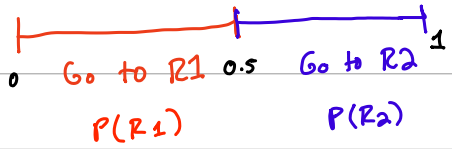
Reading 13

☆ Reinforcement Learning

- Reinforcement learning is a methodology that basically lets computers learn and adapt based on experience.

- Scenario: there are 2 restaurants we've never been to. How do we decide where to go?

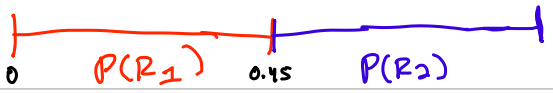
- We start with each place having an probability that we go there.



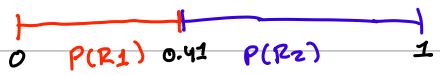
- Pick random #
  - ↳ If you get 0.7 we go to R2
    - ↳ we are satisfied, but we have low confidence if it happens every time
    - ↳ so we increase P(R2) by a little bit.
      - ↳ by using this equation:

$$\text{New } p(R2) = p(R2) + [\text{learning rate} \times (\underbrace{\text{restaurant score}}_{[0,1]} - p(R2)]$$

    - ↳ then New P(R1) = 1 − P(R2)
    - ↳ the learning rate $\alpha \in [0,1]$ influences how much the probabilities change.
      - ↳ if α is small, the probability changes minimally
      - ↳ if α is large, the probability changes more

- So now we are here: 

- Pick another random number $\in [0,1]$
  - ↳ you get 0.2 so you go to R1
    - ↳ you are unsatisfied with the food so the restaurant score is low.
    - ↳ So then you can update $P(R_1)$ with same equation as above (same learning rate)
      - ↳ update $P(R_2)$: $P(R_2) = 1 - P(R_1)$

- So now 

  - ↳ so you just keep repeating same process

- After some point $P(R_1)$ and $P(R_2)$ would stabilize.

- Terminology:
  - R_1 and R_2 are the environment
    - ↳ the environment is something we want to explore and can interact with

- The agent is whatever is exploring the environment.
- The probabilities $P(R_1)$ and $P(R_2)$ are called the policy.
- The restaurant score which we used to update the policy is called the reward.

- In reinforcement learning the goal is to modify the policy so we can maximize the reward.

- Remember the reward can change and doesn't always have to be the same.

☆ Reinforcement Learning with Neural Networks

Scenario: Same as above

- Now to solve the problem we will use a NN that takes our hunger as input and gives us $P(R_2)$ as output.

- Since the quality at R_1 and R_2 varies, we don't have ground truth.
  - ↳ so we cannot use back propogation because it requires $(\hat{y} - y)$
    - ↳ so we must train our NN using reinforcement learning.
      - ↳ there are lots of algorithms for this, we'll focus on policy gradients.

- So we can guess what the ground truth should be and use those guesses to calculate the gradient we need to descend.

- So if we input 0, our NN outputs 0.5 for $P(R_2)$
  - ↳ then pick random # to determine where to go.
    - ↳ if we get 0.2, the we go to R1
      - ↳ then we have to make a guess
        - ↳ if we guess that when hunger = 0 we should go to R_1, then ideally $P(R_1) = 1$.
          - ↳ and $P(R_2) = 0$
        - ↳ we can now quantify the difference between the ideal value and actual value of $P(R_1)$
        - ↳ if the guess is correct then the positive value of the derivative tells us to decrease the bias.
          - ↳ then you also set reward to a positive value
        - ↳ if the guess is incorrect then the negative value of the derivative tells us to increase the bias.
          - ↳ then you also set reward to a negative value.
        - ↳ so you multiply derivative by reward which points us in the updated correct direction.

↳ When we multiply a derivative that is based on an incorrect guess by a negative reward, we flip the direction and correct the mistake.

↳ We then plug the updated derivative into Gradient Descent to calculate the step size.

↳ then we run the process again with the updated policy.