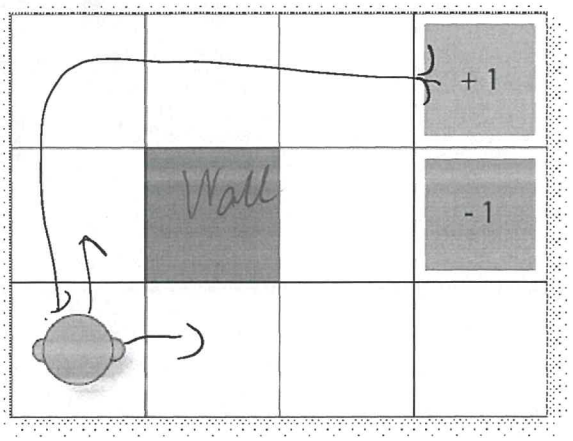
Lecture 8

Markov Decision Processes

Tuesday, 4 September 2018 10:29 AM



The grey square is a wall.

The two labelled cells give a reward: 1 and -1 respectively.

But! Things can go wrong:

- If the agent tries to move north, 80% of the time, this works as planned (provided the wall is not in the way)
- 10% of the time, trying to move north takes the agent west (provided the wall is not in the way);
- 10% of the time, trying to move north takes the agent east (provided the wall is not in the way)
- If the wall is in the way of the cell that would have been taken, the agent stays put.
- Similar for all other directions

80% ⇒ North
10% ⇒ west
10% ⇒ east

Classical Planning:

- Set of states S
- Initial state I
- Transition function A
- Goals G

MDPs:

- Set of states S
- Initial state I
- Transition probabilities:
  - P_a(s'|s)
- Reward function r(s, a, s') in real ~~numbers~~
- Discount factor (gamma)

Pr of action

$s \to s'$

$\gamma \in (0, 1)$

future

γ close to 0.9

Discounted reward:

$V = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 \ldots$

$= r_1 + \gamma (r_2 + \gamma r_3 + \gamma^2 r_4 \ldots)$

$V_t = r_1 + \gamma V_{t+1}$

$r_1 + \gamma (r_2 + \gamma V_{t+2})$

$V_t = r_t + \gamma V_{t+1}$

Probabilistic PDDL:

```
(define (domain bomb-and-toilet)
    (:requirements :conditional-effects :probabilistic-effects)
    (:predicates (bomb-in-package ?pkg) (toilet-clogged)
                 (bomb-defused))

    (:action dunk-package
        :parameters (?pkg)
        :effect (and (when (bomb-in-package ?pkg)
                           (bomb-defused))
                     (probabilistic 0.05 (toilet-clogged))))
```

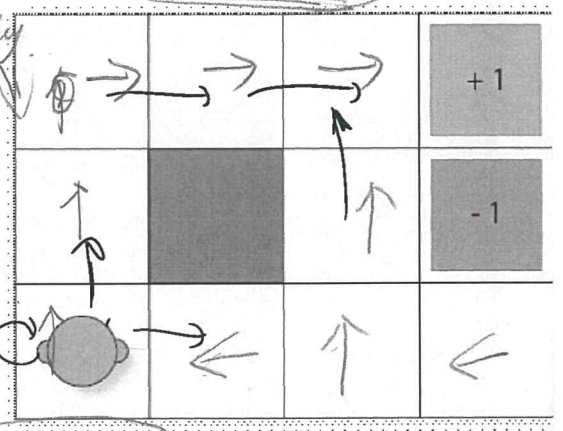
deterministic policy

Solution for MDP is a *policy*:

- at(0,0) => move_right
- at(0,1) => move_right
- at(0,2) => move_right
- at(0,3) => stay
- at(1,0) => move_up
- at(1,2) => move_up
- at(1,3) => move_up
- at(2,0) => move_up
- at(2,1) => move_left
- at(2,2) => move_up
- at(2,3) => move_left

function

an example, not best



$\to \pi(s) \to a$

Policy

$\pi(s)$

deterministic

we focus on

Stochastic $\pi(s, a)$

maximum to get

Expected return exercise:

You can steal:

A) An iPhone, which you think you have a 20% chance of selling for $500, or an 80% chance of selling for $250.

B) A Samsung, which you think you have a 50% chance of selling for $500, or a 50% chance of selling for $200.

Which do you steal? https://pollev.com/timothymille936

A: 0.2*500 + 0.8*250 = 300

B: 0.5*500 + 0.5*200 = 350

Expected return of a

reward $\to s'$

Bellman equations:

$$V(s) = \max_{a \in A} \left(\sum_{s' \in S} P_a(s'|s) \left[r(s, a, s') + \gamma V(s') \right] \right)$$

prob. · reward function · immediate reward · discounted future reward

$V(s) = \max_{a \in A(s)} \sum_{s' \in S} P_a(s'|s) [r(s, a, s') + \gamma V(s')]$

prob.

(immediate reward + discounted reward)

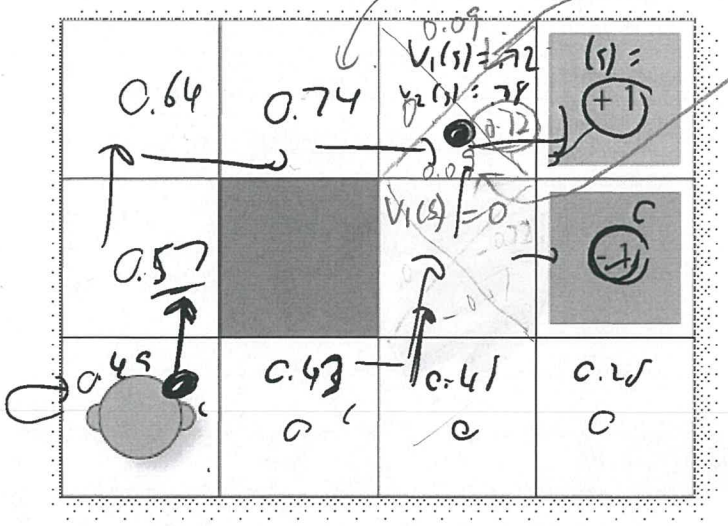
Value iteration and policy iteration

Tuesday, 4 September 2018 9:34 AM

$$V(s) = \max a \in A(s) \sum_{s' \in S} P_a(s'|s)[r(s,a,s') + \gamma V(s')]$$

Value iteration:

1) Set V_0 to arbitrary value for each s in S (choose 0 as the value)
2) While diff is >= epsilon
   a. For each s in S do
      i. V_t+1(s) := $\max a \in A(s) \sum_{s' \in S} P_a(s'|s)[r(s,a,s') + \gamma V\_t(s')]$
3) Select policy

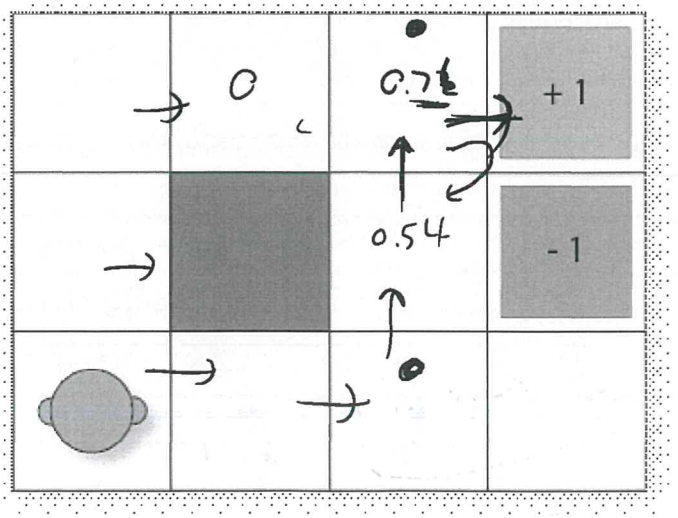


The two labelled cells give a reward: 1 and -1 respectively. (Actually, we will assume V(s)=1 or -1)

But! Things can go wrong:

- If the agent tries to move north, 80% of the time, this works as planned (provided the wall is not in the way)
- 10% of the time, trying to move north takes the agent west (provided the wall is not in the way);
- 10% of the time, trying to move north takes the agent east (provided the wall is not in the way)
- If the wall is in the way of the cell that would have been taken, the agent stays put.
- Similar for all other directions

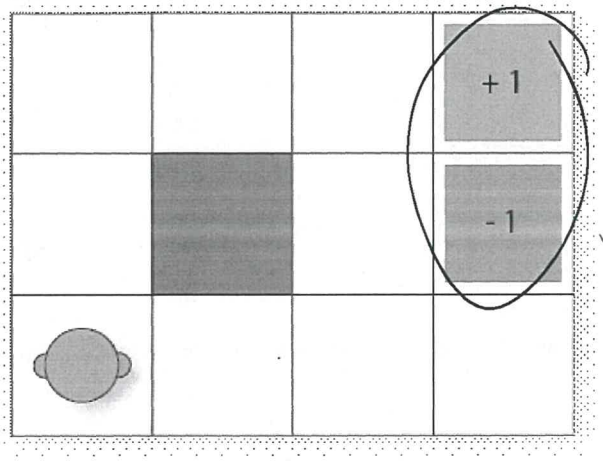
Policy iteration:



Monte-Carlo Tree Search

Rec Lecture 10

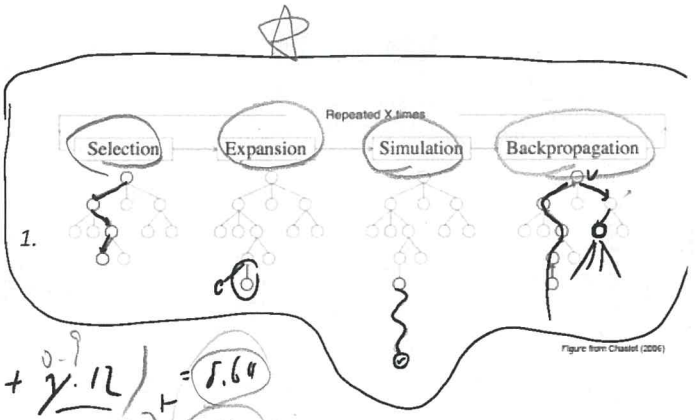
Tuesday, 11 September 2018 13:58 PM



$$V(s) = \max_{a \in A(s)} \sum_{s' \in \text{children}} P(s'|s)\,[r(s,a,s') + \gamma V(s')]$$

Monte-Carlo Tree Search (MCTS) overview

1. *V(s)* is the estimate of the real value of a state.
2. But we will also use it as an heuristic.
3. The search tree is incrementally built.
4. MCTS is an *anytime* algorithm: we terminate whenever and give the best answer so far.



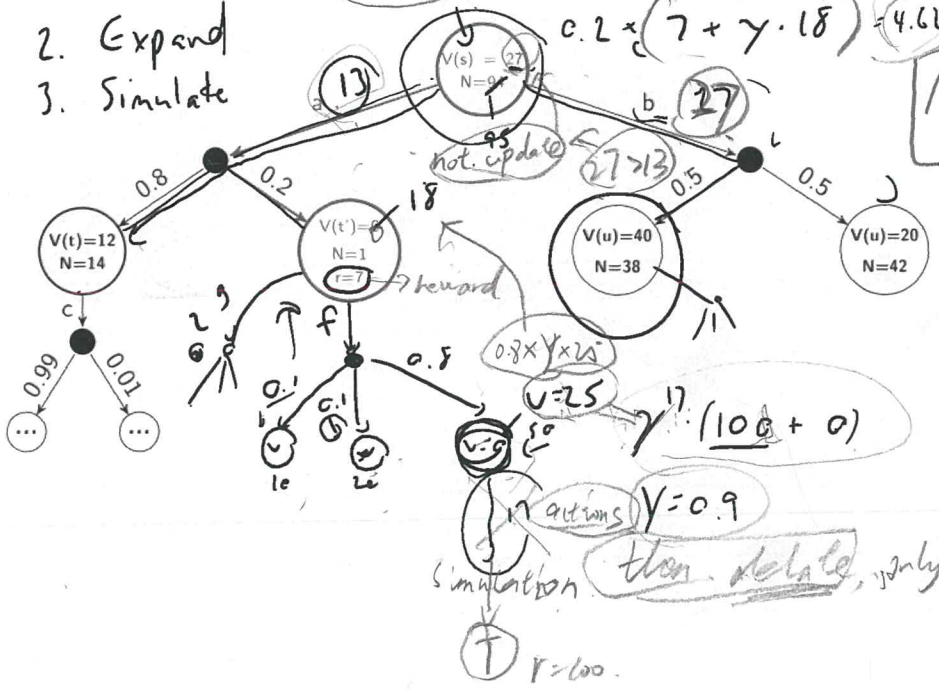
1. Select
2. Expand
3. Simulate
4. Backprop

information values

$0.8 \times (0 + \gamma \cdot 12) = 5.04$

$0.2 \times (7 + \gamma \cdot 18) = 4.62$

N: the number times we have visited so far in MCTS



$0.5 \times (0 + 0.9 \times 40) = 18$

$0.5 \times (0 + 0.9 \times 20) = 9$

$b = 27$

$\gamma^{17} \cdot (100 + 0)$

17 actions

simulation

then delete, only keep $V = 25$

$r = 100$

Exercise

Your phone-stealing habits continue, but you are doing well and opening up to a new market of people. Each day, you can sell a bag of 20 iPhones, Samsung, Huawei, or Pixel phones, but the price varies with each buyer, and you don't know the probability that people will buy them. You decide to alternate: iPhones one day, Samsung the next, then Huawei, then Pixel. After 100 days, you notice the following average return per day:

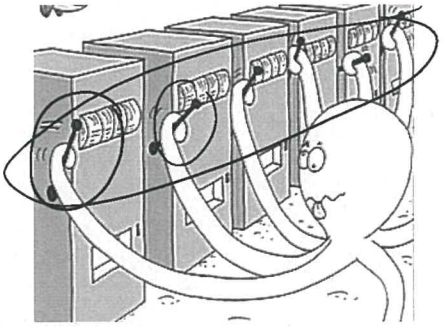
- Samsung: $400
- iPhone: $250
- Pixel: $200
- Huawei: $150

To help with your supply, you need to decide what you want to do for the next 100 days.
Assuming the next 100 days will look similar to the previous 100 days, what should you do?

https://pollev.com/timothymille936

Multi-armed bandits

Imagine that you have N number of slot machines (or poker machines in Australia), which are sometimes called one-armed bandits. Over time, each bandit pays a random reward from an unknown probability distribution. Some bandits pay higher rewards than others. The goal is to maximize the sum of the rewards of a sequence of lever pulls of the machine.



$$\pi(s) = \arg\max_a \left(Q(s,a) + \sqrt{\frac{2 \ln N}{N(s,a)}} \right)$$

Exploration vs exploitation

1. ϵ-greedy: ϵ in [0,1] (typically around 0.1), choose the best with probability $1-\epsilon$, and other choose randomly.
2. ϵ-decreasing: as above but ϵ decreases over time
3. Softmax: choose proportionally

Upper confidence bounds (UCB)

$$R(b) = \underbrace{\operatorname{argmax}_a Q(s,a)}_{\text{exploit}} + \underbrace{\sqrt{\frac{2 \ln(N(s))}{N(s,a)}}}_{\text{explore}}$$

$Q(s')$

number of a chosen in s

count

high reward + less visited

high for actions that have been explored less.

Upper confidence tree (UCT)

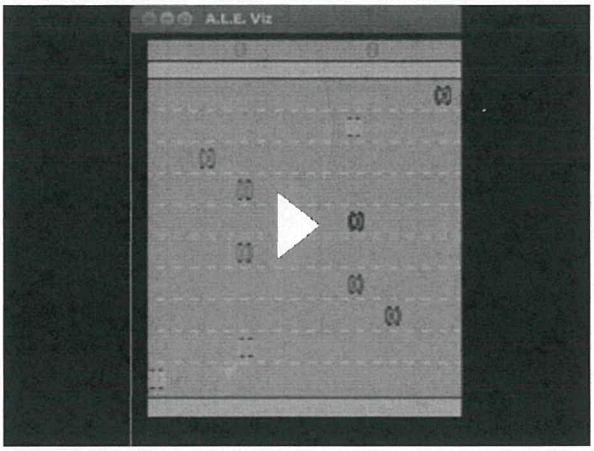
Q-value high for actions that have a high reward.

UCT = UCB + MCTS (almost!)

$$\pi(s) = \arg\max_{a \in A(s)} Q(s,a) + 2C_p \sqrt{\frac{2 \ln N(s)}{N(s,a)}}$$

UCT playing Mario Brothers: A MCTS-based Mario-playing controller

UCT playing Freeway: UCT Freeway - atari 2600



Value/policy iteration vs. MCTS

Reinforcement Learning

lecture 11

Reinforcement learning: what if we do not know transitions P and reward function r of an MDP?

The Mystery Game:
https://programmingheroes.blogspot.com/2016/02/udacity-reinforcement-learning-mystery-game.html

Q-learning

$Q(s,a)$

1. Initialise $Q(s,a)$ arbitrarily
2. For each episode:
   a. Initialise s (go to the initial state)
   b. Repeat for each step in the episode
      i. Select the next action a to apply from s (using e.g. epsilon greedy, UCT) use $Q(s,a)$
      ii. Execute action a and observe the reward r and new state s'
      iii. $Q(s,a) := Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
      iv. $s := s'$
   c. Until s is terminal

learning rate / low: past · high: future

update

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

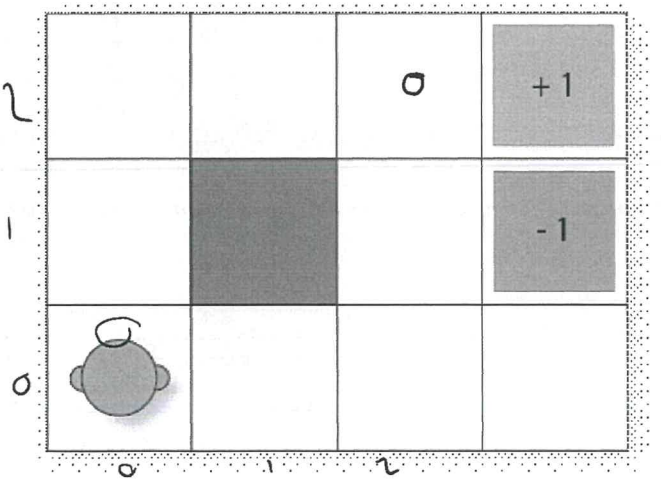
learning rate · reward estimate of optimal future value · old value · do not count extra $Q(s,a)$

Q-Tables

State	Action			
	North	South	East	West
(0,0)	0.53	0.36	0.36	0.21
(0,1)	0.61	0.27	0.23	0.23
...	0.792			
(2,2)	0.79	0.72	0.90	0.72
(2,3)	0.90	0.78	0.99	0.81

$Q(s_{(2,2)}, \text{South}) = 0.72$

$Q(s,a) = Q(s,a) + \alpha[r + \gamma \max Q(s',a') - Q(s,a)]$



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

Learning rate $\alpha = 0.1$
Discount reward factor $\gamma = 0.9$

Q-learning:
Q((2,2), North) = 0.79 + 0.1*(0 + 0.9*0.9 - 0.79) = 0.792

SARSA, with assumption that a' is West
Q((2,2), North) = $0.79 + 0.1 \times (0 + 0.7 \times 0.72 - 0.79) = 0.7828$

If a' is East is for SARSA, the update would be just the same as for Q-learning because East is the max action from (2,2)

Q-learning: Off-policy

1. Initialise $Q(s,a)$ arbitrarily
2. For each episode:
   a. Initialise s (go to the initial state)
   b. Repeat for each step in the episode
      i. Select the next action a to apply from s (using e.g. epsilon greedy, UCT) use $Q(s,a)$

follow the policy

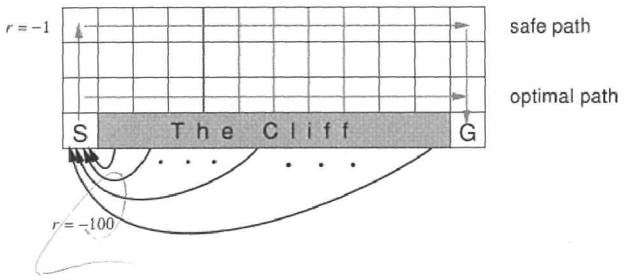
SARSA: On-policy learning

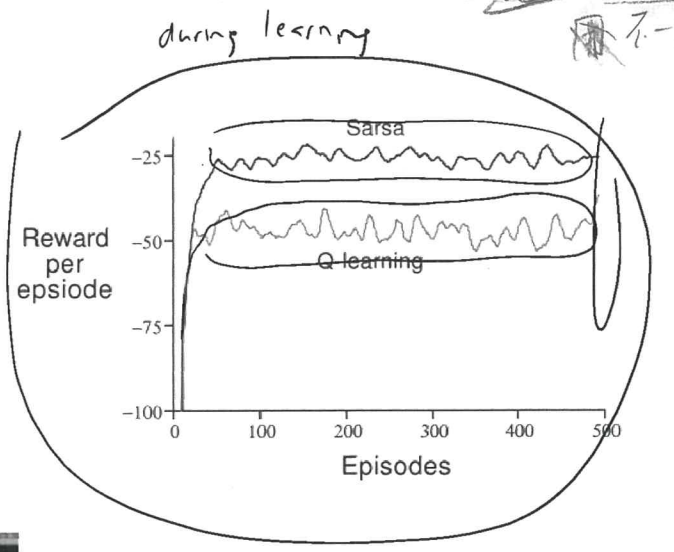
1. Initialise $Q(s,a)$ arbitrarily
2. For each episode:
   a. Initialise s (go to the initial state)
   b. Select the next action a to apply from s (using e.g. epsilon greedy, UCT)
   c. Repeat for each step in the episode

b. Repeat for each step in the episode
   i. Select the next action a to apply from s (using e.g. epsilon greedy, UCT) use $Q(s,a)$
   ii. Execute action a and observe the reward r and new state s'
   iii. $Q(s,a) := Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
   iv. $s := s'$
c. Until s is terminal

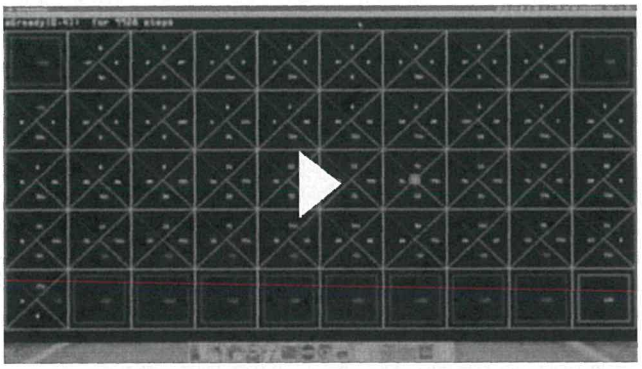
b. Select the next action a to apply from s (using e.g. epsilon greedy, UCT)
c. Repeat for each step in the episode
   i. Execute action a and observe the reward r and new state s'
   ii. Select the next action a' to apply from s' (using e.g. epsilon greedy, UCT)
   iii. $Q(s,a) := Q(s,a) + \alpha\,[r + \gamma Q(s',a') - Q(s,a)]$
   iv. $s := s'; a := a'$;
d. Until s is terminal

Off-policy vs. on policy learning

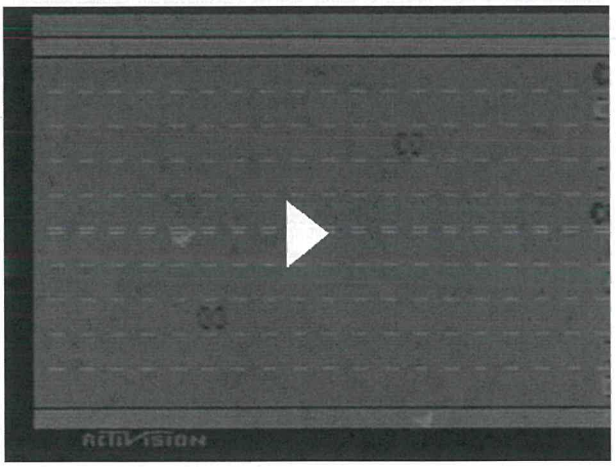




Gridworld Q-Learning - Example 3 - The Cliff

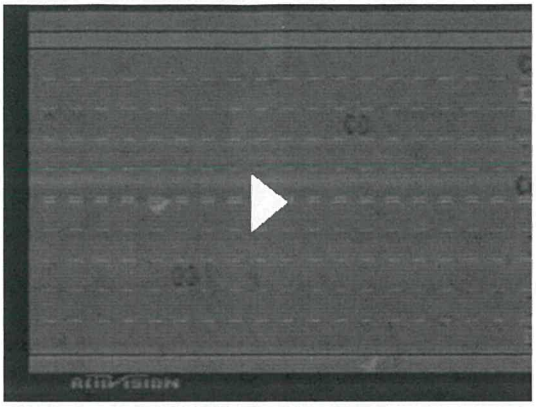


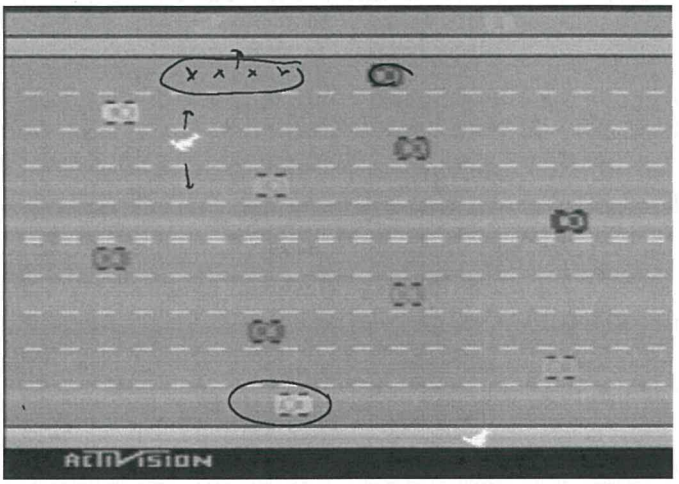
Learning to Play Freeway, using Reinforcement Learning



Q-Function Approximation, Lecture 12

Learning to Play Freeway, using Reinforcement Learning





$Q(s,a) \approx f$

Features and representation:
- Feature function returning a feature vector: $f(s,a) = \begin{pmatrix} f_1(s,a) \\ \vdots \\ f_n(s,a) \end{pmatrix}$ $= 1/12$, $= 1/3$, $: 0$, $= 1$, $= 0$, $= 0$, $= 0$

$f(s,a) = \begin{pmatrix} f_1(s,a) \\ f_2(s,a) \\ \vdots \\ f_n(s,a) \end{pmatrix}$

- Weight vector: $n \times |A|$ (number of features × number of actions)

Approximating Q(s,a):

$Q(s,a) = f_1(s,a) \cdot w_1^a + f_2(s,a) \cdot w_2^a \cdots + f_n(s,a) \cdot w_n^a$

$Q(s,a) = f_1(s,a) \cdot w_1^a + f_2(s,a) \cdot w_2^a + \cdots f_n(s,a) \cdot w_n^a = \sum_{i=0}^{n} f_i(s,a)\, w_i^a$

Q-learning Update:

execute a

→ for each feature i:

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

same; $Q(s',a')$ for SARSA

Example (Freeway):

(1) Assume $Q(s,a) = 0$ for all s,a — ① initialization

(2) Update: — ② update

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

$w_{up}^{up} \leftarrow 0 + [10 + 0.9 \times 0 - 0] \frac{1}{1}$ ($\frac{1}{0} = \frac{1}{1}$)

$w_i \leftarrow 5$

last row up

Deep Q-learning:

$\vartheta \leftarrow \vartheta_i + \alpha [r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \partial Q(s,a)/\partial \vartheta$ (gradient)

Q-learning:

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

SARSA:

$w_i^a \leftarrow w_i^a + \alpha [r + \gamma Q(s',a') - Q(s,a)] f_i(s,a)$

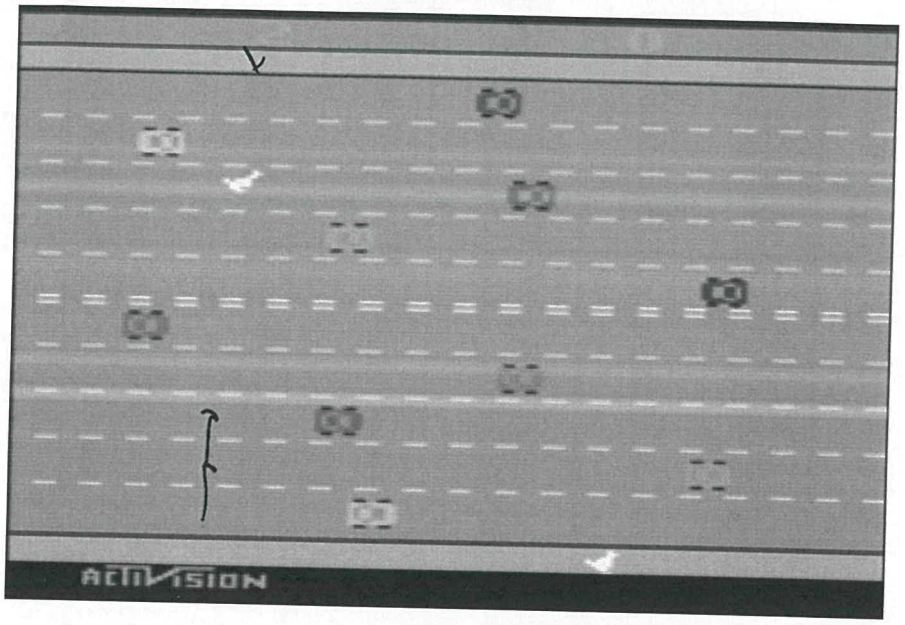
Deep Q-learning for robot grasping:

Learning Hand-Eye Coordination for Robotic Grasping

Reward Shaping

Lecture 12

Friday, 21 September 2018 15:22 PM



Key idea: add some small additional reward for particular behaviour:

$Q(s,a) := Q(s,a) + \alpha[r + F(s,s') + \gamma \max a' Q(s',a') - Q(s,a)]$

shaped reward

Potential-based Reward shaping:

$F(s,s') = \gamma \Phi(s') - \Phi(s)$

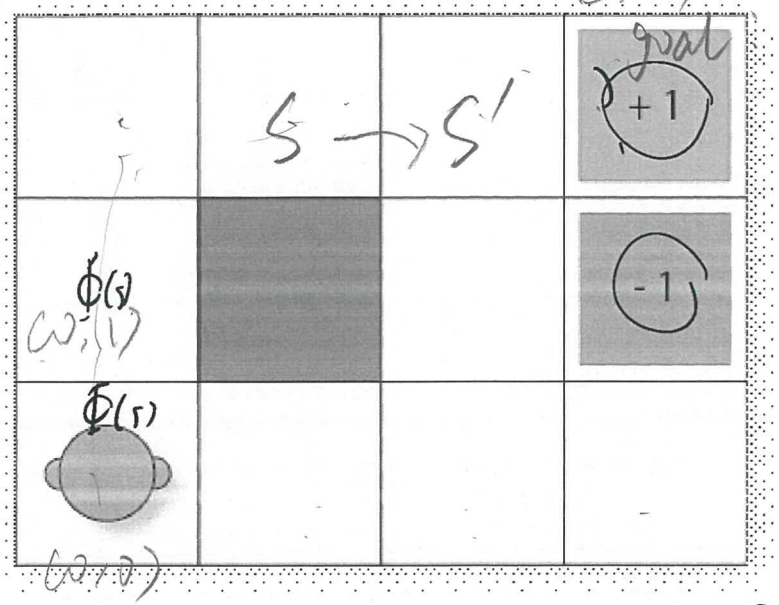
potential function.

GridWorld example:

$\Phi(s) = 1 / (|x(g) - x(s)| + |y(g) - y(s)|)$

$\Phi(s) = \frac{1}{|x(g) - x(s)| + |y(g) - y(s)|}$

Manhattan distance



Alternative to reward shaping: Q-function initialisation

$F(s,s') = F((1,2),(2,2))$

$= \gamma\Phi(2,2) - \Phi(1,2)$

$= 0.9 \times \frac{1}{|3-2|+|1-2|} - \frac{1}{|3-1|+|2-2|}$

F((1,2), (2,2)) = $\gamma \cdot \Phi(s') - \Phi(s)$

$= 0.9 \times \frac{1}{1} - \frac{1}{2}$ $0.9 \cdot \frac{1}{1} - \frac{1}{2}$

(0.4) $= 0.4$

F((0,0), (0,1)) =

$= 0.9 \times \frac{1}{|3-0|+|2-0|} - \frac{1}{|3-0|+|2-0|}$

$= 0.9 \times \frac{1}{5} - \frac{1}{4}$

Lecture 13

n-step temporal difference learning

Discounted future rewards — update Q(s,a) after n steps

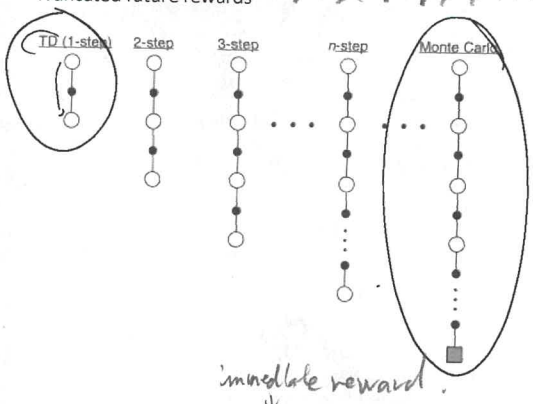
$G_t = r_t + V(s')$

$= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \dots + V(s_{t+n}) = r_t + \gamma G_{t+1}$

$= r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \dots$

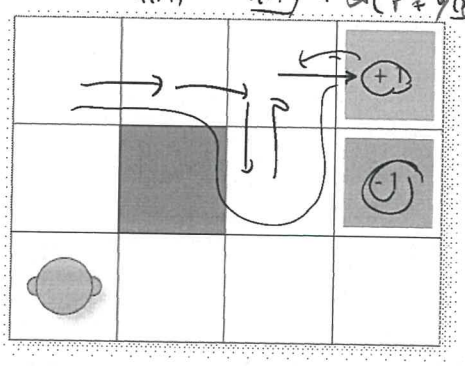
$= r_1 + \gamma(r_2 + \gamma r_3 + \gamma^2 r_4 + \dots)$

Truncated future rewards



With 1-step learning

State	Action			
	North	South	East	West
(0,0)	0	0	0	0
(0,1)	0	0	0	0
(0,2)	0	0	0	0
...				
(1,2)	0	0	0	0
(2,1)	0	0	0	0
(2,2)	0	0	0.45	0
(2,3)	0	0	0	0
...				



$G_t = r_t + V(s_{t+1})$ one step return.

$G_t^2 = r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2})$ two step return.

$G_t^n = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \dots + \gamma^n V(s_{t+n})$ n step

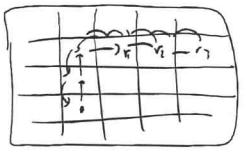
SARSA: $Q(s,a) := Q(s,a) + \alpha [r + \gamma Q(s',a') - Q(s,a)]$

Change the update:

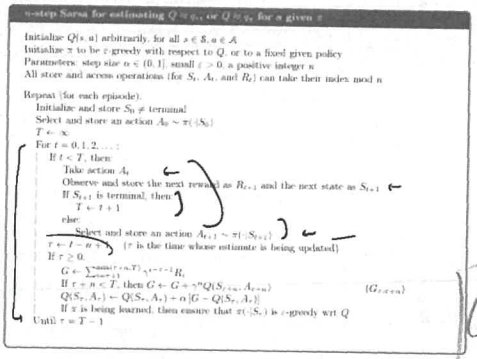
terminate

$G \leftarrow \sum_{i=t+1}^{\min(t,t+n)} \gamma^{i-t-1} r_i$ — $n=1$: next step.

$Q(s,a) \leftarrow Q(s,a) + \alpha [G + \gamma^n Q(s_{t+n}, a_{t+n}) - Q(s,a)]$



$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha [G + \gamma^n Q(s_{t+n}, a_{t+n}) - Q(s_t,a_t)]$



Initialize $Q(s,a)$ arbitrarily, for all $s \in S, a \in A$

Initialize π to be ε-greedy with respect to Q, or to a fixed given policy

Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n

All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):

Initialize and store $S_0 \neq$ terminal

Select and store an action $A_0 \sim \pi(\cdot|S_0)$

$T \leftarrow \infty$

For $t = 0, 1, 2, \dots$:

If $t < T$, then:

Take action A_t

Observe and store the next reward as R_{t+1} and the next state as S_{t+1}

If S_{t+1} is terminal, then:

$T \leftarrow t+1$

else:

Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$

$\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)

If $\tau \geq 0$:

$G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$

If $\tau + n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$

$Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha [G - Q(S_\tau, A_\tau)]$

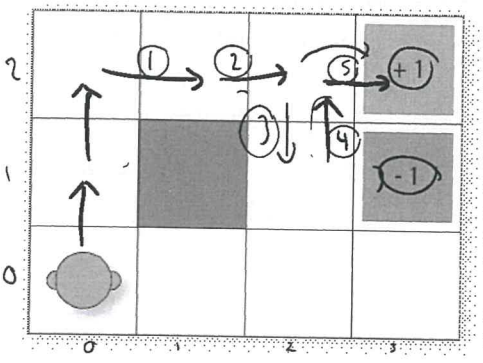
If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε-greedy wrt Q

Until $\tau = T - 1$

$G_t = \sum_{i=t+1}^{t+n} \gamma^{i-t-1} r_i$

$Q(s,a) = Q(s,a) + \alpha [G_t + \gamma^n Q(s',a') - Q(s,a)]$

Exercise: Grid World



Compute 5-step SARSA update

$\alpha = 0.5$

$\gamma = 0.9$

$G_5 = \gamma \cdot 1 \quad \gamma \cdot 1 = 0.9$

$G_4 = 0 + \gamma^2 \cdot 1 \quad 0 + \gamma^2 \cdot 1 = 0.81$

$G_3 = 0 + 0 + \gamma^3 \cdot 1 \quad 0 + 0 + \gamma^3 \cdot 1 =$

$G_2 = \quad 0 + 0 + 0 + \gamma^4 \cdot 1$

$G_1 = \quad 0 + 0 + 0 + 0 + \gamma^5 \cdot 1$

$Q(s_{(2,2)},E) = 0 + 0.5[G_5 + 0 - 0] = 0.45$

$Q(s_{(1,2)},N) = 0 + 0.5[0.81 + 0 - 0] = 0.405$

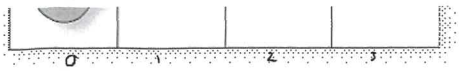
...

$Q(s_{(2,2)},S) =$

$0 + 0.5[G_5 + 0 - 0] = 0.45$

$0 + 0.5[G_4 + 0 - 0] = 0.405$

0.81



$Q(s_{(2,2)},E) = 0 + 0.5[5 + 0 - 0] = 0.45$

$Q(s_{(1,2)},N) = 0 + 0.5[0.81 + 0 - 0] = 0.405$

...

$Q(s_{(2,2)},S) =$ "

$Q(s_{(2,1)},E) =$ "

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

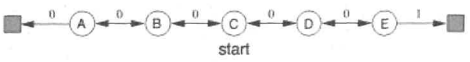
With 1-step learning

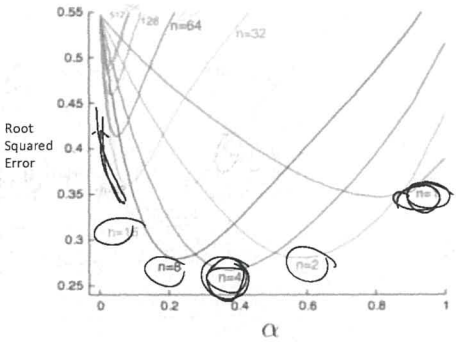
State	Action			
	North	South	East	West
(0,0)	0	0	0	0
(0,1)	0	0	0	0
(0,2)	0	0	0	0
...				
(1,2)	0	0	0	0
(2,1)	0	0	0	0
(2,2)	0	0	0.45	0
(2,3)	0	0	0	0
...				

With 5-step learning

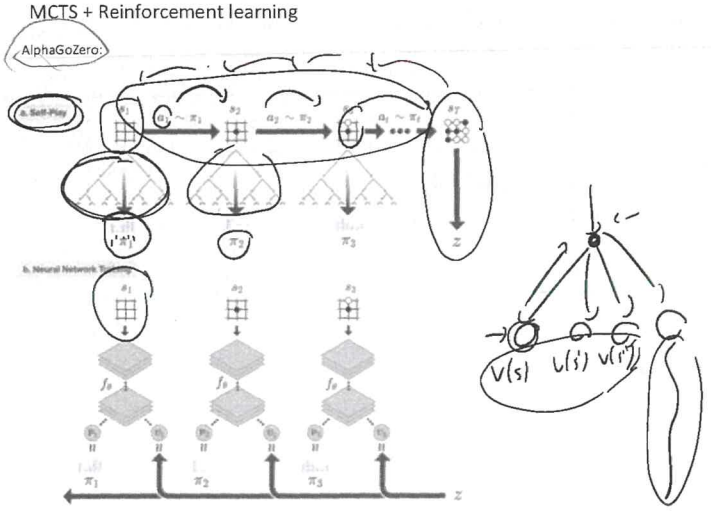
State	Action			
	North	South	East	West
(0,0)	0	0	0	0
(0,1)	0	0	0	0
(0,2)	0	0	0.2953	0
...				
(1,2)	0	0	0.3281	0
(2,1)	0.405	0	0	0
(2,2)	0	0.3645	0.45	0
(2,3)	0	0	0	0
...				

Example: Random walk





MCTS + Reinforcement learning



Normal form games

Prisoner's dilemma

Consider the following (hopefully fictional) scenario. You and your partner in crime have been arrested for armed robbery. To avoid a lengthy trial, the police put you and your partner in crime in separate cells, *unable to communicate with each other*. The police offer each of you the following deal:

- If neither of you admit your guilt, you will be charged with carrying illegal weapons, and receive 1 year each in prison.
- If one of you admits both are guilty, while the other does not, the prisoner that admitted will get off free in exchange for the confession, while the other prisoner will receive 4 years in prison.
- If both of you admit you are guilty, you will each receive 2 years in prison — the length is reduced from 4 years in exchange for your confession.

Assuming that you care little for your partner (you are an armed robber, after all), what choice will you make? Will you cooperate with your partner in crime and deny your guilt, or will you betray your partner and admit your guilt? Remember, you have no idea which choice your partner will make, and you cannot communicate.

A: Admit
B: Deny

https://pollev.com/timothymille936

if B admit => A admit
if B deny => A admit

Normal Form game

Nash equilibrium
is there any action make better

Players ≥ 2 players
→ moves (actions)
→ utility / payoffs $u_i(a_j)$

Nash equilibrium		B admit	B deny
A	admit	-2, -2	0, -4
	deny	-4, 0	-1, -1

① dominant strategy] games
② Nash equilibrium → best response] games
(admit, admit)

The Advertising Game

Firm 2 — best response

Firm 1	Not	Adv
Not	16, 12	7, 13 ← equilibrium
~~Adv~~	~~12, 7~~	6, 6 ← dominated strategy

F1 : dominant strategy (NOT)
F2 : no dominant strategy. equilibrium

If Firm 2:

A: Advertise
B: Not Advertise

https://pollev.com/timothymille936

Split or Steal?

Player 1 \ Player 2	Split	Steal
Split	0.5, 0.5	0, 1
Steal	1, 0	0, 0

Three equilibria

A: Split, Split
B: Split, Steal
C: Steal, Split
D: Steal, Steal

https://pollev.com/timothymille936

golden balls. the weirdest split or steal ever!

Mixed strategies

pure strategy 之前的

Matching pennies

Mixed strategies

Player Even \ Player Odd	Heads (Y)	Tails (1-Y)
Heads (X)	1, -1	-1, 1
Tails (1-X)	-1, 1	1, -1

prob when even play Head: X

A: Heads
B: Tails

https://pollev.com/timothymille936

$E(H) = E(T)$

$1-2X = 2X-1$

$2 = 4X$

$X = \frac{1}{2}$

$1-X = \frac{1}{2}$

$E(T) = E(H)$

Indifference:

Odd: 換成 odd play Heads.

$E(H) = -1X + 1-X$

$= 1-2X$

$E(T) = X + -1(1-X)$

$= 2X-1$

Even:

$E(H) = 2Y-1$

$E(T) = 1-2Y$

$2Y-1 = 1-2Y$

$Y = \frac{1}{2}$

Security games

Defender \ Adversary	Terminal 1 (Y)	Terminal 2 (1-Y)
Terminal 1 (X)	5, -3	-1, 1
Terminal 2 (1-X)	-5, 5	2, -1

3/13, 10/13

3/5, 2/5

6/10, 4/10

Adversary:

$E(T1) = -3X + 5(1-X)$

$= 5-8X$

$E(T2) = X + -1(1-X)$

$= 2X-1$

Adversary:

$E(T_1) = -3X + 5(1-X)$

$= -8X+5$

$E(T_2) = X - (1-X) = 2X-1$

$E(T_1) = E(T_2)$

$-8X+5 = 2X-1$

$6 = 10X$

$X = \frac{3}{5}$

Defender:

$E(T_1) = 5Y - (1-Y) = 6Y-1$

$E(T_2) = -5Y + 2(1-Y) = -7Y+2$

Defender:

$E(T1) = 6Y-1$

$E(T2) = -7Y+2$

$6Y-1 = -7Y+2$ $E(T1) = E(T2)$

$13Y = 3$

$Y = \frac{3}{13}$

$E(T_1) = X + -1(1-X)$

$= 2X - 1$

$E(T_1) = E(T_2)$

$5 - 5X = 2X - 1$

$6 = 10X$

$X = 6/10$

$1 - X = 4/10$

Defender strategy

$E(T_1) = \ldots$

$E(T_1) = E(T_2)$

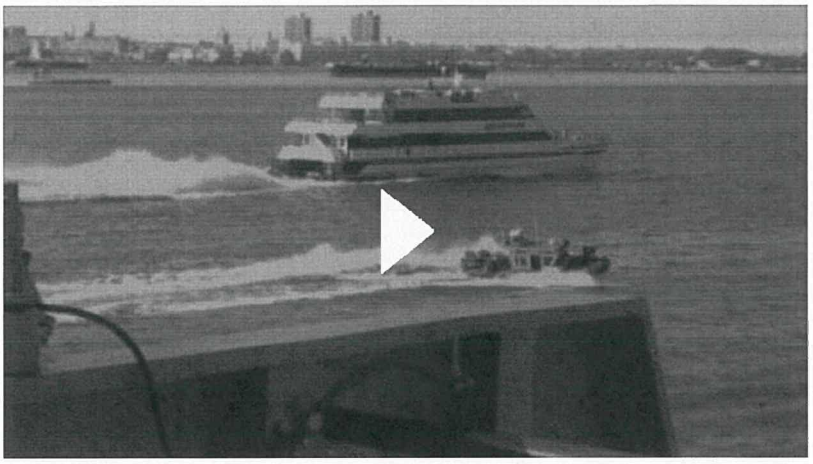
$6Y - 1 = 2 - 7Y$

$Y = 3/13$

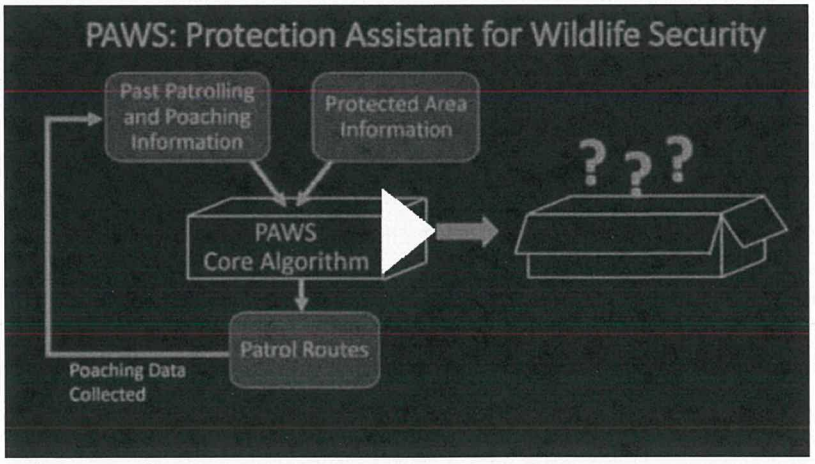
$1 - Y = 10/13$

Adversary strategy

Ferry escort



Save the Wildlife, Save the Planet: Protection Assistant for Wildlife Security (PAWS)



Player 2

$E(\text{left}) = X + 0$

$E(\text{Right}) = 0 \cdot X + 3(1-X)$

$= -3X + 3$

$E(\text{left}) = E(\text{Right})$

$X = -3X + 3$

$4X = 3$

$X = 3/4$

$1 - X = 1/4$

Player 1

$E(\text{Up}) = 3Y$

$E(\text{Down}) = 1 - Y$

$3Y = 1 - Y$

$4Y = 1$

$Y = 1/4$

Pure and mixed equilibria

		Player 2: Left (Y)	Player 2: Right ($1-Y$)
Player 1	Up (X)	3, 1	0, 0
Player 1	Down ($1-X$)	0, 0	1, 3

$E(L) = 1X + 0$

$E(R) = 0 + 3(1-X)$

$E(L) = E(R)$

$4X = 3 - 3X$

$= X$ Player 1	Up	3, 1	0, 0
$1-X$	Down	0, 0	1, 3

$4X = 3 - 3X$

$X = 3/4$

Two pure equilibria

One mixed equilibria

1 $(X = 3/4, 1-X = 1/4)$

2 $(Y = 1/4, 1-Y = 3/4)$

Extended form games

- Buyer/seller game

Seller:
1) Sell low quality = 3
2) Sell high quality = 2
3) Don't sell low quality = 1
4) Don't sell high quality = 0

Buyer:
1) Buy high quality, payoff = 2
2) Buy nothing, payoff = 1
3) Buy low quality, payoff = 0

Buyer

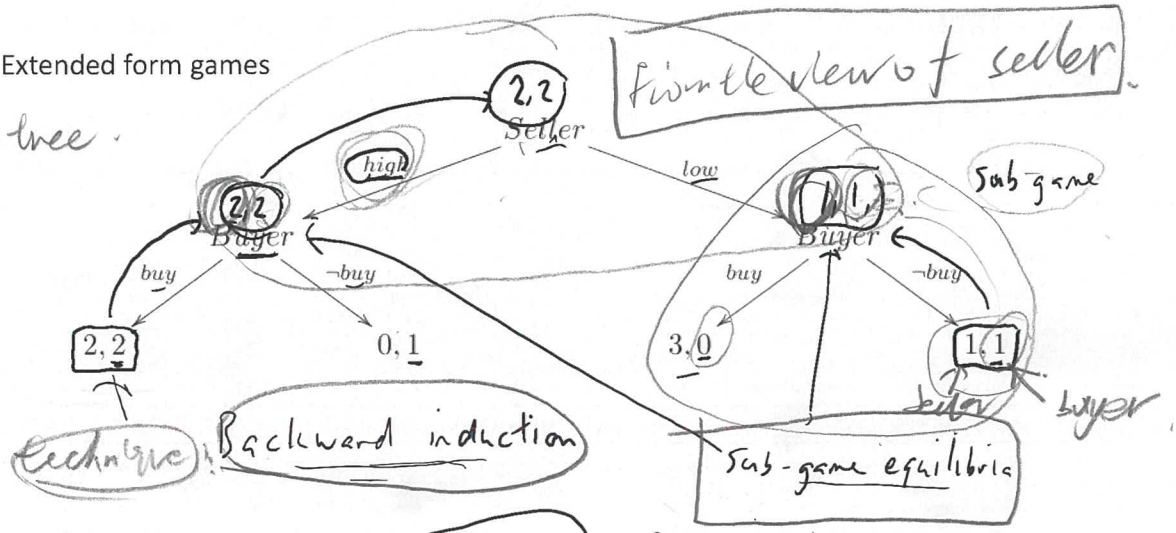
Seller	buy	don't buy
high	2, 2	0, 1
low	3, 0	1, 1

dominated strategy

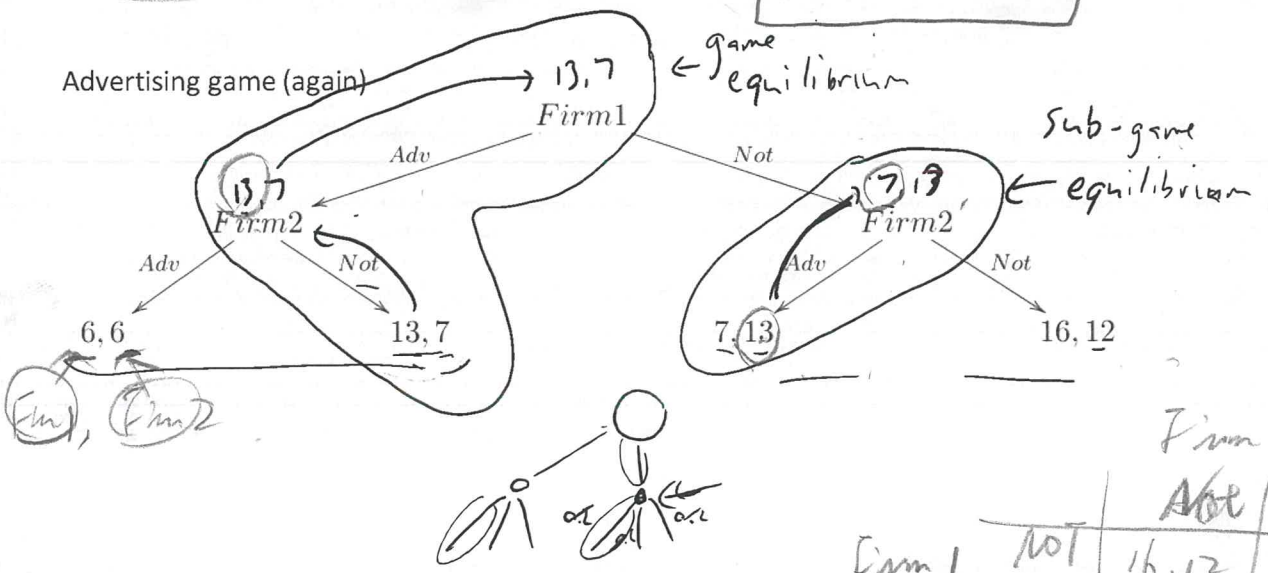
one pure equilibrium

Extended form games

game tree



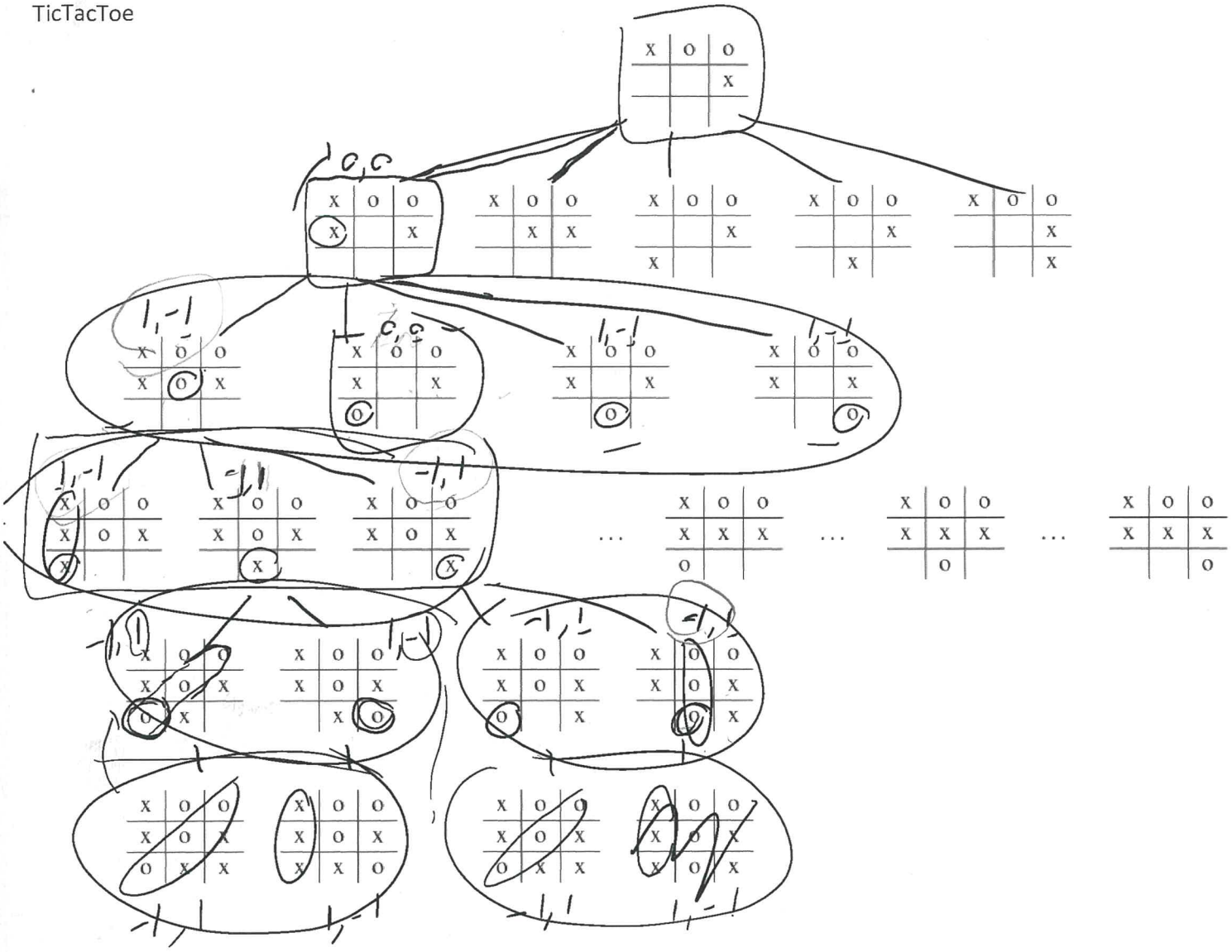
Advertising game (again)



different outputs

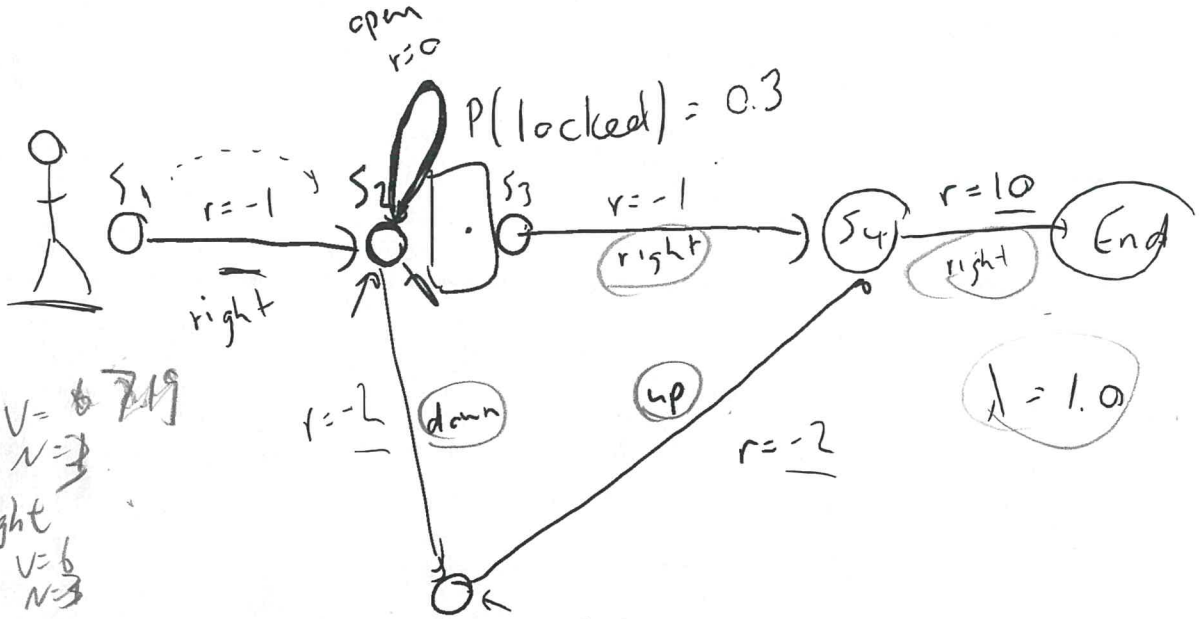
Firm 1 \ Firm 2	Not	Adv
Not	16, 12	7, 13
Adv	13, 7	6, 6

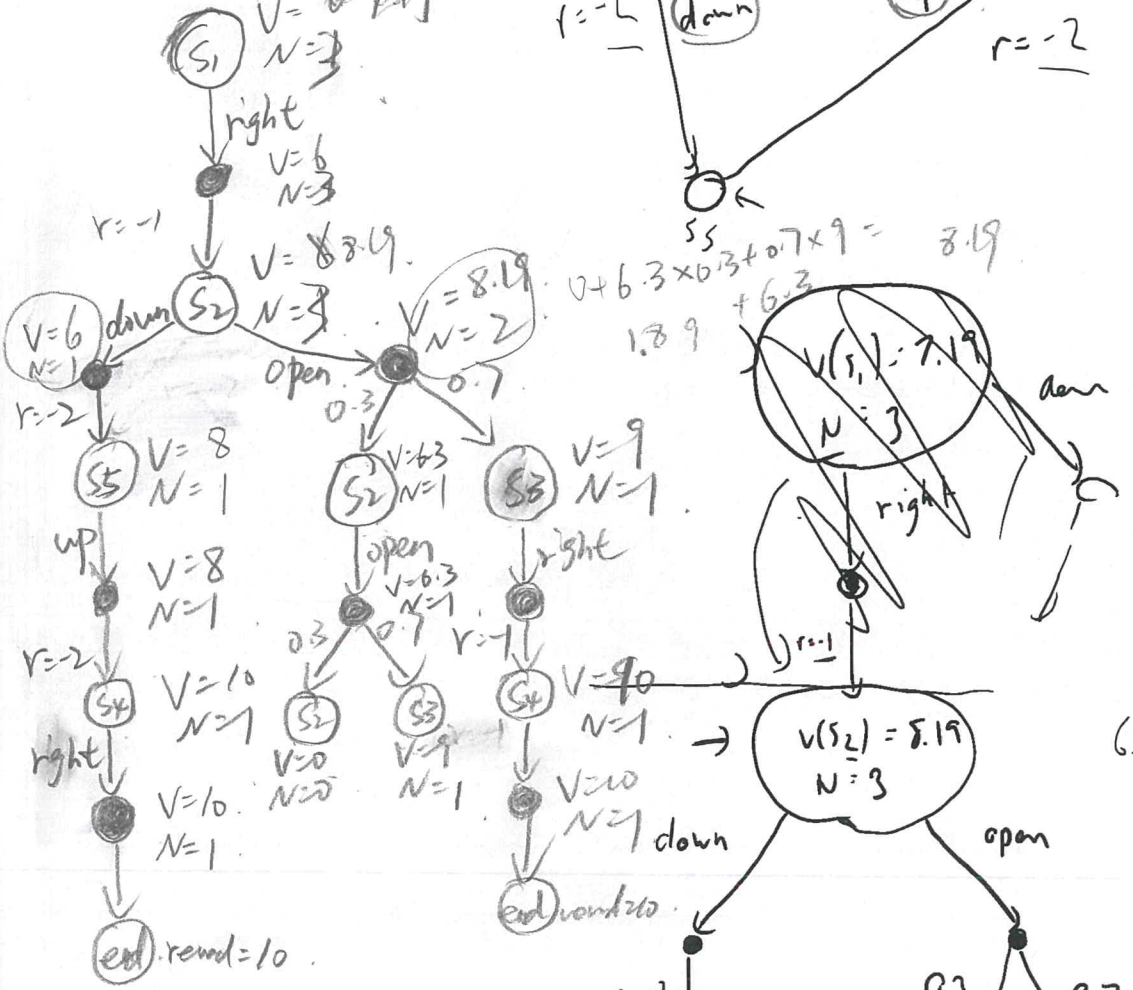
TicTacToe



Revision (MCTS, Q-learning & SARSA)

MCTS $V(s) = \max_{a \in A_s} \sum P_a(s'|s)\,[r(s,a,s') + \gamma V(s')]$



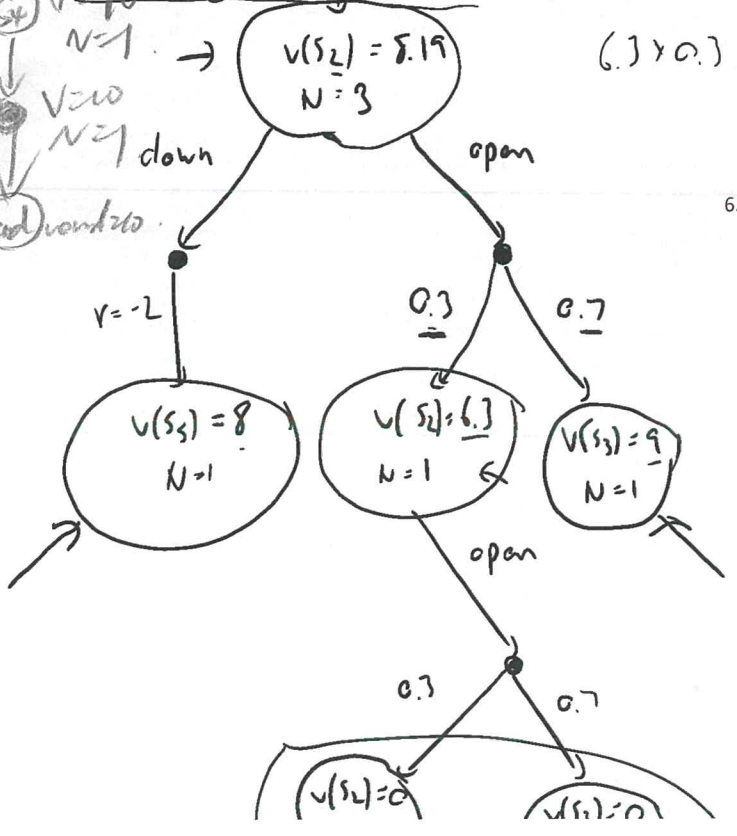


0.3x0 + 0.7x9 = 6.3

0 + 6.3x0.3 + 0.7x9 = 8.19

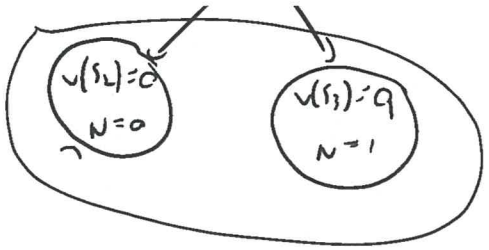
1.89 + 6.3

① Select
② Expand
③ Simulate
④ Backpropagate



(6.3 x 0.3) + (9 x 0.7) / 6.3

6.3*0.3 + 9*0.7 = 8.19


v(s4)=0
N=0
v(s3)=9
N=1