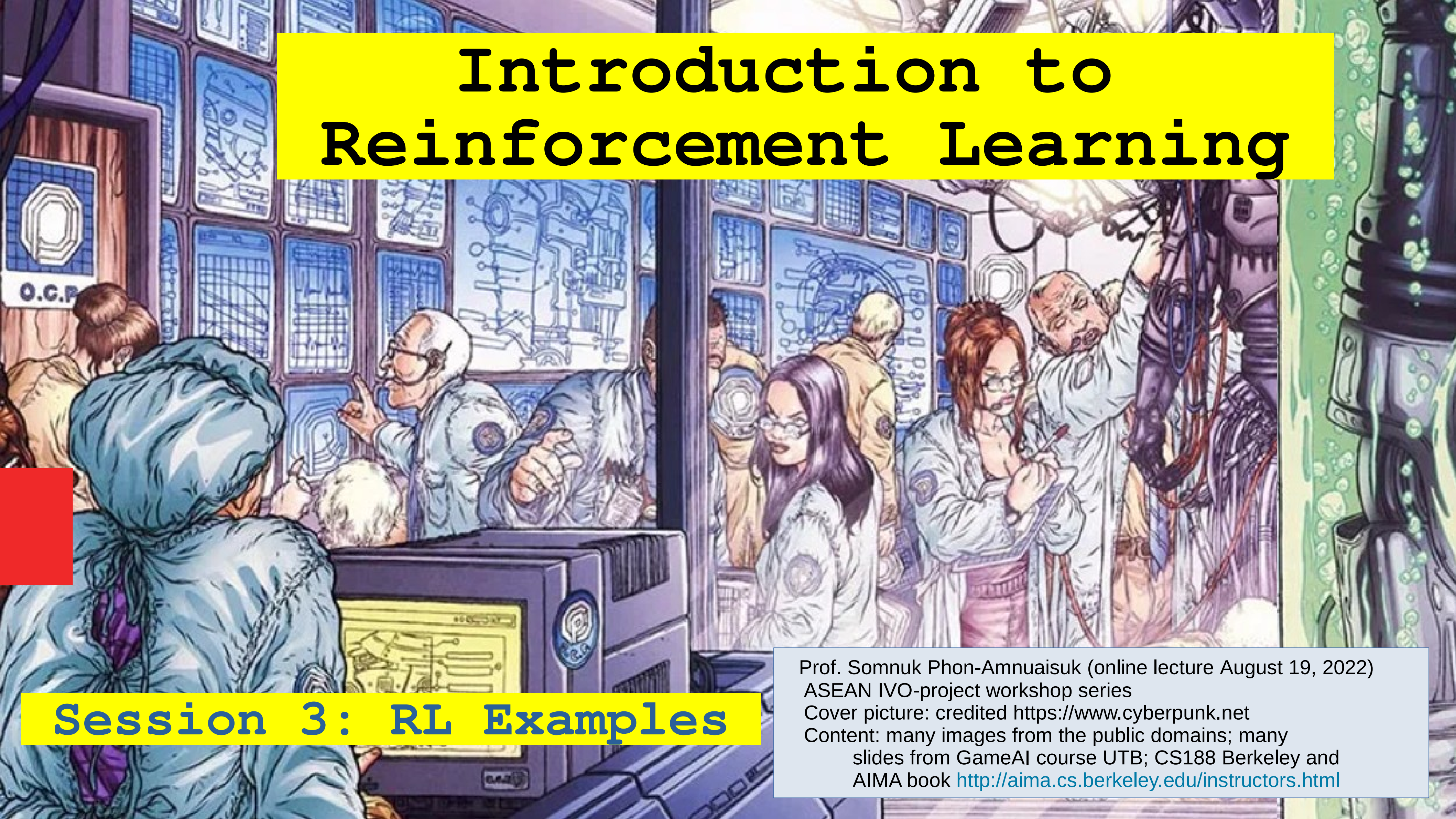


Introduction to Reinforcement Learning

Session 3: RL Examples

Prof. Somnuk Phon-Amnuaisuk (online lecture August 19, 2022)
ASEAN IVO-project workshop series
Cover picture: credited https://www.cyberpunk.net
Content: many images from the public domains; many slides from GameAI course UTB; CS188 Berkeley and AIMA book http://aima.cs.berkeley.edu/instructors.html


meme
A meme (/ˈ mi: m/ meem), a neologism coined by Richard Dawkins, is "an idea, behavior, or style that spreads from person to person within a culture". A meme acts as a unit for carrying cultural ideas, symbols, or practices that can be transmitted from one mind to another through writing, speech, gestures, rituals, or other imitable phenomena with a mimicked theme.

Outline

- Designing of states and actions, examples from OpenAI Gym and Unity Game Engine
- Look at an implementation from Dung-Yi, Chao, Purdue University: Reinforcement Learning on Route Planning through Google map for Self-driving System

Q Learning

$$Q^{new}(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Bigg(\underbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t,a_t)}_{\text{old value}}\Bigg)}^{\text{temporal difference}}$$

In 2014 Google DeepMind patented an application of Q-learning to deep learning, titled "deep reinforcement learning" or "deep Q-learning" that can play Atari 2600 games at expert human levels.

Initialized

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	327	0	0	0	0	0	0
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	499	0	0	0	0	0	0

Training

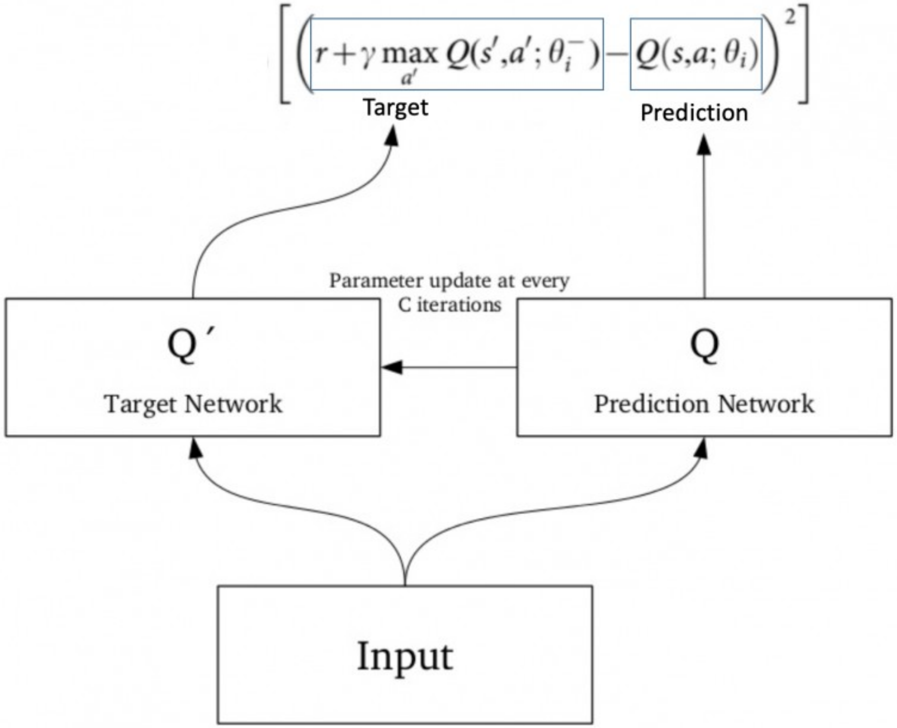
Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	328	-2.30108105	-1.97092096	-2.30357004	-2.20591839	-10.3607344	-8.5583017
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	499	9.96984239	4.02706992	12.96022777	29	3.32877873	3.38230603

Deep Q Learning

- The DeepMind system used a deep convolutional neural network, with layers of tiled convolutional filters to mimic the effects of receptive fields.
- **Reinforcement learning is unstable** or divergent when a nonlinear function approximator such as a neural network is used to represent Q. This instability comes from the correlations present in the sequence of observations, the fact that small updates to Q may significantly change the policy and the data distribution, and the correlations between Q and the target values.
- The technique used **experience replay**, a biologically inspired mechanism that uses a random sample of prior actions instead of the most recent action to proceed. This removes correlations in the observation sequence and smooths changes in the data distribution. Iterative updates adjust Q towards target values that are only periodically updated, further reducing correlations with the target.

Deep Q Learning

$$Q^{new}(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\underbrace{\Bigg(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t,a_t)}_{\text{old value}}\Bigg)}^{\text{temporal difference}}$$

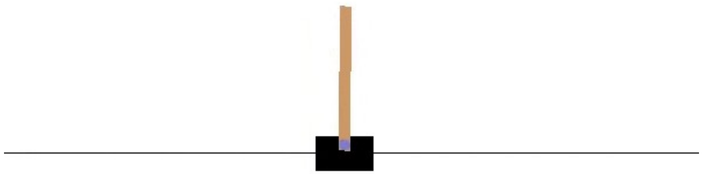


1. Select an action from possible Q-values actions, select using the epsilon-greedy policy.
2. Perform this action **a** in a state **s** and move to a new state **s’** to receive a reward **r**.
3. Record this transition in our replay buffer as <s,a,r,s’>
4. After C iterations, sample data randomly from the replay-buffer and train the ANN using fix target.
5. Update the target Q network
6. Repeat

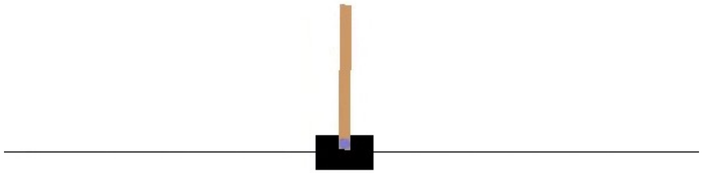
https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/

OpenAI Gym

```
import gym

env = gym.make("CartPole-v1")
observation, info = env.reset(seed=42, return_info=True)

for _ in range(1000):
    action = env.action_space.sample()
    observation, reward, done, info = env.step(action)

    if done:
        observation, info = env.reset(return_info=True)
env.close()
```



OpenAI Gym



```
# CART POLE
env = gym.make('CartPole-v0')
print(env.observation_space.low)
print(env.observation_space.high)
```

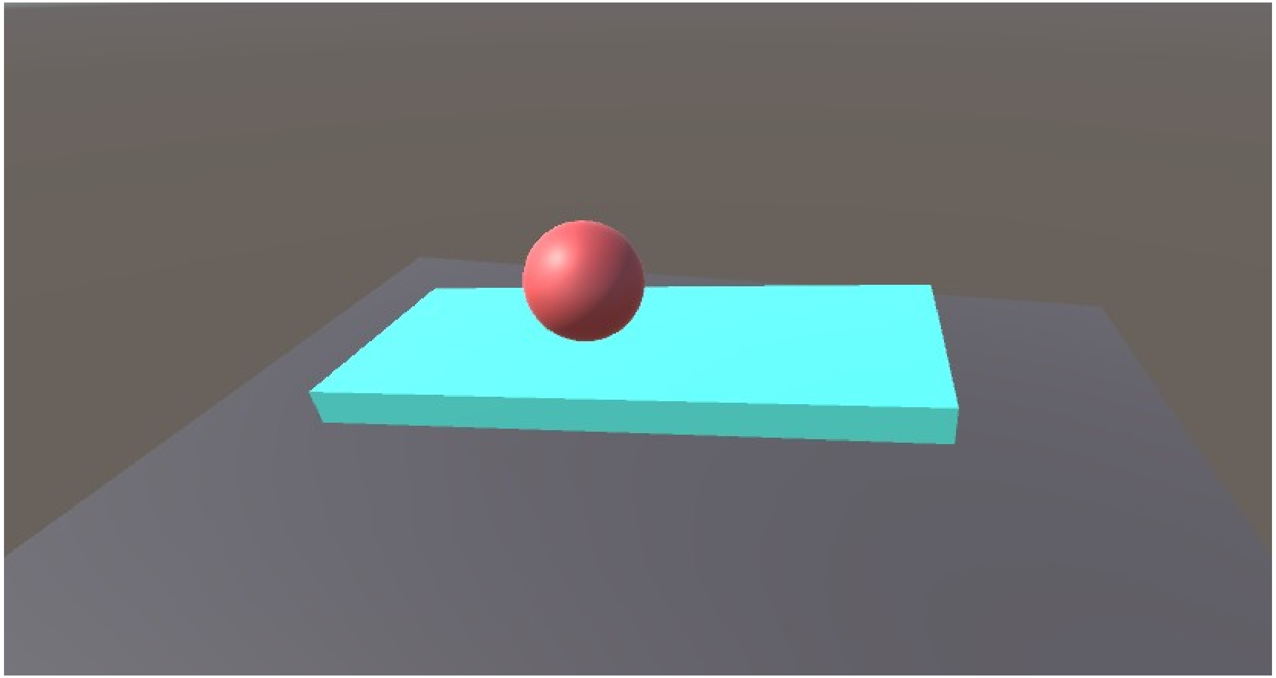
```
[-4.8000002e+00 -3.4028235e+38 -4.1887903e-01 -3.4028235e+38]
[4.8000002e+00 3.4028235e+38 4.1887903e-01 3.4028235e+38]
```

```
print(len(env.observation_space.low, env.action_space.n)
```

```
4 2
```

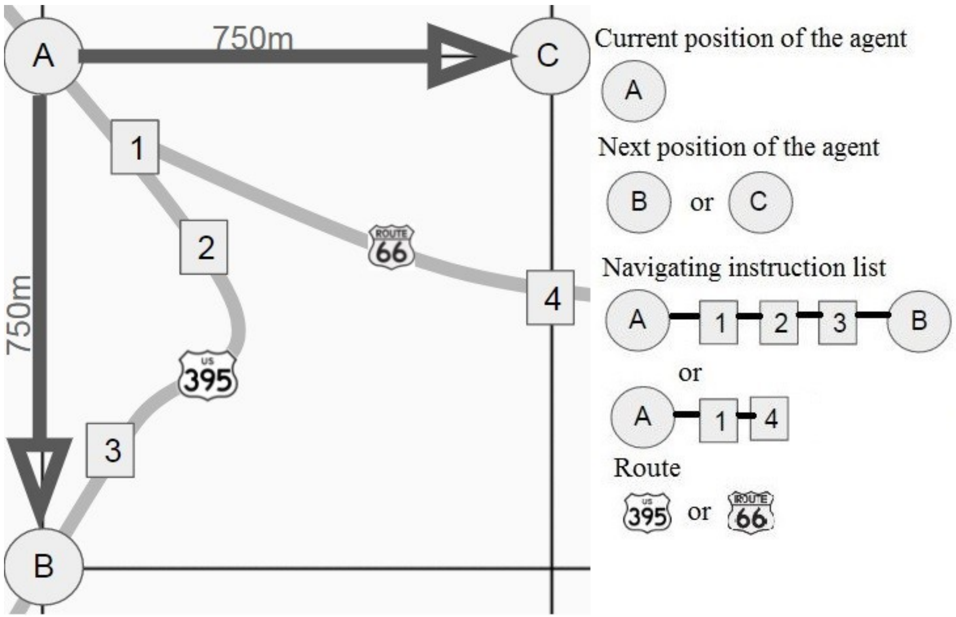
Balance Ball Example

- Game-play
- States
- Actions
- Rewards



Electrical Vehicle Route Planning on Google Map with RL

- Traditional Google API provides the best route in term of traveling duration.
- In this work, the authors find the best route with minimum energy cost and acceptable duration.
- Use elevation to approximate energy consumption among possible routes.

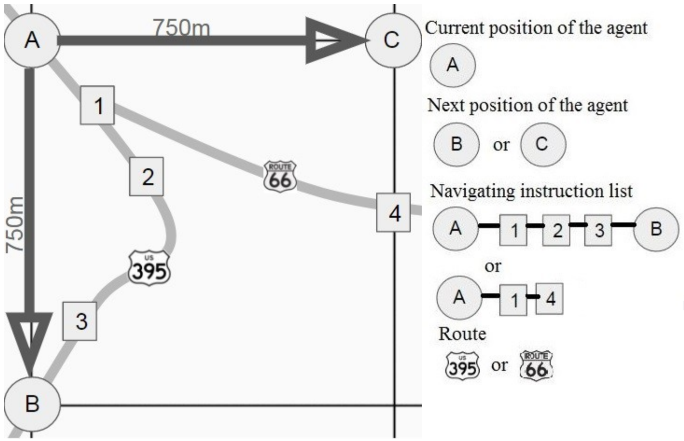


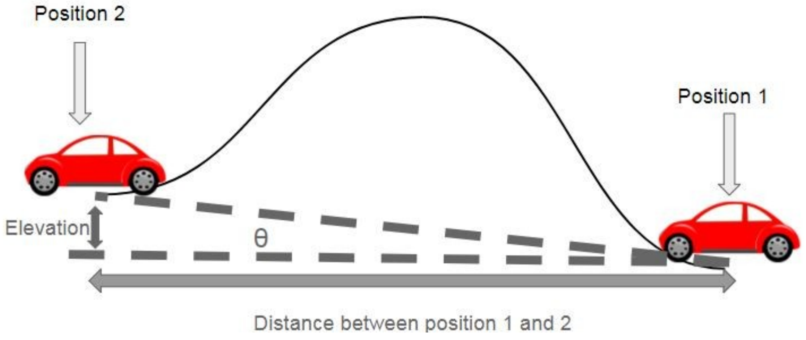
Dung-Yi, Chao (Purdue University)

Problem Formulation

Other assumptions:

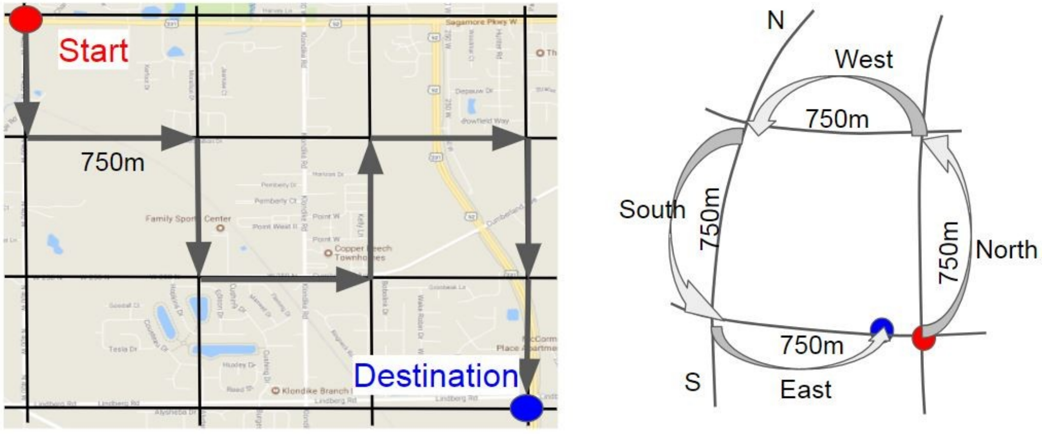
- Each action is based on the energy required to travel with 750m displacement on the grid map.
- To compute the energy required between position A and position 1, for example, we use the duration and distance between position A and position 1 to calculate the average velocity V. Combine V with the elevation, we can get the angle θ of the road and consider the height of the road as linear increasing or decreasing.
- Because we don't take regenerative braking into account in our experiment, we treat the downhill road flat.





States and Actions

- State: latitude and longitude
- Actions: N, E, S, W



(a) (b)

Rewards

- The fundamental concept of defining the reward is based on the energy consumption in one stride from the current position to the next position.
- For example, from A to B shown in the figure. The energy is calculated by the method provided previous section. We then divide the energy by 10000 and times -1. In order to minimize the number of total steps during training, we add -0.1 to each transition if the next position is reachable. In other words, the reward r for taking any reachable step will be

  $$r = -0.1 - (\text{energy consumption} / 10000).$$

Q & A