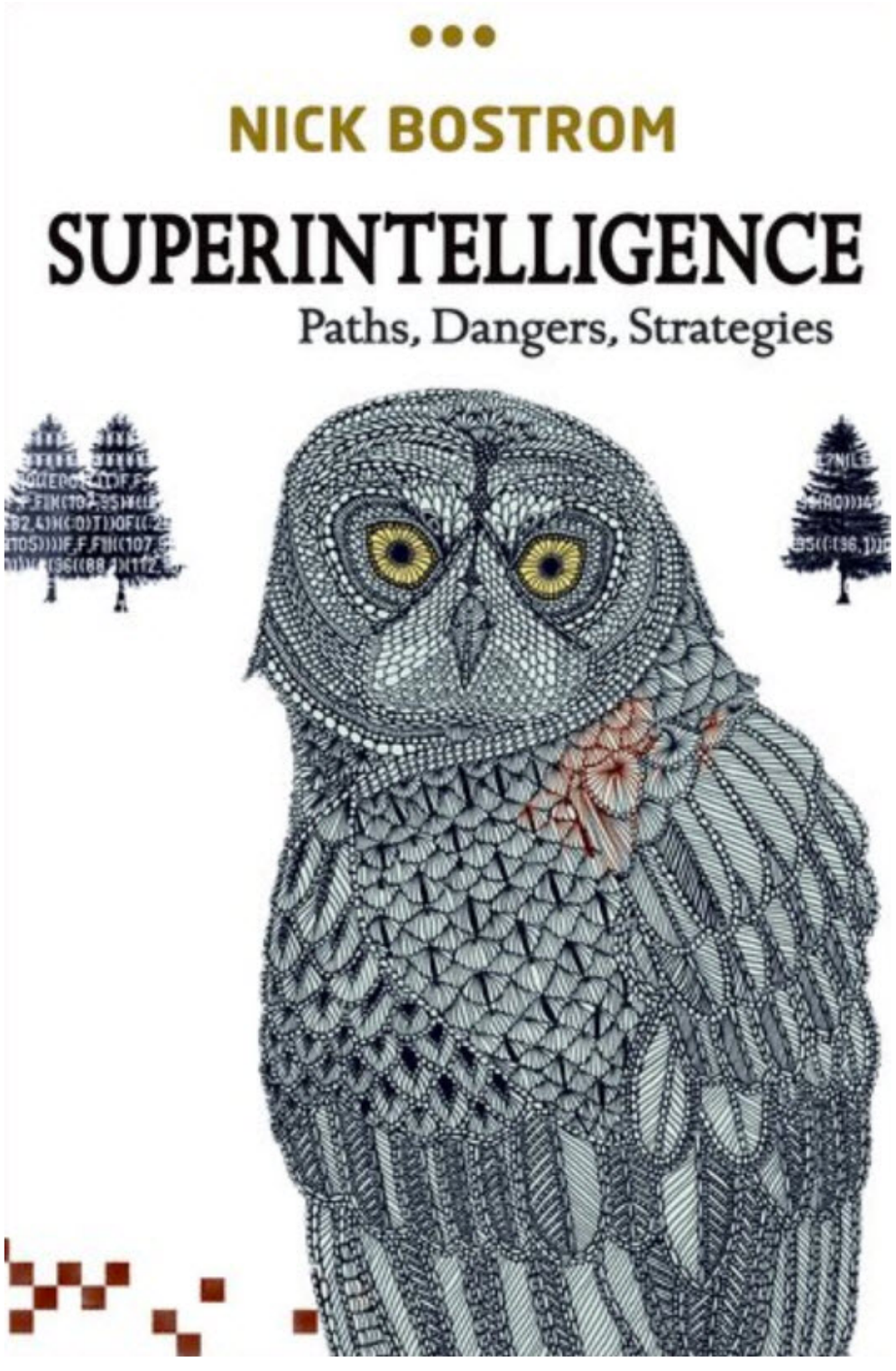
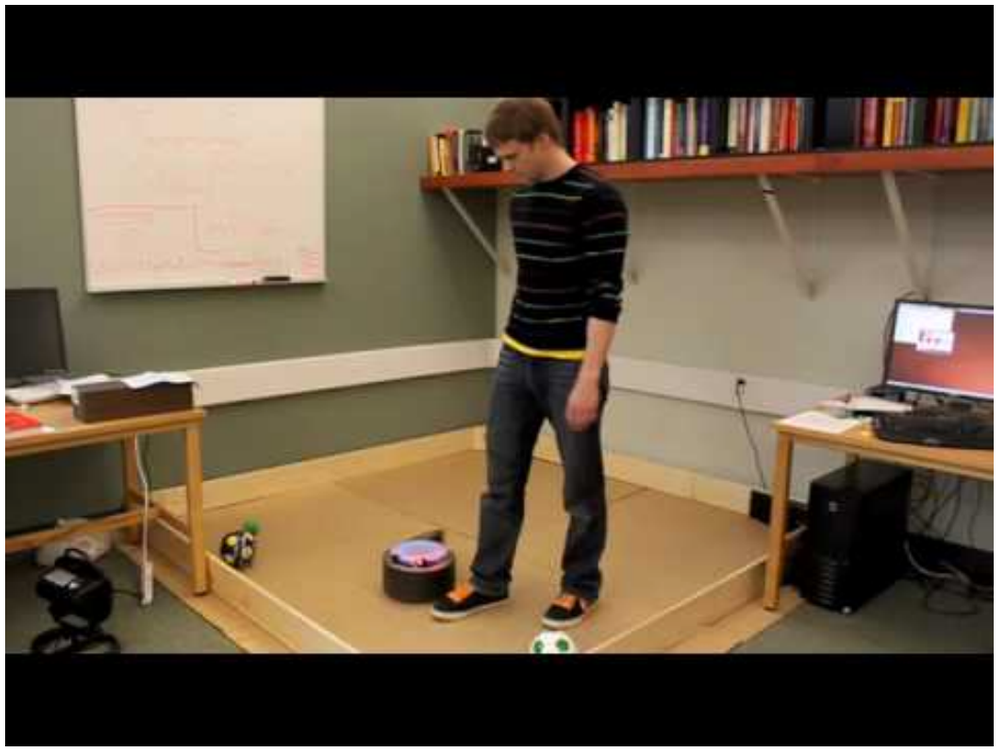


- **Hawking**:
  - “A super intelligent AI will be extremely good at accomplishing its goals, and if those goals aren't aligned with ours, we're in trouble”
- **Musk**:
  - “We need to be super careful with AI. Potentially more dangerous than nukes.”

- **Sutton**: https://www.youtube.com/watch?v=pD-FWetbvN8&feature=youtu.be

Consider the following domains

- Continuous obstacle avoidance



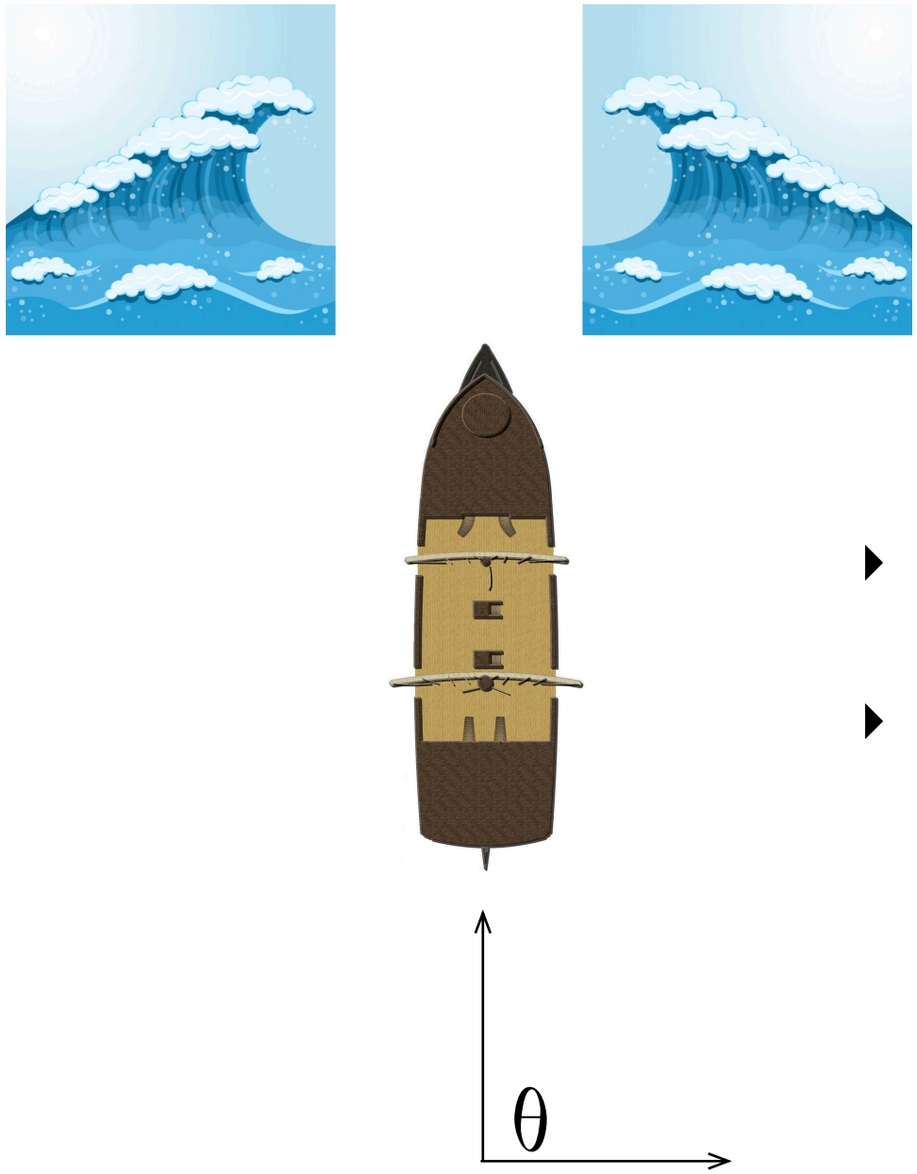
  - no terminal states
  - reward = forward velocity - bump

- With discounting we favor near term rewards…what if that leads to some part of the office where the robot was constantly bumping?

Putting

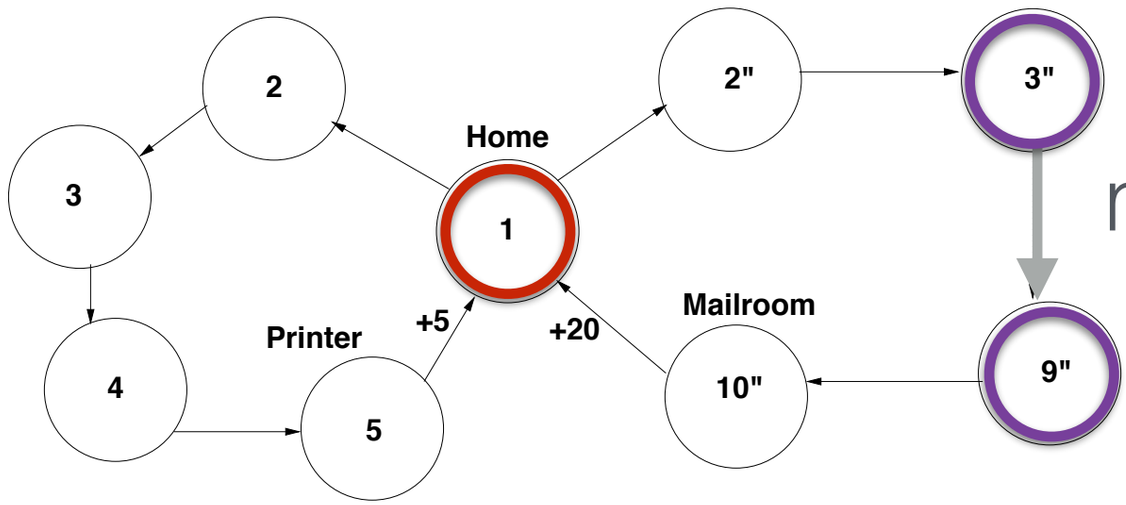
Pit world

Ship steering

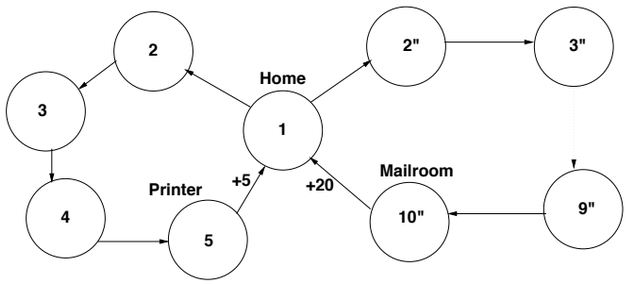


- no terminal states
- reward = forward velocity

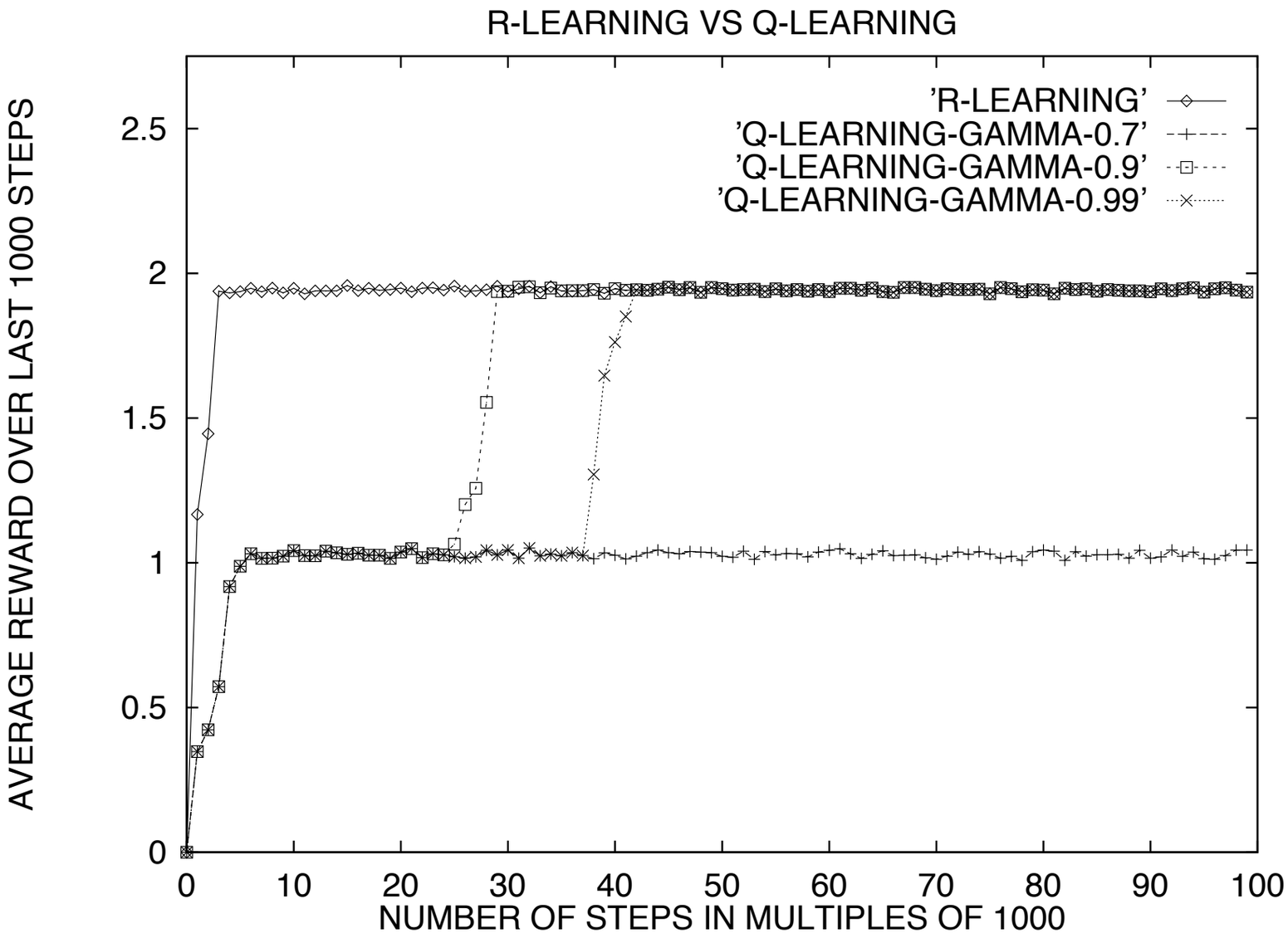
Small Robot MDP



many transitions



Small MDP



- If γ is small, 0.7, Q-learning will converge to the sub-optimal (wrt average reward) solution
- Larger γ cause slow convergence to the correct solution

Approaches to control

1. Previous approach: *Action-value methods*:
   - learn the value of each action;
   - pick the max (usually)
2. New approach: *Policy-gradient methods*:
   - learn the parameters of a stochastic policy
   - update by gradient ascent in performance
   - includes ***actor-critic*** *methods*, which learn *both* value and policy parameters

Issues with discounting

- In continuing tasks we have, up till now, assumed rewards more distant in time are less valuable then immediate rewards
- Sometimes we don't naturally have terminations, and we care about long term performance
  - we can try to simulate this with very large discounts
  - but learning is very slow
- Other algorithms exist, that we will learn about today

Issues with discounting

- In the function approximation setting (e.g., using tile coding), and learning control (e.g., Sarsa)
  - We must specify which states we care about
- We can specify a start state or distribution over start states as the states we care about getting maximum reward from
- However, if you made this distribution equal to the on-policy distribution—distribution over states we see under π
  - you can prove that the discount rate has no effect on the ordering of policies
  - the result is actually the same as using the **average reward formulation**

Average reward review

- Seek to maximize the average expected reward per time step

$$r(\pi) \doteq \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}[R_t \mid A_{0:t-1} \sim \pi]$$

Maximize:

$$= \lim_{t\to\infty} \mathbb{E}[R_t \mid A_{0:t-1} \sim \pi]\,,$$

- Notice r(π) is a function of the policy but *not* **state**
  - independent of state state, unlike the return
- We assume the MDP is ergodic
  - can reach any state, from any other state, under any policy

Average reward

- Returns:

$$G_t \doteq R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + R_{t+3} - r(\pi) + \cdots$$

- Update target:

$$U_t \doteq R_{t+1} - \bar{R}_t + \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}) \quad \text{or} \quad U_t \doteq R_{t+1} - \bar{R}_t + \hat{v}(S_{t+1}, \mathbf{w})$$

estimate of $r(\pi)$

Differential semi-gradient Sarsa for estimating $\hat{q} \approx q_*$

Input: a differentiable function $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Parameters: step sizes $\alpha, \beta > 0$

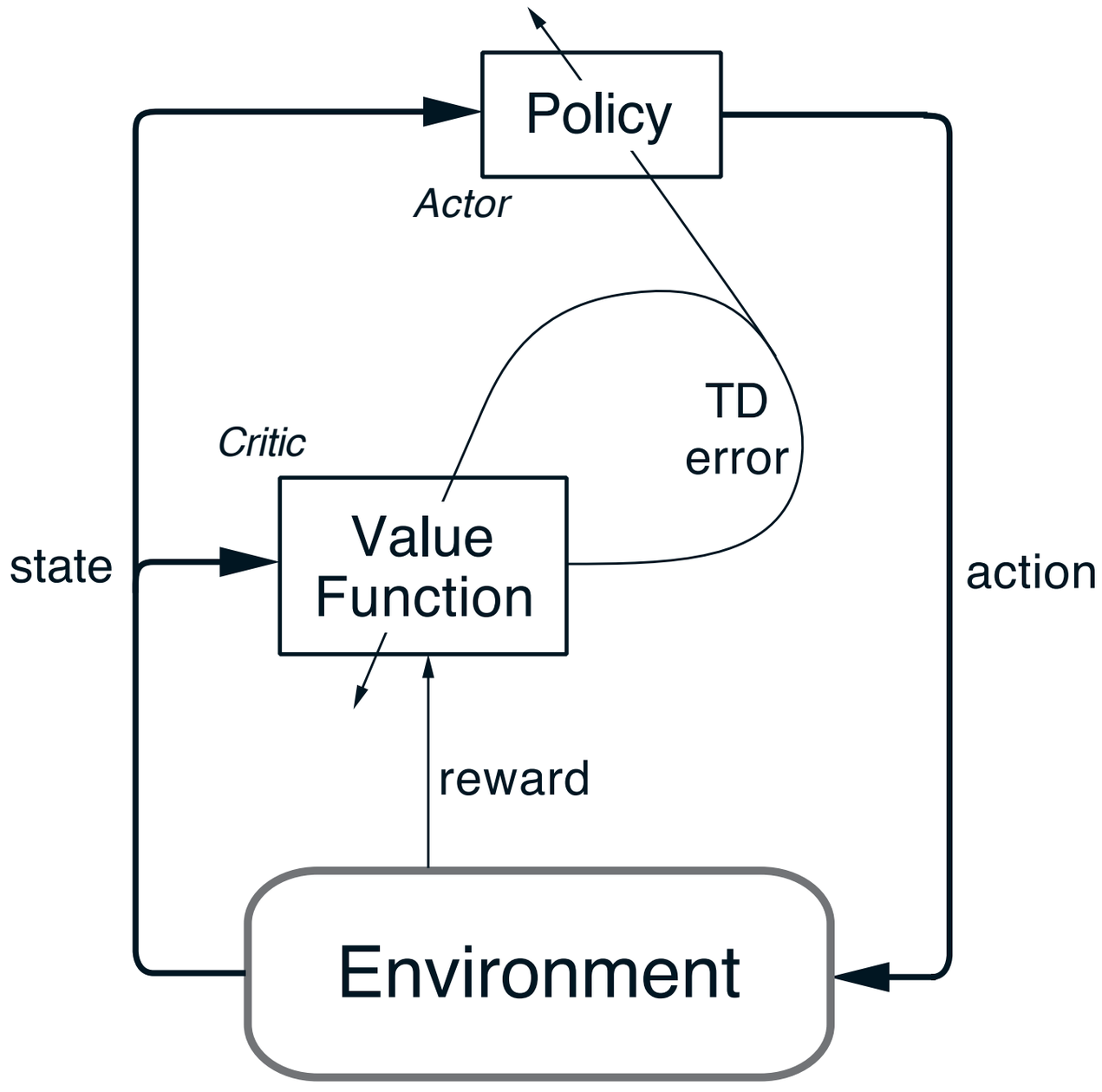
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
Initialize average reward estimate $\bar{R}$ arbitrarily (e.g., $\bar{R} = 0$)
Initialize state S, and action A

Repeat (for each step):
    Take action A, observe R, S'
    Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
    $\delta \leftarrow R - \bar{R} + \hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})$
    $\bar{R} \leftarrow \bar{R} + \beta\delta$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S, A, \mathbf{w})$
    $S \leftarrow S'$
    $A \leftarrow A'$

Actor Critic methods

- Many of the earliest TD methods were Actor-critic architectures (1977)
- AC methods learn the parameters of a policy **AND** a value function
- Two parts:
  - **actor:** implements the policy and improves the policy based on input from the critic
  - **critic**: learns v_π, and critiques the actions selected by the actor
- The critique is the TD-error for A_t
  - if positive, actor increases tendency to select A_t in the future
  - if negative, actor decreases tendency to select A_t in the future

Actor Critic Architecture



Tabular Actor Critic with traces

$$\delta_t \doteq R_{t+1} + \gamma V_t(S_{t+1}) - V(S_t),$$
$$H_{t+1}(s,a) \doteq H_t(s,a) + \beta\,\delta_t\, E_t(s,a)$$

- where H(s,a) are the modifiable policy parameters of the **actor: not a value function**
- V is the value function updated via tabular TD(λ)
- Actor and critic each have their own separate eligibility traces
- And actions are selected stochastically, via softmax:

$$\pi_t(a|s) \doteq \Pr\{A_t = a \mid S_t = s\} = \frac{e^{H_t(s,a)}}{\sum_b e^{H_t(s,b)}},$$

Gradient-bandit methods

- Recall chapter 2 talked about a gradient method for the bandit task
  - did not learn action-values
- Learned a set of action preferences: $H_t(a)$
  - encoded our preferences over actions
  - not directly related to value (in terms of reward)
- Algorithm derived by gradient of **average reward** objective
  - gradient ascent, increase average reward with action selections

Policy gradient methods

- Up to now we have focused on action-value methods
  - learning a value for each action
  - pick the max, or ϵ-greedy
- Another class of methods directly learns the policy
  - like gradient-bandit alg & Actor Critic
- Given a parametric representation of the policy
  - learn the parameters $\boldsymbol{\theta}$ of a stochastic policy
  - by following the gradient of a performance measure

Why would we do policy-gradient instead?

- Sometimes the policy is more simple to approximation than the value function

  - e.g., in Mountain Car the bang-bang policy is very simple and nearly optimal

- Action value methods want to find the best action in a state and select it the majority of the time

- This means that small changes in the value function can result in dramatic changes in the value function

  - Sarsa chatters

- In some problems the optimal policy is stochastic
  - e.g., do action A 33% of the time, B 33%, and C 33%
  - useful in domains like games—bluffing
- Avoid the max operation—search—on every time step
  - imagine hundreds of actions
- Better framework for dealing with continuous actions

Steps required to build a policy gradient method

1. Specify a parameterized policy: $\pi(a|s,\theta)$
2. Specify an objective function:
    1. we will use **average reward** $r(\pi)$. Discounted formulations also exist
3. Derive a stochastic gradient ascent algorithm
    1. compute the gradient of the objective: $r(\pi)$
    2. change in policy parameters proportional to gradient of objective with respect to policy parameters
    3. use the *policy gradient theorem* to help compute the gradient

How to compute the gradient: $\nabla r(\pi)$

- It is hard, but policy gradient theorem to the rescue:

policy-gradient theorem $$\nabla r(\pi) = \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a).$$

- Why is it useful?
- Because it is expressed in terms of q_π — which we know how to approximate, with Q_π
    - and the effect of the parameter changes on the action probabilities
- It is weighted by the on-policy distribution—states we see following π
    - thus we can approximate this expectation via stochastic sampling
    - averaging samples we see while following π, as we have done all along
- REINFORCE is a PG algorithm that uses samples of the return to estimate q_pi

Policy gradient AC family of methods

Initialize parameters of policy $\boldsymbol{\theta} \in \mathbb{R}^n$, and state-value function $\mathbf{w} \in \mathbb{R}^m$

Initialize eligibility traces $\mathbf{e}^{\boldsymbol{\theta}} \in \mathbb{R}^n$ and $\mathbf{e}^{\mathbf{w}} \in \mathbb{R}^m$ to $\mathbf{0}$

Initialize $\bar{R} = 0$

On each step, in state S:

Choose A according to $\pi(\cdot|S, \boldsymbol{\theta})$

Take action A, observe S', R

$\delta \leftarrow R - \bar{R} + \hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$ form TD error from critic

$\bar{R} \leftarrow \bar{R} + \beta\, \delta$ update average reward estimate

$\mathbf{e}^{\mathbf{w}} \leftarrow \lambda \mathbf{e}^{\mathbf{w}} + \nabla_{\mathbf{w}} \hat{v}(S, \mathbf{w})$ update eligibility trace for critic

$\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta\, \mathbf{e}^{\mathbf{w}}$ update critic parameters

$\mathbf{e}^{\boldsymbol{\theta}} \leftarrow \lambda \mathbf{e}^{\boldsymbol{\theta}} + \frac{\nabla \pi(A|S,\boldsymbol{\theta})}{\pi(A|S,\boldsymbol{\theta})}$ update eligibility trace for actor

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \delta\, \mathbf{e}^{\boldsymbol{\theta}}$ update actor parameters

- Eligibility trace denoted by ***e***

The details, two gradients to figure out!

- The gradient involved in the critic—update of the eligibility trace of the critic is easy:
  - with linear function approximation, we know the gradient of the value function with respect to the weights is just the feature vector

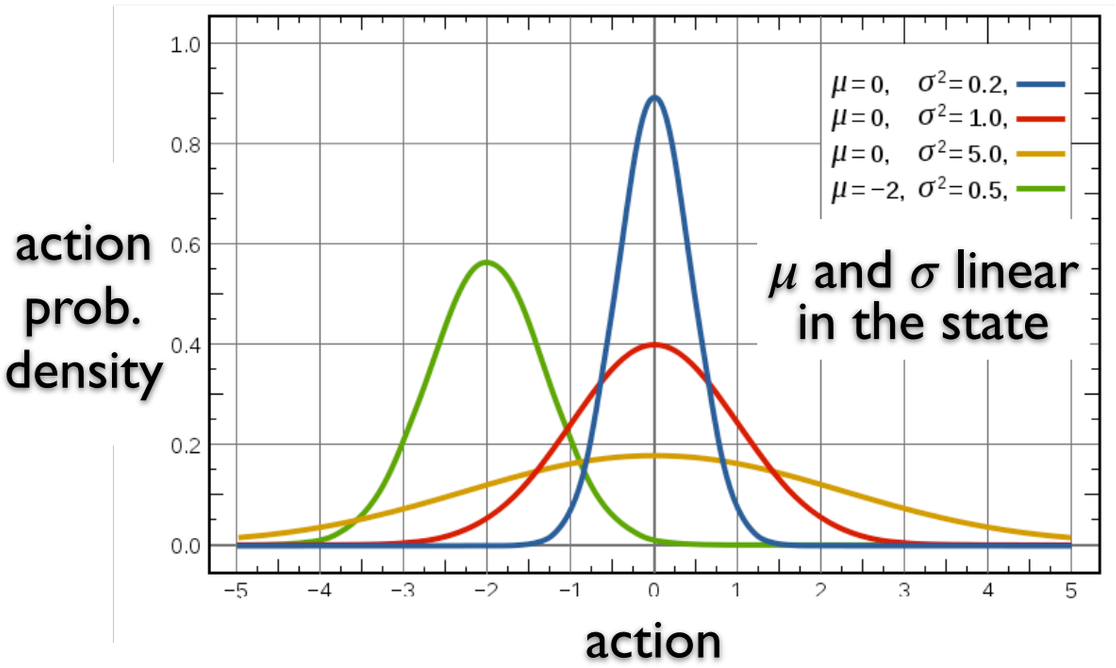
$$\nabla_{\mathbf{w}} \hat{v}(S, \mathbf{w}) = \phi(S)$$

- The gradient involved in the actor—update of the eligibility trace of the actor depends on the policy parameterization

- To select an action on each time step:
  - sample from a normal distribution

$$A_{t+1} \sim \mathcal{N}(\mu(s), \sigma(s))$$

- Both μ(s) and σ(s) change over time
- μ(s) converges to the expected continuous action in state
- σ(s) represents stochasticity of the action choice
  - initially high facilitating exploration
  - in settings requiring deterministic action choice, approaches zero



linear-gaussian policies & continuous actions

- The policy parameters define the mean and std. dev. of a gaussian distribution:

$$\mu(s, \boldsymbol{\theta}) \doteq \boldsymbol{\theta}_\mu{}^\top \mathbf{x}_\mu(s) \quad \text{and} \quad \sigma(s, \boldsymbol{\theta}) \doteq \exp\left(\boldsymbol{\theta}_\sigma{}^\top \mathbf{x}_\sigma(s)\right),$$

- Depending on the current state—encoded by $\boldsymbol{\phi}$, we get a different mean and std. dev.

- The policy is defined by this parameterized gaussian:

$$\pi(a|s, \boldsymbol{\theta}) \doteq \frac{1}{\sigma(s, \boldsymbol{\theta})\sqrt{2\pi}} \exp\left(-\frac{(a - \mu(s, \boldsymbol{\theta}))^2}{2\sigma(s, \boldsymbol{\theta})^2}\right)$$

Gradient of the linear-gaussian

$$\nabla \ln \pi(a|s, \boldsymbol{\theta}_\mu) = \frac{\nabla \pi(a|s, \boldsymbol{\theta}_\mu)}{\pi(a|s, \boldsymbol{\theta})} = \frac{1}{\sigma(s, \boldsymbol{\theta})^2} \big(a - \mu(s, \boldsymbol{\theta})\big) \mathbf{x}_\mu(s), \text{ and}$$

$$\nabla \ln \pi(a|s, \boldsymbol{\theta}_\sigma) = \frac{\nabla \pi(a|s, \boldsymbol{\theta}_\sigma)}{\pi(a|s, \boldsymbol{\theta})} = \left(\frac{\big(a - \mu(s, \boldsymbol{\theta})\big)^2}{\sigma(s, \boldsymbol{\theta})^2} - 1 \right) \mathbf{x}_\sigma(s).$$

Continuous-action, actor critic with linear FA and traces

On each step, in state S :

$A \sim \mathcal{N}(\mu(S), \sigma(S))$

Take action A ,observe S', R

$\delta \leftarrow R - \bar{R} + w^\top \phi(S') - w^\top \phi(S)$

$\bar{R} \leftarrow \bar{R} + \beta\delta$

$e_w \leftarrow \lambda e_w + \phi(S)$

$w \leftarrow w + \alpha_w \delta e_w$

$e_\mu \leftarrow \lambda e_\mu + \frac{1}{\sigma(S)^2}(A - \mu(s))\phi(S)$

$\theta_\mu \leftarrow \theta_\mu + \alpha_\mu \delta e_\mu$

$e_\sigma \leftarrow \lambda e_\sigma + \left[(A - \mu(s))^2/\sigma(s)^2 - 1\right]\phi(S)$

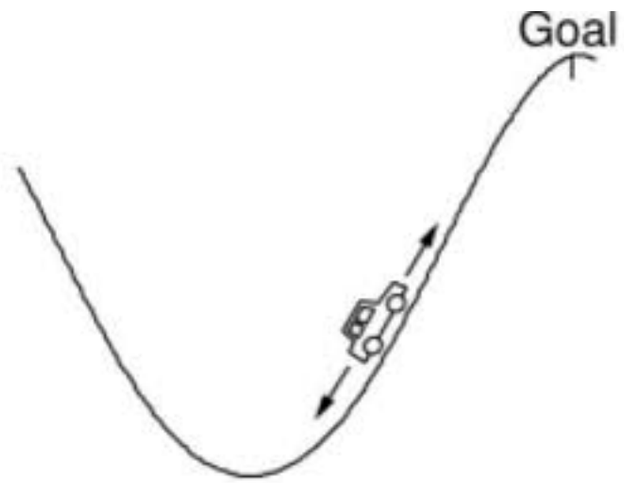
$\theta_\sigma \leftarrow \theta_\sigma + \alpha_\sigma \delta e_\sigma$

- Eligibility trace denoted by ***e***
- Feature vector denoted by $\boldsymbol{\phi}$

$$\mu(s) \stackrel{\text{def}}{=} \theta_\mu^\top \phi(s) \qquad \sigma(s) \stackrel{\text{def}}{=} \exp \theta_\sigma^\top \phi(s)$$

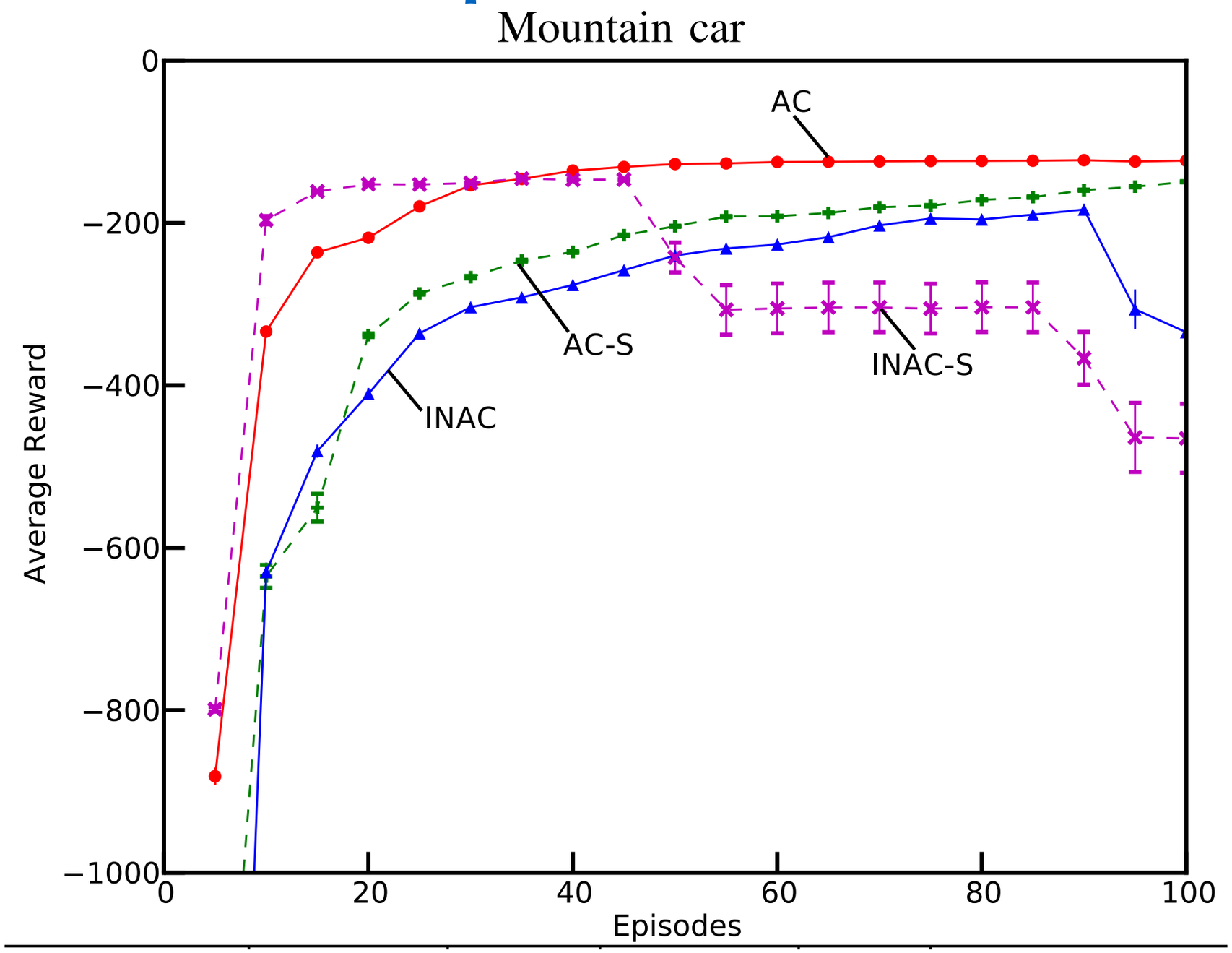
Policy gradient methods in action!

Continuous action mountain car

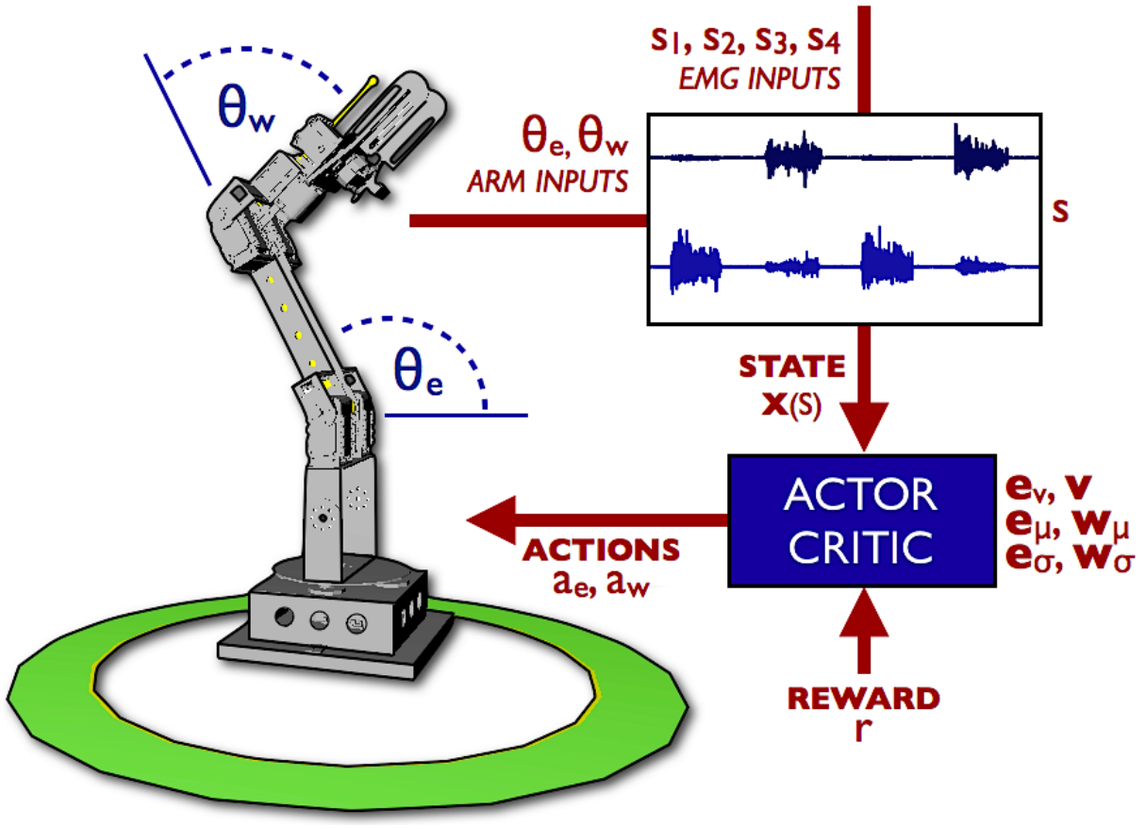


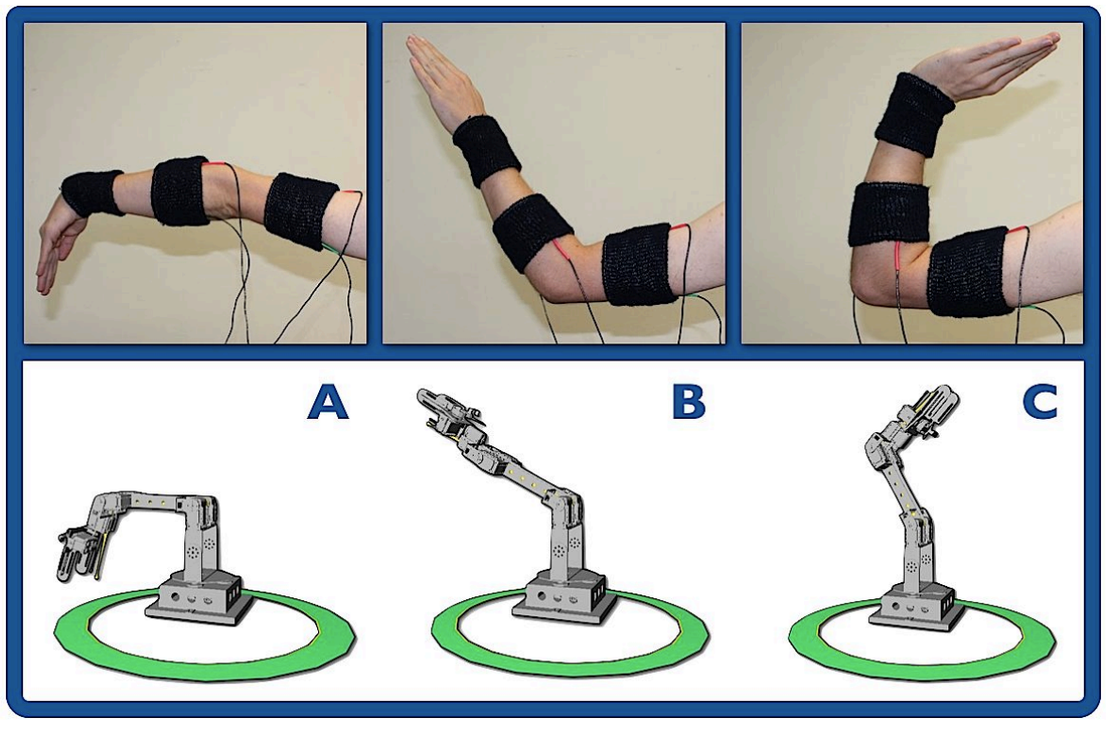
- Same as regular mountain car
  - Except that the action is a one-dim action in [-1,1]
  - episodes cut off after 5000 steps (no terminal state update)

Actor critic methods on MC (episodic)

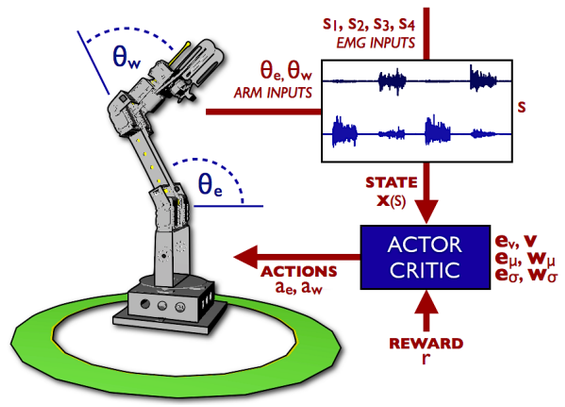


Policy gradient methods





Mii-Electric control of a robot arm



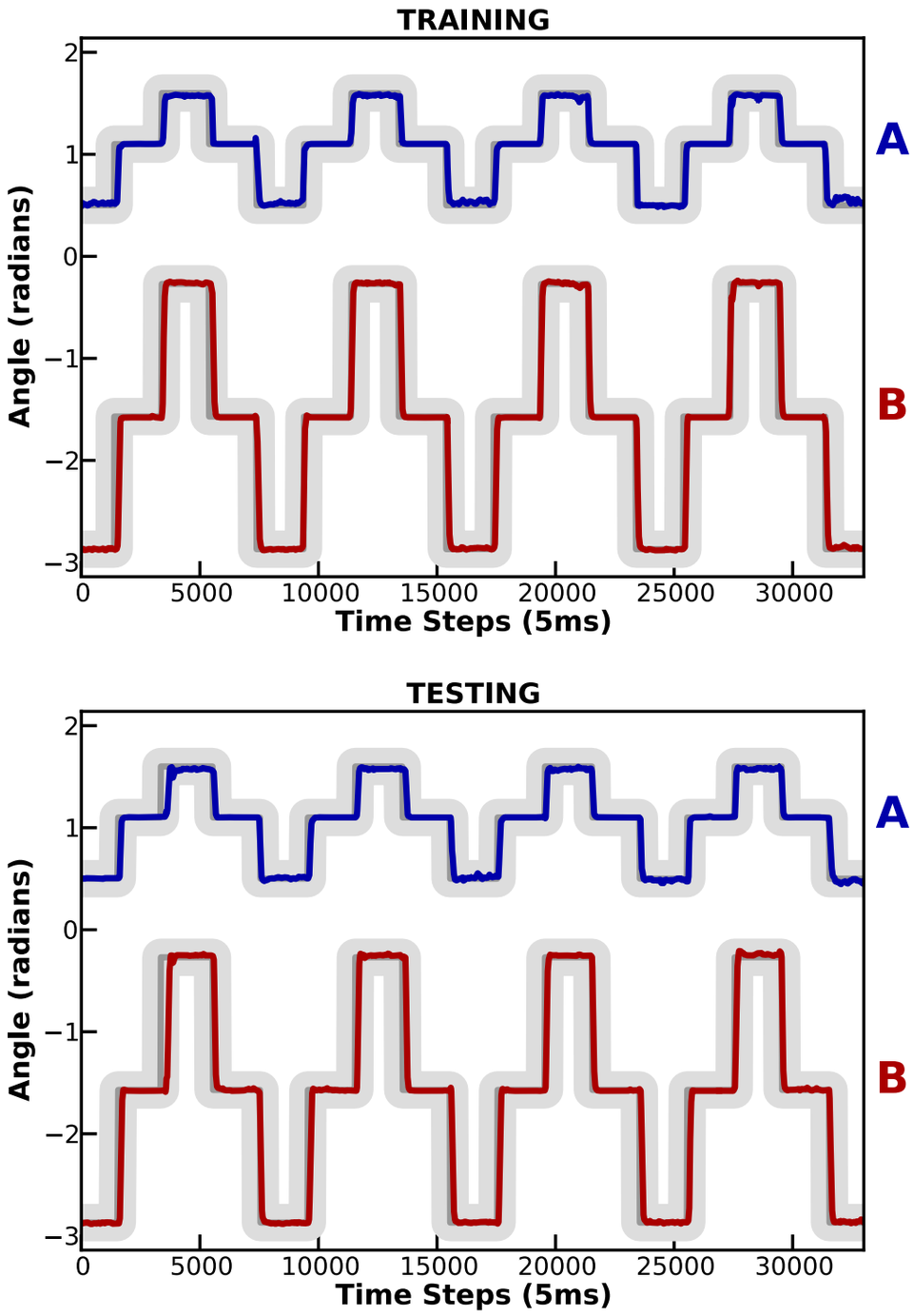
- 2D continuous action space:
  - wrist-joint angular velocity
  - elbow-joint angular velocity

- 4D state space: <wrist-angle, elbow-angle, two EMG signals computed from 4 muscle groups: bicep, tricep, wrist flexor, wrist extensors>
- Reward = 1.0, if two angles were with 0.1 rads of target, -0.5 on every other time step

Mii-Electric control of a robot arm

- States were converted to features via tile coding
  - four independent tilings, corresponding to partitions of [5,8,12,20]
    - each with num_tilings = 25
  - number of features = 4,636,425 and 100 active features
- Time step = 5ms
- Length of experiment 750k steps
- Number of independent runs = 1

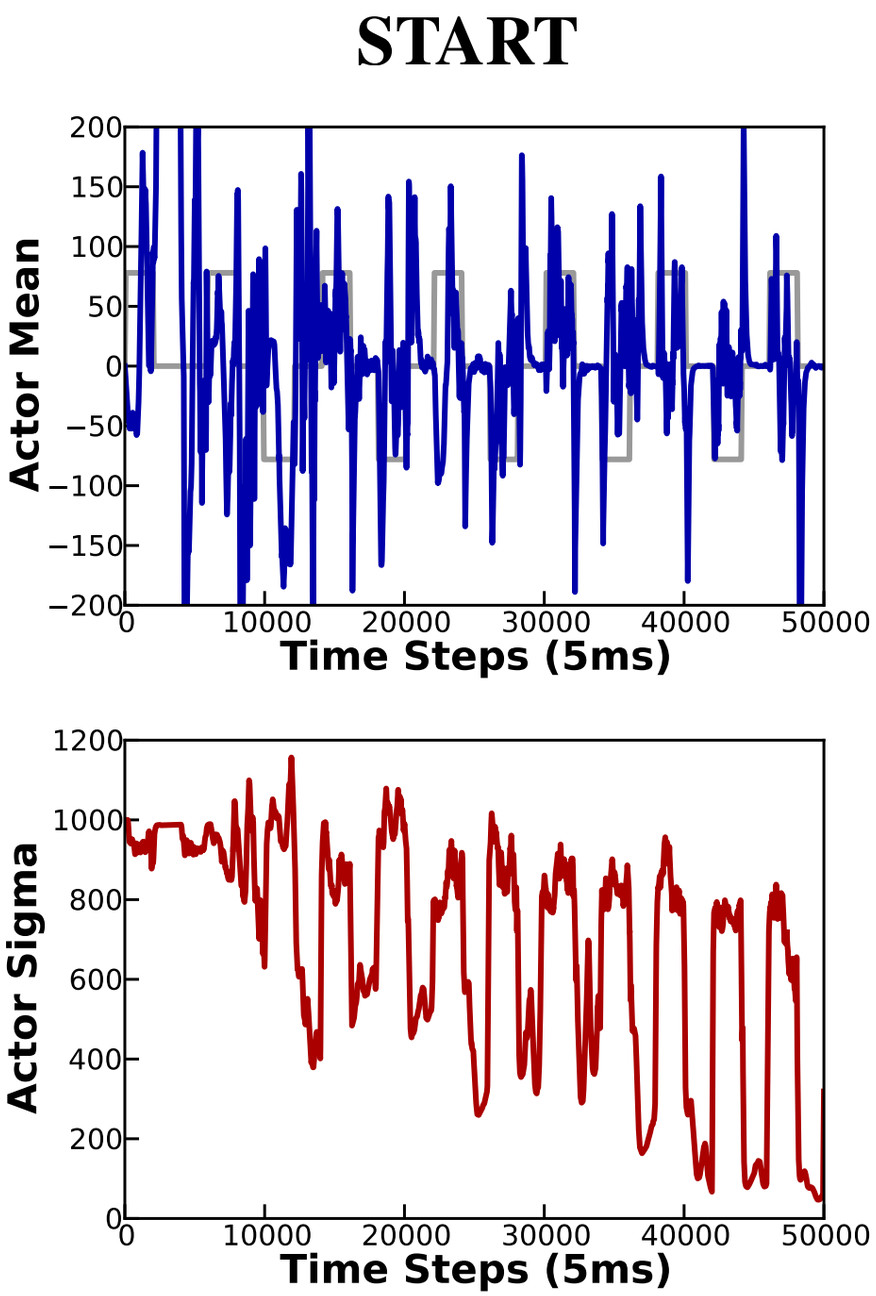
Training and final performance

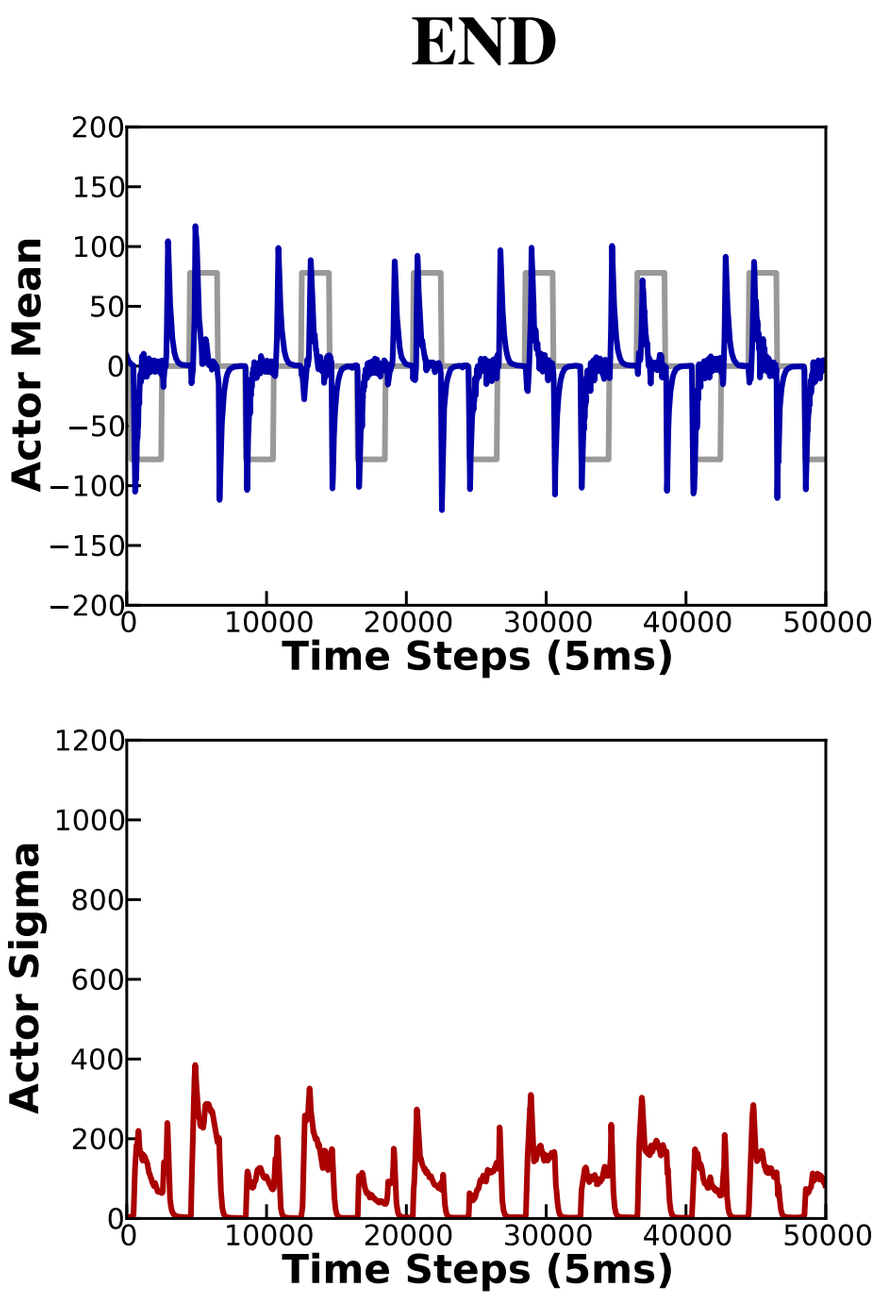


- A = elbow angle
- B = wrist angle

- within gray region indicates success

Convergence of policy parameters





References

- Policy gradient theorem (deviations)
  - http://incompleteideas.net/sutton/papers/SMSM-NIPS99.pdf
- Continuous action actor critic methods, and applications
  - https://link.springer.com/article/10.1007/BF00114727
  - http://incompleteideas.net/sutton/papers/DPS-ACC-12.pdf

Discussion posts

- Remember if I mark your discussion post with a ‘!’ And ask you a question I am asking you to improve it or clarify something

    - If you don’t, then it does not count

- If I tell you someone has already asked that question, please reply with another question

- Do people use average reward methods? Why not more?
  - See Sutton's pubs, but more generally Google Scholar
- Non-stationary settings an formal results …
  - Need to make assumptions. This book is not about this case
- Do we really want systematic exploration?
  - Depends

- Can’t we just avoid PG and pass the actions into the NN for continuous action cases?

- There is a strong interaction between representation learning (layers of the NN) and exploration

- What about discounting inside the agents (e.g., DQN or GVFs)?

- Paper on average reward Deep RL: https://arxiv.org/pdf/2106.07329.pdf

- Should we get rid of discounting all together?
- Are policy gradient methods difficult to use? If so why?
- If there an off-policy policy gradient theorem?
- What if some action dimensions are discrete and others continuous?
- What if actions are not valid in some states?

- Are tasks every really continuing?