Robot Learning: Reinforcement Learning

Lecture 10

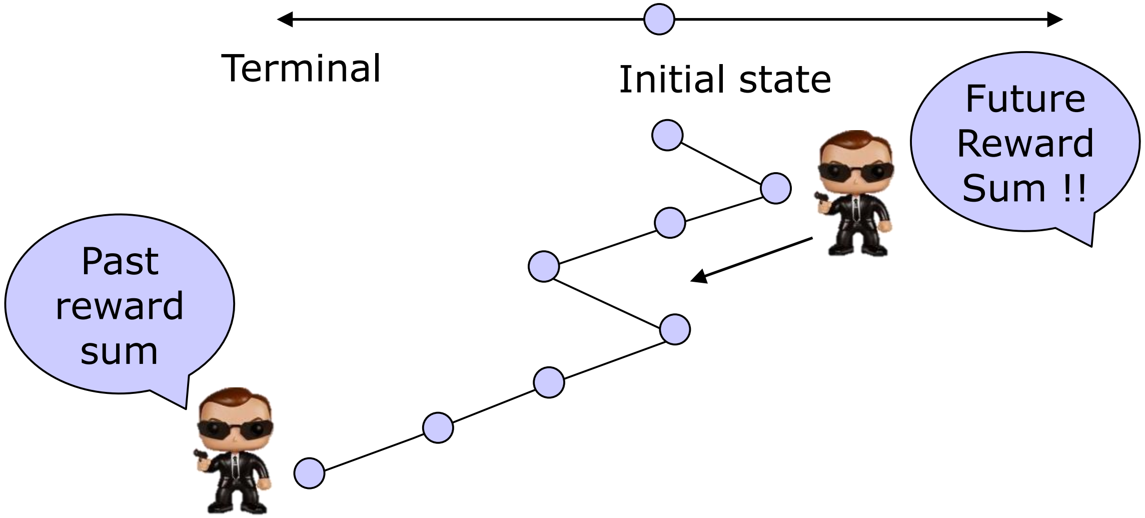
양정연
2020/12/10

1 Reward and Return in RL

Past or Future Rewards

- 1. Viewpoint at the Terminal
  - Return is the sum of all PAST rewards

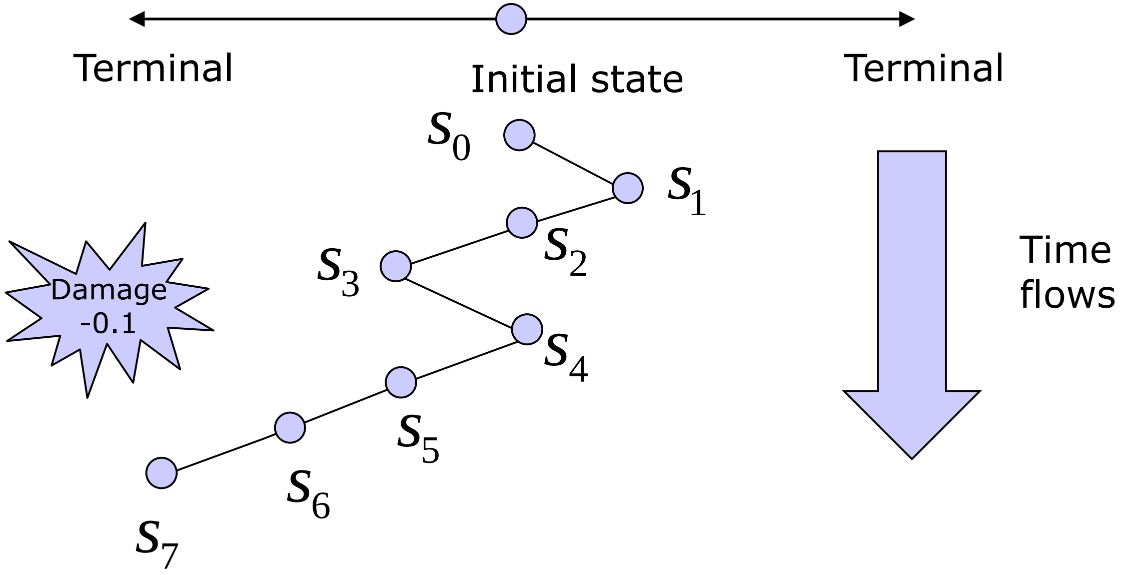
- 2. Agent’s viewpoint (RL uses this)
  - Return is the sum of all Future rewards.



Reward and Return

- Reward : get a reward in each state transition
  - Whenever an agent moves, it gets a reward from environment
  - Ex) +1,+2 at terminals and -0.1 at each turn

- State : state varies by time flows ($s_0 \rightarrow s_1 \rightarrow s_2 \ldots \rightarrow s_t$)

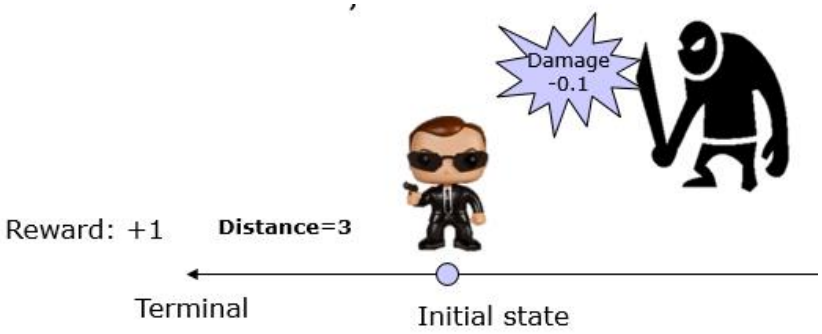


Reward and Return

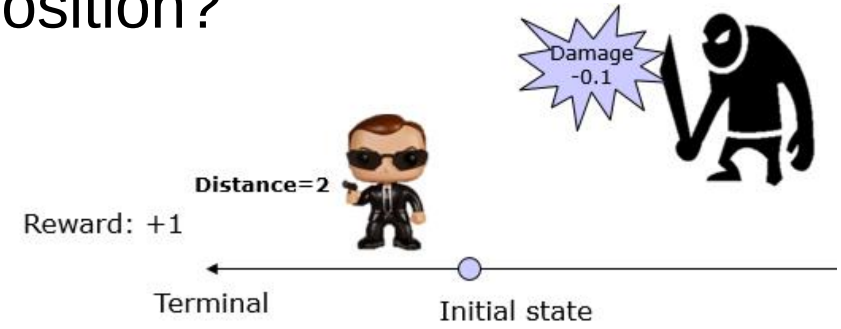
- Return : summation of all rewards.

$$R = \sum_{k=1}^{\infty} r_k$$

  - Ex) Rewards are -0.1,-0.1, 1.
  - Return is -0.1-0.1+1 = 0.8

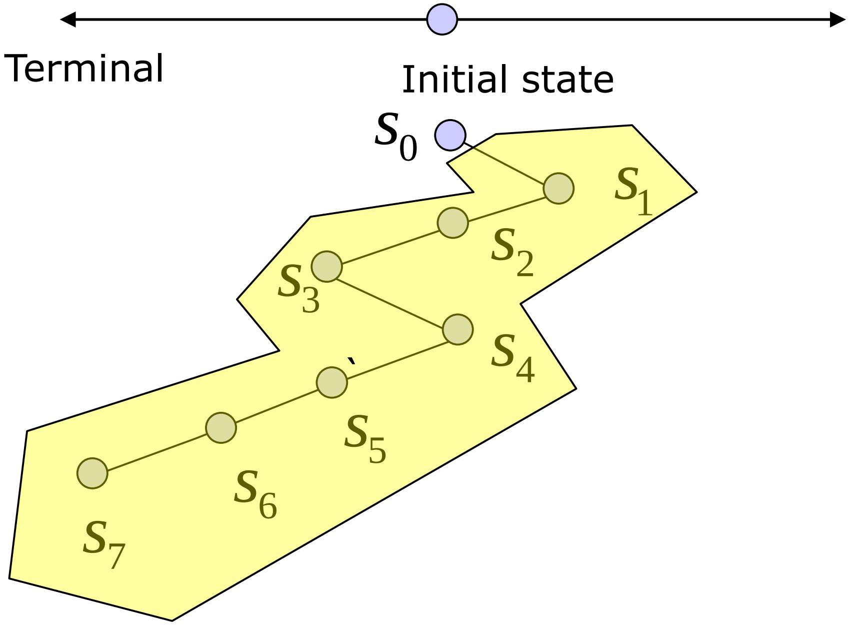


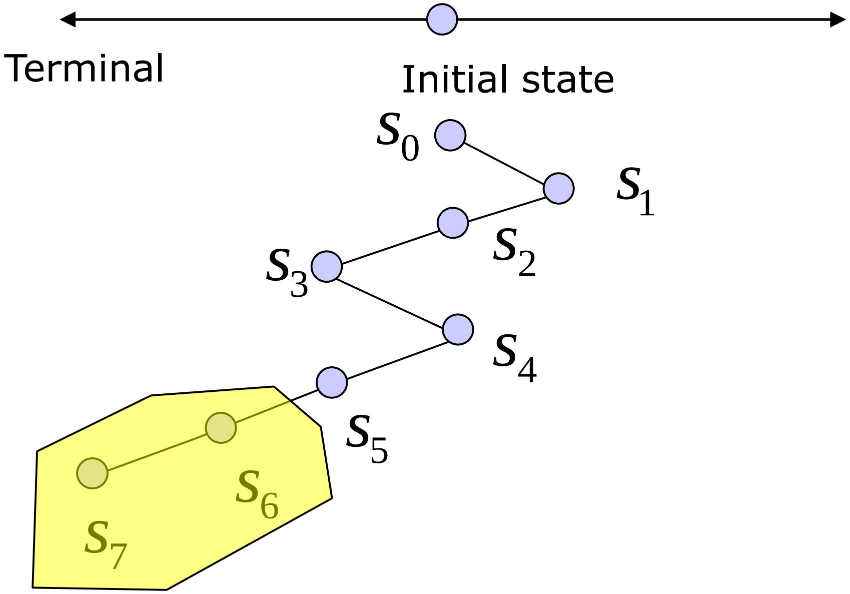
- Question: Return at another position?
  - Ex) Rewards are -0.1, and 1
  - Returns is -0.1+1 = 0.9



Return at Different Position

- Return is a function w.r.t. State Position





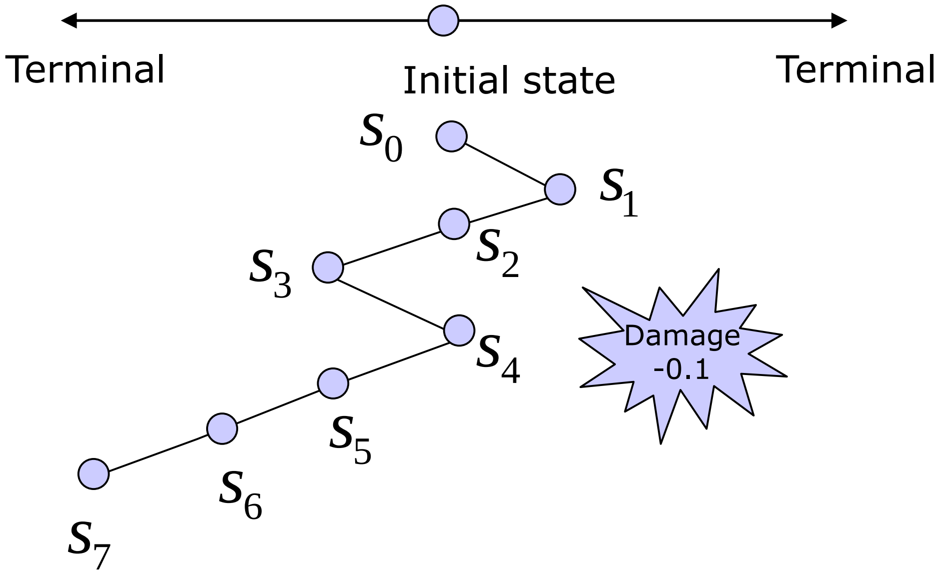
$$R(\mathrm{s}=\mathrm{s}_0)=\sum_{k=1}^{\infty} r_k=(r_1+r_2+r_3+r_4+r_5+r_6+r_7)$$

$$=-0.1\cdot 6+1=0.4$$

$$R(s=s_5)=\sum_{k=1}^{\infty} r_k=(r_6+r_7)$$

$$=-0.1+1=0.9$$

Example of a Single Return



A Single Return with one case

$$R_t = \sum_{k=1}^{\infty} r_{t+k}$$

$$R_{t=0}(s=s_0) = \sum_{k=1}^{\infty} r_{t+k} = \sum_{k=1}^{7} r_{0+k} = (r_1 + r_2 + r_3 + r_4 + r_5 + r_6 + r_7)$$

$$= -0.1 - 0.1 - 0.1 - 0.1 - 0.1 + (-0.1 + 1) = 0.4$$

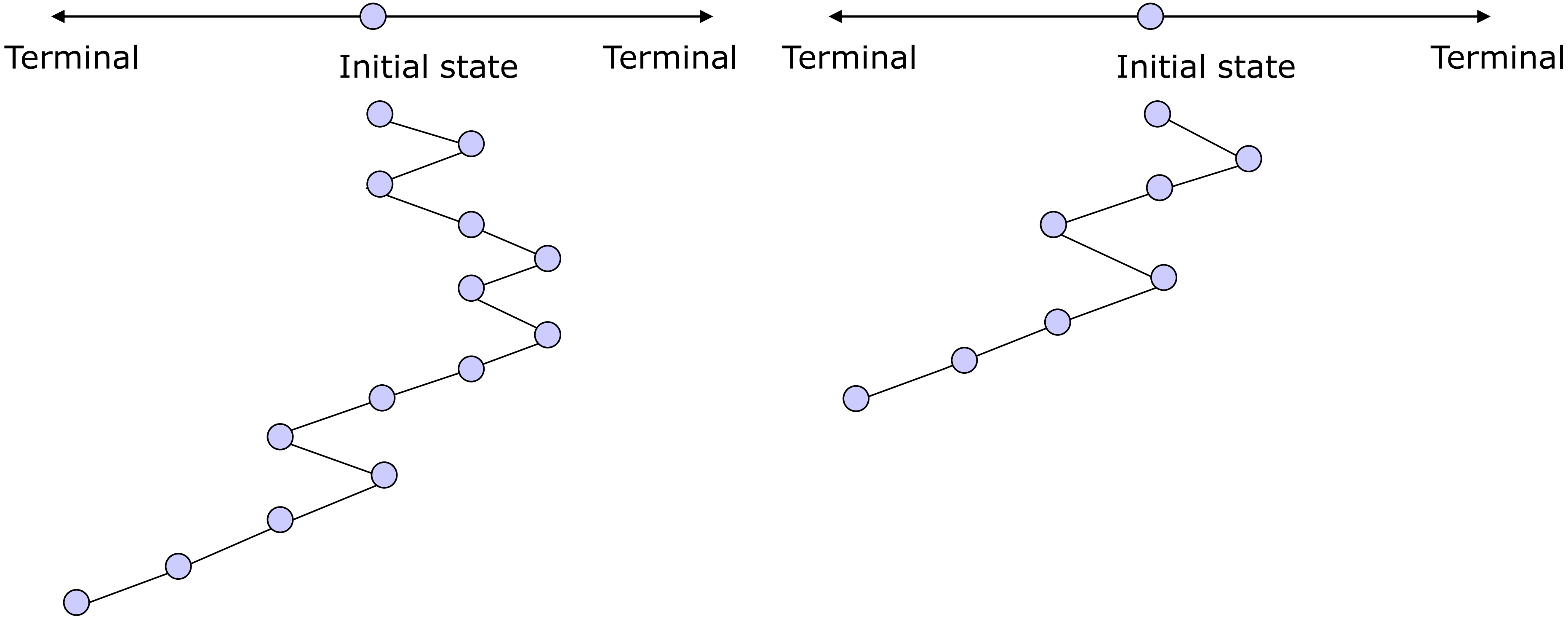
$$\Rightarrow R_{t=4}(s=s_4) = \sum_{k=1}^{\infty} r_{t+k} = \sum_{k=1}^{3} r_{4+k} = (r_5 + r_6 + r_7)$$

$$= -0.1 + (-0.1 + 1) = 0.6$$

Watch this, S0 =S4!

However,
because S4 is closer to S7,
Rt=0 is smaller than Rt=4
(0.4< 0.6)

However, There are Many Return Values



Terminal

Initial state

Terminal

Terminal

Initial state

Terminal

- Many possible returns are averaged for Learning

$$E\{R_t\} = E\left(\sum_{k=1}^{\infty} r_{t+k}\right)$$

Summary of Reinforcement Learning

- Future Reward
  - If an agent moves in future, how much reward does an agent obtains? (Not the past reward)

- Return = sum of all possible future rewards

$$R_t = \sum_{k=1}^{\infty} r_{t+k}$$

- Bigger Expectation of Return(sum of all future rewards) is Better for us → Reinforcement Learning!

$$E\{R_t\} = E\left(\sum_{k=1}^{\infty} r_{t+k} \right)$$

Expectation is Hard works.

- State value is based on Expectation
- In other words, we collect many path data.
  - How we estimate expectation? We need Brilliant Idea!!
- Expectation is estimated by Iterative Method

$$E(x)_N = \frac{1}{N}\sum_i^N x_i \rightarrow E(x)_{N+1} = \frac{1}{N+1}\sum_i^{N+1} x_i$$

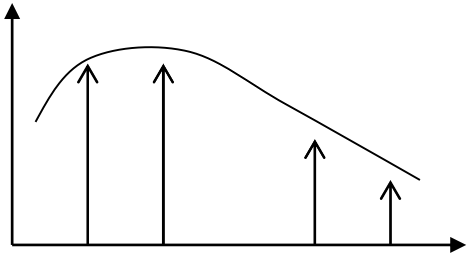
$$E(x)_{N+1} = \frac{1}{N+1}\left(x_{N+1} + \sum_i^N x_i\right) = \frac{1}{N+1}\left(x_{N+1} + NE(x)_N + E(x)_N - E(x)_N\right)$$

$$= E(x)_N + \frac{1}{N+1}\left(x_{N+1} - E(x)_N\right)$$

$$\cong E(x)_N + \alpha\left(x_{N+1} - E(x)_N\right) = \alpha x_{N+1} + (1-\alpha)E(x)_N \Rightarrow \text{Infinite Impulse Response}$$

Estimated Expectation with IIR Filter

- In Digital signal processing (DSP)
- Finite Impulse Response (FIR) Vs. Infinite Impulse Response(IIR)
- Basic concept
  - A set of Impulses represents system behaviors.



$$\frac{Y(s)}{X(s)} = G(s), \quad \text{Laplace Transform of Impuse, } \delta(\text{t}) \text{ is 1}$$

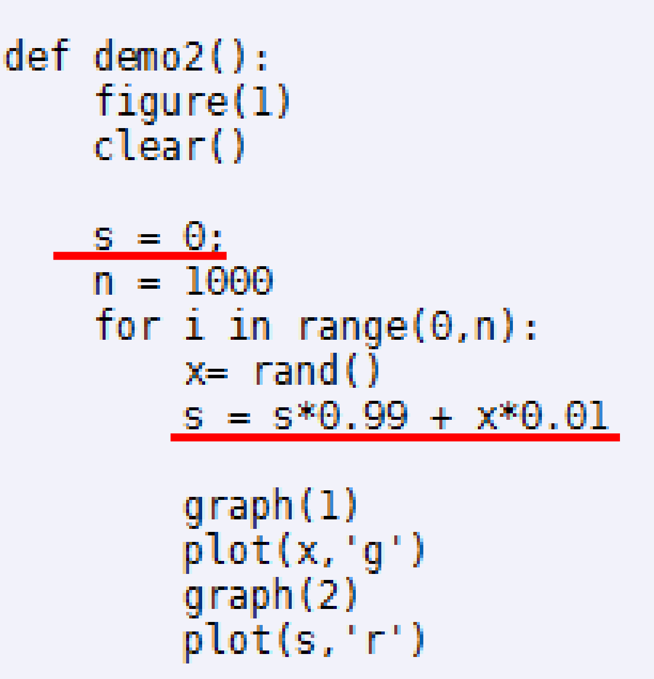
$$\therefore \text{Y(s)=G(s)}$$

  - FIR is a set of impulses, but IIR is the recursive set of impulses.

$$IIR: f_{k+1} = \alpha x_k + (1-\alpha) f_k$$

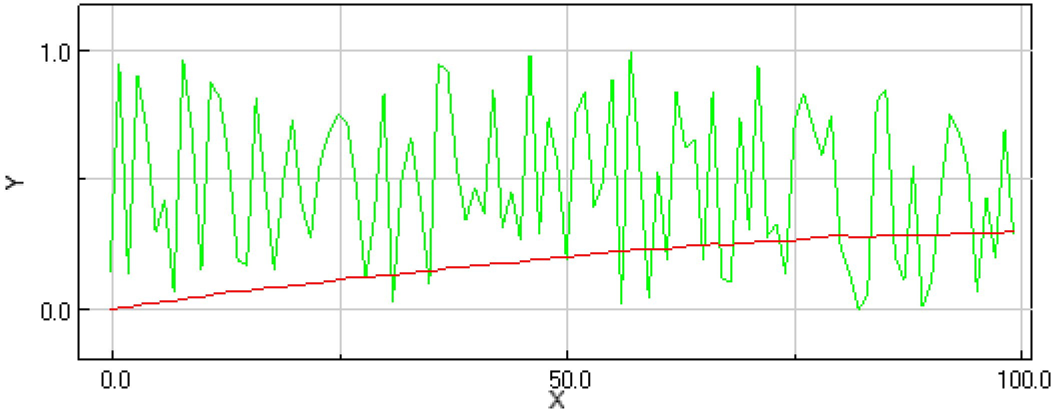
Average Filter
Ex) ex/ml/l10iir.py

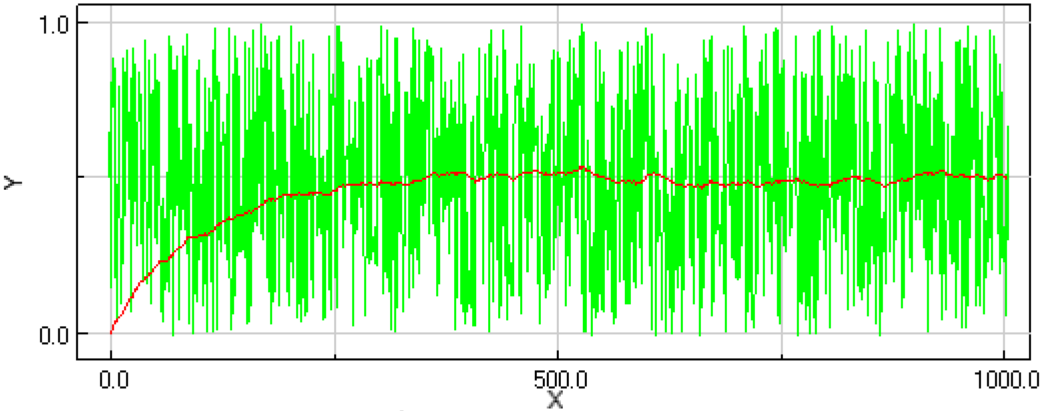
- IIR Filter : $f_{k+1} = \alpha x_k + (1-\alpha) f_k$



```
def demo2():
    figure(1)
    clear()

    s = 0;
    n = 1000
    for i in range(0,n):
        x= rand()
        s = s*0.99 + x*0.01

        graph(1)
        plot(x,'g')
        graph(2)
        plot(s,'r')
```

Random x = 100





Random x = 1000

- S becomes
- averaged value, 0.5.

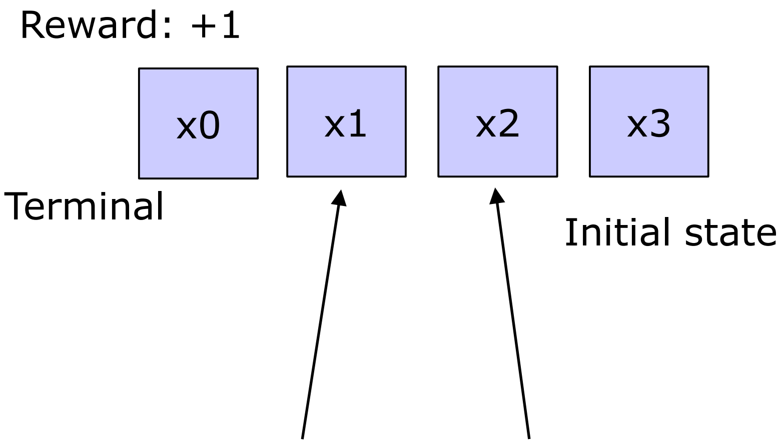
Important Meaning of Return 1

- Think next two cases
  - Case 1) X3→X2→X1→X0
  - Case 2) X3→X2→ X3→X2→ X3→X2→X3→X2→X1→X0
- With Negative Reward(eg, -0.1)
  - Case 1) -0.1*2+1 = 0.8(Return)
  - Case 2) -0.1*8+1 =0.2(Return)
  - 0.8 is better than 0.2.

- Without Negative Reward
  - Case 1) 0*2 + 1= 1
  - Case 2) 0*8+1 = 1
  - Question : case 1) and case 2) are equal?????

Important Meaning of Return 2

- We Must think that Returns will be Expected.
  - The Returns of Case 1) and Case 2) will be averaged.
- After Many cases are averaged, what happens?



Reward: +1

x0

x1

x2

x3

Terminal

Initial state

$$E\{R\} at\ s = x_1 \ > \ E\{R\} at\ s = x_2$$

Why?

Because, x1 has Bigger probability to reach x0 than x2.

Why?
X1 is closer to X0

Expected Return finds optimality without Negative Reward

- Remind that -0.1 reward is **helpful** to find the optimality
  - Long distance journey is NOT good for an agent.
  - Case 1) X3→X2→X1→X0 (best) → -0.1*2+1 = 0.8
  - Case 2) X3→X2→X3→X2→X3→X2→X3→X2→X1→X0 (poor) → -0.1*8+1 =0.2
- But, without negative reward, **expected return is also good** for which direction is Good or Not.
- Anyway, we can introduce the accelerating method by using discounted return.

Summary of RL

- Future Reward
  - If an agent moves in future, how much reward does an agent obtains? (Not the past reward)

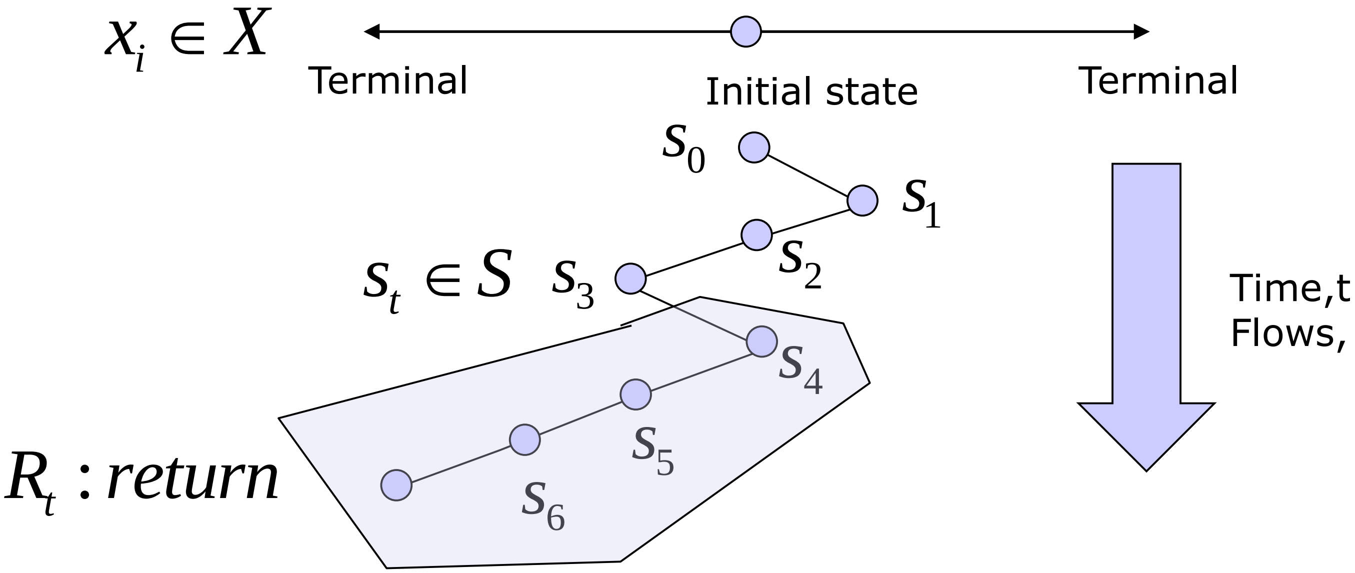
- Return = sum of all possible future reward

- Discounted Return : $R_{t=0} = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$
  - When a reward is far from the current state, discounted rate is larger.
  - This makes an effect on finding the optimal path without wasting repetitive state transitions like [3,2,3,2,3,2,3,2,1,0]

- Episode : one sequence from initial to terminal state

2 Monte-Carlo(MC) method

Monte Carlo (MC) Method



- If a state, s is equal to a position at x,

$$if \;\; s_t = x_i \quad \Rightarrow V(s_t) = V(x_i)$$

- From state, s, we can tell the function of position x.

Monte Carlo (MC) Method

- Expected Return= State value Function

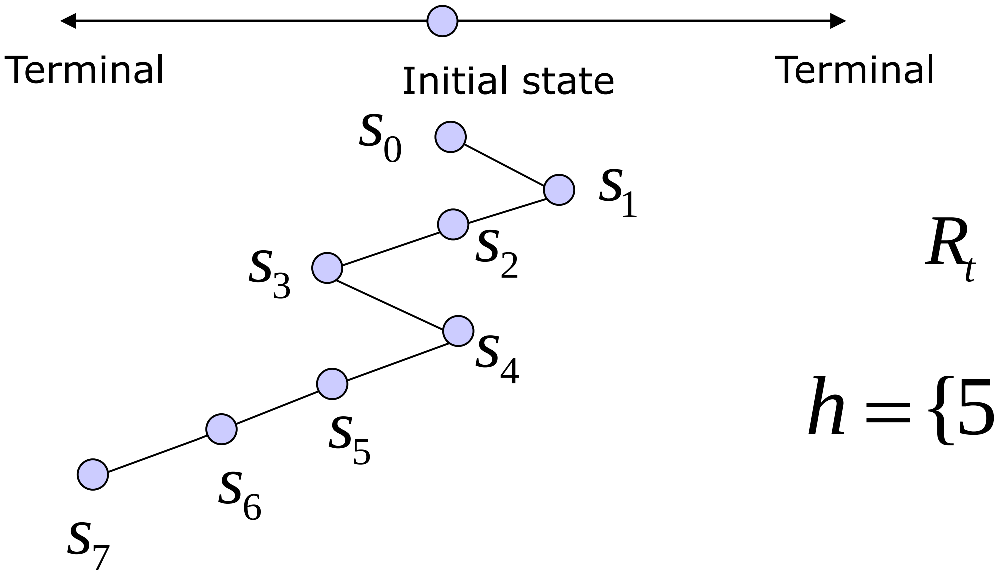
$$E(R_t) = E\left(\sum_{k}^{\infty} r(s_k)\right)$$
$$= E\left(r(s_k) + r(s_{k+1}) + r(s_{k+2}) + r(s_{k+3}) + \ldots\right) = V(s)$$

- Monte Carlo: Update V(s) with Return R along saved state transition history
  - MC does not use discounted return, but uses Return.

$$h = \{x_5, x_6, x_5, x_4 \ldots\ldots x_{\text{terminal}}\} \quad if \;\; s_t = x_i$$

$$V(s') = (1-\alpha)V(s) + \alpha R_t \text{ along all history,h}$$

Example of MC Method



$$R_t = \sum_{k=1}^{\infty} r_{t+k} = 1$$

$$h = \{5, 6, 5, 4, 5, 3, 2, 1, 0\}$$

$$V(5) = (1-\alpha)V(5) + \alpha R = \alpha$$

$$V(6) = (1-\alpha)V(6) + \alpha R = \alpha$$

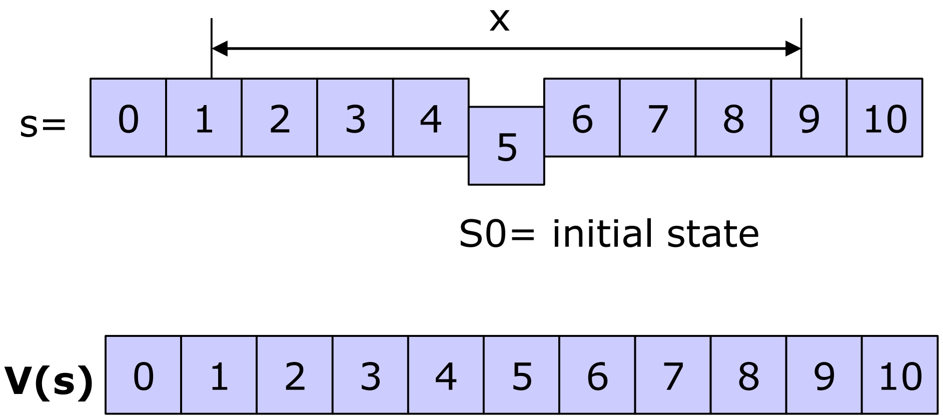
$$V(5) = (1-\alpha)V(5) + \alpha R = (1-\alpha)\alpha + \alpha$$

$$V(4) = (1-\alpha)V(4) + \alpha R = \alpha$$

...

Example of MC Method, l10mc1. py

- +1 reward at left, +2 reward at right, otherwise r=0
- How it works



s=	0	1	2	3	4	5	6	7	8	9	10

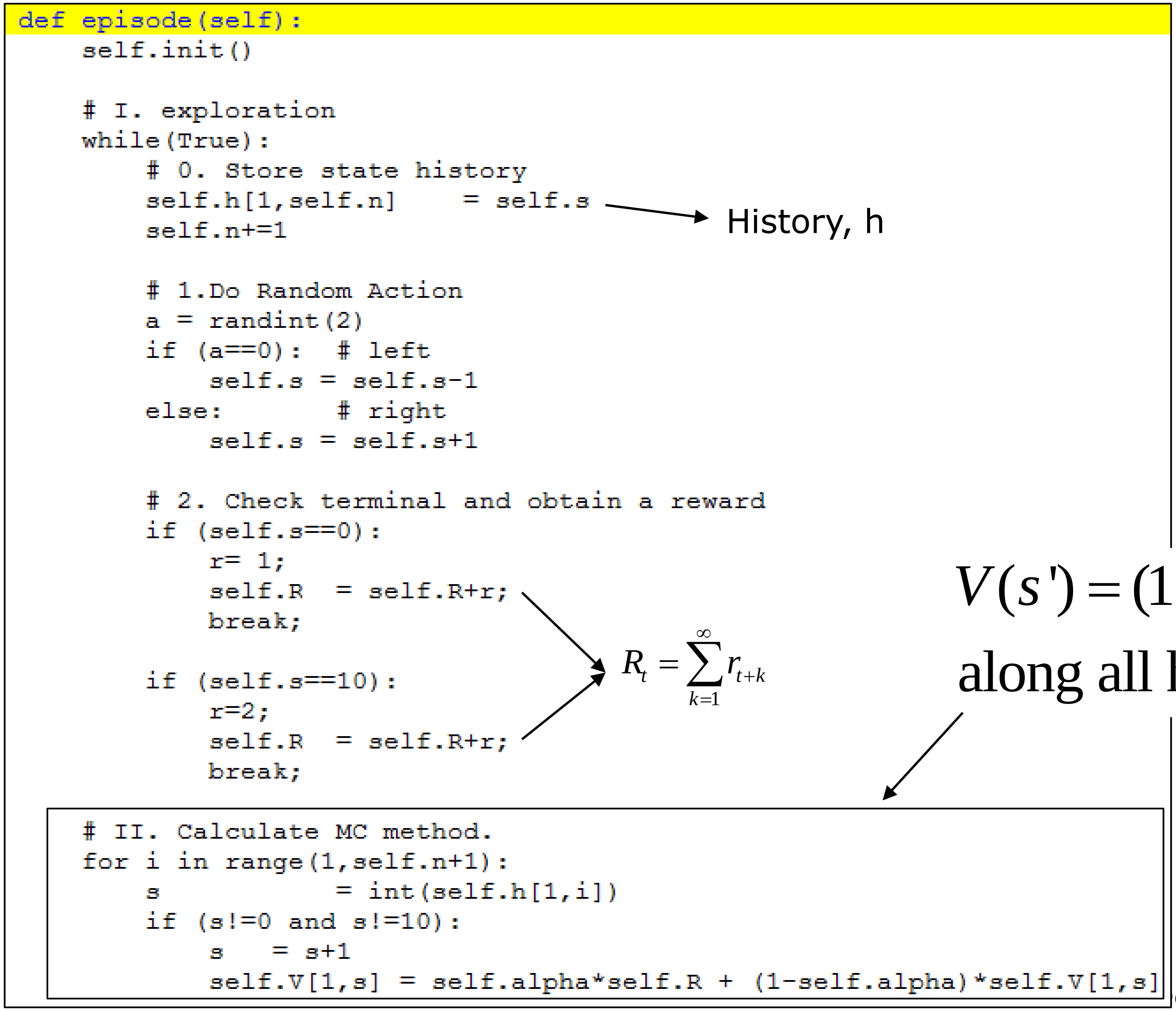
S0= initial state

V(s)	0	1	2	3	4	5	6	7	8	9	10

$$h = \{5, 6, 5, 4, 5, 3, 2, 1, 0\}$$

```
class MC:
    s0   = 5
    s    = s0
    a    = 0;
    R    = 0;
    V    =array(1,1);
    h    =array(1,1);
    x    =linspace(1,9,9)
    n    =1;
    alpha=0.01;

    def __init__(self):
        self.V   = zeros(1,10)

    def init(self):
        self.s   = self.s0
        self.R   = 0;
        self.h   = array(1,1);
        self.n   = 1;
```

Example of Episode



```
def episode(self):
    self.init()

    # I. exploration
    while(True):
        # 0. Store state history
        self.h[1,self.n]     = self.s
        self.n+=1

        # 1.Do Random Action
        a = randint(2)
        if (a==0):  # left
            self.s = self.s-1
        else:       # right
            self.s = self.s+1

        # 2. Check terminal and obtain a reward
        if (self.s==0):
            r= 1;
            self.R  = self.R+r;
            break;

        if (self.s==10):
            r=2;
            self.R  = self.R+r;
            break;

    # II. Calculate MC method.
    for i in range(1,self.n+1):
        s            = int(self.h[1,i])
        if (s!=0 and s!=10):
            s   = s+1
            self.V[1,s] = self.alpha*self.R + (1-self.alpha)*self.V[1,s]
```

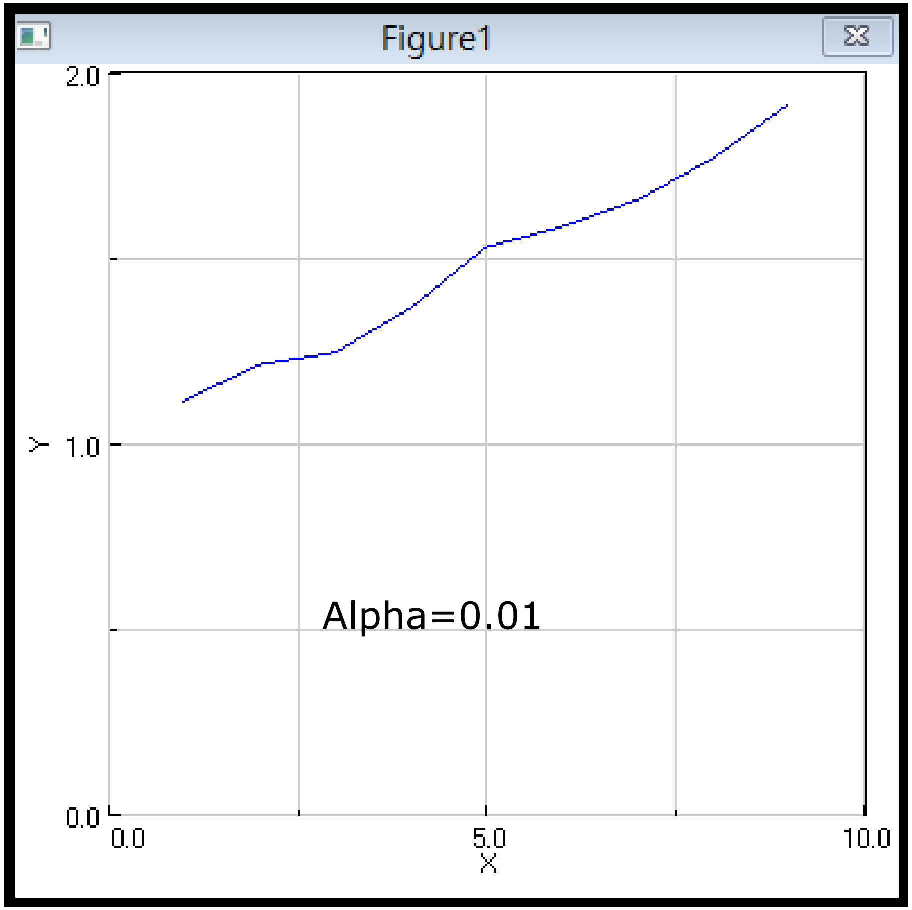
History, h

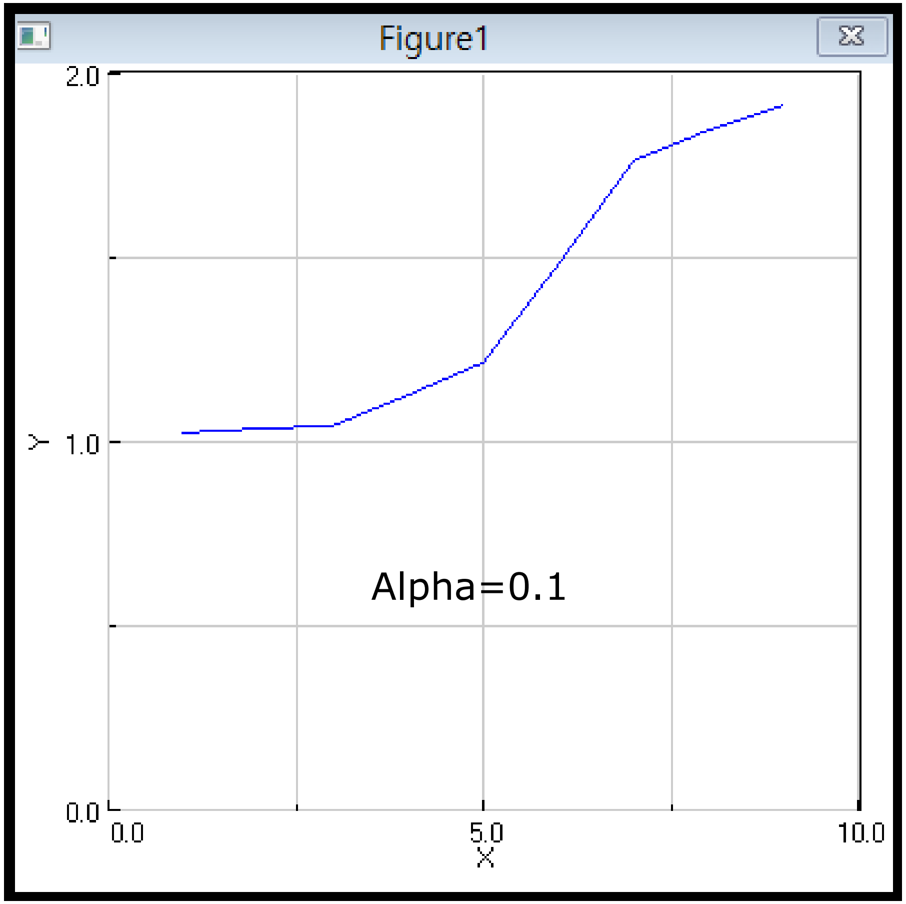
$$R_t = \sum_{k=1}^{\infty} r_{t+k}$$

$$V(s') = (1-\alpha)V(s) + \alpha R_t$$

along all history,h

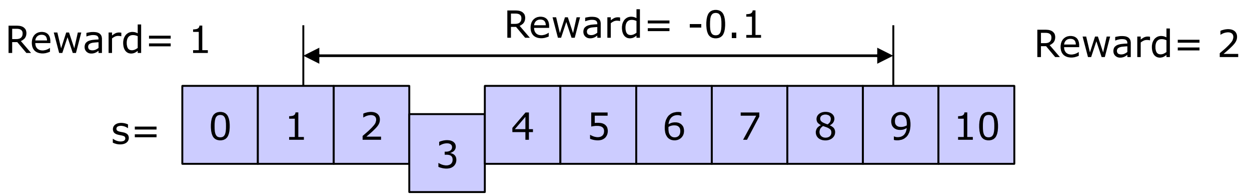
Example results with 1000 Episodes

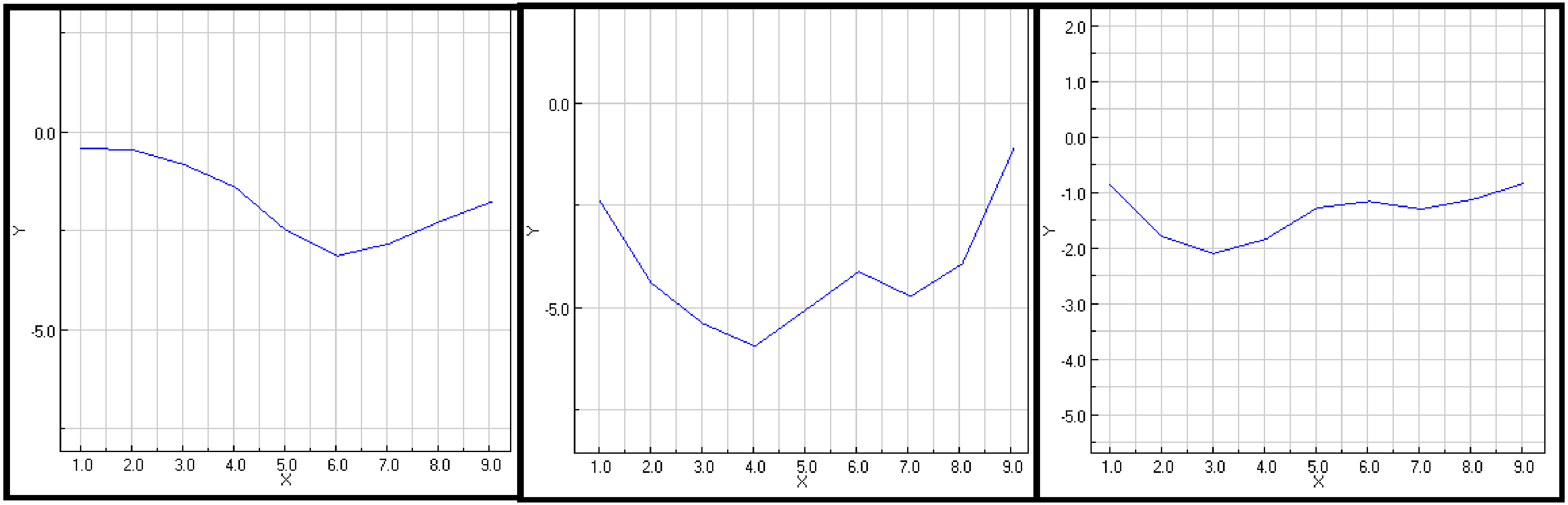




- V(s) says that Right Direction is better

Example of More Complex Cases, l10mc2

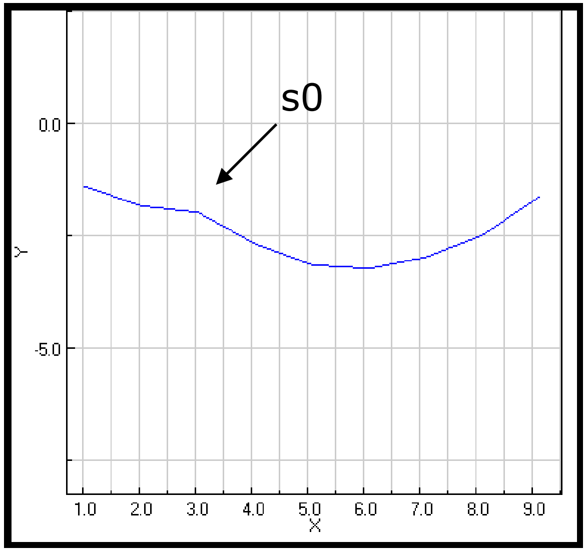


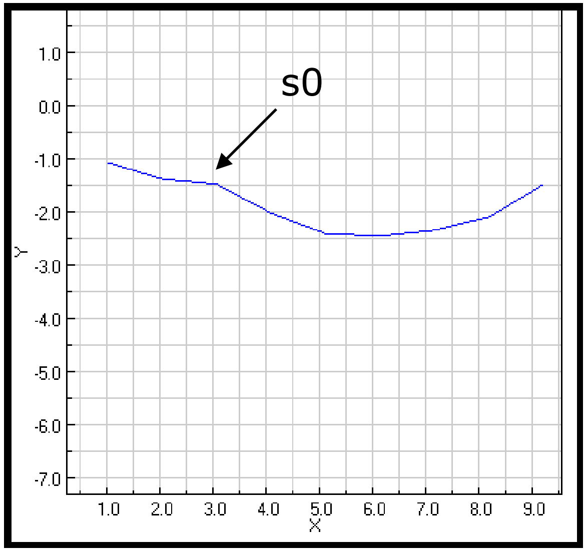


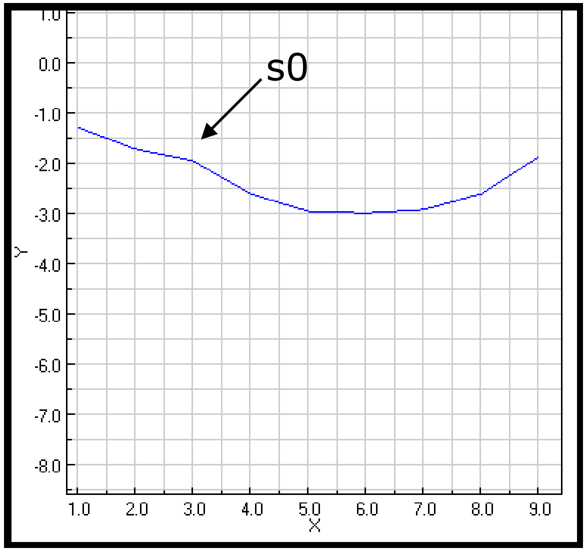
5000 episodes with alpha=0.01

From these results, it is not easy to say which one is better

5000 Episodes with low alpha value(0.001)







- When number of episodes increases, low alpha value contributes for convergences, but it is not so tough.
- The results says that RL gives us determination in the more detailed ways

Summary of Monte-Carlo Method

- MC directly uses Return for update state value.
  - It is very Intuitive method.
  - MC is often used for verifying system characteristics.
  - Many casino games are analyzed by MC.. ^^
- MC does not use Discounted Return,
  - No gamma
- Shortcomings:
  - MC stores all history of state transitions
  - If state transition becomes longer, it becomes a handicap.

3 Discounted Return

Discounted Return

- Discounted return is using the weighted reward.
- Far future rewards are strongly reduced.
- Near future rewards are slightly reduced.
- eg. S3→S2→ S3→S2→ S3→S2→…... S3→S2→S1→S0
  - Far future rewards are meaningless.
  - The result of long journey becomes neglected….
  - Gamma Reduction Ratio is used.

Definition of Discounted Return

- Discounted Return

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \qquad (0<\gamma<1)$$

- Why Discounted Return is effective without -0.1 rewards
  - Best case, s= [3, 2, 1, 0] reward +1 at s=0

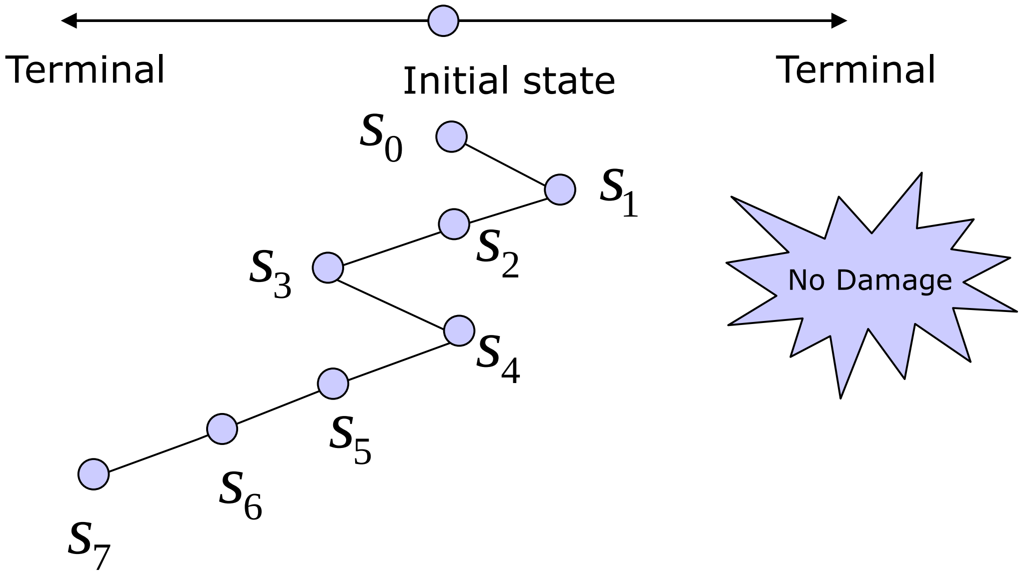
$$R_{t=0}\ (or\ \mathrm{R}_{s=s_0}) = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = \gamma^0 0 + \gamma^1 0 + \gamma^2 1 = \gamma^2$$

  - Not an optimal case, s=[3,2,3,2,1,0] reward + at s=0

$$R_{t=0}\ (or\ \mathrm{R}_{s=s_0}) = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = \gamma^0 0 + \gamma^1 0 + \gamma^2 0 + \gamma^3 0 + \gamma^4 1 = \gamma^4$$

  - Which one is a larger Return? $\gamma^2 > \gamma^4$

Examples of a Single Discounted Return



$$R_{t=0} = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

$$R_{t=0}(s = s_0) = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = \sum_{k=0}^{6} \gamma^k r_{0+k+1} = (\gamma^0 r_1 + \gamma^1 r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \gamma^4 r_5 + \gamma^5 r_6 + \gamma^6 r_7)$$

$$= 0 + \gamma^6 r_7 = \gamma^6 1 = \gamma^6$$

$$\Rightarrow R_{t=4}(s = s_4) = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} = \sum_{k=0}^{3} \gamma^k r_{4+k+1} = \gamma^0 r_5 + \gamma^1 r_6 + \gamma^2 r_7$$

$$= 0 + \gamma^2 r_7 = \gamma^2 1 = \gamma^2$$

$$\therefore R_{t=0}(s = s_0) = \gamma^6 < R_{t=4}(s = s_4) = \gamma^2$$

State Value, V(s)
Stochastic version of Discounted Return

- Expected Discounted Return (=State value)
  - Average of all future reward. Remember that there are many paths.

    ex) S=[3,2,3,2,1,0] , S=[3,2,3,2,3,2,1,0] , S=[3,4,3,2,1,0]
  - We need to average all possible cases → Expectation

$$E(R_{t,s=s_k}) = E\left(\sum_{j=0}^{\infty} \gamma^j r(s_{k+j})\right)$$

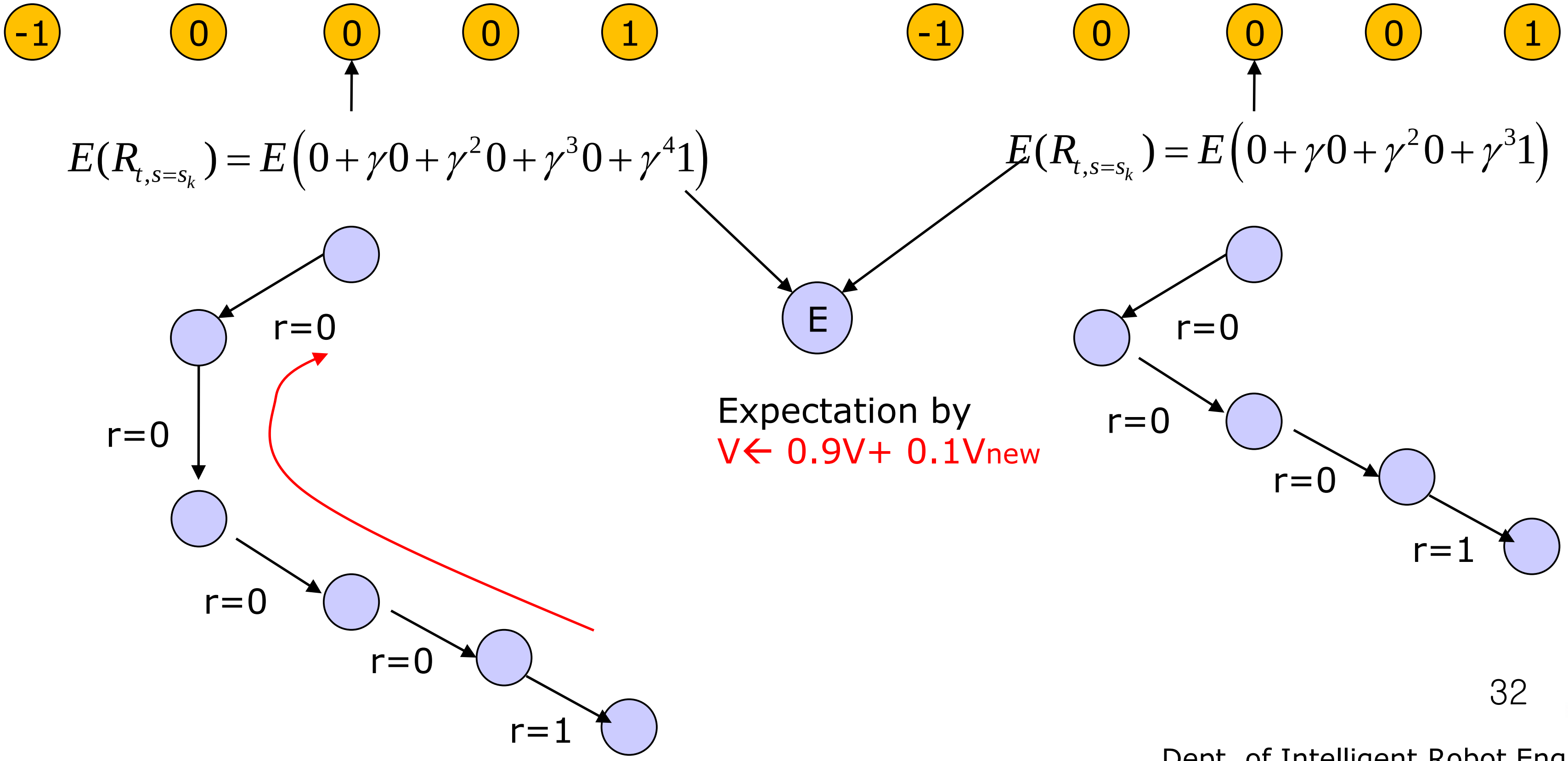
$$= E\left(r(s_k) + \gamma r(s_{k+1}) + \gamma^2 r(s_{k+2}) + \gamma^3 r(s_{k+3}) + \ldots\right)$$

- Definition of State Value, V(s) $V(s) \triangleq E(R_t)$

Meaning of Discounted Return

- Path information is resolved in **State Value**.

$$E(R_{t,s=s_k}) = E\left(r(s_k) + \gamma r(s_{k+1}) + \gamma^2 r(s_{k+2}) + \gamma^3 r(s_{k+3}) + \ldots\right)$$



$$E(R_{t,s=s_k}) = E\left(0 + \gamma 0 + \gamma^2 0 + \gamma^3 0 + \gamma^4 1\right)$$

$$E(R_{t,s=s_k}) = E\left(0 + \gamma 0 + \gamma^2 0 + \gamma^3 1\right)$$

Expectation by

V← 0.9V+ 0.1V_{new}

RL Summary

- Return :
  - sum of all possible rewards
- Discounted Return:
  - sum of all discounted rewards using gamma
- Expected Return: average of (discounted) return

  = State value, V(s)
- Episode : one sequence from initial to terminal state
- State value estimation with Two Different methods
  - 1. Monte-Carlo Method
  - 2. Temporal Difference Method

4 Temporal Difference

Temporal Difference in RL

- Back to State Value Definition

$$E(R_{t,s=s_k}) = E\left(r(s_k) + \gamma r(s_{k+1}) + \gamma^2 r(s_{k+2}) + \gamma^3 r(s_{k+3}) + \ldots\right)$$

- State value

$$V(s) \triangleq E(R_t)$$

- Without History information → Temporal Difference

$$V(s_k) = E\left(r(s_k) + \gamma r(s_{k+1}) + \gamma^2 r(s_{k+2}) + \gamma^3 r(s_{k+3}) + \ldots\right)$$

$$= E(r(s_k)) + \gamma E\left\{r(s_{k+1}) + \gamma^1 r(s_{k+2}) + \gamma^2 r(s_{k+3}) + \ldots\right\}$$

$$= r + \gamma V(s_{k+1})$$

Temporal Difference: The Crucial Idea in RL

- Observe the Current State, s
- State value: V(s)
- Random Movement by Action: a

$$s \xrightarrow{a} s'$$

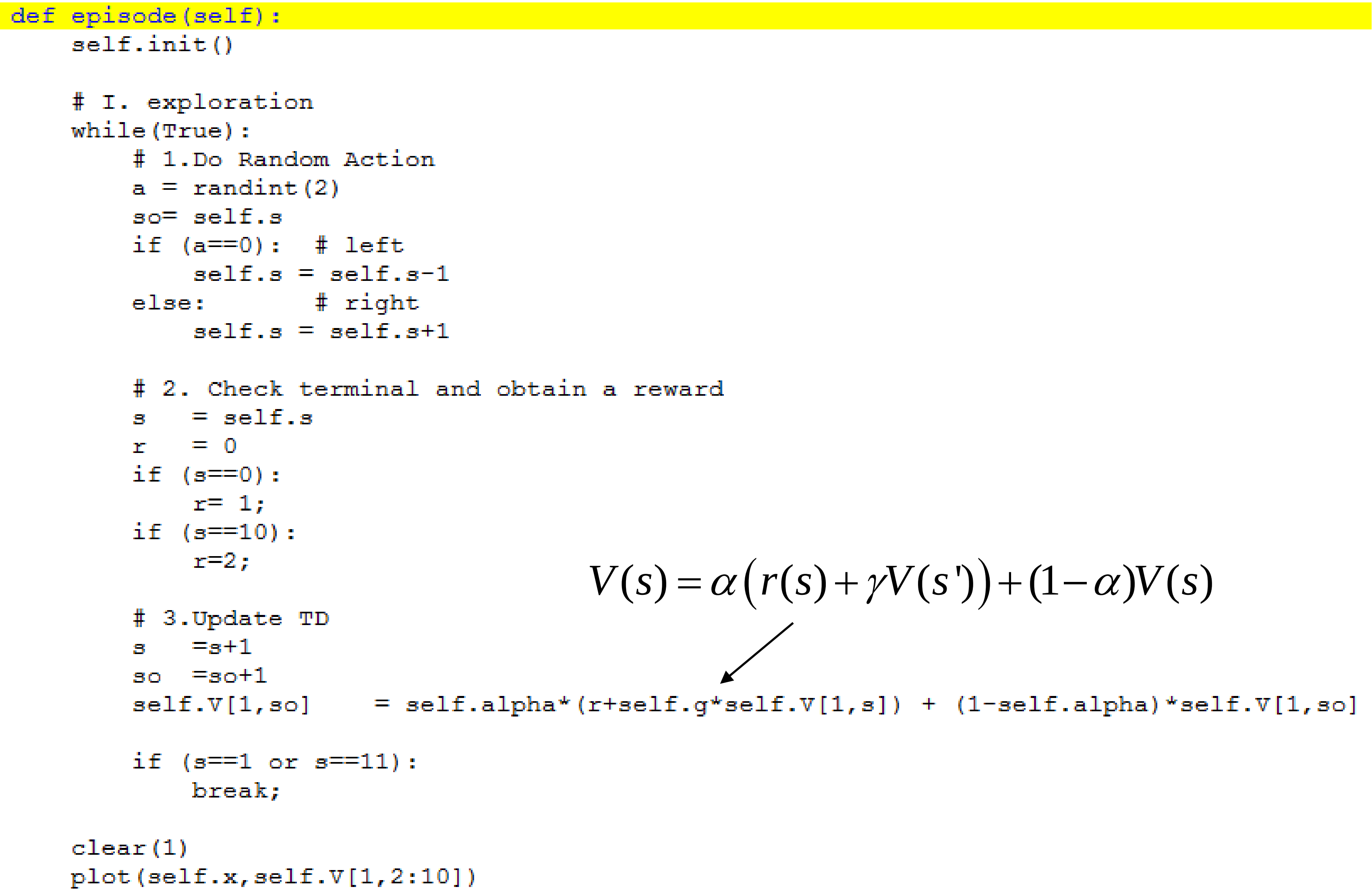
- Sense-and-action
- Update State Value, V

$$V(s) = r(s) + \gamma V(s')$$

- Think expectation by alpha (0.01 in general)

$$\therefore V(s) = (1-\alpha)V(s) + \alpha\left(r(s) + \gamma V(s')\right)$$

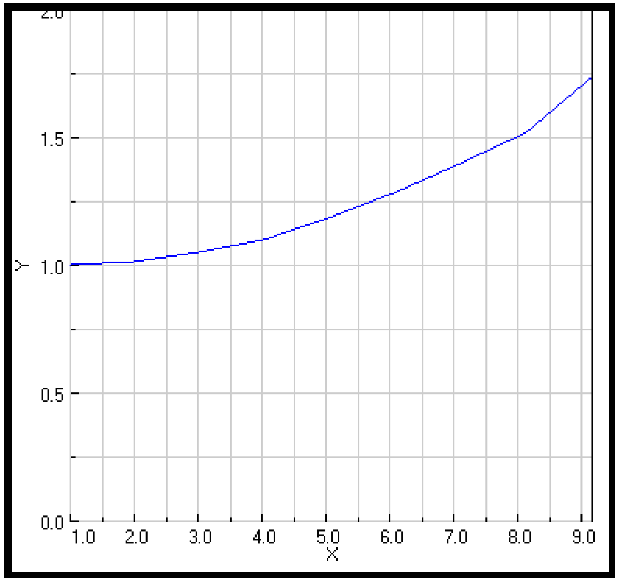
Example of l10td1.py



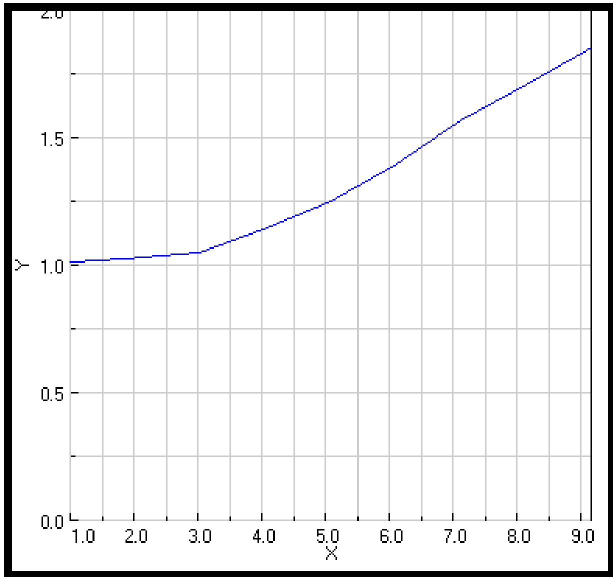
```
def episode(self):
    self.init()

    # I. exploration
    while(True):
        # 1.Do Random Action
        a = randint(2)
        so= self.s
        if  (a==0):  # left
            self.s = self.s-1
        else:        # right
            self.s = self.s+1

        # 2. Check terminal and obtain a reward
        s   = self.s
        r   = 0
        if  (s==0):
            r= 1;
        if  (s==10):
            r=2;

        # 3.Update TD
        s   =s+1
        so  =so+1
        self.V[1,so]    = self.alpha*(r+self.g*self.V[1,s]) + (1-self.alpha)*self.V[1,so]

        if  (s==1 or s==11):
            break;

    clear(1)
    plot(self.x,self.V[1,2:10])
```

$$V(s)=\alpha\left(r(s)+\gamma V(s')\right)+(1-\alpha)V(s)$$

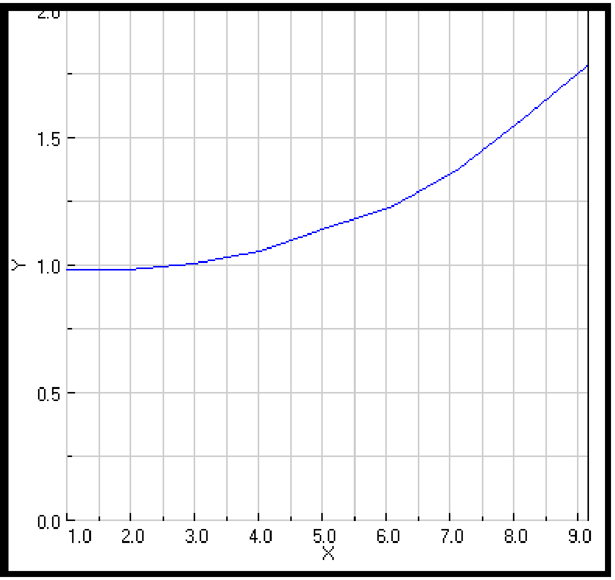
Result of l10td1



1000 episodes with alpha=0.1



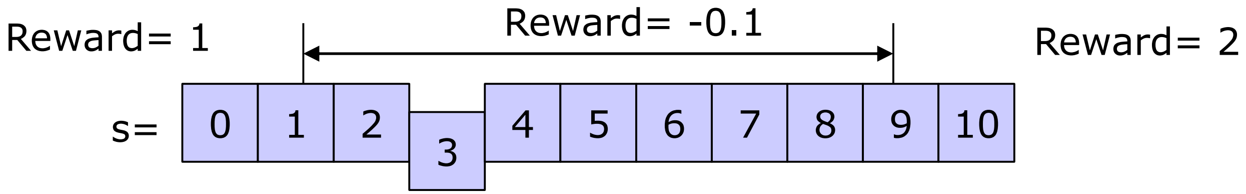
2000 episodes with alpha=0.1



2000 episodes with alpha=0.01

- MC shows nearly STRAIGHT Line.
- TD shows Curved results, Why?
  - Think Gamma

Example of More Complex Cases, l10td2



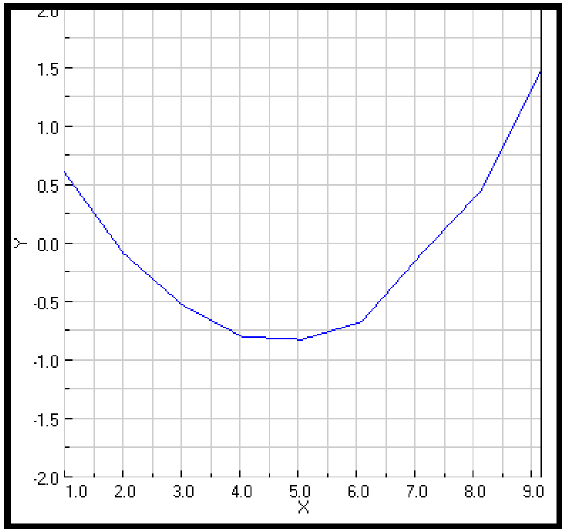
Reward= 1

Reward= -0.1

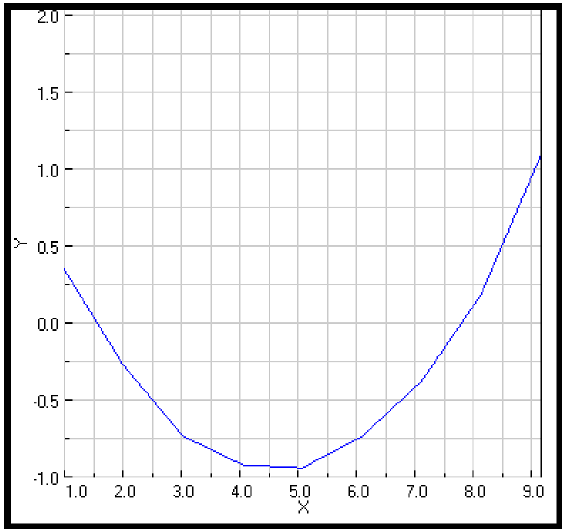
Reward= 2

s=

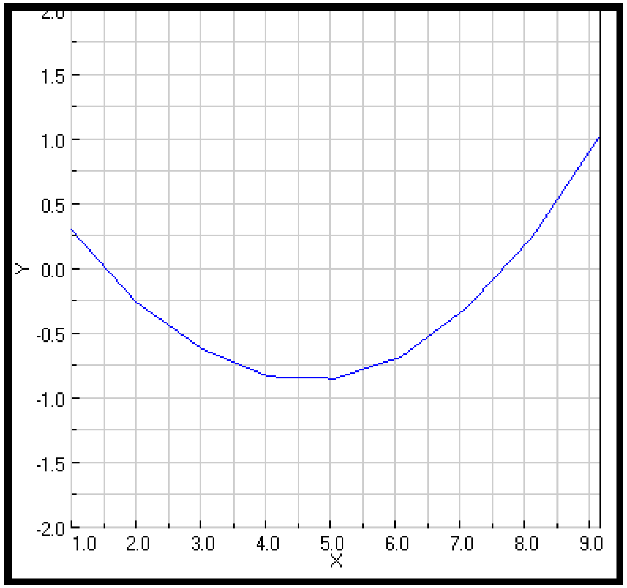
0	1	2	3	4	5	6	7	8	9	10



1000 episodes with
alpha=0.1



1000 episodes with
alpha=0.01



2000 episodes with
alpha=0.01

- TD shows better performance than MC

5 HW. MC and TD

Ex-1) Baskin Robbins Game

- Initial state, S=0
- Terminal state, S=31
- RL Agent says 1,2, or 3.
- Then we says 1,2, or 3.
- Finally, RL wins if you says the number over 31.

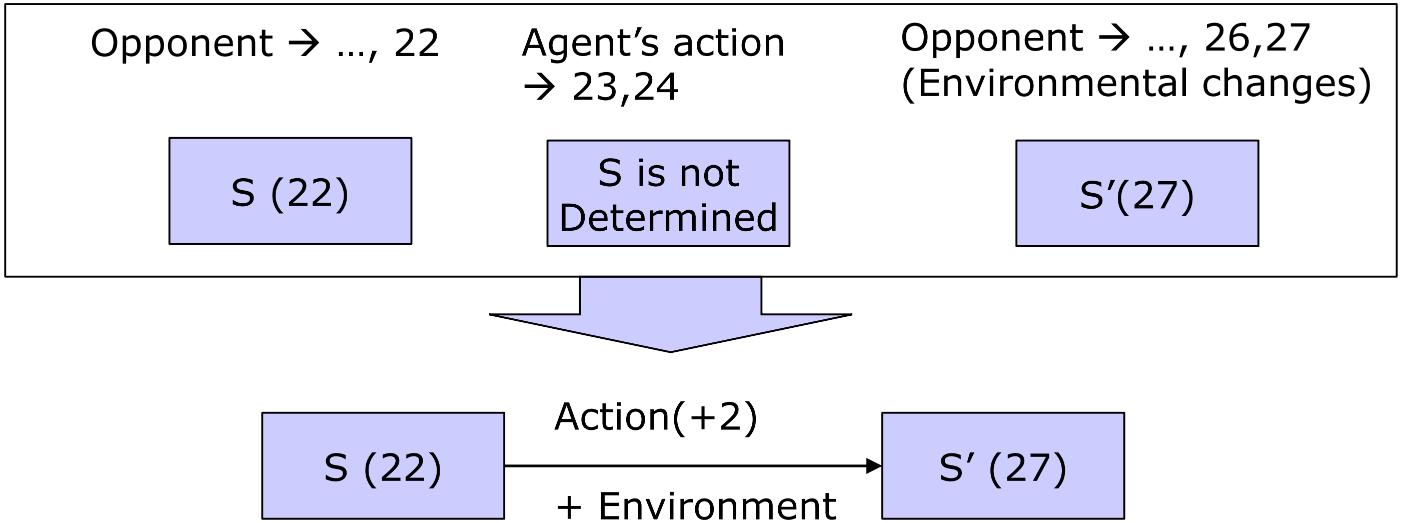
- Reward
  - If RL loses, RL obtains -1
  - If RL wins, RL obtains +1.

- How it works?...

Baskin Robbins 31 Game

- Example
  - Agent 1,2 678, , …. , 23,24, ,28,29,30
  - Opponent 345, 9,10,11 … 22 26,**27** ,31
  - Opponent speaks 31 and loses a game.

- RL designs



Opponent → …, 22

Agent's action
→ 23,24

Opponent → …, 26,27
(Environmental changes)

S (22)

S is not Determined

S'(27)

S (22)

Action(+2)

+ Environment

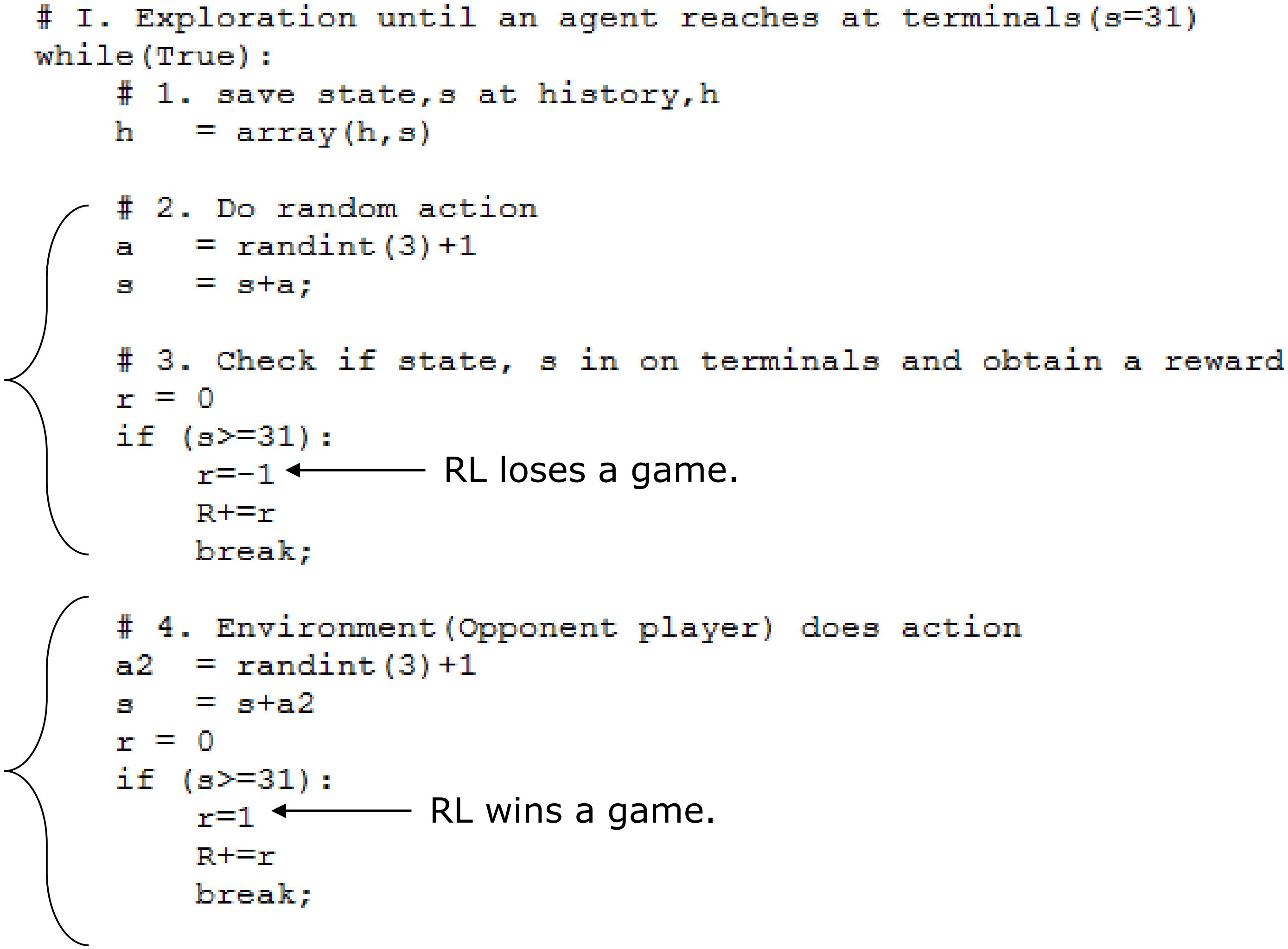
S' (27)

Hint for Every Problems.

- In Baskin Robbins game, the next state is NOT determined Because your turn is added.
  - RL moves from 0 to 3, then your turn moves from 3 to 4~6.
  - RL feels that action 1, 2, or 3 can move from 2 to 6.
  - Thus, RL works on stochastic way.

- Like what you did in Baskin Robbins game, RL results says that RL obtains the best reward at 27.

How to Build Baskin Robbins Game? MC example



```
# I. Exploration until an agent reaches at terminals(s=31)
while(True):
    # 1. save state,s at history,h
    h   = array(h,s)

    # 2. Do random action
    a   = randint(3)+1
    s   = s+a;

    # 3. Check if state, s in on terminals and obtain a reward
    r = 0
    if (s>=31):
        r=-1
        R+=r
        break;

    # 4. Environment(Opponent player) does action
    a2  = randint(3)+1
    s   = s+a2
    r = 0
    if (s>=31):
        r=1
        R+=r
        break;
```

RL's turn

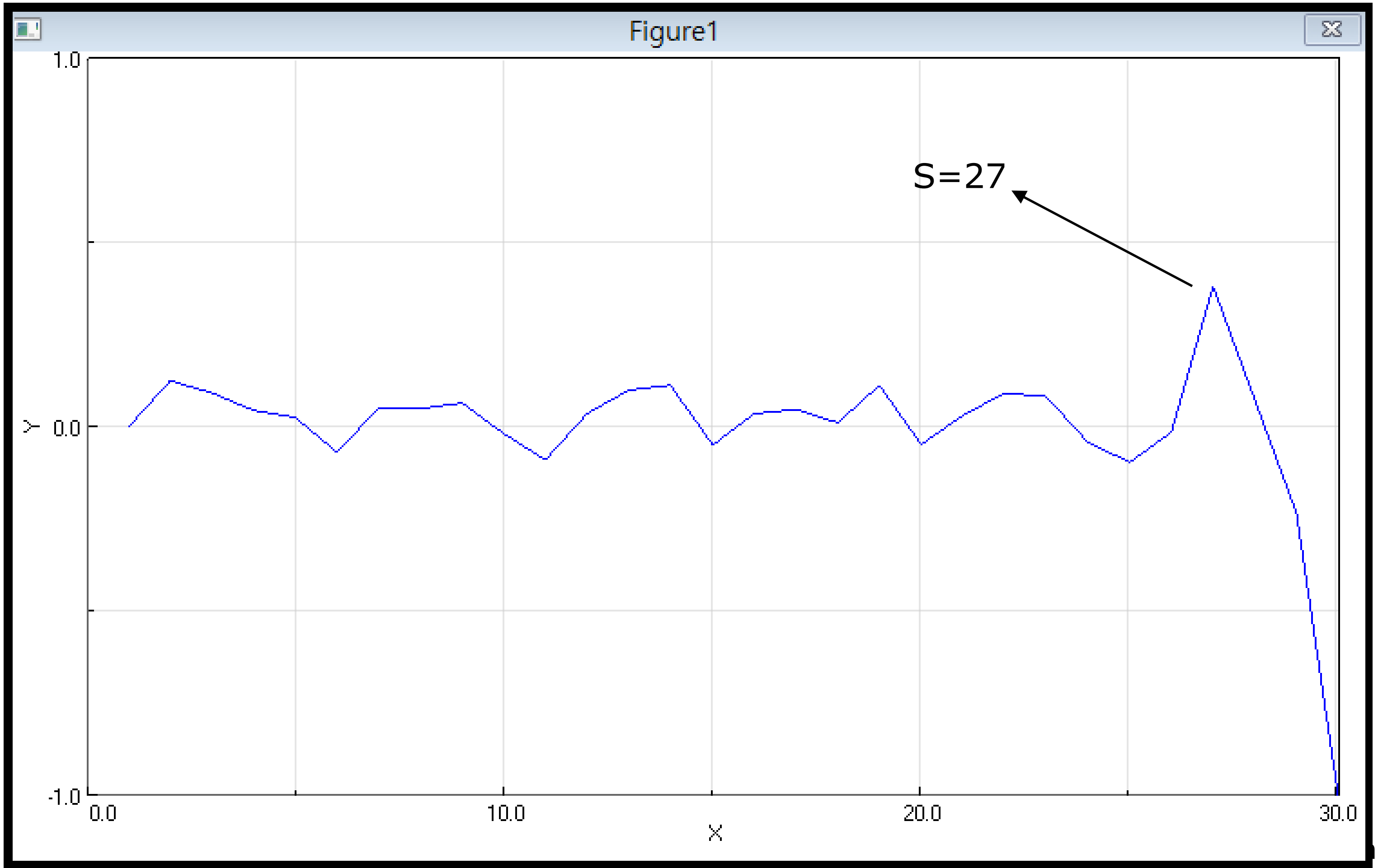
RL loses a game.

Your turn

RL wins a game.

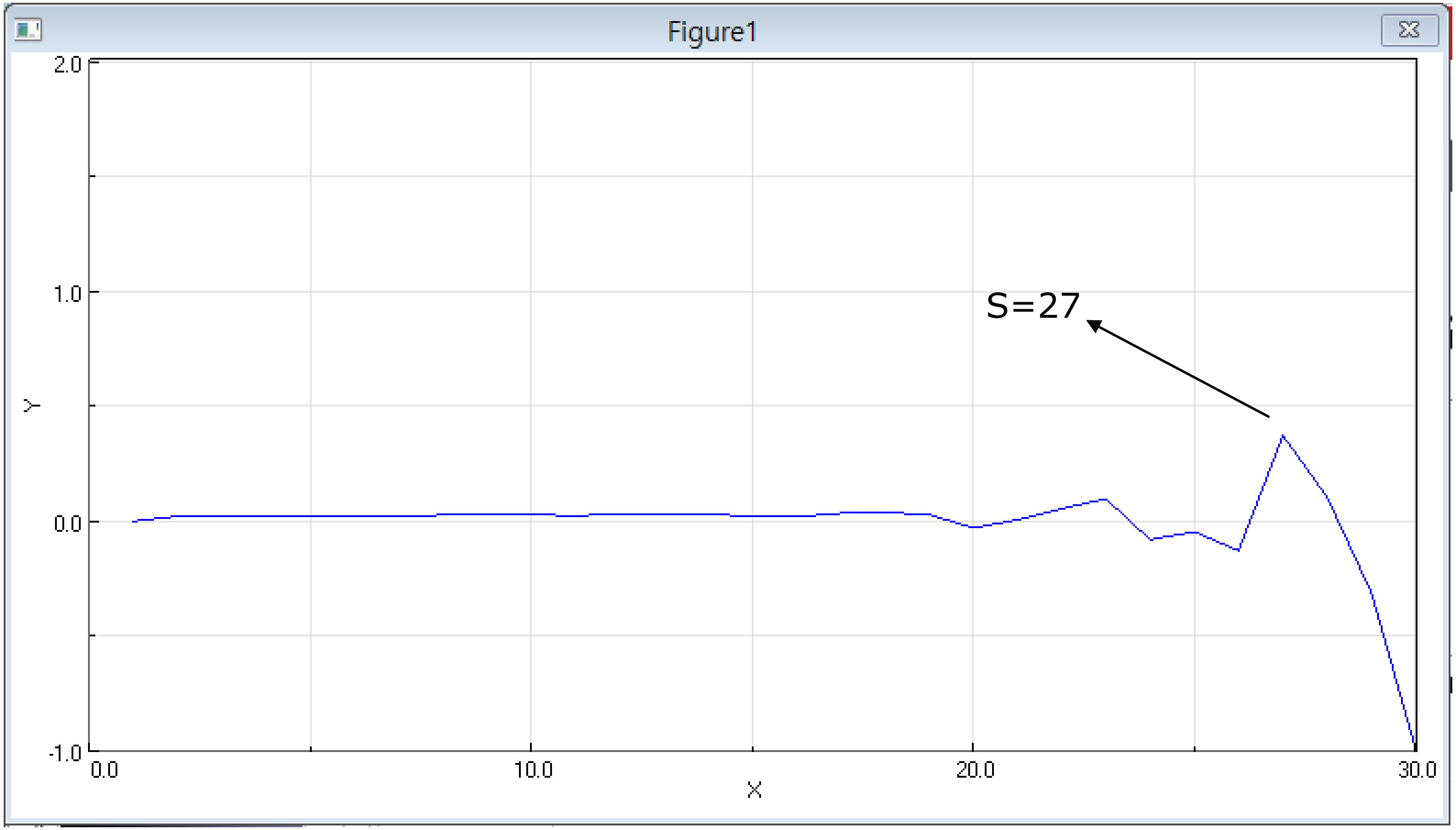
Prob.1. Complete "YOUR" Baskin Robbins Game with MC

- Example of MC result



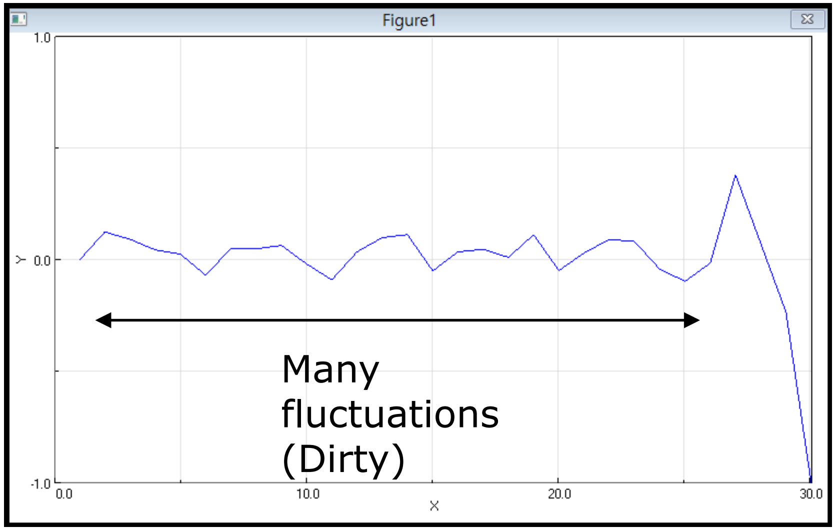
Prob.2. Complete "Your" Baskin Robbins Game with TD

- Example of TD result



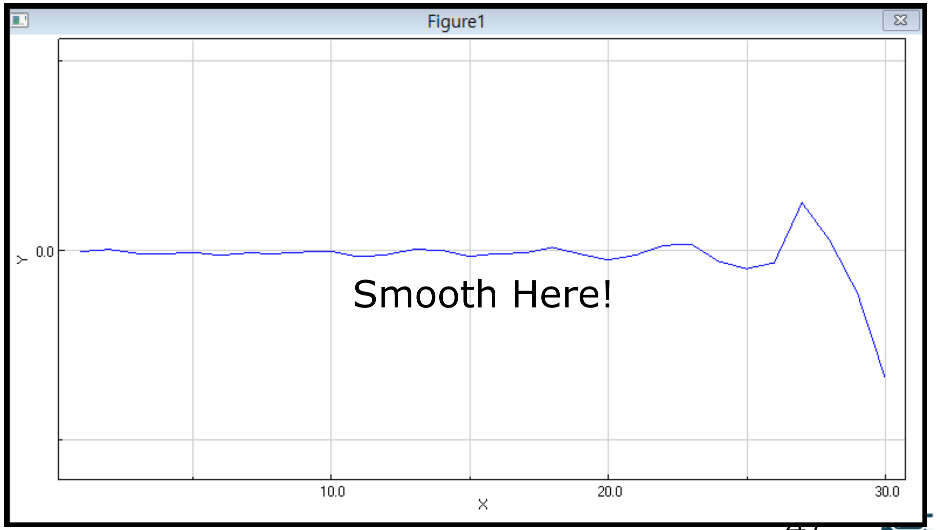
Discussion

- Prob. 3. Explain Why 27 is so important?
- Prob. 4.a. Why MC has so many fluctuations?
- Prob. 4.b. How can we REDUCE many fluctuations like below result? <u>Show your Result</u>



Many fluctuations (Dirty)

Prob. 4.a

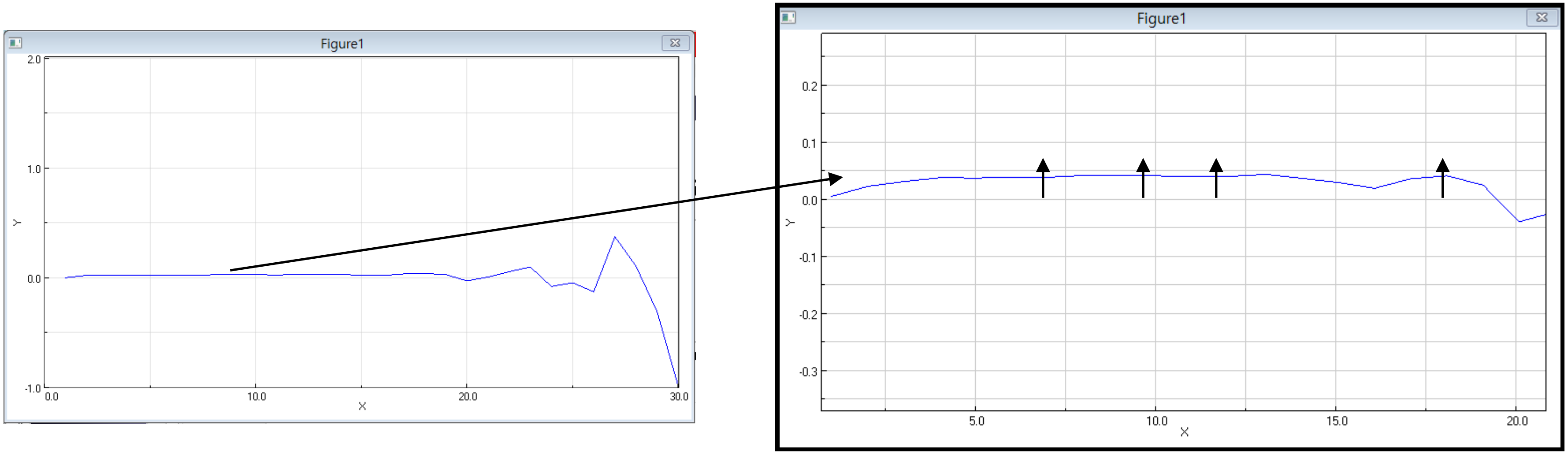


Smooth Here!

Prob. 4.b

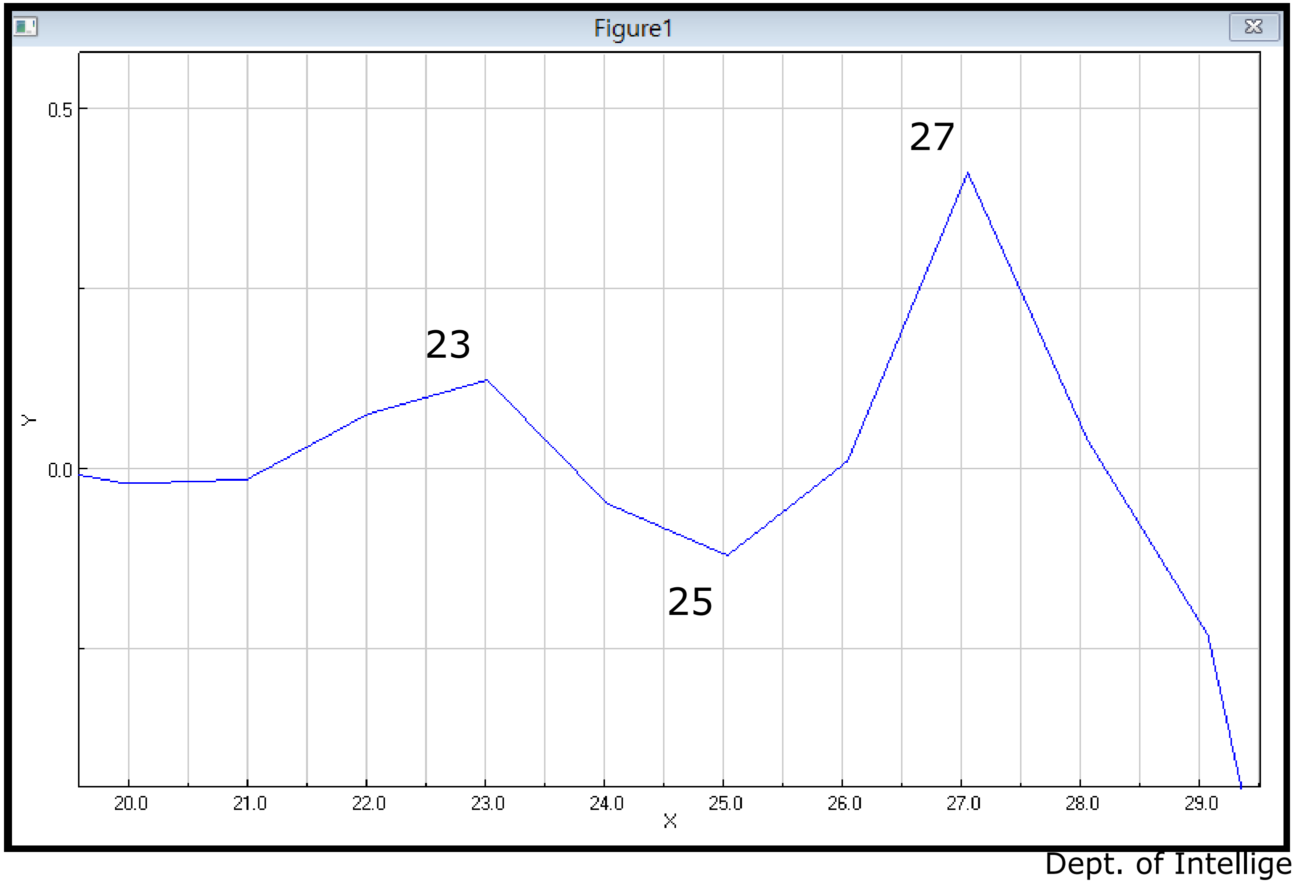
Prob.5. Discussion about TD Results

- Prob. 5.a. From TD Results, V(s) is slightly positive from s=0 to s=20. What is the meaning of it?



Prob. 5.b.

- Prob. 5.b. After 2000, 4000,and 6000 episodes, TD shows this tendency.
  - 23 is better than 25, and 27 is better than 23.
  - What is the meaning of it?



Ex-2) Q-Learning : l9q1.py

- Q-learning has two modes.
- 1. Exploration: random searching for update Q value

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

- 2. Exploitation: Following Maximum Q value
  - – An agent follows Maximum Q value
  - – Argmax(Q(s,a) = a* → Best policy(action)