Meta-Learning with Multi-Level Hierarchies via Context Variables

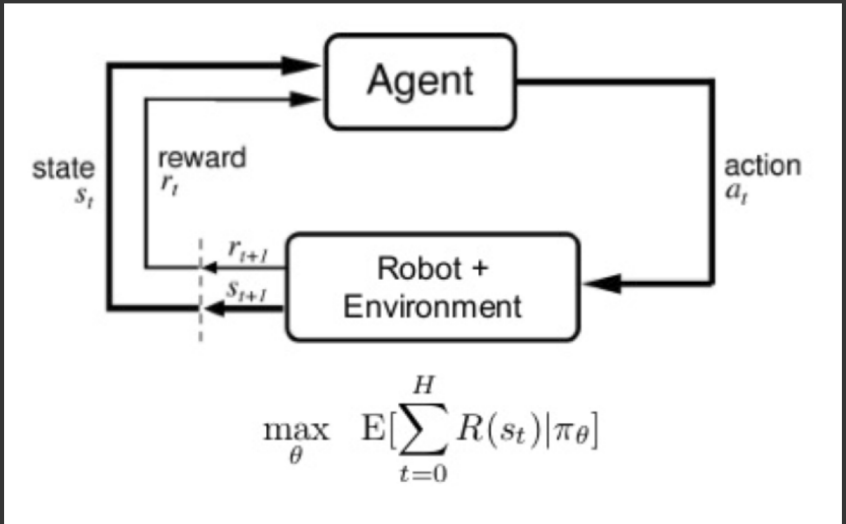
Willie McClinton[1], Andrew Levy[2], and George Konidaris[2]
[1]University of South Florida Department of Computer Science
[2]Brown University Department of Computer Science
Email: wmcclinton@mail.usf.edu


Boston Dyn


Boston Dyn

Deep Reinforcement Learning approaches, while good at learning complex functions, cannot (1) transfer knowledges to new experiences and (2) reason on large time scales in complex domains.

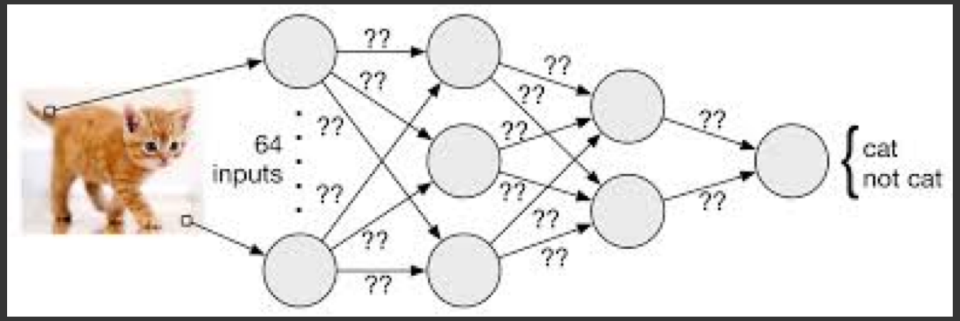
Reinforcement Learning

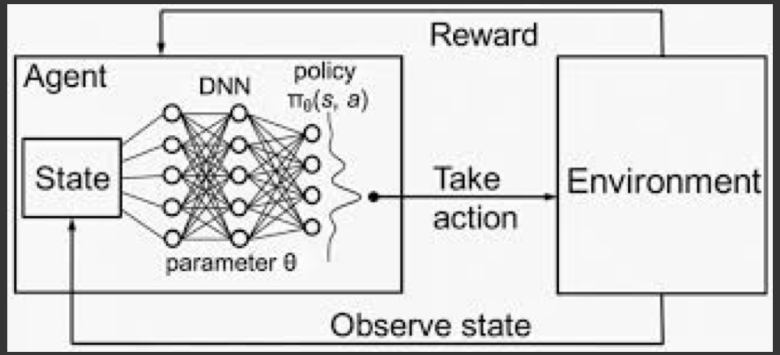


Note:
$\pi_\theta(s) = a$ is a parametrized policy for our agent

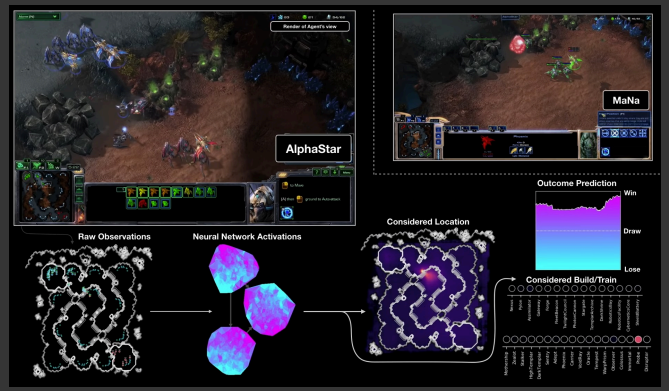
Sutton, R. and Barto, *A.* (2018) *Reinforcement Learning: An Introduction*

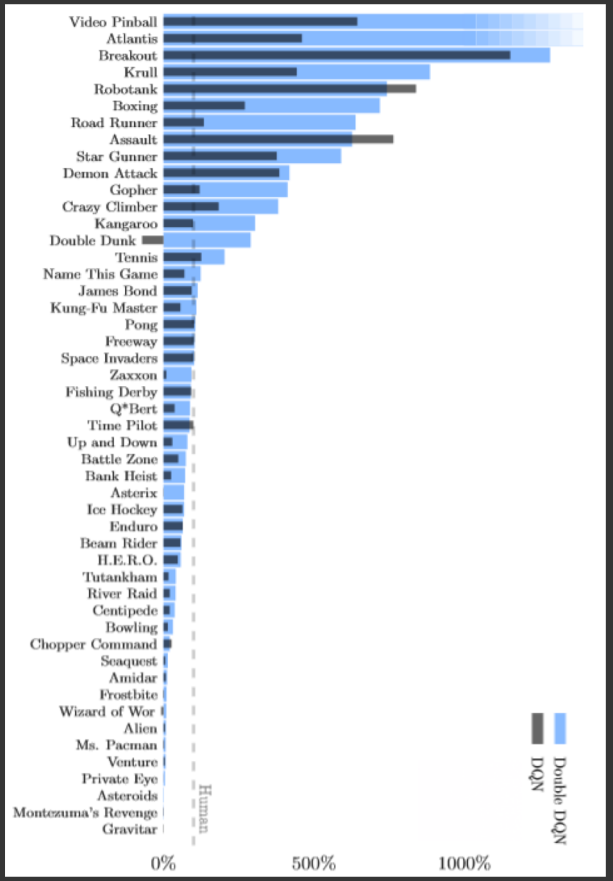
Deep RL


64
inputs
??
cat
not cat


Reward
Agent
DNN
policy
$\pi_\theta(s, a)$
State
parameter θ
Take
action
Environment
Observe state

DRL Successes





Schulman et al, 2015; Vinyals et al, 2019; Silver et al, Nature 2015; Mnih et al, NIPS 2013/ Nature 2015

DRL Faults

- Slow
  - Many of the Atari games took days to train when Humans can get fairly good score in a matter of hours
- Sample Inefficient
  - Most RL tasks run in simulation because the equivalent time it would take the in the real world would be infeasible
    - Months of training for Atari and Locomotion

Approaches

Meta Learning

- TRPO, DQN, A3C, .. are all general algorithms that can be used in any environment that can be mathematically described in an RL setting.
- Environments encountered in the real world are only a very small subset of all possible environments
- Meta Learning attempts to learn a prior using this in order to speed up learning in RL algorithms.

Meta Learning

- TRPO, DQN, A3C, .. are all general algorithms that can be used in any environment that can be mathematically described in an RL setting.
- Environments encountered in the real world are only a very small subset of all possible environments
- Meta Learning attempts to learn a prior using this in order to speed up learning in RL algorithms.

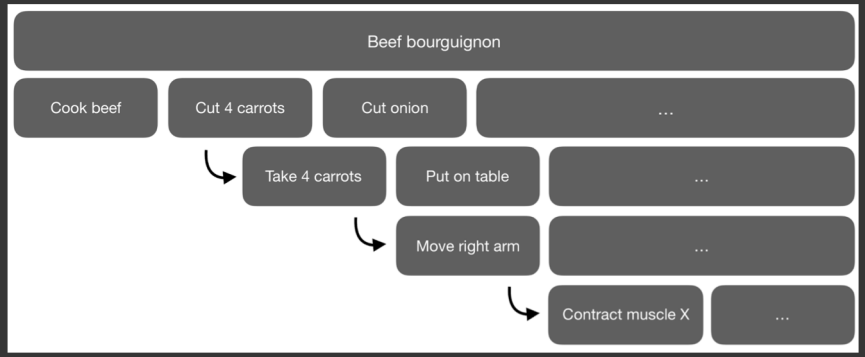
Meta Learning

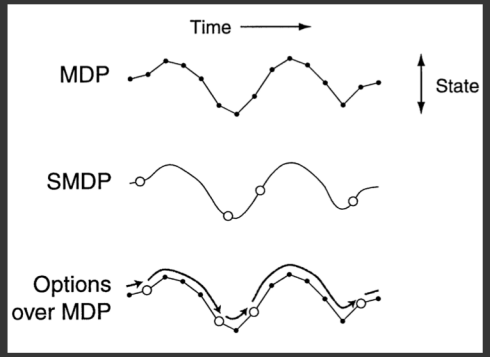
- TRPO, DQN, A3C, .. are all general algorithms that can be used in any environment that can be mathematically described in an RL setting.
- Environments encountered in the real world are only a very small subset of all possible environments
- Meta Learning attempts to learn a prior using this in order to speed up learning in RL algorithms.

Meta Learning

- TRPO, DQN, A3C, .. are all general algorithms that can be used in any environment that can be mathematically described in an RL setting.
- Environments encountered in the real world are only a very small subset of all possible environments
- Meta Learning attempts to learn a prior using this in order to speed up learning in RL algorithms.

Hierarchical RL





Feudal Learning

Options Framework

(Markov) option is a triple $o =< I_o, \pi_o, \beta_o >$ with:

- I_o: the initiation set
- $\pi_o : S \times A \rightarrow [0, 1]$: the option's policy
- $\beta_o : S \rightarrow [0, 1]$: the termination condition

Meta Learning

- No Training From Scratch
- Quick Adaptation
- Transfer learning

Hierarchical RL

- Long-term credit assignment
- Structured exploration
- Transfer learning

Can combining Meta RL and Hierarchical RL lead to the benefits of both in one framework?

Meta Learning Context Variable Z

Defining Context

$c^{0:t} = \{(s_i, a_i, r_i, s_{i+1}) : i \epsilon \{0,1,...t\}\}$ this represents the state transitions sampled from the environment during a task.

Meta Learning Context Variable Z

Defining Z Distribution

$q_\phi(\mu, \sigma|c)$ where $c = (s, a, r, s)$ is a transition in the environment and μ is the mean and σ is the std dev of Gaussian distribution representing the probability that transition is from a context $z \, \epsilon \, \mathbb{R}$

Meta Learning Context Variable Z

Defining Z Distribution

$q_\phi(\mu, \sigma | c^{0:t}) = \Gamma_{i=0}^{t} q(\mu, \sigma | c^i)$ the joint probability the trajectory is from a context $z \, \epsilon \, \mathbb{R}$

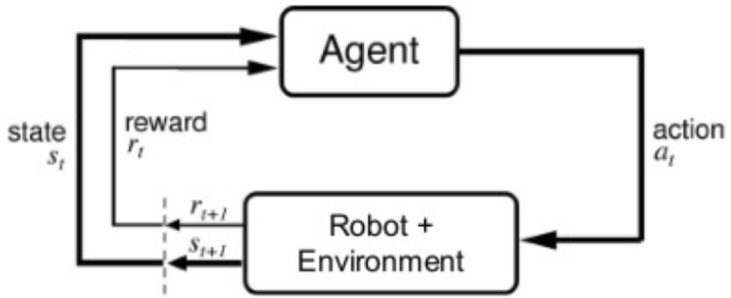
*Γ is a product of Gaussians and can be thought of as updating the prior given data to get a posterior

Meta Learning Context Variable Z

Sampling Z

$z \sim q_\phi(\mu, \sigma | c^{0:t})$ the context variable z is sampled from a Gaussian distribution computer over the context c

Hierarchical Formulation



$$\max_{\theta} \quad \mathrm{E}[\sum_{t=0}^{H} R(s_t)|\pi_\theta]$$

$$\pi_0 : S \text{ x } G \rightarrow A \text{ or } \pi_0(s_t, g_{0_t}) = a_t$$

at each time step $\pi_0(s_t, g_{0_t}) = a_t$

$$\pi_i : S \text{ x } G \rightarrow G \text{ or } \pi_i(s_t, g_{i_t}) = g_{i-1_t}$$

at every k^i time steps $\pi_i(s_t, g_{i_t}) = g_{i-1_t}$

$$\pi_h : S \text{ x } C \rightarrow G \text{ or } \pi_h(s_t, z) = g_{h-1_t}$$

at every k^h time steps $\pi_h(s_t, z) = g_{h-1_t}$

*Imagine z represents latent variable of hidden goal

Algorithm

Algorithm 1: PEACH Meta-training

Bath of training tasks $\{\mathcal{T}_i\}_{i=1...T}$ from $p(\mathcal{T})$
learning rates = $\alpha_1, \alpha_2, \alpha_3$
Initialize replay Buffers $\mathcal{B}^i$ for each training task
while *not done* **do**
  for *each* T_i **do**
    Initialize context $c^i = \{\}$
    for l = *1,...,L* **do**
      Sample $z \sim q_\phi(z|c^i)$
      TRAIN-LEVEL(k-1, s, z) and add every $\{(s, a, s', r)\}$ *to* $\mathcal{B}^i$
      Update lower actor and critic networks $\theta_{\pi_{H-1}}, ..., \theta_{\pi_0}$ *and* $\theta_{Q_{H-1}}, ..., \theta_{Q_0}$
      Update $c^i = \{(s_j, a_j, s'_j, r_j)\}_{j:1...\mathcal{N}} \sim \mathcal{B}^i$
    end
  end
  for *each gradient step* **do**
    for *each* T_i **do**
      Sample context $c^i \sim S_c(B^i)$ *and RL batch* $b^i \sim B^i$
      Sample $z \sim q_\phi(z|c^i)$
      $\mathcal{L}^i{}_{actor} = \mathcal{L}_{actor}(b^i, z)$
      $\mathcal{L}^i{}_{critic} = \mathcal{L}_{critic}(b^i, z)$
      $\mathcal{L}^i{}_{KL} = \beta D_{KL}(q(z|c^i)||r(z))$
    end
    Update context variable encoder:
    $\phi \rightarrow \phi - \alpha_1 \nabla_\phi \Sigma_i (\mathcal{L}^i{}_{critic} + \mathcal{L}^i{}_{KL})$
    Update top actor critic:
    $\theta_{\pi_H} \rightarrow \theta_{\pi_H} - \alpha_2 \nabla_\theta \Sigma_i \mathcal{L}^i{}_{actor}$
    $\theta_{Q_H} \rightarrow \theta_{Q_H} - \alpha_3 \nabla_\theta \Sigma_i \mathcal{L}^i{}_{critic}$
  end
end

Algorithm

Gathering Data + Training Hierarchies

Algorithm 1: PEACH Meta-training

Bath of training tasks $\{\mathcal{T}_i\}_{i=1...T}$ from $p(\mathcal{T})$
learning rates = $\alpha_1, \alpha_2, \alpha_3$
Initialize replay Buffers $\mathcal{B}^i$ for each training task
while *not done* **do**
  for *each* T_i **do**
    Initialize context $c^i = \{\}$
    for l = *1,...,L* **do**
      Sample $z \sim q_\phi(z|c^i)$
      TRAIN-LEVEL(k-1, s, z) and add every $\{(s, a, s', r)\}$ *to* $\mathcal{B}^i$
      Update lower actor and critic networks $\theta_{\pi_{H-1}}, ..., \theta_{\pi_0}$ *and* $\theta_{Q_{H-1}}, ..., \theta_{Q_0}$
      Update $c^i = \{(s_j, a_j, s'_j, r_j)\}_{j:1...\mathcal{N}} \sim \mathcal{B}^i$
    end
  end
  for *each gradient step* **do**
    for *each* T_i **do**
      Sample context $c^i \sim S_c(B^i)$ *and RL batch* $b^i \sim B^i$
      Sample $z \sim q_\phi(z|c^i)$
      $\mathcal{L}^i{}_{actor} = \mathcal{L}_{actor}(b^i, z)$
      $\mathcal{L}^i{}_{critic} = \mathcal{L}_{critic}(b^i, z)$
      $\mathcal{L}^i{}_{KL} = \beta D_{KL}(q(z|c^i)||r(z))$
    end
    Update context variable encoder:
    $\phi \rightarrow \phi - \alpha_1 \nabla_\phi \Sigma_i (\mathcal{L}^i{}_{critic} + \mathcal{L}^i{}_{KL})$
    Update top actor critic:
    $\theta_{\pi_H} \rightarrow \theta_{\pi_H} - \alpha_2 \nabla_\theta \Sigma_i \mathcal{L}^i{}_{actor}$
    $\theta_{Q_H} \rightarrow \theta_{Q_H} - \alpha_3 \nabla_\theta \Sigma_i \mathcal{L}^i{}_{critic}$
  end
end

Algorithm

Computing Meta-Training Loss

Algorithm 1: PEACH Meta-training

Bath of training tasks $\{\mathcal{T}_i\}_{i=1...T}$ from $p(\mathcal{T})$
learning rates = $\alpha_1, \alpha_2, \alpha_3$
Initialize replay Buffers $\mathcal{B}^i$ for each training task
while *not done* **do**
  for *each* T_i **do**
    Initialize context $c^i = \{\}$
    for l = *1,...,L* **do**
      Sample $z \sim q_\phi(z|c^i)$
      TRAIN-LEVEL(k-1, s, z) and add every $\{(s, a, s', r)\}$ *to* $\mathcal{B}^i$
      Update lower actor and critic networks $\theta_{\pi_{H-1}}, ..., \theta_{\pi_0}$ *and* $\theta_{Q_{H-1}}, ..., \theta_{Q_0}$
      Update $c^i = \{(s_j, a_j, s'_j, r_j)\}_{j:1...\mathcal{N}} \sim \mathcal{B}^i$
    end
  end
  for *each gradient step* **do**
    for *each* T_i **do**
      Sample context $c^i \sim S_c(B^i)$ *and RL batch* $b^i \sim B^i$
      Sample $z \sim q_\phi(z|c^i)$
      $\mathcal{L}^i{}_{actor} = \mathcal{L}_{actor}(b^i, z)$
      $\mathcal{L}^i{}_{critic} = \mathcal{L}_{critic}(b^i, z)$
      $\mathcal{L}^i{}_{KL} = \beta D_{KL}(q(z|c^i)||r(z))$
    end
    Update context variable encoder:
    $\phi \rightarrow \phi - \alpha_1 \nabla_\phi \Sigma_i (\mathcal{L}^i{}_{critic} + \mathcal{L}^i{}_{KL})$
    Update top actor critic:
    $\theta_{\pi_H} \rightarrow \theta_{\pi_H} - \alpha_2 \nabla_\theta \Sigma_i \mathcal{L}^i{}_{actor}$
    $\theta_{Q_H} \rightarrow \theta_{Q_H} - \alpha_3 \nabla_\theta \Sigma_i \mathcal{L}^i{}_{critic}$
  end
end

Algorithm

Gradient Step For Top Level

Algorithm 1: PEACH Meta-training

Bath of training tasks $\{\mathcal{T}_i\}_{i=1...T}$ from $p(\mathcal{T})$
learning rates = $\alpha_1, \alpha_2, \alpha_3$
Initialize replay Buffers $\mathcal{B}^i$ for each training task
while *not done* **do**
  for *each* T_i **do**
    Initialize context $c^i = \{\}$
    for l = *1,...,L* **do**
      Sample $z \sim q_\phi(z|c^i)$
      TRAIN-LEVEL(k-1, s, z) and add every $\{(s, a, s', r)\}$ *to* $\mathcal{B}^i$
      Update lower actor and critic networks $\theta_{\pi_{H-1}}, ..., \theta_{\pi_0}$ *and* $\theta_{Q_{H-1}}, ..., \theta_{Q_0}$
      Update $c^i = \{(s_j, a_j, s'_j, r_j)\}_{j:1...\mathcal{N}} \sim \mathcal{B}^i$
    end
  end
  for *each gradient step* **do**
    for *each* T_i **do**
      Sample context $c^i \sim S_c(B^i)$ *and RL batch* $b^i \sim B^i$
      Sample $z \sim q_\phi(z|c^i)$
      $\mathcal{L}^i{}_{actor} = \mathcal{L}_{actor}(b^i, z)$
      $\mathcal{L}^i{}_{critic} = \mathcal{L}_{critic}(b^i, z)$
      $\mathcal{L}^i{}_{KL} = \beta D_{KL}(q(z|c^i)||r(z))$
    end
    Update context variable encoder:
    $\phi \rightarrow \phi - \alpha_1 \nabla_\phi \Sigma_i (\mathcal{L}^i{}_{critic} + \mathcal{L}^i{}_{KL})$
    Update top actor critic:
    $\theta_{\pi_H} \rightarrow \theta_{\pi_H} - \alpha_2 \nabla_\theta \Sigma_i \mathcal{L}^i{}_{actor}$
    $\theta_{Q_H} \rightarrow \theta_{Q_H} - \alpha_3 \nabla_\theta \Sigma_i \mathcal{L}^i{}_{critic}$
  end
end

Current Work

- Testing Framework on Mujoco environments
- Understanding the limitations of goal based reward functions
- Finding the sample efficiency of using context variables to differentiate task

Acknowledgements

- I would like to thank Brown University and Dr. George Konidaris for funding this research with the Brown CS Startup Fund, as well as, Leadership Alliance for accommodating me during the Summer Program.

Questions?