Lecture 8: Actor-Critic Algorithms

Zhi Wang & Chunlin Chen

Department of Control and Systems Engineering
Nanjing University

Nov. 22nd, 2021

Table of Contents

Today's lecture

- Improving the policy gradient with a critic
- The policy evaluation problem
- Discount factors
- The actor-critic algorithm
- Goals
  - Understand how policy evaluation fits into policy gradients
  - Understand how actor-critic algorithms work

Review: policy gradients

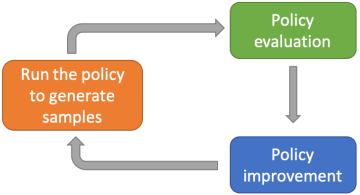
REINFORCE algorithm: Loop:

1. sample $\{\tau^i\}$ from $\pi_\theta(a_t|s_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\right)\left(\sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}^i, a_{t'}^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

"reward-to-go":

$$\hat{Q}_{t,i}^{\pi} = \hat{Q}^{\pi}(s_t^i, a_t^i) = \sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}^i, a_{t'}^i)$$

$$\hat{Q}^{\pi}(s_t, a_t) = \sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}, a_{t'})$$

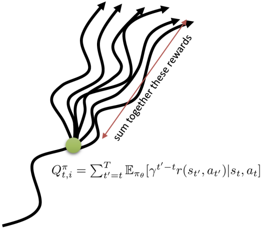


$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

Improving the policy gradient

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\underbrace{\left(\sum_{t'=t}^{T}\gamma^{t'-t}r(s_{t'}^i,a_{t'}^i)\right)}_{\hat{Q}^{\pi}_{t,i}:\ \text{reward-to-go}}$$

- $\hat{Q}^{\pi}_{t,i}$: estimate of expected reward if we take action a_t^i in state s_t^i
- **Question**: can we get a better estimate?
- $Q^\pi(s_t,a_t) = \sum_{t'=t}^{T}\mathbb{E}_{\pi_\theta}[\gamma^{t'-t}r(s_{t'},a_{t'})|s_t,a_t]$: true *expected* reward-to-go



$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)Q^\pi(s_t^i,a_t^i)$$

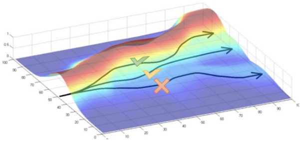
Review: Reducing variance - Baselines

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau)[r(\tau) - b]$$

a convenient identity

$$\pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) = \nabla_\theta \pi_\theta(\tau)$$

$$b = \frac{1}{N}\sum_{i=1}^{N} r(\tau)$$



- But... are we allowed to do that?

$$\mathbb{E}[\nabla_\theta \log \pi_\theta(\tau) b] = \int \pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) b \, \mathrm{d}\tau = \int \nabla_\theta \pi_\theta(\tau) b \, \mathrm{d}\tau$$

$$= b\nabla_\theta \int \pi_\theta(\tau)\, \mathrm{d}\tau = b\nabla_\theta 1 = 0$$

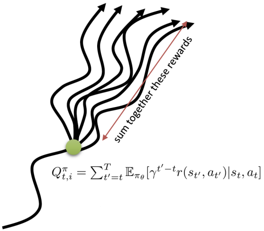
- Subtracting a baseline is unbiased in expectation!
- Average reward is not the best baseline, but it's pretty good!

What about the baseline?

- $Q^\pi(s_t, a_t) = \sum_{t'=t}^{T} \mathbb{E}_{\pi_\theta}[\gamma^{t'-t} r(s_{t'}, a_{t'}) | s_t, a_t]$:
  true *expected* reward-to-go
- Let's try to use the average reward as the baseline:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \left[Q^\pi(s_t^i, a_t^i) - b \right]$$

$$\underbrace{b = \frac{1}{N} \sum_{i=1}^{N} Q^\pi(s_t^i, a_t^i) \approx \mathbb{E}_{a_t \sim \pi_\theta(a_t | s_t)} \left[Q^\pi(s_t^i, a_t^i) \right]}_{\text{What is this?}}$$



Review: Relationship between Q and V

- State value function:

$$V^\pi(s) = \mathbb{E}_\pi\left[\sum_{k=1}^{\infty}\gamma^k R_{t+k+1}|S_t = s\right]$$

- Action value function:

$$Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{k=1}^{\infty}\gamma^k R_{t+k+1}|S_t = s, A_t = a\right]$$

- **What is the relationship between $V^\pi(s)$ and $Q^\pi(s,a)$?**

Review: Relationship between Q and V

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{k=1}^{\infty}\gamma^k R_{t+k+1}|S_t = s\right]$$

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{k=1}^{\infty}\gamma^k R_{t+k+1}|S_t = s, A_t = a\right]$$

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{\pi}\left[\sum_{k=1}^{\infty}\gamma^k R_{t+k+1}|S_t = s\right] \\ &= \sum_{a}\pi(a|s)\mathbb{E}_{\pi}\left[\sum_{k=1}^{\infty}\gamma^k R_{t+k+1}|S_t = s, A_t = a\right] \\ &= \sum_{a}\pi(a|s)Q^{\pi}(s,a) = \mathbb{E}_{a\sim\pi}[Q^{\pi}(s,a)] \end{aligned}$$

Review: State- & action- value function

- Action value function $Q^\pi(s,a)$: total reward from taking a in s

$$
\begin{aligned}
Q^\pi(s,a) &= \mathbb{E}_\pi[R_{t+1} + \gamma V^\pi(S_{t+1}) | S_t = s, A_t = a] \\
&= \sum_{s',r} p(s',r|s,a)[r + \gamma V^\pi(s')]
\end{aligned}
$$

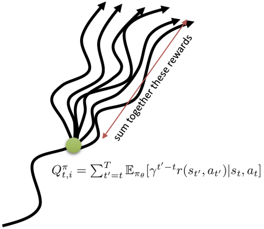
- State value function $V^\pi(s)$: total reward from s

$$
V^\pi(s) = \mathbb{E}_{a\sim\pi(a|s)}[Q^\pi(s,a)]
$$

The state value function is the baseline!

$$b = \frac{1}{N}\sum_{i=1}^{N} Q^{\pi}(s_t^i, a_t^i) \approx \mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[Q^{\pi}(s_t^i, a_t^i)]$$

$$V^{\pi}(s_t) = \mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[Q^{\pi}(s_t^i, a_t^i)]$$



$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i) \underbrace{\left[Q^{\pi}(s_t^i, a_t^i) - V^{\pi}(s_t^i)\right]}_{\text{What is this?}}$$

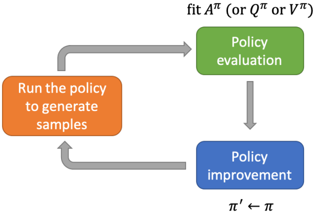
The "advantage" function

- $Q^\pi(s_t, a_t) = \sum_{t'=t}^{T} \mathbb{E}_{\pi_\theta}[\gamma^{t'-t} r(s_{t'}, a_{t'}) | s_t, a_t]$:
  - total reward from taking a_t in s_t following policy π
- $V^\pi(s_t) = \mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)]$:
  - total reward from s_t following policy π
- $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$:
  - the advantage of a_t: how much better a_t is

$$
\begin{aligned}
\nabla_\theta J(\theta) &\approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \left[Q^\pi(s_t^i, a_t^i) - V^\pi(s_t^i, a_t^i) \right] \\
&= \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) A^\pi(s_t^i, a_t^i)
\end{aligned}
$$

The "advantage" function

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)A^\pi(s_t^i, a_t^i)$$



- the better this estimate, the lower the variance

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\left(\sum_{t'=t}^{T}\gamma^{t'-t}r(s_{t'}^i, a_{t'}^i) - b\right)$$

- unbiased, but high variance single-sample estimate

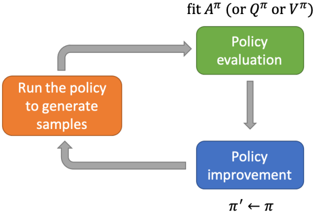
Value function fitting

$$Q^\pi(s_t, a_t) = \sum_{t'=t}^{T} \mathbb{E}_{\pi_\theta}\left[\gamma^{t'-t} r(s_{t'}, a_{t'}) | s_t, a_t\right]$$

$$V^\pi(s_t) = \mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[Q^\pi(s_t, a_t)]$$

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$$

Fit what to what? Q^π, V^π, or A^π?



- In dynamic programming: $Q^\pi(s, a) = \sum_{s', r} p(s', r|s, a)[r + \gamma V^\pi(s')]$
- Act in a model-free way: $Q^\pi(s_t, a_t) \approx r(s_t, a_t) + \gamma V^\pi(s_{t+1})$
  - Forget about the model $p(s', r|s, a)$
- $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) \approx \underbrace{r(s_t, a_t) + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)}_{\text{TD error}}$
- **Let's just fit $V^\pi(s)$!**

Review: For large/continuous state/action spaces

- **Curse of dimensionality**: Computational requirements grow exponentially with the number of state variables
  - Theoretically, all state-action pairs need to be visited infinite times to guarantee an optimal policy
  - In many practical tasks, almost every state encountered will never have been seen before
- **Generalization**: How can experience with a limited subset of the state space be usefully generalized to produce a good **approximation** over a much larger subset?

Review: Curse of dimensionality

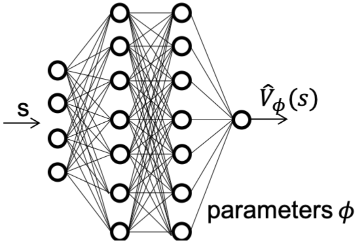
0.5	0.8	0.3	0.4
0.4	0.3	0.8	0.5
0.7	0.6	0.6	0.7
0.9	0.5	0.1	0.2

- In discrete case, represent $V(s)$ as a table
  - 16 states, 4 actions per state
  - can store full $V(s)$ in a table
  - iterative sweeping over the state space

- An image
  - $|\mathcal{S}| = (255^3)^{200\times 200}$
  - more than atoms in the universe
  - can we store such a large table?

Review: Function approximation

- It takes examples from a desired function (e.g., a value function) and attempts to generalize from them to construct an approximation to the entire function
  - Linear function approximation: $V(s) = \sum_i \phi_i(s) w_i$
  - Neural network approximation: $V(s) = V_\phi(s)$



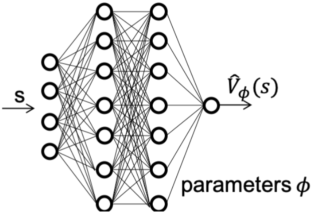
Review: Function approximation

- Function approximation is an instance of **supervised learning**, the primary topic studied in machine learning, artificial neural networks, pattern recognition, and statistical curve fitting
  - In theory, any of the methods studied in these fields can be used in the role of function approximator within RL algorithms
  - RL with function approximation involves a number of **new issues** that do not normally arise in conventional supervised learning, e.g., non-stationarity, bootstrapping, and delayed targets

Value function fitting

$$A^\pi(s_t, a_t) \approx r(s_t, a_t) + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)$$

$$\hat{A}^\pi(s_t, a_t) \approx r(s_t, a_t) + \gamma \hat{V}^\pi_\phi(s_{t+1}) - \hat{V}^\pi_\phi(s_t)$$



Modified REINFORCE algorithm: Loop:

1. sample $\{\tau^i\}$ from $\pi_\theta(a_t|s_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\hat{A}^\pi(s_t^i, a_t^i)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

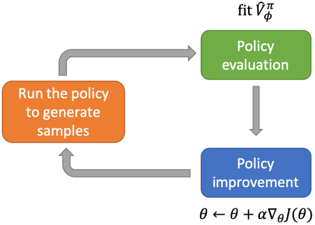


Table of Contents

Review: Policy evaluation in dynamic programming

- Compute the state-value function V^π for an arbitrary policy π

$$
\begin{aligned}
V^\pi(s) &= \mathbb{E}_\pi[G_t|S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma \mathbb{E}_\pi[G_{t+1}|S_{t+1} = s']] \\
&= \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a)[r + \gamma V^\pi(s')]
\end{aligned}
$$

- If the environment's dynamics are completely known
  - In principal, the solution is a straightforward computation

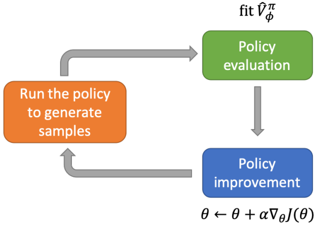
Review: Policy evaluation in Monte Carlo

- Considering Monte Carlo methods for learning the state-value function for a given policy
  - $V^{\pi}(s)$: the expected return–expected cumulative future discounted reward–starting from s
  - Estimate $V^{\pi}(s)$ from **experience**: simply average the returns observed after visits to s
  - As more returns are observed, the average should converge to the expected value

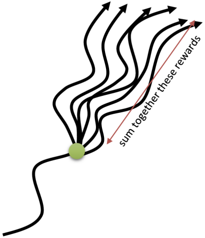
$$
\begin{aligned}
V^{\pi}(s) &= \mathbb{E}_{\pi}[G_t | S_t = s] \\
&= \mathbb{E}_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... | S_t = s]
\end{aligned}
$$

Monte-Carlo evaluation with function approximation

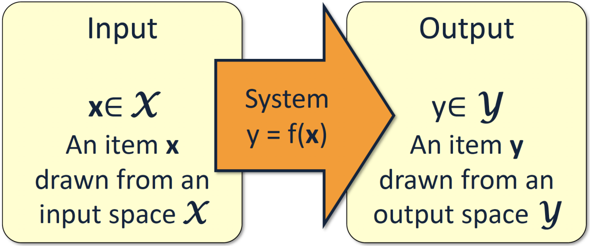
- $V^{\pi}(s_t) = \sum_{t'=t}^{T} \mathbb{E}_{\pi_\theta}\left[\gamma^{t'-t} r(s_{t'}, , a_{t'}) | s_t\right]$
- $J(\theta) = \mathbb{E}_{s_0 \sim p(s_0)}[V^{\pi}(s_0)]$
- **Question**: how can we perform policy evaluation?



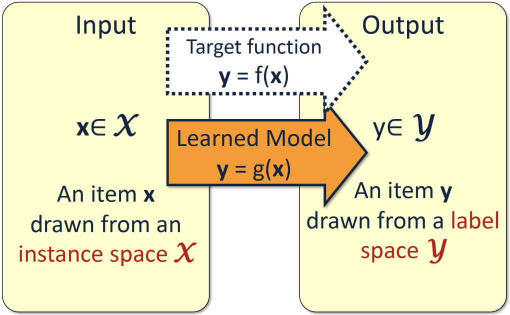
- Monte Carlo policy evaluation
  - this is what policy gradient does
  - requires to reset the simulator
- $V^{\pi}(s_t) \approx \sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}, , a_{t'})$
- $V^{\pi}(s_t) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}, , a_{t'})$



Review: Regression in supervised learning



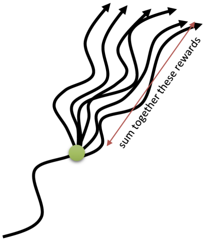
- We consider systems that apply a function $f(\cdot)$ to input items $\boldsymbol{x}$ and return an output $\boldsymbol{y} = f(\boldsymbol{x})$
- In supervised learning, we deal with systems whose $f(\cdot)$ is learned from samples $(\boldsymbol{x}, \boldsymbol{y})$



- We need to choose what kind of model we want to learn
  - Linear model, nonlinear model...
  - Parametric model, nonparametric model...
  - Decision trees, neural networks, Gaussian processes...

Monte-Carlo evaluation using supervised regression

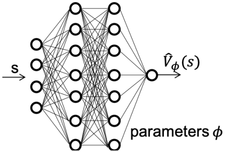
- $V^\pi(s_t) \approx \sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'},, a_{t'})$
  - not as good as this:
    $V^\pi(s_t) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'},, a_{t'})$
  - but still pretty good!



- training data: $\left(s_t^i, \underbrace{\sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}^i, a_{t'}^i)}_{\text{label: } y_t^i}\right)$

- supervised regression:

$$\mathcal{L}(\phi) = \frac{1}{2} \sum_i \sum_t ||\hat{V}_\phi^\pi(s_t^i) - y_t^i||^2$$



Review: Policy evaluation in temporal-difference learning

- MC and TD in common
  - Use experience to solve the prediction problem, update their estimate of V^π for the non-terminal state S_t occurring in that experience

- MC: **must wait** until the return following the visit is known (end of an episode)

$$
\begin{aligned}
V^\pi(s) &= \mathbb{E}_\pi[G_t|S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...|S_t = s]
\end{aligned}
$$

- TD: need to wait only until the next time step, bootstrapping

$$
\begin{aligned}
V^\pi(s) &= \mathbb{E}_\pi[G_t|S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma V^\pi(S_{t+1})|S_t = s]
\end{aligned}
$$

Can we do better? – From MC to TD evaluation

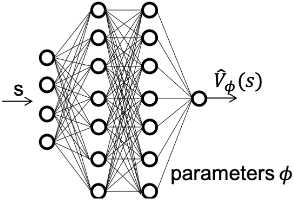
$$
\begin{aligned}
V^{\pi}(s) &= \mathbb{E}_{\pi}[G_t|S_t = s] \\
&= \mathbb{E}_{\pi}[R_{t+1} + \gamma G_{t+1}|S_t = s] \\
&= \mathbb{E}_{\pi}[R_{t+1} + \gamma V^{\pi}(S_{t+1})|S_t = s] \\
&= \sum_{a} \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V^{\pi}(s')]
\end{aligned}
$$

- MC: The expected G_t is not known, a sample return is used in place of the real expected return
- DP: The true V^{π} is not known, and the current estimate $V(S_{t+1})$ is used instead
- TD: It samples the expected values R_{t+1}, and it uses the current estimate $V(S_{t+1})$ instead of the true V^{π}
  - Combine the sampling of MC with the bootstrapping of DP

TD policy evaluation with function approximation

- Monte Carlo target: $y_t^i = \sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}^i, a_{t'}^i)$
- TD target for $V^\pi(s_t^i)$:

$$
\begin{aligned}
y_t^i &= \sum_{t'=t}^{T} \mathbb{E}_{\pi_\theta}\left[\gamma^{t'-t} r(s_{t'}^i, a_{t'}^i) | s_t^i\right] \\
&\approx r(s_t^i, a_t^i) + \gamma V^\pi(s_{t+1}^i) \\
&\approx r(s_t^i, a_t^i) + \gamma \hat{V}_\phi^\pi(s_{t+1}^i)
\end{aligned}
$$



  - Directly use previous fitted value function!
  - the "bootstrapped" estimate

- training data:

$$(s_t^i, \underbrace{r(s_t^i, a_t^i) + \gamma \hat{V}_\phi^\pi(s_{t+1}^i)}_{\text{label: } y_t^i})$$

- supervised regression:

$$\mathcal{L}(\phi) = \frac{1}{2} \sum_i \sum_t ||\hat{V}_\phi^\pi(s_t^i) - y_t^i||^2$$

Policy evaluation examples

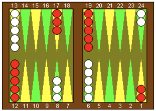


Figure 2. An illustration of the normal opening position in backgammon. TD-Gammon has sparked a near-universal conversion in the way experts play certain opening rolls. For example, with an opening roll of 4-1, most players have now switched from the traditional move of 13-9, 6-5, to TD-Gammon's preference, 13-9, 24-23. TD-Gammon's analysis is given in Table 2.

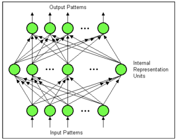
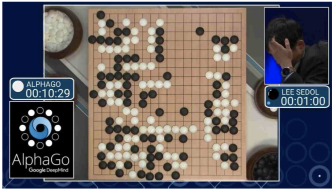


Figure 1. An illustration of the multilayer perception architecture used in TD-Gammon's neural network. This architecture is also used in the popular backpropagation learning procedure. Figure reproduced from [9].



- TD-Gammon, Gerald Tesauro 1992
  - reward: game outcome
  - value function $\hat{V}_\phi^\pi(s_t)$: expected outcome given board state

- AlphaGo, Silver et al. 2016
  - reward: game outcome
  - value function $\hat{V}_\phi^\pi(s_t)$: expected outcome given board state

Table of Contents

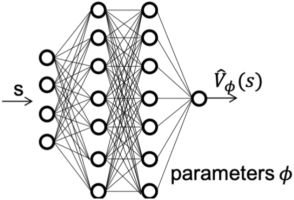
An actor-critic algorithm

Batch actor-critic algorithm. Loop:

1. sample $\{(s_i, a_i, r_i, s'_i)\}$ from $\pi_\theta(a|s)$
2. **policy evaluation**: fit $\hat{V}^\pi_\phi(s)$ to samples using supervised regression
3. evaluate $\hat{A}^\pi(s_i, a_i) = r_i + \gamma \hat{V}^\pi_\phi(s'_i) - \hat{V}^\pi_\phi(s_i)$
4. **policy improvement**: $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(a_i|s_i) \hat{A}^\pi(s_i, a_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

training data: $(s^i_t, \underbrace{r(s^i_t, a^i_t) + \gamma \hat{V}^\pi_\phi(s^i_{t+1})}_{\text{label: } y^i_t})$

$$\mathcal{L}(\phi) = \frac{1}{2} \sum_i \sum_t ||\hat{V}^\pi_\phi(s^i_t) - y^i_t||^2$$



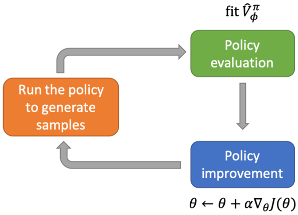
An actor-critic algorithm

Batch actor-critic algorithm. Loop:

1. sample $\{(s_i, a_i, r_i, s'_i)\}$ from $\pi_\theta(a|s)$
2. **policy evaluation**: fit $\hat{V}^\pi_\phi(s)$ to samples using supervised regression
3. evaluate $\hat{A}^\pi(s_i, a_i) = r_i + \gamma \hat{V}^\pi_\phi(s'_i) - \hat{V}^\pi_\phi(s_i)$
4. **policy improvement**: $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(a_i|s_i) \hat{A}^\pi(s_i, a_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

training data: $\left(s^i_t, \underbrace{r(s^i_t, a^i_t) + \gamma \hat{V}^\pi_\phi(s^i_{t+1})}_{\text{label: } y^i_t}\right)$

$$\mathcal{L}(\phi) = \frac{1}{2} \sum_i \sum_t ||\hat{V}^\pi_\phi(s^i_t) - y^i_t||^2$$



Review: Discount rate $\gamma \in [0, 1]$

- Assume that: $0 \leq r_{min} \leq r \leq r_{max} \leq \infty$
- Without discount factor: unbounded

$$\begin{aligned} V(s_t) &= \mathbb{E}[r_t + r_{t+1} + r_{t+2} + ...] \\ &\geq r_{min} + r_{min} + r_{min} + ... \\ &= \infty \end{aligned}$$

- With discount factor: bounded

$$\begin{aligned} V(s_t) &= \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...] \\ &\leq r_{max} + \gamma r_{max} + \gamma^2 r_{max} + ... \\ &= \frac{r_{max}}{1-\gamma} \end{aligned}$$

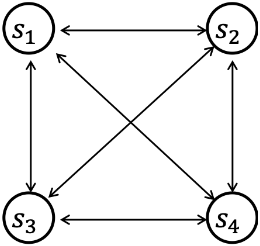
Review: Discount rate $\gamma \in [0, 1]$

- The expected **discounted** return
  - $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} ... = \sum_{k=1}^{\infty} \gamma^k R_{t+k+1}$
- The discount rate determines the present value of future rewards: a reward received k time steps in the future is worth only γ^{k-1} times what it would be worth if it were received immediately
- $\gamma \to 0$, the agent is "myopic", only maximizing immediate rewards
  - Akin to supervised learning that maximizes the log-likelihood of each sample, $\log p(y_i|x_i)$
- $\gamma \to 1$, the agent is "farsighted", taking future rewards into account
- Returns at successive time steps are related to each other

$$\begin{aligned} G_t &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + ...) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

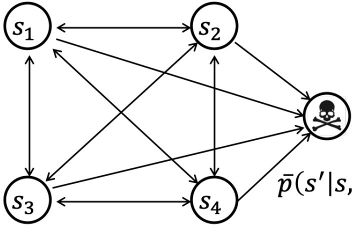
Review: γ changes the MDP

Without discount:



$p(s'|s,a)$

With discount:



$\bar{p}(s'|s,a) = 1 - \gamma$

$\bar{p}(s'|s,a) = \gamma p(s'|s,a)$

Batch actor-critic algorithm. Loop:

1. sample $\{(s_i, a_i, r_i, s_i')\}$ from $\pi_\theta(a|s)$
2. **policy evaluation**: fit $\hat{V}_\phi^\pi(s)$ to samples using supervised regression
3. evaluate $\hat{A}^\pi(s_i, a_i) = r_i + \gamma \hat{V}_\phi^\pi(s_i') - \hat{V}_\phi^\pi(s_i)$
4. **policy improvement**: $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(a_i|s_i) \hat{A}^\pi(s_i, a_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Online actor-critic algorithm. Loop:

1. take action $a \sim \pi_\theta(a|s)$, get (s_i, a_i, r_i, s_i')
2. **policy evaluation**: update $\hat{V}_\phi^\pi$ using target $r_i + \gamma \hat{V}_\phi^\pi(s_i')$
3. evaluate $\hat{A}^\pi(s_i, a_i) = r_i + \gamma \hat{V}_\phi^\pi(s_i') - \hat{V}_\phi^\pi(s_i)$
4. **policy improvement**: $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a_i|s_i) \hat{A}^\pi(s_i, a_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

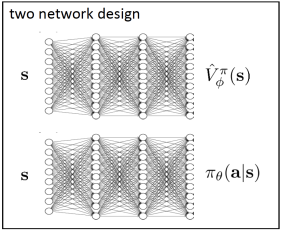
Batch-mode (offline) vs. online

- Batch-model (offline) algorithms
  - Collect a batch of samples using some policy
  - Fit the state- or action-value function iteratively
- Online algorithms
  - Take some action to collect one sample
  - Fit the value function
  - Iteratively alternate the above two steps

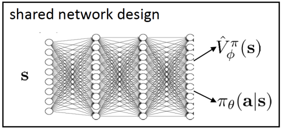
Architecture design

Online actor-critic algorithm. Loop:

1. take action $a \sim \pi_\theta(a|s)$, get (s_i, a_i, r_i, s_i')
2. **policy evaluation**: update $\hat{V}_\phi^\pi$ using target $r_i + \gamma \hat{V}_\phi^\pi(s_i')$
3. evaluate $\hat{A}^\pi(s_i, a_i) = r_i + \gamma \hat{V}_\phi^\pi(s_i') - \hat{V}_\phi^\pi(s_i)$
4. **policy improvement**: $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a_i|s_i) \hat{A}^\pi(s_i, a_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$



+ simple & stable
- no shared features between actor & critic



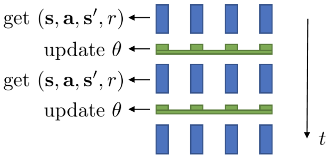
Parallelization

Online actor-critic algorithm. Loop:

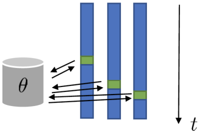
1. take action $a \sim \pi_\theta(a|s)$, get (s_i, a_i, r_i, s'_i)
2. policy evaluation: update $\hat{V}^\pi_\phi$ using target $r_i + \gamma \hat{V}^\pi_\phi(s'_i)$
3. evaluate $\hat{A}^\pi(s_i, a_i) = r_i + \gamma \hat{V}^\pi_\phi(s'_i) - \hat{V}^\pi_\phi(s_i)$
4. policy improvement: $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(a_i|s_i) \hat{A}^\pi(s_i, a_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

works best with a batch (e.g., parallel workers)

synchronized parallel actor-critic



asynchronous parallel actor-critic



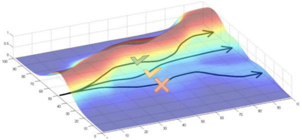
Review: Reducing variance - Baselines

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau)[r(\tau) - b]$$

a convenient identity

$$\pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) = \nabla_\theta \pi_\theta(\tau)$$

$$b = \frac{1}{N}\sum_{i=1}^{N} r(\tau)$$



- But... are we allowed to do that?

$$\mathbb{E}[\nabla_\theta \log \pi_\theta(\tau) b] = \int \pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) b \,\mathrm{d}\tau = \int \nabla_\theta \pi_\theta(\tau) b \,\mathrm{d}\tau$$
$$= b\nabla_\theta \int \pi_\theta(\tau)\,\mathrm{d}\tau = b\nabla_\theta 1 = 0$$

- Subtracting a baseline is unbiased in expectation!
- Average reward is not the best baseline, but it's pretty good!

Review: Analyzing the variance

$$var = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[g(\tau)^2(r(\tau)-b)^2] - \underbrace{\mathbb{E}_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau)(r(\tau)-b)]^2}_{\mathbb{E}_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)]^2}$$

(baselines are unbiased in expectation)

$$\begin{aligned}
\frac{\mathrm{d}var}{\mathrm{d}b} &= \frac{\mathrm{d}}{\mathrm{d}b}\mathbb{E}[g(\tau)^2(r(\tau)-b)^2] \\
&= \frac{\mathrm{d}}{\mathrm{d}b}(\mathbb{E}[g(\tau)^2 r(\tau)^2] \underbrace{-2\mathbb{E}[g(\tau)^2 r(\tau) b] + b^2\mathbb{E}[g(\tau)^2]}_{\text{dependent of } b}) \\
&= -2\mathbb{E}[g(\tau)^2 r(\tau)] + 2b\mathbb{E}[g(\tau)^2] = 0
\end{aligned}$$

$$b^* = \frac{\mathbb{E}[g(\tau)^2 r(\tau)]}{\mathbb{E}[g(\tau)^2]}$$

This is just expected reward, but weighted by gradient magnitudes!

Critics as state-dependent baselines

Actor-critic: $\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\left(r(s_t^i,a_t^i)+\gamma\hat{V}_\phi^\pi(s_{t+1}^i)-\hat{V}_\phi^\pi(s_t^i)\right)$

- $+$ lower variance (due to critic)
- - not unbiased (if the critic is not perfect)

Policy gradient: $\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\left(\left(\sum_{t'=t}^{T}\gamma^{t'-t}r(s_{t'}^i,a_{t'}^i)\right)-b\right)$

- $+$ no bias
- - higher variance (because single-sample estimate)

Can we use $\hat{V}_\phi^\pi$ and still keep the estimator unbiased?

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\left(\left(\sum_{t'=t}^{T}\gamma^{t'-t}r(s_{t'}^i,a_{t'}^i)\right)-\hat{V}_\phi^\pi(s_t^i)\right)$$

- $+$ no bias
- $+$ lower variance (baseline is closer to the return)

Table of Contents

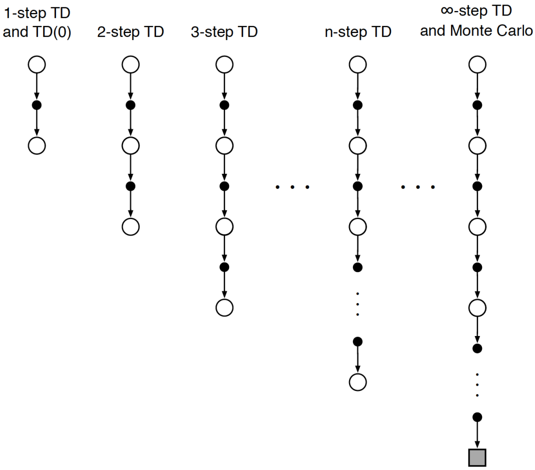
n-step bootstrapping: Combine MC and one-step TD

- Neither MC or one-step TD is always the best, we generalize both methods so that one can shift from one to the other smoothly as needed to meet the demands of a particular task
- One-step TD: In many applications, one wants to be able to update the action very fast to take into account anything that has changed
- However, bootstrapping works best if it is over a length of time in which a significant and recognizable state change has occurred

$n = 1$	n-step TD	$n = \infty$
TD(0)	$\leftrightarrow$	MC

n-step TD prediction

- Perform an update based on an intermediate number of rewards, more than one, but less than all of them until termination



Review: MC and TD(0) updates

- In MC updates, the target is the **complete return**

$$G_t = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{T-t+1} R_T$$

$$\begin{aligned} V(S_t) &\leftarrow V(S_t) + \alpha[G_t - V(S_t)] \\ &= V(S_t) + \alpha[R_{t+1} + \gamma R_{t+2} + ... + \gamma^{T-t+1} R_T - V(S_t)] \end{aligned}$$

- In TD(0) updates, the target is the **one-step return**

$$G_{t:t+1} = R_{t+1} + \gamma V(S_{t+1})$$

$$\begin{aligned} V(S_t) &\leftarrow V(S_t) + \alpha[G_{t:t+1} - V(S_t)] \\ &= V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \end{aligned}$$

n-step TD update rule

- For n-step TD, set the target as the n-**step return**

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

- All n-step returns can be considered approximations to the complete return, truncated after n steps and then corrected for the remaining missing terms by $V(S_{t+n})$

$$\begin{aligned} V(S_t) &\leftarrow V(S_t) + \alpha[G_{t:t+n} - V(S_t)] \\ &= V(S_t) + \alpha[R_{t+1} + \gamma R_{t+2} + ... + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n}) - V(S_t)] \end{aligned}$$

Actor-critics with n-step returns

- TD(0): $\hat{A}^\pi(s_t, a_t) = \boxed{r(s_t, a_t) + \gamma \hat{V}^\pi_\phi(s_{t+1})} - \hat{V}^\pi_\phi(s_t)$
  - + lower variance
  - \- higher bias if value if wrong (it always is)
- Monte Carlo: $\hat{A}^\pi_{\text{MC}}(s_t, a_t) = \boxed{\sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}, a_{t'})} - \hat{V}^\pi_\phi(s_t)$
  - + no bias
  - \- higher variance (because single-sample estimate)

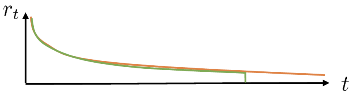
- **Question**: Can we combine these two, to control bias/variance trade-off?

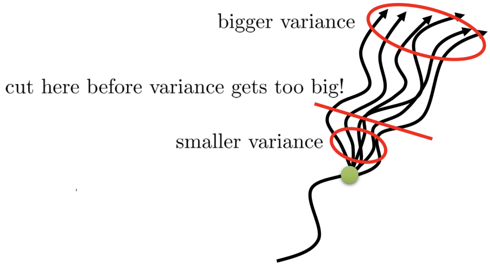
Actor-critics with n-step returns

- TD(0): $\hat{A}^{\pi}(s_t, a_t) = \boxed{r(s_t, a_t) + \gamma \hat{V}^{\pi}_{\phi}(s_{t+1})} - \hat{V}^{\pi}_{\phi}(s_t)$
  - $+$ lower variance
  - \- higher bias if value if wrong (it always is)
- Monte Carlo: $\hat{A}^{\pi}_{\text{MC}}(s_t, a_t) = \boxed{\sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}, a_{t'})} - \hat{V}^{\pi}_{\phi}(s_t)$
  - $+$ no bias
  - \- higher variance (because single-sample estimate)
- n-step TD: $\hat{A}^{\pi}_{n}(s_t, a_t) = \boxed{\sum_{t'=t}^{t+n} r(s_{t'}, a_{t'}) + \gamma^{n} \hat{V}^{\pi}_{\phi}(s_{t+n})} - \hat{V}^{\pi}_{\phi}(s_t)$
  - choosing $n > 1$ often works better!

Actor-critics with n-step returns

- n-step TD: $\hat{A}^\pi(s_t, a_t) = \boxed{\sum_{t'=t}^{t+n} r(s_{t'}, a_{t'}) + \gamma^n \hat{V}^\pi_\phi(s_{t+n})} - \hat{V}^\pi_\phi(s_t)$
  - choosing $n > 1$ often works better!





Eligibility traces: unify/generalize TD and MC

- Almost any TD method can be combined with eligibility traces to obtain a more general method that may learn more efficiently
  - e.g., the popular TD(λ) algorithm, λ refers the use of an eligibility trace
  - Produce a family of methods spanning a spectrum that has MC methods at one end ($\lambda = 1$) and one-step TD methods at the other ($\lambda = 0$)
- Eligibility traces offer an elegant algorithmic mechanism with significant computational advantages (compared to n-step TD)
  - Only a single trace vector is required rather than a store of the last n feature vectors
  - Learning also occurs continually and uniformly in time rather than being delayed and then catching up at the end of the episode
  - Learning can occur and effect behavior immediately after a state is encountered rather than being delayed n-steps

The λ-return

- How to interrelate TD and MC?
  - e.g., average one-step and infinite-step returns, $G = (G_t + G_{t:t+1})/2$
  - An update that averages simpler component updates is called a **compound update**
- The TD(λ) algorithm can be understood as one particular way of averaging n-step updates

$$
\begin{aligned}
G_t^\lambda &= (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_{t:t+n} \\
&= (1-\lambda)\sum_{n=1}^{T-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t-1} G_t
\end{aligned}
$$

Backup diagram for TD(λ)

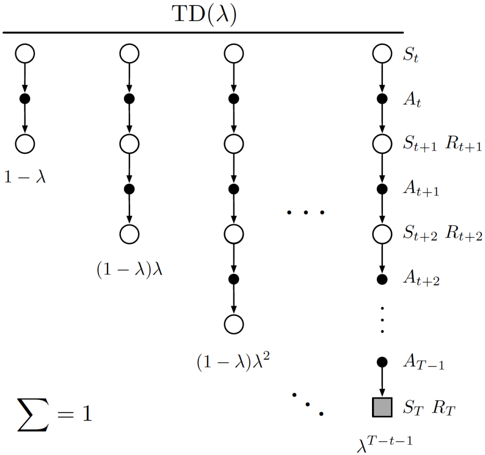


Figure 12.1: The backup digram for TD(λ). If $\lambda = 0$, then the overall update reduces to its first component, the one-step TD update, whereas if $\lambda = 1$, then the overall update reduces to its last component, the Monte Carlo update.

The weight distribution

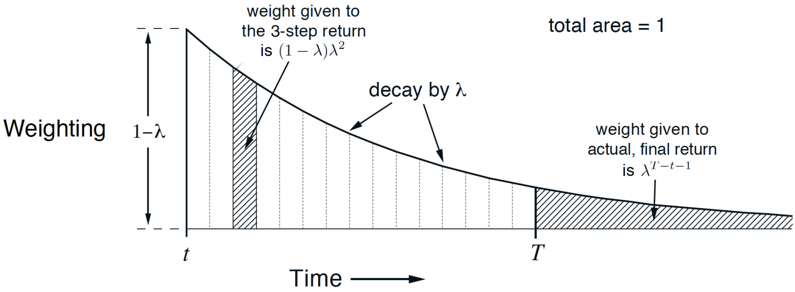


Figure 12.2: Weighting given in the λ-return to each of the n-step returns.

- n-step TD: $\hat{A}^{\pi}_{n}(s_t, a_t) = \sum_{t'=t}^{t+n} r(s_{t'}, a_{t'}) + \gamma^n \hat{V}^{\pi}_{\phi}(s_{t+n}) - \hat{V}^{\pi}_{\phi}(s_t)$
- Weighted combination of all n-step returns: $w_n \propto \lambda^{n-1}$

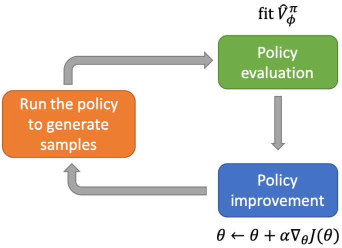
  $\hat{A}^{\pi}_{\mathsf{GAE}}(s_t, a_t) = \sum_{n=1}^{T} w_n \hat{A}^{\pi}_{n}(s_t, a_t)$

$$\hat{A}^{\pi}_{\mathsf{GAE}}(s_t, a_t) = \sum_{t'=t}^{T} (\gamma\lambda)^{t'-t} \delta_{t'}$$

$$\delta_{t'} = r(s_{t'}, a_{t'}) + \gamma \hat{V}^{\pi}_{\phi}(s_{t'+1}) - \hat{V}^{\pi}_{\phi}(s_{t'})$$

Review

- Actor-critic algorithms
  - Actor: the policy
  - Critic: value function
  - Reduce variance of policy gradient
- Policy evaluation
  - Fitting value function to policy
- Discount factors
  - Bound the value function
  - Also a variance reduction trick
- Actor-critic algorithm design
  - One network (with two heads) or two networks
  - Batch mode, or online (+ parallel)
- State-dependent baselines
  - Another way to use the critic
  - Can combine: n-step returns or eligibility traces



Actor-critic examples

- High-dimensional continuous control with generalized advantage estimation (Schulman et al., 2016)
  - Batch-mode actor-critic
  - Blends Monte Carlo and function approximator estimators (GAE)
- Asynchronous methods for deep reinforcement learning (Mnih, Badia, Mirza, Graves, Lillicrap, Harley, Silver, Kavukcuoglu, 2016)
  - Online actor critic, parallelized batch
  - n-step returns with $n = 4$
  - Single network for actor and critic

Learning objectives of this lecture

- You should be able to...
  - Extend policy gradient methods to actor-critic algorithms
  - Use policy evaluation to fit the critic, i.e., the value function
  - Be able to implement the basic actor-critic algorithm

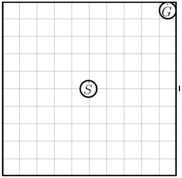
  - Know the actor-critics with n-step returns
  - Know the actor-critics with eligibility traces, i.e., generalized advantage estimation

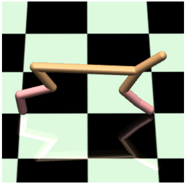
Actor-critic suggested readings

- Lecture 6 of CS285 at UC Berkeley, **Deep Reinforcement Learning, Decision Making, and Control**
  - `http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-6.pdf`
- Classic papers
  - Sutton, McAllester , Singh, Mansour (1999). **Policy gradient methods for reinforcement learning with function approximation**: actor critic algorithms with value function approximation
- DRL actor-critic papers
  - Mnih , Badia , Mirza, Graves, Lillicrap , Harley, Silver, Kavukcuoglu (2016). **Asynchronous methods for deep reinforcement learning**: A3C parallel online actor-critic.
  - Schulman, Moritz, L., Jordan, Abbeel (2016). **High dimensional continuous control using generalized advantage estimation**: batch mode actor-critic with blended Monte Carlo and function approximator returns
  - Gu, Lillicrap , Ghahramani , Turner, L. (2017). **Q-Prop: sample efficient policy gradient with an off-policy critic**: policy gradient with Q-function control variate
  - Tuomas Haarnoja, et al. (2018). **Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor**.

Homework 4

- Study the actor-critic algorithm in detail
- Implement the actor-critic algorithm on problems 1 & 2
  - Problem 1: the point maze navigation, continuous state-action space ($s, a \in \mathbb{R}^2,\ s \in [-0.5, 0.5]^2, a \in [-0.1, 0.1]^2$)
  - Problem 2: the MuJoCo HalfCheetah, make the robot run forward
  - Compare actor-critic with policy gradient
- Write a report introducing the algorithms and your experimentation
  - Explanations, steps, evaluation results, visualizations...
  - Submit the code and the report to *mg20150005@smail.nju.edu.cn*





THE END