Interpretable Hierarchical Reinforcement Learning

Divyat Mahajan
divyatm@iitk.ac.in

Harsh Sinha
harshsin@iitk.ac.in

Indian Institute of Technology Kanpur
Mentor: Prof. Vinay Namboodiri

Hierarchical Reinforcement Learning

- One of the major issues with RL is training from scratch for every task

Hierarchical Reinforcement Learning

- One of the major issues with RL is training from scratch for every task
- In real world, many many tasks have sub-parts which may be reused for other tasks.

Hierarchical Reinforcement Learning

- One of the major issues with RL is training from scratch for every task
- In real world, many many tasks have sub-parts which may be reused for other tasks.
- Hierarchical Learning intends to solve this by learning different sub-parts independently and having a master choose the sub-policy to be activated.

Problem Statement

- Design a model for reinforcement learning for multi goal environment that trains on few goals and generalizes to different unseen goals

Problem Statement

- Design a model for reinforcement learning for multi goal environment that trains on few goals and generalizes to different unseen goals
- We propose solution based on hierarchical learning methods that include a goal specific master policy and generic sub policies

Problem Statement

- Design a model for reinforcement learning for multi goal environment that trains on few goals and generalizes to different unseen goals
- We propose solution based on hierarchical learning methods that include a goal specific master policy and generic sub policies
- We also maintain interpretability in regards to the sub-policies activated for different tasks/goals

Info RL

- Standard reinforcement learning algorithms typically learn an optimal way of solving the task but they are not able to figure out multiple near optimal ways of solving a task

Info RL

- Standard reinforcement learning algorithms typically learn an optimal way of solving the task but they are not able to figure out multiple near optimal ways of solving a task
- InfoRL disentangles the multiple near optimal ways of solving a task by maximising the mutual information between sampled latent code and the policy output

Info RL

- Standard reinforcement learning algorithms typically learn an optimal way of solving the task but they are not able to figure out multiple near optimal ways of solving a task
- InfoRL disentangles the multiple near optimal ways of solving a task by maximising the mutual information between sampled latent code and the policy output
- Maximising the mutual information leads to a latent code corresponding to a specific policy and maximisation of the environment reward ensures the policies learnt are near optimal

Info RL

- Standard reinforcement learning algorithms typically learn an optimal way of solving the task but they are not able to figure out multiple near optimal ways of solving a task
- InfoRL disentangles the multiple near optimal ways of solving a task by maximising the mutual information between sampled latent code and the policy output
- Maximising the mutual information leads to a latent code corresponding to a specific policy and maximisation of the environment reward ensures the policies learnt are near optimal
- This approach requires the environment to be complex enough to have multiple near optimal ways of solving a task

Meta Learning Shared Hierarchies

- The paper presents an approach for learning hierarchically structured policies with the aim of generalising quickly over unseen tasks

Meta Learning Shared Hierarchies

- The paper presents an approach for learning hierarchically structured policies with the aim of generalising quickly over unseen tasks
- The approach consists of task specific master policy with parameters θ and a set of shared parameters ϕ s.t. we have k subsets of these shared parameters where each ϕ_k denotes a sub-policy $\pi_{\phi_k}(a|s)$

Meta Learning Shared Hierarchies

- The paper presents an approach for learning hierarchically structured policies with the aim of generalising quickly over unseen tasks
- The approach consists of task specific master policy with parameters θ and a set of shared parameters ϕ s.t. we have k subsets of these shared parameters where each ϕ_k denotes a sub-policy $\pi_{\phi_k}(a|s)$
- The master network can be seen as essentially choosing the order of sub policies to activated for the given task

Meta Learning Shared Hierarchies

- The paper presents an approach for learning hierarchically structured policies with the aim of generalising quickly over unseen tasks
- The approach consists of task specific master policy with parameters θ and a set of shared parameters ϕ s.t. we have k subsets of these shared parameters where each ϕ_k denotes a sub-policy $\pi_{\phi_k}(a|s)$
- The master network can be seen as essentially choosing the order of sub policies to activated for the given task
- The update scheme is designed to ensure that we learn sub-policies that are optimal for the task we trained on and also generalise to new tasks:

Meta Learning Shared Hierarchies

- The paper presents an approach for learning hierarchically structured policies with the aim of generalising quickly over unseen tasks
- The approach consists of task specific master policy with parameters θ and a set of shared parameters ϕ s.t. we have k subsets of these shared parameters where each ϕ_k denotes a sub-policy $\pi_{\phi_k}(a|s)$
- The master network can be seen as essentially choosing the order of sub policies to activated for the given task
- The update scheme is designed to ensure that we learn sub-policies that are optimal for the task we trained on and also generalise to new tasks:
  - Warmup period: Given the set of shared sub policies ϕ_k, it learns an optimal θ

Meta Learning Shared Hierarchies

- The paper presents an approach for learning hierarchically structured policies with the aim of generalising quickly over unseen tasks
- The approach consists of task specific master policy with parameters θ and a set of shared parameters ϕ s.t. we have k subsets of these shared parameters where each ϕ_k denotes a sub-policy $\pi_{\phi_k}(a|s)$
- The master network can be seen as essentially choosing the order of sub policies to activated for the given task
- The update scheme is designed to ensure that we learn sub-policies that are optimal for the task we trained on and also generalise to new tasks:
  - Warmup period: Given the set of shared sub policies ϕ_k, it learns an optimal θ
  - Joint Update: Optimise both the shared and task specific parameters to obtain optimal values for shared sub policies ϕ_k

Meta Learning Shared Hierarchies

- The warmup period is important since we should update the shared sub policies only when the task specific parameters θ are near their optimal value

Meta Learning Shared Hierarchies

- The warmup period is important since we should update the shared sub policies only when the task specific parameters θ are near their optimal value
- The argument to generalise over unseen tasks is based on the assumption that the warmup period will learn the near optimal values of θ for the fixed set of sub policies.

Meta Learning Shared Hierarchies

- The warmup period is important since we should update the shared sub policies only when the task specific parameters θ are near their optimal value
- The argument to generalise over unseen tasks is based on the assumption that the warmup period will learn the near optimal values of θ for the fixed set of sub policies.
- After learning a set of optimal sub policies, the model can generalise to new unseen tasks by using only warmup period updates

Meta Learning Shared Hierarchies

Algorithm 1 Meta Learning Shared Hierarchies

Initialize ϕ
repeat
  Initialize θ
  Sample task $M \sim P_M$
  for $w = 0, 1, ...W$ (warmup period) **do**
    Collect D timesteps of experience using $\pi_{\phi,\theta}$
    Update θ to maximize expected return from $1/N$ timescale viewpoint
  end for
  for $u = 0, 1, ...U$ (joint update period) **do**
    Collect D timesteps of experience using $\pi_{\phi,\theta}$
    Update θ to maximize expected return from $1/N$ timescale viewpoint
    Update ϕ to maximize expected return from full timescale viewpoint
  end for
until convergence

Universal Value Function Approximators

- The paper proposes value functions $V(s, g|\theta)$ where (s, g) are the states, goal pairs and θ represent the parameters which can generalize on states and goal alike

Universal Value Function Approximators

- The paper proposes value functions $V(s, g|\theta)$ where (s, g) are the states, goal pairs and θ represent the parameters which can generalize on states and goal alike
- Universal value function approximator (UVFA) can generalise to unseen goals by exploiting the structure of the goal space and the state space

Universal Value Function Approximators

- The paper proposes value functions $V(s, g|\theta)$ where (s, g) are the states, goal pairs and θ represent the parameters which can generalize on states and goal alike
- Universal value function approximator (UVFA) can generalise to unseen goals by exploiting the structure of the goal space and the state space
- Supervised learning of UVFA is difficult since an agent would only see a fraction of the state space and the goal space during training phase

Universal Value Function Approximators

- The paper proposes value functions $V(s, g|\theta)$ where (s, g) are the states, goal pairs and θ represent the parameters which can generalize on states and goal alike
- Universal value function approximator (UVFA) can generalise to unseen goals by exploiting the structure of the goal space and the state space
- Supervised learning of UVFA is difficult since an agent would only see a fraction of the state space and the goal space during training phase
- They use matrix factorization combined with regression for effective learning of UVFA

Universal Value Function Approximators

- The paper proposes value functions $V(s, g|\theta)$ where (s, g) are the states, goal pairs and θ represent the parameters which can generalize on states and goal alike
- Universal value function approximator (UVFA) can generalise to unseen goals by exploiting the structure of the goal space and the state space
- Supervised learning of UVFA is difficult since an agent would only see a fraction of the state space and the goal space during training phase
- They use matrix factorization combined with regression for effective learning of UVFA
  - Obtain embedding vectors $\hat{\phi(s)}$ and $\hat{\psi(g)}$ by performing matrix factorization of the sparsely filled value function table $V_g(s)$

Universal Value Function Approximators

- The paper proposes value functions $V(s, g|\theta)$ where (s, g) are the states, goal pairs and θ represent the parameters which can generalize on states and goal alike
- Universal value function approximator (UVFA) can generalise to unseen goals by exploiting the structure of the goal space and the state space
- Supervised learning of UVFA is difficult since an agent would only see a fraction of the state space and the goal space during training phase
- They use matrix factorization combined with regression for effective learning of UVFA
  - Obtain embedding vectors $\phi(\hat{s})$ and $\psi(\hat{g})$ by performing matrix factorization of the sparsely filled value function table $V_g(s)$
  - Solve the regression problems: Map the state/goal to its target embedding i.e. state s to $\phi(\hat{s})$ and the goal g to $\psi(\hat{g})$.

Architecture

- The master/planner network predicts the latent code given state and goal as input

Architecture

- The master/planner network predicts the latent code given state and goal as input
- The policy network takes the generated latent code along with the current state as input and predicts the action that agent must take

Architecture

- The master/planner network predicts the latent code given state and goal as input
- The policy network takes the generated latent code along with the current state as input and predicts the action that agent must take
- The posterior network reconstructs the latent code given state and action as input

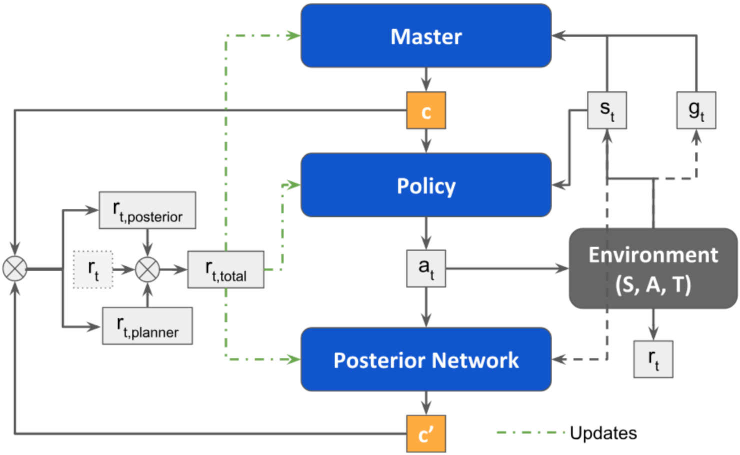
Architecture

- The master/planner network predicts the latent code given state and goal as input
- The policy network takes the generated latent code along with the current state as input and predicts the action that agent must take
- The posterior network reconstructs the latent code given state and action as input
- Mutual Information Maximisation is approximate via the reconstruction loss due to the Posterior Network

Architecture

- The master/planner network predicts the latent code given state and goal as input
- The policy network takes the generated latent code along with the current state as input and predicts the action that agent must take
- The posterior network reconstructs the latent code given state and action as input
- Mutual Information Maximisation is approximate via the reconstruction loss due to the Posterior Network
- An additional reward is included: Planner Reward that assists the master network to learn a simpler relation between the goals and the latent code

Architecture



Architecture

Algorithm 1: Interpretable Hierarchical Reinforcement Learning

initialize master(ϕ_m), policy(π_θ), and posterior(ϕ_p);
repeat
- sample goal $g \sim \mathbb{G}_T$;
- $s \sim S_0$;
- c $\sim$ master_network(s, g);
- **repeat**
  - sample trajectories $\tau, s, r \sim \pi_\theta(c, s)$;
  - sample state action pairs $\xi \sim \tau$;
  - $c' \sim \phi_p(\xi)$;
  - $r = r + posterior\_reward(c, c')$;
  - update_policy(r);
  - update_master(r);
  - update_posterior(c, c');
- **until** *convergence*;

until *N times*;

Experiments

- The master network and the policy network predicts are modelled using Q-Tables

Experiments

- The master network and the policy network predicts are modelled using Q-Tables
- The posterior network is modelled using Neural Networks

Experiments

- The master network and the policy network predicts are modelled using Q-Tables
- The posterior network is modelled using Neural Networks
- The master policy is trained at fixed intervals while we alternate between the updates of the policy network and the posterior network

Experiments

- The master network and the policy network predicts are modelled using Q-Tables
- The posterior network is modelled using Neural Networks
- The master policy is trained at fixed intervals while we alternate between the updates of the policy network and the posterior network
- Planner reward is added that penalises the master network for too much variance in the set of latent codes generated for the whole episode

Experiments

- The master network and the policy network predicts are modelled using Q-Tables
- The posterior network is modelled using Neural Networks
- The master policy is trained at fixed intervals while we alternate between the updates of the policy network and the posterior network
- Planner reward is added that penalises the master network for too much variance in the set of latent codes generated for the whole episode
- Both master and the policy networks choose actions at the same frequency

Environment

- 10x10 grid environment, with chooseble multiple goals.

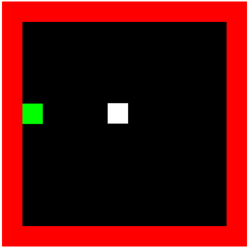


Figure: Grid Env
Green – Goal, White – Initialization

Environment

- 10x10 grid environment, with chooseble multiple goals.
- 4 actions (U, D, L, R) allowed.

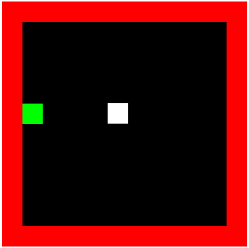


Figure: Grid Env
Green – Goal, White – Initialization

Environment

- 10x10 grid environment, with chooseble multiple goals.
- 4 actions (U, D, L, R) allowed.
- We train on 8 (or 4) goals only and test on all locations, with fine-tuning.

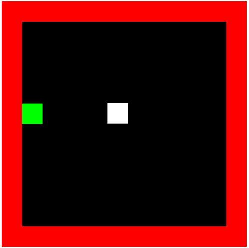


Figure: Grid Env
Green – Goal, White – Initialization

Environment

- 10x10 grid environment, with chooseble multiple goals.
- 4 actions (U, D, L, R) allowed.
- We train on 8 (or 4) goals only and test on all locations, with fine-tuning.
- +1 reward for getting to the goal, -1 for dying, in addition to small dense rewards.

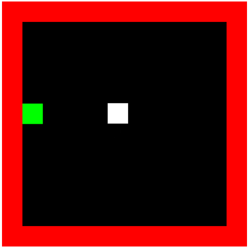
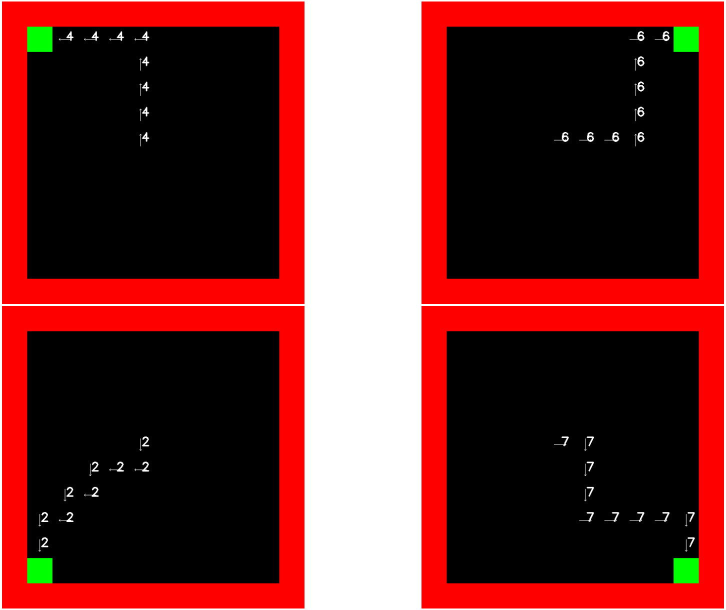
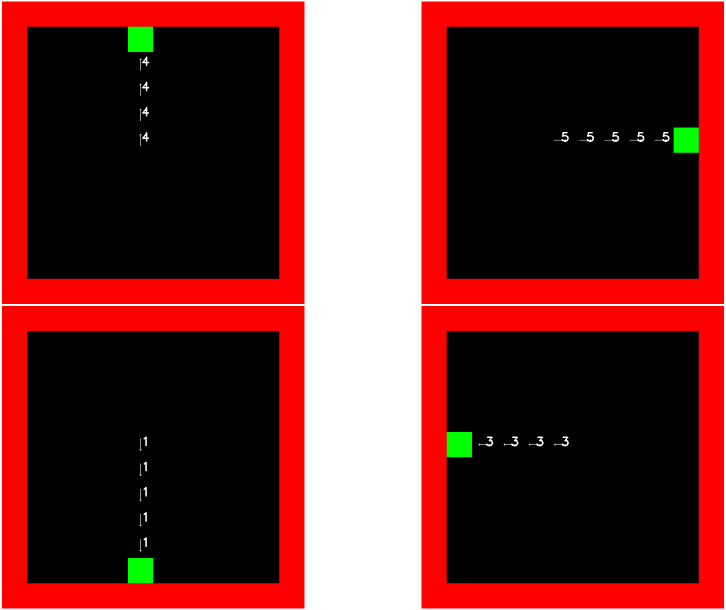


Figure: Grid Env
Green – Goal, White – Initialization

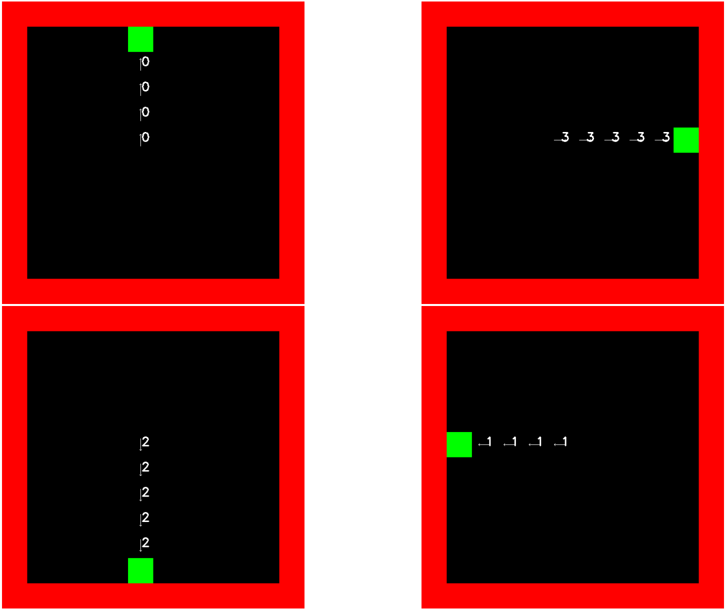
Results for training on 8 goals



Results for training on 8 goals



Results for training on 4 goals



Generalization Results after training on 8 goals

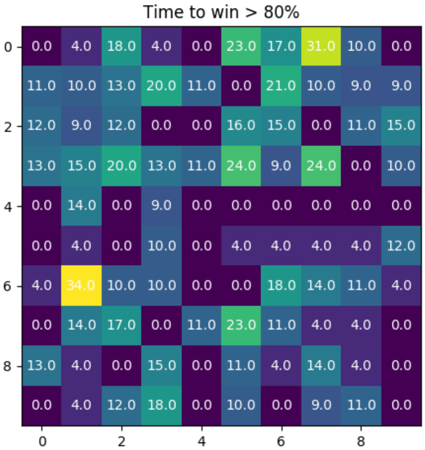


Figure: Generalization Update Steps

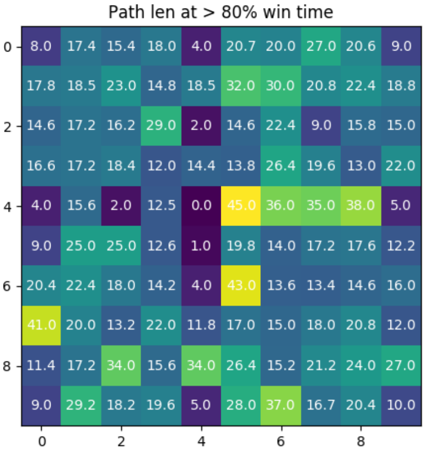


Figure: Generalization Path Length

Generalization Results after training on 8 goals

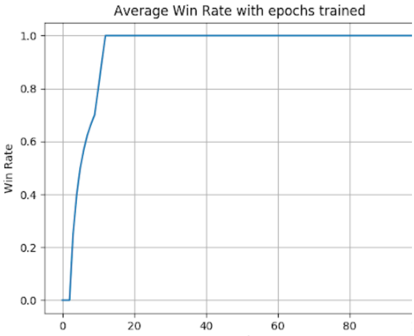


Figure: Fine-tuning time average win rate for goal(2, 2)

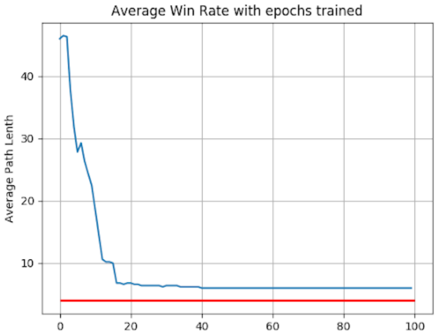


Figure: Fine-tuning time average path length for goal(2, 2)

Results

- We are able to successfully disentangle the near optimal trajectories for different goals

Results

- We are able to successfully disentangle the near optimal trajectories for different goals
- We are able to generalise to all the other unseen goals in the grid by fine-tuning the master policy for a small number of updates (approximately 20)

Results

- We are able to successfully disentangle the near optimal trajectories for different goals
- We are able to generalise to all the other unseen goals in the grid by fine-tuning the master policy for a small number of updates (approximately 20)
- Figure 3 shows that the network does not randomly reach the goal, instead it targets the goal, though for most of the goals, the number of steps taken is larger than the optimal number.

Results

- We are able to successfully disentangle the near optimal trajectories for different goals
- We are able to generalise to all the other unseen goals in the grid by fine-tuning the master policy for a small number of updates (approximately 20)
- Figure 3 shows that the network does not randomly reach the goal, instead it targets the goal, though for most of the goals, the number of steps taken is larger than the optimal number.
- The results can be further improved by the use of matrix factorization techniques on the value function.

Future Work

- Model the master network and the policy network using neural networks.

Future Work

- Model the master network and the policy network using neural networks.
- Test our approach on more complex environments like FetchReach-v1 environment

Future Work

- Model the master network and the policy network using neural networks.
- Test our approach on more complex environments like FetchReach-v1 environment
- Test our approach in complex environments where the dynamics change and our approach generalises to new dynamics

Future Work

- Model the master network and the policy network using neural networks.
- Test our approach on more complex environments like FetchReach-v1 environment
- Test our approach in complex environments where the dynamics change and our approach generalises to new dynamics
- Compare our approach with some baseline methods like UVFA for the generalisation part

Thank You