Our robot's goal in this world is to get outside (room 5) from its starting position in room 2.

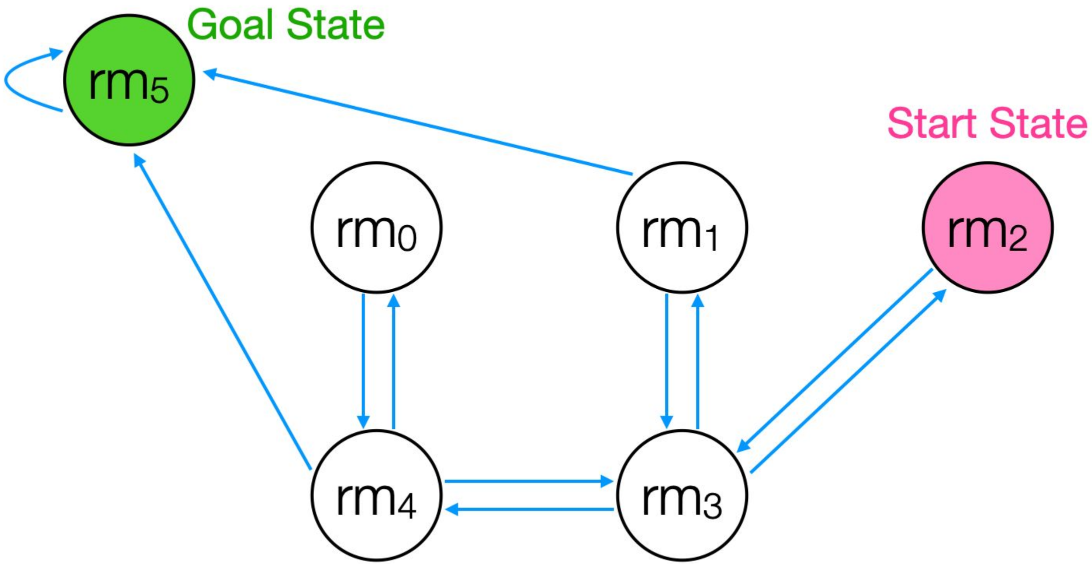
Make the necessary changes to the previously drawn graph



Tutorial Questions

Our robot's goal in this world is to get outside (room 5) from its starting position in room 2.

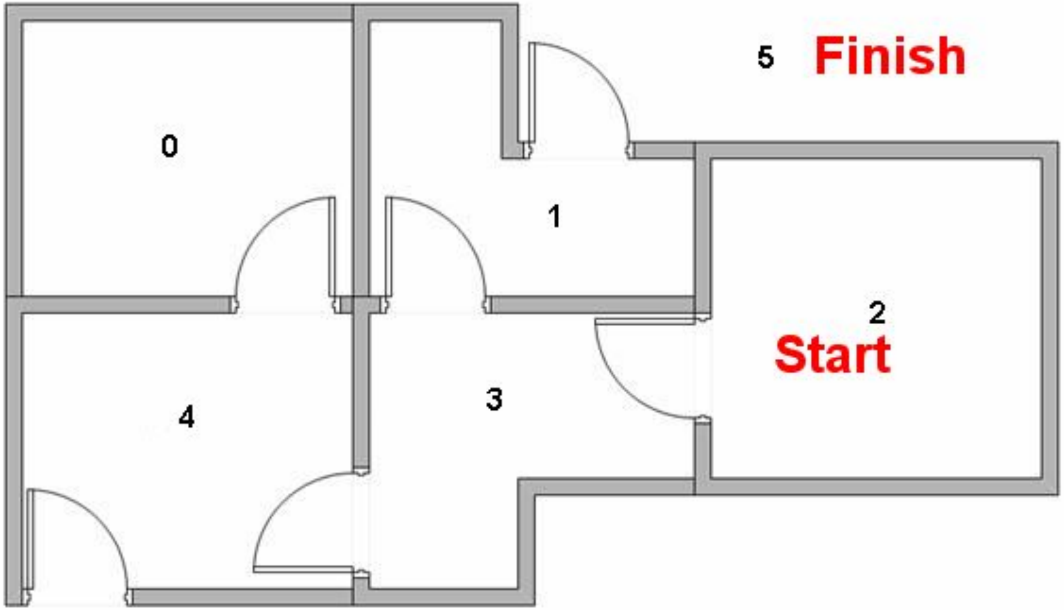
Make the necessary changes to the previously drawn graph



Tutorial Questions

Build a table / matrix to represent $Q(s_t,a_t)$

- rows represent the world states (rooms)
- columns represent actions that the robot can take.
- all valid state-action pairs are initialized to 0
- invalid state-action pairs are initialized to -1



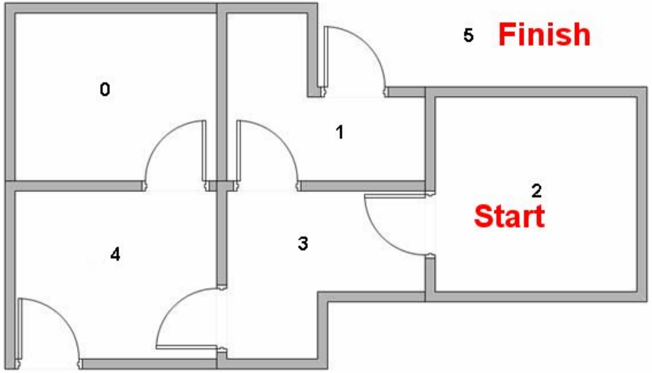
Tutorial Questions

Your goal in this exercise is to update $Q(s_t,a_t)$ according to the Q-learning algorithm. In this exercise we will make the following assumptions:

- $\alpha = 1$
- $\gamma = 0.8$
- Transitions are always successful
- $Q(s_t,a_t) \leftarrow \alpha\,(r_t + \gamma \max_a Q(s_{t+1},a)) = r_t + 0.8 \max_a Q(s_{t+1},a)$

	North	South	East	West
rm0	-1	0	-1	-1
rm1	0	0	-1	-1
rm2	-1	-1	-1	0
rm3	0	-1	0	0
rm4	0	0	0	-1
rm5	0	0	0	0

Tutorial Questions

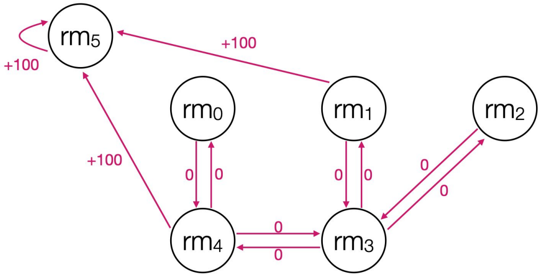


For each action within each trajectory, update according to the Q-learning algorithm after each trajectory (assume that your agent always starts in room 2 at the beginning of each trajectory).

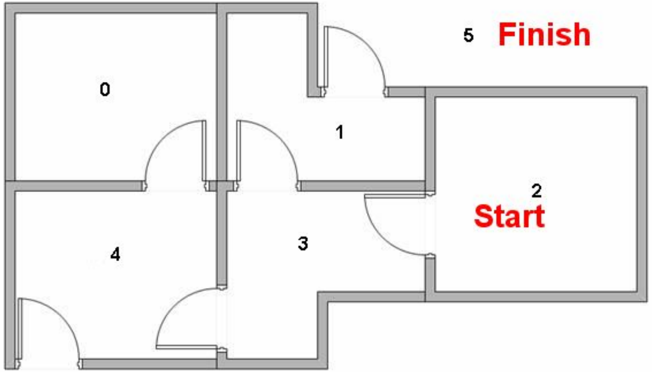
- Trajectory 1: west, west, south, north

$Q(s_t,a_t) \leftarrow \alpha\,(r_t + \gamma \max_a Q(s_{t+1},a)) = r_t + 0.8 \max_a Q(s_{t+1},a)$

	North	South	East	West
rm0	-1	0	-1	-1
rm1	0	0	-1	-1
rm2	-1	-1	-1	0
rm3	0	-1	0	0
rm4	0	0	0	-1
rm5	0	0	0	0



Tutorial Questions

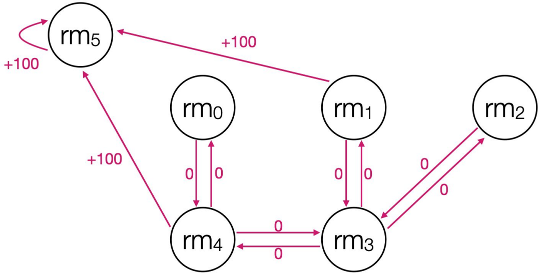


For each action within each trajectory, update according to the Q-learning algorithm after each trajectory (assume that your agent always starts in room 2 at the beginning of each trajectory).

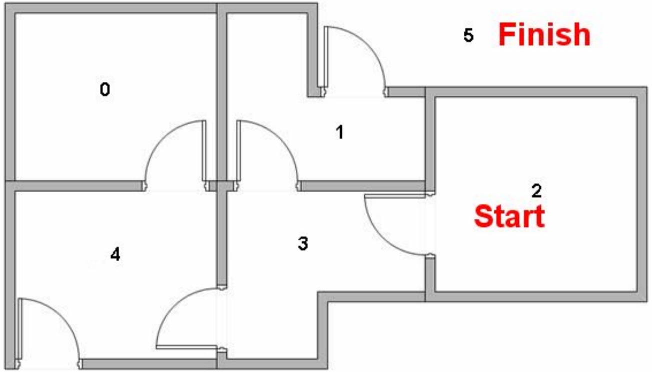
- Trajectory 1: west, west, south, north

$Q(s_t,a_t) \leftarrow \alpha\ (r_t + \gamma \max_a Q(s_{t+1},a)) = r_t + 0.8 \max_a Q(s_{t+1},a)$

	North	South	East	West
rm0	-1	0	-1	-1
rm1	0	0	-1	-1
rm2	-1	-1	-1	**0**
rm3	0	-1	0	**0**
rm4	0	**100**	0	-1
rm5	**100**	0	0	0



Tutorial Questions

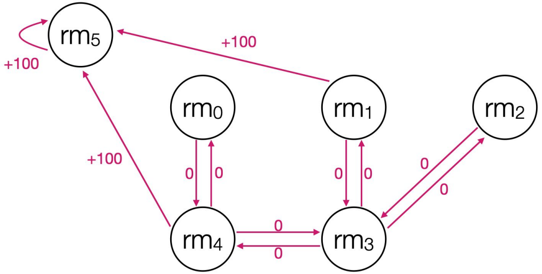


For each action within each trajectory, update according to the Q-learning algorithm after each trajectory (assume that your agent always starts in room 2 at the beginning of each trajectory).

- Trajectory 2: west, north, north, south

$Q(s_t,a_t) \leftarrow \alpha\ (r_t + \gamma \max_a Q(s_{t+1},a)) = r_t + 0.8 \max_a Q(s_{t+1},a)$

	North	South	East	West
rm0	-1	0	-1	-1
rm1	0	0	-1	-1
rm2	-1	-1	-1	0
rm3	0	-1	0	0
rm4	0	100	0	-1
rm5	100	0	0	0

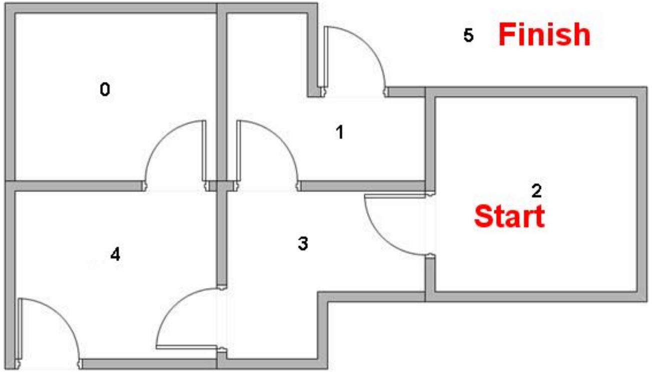


Tutorial Questions

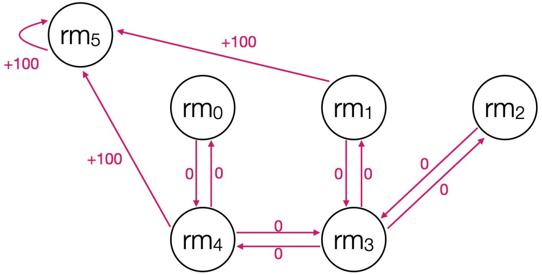
For each action within each trajectory, update according to the Q-learning algorithm after each trajectory (assume that your agent always starts in room 2 at the beginning of each trajectory).

- Trajectory 2: west, north, north, south

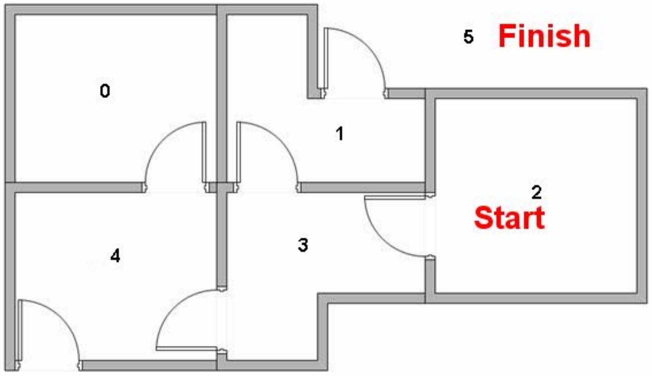
$Q(s_t,a_t) \leftarrow \alpha\,(r_t + \gamma \max_a Q(s_{t+1},a)) = r_t + 0.8 \max_a Q(s_{t+1},a)$



	North	South	East	West
rm0	-1	0	-1	-1
rm1	**180**	0	-1	-1
rm2	-1	-1	-1	**0**
rm3	**0**	-1	0	0
rm4	0	100	0	-1
rm5	100	**180**	0	0



Tutorial Questions

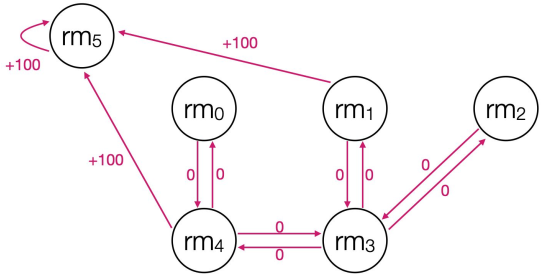


For each action within each trajectory, update according to the Q-learning algorithm after each trajectory (assume that your agent always starts in room 2 at the beginning of each trajectory).

- Trajectory 3: west, west, north, south, south

$Q(s_t,a_t) \leftarrow \alpha\ (r_t + \gamma \max_a Q(s_{t+1},a)) = r_t + 0.8 \max_a Q(s_{t+1},a)$

	North	South	East	West
rm0	-1	0	-1	-1
rm1	180	0	-1	-1
rm2	-1	-1	-1	0
rm3	0	-1	0	0
rm4	0	100	0	-1
rm5	100	180	0	0

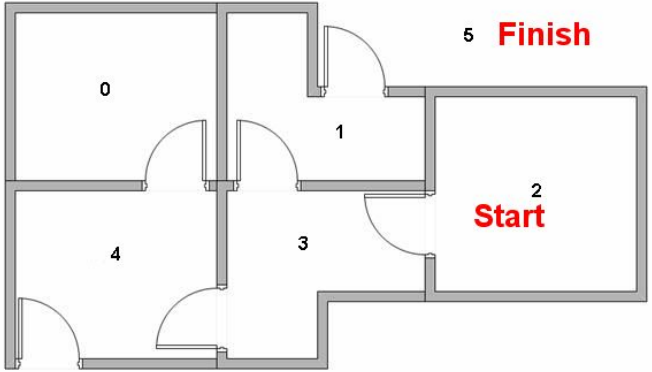


Tutorial Questions

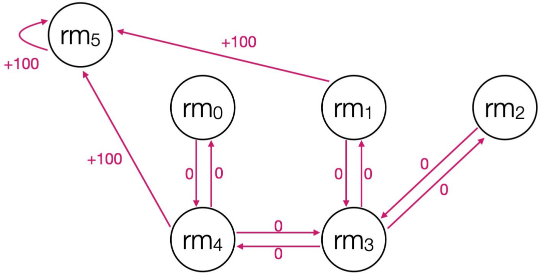
For each action within each trajectory, update according to the Q-learning algorithm after each trajectory (assume that your agent always starts in room 2 at the beginning of each trajectory).

- Trajectory 3: west, west, north, south, south

$$Q(s_t,a_t) \leftarrow \alpha\,(r_t + \gamma \max_a Q(s_{t+1},a)) = r_t + 0.8 \max_a Q(s_{t+1},a)$$



	North	South	East	West
rm0	-1	**80**	-1	-1
rm1	180	0	-1	-1
rm2	-1	-1	-1	**0**
rm3	0	-1	0	**80**
rm4	**0**	**244**	0	-1
rm5	100	180	0	0

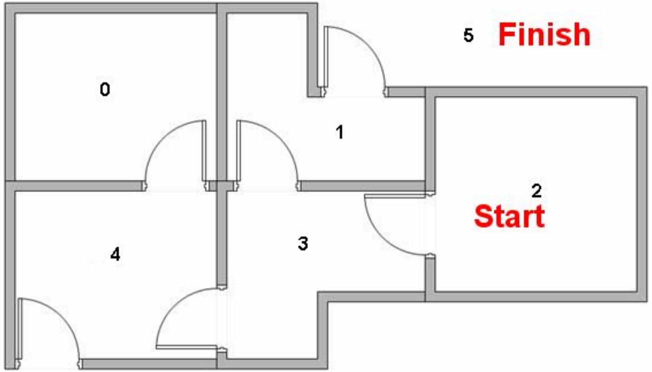


Tutorial Questions

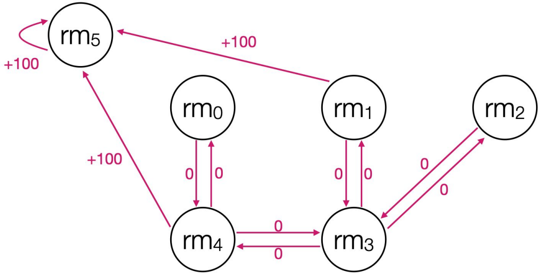
For each action within each trajectory, update according to the Q-learning algorithm after each trajectory (assume that your agent always starts in room 2 at the beginning of each trajectory).

- Trajectory 4: west, west, east, north, north

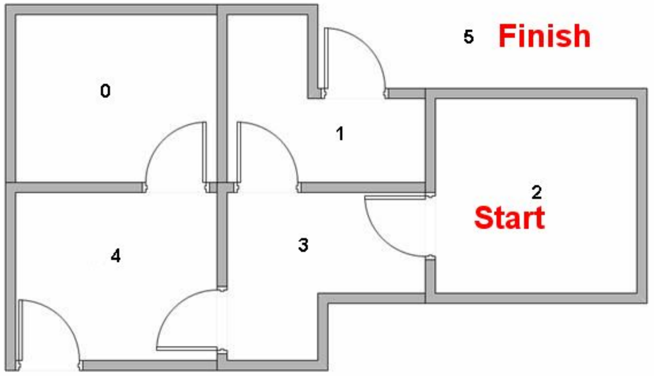
$$Q(s_t,a_t) \leftarrow \alpha\,(r_t + \gamma \max_a Q(s_{t+1},a)) = r_t + 0.8 \max_a Q(s_{t+1},a)$$



	North	South	East	West
rm0	-1	80	-1	-1
rm1	180	0	-1	-1
rm2	-1	-1	-1	0
rm3	0	-1	0	80
rm4	0	244	0	-1
rm5	100	180	0	0



Tutorial Questions



For each action within each trajectory, update according to the Q-learning algorithm after each trajectory (assume that your agent always starts in room 2 at the beginning of each trajectory).

- Trajectory 4: west, west, east, north, north

$Q(s_t,a_t) \leftarrow \alpha\,(r_t + \gamma \max_a Q(s_{t+1},a)) = r_t + 0.8 \max_a Q(s_{t+1},a)$

	North	South	East	West
rm0	-1	80	-1	-1
rm1	**244**	0	-1	-1
rm2	-1	-1	-1	**64**
rm3	**144**	-1	0	**195.2**
rm4	0	244	**156.16**	-1
rm5	100	180	0	0