Lecture I1. Policy Gradient Theorem

Sim, Min Kyu, Ph.D., mksim@seoultech.ac.kr

SEOULTECH 서울과학기술대학교 데이터사이언스학과

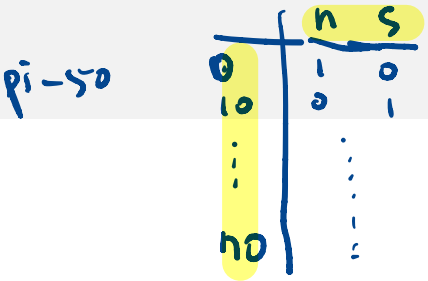
1. I. Motivation
2. II. Mathematical Foundation of Policy Gradient
3. III. Problem statement

I. Motivation

Recap: Value-based approach

- $q(s, a)$ stands for the expected value function given state s and action a.
- In ~~H1, p10~~
  - `q_s_a` tabularized the $q(s, a)$.
- In ~~H1, p11~~
  - `q_net()` functionally approximates $q(s, a)$.
  - If the set of parameter is written as ω, then $q(s, a) \approx q_\omega(s, a)$
  - In other words, $q(s, a)$ is parameterized using the set of parameter ω.
- These efforts so far are called 'value-based' approach.
  - First, it approximates the state-action value function $q()$.
  - Then, use the value function to derive optimal policy π^*.

Policy-based approach



Recap

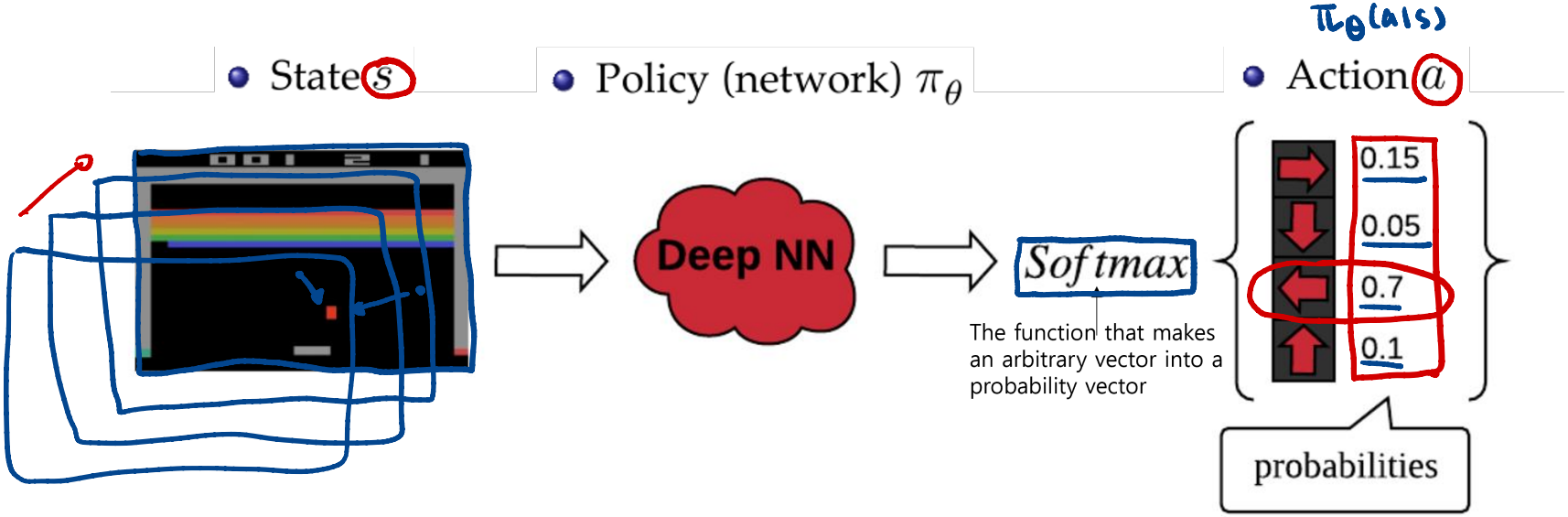
- Remind that π is a mapping from state to action. In other words, $\pi : \mathcal{S} \to \mathcal{A}$
- In `E1,p21` we defined this policy function as
  - $\pi(s)$ returns a single action on the state s under the policy π ex) $\pi(10) = s$, $\pi(10) = s$ ..
  - $\pi(a|s)$ returns the probability of choosing an action a on the state s under the policy π ex) $\pi(a|\text{state}=20) = [0.7 \quad 0.3]$. ($\pi(\text{normal}|20) = 0.7$, $\pi(\text{speed}|20) = 0.3$)
  - In other words, $\pi(a|s) = \mathbb{P}(A_t = a|S_t = s)$

Policy-based approach

$q(s,a) \approx q_w(s,a)$: value based
(classic) (deep)

- Policy-based approach parameterizes the function $\pi()$ with the set of parameters θ. In other words,
$$\pi(s,a) \approx \pi_\theta(s,a) = \mathbb{P}(A_t = a|S_t = s, \theta_t = \theta) = \mathbb{P}_\theta(A_t = a|S_t = s)$$
- The last term reads: 'the probability that the action a is taken given that the state s under the policy parameterized with θ.'

The schematic diagram for policy-gradient



- The goal is to directly find the optimal policy network π_{θ^*} that maximize the value function V^π without estimating the value function.

Value-based versus Policy-based Approach

- `E1,p8` briefly introduced the two approaches.
  1. (Policy approach) Find *optimal policy*, π^* first.
     - (MDP) Policy iteration
     - (RL) Policy-based agent - policy gradient, REINFORCE, Actor-Critic.
  2. (Value approach) Find *optimal value function* $V^{\pi^*}() = V^*()$ first.
     - (MDP) Value iteration
     - (RL) Value-based agent - Q-Learning, Deep-Q Learning

	value function	policy function	variants
value-based	explicit	implicit	Q-learning, DQN
policy-based	implicit	explicit	policy-gradient, REINFORCE
Actor-Critic	explicit (critic)	explicit (actor)	Advantage actor-critic

Advantages of Policy-Based RL

Advantages

- Better convergence. (still debating, depending on problems)
- Effective in high-dimensional problem.
- Only way to deal with **continuous action space** and **random policies**.

Disadvantages

- Tends to converge to a local optimum rather than global optimum.
- Evaluating a policy is typically inefficient and has a high variance.

II. Mathematical Foundation of Policy Gradient

Objective function

- In value-based approaches, the objective function was obviously $max\ V^{\pi}(s)$.
- In policy-based approaches, what should be the objective function? Is it $max\ \pi(a|s)$?
- Remind that the objective of our beloved agent is to find a policy that maximizes the expected discounted sum of future rewards. We need to formularize some function of the parameter θ, where the function quantifies the expected value.
- Let the objective function be $J(\theta)$, which is maximized when $\theta = \theta^*$ where θ^* is associated with the optimal policy π_{θ^*}.
- Remind that the value function was $V(s) = V_t(s) = \mathbb{E}[G_t|S_t = s]$. How to connect $V(s)$ with $J(\theta)$?

Building up the objective function

- In order to deal with the value function's dependence on the initial state s, let us first assume the initial state is given as state s. Then, it follows

$$J(\theta) = \mathbb{E}_\theta[G_t | S_t = s],$$

  where θ again parameterizes the policy. The term in RHS reads as "expected return at the initial state s under the policy (*which is parameterized by*) θ.

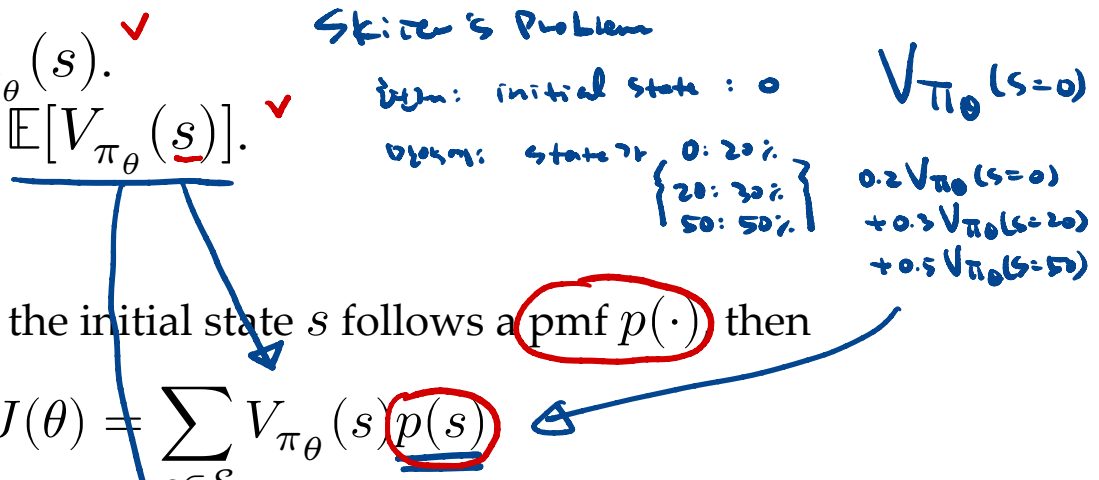
- Please find the following expansion comfortable as well.

$$J(\theta) = \mathbb{E}_\theta[G_t | S_t = s] = V_{\pi_\theta}(s)$$

  This is possible because the state-value function $V(\cdot)$ itself is an expected value.

- The two formula above restrictively assumed the initial state being s. What if s is uncertain and follows some distribution?

- For an initial state s, $J(\theta) = V_{\pi_\theta}(s)$.
- For an arbitrary state s, $J(\theta) = \mathbb{E}[V_{\pi_\theta}(s)]$.



- In other words,
  - If the state space is discrete and the initial state s follows a pmf $p(\cdot)$, then

$$J(\theta) = \sum_{s \in \mathcal{S}} V_{\pi_\theta}(s) p(s)$$

  - If the state space is continuous and the initial state s follows a pdf $f(\cdot)$, then

$$J(\theta) = \int_{s \in \mathcal{S}} V_{\pi_\theta}(s) f(s) \, ds$$

- Discussion on stationary distribution.

Improving the $J(\theta)$

- So far, where $V_{\pi_\theta}(s) = \mathbb{E}_\theta[G_t | S_t = s]$,
  - $J(\theta) = \mathbb{E}[V_{\pi_\theta}(s)]$ (obj fn)
  - If discrete, $J(\theta) = \sum_{s \in \mathcal{S}} V_{\pi_\theta}(s) p(s)$
  - If continuous, $J(\theta) = \int_{s \in \mathcal{S}} V_{\pi_\theta}(s) f(s)\, ds$

stationary dist.

- Our goal is clear now, first to start with some initial value $\theta = \theta_0$, then improve θ so that it maximizes $J(\theta)$. By the time the maximization is done, we will have found θ^*

$\pi_{\theta^*} \rightarrow \pi^*$

θ를 바꿔서 $J(\theta)$를 올리는 algo.

Gradient ascent

- Please review the gradient descent algorithm in G1, p19.
- Gradient descent in deep learning

$$\omega \leftarrow \omega - \alpha \cdot \nabla_{\omega} L$$

was the updating rule to decrease the loss function L.

- Gradient ascent in policy gradient

$$\theta \leftarrow \theta + \alpha \cdot \nabla_{\theta} J(\theta)$$

is the updating rule to increase the objective function $J(\theta)$.

- For implementation purpose, what remains is to calculate $\nabla_{\theta} J(\theta)$ from the neural net $\pi_{\theta}(s, a)$.

$\nabla_\theta J(\theta)$

- Assume discrete state space for simplicity. Then,

$$J(\theta) = \sum_{s \in \mathcal{S}} p(s) V_{\pi_\theta}(s) \quad \text{(from p.13)}$$

.

- Assume 1-step MDP for simplicity and for the moment. In other words, $G_0 = r_0 + r_1 + \cdots + r_\infty = r_0$. Then, it follows

$r_1 = \cdots = 0$

$$J(\theta) := \sum_{s \in \mathcal{S}} p(s) V_{\pi_\theta}(s)$$

π_θ에서의 state s의 value

$$= \sum_{s \in \mathcal{S}} p(s) \left(\sum_{a \in \mathcal{A}} \pi_\theta(s,a) R(s,a) \right)$$

$R(s,a) \to r_0$

$= \sum_{s \in \mathcal{S}} p(s) \mathbb{E}_{\pi_\theta}[G_t \mid S_t = s]$

$= \sum_{s \in \mathcal{S}} p(s) \mathbb{E}_{\pi_\theta}[r_t \mid S_t = s] = \sum_{s \in \mathcal{S}} p(s) \left(\pi_\theta(s,a_1) R(s,a_1) + \pi_\theta(s,a_2) R(s,a_2) + \cdots \right)$

- Taking gradient on both hand sides leads to

$$\nabla_\theta J(\theta) = \nabla_\theta \left(\sum_{s \in \mathcal{S}} p(s) \left(\sum_{a \in \mathcal{A}} \pi_\theta(s,a) R(s,a) \right) \right)$$

- There is only one term in RHS that is depends on θ. Therefore,

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \left(\sum_{s \in \mathcal{S}} p(s) \left(\sum_{a \in \mathcal{A}} \pi_\theta(s,a) \cdot R(s,a) \right) \right) \\
&= \sum_{s \in \mathcal{S}} p(s) \left(\sum_{a \in \mathcal{A}} \nabla_\theta \pi_\theta(s,a) \cdot R(s,a) \right) \\
&= \sum_{s \in \mathcal{S}} p(s) \left(\sum_{a \in \mathcal{A}} \frac{\pi_\theta(s,a)}{\pi_\theta(s,a)} \nabla_\theta \pi_\theta(s,a) \cdot R(s,a) \right) \\
&= \sum_{s \in \mathcal{S}} p(s) \left(\sum_{a \in \mathcal{A}} \pi_\theta(s,a) \frac{\nabla_\theta \pi_\theta(s,a)}{\pi_\theta(s,a)} \cdot R(s,a) \right) \\
&= \sum_{s \in \mathcal{S}} p(s) \left(\sum_{a \in \mathcal{A}} \pi_\theta(s,a) \nabla_\theta log \pi_\theta(s,a) \cdot R(s,a) \right) \\
&= \sum_{s \in \mathcal{S}} p(s) \sum_{a \in \mathcal{A}} \pi_\theta(s,a) [\nabla_\theta log \pi_\theta(s,a) \cdot R(s,a)] \\
&= \mathbb{E}_{\pi_\theta} [\nabla_\theta log \pi_\theta(s,a) \cdot R(s,a)]
\end{aligned}$$

$\frac{f'(x)}{f(x)} = (\log f(x))'$

1 period MDP / general case

$\nabla_\theta J(\theta) = \nabla_\theta [\sum_s p(s) (\sum_a \pi_\theta q_{\pi_\theta})]$

$= \sum_s p(s) \sum_a [(\nabla_\theta \pi_\theta) q_{\pi_\theta} + \pi_\theta \cdot (\nabla_\theta q_{\pi_\theta})]$

$(fg)' = f'g + fg'$

(with some luck)

$= \mathbb{E}_{\pi_\theta} [\nabla_\theta \log \pi_\theta \cdot q_{\pi_\theta}]$

???

Improvement strategy - 1-period setting

Summary so far

- $J(\theta) = \sum_{s\in\mathcal{S}} p(s) V_{\pi_\theta}(s) = \sum_{s\in\mathcal{S}} p(s) \left(\sum_{a\in\mathcal{A}} \pi_\theta(s,a) R(s,a) \right)$
- $\theta \leftarrow \theta + \alpha \cdot \nabla_\theta J(\theta)$
- $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta log \pi_\theta(s,a) \cdot R(s,a)]$

Improvement Process

- The RHS of the last equation is expected value under π_θ.

1. We will have our agent run on π_θ.
2. Collect $\nabla_\theta log \pi_\theta(s,a)$, which is nothing but the gradient of the neural net.
3. Observe $R(s,a)$ which is reward of action a from state s.
4. Taking an average to calculate $\nabla_\theta J(\theta)$.
5. Then, use the second formula to update θ.
6. Go back to Step 1 until it converges.

Revisit p15 and generalize it.

- In p15, we had $J(\theta) = \sum_{s\in\mathcal{S}} p(s) V_{\pi_\theta}(s) = \sum_{s\in\mathcal{S}} p(s) \left(\sum_{a\in\mathcal{A}} \pi_\theta(s,a) R(s,a)\right)$. Then, we obtained $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta log\pi_\theta(s,a) \cdot R(s,a)]$. This was based on the assumption of the 1-step MDP of $G_0 = r_0$.
- However, Q_{π_θ} must replace $R(s,a)$ from the beginning, in general cases.

- Now, our concern is, can we simply write $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta log\pi_\theta(s,a) \cdot Q_{\pi_\theta}(s,a)]$ just like $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta log\pi_\theta(s,a) \cdot R(s,a)]$? In other words, the derivative work in p16 is valid if we replace $R(s,a)$ with $Q_{\pi_\theta}(s,a)$?
- Fortunately, this is true and this is called *policy gradient theorem*. The proof can be found in Ch.13 Policy Gradient Methods in Sutton.

Proof for policy gradient theorem in Sutton (1/2)

정책 경사도 정리의 증명(에피소딕 문제의 경우)

간단한 계산 및 수식 정리를 통해, 기본 원칙으로부터 정책 경사도 정리를 증명할 수 있다. 표기법을 간단하게 하기 위해, π가 $\boldsymbol{\theta}$의 함수라는 것은 따로 표현하지 않을 것이다. 또한, 모든 경사도는 $\boldsymbol{\theta}$에 대해 계산된 경사도다. 먼저, 상태 가치 함수의 경사도는 행동 가치 함수를 이용하여 다음과 같이 표현할 수 있다.

모든 $s \in \mathcal{S}$에 대해

$$\begin{aligned}
\nabla v_\pi(s) &= \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right] && \text{(연습문제 3.18)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \Big] && \text{(곱의 법칙)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r|s,a)\big(r + v_\pi(s')\big) \Big] \\
& && \text{(연습문제 3.19와 식 3.2)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \nabla v_\pi(s') \Big] && \text{(식 3.4)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) && \text{(아래와 연결됨)} \\
&\qquad \sum_{a'} \big[\nabla\pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \big] \Big] \\
&= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla\pi(a|x) q_\pi(x,a)
\end{aligned}$$

Proof for policy gradient theorem in Sutton (2/2)

위 식은 식의 결과를 다시 그 식에 대입하는 과정을 무한히 반복하여 얻어진 것이다. 이 때 $\Pr(s \to x, k, \pi)$는 정책 π하에서 상태 s에서 상태 x로 k단계만에 전이할 확률을 나타낸다. 이제 다음과 같은 관계가 명확해진다.

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \nabla v_\pi(s_0) \\
&= \sum_s \left(\sum_{k=0}^{\infty} \Pr(s_0 \to s, k, \pi) \right) \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&= \sum_s \eta(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) && \text{(240쪽의 글상자)} \\
&= \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&= \sum_{s'} \eta(s') \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) && \text{(식 9.3)} \\
&\propto \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) && \text{(Q.E.D)}
\end{aligned}
$$

Summary again without the resticted assumption of 1-period MDP.

- $J(\theta) = \sum_{s\in\mathcal{S}} p(s) V_{\pi_\theta}(s) = \sum_{s\in\mathcal{S}} p(s) \left(\sum_{a\in\mathcal{A}} \pi_\theta(s,a) Q_{\pi_\theta}(s,a) \right)$
- $\theta \leftarrow \theta + \alpha \cdot \nabla_\theta J(\theta)$
- $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta log \pi_\theta(s,a) \cdot Q_{\pi_\theta}(s,a)]$ (policy gradient theorem)

Challenge in implementation for improvement.

- Our previous hope (p17) was⋯
  1. We will have our agent run on π_θ.
  2. Collect $\nabla_\theta log \pi_\theta(s,a)$, which is nothing but the gradient of the neural net.
  3. ~~Observe $R(s,a)$ which is reward of action a from state s.~~
  4. Taking an average to calculate $\nabla_\theta J(\theta)$.
  5. Then, use the second formula to update θ.
  6. Go back to Step 1 until it converges.
- In the generalized setting (without the restricted assumption of 1-period), the third step must involve $Q_{\pi_\theta}(s,a)$ instead of $R(s,a)$. But the implementation may not be so obvious. Unlike $R(s,a)$, $Q_{\pi_\theta}(s,a)$ is not a true value, so must be estimated. (∵ In our current policy-based approach, Q is not explicit. But π_θ is explicit)

Preview of upcoming algorithms

- $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta log \pi_\theta(s,a) \cdot Q_{\pi_\theta}(s,a)]$ (policy gradient theorem)
- From the above policy gradient algorithm, several policy-based approach algorithms are developed to work with $Q_{\pi_\theta}(s,a)$.
- The algorithm named as "REINFORCE" replaces it with its unbiased estimator, G_t. That is, it generates a full stochastic path to observe G_t. This is basically Monte-Carlo approach for estimation.
- The algorithm named as "Q Actor-Critic" proposes using another neural net, namely, $Q_\omega(s,a) \approx Q_{\pi_\theta}(s,a)$. This algorithm is characterized with using two neural nets, actor π_θ and critic Q_ω.
- The subsequent algorithm called as "Advantage actor-critic" pursues faster convergence by modifying the Q Actor-Critic.

III. Problem statement

Problem Statement (1/2)

- In each year, a sea may have a large or small number of belt fish. The state of the sea can be defined as High (H), Medium (M), and Low (L). In each state, the number of belt fish is 5000, 1000, and 100, respectively.
- There is a fish catching ship operating in the sea. If the ship catches many fish this year, then the number of belt fish in the next year will be small. Conversely, if the ship catches small number of fish this year, then the number of belt fish will be higher in the next year. In each year, the captain of the ship decides how much fishing will be done among the available belt fish. We let this action of fishing intensity as $a_t \in [0, 1]$. Of course, this action affects the state of sea in the next year.
- Specifically, if the current sea state is L, then the next sea state is L with probability a and M with probability $1 - a$. If the current sea state is M, then the next sea state is L with probability a and H with probability $1 - a$. If the current sea state is H, then the next sea state is M with probability a and H with probability $1 - a$.

Problem Statement (2/2)

- The ship operates perpetually, and aims to maximize the discounted sum of future rewards.
- Assume that the discount factor (γ) is 0.95. What is the optimal action when the sea state is H, M, and L, respectively.
- In each state, under the optimal policy, what is the expected value function for each state?
- Compare the optimal policy and its result against a benchmark policy of doing $a_t = 0.5$ for all state.

Problem Statement (1/2)

S, A, R, γ, P

$R(S, a) = \begin{cases} 5000\,a & \text{if } S = H \\ 1000\,a & \text{if } S = M \\ 100\,a & \text{if } S = L \end{cases}$

S
- 매년 바다에는 갈치가 많이 존재할 수도 있고, 적게 존재할 수도 있다. 바다의 상태는 High (H), Medium (M), Low (L)로 정의되며 각각의 상태에서 갈치는 각각 5000, 1000, 100마리 존재한다.

A
- 바다에서 갈치를 잡는 어선 한 척이 있는데, 갈치를 올해에 많이 어획한다면 내년에는 바다의 갈치의 수가 줄어들 것이고, 반대로 올해에 조금 어획한다면, 내년에는 바다의 갈치의 수가 늘어날 것이다. 어선의 선장은 매년 어획가능한 갈치의 몇 퍼센트 (이를 action $a_t \in [0, 1]$ 이라 하자)를 어획할 것인지를 결정한다. 그리고 이 결정은 내년의 갈치의 수에 영향을 준다.

P
- 구체적으로, 현재 바다의 상태가 L이라면 다음해 바다의 상태는 a의 확률로 L이 되고, $1 - a$의 확률로 M이 된다. 현재 바다 상태가 M이라면, 다음해 바다의 상태는 a의 확률로 L이 되고, $1 - a$의 확률로 H가 된다. 현재 바다 상태가 H라면, 다음해 바다 상태는 a의 확률로 M이 되고, $1 - a$의 확률로 H가 된다.

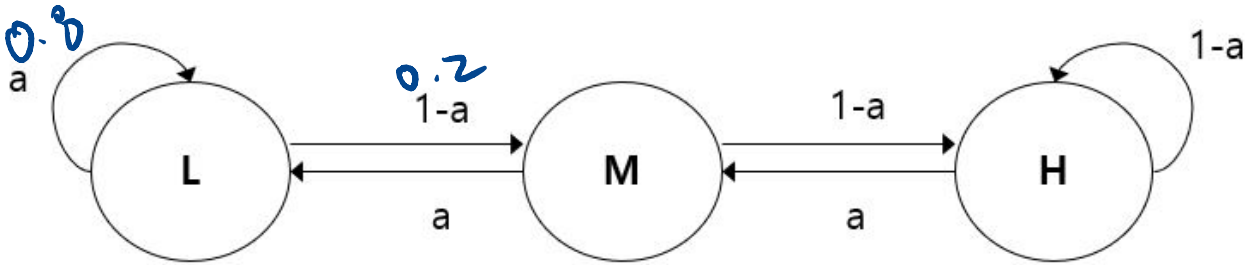
Problem Statement (2/2)

- 갈치 잡이 어선은 같은 바다에서 대대로 어획을 이어가며, 무한대의 시간에 걸쳐서 어획량을 최대화 하는 것을 목표로 한다.
- Discount factor (γ)를 0.95라고 가정하자. 바다의 상태가 각각 H, M, L일때에 갈치 잡이 어선은 각각 몇 퍼센트의 갈치를 어획해야 하는지 결정하라.
- 그리고 이때 각각의 state에서 기대되는 discounted sum of reward를 추정하라.
- 구해진 최적의 policy와 그 결과를 모든 state에서 $a_t = 0.5$를 행하는 경우와 비교하라.

MDP Formulation

- The tuple of $(\mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{A})$ need to be defined
- State-space
  - State S_t is defined as sea grade at time t, $\mathcal{S} = (L, M, H)$
- Action-space
  - Action A_t is defined as how much percentage of fishing is done at time t.
  - a_t is defined in continuous action space, i.e. $a_t \in [0, 1]$.
- Reward
  - Reward r_t depends on current state S_t as well as action A_t. In other words, reward r_t is a function of S_t and A_t
  - In light of this, the reward function $R(\cdot)$ is defined as a bivariate function, $R(s, a) = \mathbb{E}[r_t | S_t = s, A_t = a]$
  - Specifically,
    - $R(L, a) = 100a$
    - $R(M, a) = 1000a$
    - $R(H, a) = 5000a$

- Discount factor γ is assume to have 0.95.
- Transition probability
  - Transition function can be denoted as $S_{t+1} = f(S_t, A_t, \text{some randomness})$.
  - Specifically,
    - $\mathbb{P}(S_t = L | S_t = L) = a$
    - $\mathbb{P}(S_t = L | S_t = M) = 1 - a$
    - $\mathbb{P}(S_t = M | S_t = L) = a$
    - $\mathbb{P}(S_t = M | S_t = H) = 1 - a$
    - $\mathbb{P}(S_t = H | S_t = M) = a$
    - $\mathbb{P}(S_t = H | S_t = H) = 1 - a$
  - The transition diagram is given below.



"Man is gifted with reason; he is life being aware of itself; he has awareness of himself, of his fellow man, of his past, and of the possibilities of his future. - Erich Fromm"