

RL@ PicsArt
Day3, part1

Policy gradient methods



Small experiment

The next slide contains a question

Please respond as fast as you can!

Small experiment



left or right?

Small experiment



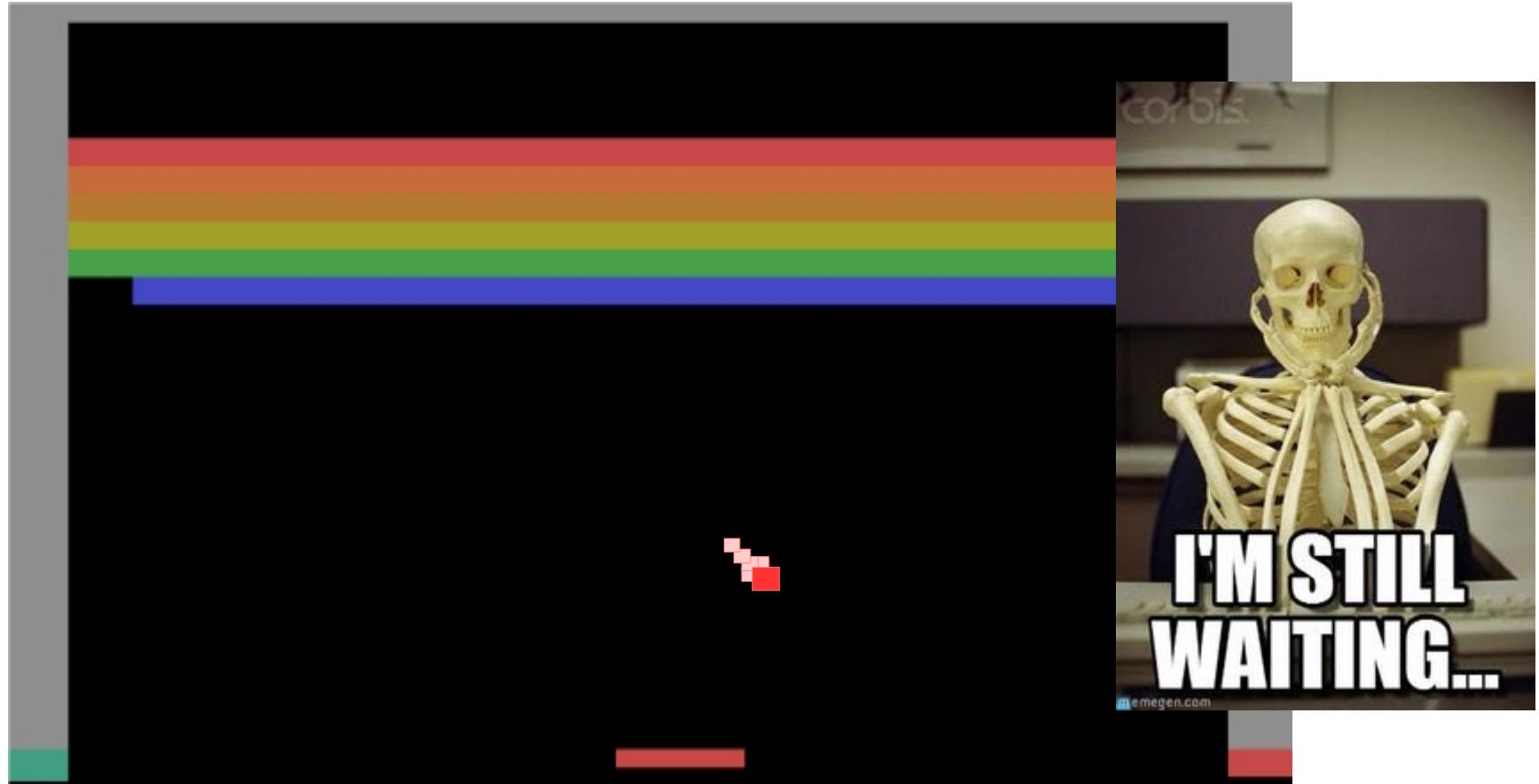
Right! Ready for next one?

Small experiment



What's $Q(s, \text{right})$ under $\gamma=0.99$? 5

Small experiment



What's $Q(s, \text{right})$ under $\gamma=0.99?$ 6

Approximation error

DQN is trained to minimize

$$L \approx E[Q(s_t, a_t) - (r_t + \gamma \cdot \max_{a'} Q(s_{t+1}, a'))]^2$$

Simple 2-state world

	True	(A)	(B)
$Q(s_0, a_0)$	1	1	2
$Q(s_0, a_1)$	2	2	1
$Q(s_1, a_0)$	3	3	3
$Q(s_1, a_1)$	100	50	100

Trivia: Which prediction is better (A/B)?

Approximation error

DQN is trained to minimize

$$L \approx E[Q(s_t, a_t) - (r_t + \gamma \cdot \max_{a'} Q(s_{t+1}, a'))]^2$$

Simple 2-state world

	True	(A)	(B)
$Q(s_0, a_0)$	1	1	2
$Q(s_0, a_1)$	2	2	1
$Q(s_1, a_0)$	3	3	3
$Q(s_1, a_1)$	100	50	100

better policy **less MSE**

Approximation error

DQN is trained to minimize

$$L \approx E[Q(s_t, a_t) - (r_t + \gamma \cdot \max_{a'} Q(s_{t+1}, a'))]^2$$

Simple 2-state world

	True	(A)	(B)
$Q(s_0, a_0)$	1	1	2
$Q(s_0, a_1)$	2	2	1
$Q(s_1, a_0)$	3	3	3
$Q(s_1, a_1)$	100	50	100

Q-learning will prefer worse policy (B)!

better
policy

less
MSE

Conclusion

- Often computing q-values is harder than picking optimal actions!
- We could avoid learning value functions by directly learning agent's policy $\pi_\theta(a|s)$

Conclusion

- Often computing q-values is harder than picking optimal actions!
- We could avoid learning value functions by directly learning agent's policy $\pi_\theta(a|s)$

Q: what algorithm works that way?
(of those we studied)

Conclusion

- Often computing q-values is harder than picking optimal actions!
- We could avoid learning value functions by directly learning agent's policy $\pi_\theta(a|s)$

Q: what algorithm works that way?

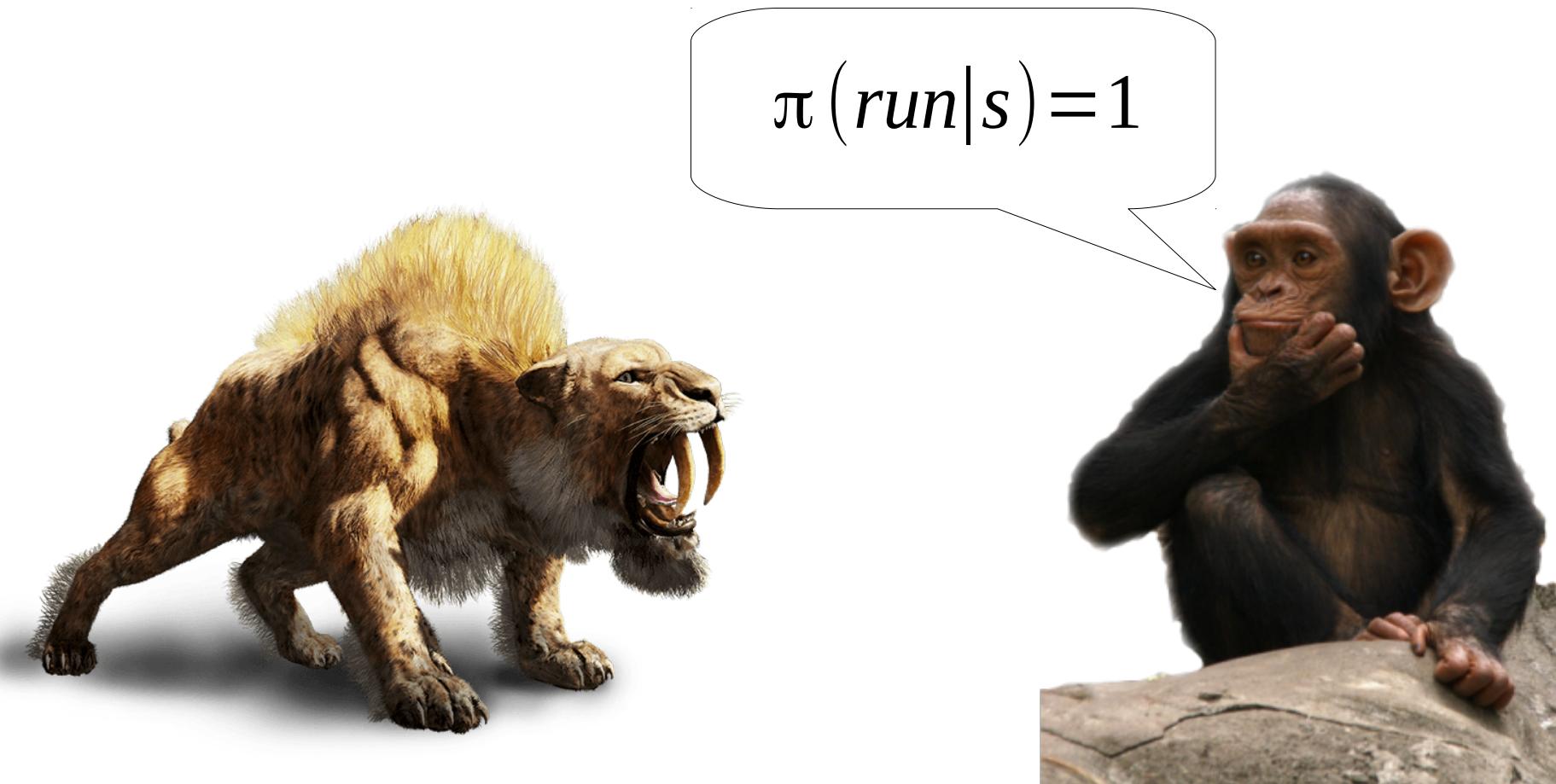
e.g. **crossentropy method**

NOT how humans survived



how humans survived

</chapter>



Policies

In general, two kinds

- Deterministic policy

$$a = \pi_\theta(s)$$

- Stochastic policy

$$a \sim \pi_\theta(a|s)$$

Policies

In general, two kinds

- Deterministic policy

$$a = \pi_\theta(s)$$

- Stochastic policy

$$a \sim \pi_\theta(a|s)$$

Trivia: Any case where stochastic is better?

Policies

In general, two kinds

- Deterministic policy

$$a = \pi_\theta(s)$$

- Stochastic policy

$$a \sim \pi_\theta(a|s)$$

e.g. rock-paper
-scissors

Trivia: Any case where stochastic is better?

Policies

In general, two kinds

- Deterministic policy

$$a = \pi_{\theta}(s)$$

same action each time

- Stochastic policy

$$a \sim \pi_{\theta}(a|s)$$

sampling takes care
of exploration

Policies

In general, two kinds

- Deterministic policy

$$a = \pi_{\theta}(s)$$

same action each time

- Stochastic policy

$$a \sim \pi_{\theta}(a|s)$$

sampling takes care
of exploration

Trivia: how to represent policy in continuous action space? ¹⁹

Policies

In general, two kinds

- Deterministic policy

$$a = \pi_{\theta}(s)$$

same action each time

- Stochastic policy

$$a \sim \pi_{\theta}(a|s)$$

sampling takes care
of exploration

categorical, normal, mixture of normal, whatever

Two approaches

- **Value based:**

Learn value function $Q_\theta(s, a)$ or $V_\theta(s)$

Infer policy $\pi(a|s) = \#_a[a = \operatorname{argmax} Q_\theta(s, a)]$

- **Policy based:**

Explicitly learn policy $\pi_\theta(a|s)$ or $\pi_\theta(s) \rightarrow a$

Implicitly maximize reward over policy

Recap: crossentropy method

- Initialize policy params $\theta_0 \leftarrow random$
- Loop:
 - Sample N sessions
 - elite = take M best sessions and concatenate

$$\theta_{i+1} = \theta_i + \alpha \nabla \sum_i \log \pi_{\theta_i}(a_i | s_i) \cdot [s_i, a_i \in Elite]$$

Policy gradient main idea

Why so complicated?

We'd rather simply maximize G over π !

Objective

Expected reward:

$$J = \underset{\substack{s \sim p(s) \\ a \sim \pi_\theta(s|a)}}{E} R(s, a, s', a', \dots)$$

...

Expected discounted reward:

$$J = \underset{\substack{s \sim p(s) \\ a \sim \pi_\theta(s|a)}}{E} G(s, a)$$

Objective

Expected reward: $R(z)$ setting

$$J = \underset{\substack{s \sim p(s) \\ a \sim \pi_\theta(s|a)}}{E} R(s, a, s', a', \dots)$$

...

Expected discounted reward: $G(s, a) = r + \gamma^* G(s', a')$

$$J = \underset{\substack{s \sim p(s) \\ a \sim \pi_\theta(s|a)}}{E} G(s, a)$$

Objective

Consider an 1-step process for simplicity

$$J = \underset{\substack{s \sim p(s) \\ a \sim \pi_\theta(s|a)}}{E} R(s, a)$$

Objective

Consider an 1-step process for simplicity

$$J = \underset{\substack{s \sim p(s) \\ a \sim \pi_\theta(s|a)}}{E} R(s, a) = \int_s p(s) \int_a \pi_\theta(a|s) R(s, a) da ds$$

Objective

Consider an 1-step process for simplicity

$$J = E_{\substack{s \sim p(s) \\ a \sim \pi_\theta(s|a)}} R(s, a) = \int p(s) \int \pi_\theta(a|s) R(s, a) da ds$$

s

a

Reward for 1-step
session

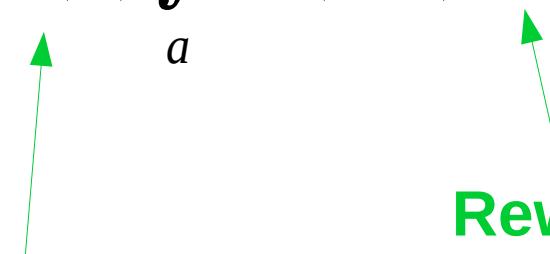
state visitation frequency
(may depend on policy)



Objective

Consider an 1-step process for simplicity

$$J = \underset{\substack{s \sim p(s) \\ a \sim \pi_\theta(s|a)}}{E} R(s, a) = \int_s p(s) \int_a \pi_\theta(a|s) R(s, a) da ds$$



state visitation frequency
(may depend on policy)

Reward for 1-step session

Q: how do we compute that?

Objective

$$J = \underset{\substack{s \sim p(s) \\ a \sim \pi_\theta(s|a)}}{E} R(s, a) = \int_s p(s) \int_a \pi_\theta(a|s) R(s, a) da ds$$

Reward for
agent's action

$$J \approx \frac{1}{N} \sum_{i=0}^N \sum_{s, a \in z_i} R(s, a)$$

sample N sessions
by following $\pi_\theta(a|s)$

Objective

$$J = \underset{\substack{s \sim p(s) \\ a \sim \pi_\theta(s|a)}}{E} R(s, a) = \int_s p(s) \int_a \pi_\theta(a|s) R(s, a) da ds$$

Reward for
agent's action

$$J \approx \frac{1}{N} \sum_{i=0}^N \sum_{s, a \in z_i} R(s, a)$$

sample N sessions

Can we optimize policy now?

Objective

$$J = \underset{\substack{s \sim p(s) \\ a \sim \pi_\theta(s|a)}}{E} R(s, a) = \int_s p(s) \int_a \pi_\theta(a|s) R(s, a) da ds$$

parameters “sit” here

$$J \approx \frac{1}{N} \sum_{i=0}^N \sum_{s, a \in z_i} R(s, a)$$

We don't know how to compute $dJ/d\theta$

Optimization

Finite differences

- Change policy a little, evaluate

$$\nabla J \approx \frac{J_{\theta+\epsilon} - J_\theta}{\epsilon}$$

Stochastic optimization

- Good old crossentropy method
- Maximize probability of “elite” actions

Optimization

Finite differences

- Change policy a little, evaluate

$$\nabla J \approx \frac{J_{\theta+\epsilon} - J_\theta}{\epsilon}$$

Stochastic optimization

- Good old crossentropy method
- Maximize probability of “elite” actions

Optimization

Finite differences

- Change policy a little, evaluate

$$\nabla J \approx \frac{J_{\theta+\epsilon} - J_\theta}{\epsilon}$$

VERY noizy, especially
if both Js are sampled

Stochastic optimization

- Good old crossentropy method
- Maximize probability of “elite” actions

stochastic MDPs;
Throws a most
sessions away

Objective

$$J = \underset{\substack{s \sim p(s) \\ a \sim \pi_\theta(s|a)}}{E} R(s, a) = \int_s p(s) \int_a \pi_\theta(a|s) R(s, a) da ds$$

Wish list:

- Analytical gradient
- Easy/stable approximations

Logderivative trick

Simple math

$$\nabla \log \pi(z) = ???$$

(try chain rule)

Logderivative trick

Simple math

$$\nabla \log \pi(z) = \frac{1}{\pi(z)} \cdot \nabla \pi(z)$$

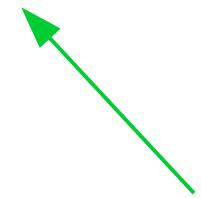
$$\pi \cdot \nabla \log \pi(z) = \nabla \pi(z)$$

Policy gradient

Analytical inference

$$\nabla J = \int_s p(s) \int_a \nabla \pi_\theta(a|s) R(s, a) da ds$$

$$\pi \cdot \nabla \log \pi(z) = \nabla \pi(z)$$



Policy gradient

Analytical inference

$$\nabla J = \int_s p(s) \int_a \nabla \pi_\theta(a|s) R(s, a) da ds$$

$$\pi \cdot \nabla \log \pi(z) = \nabla \pi(z)$$

$$\nabla J = \int_s p(s) \int_a \pi_\theta(a|s) \nabla \log \pi_\theta(a|s) R(s, a) da ds$$

Trivia: anything curious about that formula?

Policy gradient

Analytical inference

$$\nabla J = \int_s p(s) \int_a \nabla \pi_\theta(a|s) R(s, a) da ds$$

$$\pi \cdot \nabla \log \pi(z) = \nabla \pi(z)$$

$$\nabla J = \int_s p(s) \int_a \pi_\theta(a|s) \nabla \log \pi_\theta(a|s) R(s, a) da ds$$

that's expectation :)

Discounted reward case

</chapter>

- Replace R with Q :)

$$\nabla J = \int_s p(s) \int_a \nabla \pi_\theta(a|s) Q(s, a) da ds$$

True action value
a.k.a. $E[G(s,a)]$

$$\pi \cdot \nabla \log \pi(z) = \nabla \pi(z)$$

$$\nabla J = \int_s p(s) \int_a \pi_\theta(a|s) \nabla \log \pi_\theta(a|s) Q(s, a) da ds$$

that's expectation :)

Policy gradient (REINFORCE)

- Policy gradient

$$\nabla J = E_{\substack{s \sim p(s) \\ a \sim \pi_\theta(s|a)}} \nabla \log \pi_\theta(a|s) \cdot Q(s, a)$$

- Approximate with sampling

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \sum_{s, a \in z_i} \nabla \log \pi_\theta(a|s) \cdot Q(s, a)$$

REINFORCE algorithm

- Initialize NN weights $\theta_0 \leftarrow random$
- Loop:
 - Sample N sessions \mathbf{z} under current $\pi_\theta(a|s)$
 - Evaluate policy gradient

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \sum_{s,a \in z_i} \nabla \log \pi_\theta(a|s) \cdot Q(s,a)$$

- Ascend
- $$\theta_{i+1} \leftarrow \theta_i + \alpha \cdot \nabla J$$

REINFORCE algorithm

- Initialize NN weights $\theta_0 \leftarrow \text{random}$
Q: is it on- or off-policy?
- Loop:
 - Sample N sessions \mathbf{z} under current $\pi_\theta(a|s)$
 - Evaluate policy gradient

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \sum_{s,a \in z_i} \nabla \log \pi_\theta(a|s) \cdot Q(s,a)$$

- Ascend
 $\theta_{i+1} \leftarrow \theta_i + \alpha \cdot \nabla J$

REINFORCE algorithm

- Initialize NN weights $\theta_0 \leftarrow \text{random}$
- Loop:
 - Sample N sessions \mathbf{z} under ~~current~~ $\pi_\theta(a|s)$
actions under current policy
= on-policy
 - Evaluate policy gradient

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \sum_{s,a \in z_i} \nabla \log \pi_\theta(a|s) \cdot Q(s,a)$$

- Ascend

$$\theta_{i+1} \leftarrow \theta_i + \alpha \cdot \nabla J$$

REINFORCE algorithm

- Initialize NN weights $\theta_0 \leftarrow random$
- Loop:
 - Sample N sessions \mathbf{z} under current $\pi_\theta(a|s)$
 - Evaluate policy gradient

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \sum_{s,a \in z_i} \nabla \log \pi_\theta(a|s) \cdot Q(s,a)$$

What is better for learning:
random action in good state
or
great action in bad state?

REINFORCE baseline

- Initialize NN weights $\theta_0 \leftarrow random$
- Loop:
 - Sample N sessions \mathbf{z} under current $\pi_\theta(a|s)$
 - Evaluate policy gradient

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \sum_{s,a \in z_i} \nabla \log \pi_\theta(a|s) \cdot Q(s,a)$$

$$Q(s,a) = V(s) + A(s,a)$$

Actions influence $A(s,a)$ only, so $V(s)$ is irrelevant

REINFORCE baseline

- Initialize NN weights $\theta_0 \leftarrow \text{random}$
- Loop:
 - Sample N sessions \mathbf{z} under current $\pi_\theta(a|s)$
 - Evaluate policy gradient

$$\nabla J \approx \frac{1}{N} \sum_{i=0}^N \sum_{s,a \in z_i} \nabla \log \pi_\theta(a|s) \cdot (Q(s,a) - b(s))$$

Anything that doesn't depend on action
ideally, $b(s) = V(s)$

Actor-critic

- Learn both $V(s)$ and $\pi_\theta(a|s)$
- Hope for best of both worlds :)



Advantage actor-critic

Idea: learn both $\pi_\theta(a|s)$ and $V_\theta(s)$

Use $V_\theta(s)$ to learn $\pi_\theta(a|s)$ faster!

Non-trivia: how can we estimate $A(s,a)$ from (s,a,r,s') and V-function?

Advantage actor-critic

Idea: learn both $\pi_\theta(a|s)$ and $V_\theta(s)$

Use $V_\theta(s)$ to learn $\pi_\theta(a|s)$ faster!

Non-trivia: how can we estimate $A(s,a)$ from (s,a,r,s') and V-function?

Advantage actor-critic

Idea: learn both $\pi_\theta(a|s)$ and $V_\theta(s)$

Use $V_\theta(s)$ to learn $\pi_\theta(a|s)$ faster!

$$A(s, a) = Q(s, a) - V(s)$$

$$Q(s, a) = r + \gamma \cdot V(s')$$

$$A(s, a) = r + \gamma \cdot V(s') - V(s)$$

Advantage actor-critic

Idea: learn both $\pi_\theta(a|s)$ and $V_\theta(s)$

Use $V_\theta(s)$ to learn $\pi_\theta(a|s)$ faster!

$$A(s, a) = Q(s, a) - V(s)$$

$$Q(s, a) = r + \gamma \cdot V(s')$$

$$A(s, a) = r + \gamma \cdot V(s') - V(s)$$

Also: n-step
version

Advantage actor-critic

Idea: learn both $\pi_\theta(a|s)$ and $V_\theta(s)$

Use $V_\theta(s)$ to learn $\pi_\theta(a|s)$ faster!

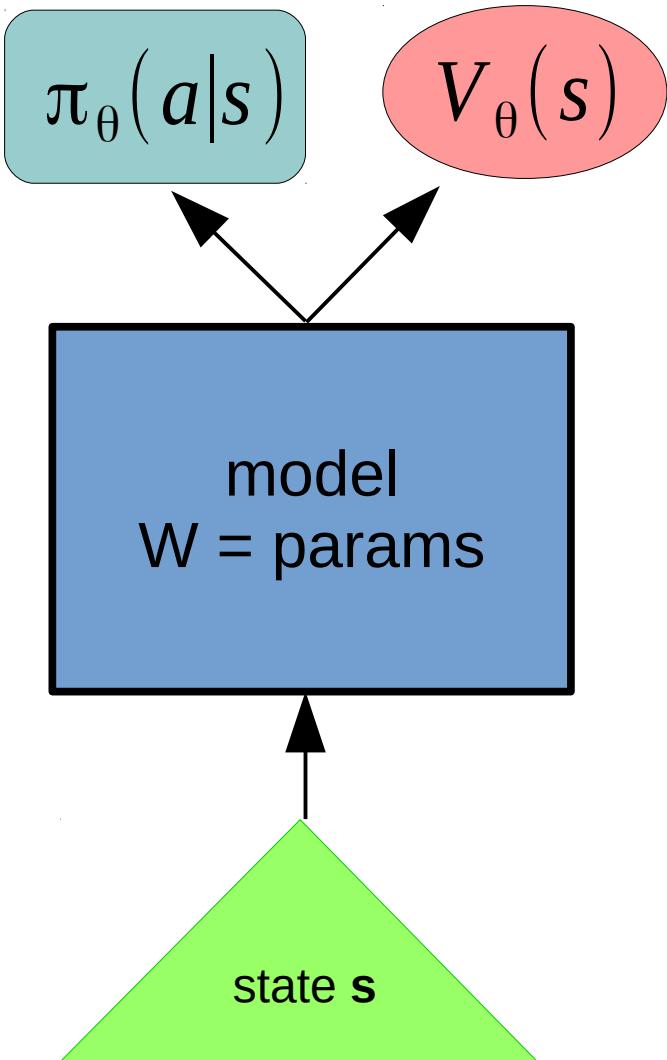
$$A(s, a) = r + \gamma \cdot V(s') - V(s)$$

$$\nabla J_{actor} \approx \frac{1}{N} \sum_{i=0}^N \sum_{s, a \in z_i} \nabla \log \pi_\theta(a|s) \cdot \underline{\underline{A(s, a)}}$$

consider
const

Trivia: how do we train V then?

Advantage actor-critic



Improve policy:

$$\nabla J_{actor} \approx \frac{1}{N} \sum_{i=0}^N \sum_{s,a \in z_i} \nabla \log \pi_\theta(a|s) \cdot A(s, a)$$

Improve value:

$$L_{critic} \approx \frac{1}{N} \sum_{i=0}^N \sum_{s,a \in z_i} (V_\theta(s) - [r + \gamma \cdot V(s')])^2$$

value-based Vs policy-based

Value-based

- Q-learning, SARSA, value-iteration

Policy-based + Actor-critic

- REINFORCE, Advantage Actor-Critic, Crossentropy Method



value-based Vs policy-based

Value-based

- Q-learning, SARSA, value-iteration

Policy-based + Actor-critic

- REINFORCE, Advantage Actor-Critic, Crossentropy Method

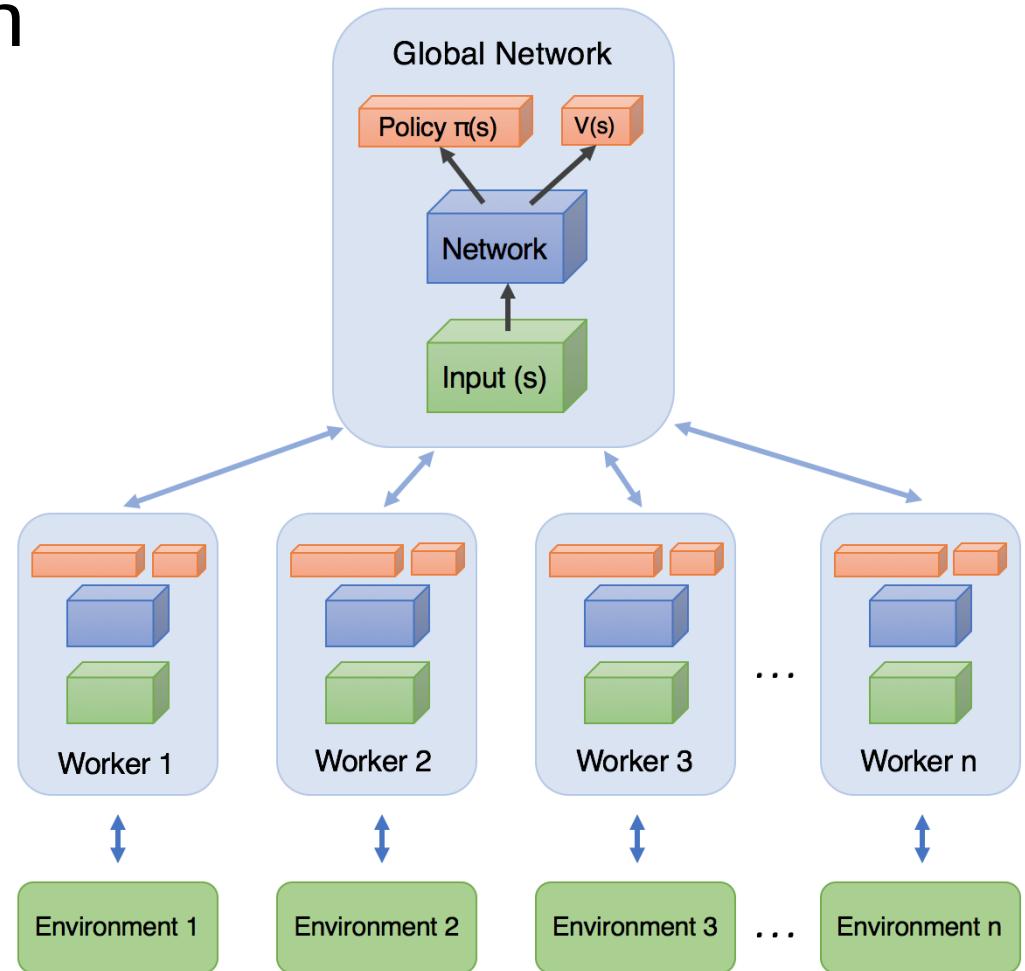
Your guess?



Case Study: A3C

A popular implementation
of advantage actor-critic
using neural net agent

- No experience replay
- Many parallel sessions
- Asynchronous updates



Duct tape zone

- $V(s)$ errors less important than in Q-learning
 - actor still learns even if critic is random, just slower
- Regularize with entropy
 - to prevent premature convergence
- Learn on parallel sessions
 - Or super-small experience replay
- Use logsoftmax for numerical stability



value-based Vs policy-based

Value-based

- Q-learning, SARSA, value-iteration
- Solves harder problem
- Explicit exploration
- Evaluates states & actions
- Easier to train off-policy

Policy-based + Actor-critic

- REINFORCE, Advantage Actor-Critic, Crossentropy Method
- Solves easier problem
- Innate exploration & stochasticity
- Easier for continuous actions
- Compatible with supervised learning



Supervised pre-training

Reinforcement learning usually takes long to find optimal policy completely from scratch.

We can use existing knowledge to help it!

- Human experience
- Known heuristic
- Previous system

Supervised pre-training

Supervised learning:

$$\nabla \text{llh} = E_{x, y_{opt} \sim D} \nabla \log P_\theta(y_{opt} | x)$$

Policy gradient:

$$\nabla J = E_{\substack{s \sim d(s) \\ a \sim \pi(a | obs(s))}} \nabla \log \pi(a | s) Q(s, a)$$

Supervised pre-training

Supervised learning:

$$\nabla \text{llh} = E_{\substack{s, a_{opt} \sim D}} \nabla \log \pi_{\theta}(a_{opt} | s)$$

Policy gradient:

$$\nabla J = E_{\substack{s \sim d(s) \\ a \sim \pi(a | obs(s))}} \nabla \log \pi(a | s) Q(s, a)$$

Supervised pre-training

Supervised learning:

$$\nabla \text{llh} = E_{s, a_{opt} \sim D} \nabla \log \pi_{\theta}(a_{opt} | s)$$

Policy gradient:

$$\nabla J = E_{\substack{s \sim d(s) \\ a \sim \pi(a | obs(s))}} \nabla \log \pi(a | s) Q(s, a)$$

Q: what's different? (apart from $Q(s, a)$)

Supervised pre-training

Supervised learning:

$$\nabla \text{llh} = E_{s, a_{opt} \sim D} \nabla \log \pi_{\theta}(a_{opt} | s)$$

Policy gradient: **reference**

$$\nabla J = E_{\substack{s \sim d(s) \\ a \sim \pi(a | obs(s))}} \nabla \log \pi(a | s) Q(s, a)$$

generated

More out there

This domain is huge!

- TRPO – policy gradient on steroids
- Deterministic policy gradients
- Special cases: