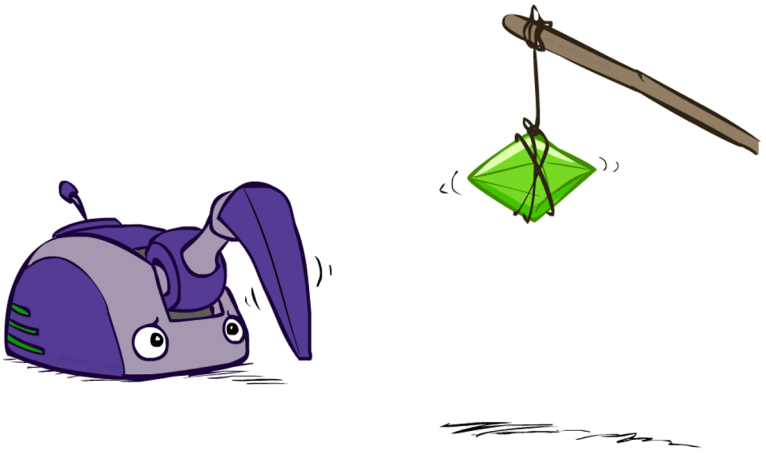
CS440 – Reinforcement Learning



Purpose: Consider an agent living in a world in which much is unknown. The agent would have to try actions and learn from experience and try not to duplicate bad actions. This is reinforcement learning.

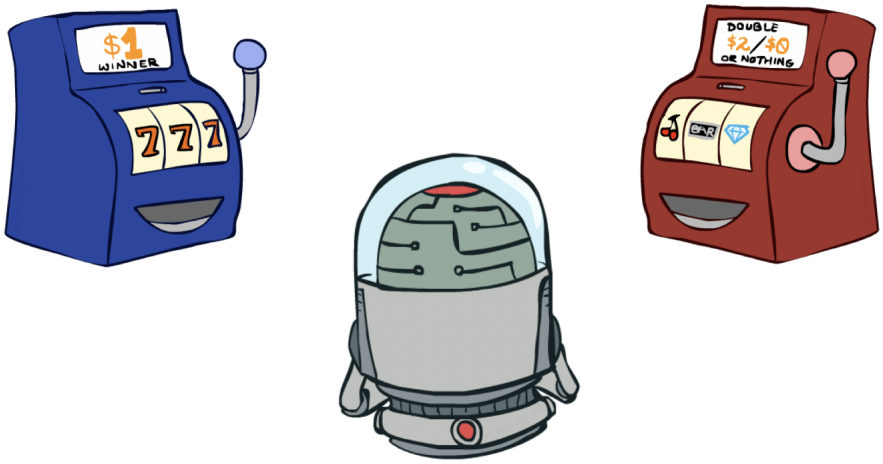
The purpose of this module is to present techniques to perform reinforcement learning.

Knowledge: This module will help you become familiar with the following content knowledge:

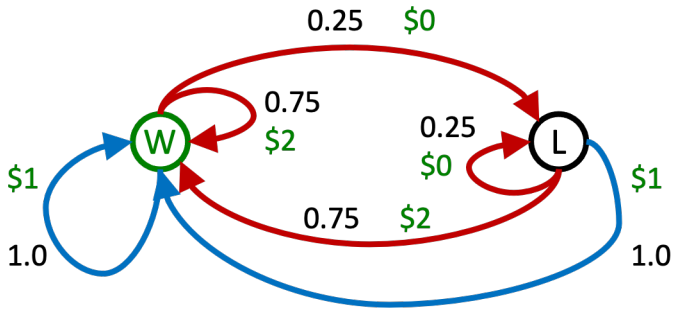
- Reinforcement learning basics
- Model-based and model free learning
- Q-learning
- Feature-based learning

Activity1 - Reinforcement Learning:
With your group perform the following tasks and answer the questions. You will be reporting your answers back to the class in 30 minutes.

1. Consider the double bandits game where our agent has a choice of playing a blue or a red machine.



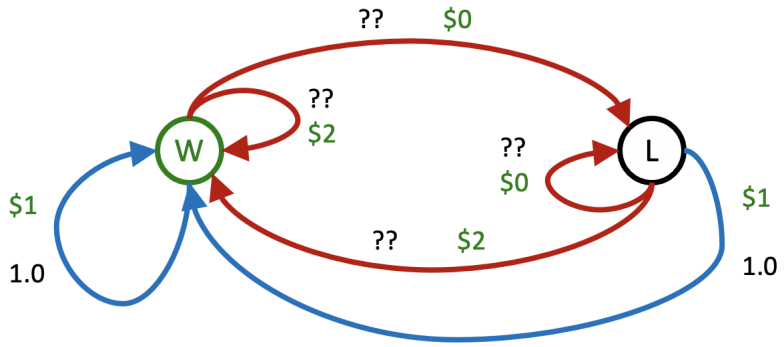
The machines have the following state diagram:



This means that if you put $1 into the blue machine you will always win $1. If you put $1 into the red machine 75% of the time you will win $2 but 25% of the time you will get nothing.

What is the expected value if you play the blue machine 100 times?
What is the expected value if you play the red machine 100 times?

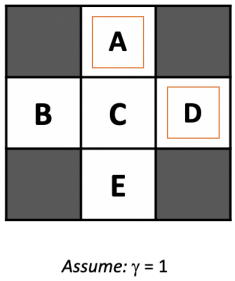
Now we are going to change the rules! The red machine's win chance has now changed but you don't know what it is.



What would you do? Well you would have to explore a bit by playing the red machine. Eventually you would use what you know and perhaps regret a bit because you would make mistakes. This becomes way more difficult because we have to learn as we go.

2. Model-based learning has a training period where you observe what happens when you try things. From the observed values you build a model about what you observe the T and R values might be.

   We have small gridworld with exits at A and D. We can perform some trials and observe the rewards. Start observing:



Observed Episodes (Training)

Episode 1	Episode 2
B, east, C, -1 C, east, D, -1 D, exit, x, +10	B, east, C, -1 C, east, D, -1 D, exit, x, +10

Episode 3	Episode 4
E, north, C, -1 C, east, D, -1 D, exit, x, +10	E, north, C, -1 C, east, A, -1 A, exit, x, -10

What did we observe about the probability T(B,east,C)?
What did we observe about T(C,east,D)?
What did we observe about T(C,east,A)?
What did we observe about the reward R(B,east,C)?
What did we observe about R(C,east,D)?
What did we observe about R(D,exit,x)?

3. The previous example was model-based learning since we were learning about and constructing the model (T and R). It is also possible to have model-free learning.

   Consider the problem of determining the expected age of students at Goucher. If we knew the model, which in this case would be the probabilities $P(a)$ of a student being age a, how would we compute the expected age?
   Now suppose we don't know the probabilities. Instead we could collect samples $[a_1, a_2, ...a_N]$ to learn students' ages. We could then get the model from those ages:

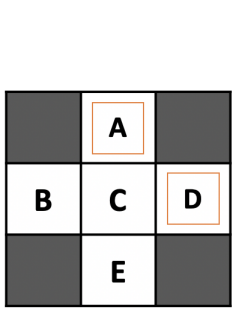
$$P(a) = \frac{num(a)}{N}$$

   Instead of learning the model, however, we could just compute the expected age as:

$$\frac{1}{N}\sum_i a_i$$

   Why does this work without learning the model?

4. A. model-free method for the gridworld is *direct evaluation*. We average the values obtained when starting at each state in the observations.



Observed Episodes (Training)

Episode 1
B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2
B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3
E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4
E, north, C, -1
C, east, A, -1
A, exit, x, -10

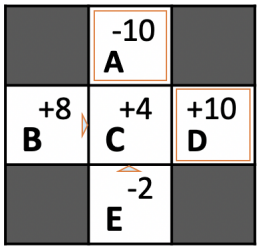
So we only have state A appear in one of the episodes which is episode 4. Therefore the average for state A would be -10.

State B appears twice. For episode 1 we would get a score of 8 (-1 to C + -1 to D + 10 on exit). For episode 2 would get a score of 8 as well. Therefore the average for B is 8.

What would be the average for C?
What would be the average for D?
What would be the average for E?

There are problems with direct evaluation. Consider the following episode with the average values.

Output Values



If both B and E go to C under this policy, do these values make sense? Why?
It takes a long time to learn good values.

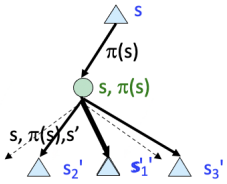
Activity2 - Learning V-values and Q-values:
With your group perform the following tasks and answer the questions. You will be reporting your answers back to the class in 30 minutes.

1. We know in gridworld we can improve our estimates with:

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R, s, \pi(s), s') + \gamma V_k^{\pi}(s')]$$

But we don't know T or R. So the idea is to take samples by actually taking the actions and then average.

Consider taking multiple samples from state s:
$sample_1 = R(s, \pi(s), s'_1) + \gamma V_k^{\pi}(s'_1)$
$sample_2 = R(s, \pi(s), s'_2) + \gamma V_k^{\pi}(s'_2)$
...
$sample_n = R(s, \pi(s), s'_n) + \gamma V_k^{\pi}(s'_n)$



We could then average the results:

$$V_{k+1}^{\pi}(s) \leftarrow \frac{1}{n} \sum_{i} sample_i$$

Why is it difficult to get these multiple samples from state s?

2. So, instead of just looking at samples from a particular state we will learn from every experience in a technique called *temporal difference learning*. We will update $V(s)$ each time we experience a transition (s, a, s', r) which means that likely outcomes s' will contribute updates more often – just what we want.

$sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$

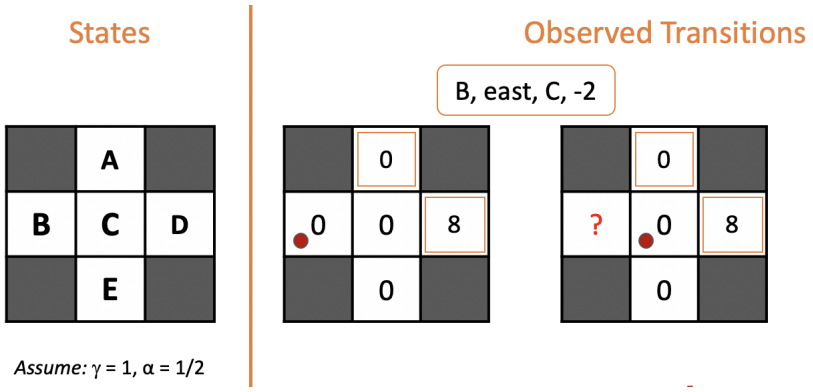
Now update $V(s)$:
$V^{\pi}(s) \leftarrow (1 - \alpha)V^{\pi}(s) + (\alpha)sample$

Or rewrite as:
$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$

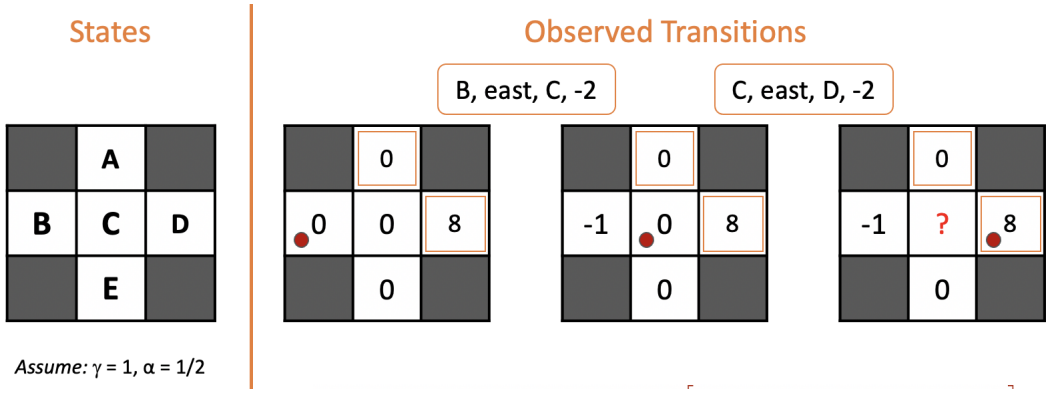
What is the purpose of α in the update of $V^{\pi}(s)$?

Work through an example of temporal difference learning:

$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha)sample$



What is $V(B)$?



What is $V(C)$?

3. Instead of learning the V-values it is advantageous to learn the Q-values. Why?

   We know:

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q_k(s'a')]$$

   Again, we don't know T and R so we will compute the average as we observe samples:

   We receive a sample (s,a,r,s') which suggests that

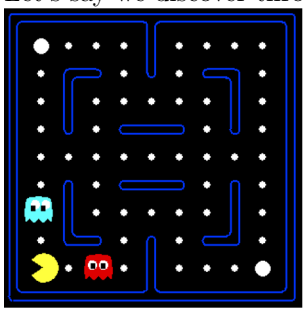
$$Q(s,a) \approx r + \gamma \max_{a'} Q(s',a')$$

   But we want to keep a running average:

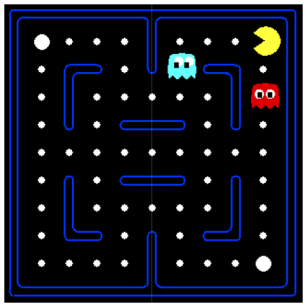
$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a')]$$

   Basic Q-learning keeps a table of all q-values. But think about pacman. Is it realistic to learn about every single state?

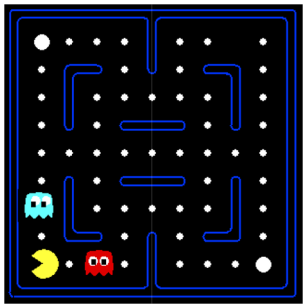
4. Let's say we discover through experience that this is a bad state:



In naive q-learning we know nothing about this state:



or even this one:



A solution is to describe a state using a vector of features. Features are functions from states to real numbers (often 0/1) that capture important properties of the state.

What are some possible relevant features for pacman?

5. We can approximate the Q-value with a weighted sum of features:

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + ... + w_n f_n(s,a)$$

Then we will adjust the weights as we learn more information. When we observe the transition (s,a,r,s'):

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

Essentially, if something bad happens we blame the features that are on and disprefer all states with that state's features.