Deep Learning

Lecture 10

from really good course in AI masters (https://ozonmasters.ru/reinforcementlearning).

In previous lecture

- Monte-Carlo methods (sample + epsilon greedy)
- Temporal difference learning (SARSA)
- Q-learning
- On-policy/off-policy
- Replay buffer
- DQN

Reinforcement Learning Objective

Lets recall Reinforcement learning objective:

$$\theta^* = \mathrm{argmax}_\theta \, \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_t \gamma^t r_t \right]$$

where:

- θ - parameters of our policy
- $p_\theta(\tau)$ - probability distribution over trajectories generated by policy θ
- $[\sum_t \gamma^t r_t]$- total episodic reward

Reinforcement learning Objective

RL
objective:

$$\theta^* = \mathrm{argmax}_\theta \overbrace{\mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\underbrace{\sum_t \gamma^t r_t}_{r(\tau)}\right]}^{J(\theta)}$$

$$J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\tau)}[r(\tau)] = \int p_\theta(\tau) r(\tau) d\tau$$

GOAL:

We want to find gradient of RL objective $J(\theta)$ with respect to policy parameters θ!

Policy Gradients

To maximize mean expected return:

$$J(\theta) = \mathrm{E}_{\tau \sim p^{\theta}(\tau)}[r(\tau)] = \int p_{\theta}(\tau) r(\tau) d\tau$$

Find:

$$\nabla_{\theta} J(\theta) = \int \nabla_{\theta} p_{\theta}(\tau) r(\tau) d\tau$$

$$= \int p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) r(\tau) d\tau \quad = \mathrm{E}_{\tau \sim p^{\theta}(\tau)}[\nabla_{\theta} \log p_{\theta}(\tau) r(\tau)]$$

Log-derivative trick:

$$\nabla_{\theta} p_{\theta}(\tau) = p_{\theta}(\tau) \frac{\nabla_{\theta} p_{\theta}(\tau)}{p_{\theta}(\tau)} \qquad = p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau)$$

Policy Gradients

Maximize mean expected return:

$$J(\theta) = \mathrm{E}_{\tau \sim p\theta(\tau)}[r(\tau)]$$

Gradients w.r.t θ:

$$\nabla_{\theta} J(\theta) = \mathrm{E}_{\tau \sim p\theta(\tau)}[\nabla_{\theta} log\, p_{\theta}(\tau) r(\tau)]$$

We can rewrite $p_{\theta}(\tau)$ as:

$$p_{\theta}(\tau) = p_{\theta}(s_0, a_0, \dots, s_T, a_T) = p(s_0) \prod^{T}_{t=0} \pi_{\theta}(a_t | s_t) p(s_{t+1} | a_t, s_t)$$

Then:

$$log\, p_{\theta}(\tau) = log\, p(s_0) + \sum^{T}_{t=0}[log\, \pi_{\theta}(a_t | s_t) + log\, p(s_{t+1} | a_t, s_t)]$$

Policy Gradients

Maximize mean expected return:

$$J(\theta) = \mathrm{E}_{\tau \sim p^{\theta}(\tau)}[r(\tau)]$$

Gradients w.r.t θ:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}\left[\nabla_\theta \, log \, p_\theta(\tau) r(\tau)\right]$$

$$\nabla_\theta \left[\cancel{log \, p(s_0)} + \sum_{t=0}^{T} [log \, \pi_\theta(a_t|s_t) + \cancel{log \, p(s_{t+1}|a_t, s_t)}]\right]$$

Policy Gradients:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}\left[\sum_{t=0}^{T} \nabla_\theta \, log \, \pi_\theta(a_t|s_t) r(\tau)\right]$$

Estimating Policy Gradients

We don't know the true expectation there:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}\left[\nabla_\theta \, log \, p_\theta(\tau) r(\tau)\right]$$

And of course we can approximate it with sampling:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{t=0}^{T}\nabla_\theta \, log \, \pi_\theta(a_{i,t}|s_{i,t}) r(\tau_i)\right]$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left[\left(\sum_{t=0}^{T}\nabla_\theta \, log \, \pi_\theta(a_{i,t}|s_{i,t})\right)\left(\sum_{t=0}^{T}\gamma^t r_{i,t}\right)\right]$$

Reinforce

Estimate policy gradients:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\left(\sum_{t=0}^{T}\nabla_\theta \, log\, \pi_\theta(a_{i,t}|s_{i,t})\right)\left(\sum_{t=0}^{T}\gamma^t r_{i,t}\right)\right]$$

Update policy parameters:

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

Reinforce (Pseudocode):

1. *Sample $\{\tau^i\}$ with π_θ (run the policy in the env)*
2. *Estimate policy gradient $\nabla_\theta J(\theta)$ on $\{\tau^i\}$*
3. *Update policy parameters: θ using estimated gradient*
4. *Go to 1*

PG is on-policy algorithm

To train REINFORCE we estimate this:

$$\nabla_\theta J(\theta) = \mathrm{E}_{\tau \sim p^\theta(\tau)}[\nabla_\theta log\, p_\theta(\tau) r(\tau)]$$

REINFORCE (Pseudocode):

1. *Sample $\{\tau^i\}$ with π_θ (run the policy in the env)*
2. *Estimate policy gradient $\nabla_\theta J(\theta)$ on $\{\tau^i\}$*
3. *Update policy parameters: θ using estimated gradient*
4. *Go to 1*

PG is on-policy algorithm

To train REINFORCE we estimate this:

$$\nabla_\theta J(\theta) = E_{\tau \sim p^\theta(\tau)}[\nabla_\theta \log p_\theta(\tau) r(\tau)]$$

We can only use samples generated with π_θ !

On-policy learning:

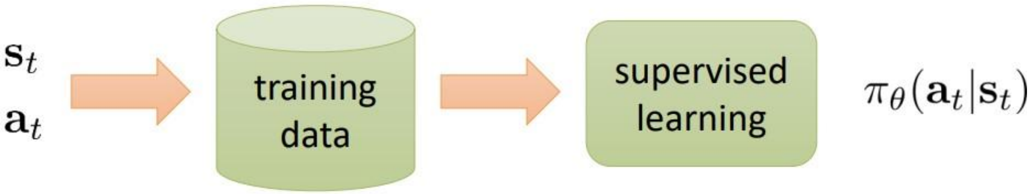
- After one gradient step samples are useless
- PG can be extremely sample inefficient!

REINFORCE (Pseudocode):

1. *Sample $\{\tau^i\}$ with π_θ (run the policy in the env)*
2. *Estimate policy gradient $\nabla_\theta J(\theta)$ on $\{\tau^i\}$*
3. *Update policy parameters: θ using estimated gradient*
4. *Go to 1*

Understanding Policy Gradient

What if we use behaviour cloning to learn a policy?



Cross Entropy-loss for each transition in dataset:

$$H(\overline{y}, y_t) = \frac{1}{|C|}\sum_{j}^{|C|} -y_j \, log \, \overline{y}_j = -log \, \overline{y}_{a_t} \, \frac{1}{|C|}$$

$$= -log \, \pi_\theta(a_t|s_t) \, c$$

Policy at s_t: $\pi_\theta(*|s_t) = \overline{y} = \begin{vmatrix} 0.2 \\ 0.7 \\ 0.1 \end{vmatrix}$

$\boxed{a_t}$

Ground Truth at state s_t: $y = \begin{vmatrix} 1 \\ 0 \\ 0 \end{vmatrix}$

Understanding Policy Gradient

Gradients with behaviour clonning:

$$\nabla_\theta J_{BC}(\theta) = \mathbb{E}_{\tau\sim D}\left[\sum_{t=0}^{T} \nabla_\theta \; - log\,\pi_\theta(a_t|s_t)\;c\right]$$

$\boxed{\text{Goal is to minimize } J_{BC}(\theta)}$

Policy Gradients:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\tau)}\left[\sum_{t=0}^{T} \nabla_\theta \, log\,\pi_\theta(a_t|s_t) r(\tau)\right]$$

$\boxed{\text{Goal is to maximize } J(\theta)}$

Understanding Policy Gradient

Gradients with behaviour clonning:

$$\nabla_\theta J_{BC}(\theta) = \mathbb{E}_{\tau \sim D}\left[\sum_{t=0}^{T} \nabla_\theta \, log \, \pi_\theta(a_t|s_t) \, c\right]$$

Goal is to maximize $-J_{BC}(\theta)$

BC trains policy to choose the same actions as the experts

Policy Gradients:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}\left[\sum_{t=0}^{T} \nabla_\theta \, log \, \pi_\theta(a_t|s_t) r(\tau)\right]$$

Goal is to maximize $J(\theta)$

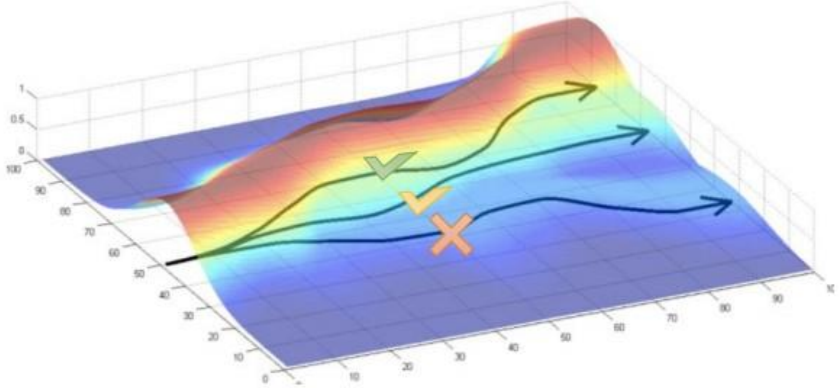
PG trains policy to choose actions that leads to higher episodic returns!

Understanding Policy Gradient

Policy Gradients:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t | s_t) r(\tau) \right]$$

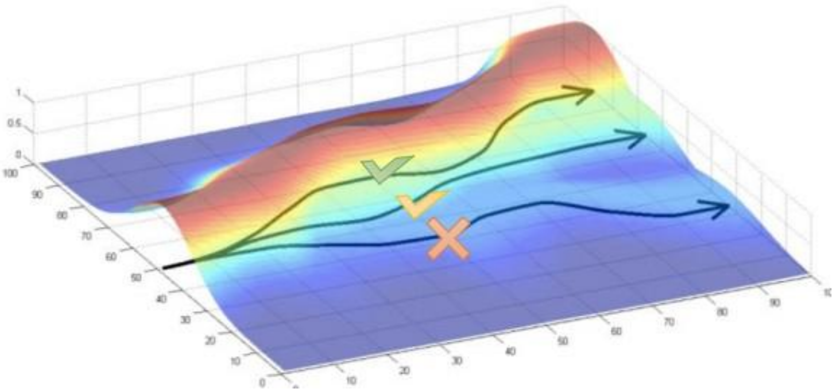
PG trains policy to choose actions that leads to higher episodic returns!



Problem with policy gradients

Problem: **high variance!**

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t | s_t) r(\tau) \right]$$

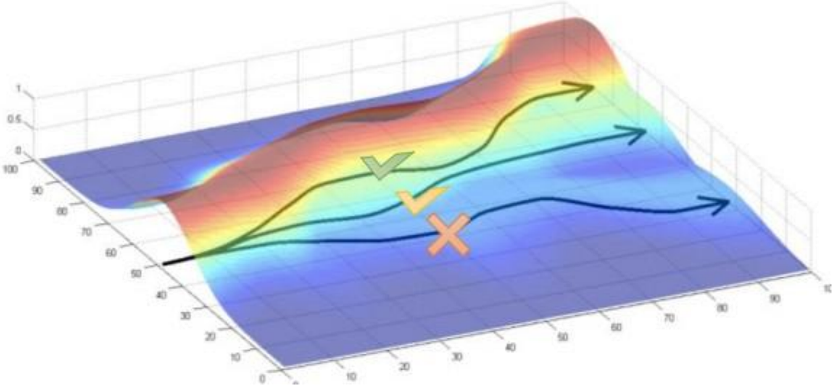


Problem with policy gradients

Problem: **high variance!**

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}\left[\sum_{t=0}^{T} \nabla_\theta \, log \, \pi_\theta(a_t|s_t) r(\tau)\right]$$

Recall value based RL: Monte-Carlo
Return has high variance!



Reducing Variance

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(\sum_{t'=0}^{T}\gamma^{t'} r_{i,t'}\right)\right]$$

Doesn't it look strange?

Causality principle: action at step t cannot affect reward at t' when $t' < t$

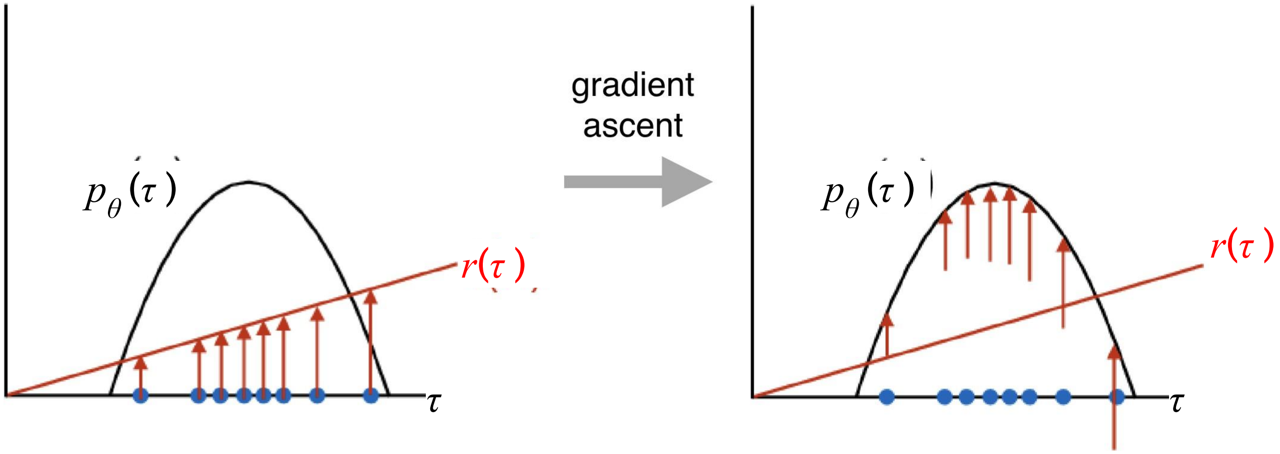
$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(\gamma^{t}\sum_{t'=t}^{T}\gamma^{t'-t} r_{i,t'}\right)\right]$$

Later actions became less relevant!

Final Version:

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

Reducing Variance

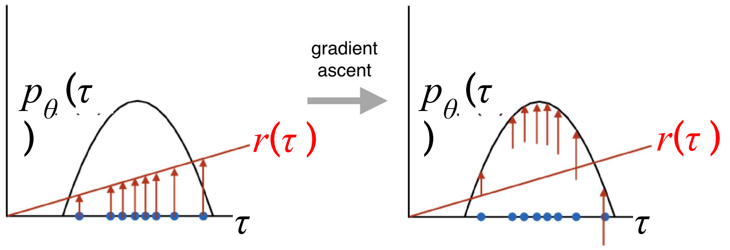


Reducing Variance: Baseline

Updates policy proportionally to how much τ (r) is better than average:

$$\nabla_\theta J(\theta) = \mathrm{E}_{\tau \sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau)(r(\tau) - b)]$$

where: $$b = \mathrm{E}_{\tau \sim p_\theta(\tau)}[r(\tau)]$$



Entropy Regularization

Value-based algorithms (DQN, Q-learning, SARSA, etc.) use ϵ-greedy policy to encourage exploration!

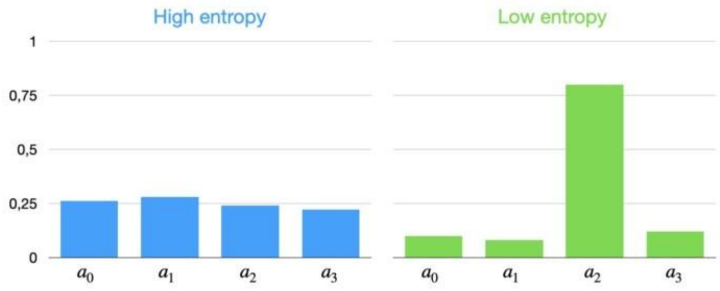
In policy-based algorithms we can utilize a more agile trick:

Entropy Regularization for strategy:

$$H(\pi_\theta(\cdot \mid s_t)) = -\sum_{a \in A} \pi_\theta(a \mid s_t) \; log \, \pi_\theta(a \mid s_t)$$

Adding $-H(\pi_\theta)$ to a loss function:

- encourage agent to act more randomly
- It is still possible to learn any possible probability distribution on actions



Actor-Critic Algorithms

Final Version with "causality improvement" and baseline:

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

Actor-Critic Algorithms

Final Version with "causality improvement" and baseline:

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

Single point estimate of $Q_{\pi_\theta}(s_{i,t}, a_{i,t})$

Now recall Value functions:

$$Q_\pi(s,a) = \mathbb{E}_\pi[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|S_t = s, A_t = a]$$

$$V_\pi(s) = \mathbb{E}_\pi[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|S_t = s]$$

Actor-Critic Algorithms

Combining *PG* and *Value Functions*!

Has lower variance than single point estimate!

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

What about baseline?

$$b = \mathbb{E}_{\tau\sim\pi_\theta}[r(\tau)] = \mathbb{E}_{a\sim\pi_\theta(a|s)}[Q_{\pi_\theta}(s,a)] = V_{\pi_\theta}(s)$$

Better account for causality here....

Advantage Actor-Critic: A2C

Combining *PG* and *Value Functions*!

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(Q_{\pi_\theta}(s_{i,t}, a_{i,t}) - V_{\pi_\theta}(s_{i,t})\right)\right]$$

Advantage Function:

$$A(a, s) = Q_{\pi_\theta}(s, a) - V_{\pi_\theta}(s)$$

how much choosing $\boldsymbol{a_t}$ is better than average policy

Advantage Actor-Critic: A2C

Combining *PG* and *Value Functions*!

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(A_{\pi_\theta}(s_{i,t}, a_{i,t})\right)\right]$$

Advantage Function:

$$A(a, s) = Q_{\pi_\theta}(s, a) - V_{\pi_\theta}(s)$$

how much choosing a_t is better than average policy

It is easier to learn only one function! ...but we can do better:

$$A(a, s) = \mathbb{E}_{s' \sim p(s'|a,s)}[r(s, a) + \gamma E_{a' \sim \pi_\theta(s'|s')}[Q_{\pi_\theta}(a', s')] - V_{\pi_\theta}(s_t)$$

$$= r(s, a) + \gamma \mathbb{E}_{s' \sim p(s'|a,s)}[V_{\pi_\theta}(s')] - V_{\pi_\theta}(s)$$

approximate with a sample

Advantage Actor-Critic: A2C

Combining *PG* and *Value Functions*!

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\left(A_{\pi_\theta}(s_{i,t},a_{i,t})\right)\right]$$

Advantage Function:

$$A_{\pi_\theta}(a_t, s_t) \approx r_t + \gamma V_{\pi_\theta}(s_{t+1}) - V_{\pi_\theta}(s_t)$$

how much choosing $\boldsymbol{a_t}$ is better than average policy

It is easier to learn V-function as it depends on fewer arguments!

A2C Algorithm

Sample $\{\tau\}$ from $\pi_\theta(a_t \mid s_t)$

Policy Improvement step:

- Train actor parameters with Policy Gradient:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})A_{\pi_\theta}(s_{i,t}, a_{i,t})\right]$$

Policy Evaluation step:

- Train Critic to estimate V-function (similar to DQN)

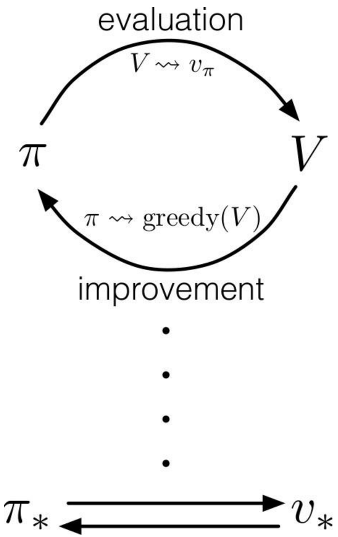
$$\nabla_\phi L(\phi) \approx \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{t=0}^{T}\nabla_\phi \left\|(r_t + \gamma V_{\hat{\phi}}(s_{t+1})) - V_\phi(s_t)\right\|^2\right]$$

ϕ: critic parameters

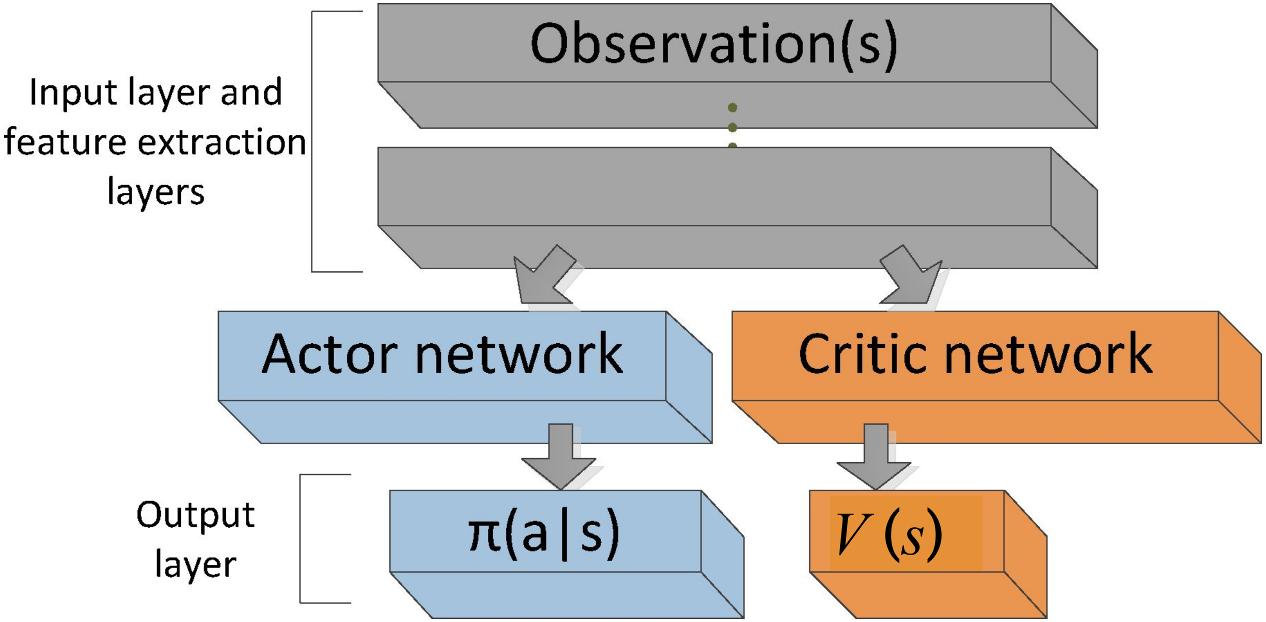
No Target Network (recall DQN) here, just stop the gradients.

Recall

Policy Iteration:



Implementation Details: Architecture



Recap for RL

- What is RL?
- State, action, policy, reward, markovian property, MDP
- Why don’t we use it everywhere?
- V-function, Q-function
- Value Iteration, Policy iteration
- Monte-Carlo methods (sample + epsilon greedy)
- Temporal difference learning (SARSA)
- Q-learning
- On-policy/off-policy
- Replay buffer
- DQN
- Policy gradients (Reinforce + improvements)
- Actor-Critic algorithm and A2C