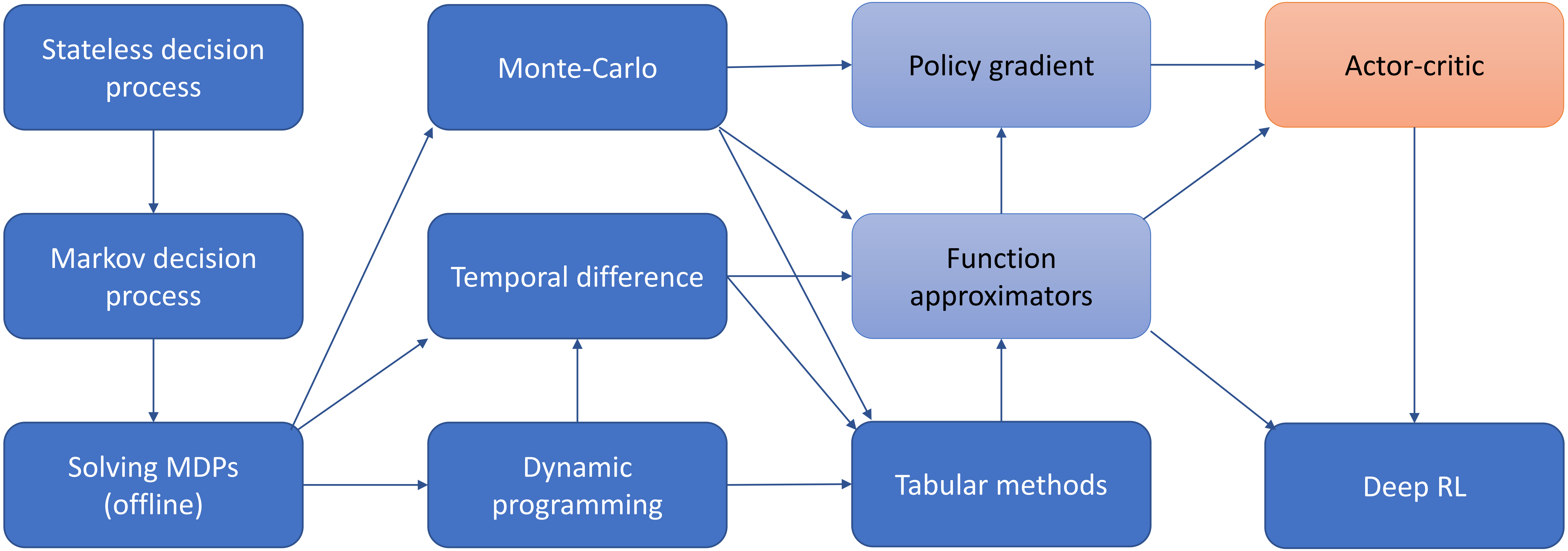
CSCE-642 Reinforcement Learning
Chapter 13.5: Actor-Critic

Instructor: Guni Sharon

CSCE-689, Reinforcement Learning



Policy Gradient

- $\theta_{t+1} = \theta_t + \alpha \widehat{\nabla J(\theta_t)}$
- PG theorem: $\widehat{\nabla J(\theta_t)} \propto \big(q_\pi(S_t, A_t) - b(S_t)\big) \nabla_\theta \log \pi(A_t | S_t; \theta)$
  - REINFORCE+baseline: $\theta_{t+1} = \theta_t + \alpha\big(G_t - b(S_t)\big) \nabla_\theta \log \pi(A_t | S_t; \theta)$
- **Pros**: learning the optimal policy directly is faster to converge in many domains when compared to value-based approaches
- **Cons**: using G_t as an estimator for $q_\pi(S_t, A_t)$ is noisy (high variance) though unbiased
  - Unstable learning

Add a critic

- $\widehat{\nabla J(\theta_t)} \propto \left(q_\pi(S_t, A_t) - b(S_t)\right) \frac{\nabla_\theta \pi(A_t|S_t;\theta)}{\pi(A_t|S_t;\theta)}$
- Define a new estimator: $\hat{q}_\pi(s, a; \theta) \approx q_\pi(s, a)$
  - To be used instead of G_t in REINFORCE
- How should we train $\hat{q}_\pi(s, a; \theta)$?
  - Monte-Carlo updates
    - High variance samples
    - Requires a full episode
  - Bootstrapping e.g., Q-learning
    - Lower variance (though introduces bias)

Critic’s duties

1. Approximate state or action or advantage ($q(s, a) - v(s)$) values
2. Trained via bootstrapping, criticizes the action chosen by the actor

   adjusting the actor’s (policy) gradient

- Is REINFORCE (+ state value baseline) an actor-critic framework?
  - $\theta_{t+1} = \theta_t + \alpha\big(G_t - \hat{v}_\pi(S_t)\big)\nabla_\theta \ln \pi(A_t|S_t;\theta)$
  - NO! the state-value function is used only as a baseline, not as a critic for the chosen action
- The bias introduced through bootstrapping is often worthwhile because it reduces variance and accelerates learning

Benefits from a critic

- REINFORCE with baseline is unbiased* and will converge asymptotically to a local optimum
  - * With a linear state value approximator, and when *b* is not a function of *a*
  - Like all Monte-Carlo methods it tends to learn slowly (produce estimates of high variance)
  - Not suitable for online or for continuing problems
- Temporal-difference methods can eliminate these inconveniences
- In order to gain the TD advantages in the case of policy gradient methods we use actor–critic methods

Actor+critic

- Actor-critic algorithms are a derivative of policy iteration, which alternates between policy evaluation—computing the value function for a policy—and policy improvement—using the value function to obtain a better policy
- In large-scale reinforcement learning problems, it is typically impractical to run either of these steps to convergence, and instead the value function and policy are optimized jointly
- The policy is referred to as the actor, and the value function as the critic

Advantage function

- Eventually we would like to shift the policy towards actions that result in higher return
- what is the benefit from taking action a at state s while following policy π?
  - $A_\pi(s,a) = q_\pi(s,a) - v_\pi(s)$
- This resembles PG with baseline but not the same as q_π is approximated:
  - $\widehat{\nabla J(\theta_t)} = A_\pi(s_t,a_t)\nabla_\theta \ln \pi(a_t|s_t;\theta)$
- Actions that improve on the current policy are encouraged
  - $A_\pi(s,a) > 0$
- Actions that damage the current policy are discouraged
  - $A_\pi(s,a) < 0$

One-step actor-critic

- $A_\pi(s,a) = q_\pi(s,a) - v_\pi(s)$
  - Does that mean that approximating the advantage function requires two function approximators ($\hat{q}$ and $\hat{v}$) ?
  - No since q values can be derived from state values (and vice versa)
  - $\hat{q}_\pi(s_t, a_t) - \hat{v}_\pi(s_t; w) = \widehat{\mathbb{E}}[r_{t+1} + \gamma \hat{v}(s_{t+1}; w)] - \hat{v}(s_t; w)$
- $\theta_{t+1} = \theta_t + \alpha\underbrace{\left(r_{t+1} + \gamma \hat{v}(s_{t+1}; w) - \hat{v}(s_t; w)\right)}_{\delta \text{ – TD error}}\nabla_\theta \ln \pi(a_t | s_t; \theta)$

- One-step Actor-Critic is a fully online, incremental algorithm, with states, actions, and rewards processed as they occur and then never revisited

One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever:
    Initialize S (first state of episode)
    $\boxed{I \leftarrow 1}$
    While S is not terminal:
        $A \sim \pi(\cdot|S, \boldsymbol{\theta})$
        Take action A, observe S', R
        $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \, \delta \nabla_{\mathbf{w}} \hat{v}(S,\mathbf{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A|S, \boldsymbol{\theta})$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$

Keep track of accumulated discount

One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever:
  Initialize S (first state of episode)
  $I \leftarrow 1$
  While S is not terminal:
    $\boxed{A \sim \pi(\cdot|S, \boldsymbol{\theta})}$
    $\boxed{\text{Take action } A\text{, observe } S', R}$
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \, \delta \nabla_{\mathbf{w}} \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A|S, \boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Follow the current policy

One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^d$
Repeat forever:
    Initialize S (first state of episode)
    $I \leftarrow 1$
    While S is not terminal:
        $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
        Take action A, observe S', R
        $\boxed{\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})}$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla_{\mathbf{w}} \hat{v}(S,\mathbf{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A|S,\boldsymbol{\theta})$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$

Compute the TD error

One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^d$
Repeat forever:
  Initialize S (first state of episode)
  $I \leftarrow 1$
  While S is not terminal:
    $A \sim \pi(\cdot|S, \boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\boxed{\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \ \delta \nabla_{\mathbf{w}} \hat{v}(S,\mathbf{w})}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A|S, \boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Update the critic without the accumulated discount. (The discount factor is included in the TD error)

One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^d$
Repeat forever:
  Initialize S (first state of episode)
  $I \leftarrow 1$
  While S is not terminal:
    $A \sim \pi(\cdot|S, \boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \ \delta\nabla_{\mathbf{w}}\hat{v}(S,\mathbf{w})$
    $\boxed{\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I\delta\nabla_{\boldsymbol{\theta}} \ln \pi(A|S, \boldsymbol{\theta})}$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Update the actor with discounting. Early actions matter more.

One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever:
  Initialize S (first state of episode)
  $I \leftarrow 1$
  While S is not terminal:
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \, \delta \nabla_{\mathbf{w}} \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A|S,\boldsymbol{\theta})$
    $\boxed{I \leftarrow \gamma I}$
    $\boxed{S \leftarrow S'}$

Update accumulated discount and progress to the next state

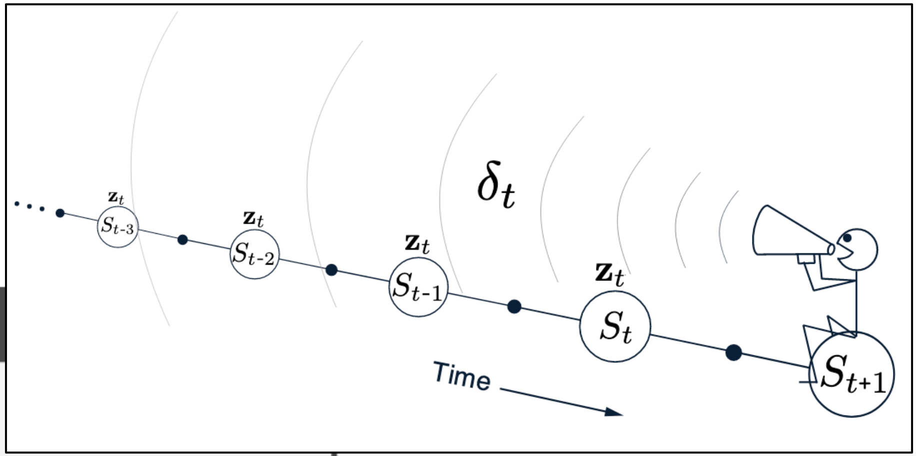
One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever:
  Initialize S (first state of episode)
  $\cancel{I \leftarrow 1}$
  While S is not terminal:
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta\nabla_{\mathbf{w}}\hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \cancel{I} \delta\nabla_{\boldsymbol{\theta}} \ln\pi(A|S,\boldsymbol{\theta})$
    $\cancel{I \leftarrow \gamma I}$
    $S \leftarrow S'$

disregard I in your implementation (it's not needed in practice)

Add eligibility traces



Actor–Critic with Eligibility Traces (episodic)

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w})$
Parameters: trace-decay rates $\lambda^{\boldsymbol{\theta}} \in [0, 1]$, $\lambda^{\mathbf{w}} \in [0, 1]$; step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$

Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$
Repeat forever (for each episode):
    Initialize S (first state of episode)
    $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d'-component eligibility trace vector)
    $\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d-component eligibility trace vector)
    $I \leftarrow 1$
    While S is not terminal (for each time step):
        $A \sim \pi(\cdot|S, \boldsymbol{\theta})$
        Take action A, observe S', R
        $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
        $\mathbf{z}^{\mathbf{w}} \leftarrow \gamma \lambda^{\mathbf{w}} \mathbf{z}^{\mathbf{w}} + \nabla_{\mathbf{w}} \hat{v}(S,\mathbf{w})$
        $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \gamma \lambda^{\boldsymbol{\theta}} \mathbf{z}^{\boldsymbol{\theta}} + I \nabla_{\boldsymbol{\theta}} \ln \pi(A|S, \boldsymbol{\theta})$
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \mathbf{z}^{\mathbf{w}}$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \delta \mathbf{z}^{\boldsymbol{\theta}}$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$

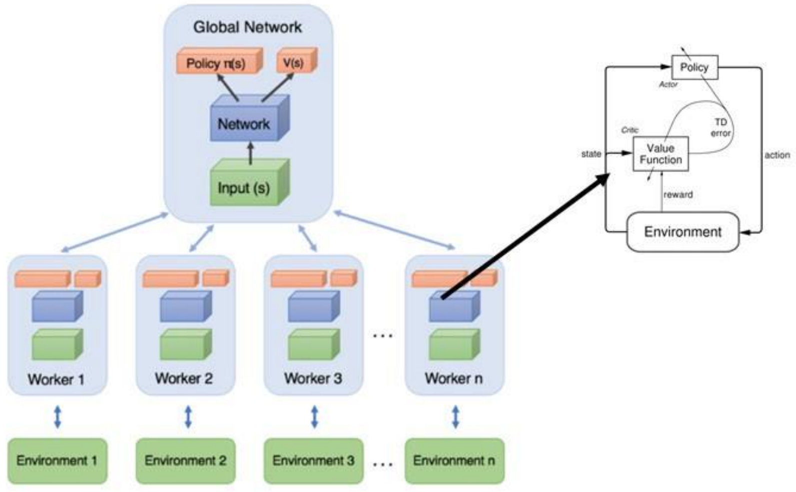
Gradient eligibility per tunable parameter for both the actor and the critic approximators

Asynchronous Methods for Deep Reinforcement Learning [Mnih et al. 2016]

- The sequence of observed data encountered by an online RL agent is non-stationary, and online RL updates are strongly correlated
- Experience replay memory mitigates this issue by reducing non-stationarity and decorrelating updates
- But, uses more memory and computation per "real" interaction
- Requires off-policy learning algorithms that can update from data generated by an older policy
- We want a deep RL mechanism that can decorrelate updates and can work with on-policy algorithms and is memory and computationally efficient

Asynchronized updates

- Asynchronously execute multiple agents in parallel, on multiple instances of the environment
- At any given time-step the parallel agents will be experiencing a variety of different states
  - Decorrelated samples
- Enables a spectrum of on-policy RL algorithms (SARSA, n-step bootstrapping, PG, …) to use DNN
- Efficient utilization of a standard multi-core CPU



Algorithm S3 Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

// Assume global shared parameter vectors θ and θ_v and global shared counter $T = 0$
// Assume thread-specific parameter vectors θ' and θ'_v
Initialize thread step counter $t \leftarrow 1$
repeat
    Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
    Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$
    $t_{start} = t$
    Get state s_t
    repeat
        Perform a_t according to policy $\pi(a_t|s_t;\theta')$
        Receive reward r_t and new state s_{t+1}
        $t \leftarrow t + 1$
        $T \leftarrow T + 1$
    until terminal s_t **or** $t - t_{start} == t_{max}$
    $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_v) & \text{for non-terminal } s_t \text{// Bootstrap from last state} \end{cases}$
    for $i \in \{t - 1, \ldots, t_{start}\}$ **do**
        $R \leftarrow r_i + \gamma R$
        Accumulate gradients wrt θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\theta'_v))$
        Accumulate gradients wrt θ'_v: $d\theta_v \leftarrow d\theta_v + \partial \left(R - V(s_i;\theta'_v)\right)^2 / \partial\theta'_v$
    end for
    Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.
until $T > T_{max}$

Algorithm S3 Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

// Assume global shared parameter vectors θ and θ_v and global shared counter $T = 0$
// Assume thread-specific parameter vectors θ' and θ'_v
Initialize thread step counter $t \leftarrow 1$
repeat
    Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
    Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$
    $t_{start} = t$
    Get state s_t
    repeat
        Perform a_t according to policy $\pi(a_t|s_t;\theta')$
        Receive reward r_t and new state s_{t+1}
        $t \leftarrow t + 1$
        $T \leftarrow T + 1$
    until terminal s_t **or** $t - t_{start} == t_{max}$
    $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_v) & \text{for non-terminal } s_t \text{// Bootstrap from last state} \end{cases}$
    for $i \in \{t-1, \ldots, t_{start}\}$ **do**
        $R \leftarrow r_i + \gamma R$
        Accumulate gradients wrt θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\theta'_v))$
        Accumulate gradients wrt θ'_v: $d\theta_v \leftarrow d\theta_v + \partial \left(R - V(s_i;\theta'_v)\right)^2 / \partial \theta'_v$
    end for
    Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.
until $T > T_{max}$

Make a copy of the current global actor and critic

Algorithm S3 Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

// Assume global shared parameter vectors θ and θ_v and global shared counter $T = 0$
// Assume thread-specific parameter vectors θ' and θ'_v
Initialize thread step counter $t \leftarrow 1$
repeat
    Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
    Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$
    $t_{start} = t$
    Get state s_t
    repeat
        Perform a_t according to policy $\pi(a_t|s_t; \theta')$
        Receive reward r_t and new state s_{t+1}
        $t \leftarrow t + 1$
        $T \leftarrow T + 1$
    until terminal s_t **or** $t - t_{start} == t_{max}$
    $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_v) & \text{for non-terminal } s_t \text{// Bootstrap from last state} \end{cases}$
    for $i \in \{t-1, \ldots, t_{start}\}$ **do**
        $\boxed{R \leftarrow r_i + \gamma R}$
        Accumulate gradients wrt θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i; \theta')(R - V(s_i; \theta'_v))$
        Accumulate gradients wrt θ'_v: $d\theta_v \leftarrow d\theta_v + \partial\left(R - V(s_i; \theta'_v)\right)^2 / \partial\theta'_v$
    end for
    Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.
until $T > T_{max}$

Compute the n-step return

Algorithm S3 Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

// Assume global shared parameter vectors θ and θ_v and global shared counter $T = 0$
// Assume thread-specific parameter vectors θ' and θ'_v
Initialize thread step counter $t \leftarrow 1$
repeat
  Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
  Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$
  $t_{start} = t$
  Get state s_t
  repeat
    Perform a_t according to policy $\pi(a_t|s_t;\theta')$
    Receive reward r_t and new state s_{t+1}
    $t \leftarrow t + 1$
    $T \leftarrow T + 1$
  until terminal s_t **or** $t - t_{start} == t_{max}$
  $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_v) & \text{for non-terminal } s_t \text{// Bootstrap from last state} \end{cases}$
  for $i \in \{t-1, \ldots, t_{start}\}$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients wrt θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\theta'_v))$
    Accumulate gradients wrt θ'_v: $d\theta_v \leftarrow d\theta_v + \partial\left(R - V(s_i;\theta'_v)\right)^2 / \partial\theta'_v$
  end for
  Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.
until $T > T_{max}$

Update thread specific gradients. For the actor, using the PG theorem. For the critic, using squared loss

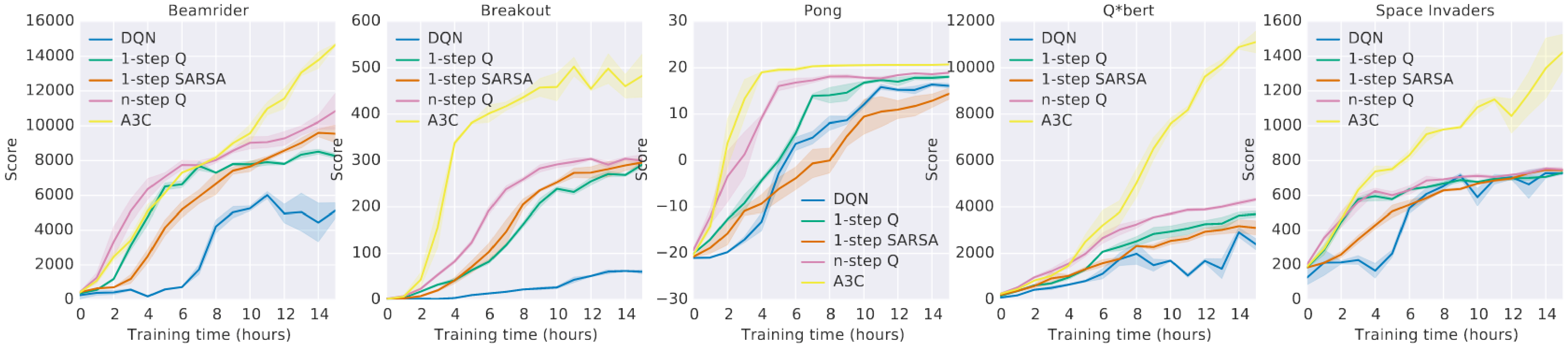
Algorithm S3 Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

```
// Assume global shared parameter vectors θ and θ_v and global shared counter T = 0
// Assume thread-specific parameter vectors θ' and θ'_v
Initialize thread step counter t ← 1
repeat
    Reset gradients: dθ ← 0 and dθ_v ← 0.
    Synchronize thread-specific parameters θ' = θ and θ'_v = θ_v
    t_start = t
    Get state s_t
    repeat
        Perform a_t according to policy π(a_t|s_t; θ')
        Receive reward r_t and new state s_{t+1}
        t ← t + 1
        T ← T + 1
    until terminal s_t or t − t_start == t_max
    R = { 0            for terminal s_t
        { V(s_t, θ'_v)  for non-terminal s_t // Bootstrap from last state
    for i ∈ {t − 1, . . . , t_start} do
        R ← r_i + γR
        Accumulate gradients wrt θ': dθ ← dθ + ∇_θ' log π(a_i|s_i; θ')(R − V(s_i; θ'_v))
        Accumulate gradients wrt θ'_v: dθ_v ← dθ_v + ∂(R − V(s_i; θ'_v))^2 / ∂θ'_v
    end for
    Perform asynchronous update of θ using dθ and of θ_v using dθ_v.
until T > T_max
```

Update the global actor and critic based on the accumulated thread specific gradients

Asynchronized advantage actor-critic (A3C)

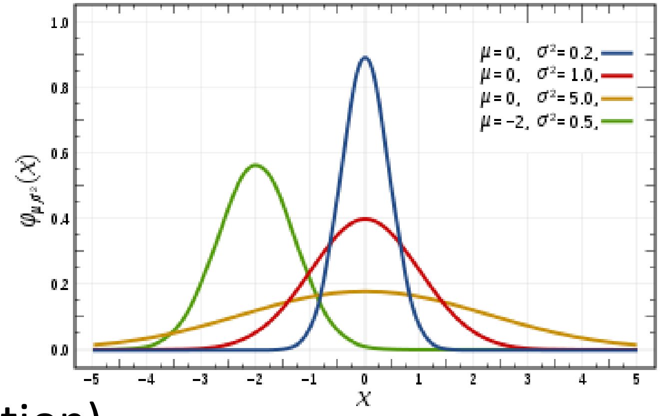
- Training time (wall clock) comparison for DQN and asynchronous algorithms on Atari 2600 games averaged over 5 runs
- DQN results for different seeds on Nvidia K40 GPU. asynchronous methods results from best 5 of 50 workers using 16 CPU cores



Continuous actions

- Policy-based methods offer practical ways of dealing with continuous action spaces
- E.g., an action that can take any value from the range $[-1,1]$
- Learning action values i.e., $q(s,a)$, is not practicle
- How can we train a policy?
- learn a probability density function!
  - $\pi: \mathcal{A} \times \mathcal{S} \mapsto \mathrm{Pr}$
  - Becomes $\pi: \mathcal{S} \mapsto$ PDF

Continuous actions

- Assume that our policy is defined by a Gaussian PDF over a continuous action:
  

  - $f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$
  - $\pi(s;\theta) \sim \frac{1}{\sigma(s;\theta)\sqrt{2\pi}} \exp\left(-\frac{\left(a-\mu(s;\theta)\right)^2}{2\sigma(s;\theta)^2}\right)$
    - Here, the red π is the constant 3.14159 (sorry for overloading notation)
- What are the tunable parameters of such a policy?
  - The mean and standard deviation for each state
  - We can use a function approximator to estimate μ and σ
    - $\mu : \mathcal{S} \times \mathbb{R}^{|\theta_\mu|} \rightarrow \mathbb{R}$ and $\sigma : \mathcal{S} \times \mathbb{R}^{|\theta_\sigma|} \rightarrow \mathbb{R}^+$ (the tunable parameters are θ_μ and θ_σ)

Updating the PDF

- Approximate μ through θ_μ and σ through θ_σ
- $\pi(s;\theta_\mu,\theta_\sigma) \sim \frac{1}{\sigma(s;\theta_\sigma)\sqrt{2\pi}} \exp\left(-\frac{\left(a-\mu(s;\theta_\mu)\right)^2}{2\sigma(s;\theta_\sigma)^2}\right)$
- Can we compute the policy gradient?
  - The policy is defined by both the mean and standard deviation and both need to be updated
  - But in what direction?
  - $A\,\boxed{\nabla_{\theta_\mu} \ln \pi(a|s;\theta_\mu,\theta_\sigma)}$
  - $A\,\boxed{\nabla_{\theta_\sigma} \ln \pi(a|s;\theta_\mu,\theta_\sigma)}$

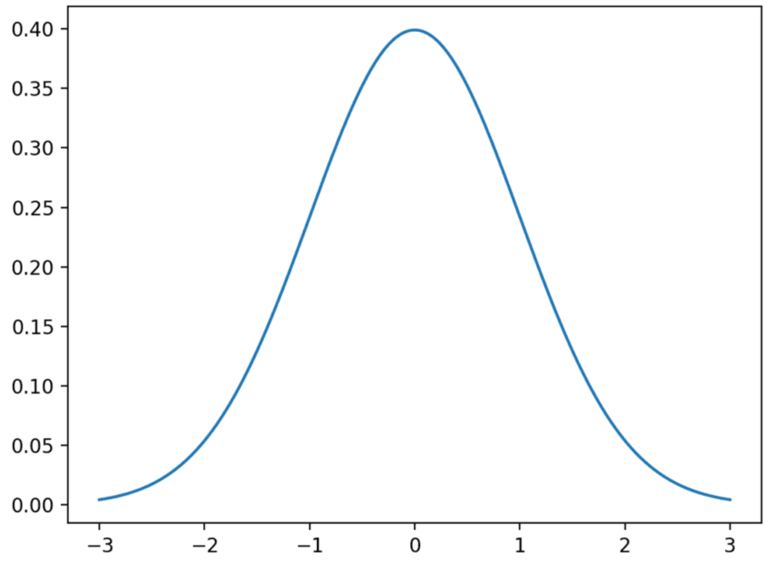
?

Gaussian gradient

- $\pi(s;\theta_\mu,\theta_\sigma) \sim \frac{1}{\sigma(s;\theta_\sigma)\sqrt{2\pi}} \exp\left(-\frac{\left(a-\mu(s;\theta_\mu)\right)^2}{2\sigma(s;\theta_\sigma)^2}\right)$
- Recall: $\frac{df}{dx}\left[f\left(g(x)\right)\right] = \frac{df}{dg}\frac{dg}{dx}$ and $\frac{df}{dx}[f(x)g(x)] = f(x)g'(x) + f'(x)g(x)$ and $\frac{df}{dx}\left[e^{g(x)}\right] = e^{g(x)}g'(x)$
- $\nabla_{\theta_\mu} \ln \pi(a|s;\theta_\mu,\theta_\sigma) = \frac{\nabla_{\theta_\mu}\pi(a|s;\theta_\mu,\theta_\sigma)}{\pi(a|s;\theta_\mu,\theta_\sigma)} = \frac{a-\mu(s;\theta_\mu)}{\sigma(s;\theta_\sigma)^2}\nabla_{\theta_\mu}\mu(s;\theta_\mu)$
- $\nabla_{\theta_\sigma} \ln \pi(a|s;\theta_\mu,\theta_\sigma) = \frac{\nabla_{\theta_\sigma}\pi(a|s;\theta_\mu,\theta_\sigma)}{\pi(a|s;\theta_\mu,\theta_\sigma)} = \left(\frac{\left(a-\mu(s;\theta_\mu)\right)^2}{\sigma(s;\theta_\sigma)^3} - \frac{1}{\sigma(s;\theta_\sigma)}\right)\nabla_{\theta_\sigma}\sigma(s;\theta_\sigma)$

Gaussian PG

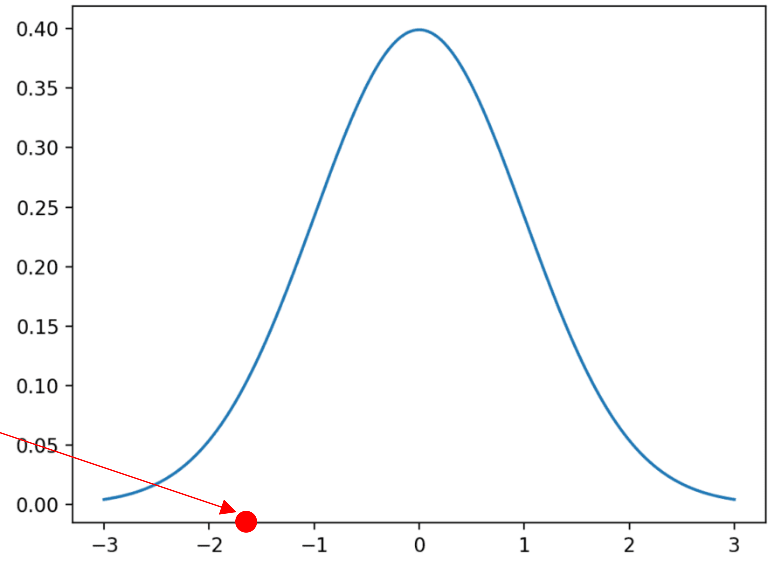
- $\nabla_{\theta_\mu} \ln \pi(a|s;\theta_\mu,\theta_\sigma) = \frac{a-\mu(s;\theta_\mu)}{\sigma(s;\theta_\sigma)^2} \nabla_{\theta_\mu}\mu(s;\theta_\mu)$
- $\nabla_{\theta_\sigma} \ln \pi(a|s;\theta_\mu,\theta_\sigma) = \left(\frac{\left(a-\mu(s;\theta_\mu)\right)^2}{\sigma(s;\theta_\sigma)^3} - \frac{1}{\sigma(s;\theta_\sigma)}\right) \nabla_{\theta_\sigma}\sigma(s;\theta_\sigma)$
    - $\theta_\mu = \theta_\mu + \alpha^\mu A \, \nabla_{\theta_\mu} \ln \pi(a|s;\theta_\mu,\theta_\sigma)$
    - $\theta_\sigma = \theta_\sigma + \alpha^\sigma A \, \nabla_{\theta_\sigma} \ln \pi(a|s;\theta_\mu,\theta_\sigma)$



Gaussian PG

- $\nabla_{\theta_\mu} \ln \pi(a|s;\theta_\mu,\theta_\sigma) = \frac{a-\mu(s;\theta_\mu)}{\sigma(s;\theta_\sigma)^2} \nabla_{\theta_\mu}\mu(s;\theta_\mu)$
- $\nabla_{\theta_\sigma} \ln \pi(a|s;\theta_\mu,\theta_\sigma) = \left(\frac{\left(a-\mu(s;\theta_\mu)\right)^2}{\sigma(s;\theta_\sigma)^3} - \frac{1}{\sigma(s;\theta_\sigma)}\right) \nabla_{\theta_\sigma}\sigma(s;\theta_\sigma)$
    - $\theta_\mu = \theta_\mu + \alpha^\mu A \nabla_{\theta_\mu} \ln \pi(a|s;\theta_\mu,\theta_\sigma)$
    - $\theta_\sigma = \theta_\sigma + \alpha^\sigma A \nabla_{\theta_\sigma} \ln \pi(a|s;\theta_\mu,\theta_\sigma)$

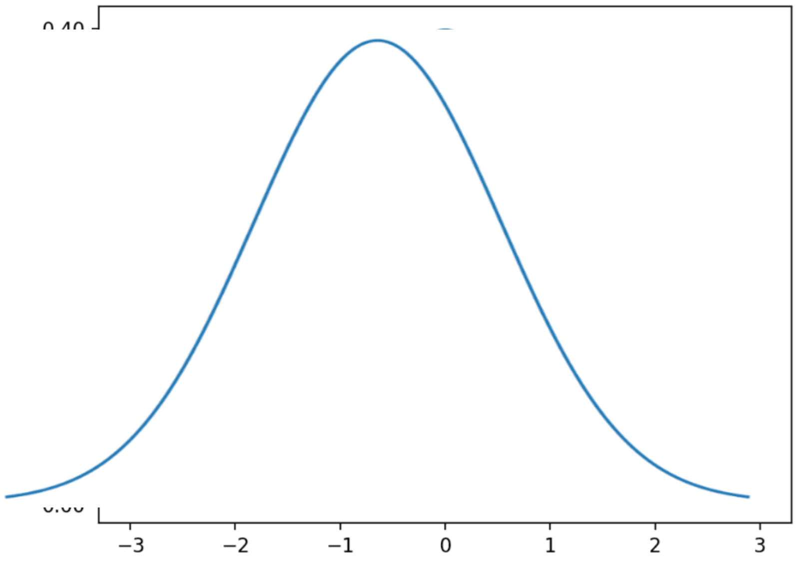
This sampled action yielded a positive advantage?



Gaussian PG

- $\nabla_{\theta_\mu} \ln \pi(a|s;\theta_\mu,\theta_\sigma) = \frac{a-\mu(s;\theta_\mu)}{\sigma(s;\theta_\sigma)^2} \nabla_{\theta_\mu}\mu(s;\theta_\mu)$
- $\nabla_{\theta_\sigma} \ln \pi(a|s;\theta_\mu,\theta_\sigma) = \left(\frac{\left(a-\mu(s;\theta_\mu)\right)^2}{\sigma(s;\theta_\sigma)^3} - \frac{1}{\sigma(s;\theta_\sigma)}\right) \nabla_{\theta_\sigma}\sigma(s;\theta_\sigma)$
    - $\theta_\mu = \theta_\mu + \alpha^\mu A \,\nabla_{\theta_\mu} \ln \pi(a|s;\theta_\mu,\theta_\sigma)$
    - $\theta_\sigma = \theta_\sigma + \alpha^\sigma A \,\nabla_{\theta_\sigma} \ln \pi(a|s;\theta_\mu,\theta_\sigma)$

Increase sigma and reduce mu



What did we learn?

- REINFORCE:
  - $\widehat{\nabla J(\theta_t)} = \boxed{G_t} \nabla_\theta \ln \pi(A_t|S_t;\theta)$
- Q Actor-Critic:
  - $\widehat{\nabla J(\theta_t)} = \boxed{\hat{q}(S_t, A_t; w)} \nabla_\theta \ln \pi(A_t|S_t;\theta)$
- REINFORCE + baseline:
  - $\widehat{\nabla J(\theta_t)} = \left(\boxed{G_t} - \boxed{\hat{v}(S_t; w)}\right) \nabla_\theta \ln \pi(A_t|S_t;\theta)$
- Advantage Actor-Critic:
  - $\widehat{\nabla J(\theta_t)} = \left(\boxed{R_{t+1} + \gamma \hat{v}(S_{t+1}; w)} - \boxed{\hat{v}(S_t; w)}\right) \nabla_\theta \ln \pi(A_t|S_t;\theta)$

What next?

- **Lecture**: Trust regions
- **Assignments**:
  - A2C
  - REINFORCE
  - Deep Q-Learning
- **Quiz (on Canvas)**:
  - Policy Gradient
  - Deep Q-Learning
  - Eligibility Traces
- **Project**:
  - Literature survey