Multi-task Reinforcement Learning

@shagunsodhani

Facebook AI Research

Who am I?

- Research Engineer at Facebook AI Research
- Interested in Lifelong Reinforcement Learning
  - *Training AI systems that can interact with and learn from the physical world and consistently improve as they do so without forgetting the previous knowledge*
- Also work on distributed optimization, generalization etc

Agenda

- Overview of (Single Task) Reinforcement Learning
    - What is reinforcement learning using the example of chatbots
    - Components - agent, environment, state-space, action-space, MDP, etc
    - Goal is set a common terminology
    - If you are well-versed in (single task) RL, feel free to jump ahead :)

Agenda

- Overview of (Single Task) Reinforcement Learning
- Introduce Multi Task Reinforcement Learning
    - Start with a general setup
    - Show some common building blocks / techniques for multi task RL
    - Add assumptions to the general setup and discuss how these assumptions lead to different setups and how they influence the model architecture.

What is not on the Agenda

- Standard RL algorithms like policy gradients etc.
- Implementation details.
- Detailed discussion on specific papers.
- I would be overloading the notation at some places, to focus more on the intuition.

Disclaimer

- This is not an exhaustive literature survey on multi task RL.
- We will look at some research papers and setups but there are a lot of other important works.
- The focus will be on providing the motivation/intuition behind the different setups.

Reinforcement Learning

Chatbot Example

- We want to develop a chatbot (aka conversational agent) that can
  - engage humans in a conversation [1]
  - improve user experience by answering their questions [2]

[1]: https://developer.amazon.com/alexaprize/

[2]: https://www.intercom.com/blog/customer-service-chatbots/

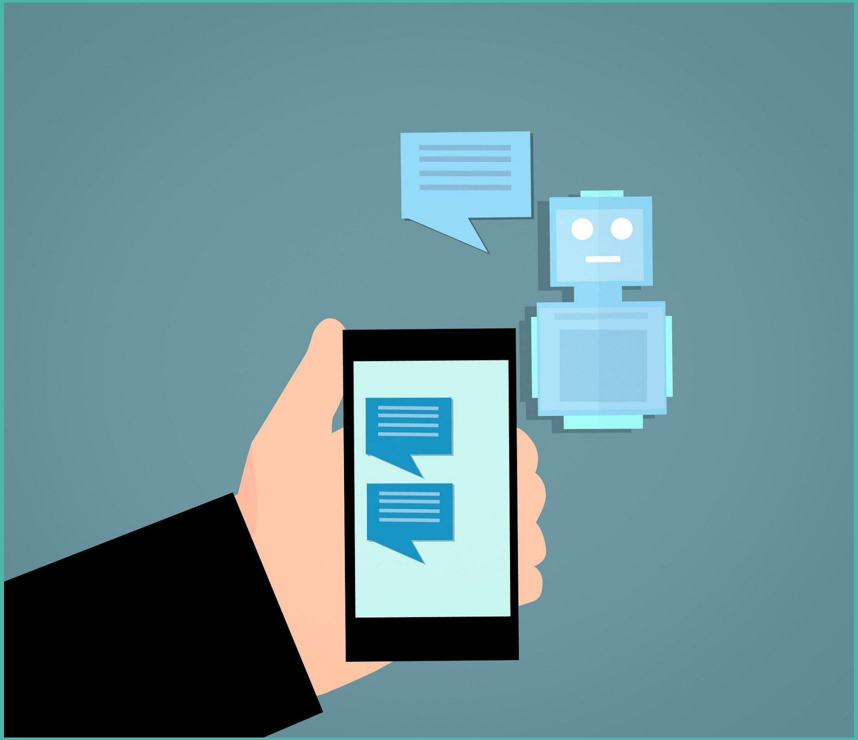


Image by mohamed Hassan from Pixabay

Chatbot Example

Alice: Hi

Bot: What kind of music do you like? I like all kinds of music but country is my favorite.

Alice: I enjoy hard rock.

Bot: Thats cool. Do you have a favorite band? Mine is led zeppelin.

Alice: Led Zeppelin is one of my favorites too! Also Pink Floyd.

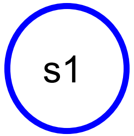
Bot: I like them too. My favorite album of theirs is dark side of the moon.

[3]: [2004.13637] Recipes for building an open-domain chatbot

Chatbot Example

Agent's view

Alice: Hi



[3]: [2004.13637] Recipes for building an open-domain chatbot

Chatbot Example

Agent’s view

Alice: Hi

Bot: What kind of music do you like? I like all kinds of music but country is my favorite.

[3]: [2004.13637] Recipes for building an open-domain chatbot

Chatbot Example

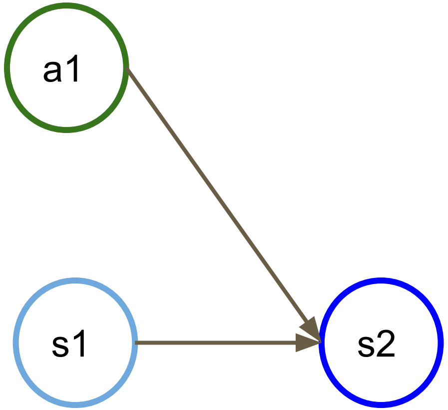
Alice: Hi

Bot: What kind of music do you like? I like all kinds of music but country is my favorite.

Alice: I enjoy hard rock.

[3]: [2004.13637] Recipes for building an open-domain chatbot

Agent's view



Chatbot Example

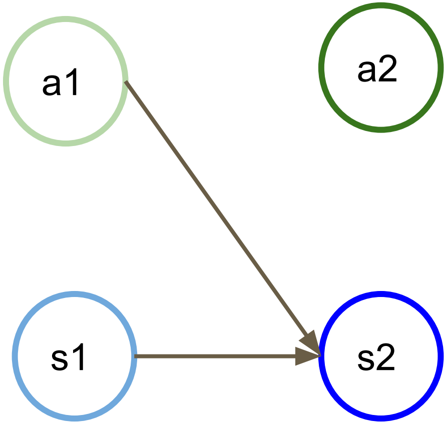
Alice: Hi

Bot: What kind of music do you like? I like all kinds of music but country is my favorite.

Alice: I enjoy hard rock.

Bot: Thats cool. Do you have a favorite band? Mine is led zeppelin.

Agent's view



[3]: [2004.13637] Recipes for building an open-domain chatbot

Chatbot Example

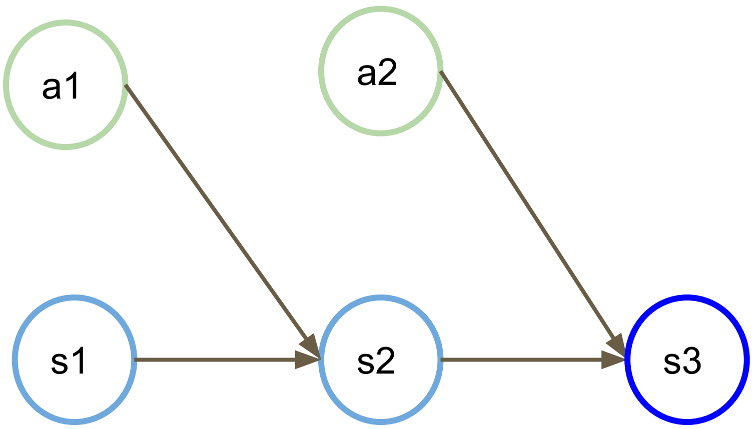
...

Alice: I enjoy hard rock.

Bot: Thats cool. Do you have a favorite band? Mine is led zeppelin.

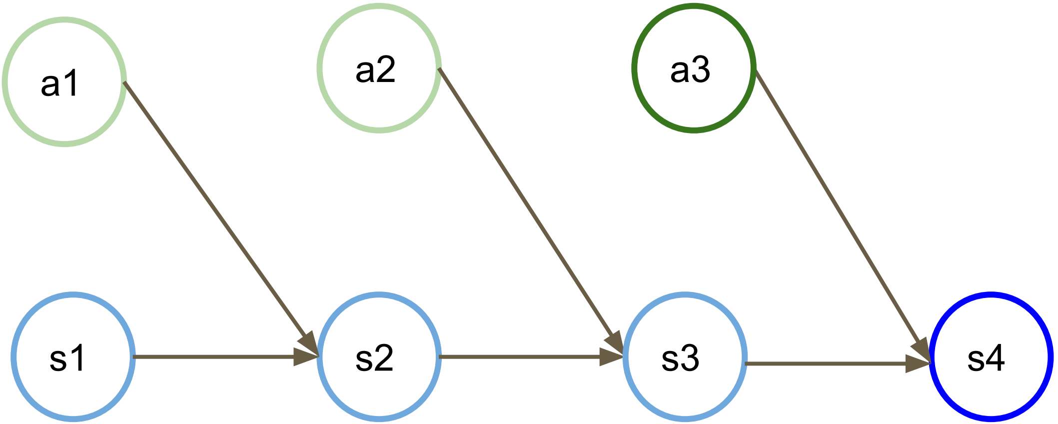
Alice: Led Zeppelin is one of my favorites too! Also Pink Floyd.

Agent's view

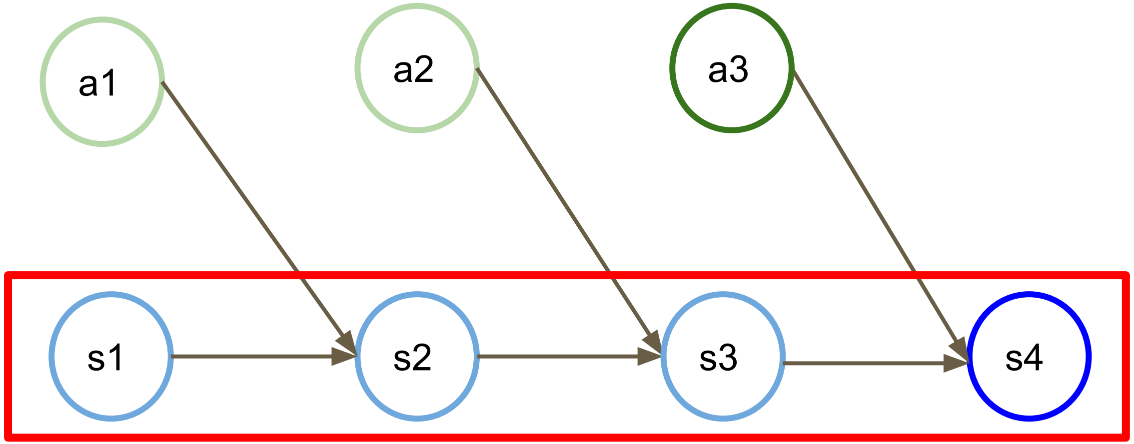


[3]: [2004.13637] Recipes for building an open-domain chatbot

Sequential Decision Making



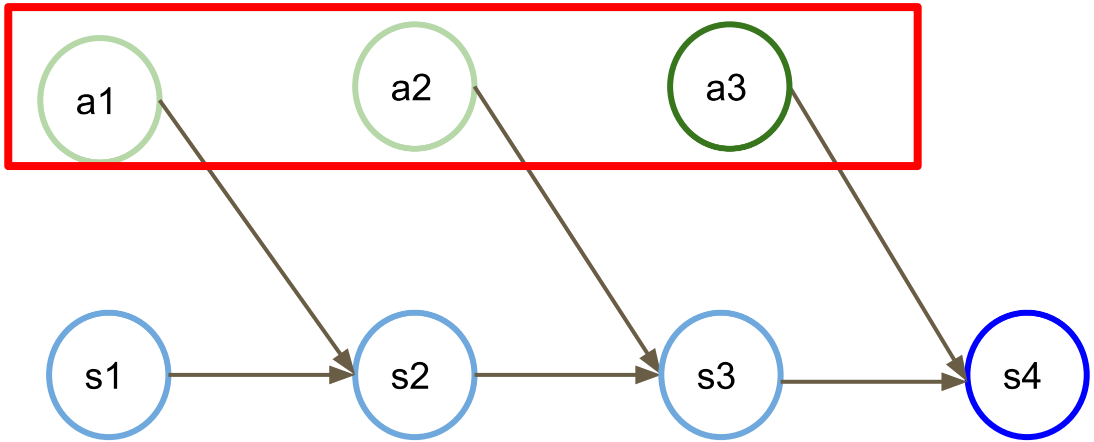
Sequential Decision Making



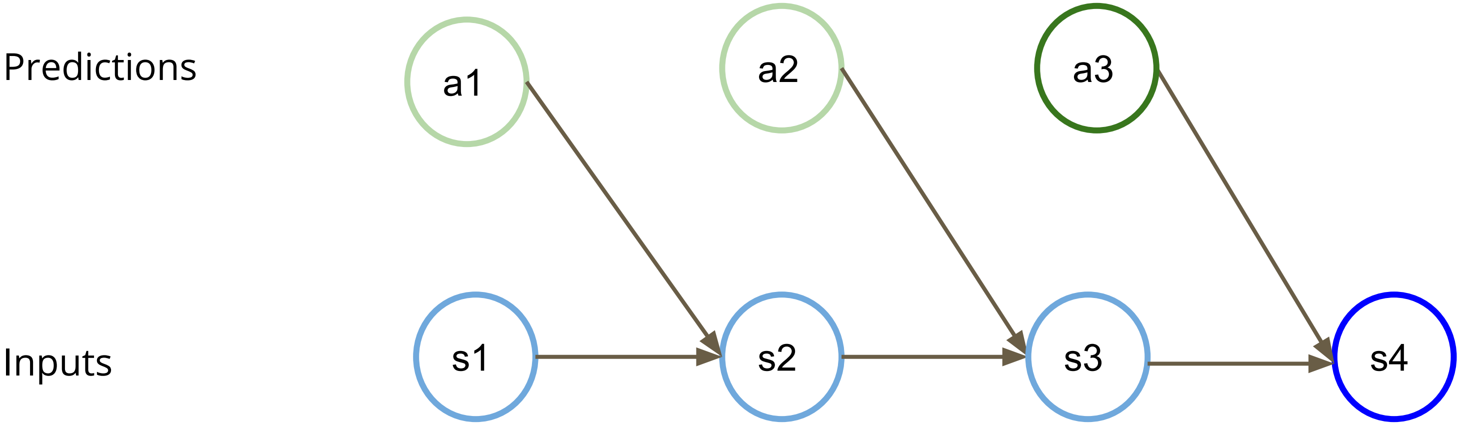
Inputs: state of a conversation

Sequential Decision Making

Action: chatbot’s dialog in a conversation



Solving a Sequential Decision Making Problem

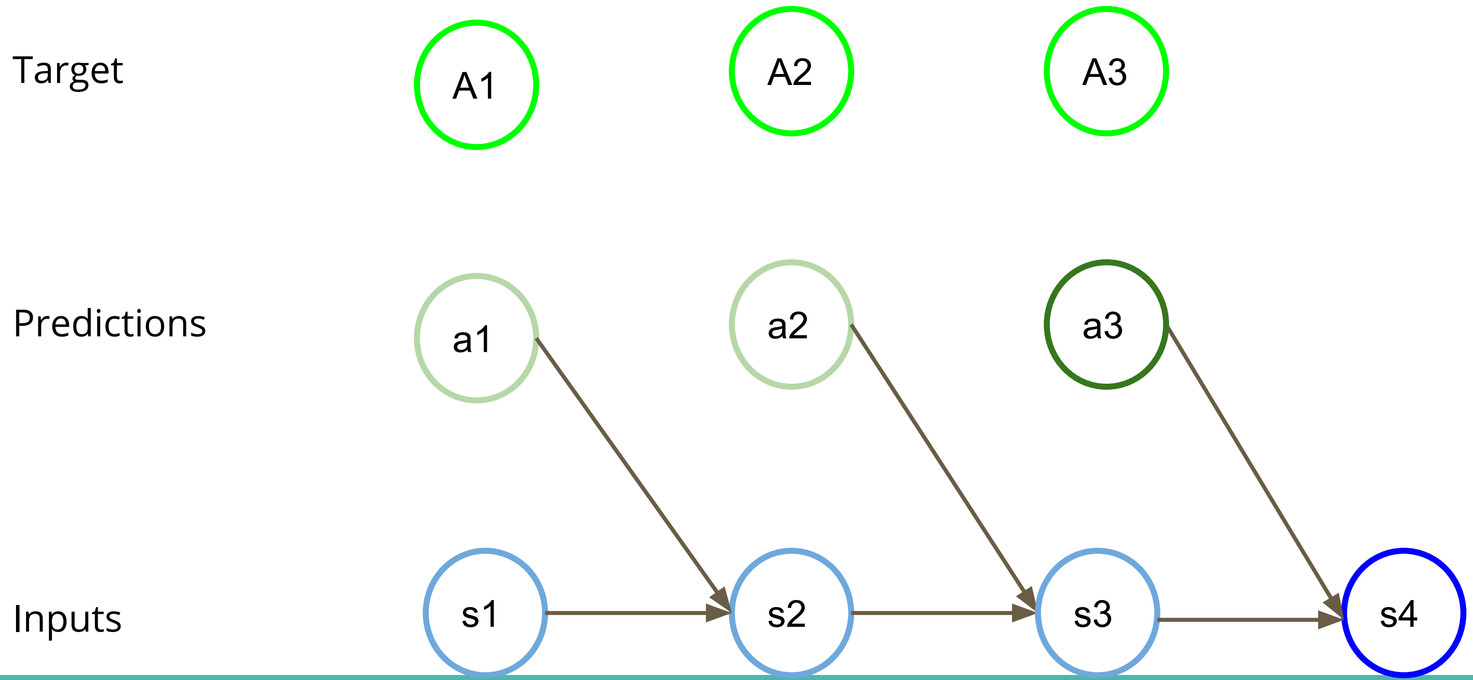


Solving a Sequential Decision Making Problem - I

We know the “right” predictions.

Solving a Sequential Decision Making Problem - I

We know the "right" predictions.

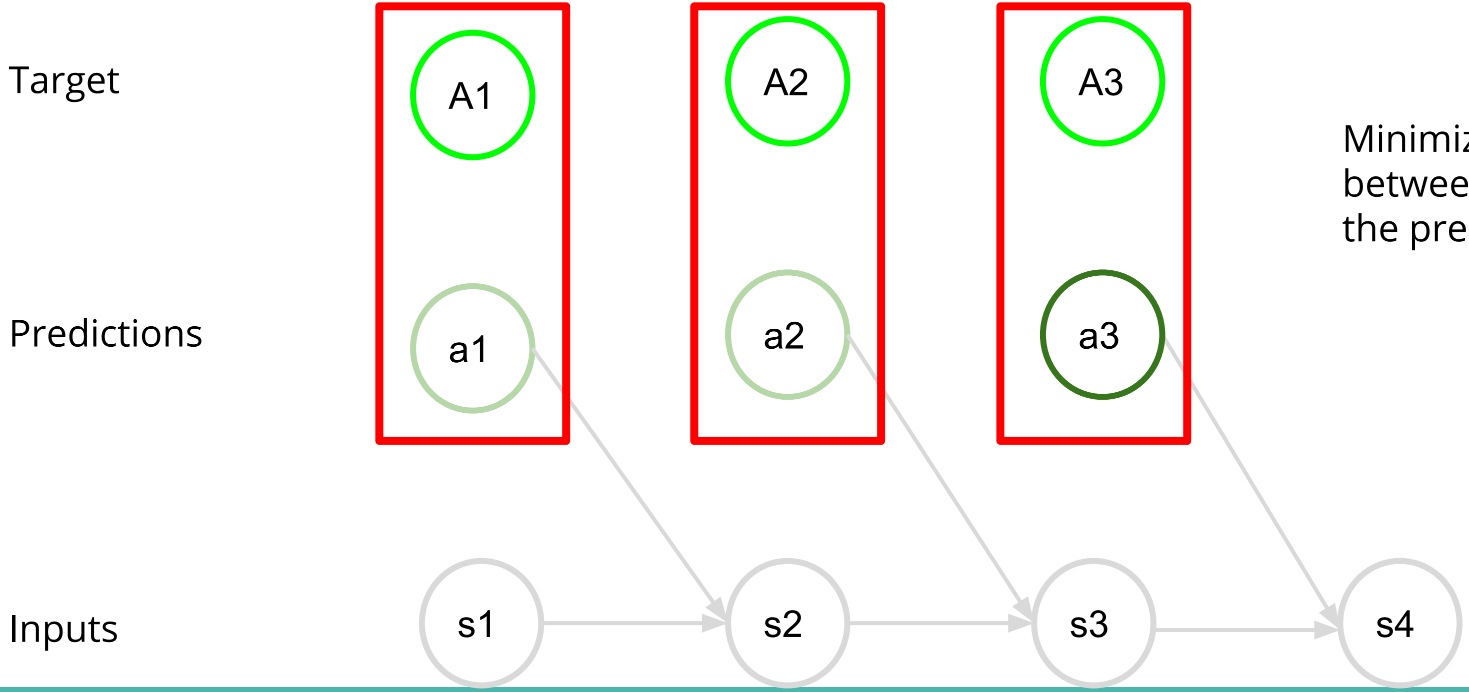


Solving a Sequential Decision Making Problem - I

We know the “right” predictions. **We can use supervised learning.**

Solving a Sequential Decision Making Problem - I

We know the “right” predictions. **We can use supervised learning.**

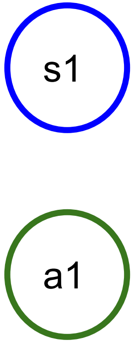


Minimize the difference between the correct and the predicted decisions

Chatbot Example

Alice: Hi

Bot: What kind of music do you like? I like all kinds of music but country is my favorite.



Chatbot Example

Alice: Hi

s1

Bot: What kind of music do you like? I like all kinds of music but country is my favorite.

a1

Bot: Do you like the weather these days? I find it a little too windy.

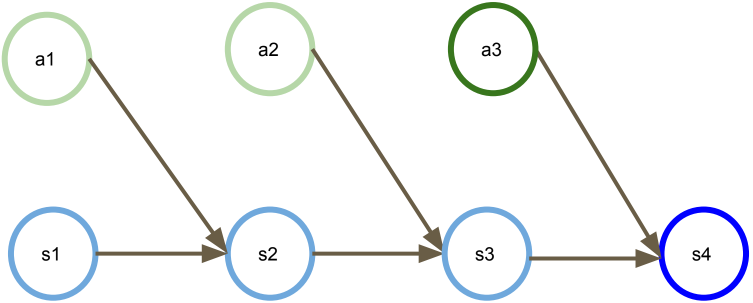
b1

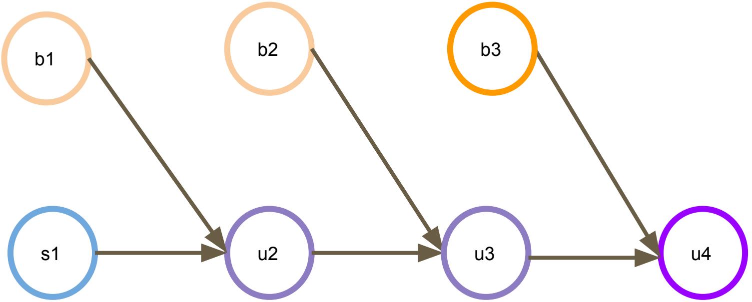
This alternate dialog is not necessarily “wrong” if the goal is to have an engaging conversation.

Which sequence is better?

Which is the more engaging conversation?

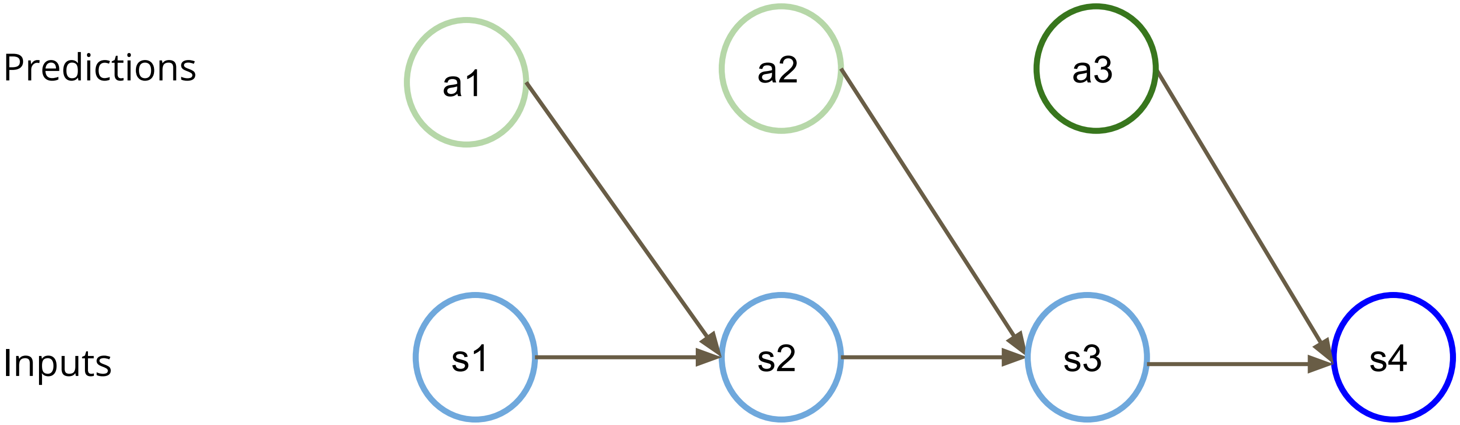
Maybe we can ask the users to provide a rating at the end of the conversation.





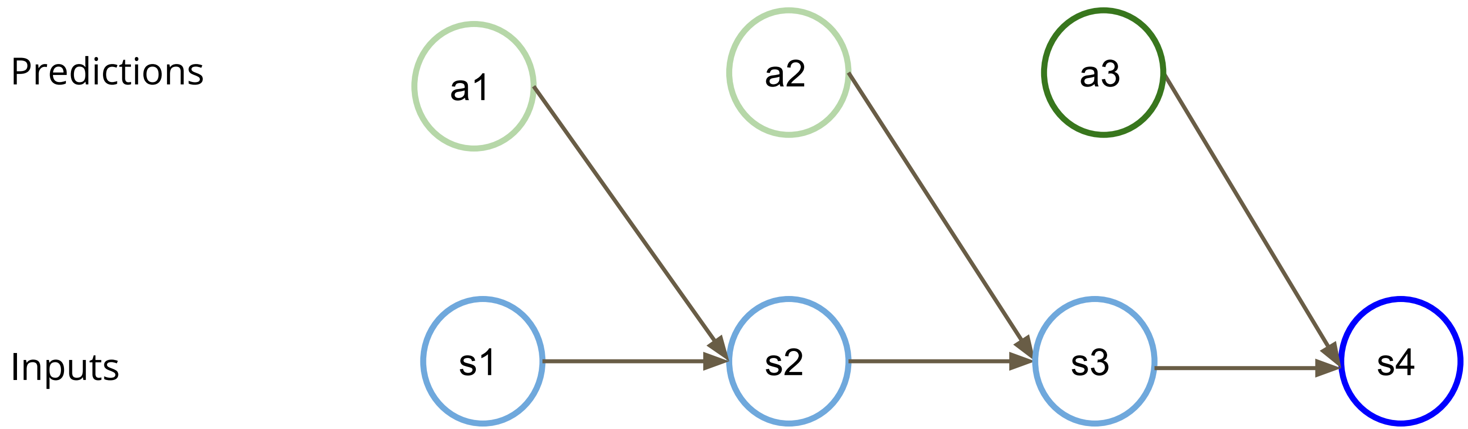
Solving a Sequential Decision Making Problem - II

We do not know the “right” predictions. But we have a sense of “goodness” of our predictions.



Solving a Sequential Decision Making Problem - II

We do not know the “right” predictions. But we have a sense of “goodness” of our predictions. **We can use Reinforcement Learning.**



Characteristics of Reinforcement Learning

1. Map input to some action.

[4]: Sutton & Barto Book: Reinforcement Learning: An Introduction

Characteristics of Reinforcement Learning

1. Map input to some action.

2. Objective is to maximize a reward signal (rating in the previous example).

[4]: Sutton & Barto Book: Reinforcement Learning: An Introduction

Characteristics of Reinforcement Learning

1. Map input to some action.
2. Objective is to maximize a reward signal (rating in the previous example).
3. Trial and error approach - the optimal action is not known, but has to be discovered by interaction.

[4]: Sutton & Barto Book: Reinforcement Learning: An Introduction

Characteristics of Reinforcement Learning

1. Map input to some action.

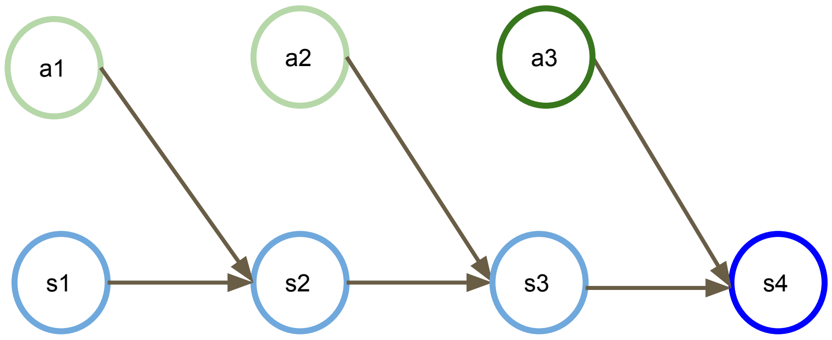
2. Objective is to maximize a reward signal (rating in the previous example).

3. Trial and error approach - the optimal action is not known, but has to be discovered by interaction.

4. Delayed rewards - current action could affect all subsequent rewards.

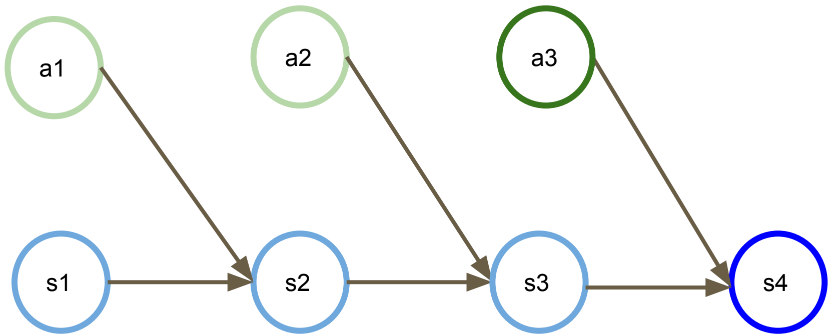
[4]: Sutton & Barto Book: Reinforcement Learning: An Introduction

Sequential Decision Making

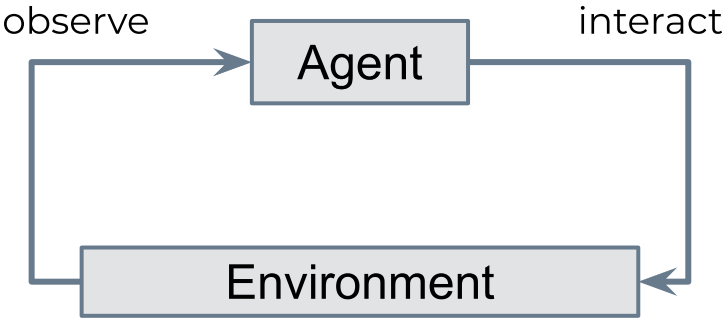


Observe -> Interact -> Observe -> Interact ...

Sequential Decision Making



Observe -> Interact -> Observe -> Interact ...



Reinforcement Learning

Agent

The learner (eg chatbot)

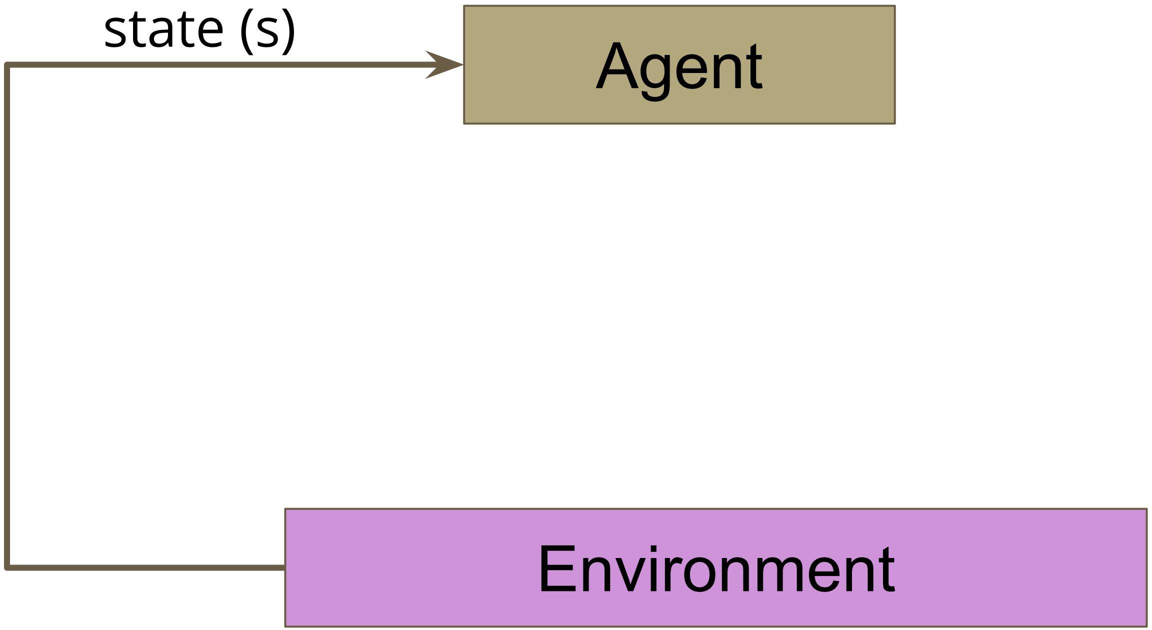
Reinforcement Learning

Agent

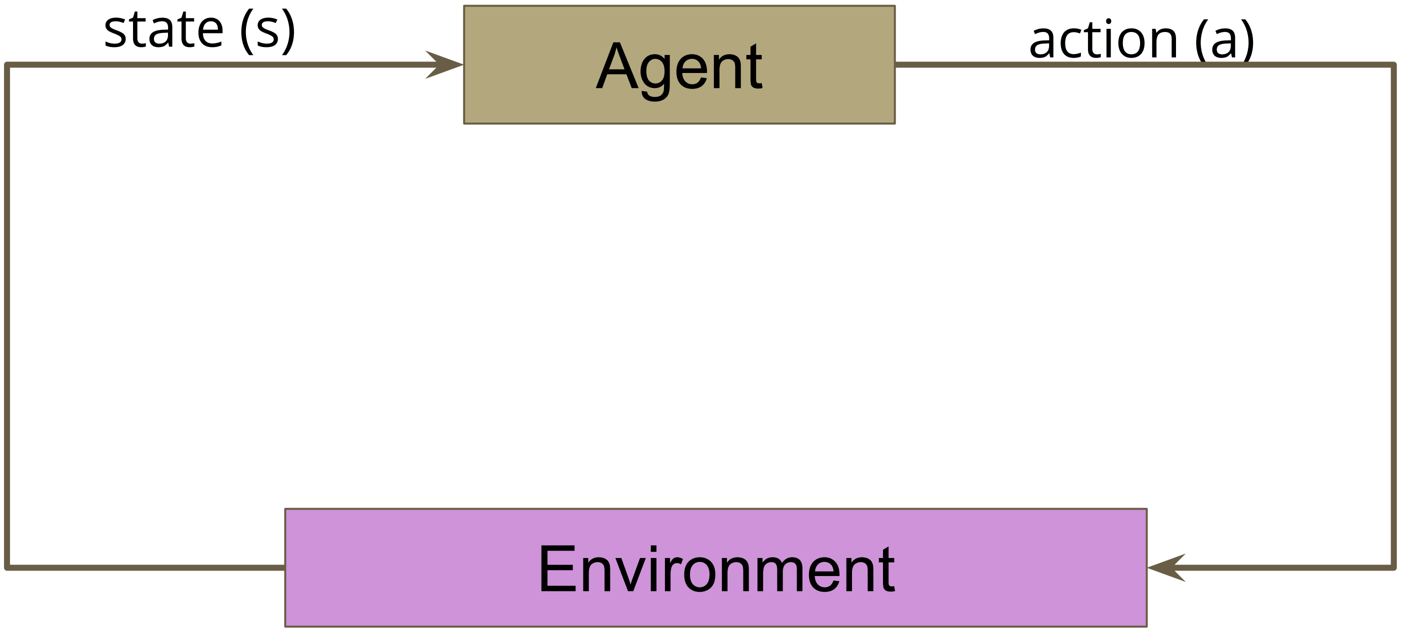
Everything outside the agent

Environment

Reinforcement Learning



Reinforcement Learning



Characteristics of Reinforcement Learning

1. **Map input to some action.**

2. Objective is to maximize a reward signal (rating in the previous example).

3. Trial and error approach - the optimal action is not known, but has to be discovered by interaction.

4. Delayed rewards - current action could affect all subsequent rewards.

[4]: Sutton & Barto Book: Reinforcement Learning: An Introduction

Reinforcement Learning

Reinforcement Learning

$$a = \pi(s)$$

Reinforcement Learning

$$a = \boxed{\pi}(s)$$

policy

Reinforcement Learning

$$a = \pi(s)$$

Policy: Function that maps input to some action

Characteristics of Reinforcement Learning

1. Map input to some action.

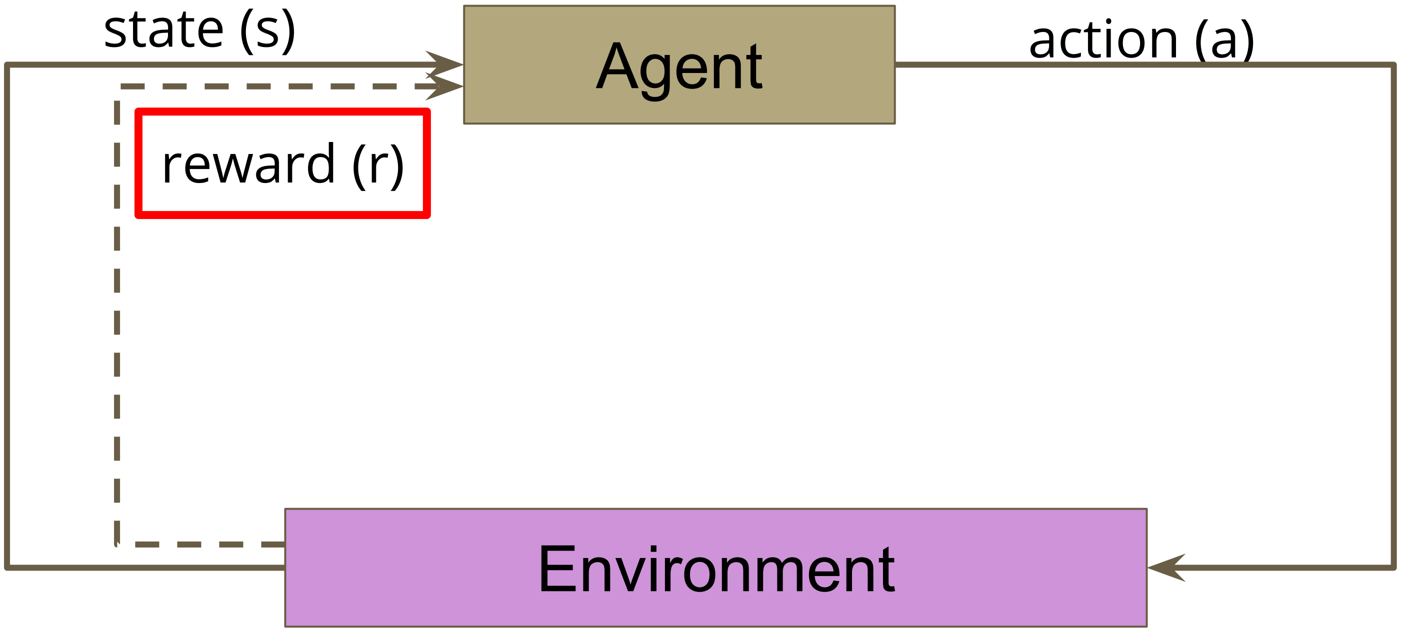
2. **Objective is to maximize a reward signal (rating in the previous example).**

3. Trial and error approach - the optimal action is not known, but has to be discovered by interaction.

4. Delayed rewards - current action could affect all subsequent rewards.

[4]: Sutton & Barto Book: Reinforcement Learning: An Introduction

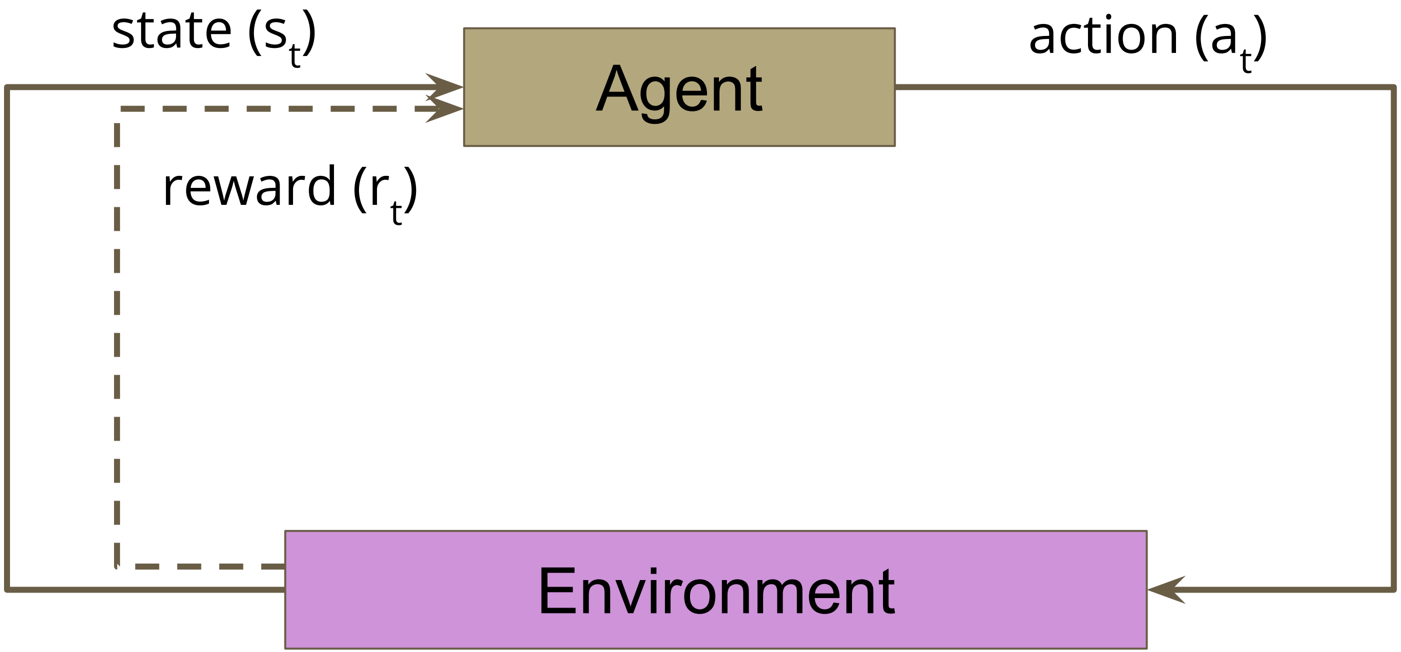
Reinforcement Learning



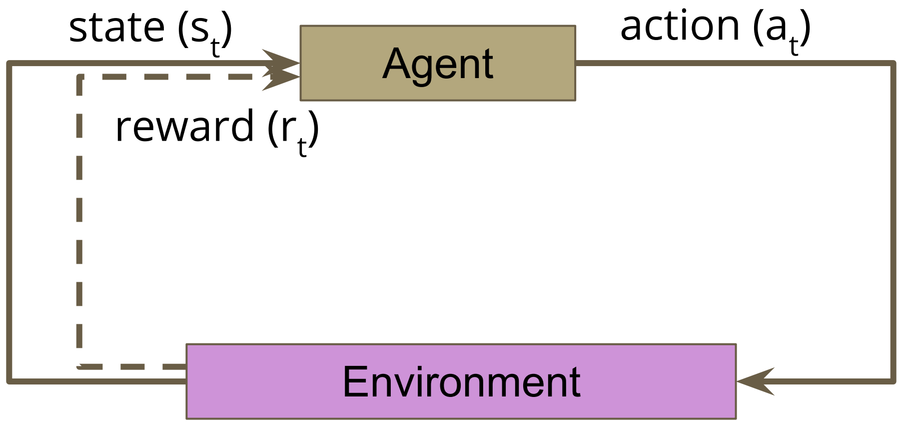
Reinforcement Learning

- Fineprint: We want to maximize the expected discounted reward and not just the immediate reward.
- “discounted” means that immediate rewards are more valuable than rewards that are far off.
- For example, 100 $ today are more valuable than 100$ in the future.

Reinforcement Learning

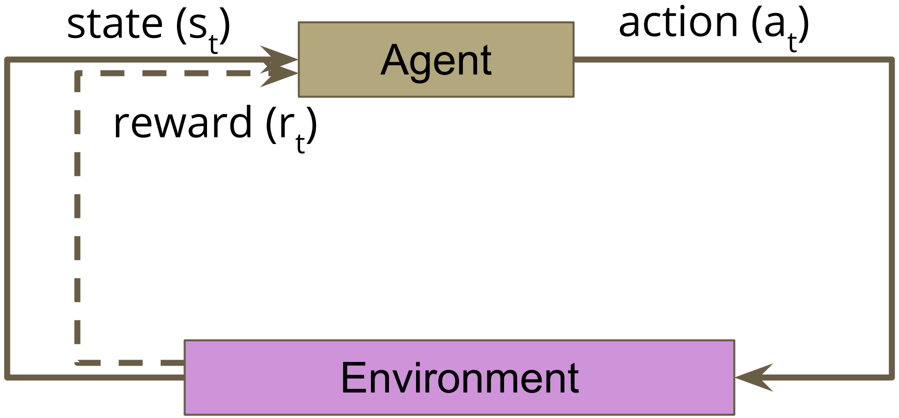


Reinforcement Learning



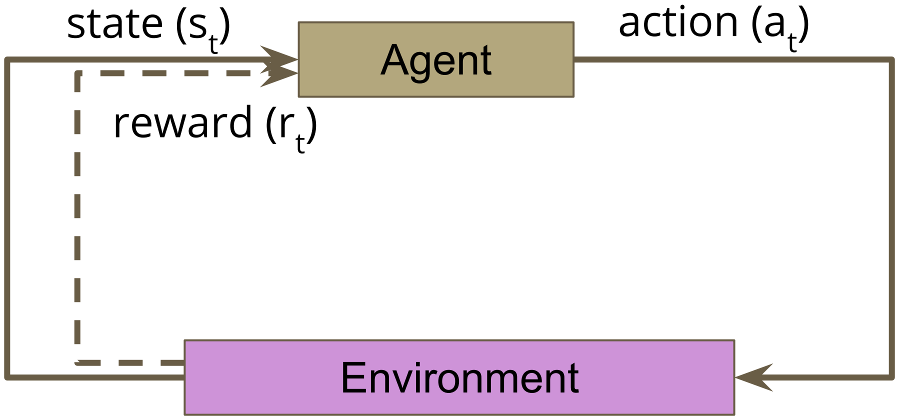
s_0

Reinforcement Learning



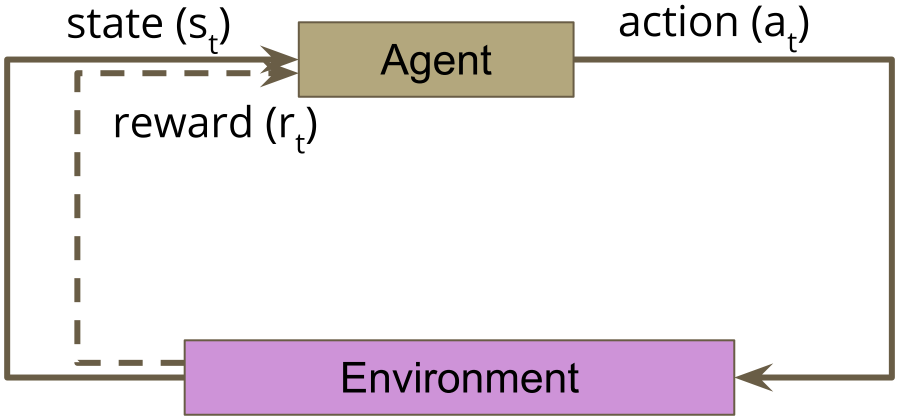
s_0 $\boxed{a_0}$

Reinforcement Learning



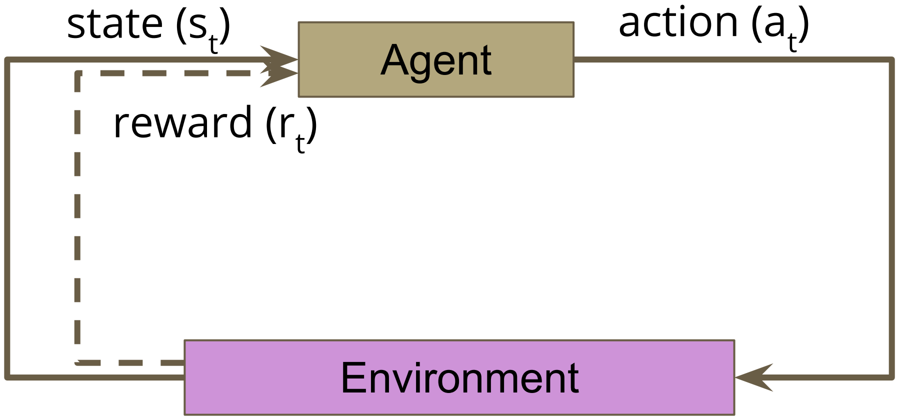
$s_0 \quad a_0 \quad \boxed{s_1 \quad r_1}$

Reinforcement Learning



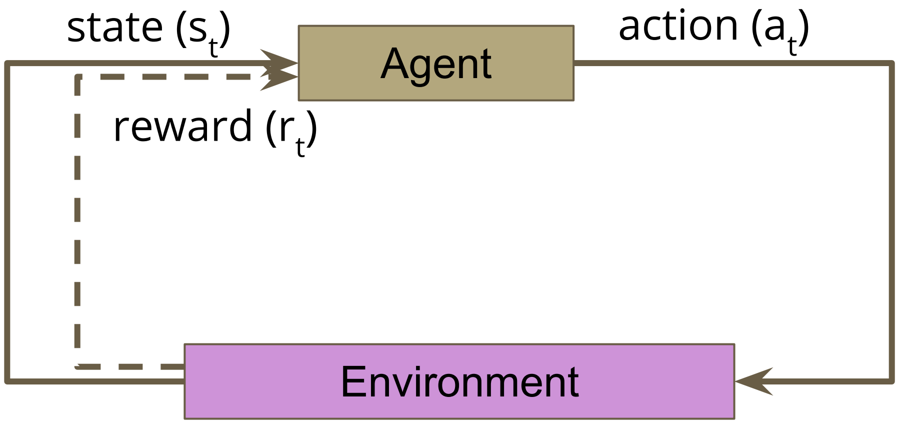
$s_0 \quad a_0 \quad s_1 \quad r_1 \quad \boxed{a}$

Reinforcement Learning



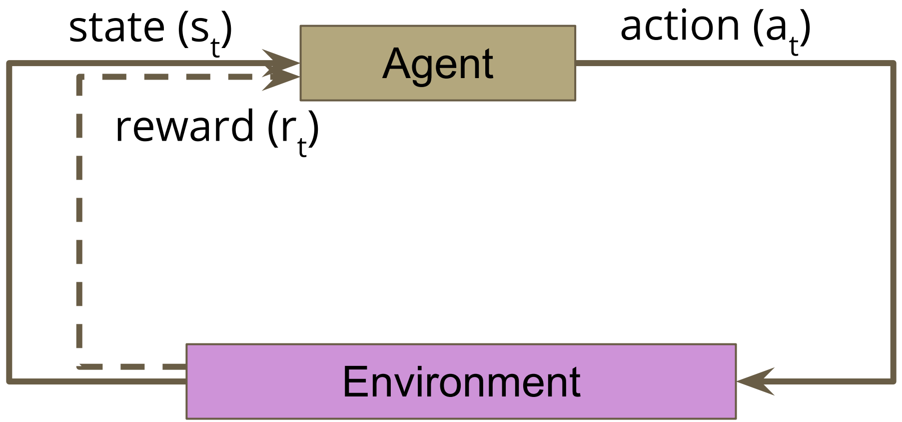
$s_0 \quad a_0 \quad s_1 \quad r_1 \quad a_1 \quad \boxed{s_2} \quad r_2$

Mathematically



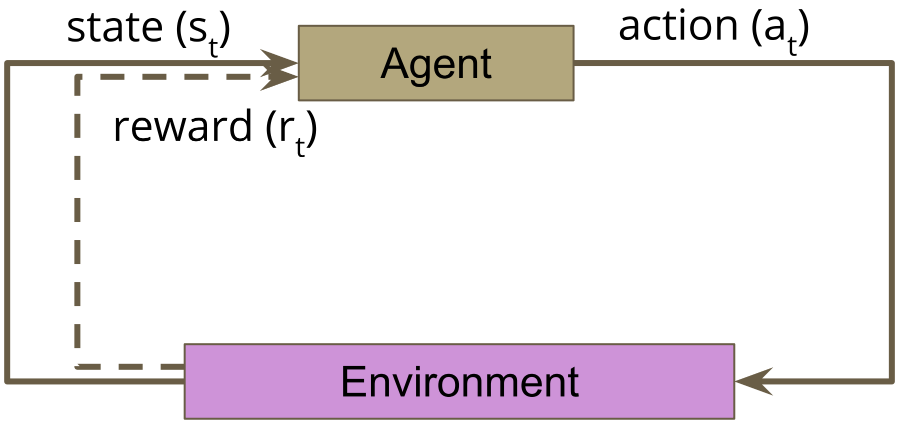
- State space - set of all possible states.

Mathematically



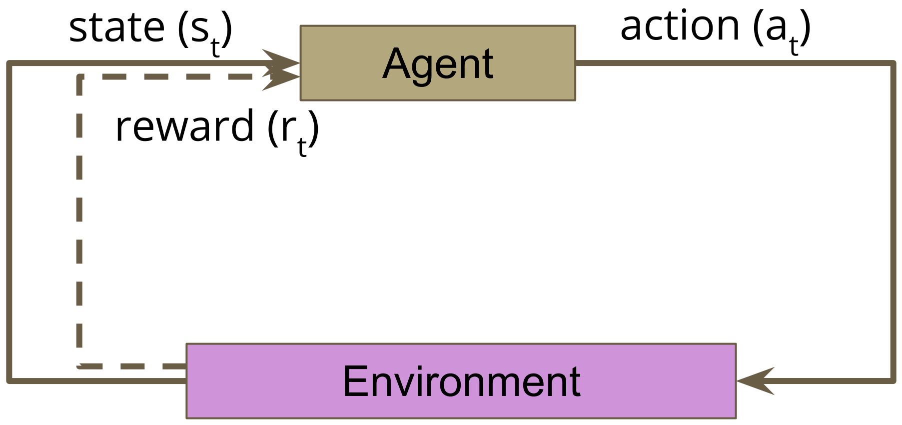
- State space - set of all possible states.
- Action space - set of all possible actions.

Mathematically



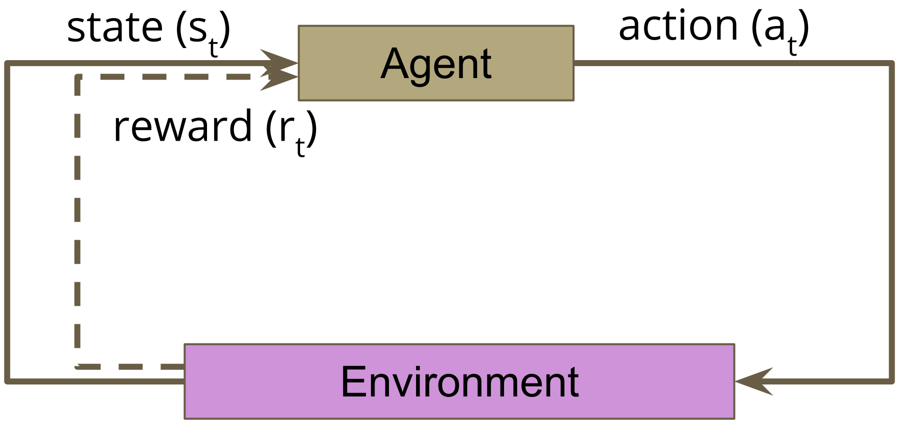
- State space - set of all possible states.
- Action space - set of all possible actions.
- Reward function - how much reward does the agent get in state s when it takes action a
  - $r_t = R(s_t, a_t)$

Mathematically



- State space - set of all possible states.
- Action space - set of all possible actions.
- Reward function - how much reward does the agent get in state s when it takes action a
  - $r_t = R(s_t, a_t)$
- Transition function - what is the next state when the agent takes action a in state s
  - $s_{t+1} = T(s_t, a_t)$

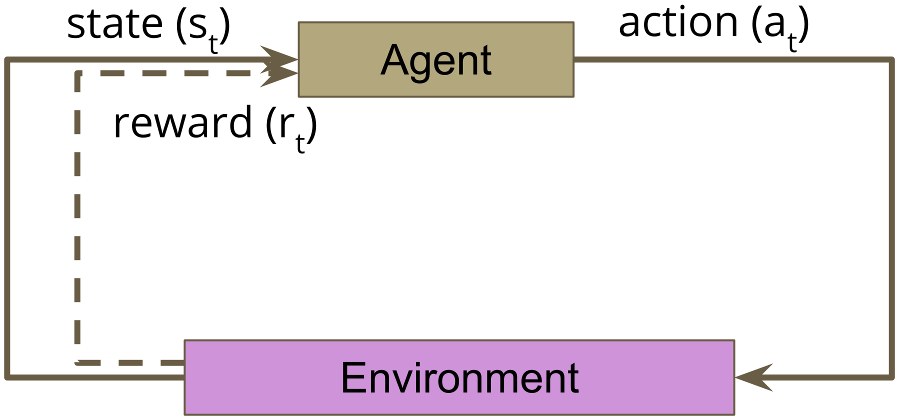
Mathematically



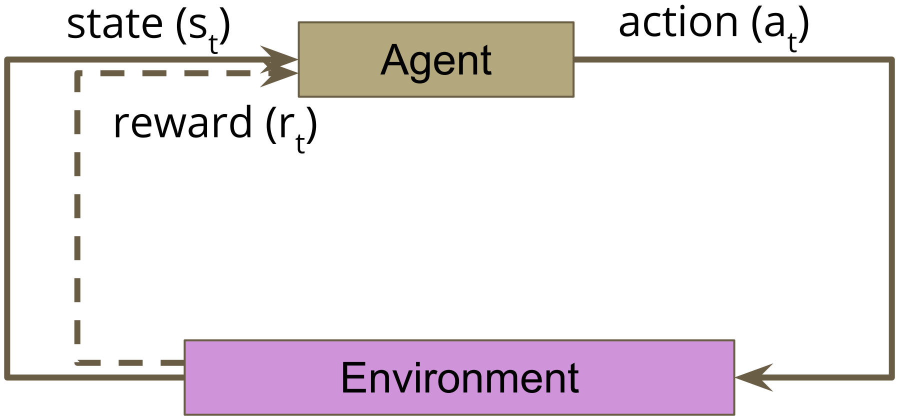
- State space - set of all possible states.
- Action space - set of all possible actions.
- Reward function - how much reward does the agent get in state s when it takes action a
  - $r_t = R(s_t, a_t)$
- Transition function - what is the next state when the agent takes action a in state s
  - $s_{t+1} = T(s_t, a_t)$

Markov Decision Process (MDP) - Formalization of sequential decision making process

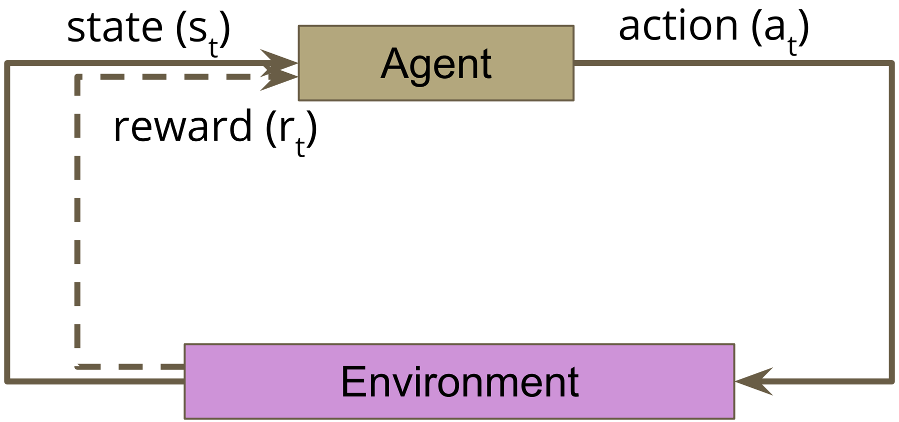
In Practice



In Practice

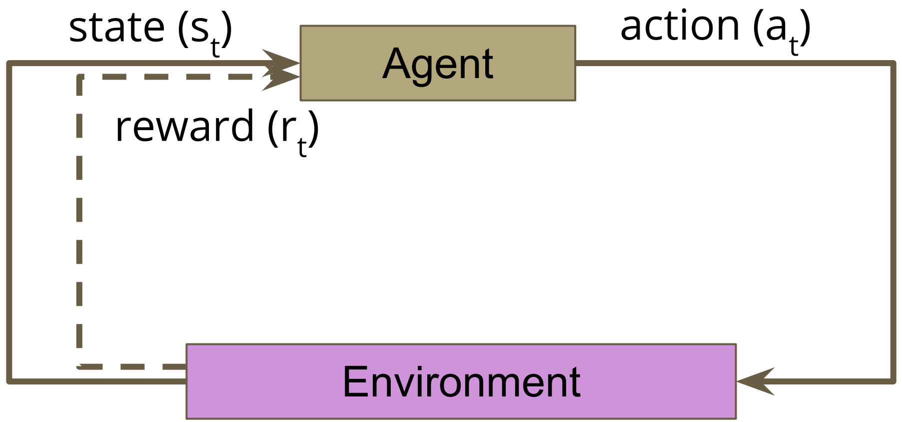


In Practice



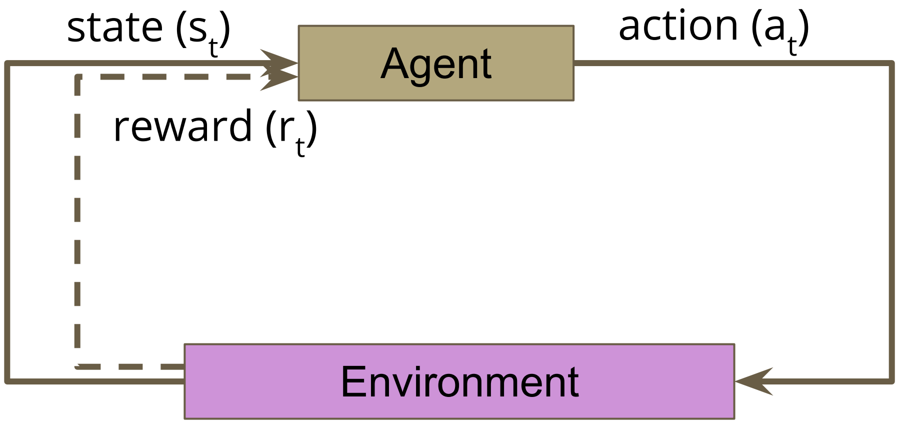
- Encoder: maps the environment's observation/state to a vector.

In Practice



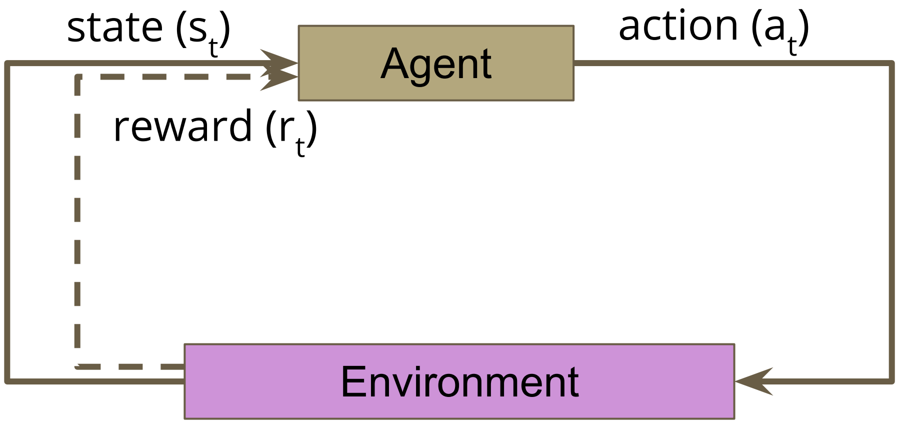
- Encoder: maps the environment's observation/state to a vector.
- Policy: map the vector to action.

In Practice



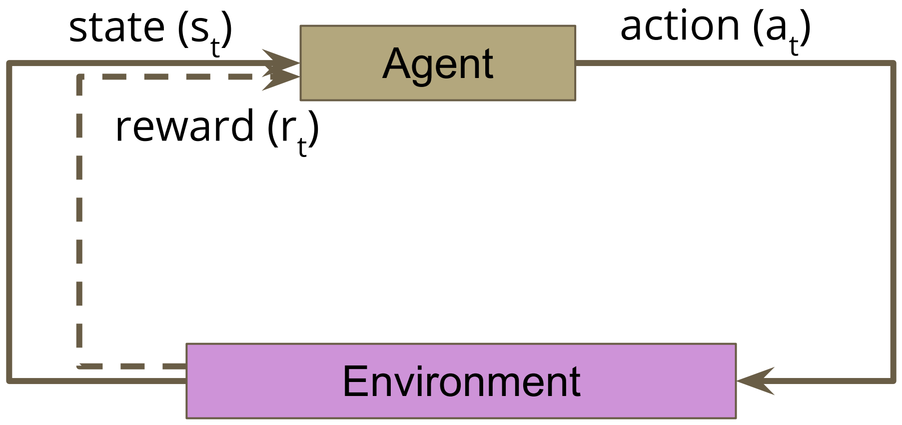
- Encoder: maps the environment’s observation/state to a vector.
- Policy: map the vector to action.
- Value functions: how good a state (or state-action pair) is.

In Practice



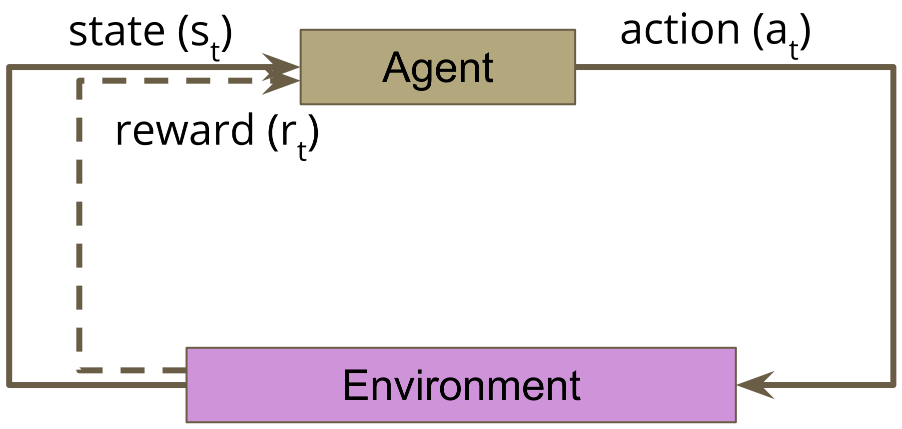
- Encoder: maps the environment's observation/state to a vector.
- Policy: map the vector to action.
- Value functions: how good a state (or state-action pair) is.
- Reward function (that we learn): predict the reward for a state-action pair.

In Practice



- Encoder: maps the environment's observation/state to a vector.
- Policy: map the vector to action.
- Value functions: how good a state (or state-action pair) is.
- Reward function (that we learn): predict the reward for a state-action pair.
- Transition function (that we learn): predict the next state, given the current state-action pair.

In Practice



- Encoder: maps the environment's observation/state to a vector.
- Policy: map the vector to action.
- Value functions: how good a state (or state-action pair) is.
- Reward function (that we learn): predict the reward for a state-action pair.
- Transition function (that we learn): predict the next state, given the current state-action pair.
- Replay buffer: if using off-policy learning

……

So far

- We have seen the different components for a single-task RL problem.
- We intentionally did not discuss any RL algorithms (e.g. policy gradients).
- We assume we have access to an algorithm that can learn the policy.
- We will now look at different multi-task RL setups.

Multi-task Reinforcement Learning

Multi-task Reinforcement Learning

- We have n RL tasks to learn.

Multi-task Reinforcement Learning

- We have n RL tasks to learn.
- Each task has its own environment, state space, action space, reward function, transition dynamics, etc.

Multi-task Reinforcement Learning

- We have n RL tasks to learn.
- Each task has its own environment, state space, action space, reward function, transition dynamics, etc.
- This is the most general case of multi-task RL where we do not make any assumptions.

What do we care about

- We have the performance on n tasks (say R_1, R_2, ... R_n) :

What do we care about

- We have the performance on n tasks (say R_1, R_2, ... R_n) :
    - Average Performance - Average(R_1, R_2, ... R_n)

What do we care about

- We have the performance on n tasks (say R_1, R_2, ... R_n) :
    - Average Performance - Average(R_1, R_2, ... R_n)
    - Median Performance - Median(R_1, R_2, ... R_n)

What do we care about

- We have the performance on n tasks (say $R_1, R_2, \ldots R_n$) :
  - Average Performance - Average($R_1, R_2, \ldots R_n$)
  - Median Performance - Median($R_1, R_2, \ldots R_n$)
  - Worst Performance - Min($R_1, R_2, \ldots R_n$)

Case I

- Each task has its own environment, state space, action space, reward function, transition dynamics, etc.
- There is nothing common between the tasks. So there is no knowledge to share across the tasks.
- The best we can do is to learn n agents, each trained for one task.
- Given a task, we lookup the agent for that task and we use that agent to solve the task.

Case I

One policy per task

π_1

π_2

π_3

π_4

One agent per task

enc_1	π_1
enc_2	π_2
enc_3	π_3
enc_4	π_4

Case II - Shared state and action space

- Examples - Navigation, locomotion, interacting with objects
- The only multi-task algorithm we know so far is: one-agent-per-task. So we start with that.
- When training the n agents, we want to share knowledge between them.
- Distral [5] provides an effective mechanism for doing that.

[5]: [1707.04175] Distral: Robust Multitask Reinforcement Learning

Distral: Robust Multitask Reinforcement Learning

Image taken from [1707.04175] Distral: Robust Multitask Reinforcement Learning

Distral: Robust Multitask Reinforcement Learning

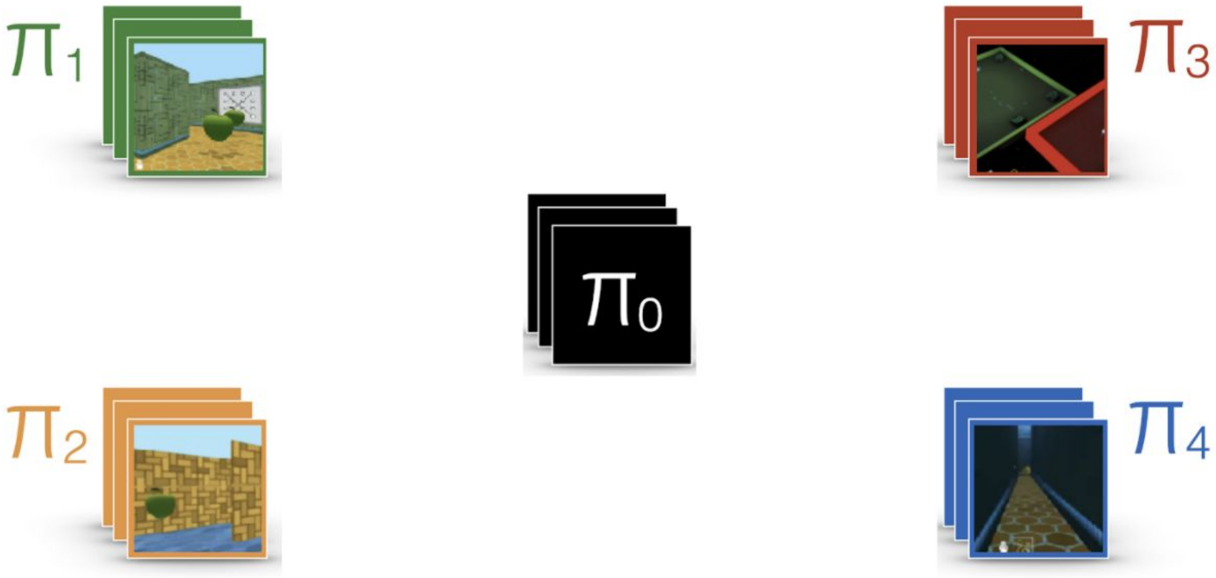


Image taken from [1707.04175] Distral: Robust Multitask Reinforcement Learning

Distral: Robust Multitask Reinforcement Learning

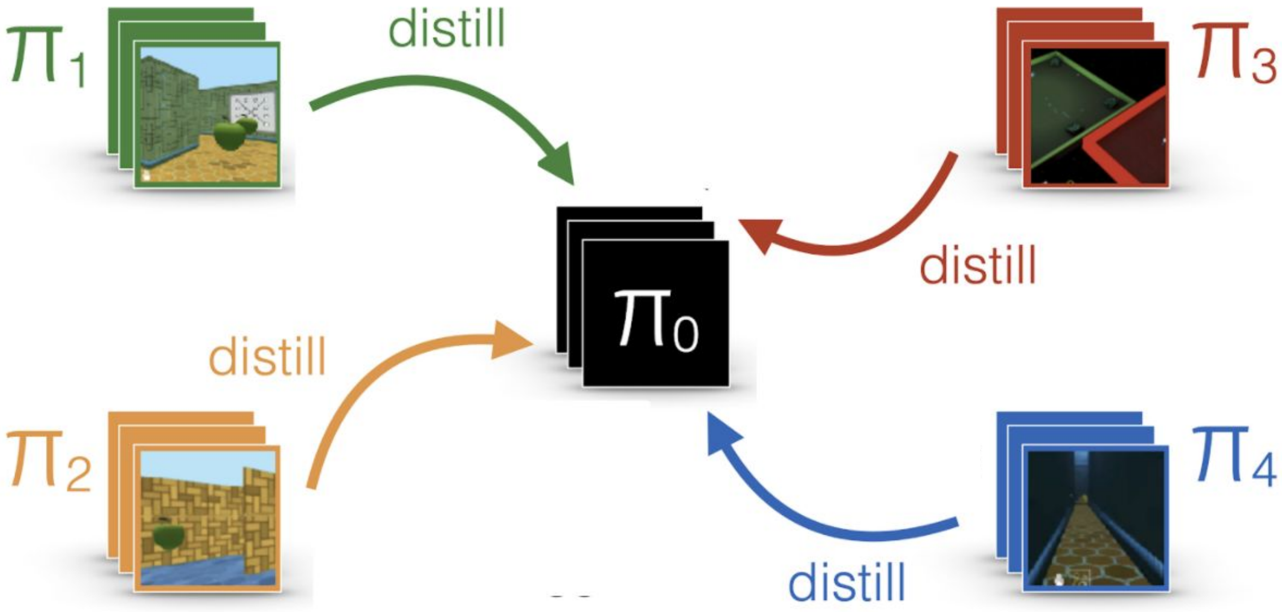


Image taken from [1707.04175] Distral: Robust Multitask Reinforcement Learning

Distral: Robust Multitask Reinforcement Learning

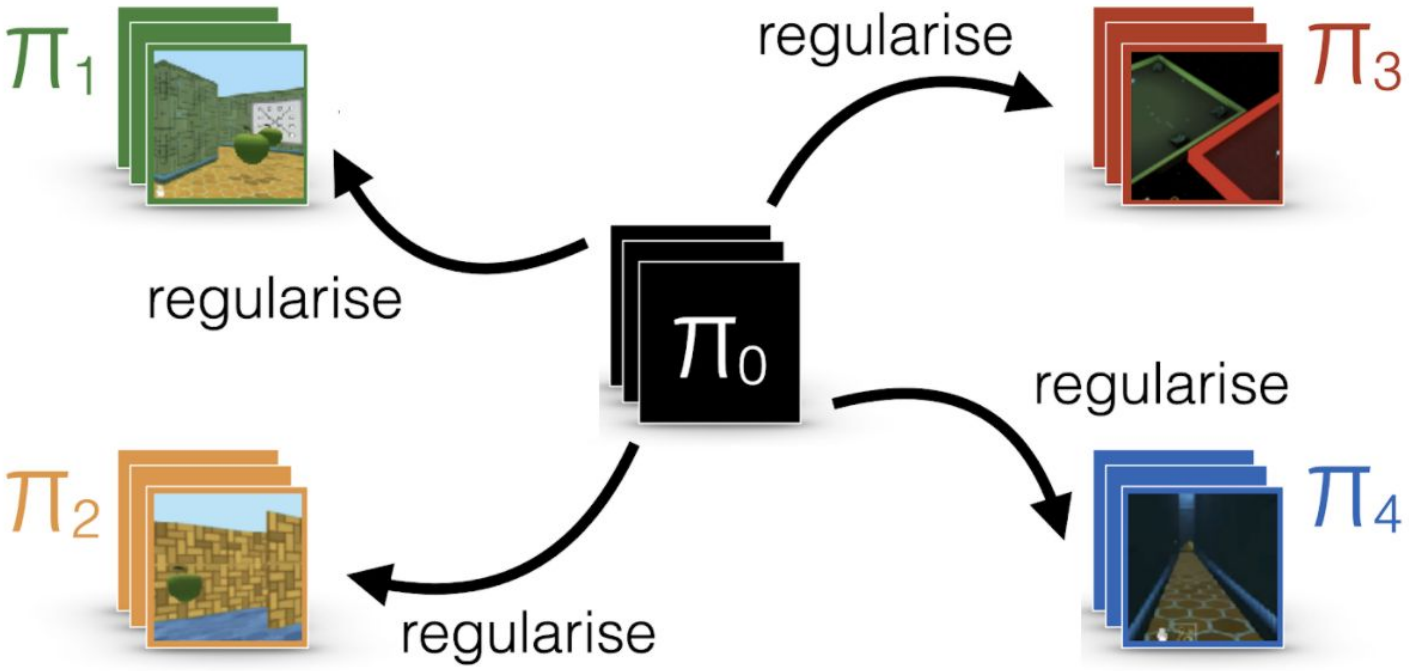


Image taken from [1707.04175] Distral: Robust Multitask Reinforcement Learning

Distral: Robust Multitask Reinforcement Learning

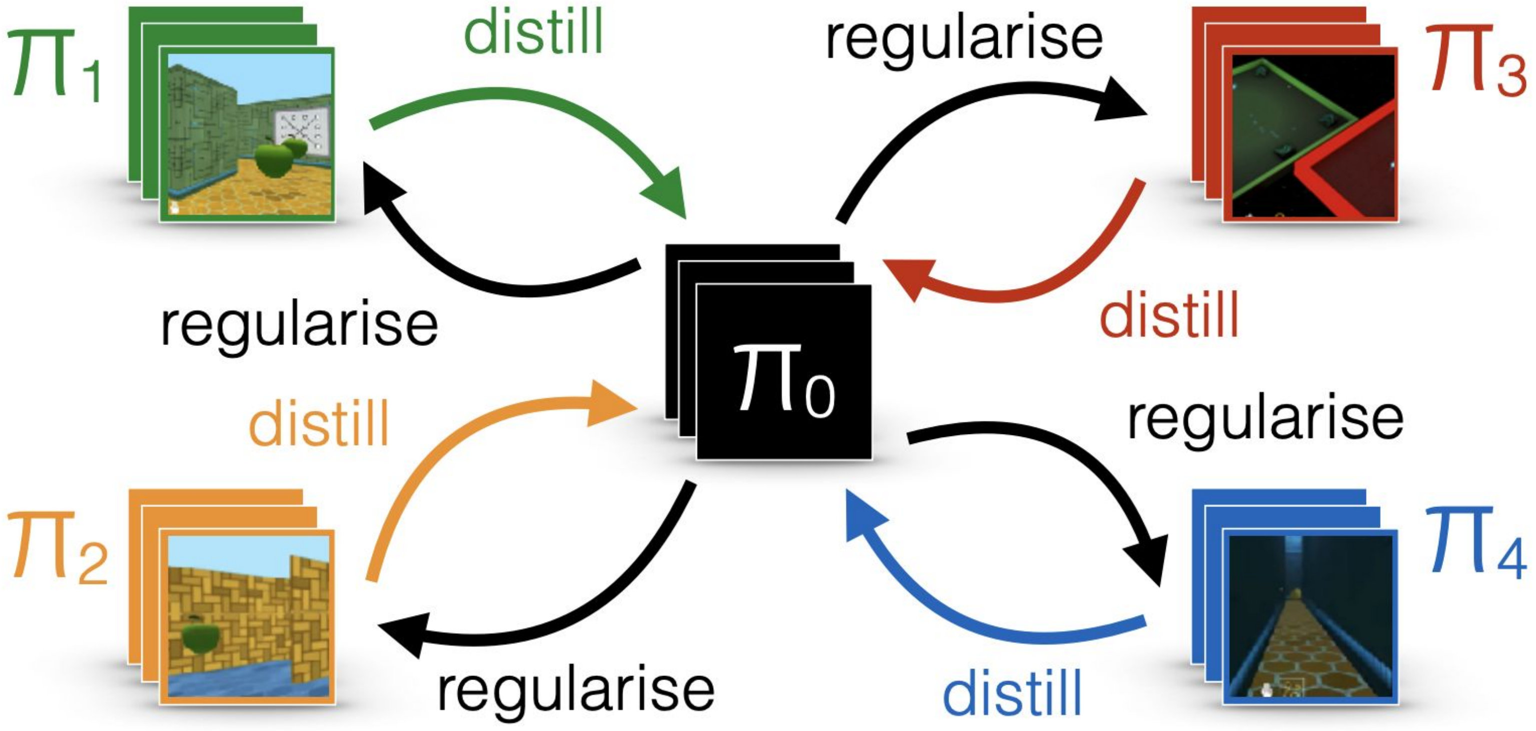


Image taken from [1707.04175] Distral: Robust Multitask Reinforcement Learning

Case II - Shared state and action space

- Distral [5] shares knowledge via distillation.
- Knowledge can also be shared by sharing parameters.

[5]: [1707.04175] Distral: Robust Multitask Reinforcement Learning

Sharing Parameters

One policy per task

π_1

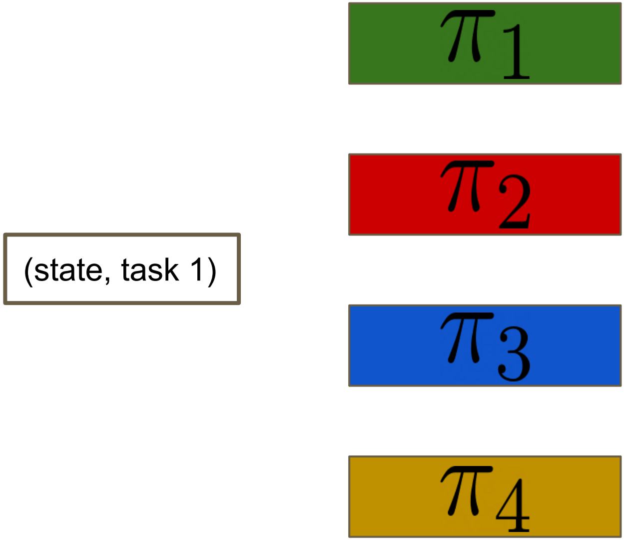
π_2

π_3

π_4

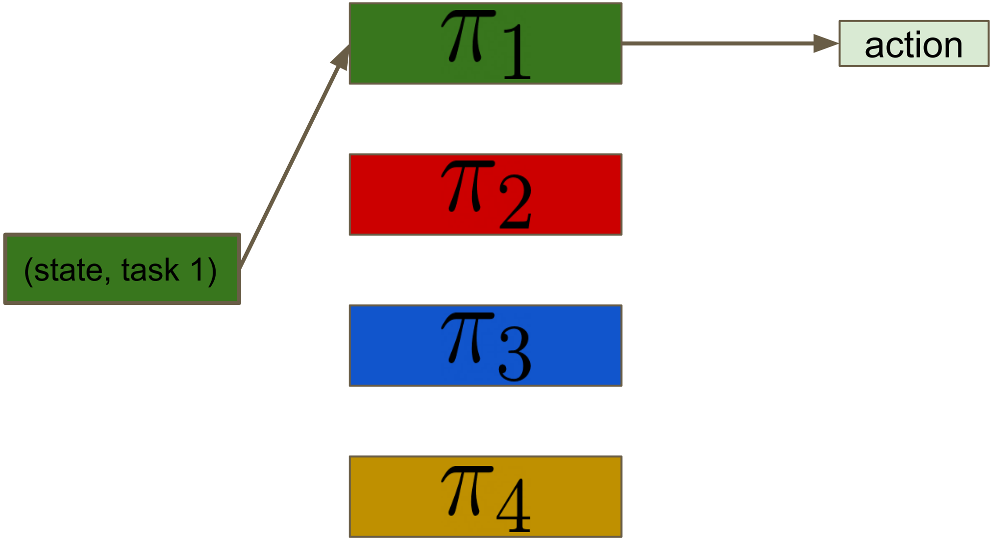
Sharing Parameters

One policy per task



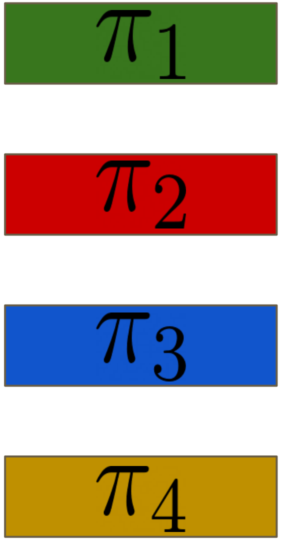
Sharing Parameters

One policy per task

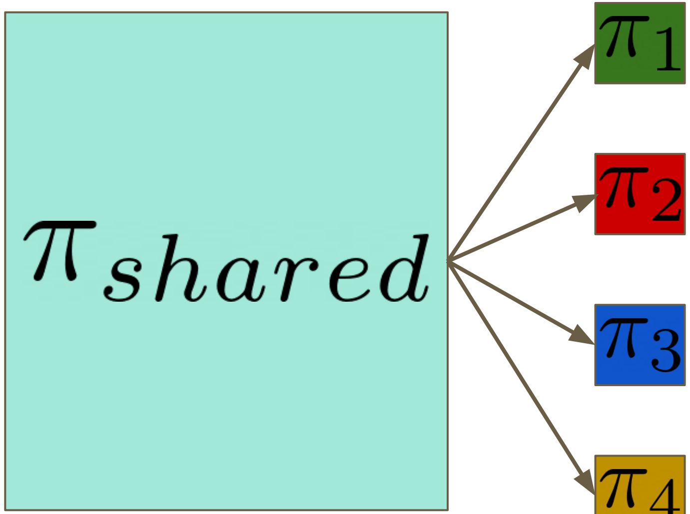


Sharing Parameters

One policy per task

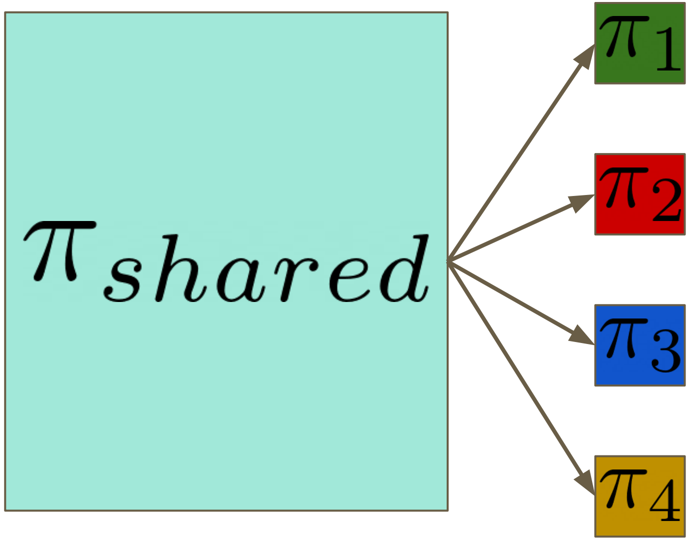


Task specific policy heads

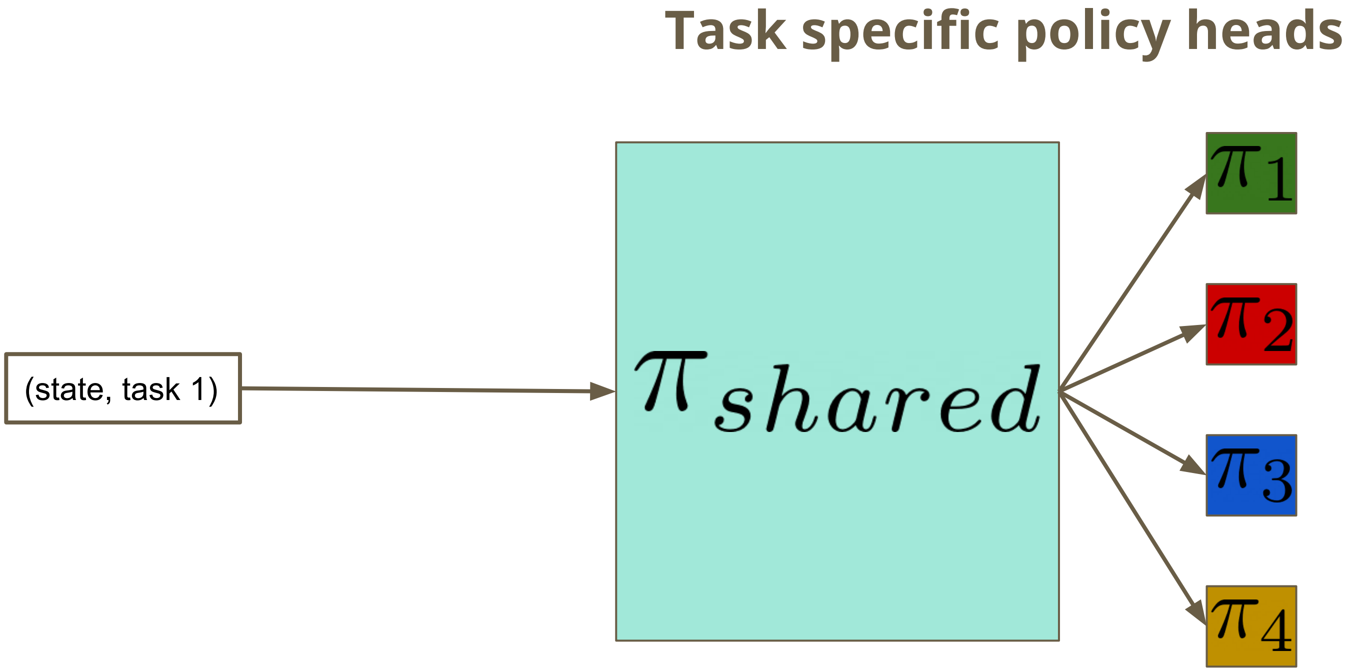


Sharing Parameters

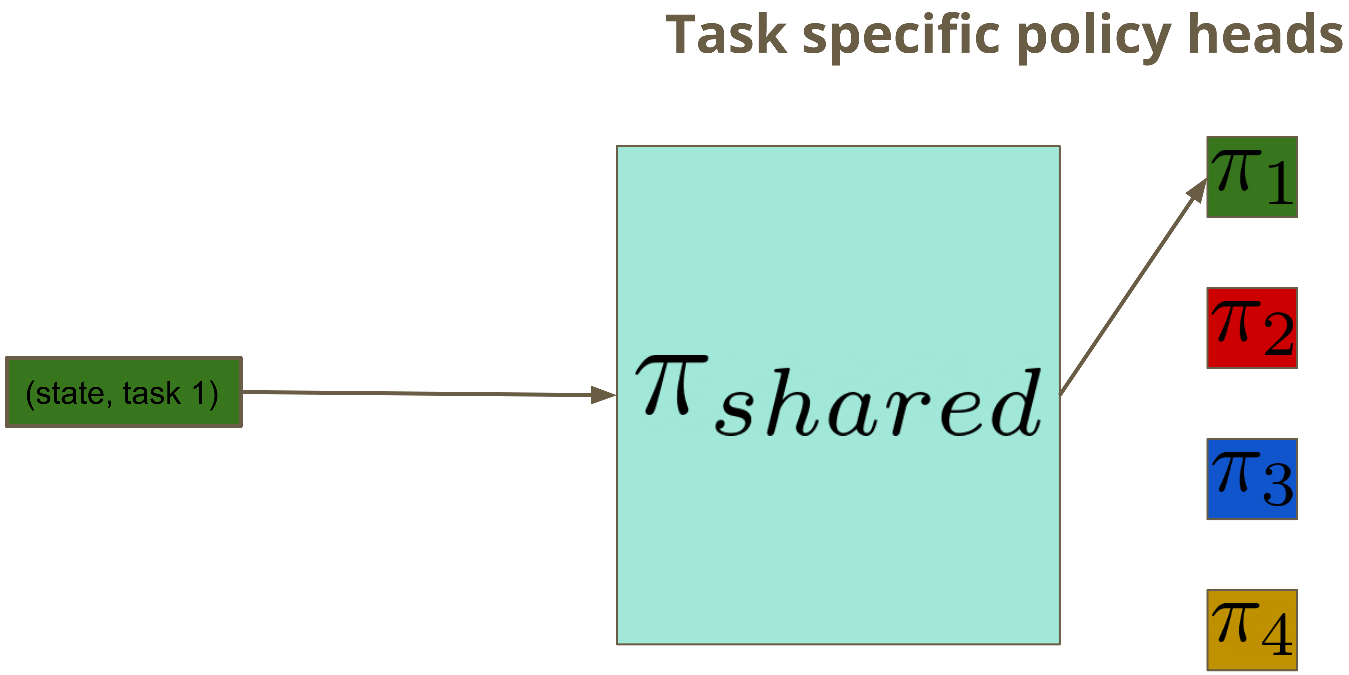
Task specific policy heads



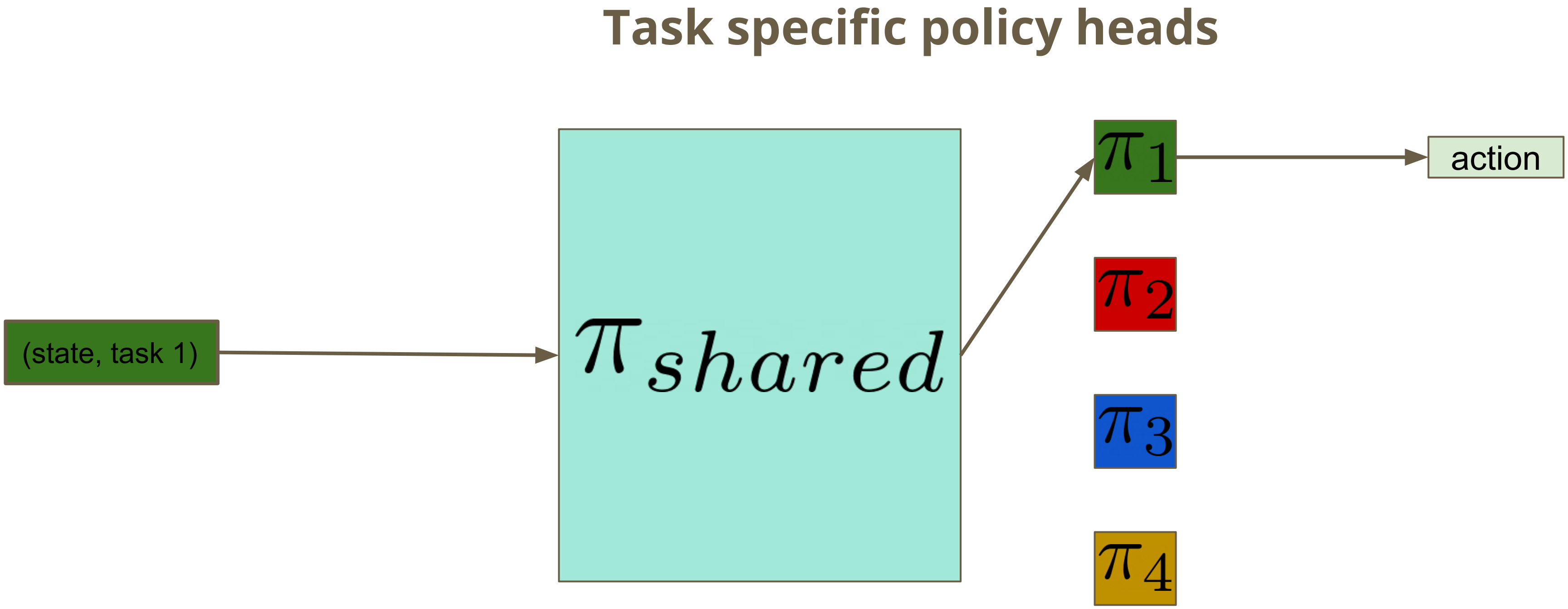
Sharing Parameters



Sharing Parameters



Sharing Parameters



Sharing Parameters

One agent per task

$$enc_1 \rightarrow \pi_1$$

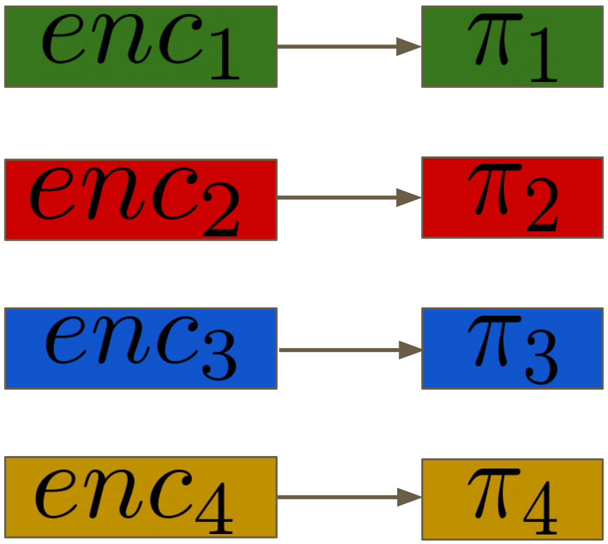
$$enc_2 \rightarrow \pi_2$$

$$enc_3 \rightarrow \pi_3$$

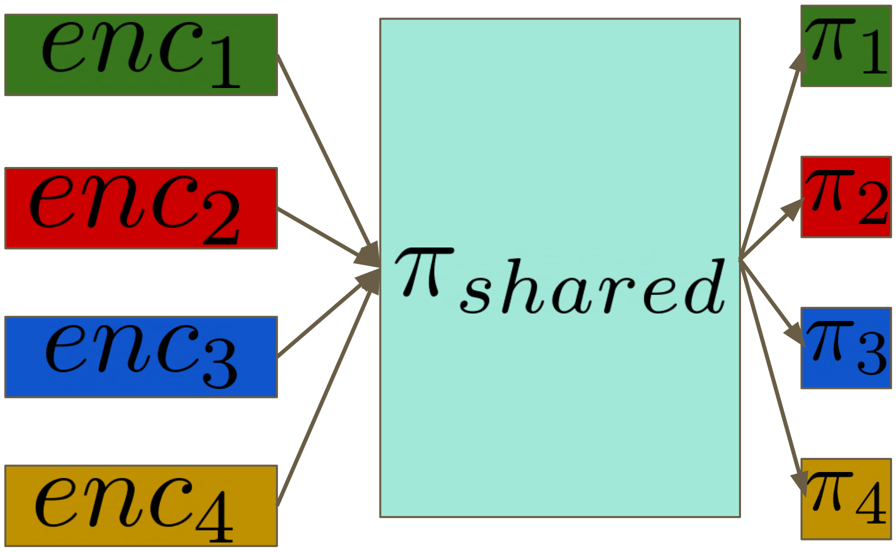
$$enc_4 \rightarrow \pi_4$$

Sharing Parameters

One agent per task

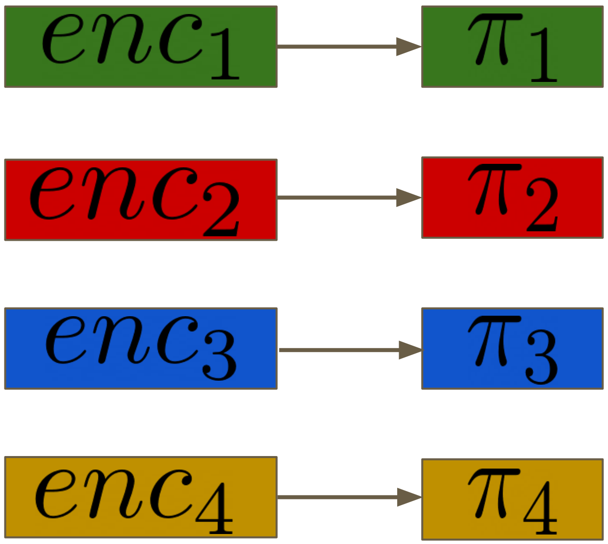


Task specific policy heads and encoders

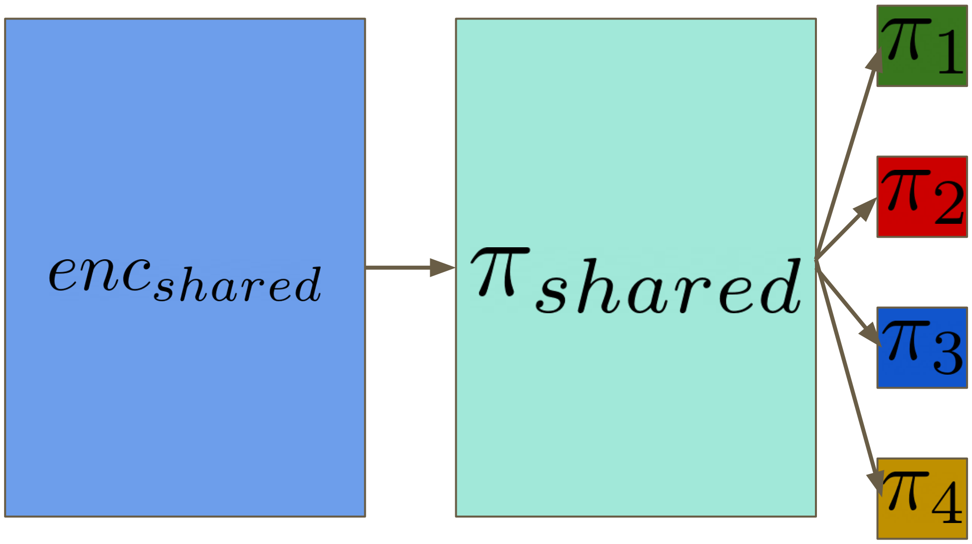


Sharing Parameters

One agent per task



Task specific policy heads and shared encoders



Sharing Parameters

Task Encoder

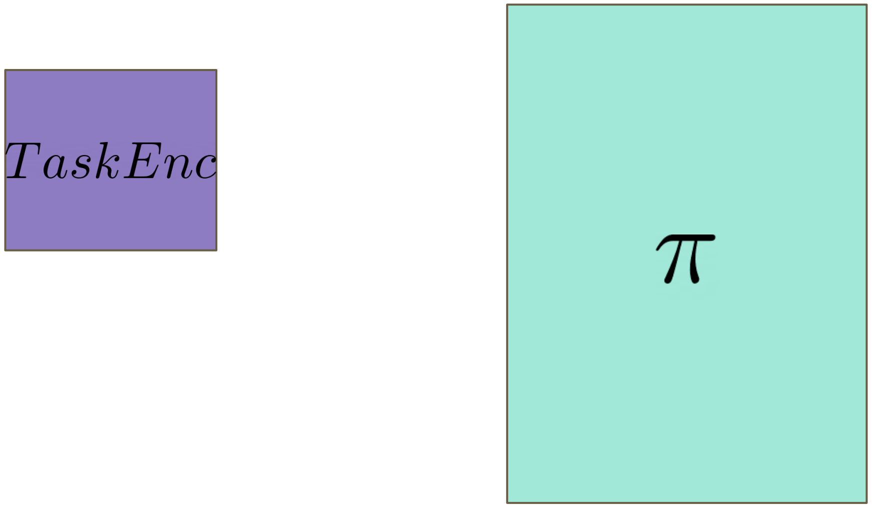
1. Learn a representation for the task.

2. If we do not know anything about the relation between different tasks, a common choice is to represent the tasks with a one-hot vector.

3. An embedding layer (followed by feed-forward networks) can be used to encode the task.

Sharing Parameters

Policy and a task encoder

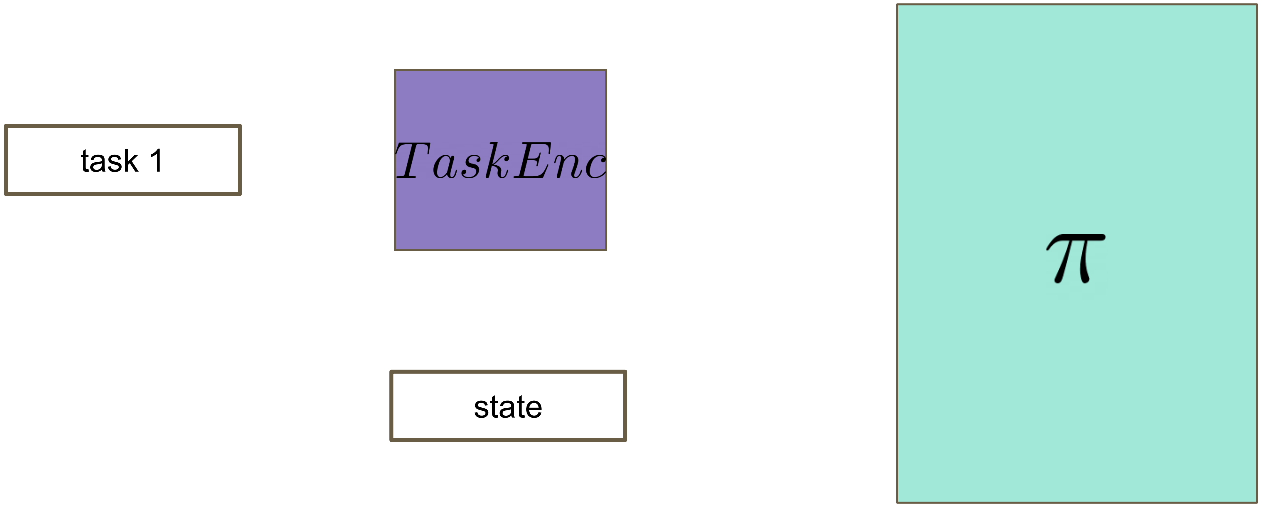


Sharing Parameters

Policy and a task encoder

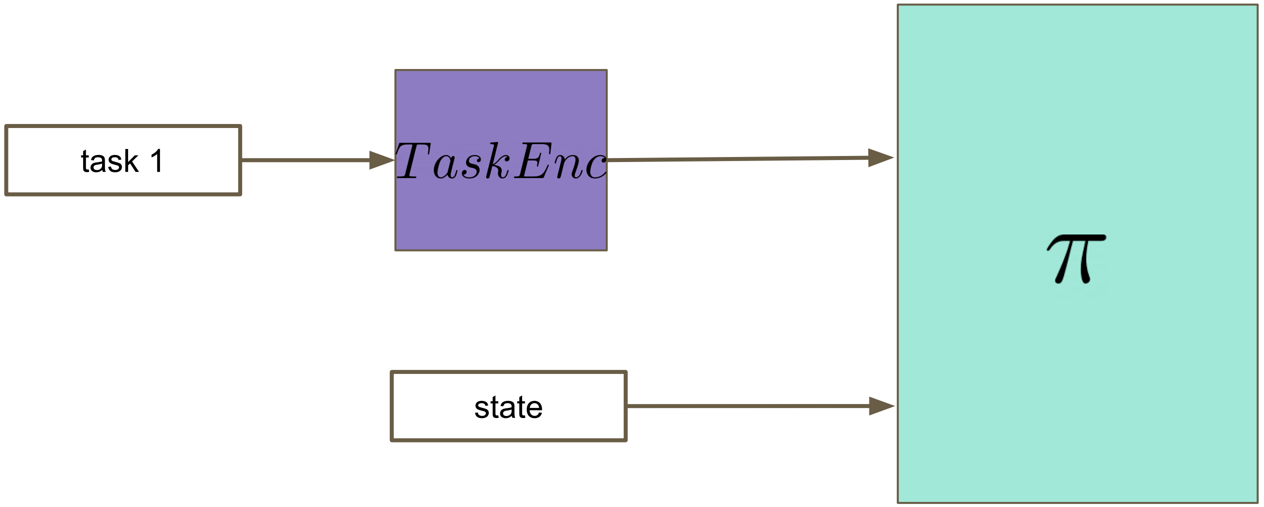
Sharing Parameters

Policy and a task encoder



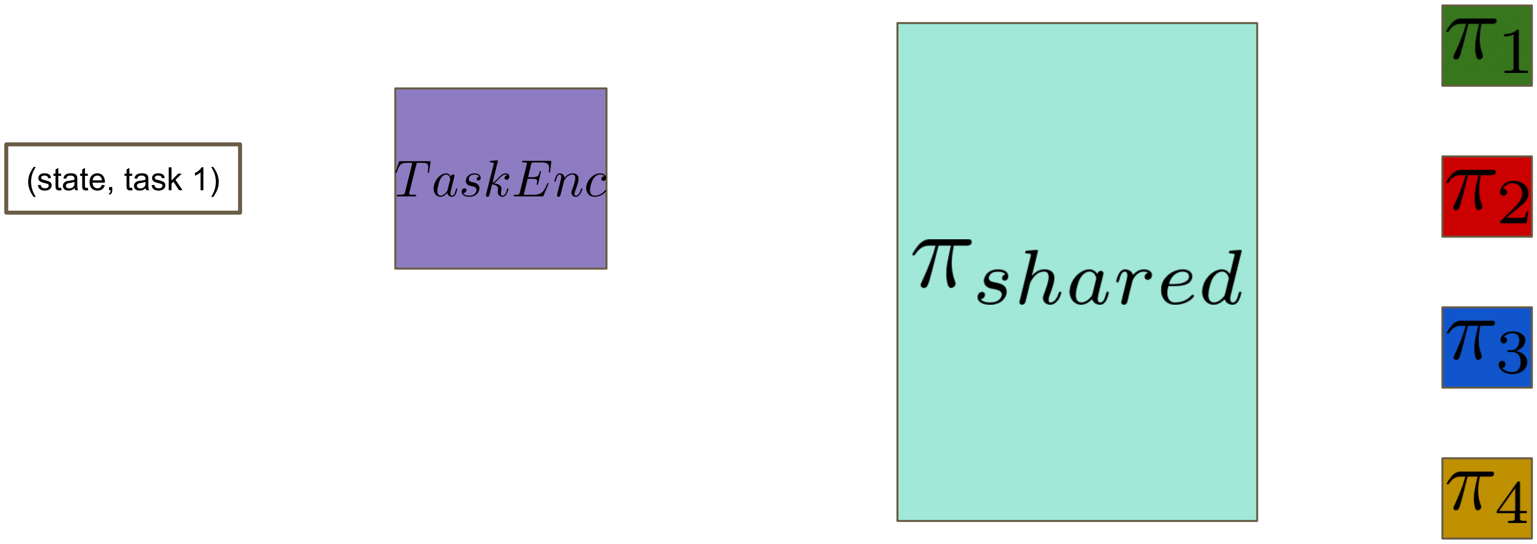
Sharing Parameters

Policy and a task encoder



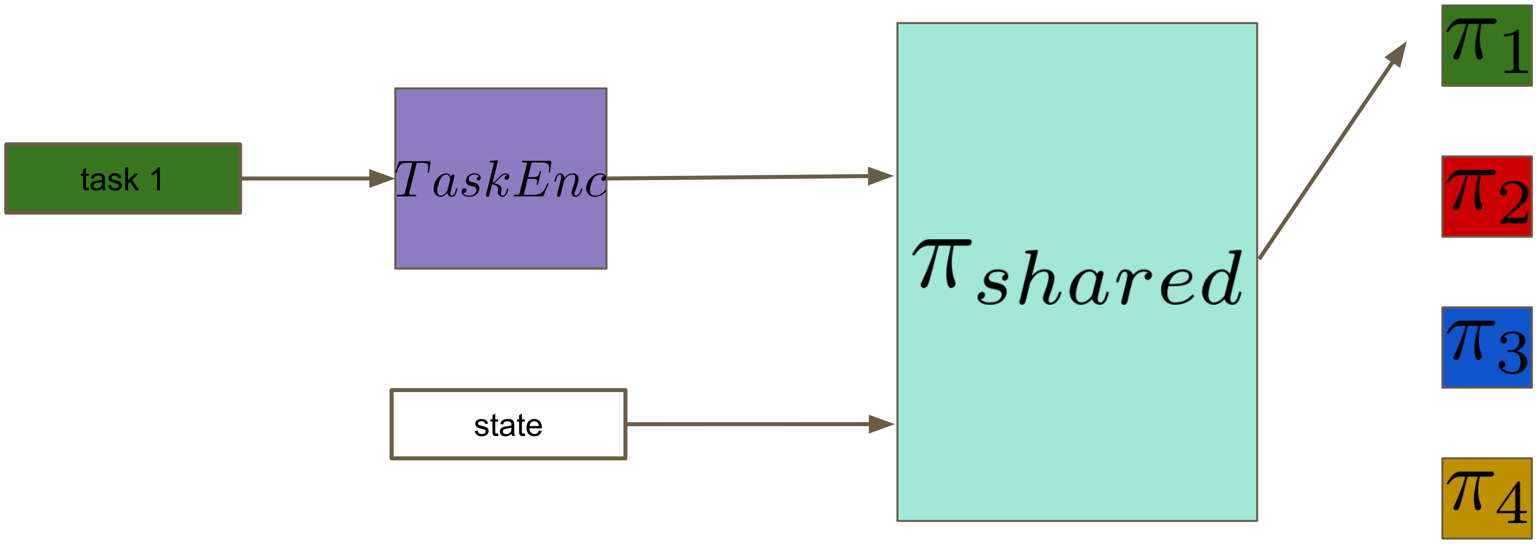
Sharing Parameters

Policy with task specific heads and a task encoder



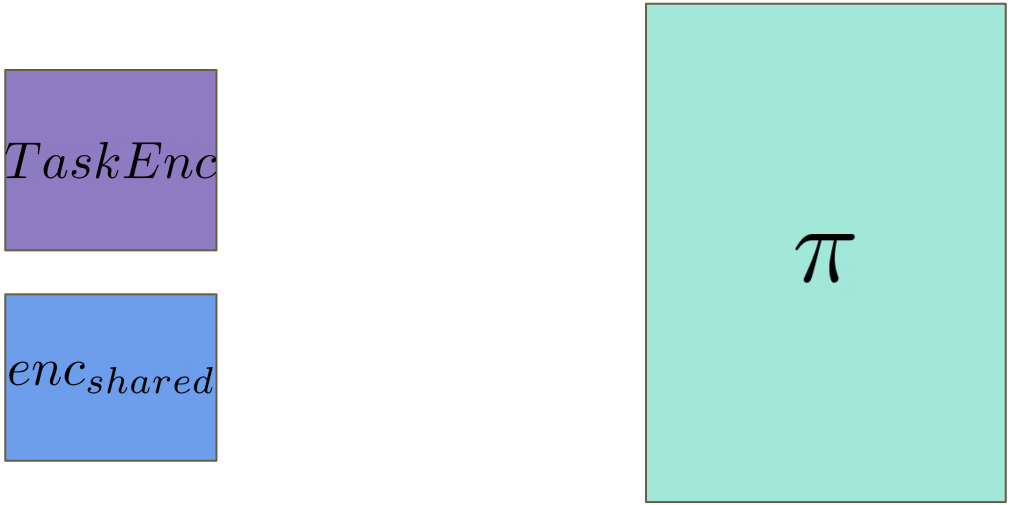
Sharing Parameters

Policy with task specific heads and a task encoder



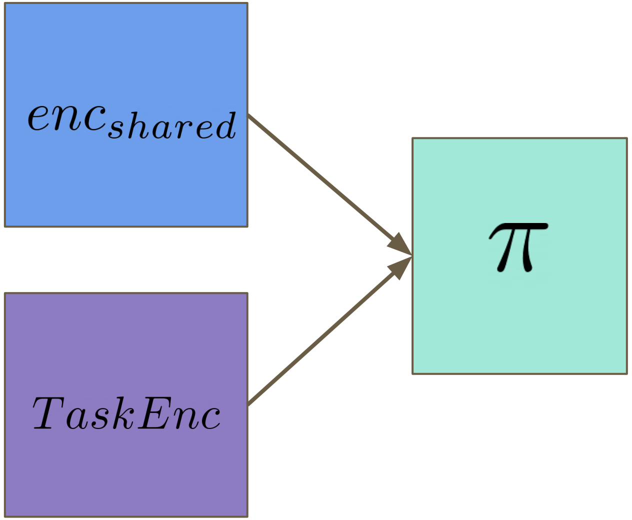
Sharing Parameters

Policy and shared encoder and a task encoder



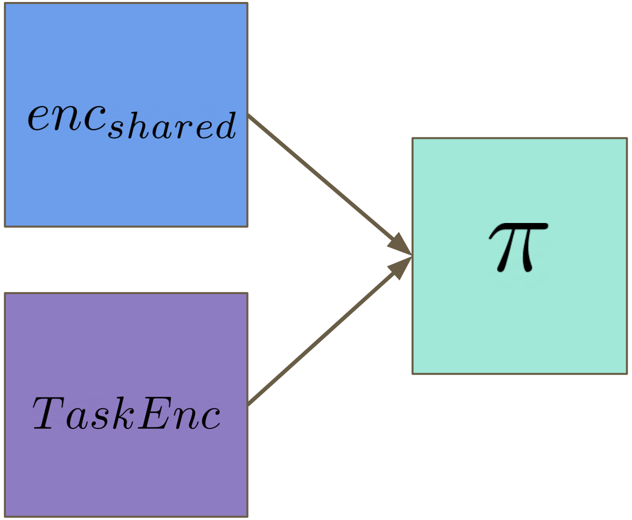
Sharing Parameters

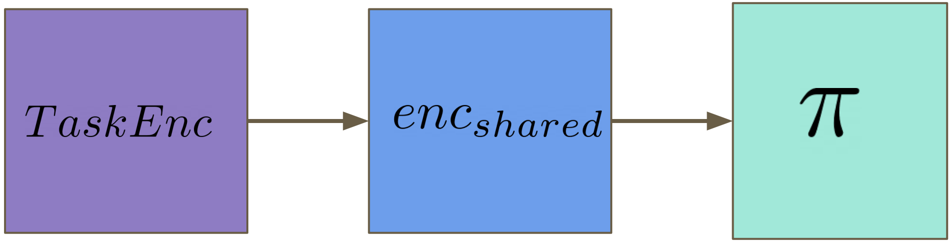
Some components are task specific, some are shared and they are arranged in different ways.



Sharing Parameters

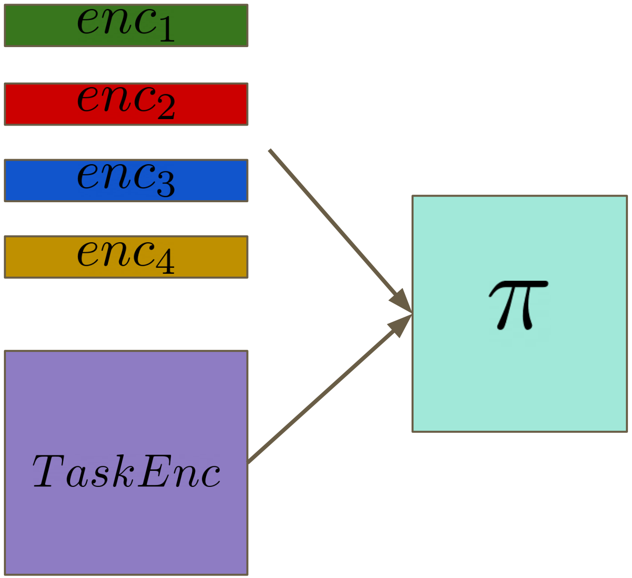
Some components are task specific, some are shared. Components can be arranged in different ways.





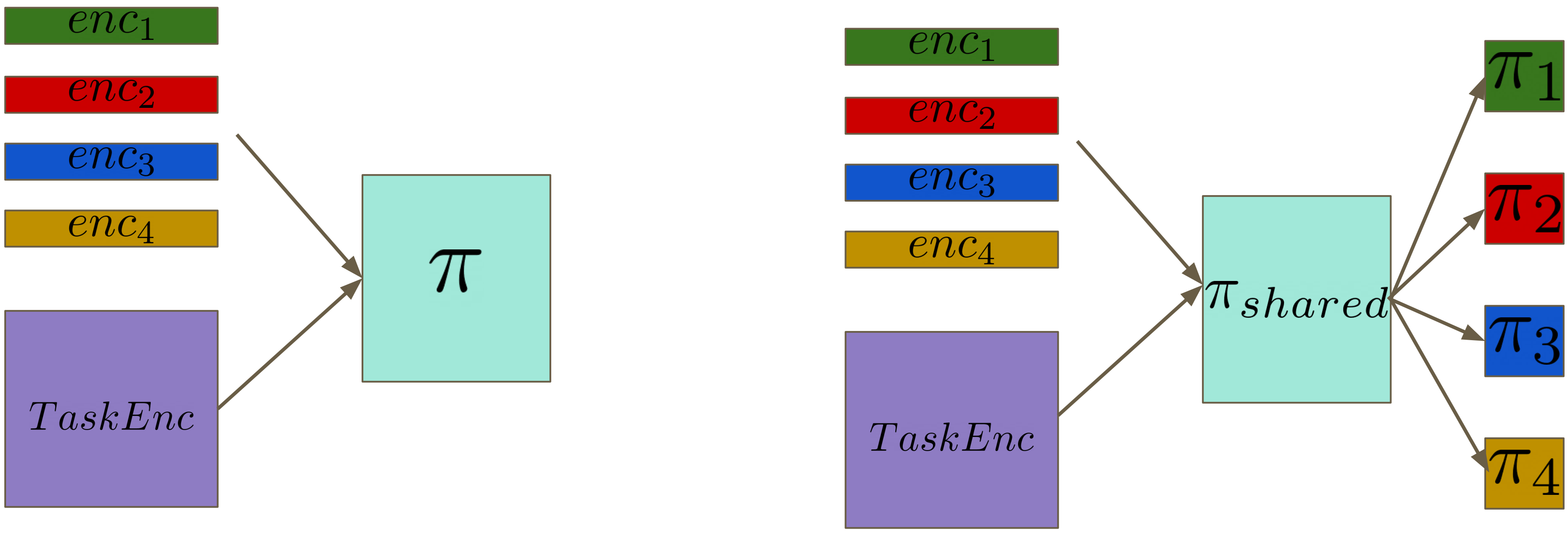
Sharing Parameters

Some components are task specific, some are shared. Components can be arranged in different ways.



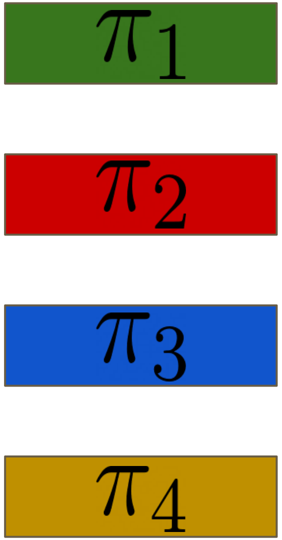
Sharing Parameters

Some components are task specific, some are shared. Components can be arranged in different ways.

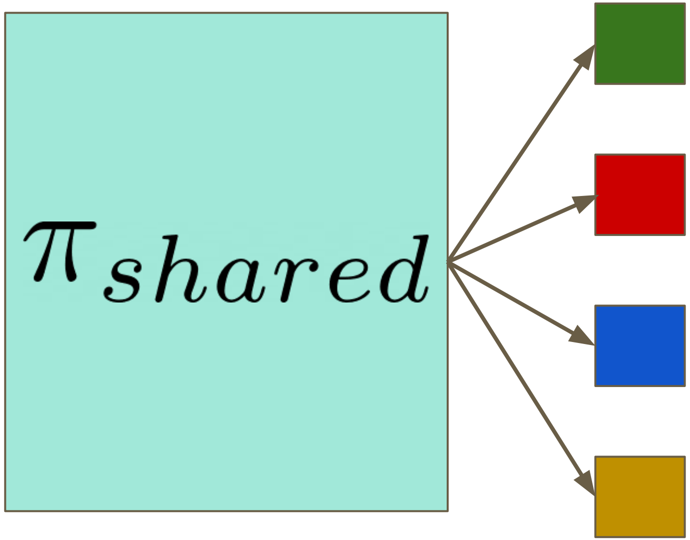


Sharing Parameters

One policy per task



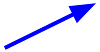
Task specific exploration bonus



Limitations: Negative Interference

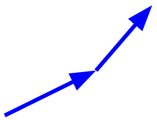
Limitations: Negative Interference

Single Task



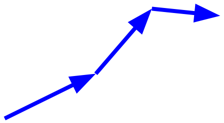
Limitations: Negative Interference

Single Task



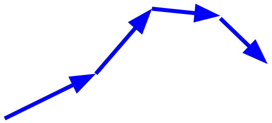
Limitations: Negative Interference

Single Task



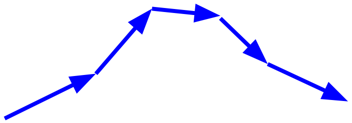
Limitations: Negative Interference

Single Task



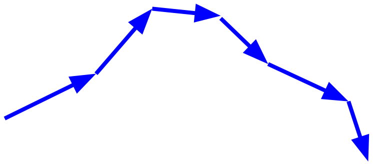
Limitations: Negative Interference

Single Task



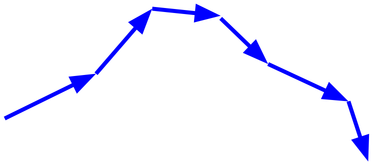
Limitations: Negative Interference

Single Task

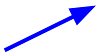


Limitations: Negative Interference

Single Task

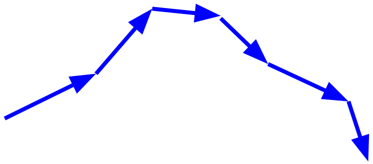


Multi Task

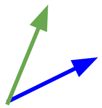


Limitations: Negative Interference

Single Task

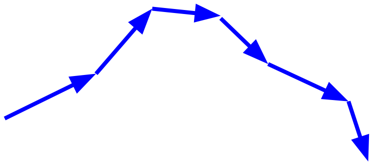


Multi Task

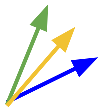


Limitations: Negative Interference

Single Task

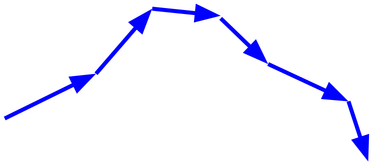


Multi Task

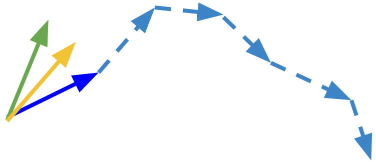


Limitations: Negative Interference

Single Task

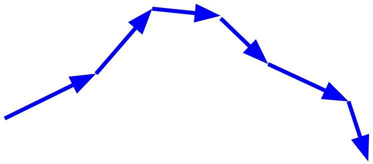


Multi Task

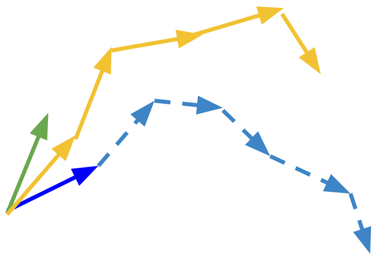


Limitations: Negative Interference

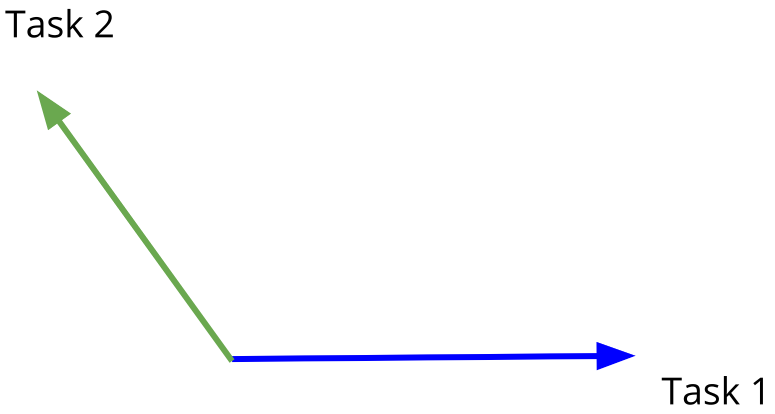
Single Task

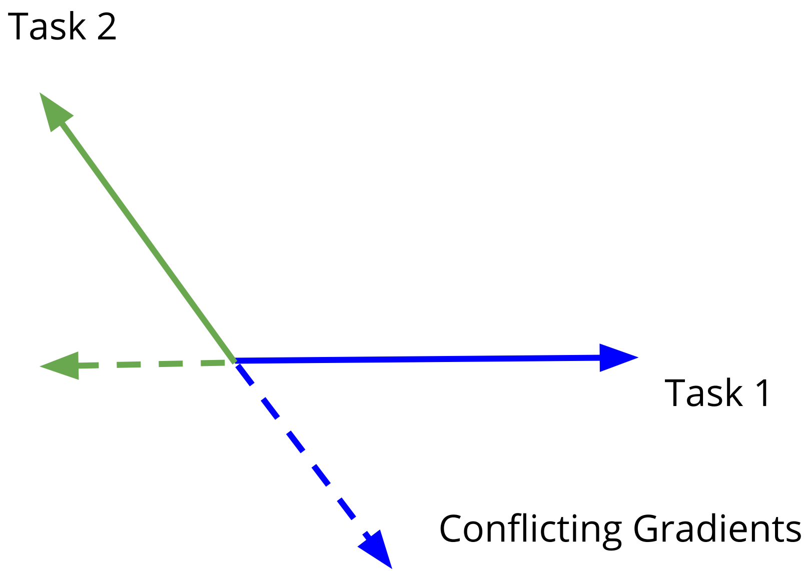


Multi Task

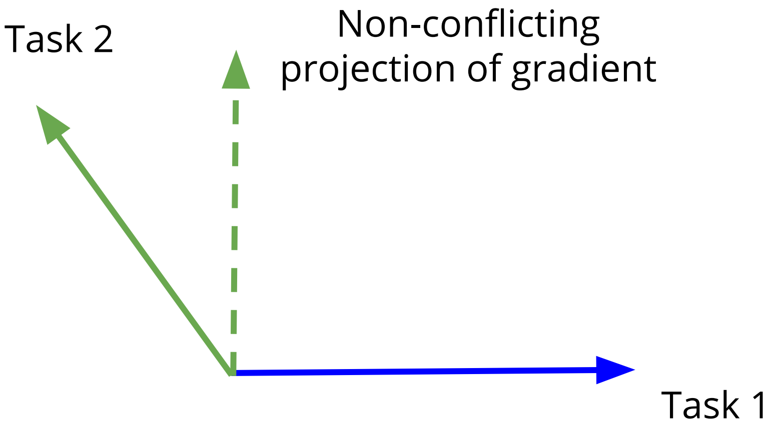


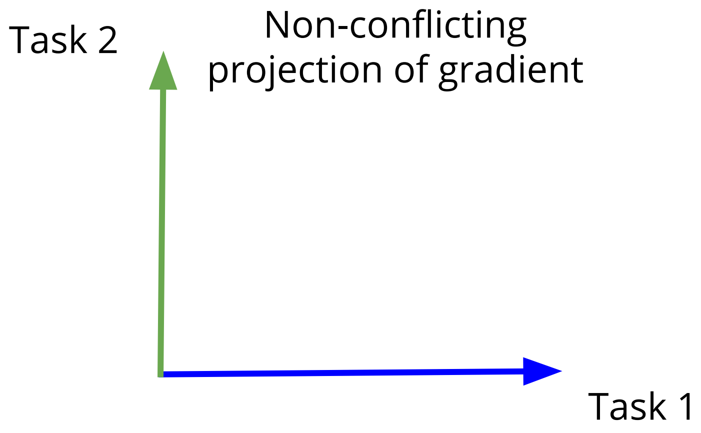
Limitations: Negative Interference





Gradient Surgery for Multi-Task Learning





[9]: [2001.06782] Gradient Surgery for Multi-Task Learning

Limitations: Loss Balancing

- Not all tasks are equal - some are easy and some are hard.
- The scale of loss for different tasks could be different.
- One task (or a subset of tasks) can dominate training, thus hindering learning on the other tasks.
- We want to train the different tasks at similar rate.

Gradient Normalization for Adaptive Loss Balancing

$$L = \sum_{i \in \{1, \cdots n\}} \alpha_i \times L_i$$

[7]: [1711.02257] GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks

Gradient Normalization for Adaptive Loss Balancing

$$L = \sum_{\boxed{i \in \{1, \cdots n\}}} \alpha_i \times L_i$$

Task Index

[7]: [1711.02257] GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks

Gradient Normalization for Adaptive Loss Balancing

$$L = \sum_{i \in \{1, \cdots n\}} \boxed{\alpha_i} \times L_i$$

Weight for the loss from the i^th task (Hyperparameter)

[7]: [1711.02257] GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks

Gradient Normalization for Adaptive Loss Balancing

$$L = \sum_{i \in \{1, \cdots n\}} \alpha_i \times \boxed{L_i}$$

Loss from the i^th task

[7]: [1711.02257] GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks

Gradient Normalization for Adaptive Loss Balancing

$$\boxed{L} = \sum_{i \in \{1, \cdots n\}} \alpha_i \times L_i$$

Multitask Learning Loss

[7]: [1711.02257] GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks

Gradient Normalization for Adaptive Loss Balancing

$$L = \sum_{i \in \{1, \cdots n\}} \boxed{\alpha_i} \times L_i$$

Hyperparameter

$$L = \sum_{i \in \{1, \cdots n\}} \boxed{w_i} \times L_i$$

Learned

[7]: [1711.02257] GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks

Gradient Normalization for Adaptive Loss Balancing

$$G_i = ||\nabla(w_i \times L_i)||_2$$

L2 norm of the gradient for the i^{th} task

[7]: [1711.02257] GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks

Gradient Normalization for Adaptive Loss Balancing

$$G_i = ||\nabla(w_i \times L_i)||_2$$

$$G^{mean} = mean(G_i)$$

Mean of the L2 norms

[7]: [1711.02257] GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks

Gradient Normalization for Adaptive Loss Balancing

$$G_i = ||\nabla(w_i \times L_i)||_2$$

$$G^{mean} = mean(G_i)$$

$$L_w = \sum_{i \in \{1, \cdots n\}} |G_i - G_{mean} \times \boxed{r_i}^{\alpha}|_1$$

Relates to the learning rate of i-th task

[7]: [1711.02257] GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks

Gradient Normalization for Adaptive Loss Balancing

$$G_i = ||\nabla(w_i \times L_i)||_2$$

$$G^{mean} = mean(G_i)$$

$$L_w = \sum_{i \in \{1, \cdots n\}} |G_i - G_{mean} \times r_i^{\boxed{\alpha}}|_1$$ **hyperparameter**

[7]: [1711.02257] GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks

What we have seen so far

1. Recipes to train an agent on n-tasks.

What we have seen so far

1. Recipes to train an agent on n-tasks.

2. What if we want the agent to perform well on an “unseen” (or new) task?

    a. For example, we train an autonomous car on concrete roads and gravel roads and now we want to see how well it runs on an ice road.

    b. We may have very little data/resources to train on the new task (“few-shot generalization”) or no data/resources at all for the new task (“zero-shot generalization”).

What we have seen so far

1. What if we want to the agent to perform well on an “unseen” (or new) task?

    a. For example, we train an autonomous car on concrete roads and gravel roads and now we want to see how well it runs on an ice road.

    b. Approaches where we have task specific components (like encoders/policy heads etc) can not be used.

    c. Approaches like Distral, PCGrad etc could be used (in theory) but may not work well in practice.

What we have seen so far

1. What if we want to the agent to perform well on an “unseen” (or new) task?

    a. We need to make additional assumptions about the tasks.

    b. For example, in the car example, we could argue that the dynamics of the car on different surfaces are related (though not the same).

    c. This seems to be a valid (and useful) assumption even if we do not care about the unseen surfaces.

Case III - State & action spaces are shared and transition dynamics are related

- Driving a car on different surfaces - concrete, gravel roads, wet, snow-covered etc

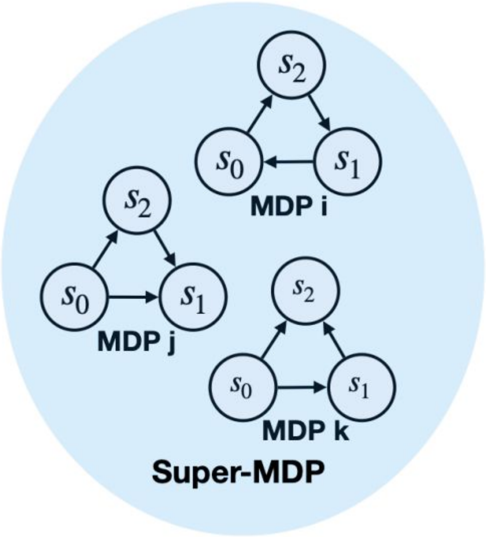
- Hidden-Parameter MDP [10]

Hidden Parameter MDP

1. We have n-tasks.

2. Each task maps to a MDP.

3. With n-tasks, we have n MDPs.



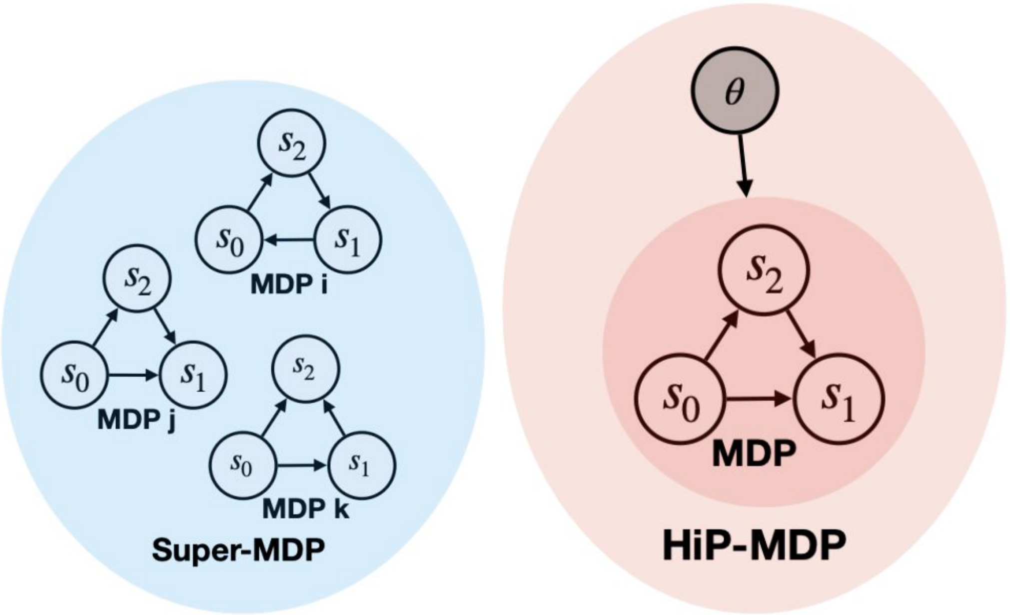
Hidden Parameter MDP

1. We have n-tasks.

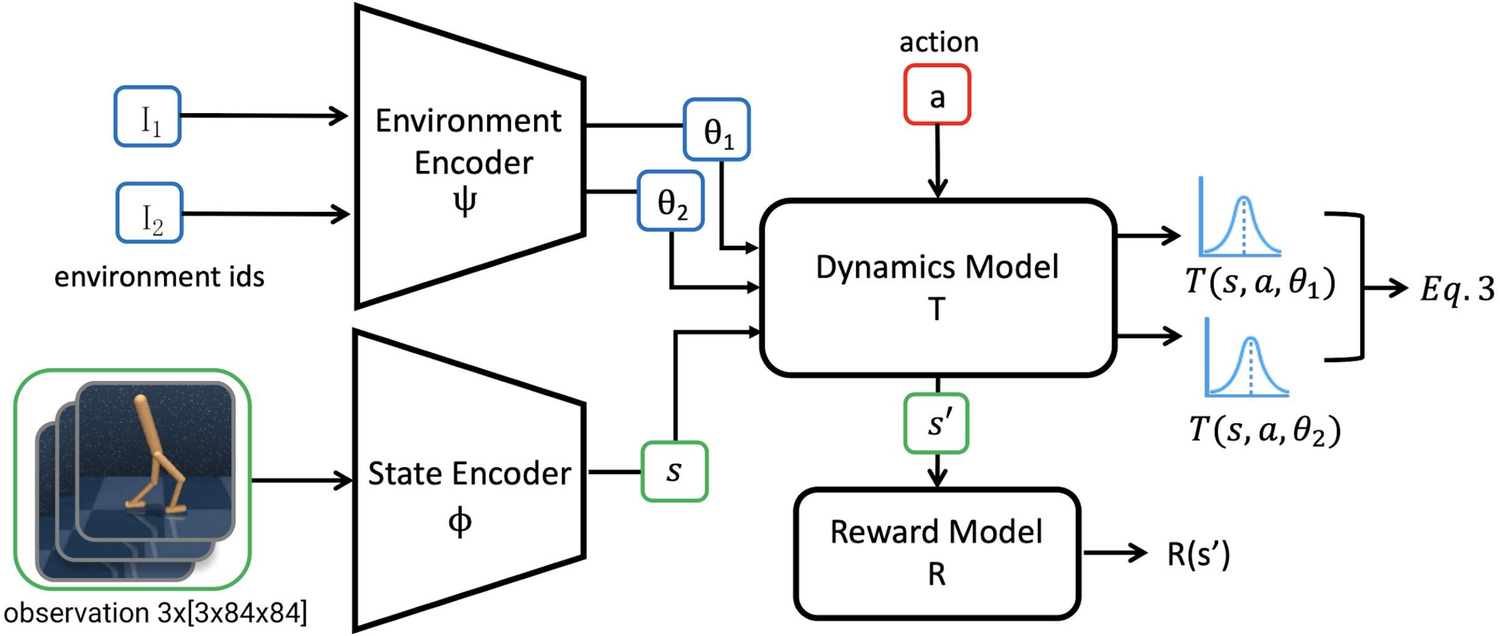
2. Each task maps to a MDP.

3. With n-tasks, we have n MDPs.

4. These n MDPs can be viewed as a single HiP-MDP with Θ as the hidden-parameter.

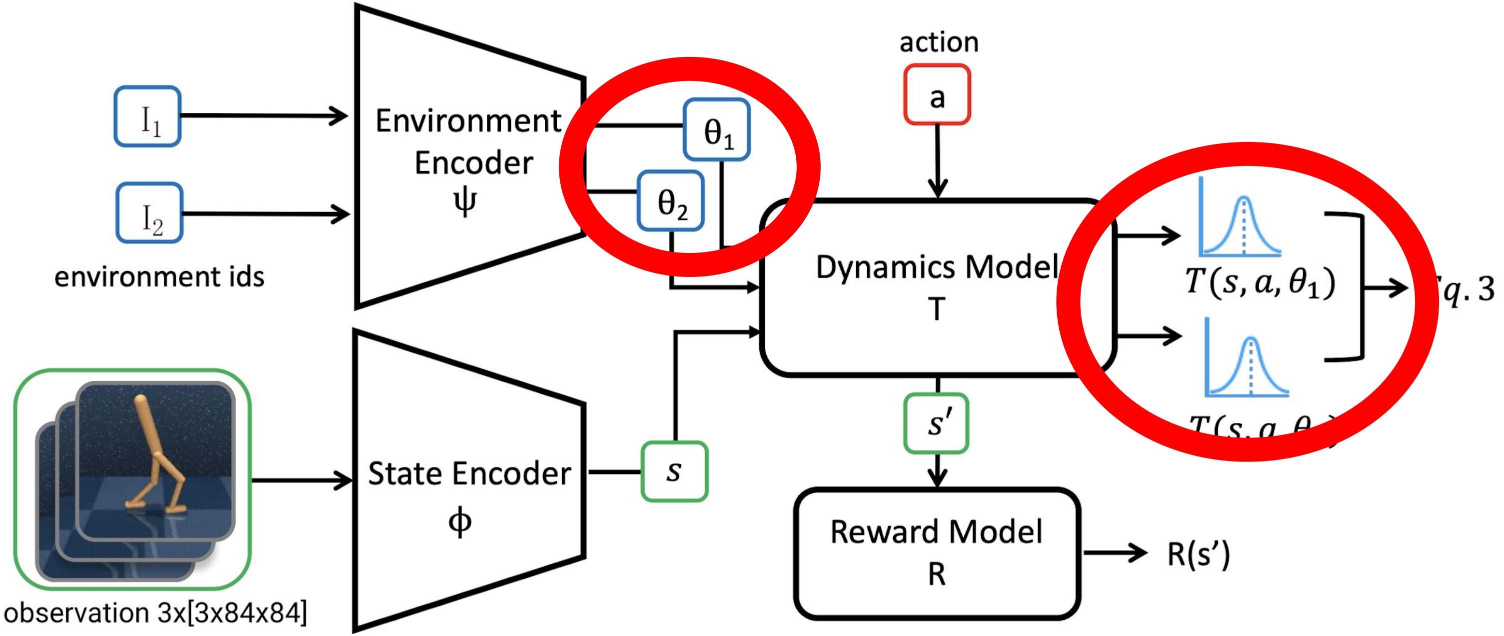


Learning Robust State Abstractions for Hidden-Parameter Block MDPs



Taken from [11]: [2007.07206] Learning Robust State Abstractions for Hidden-Parameter Block MDPs

Learning Robust State Abstractions for Hidden-Parameter Block MDPs



Taken from [11]: [2007.07206] Learning Robust State Abstractions for Hidden-Parameter Block MDPs

Learning Robust State Abstractions for Hidden-Parameter Block MDPs

$$\underbrace{MSE\left(\Big|\Big|\psi(I_1) - \psi(I_2)\Big|\Big|_2, W_2\big(T(s_t^{I_1}, \pi(s_t^{I_1}), \psi(I_1)), T(s_t^{I_2}, \pi(s_t^{I_2}), \psi(I_2))\big)\right)}_{\Theta \text{ learning error}}$$

Taken from [11]: [2007.07206] Learning Robust State Abstractions for Hidden-Parameter Block MDPs

Case IV - State & action spaces are shared and common objects across tasks

- A robotic arm manipulating objects.

- The same arm is used across all the tasks, while the objects can be shared/different across the tasks.

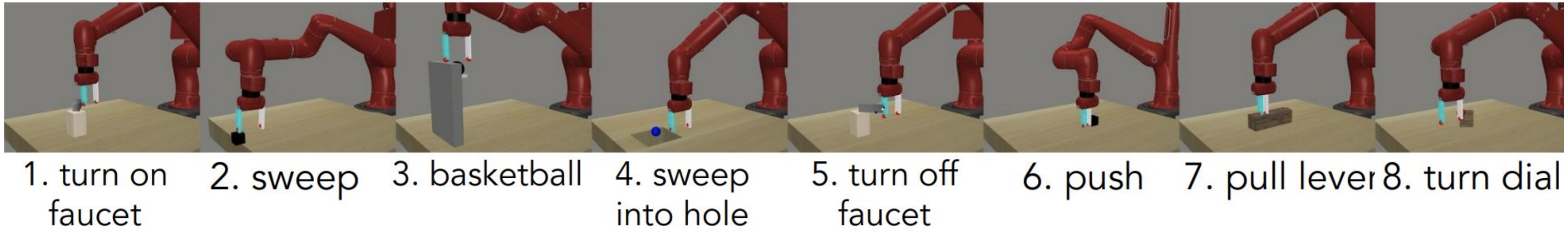


Image taken from [12]: [1910.10897] Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning

Case IV - State & action spaces are shared and common objects across tasks

- The same arm is used across all the tasks, while the objects can be shared/different across the tasks.

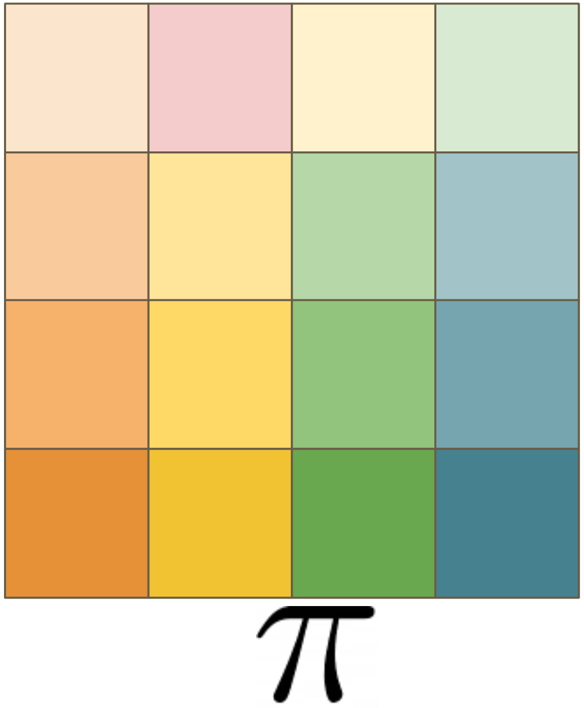
- Useful to share parameters across the different tasks.

Multi-Task Reinforcement Learning with Soft Modularization

1. We have a collection of modules.

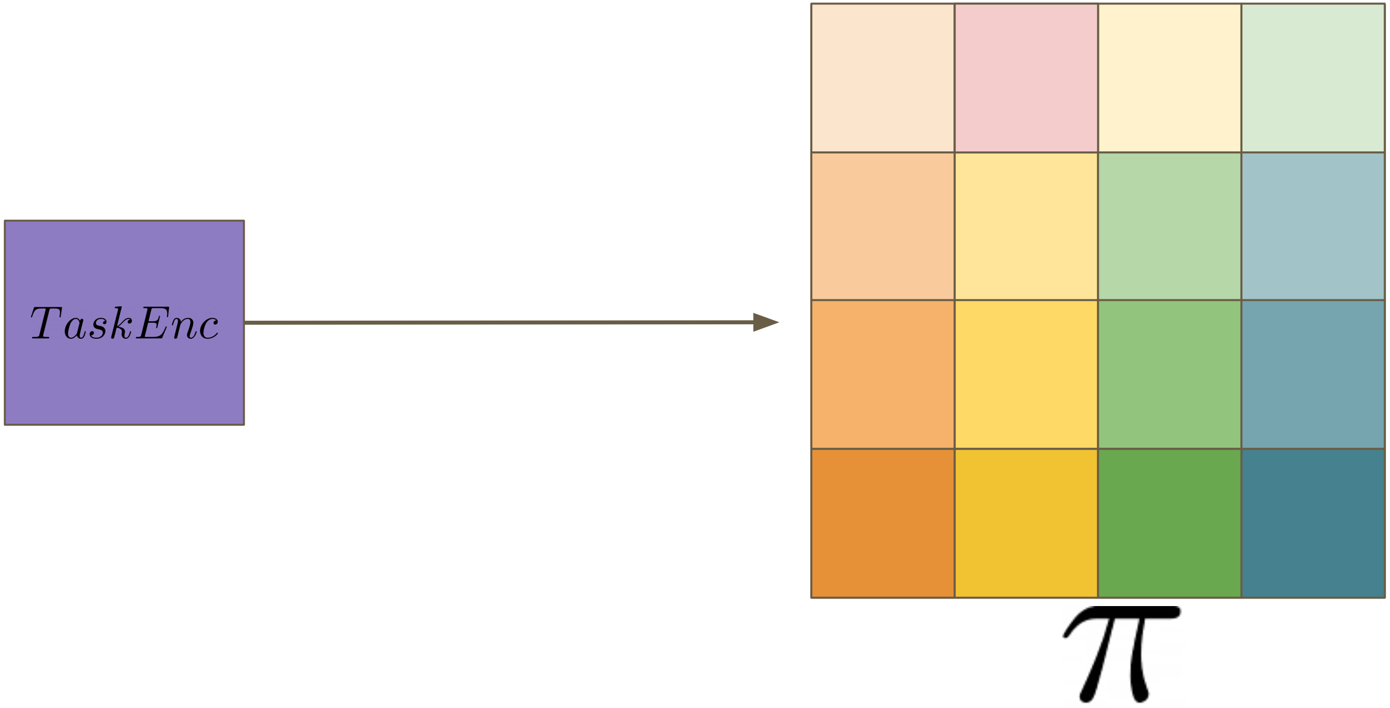
2. When a task is encountered, some of these modules are selected and combined on the fly to instantiate the policy.

3. The policy is thus conditioned on the task representation.



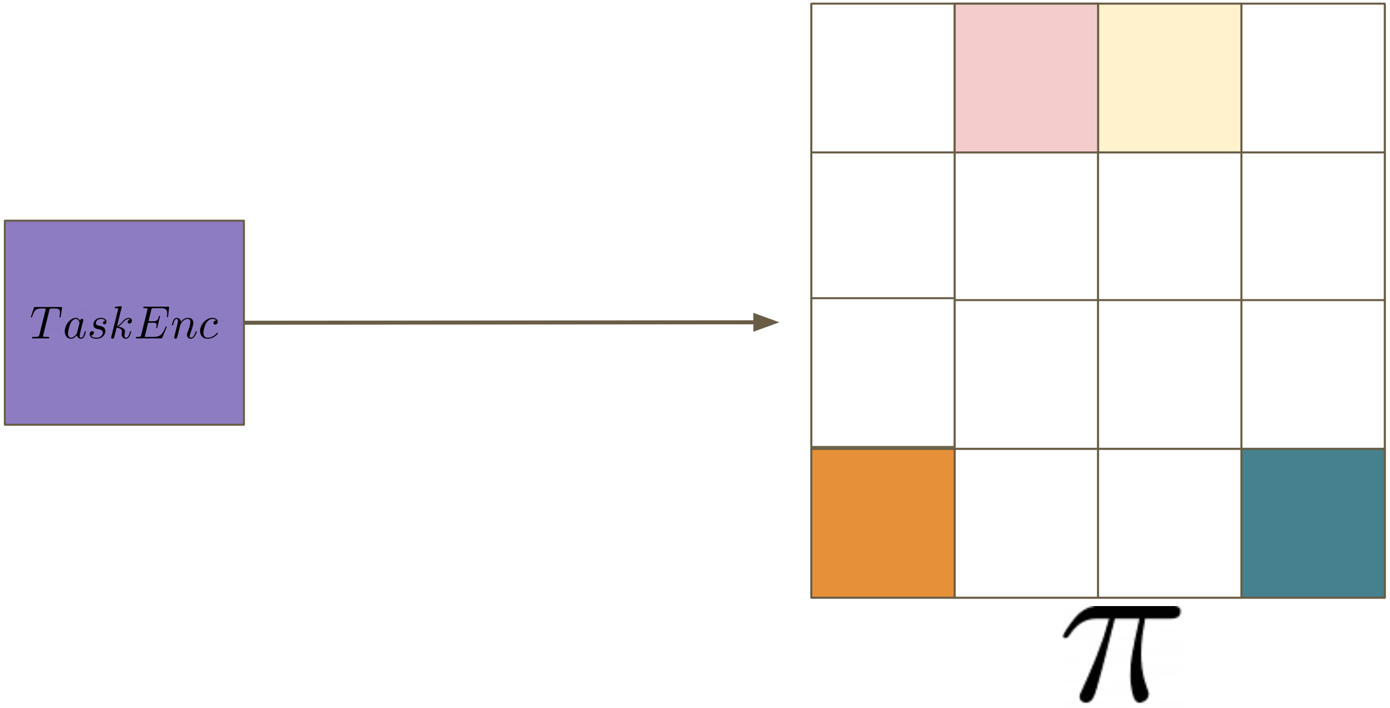
[13]: [2003.13661] Multi-Task Reinforcement Learning with Soft Modularization

Multi-Task Reinforcement Learning with Soft Modularization



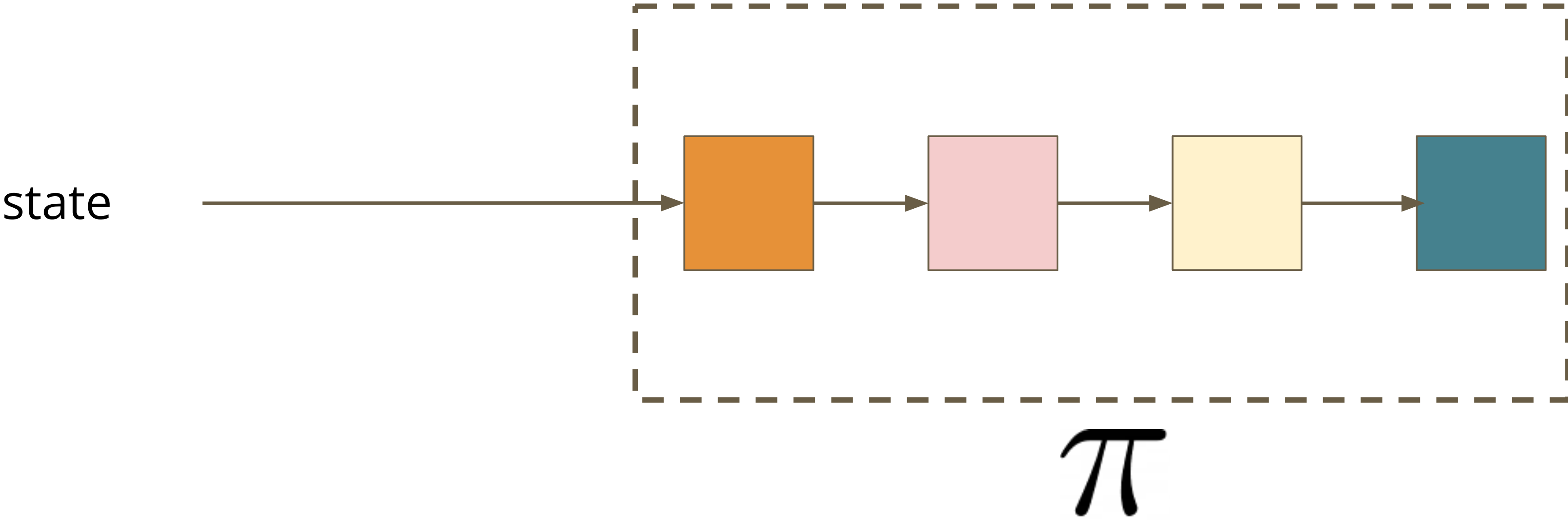
[13]: [2003.13661] Multi-Task Reinforcement Learning with Soft Modularization

Multi-Task Reinforcement Learning with Soft Modularization



[13]: [2003.13661] Multi-Task Reinforcement Learning with Soft Modularization

Multi-Task Reinforcement Learning with Soft Modularization



[13]: [2003.13661] Multi-Task Reinforcement Learning with Soft Modularization

Multi-Task Reinforcement Learning with Soft Modularization

- I explained the idea using hard selection of modules.
- In practice, the method uses soft modularization.

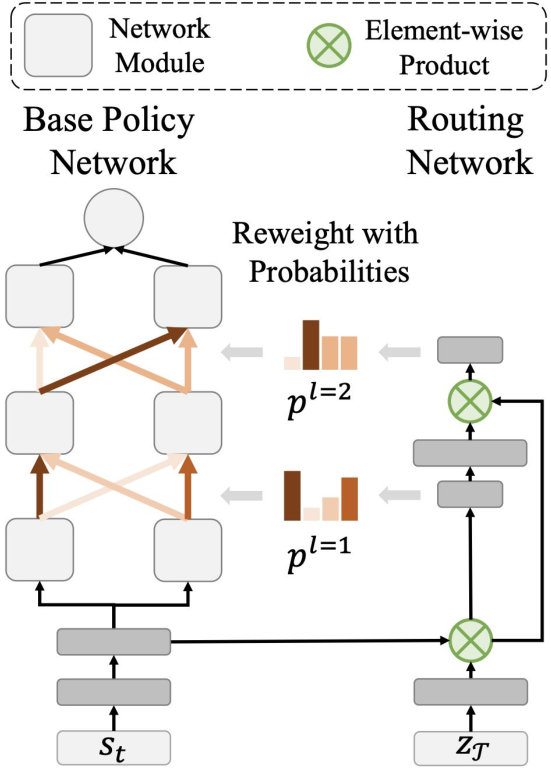


Image taken from [13]: [2003.13661] Multi-Task Reinforcement Learning with Soft Modularization

Case V - State & action spaces are shared and task metadata is available

- We have some additional side information or metadata (which is not required to solve the task).

- However, this side information can be used to infer relationship between tasks.

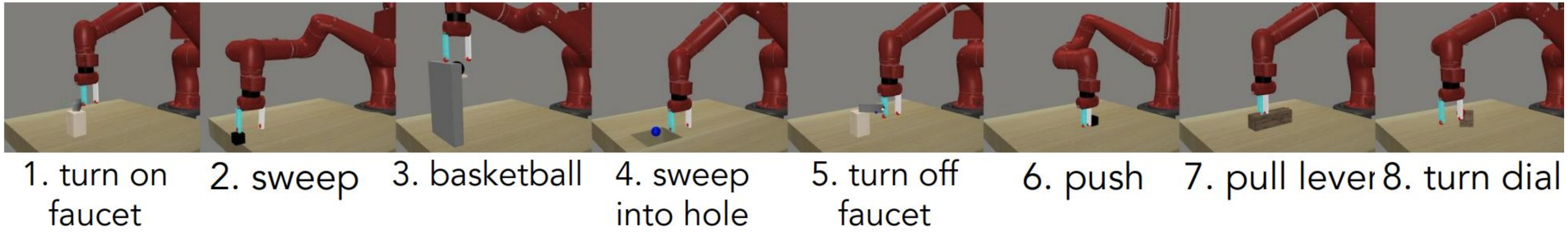


Image taken from [12]: [1910.10897] Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning

Multi-Task Reinforcement Learning with Context-based Representations

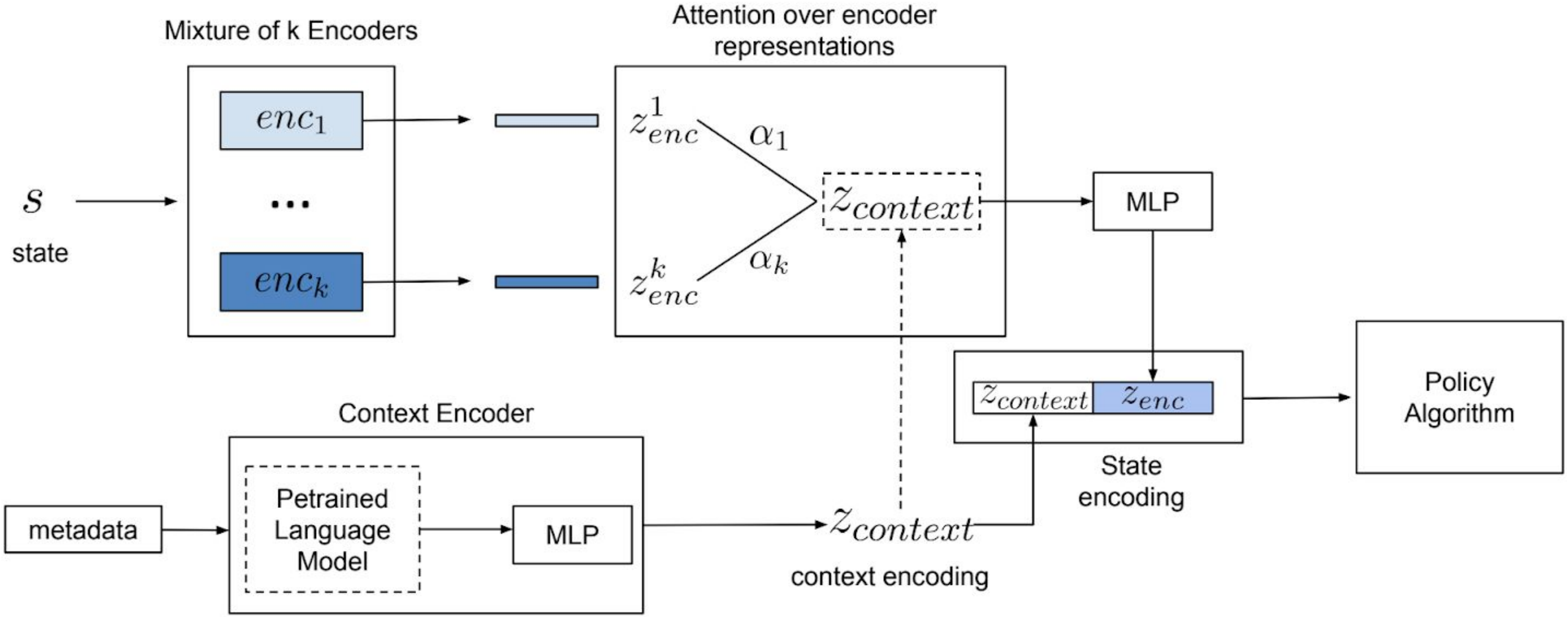


Image taken from [14]: [2102.06177] Multi-Task Reinforcement Learning with Context-based Representations

Multi-Task Reinforcement Learning with Context-based Representations

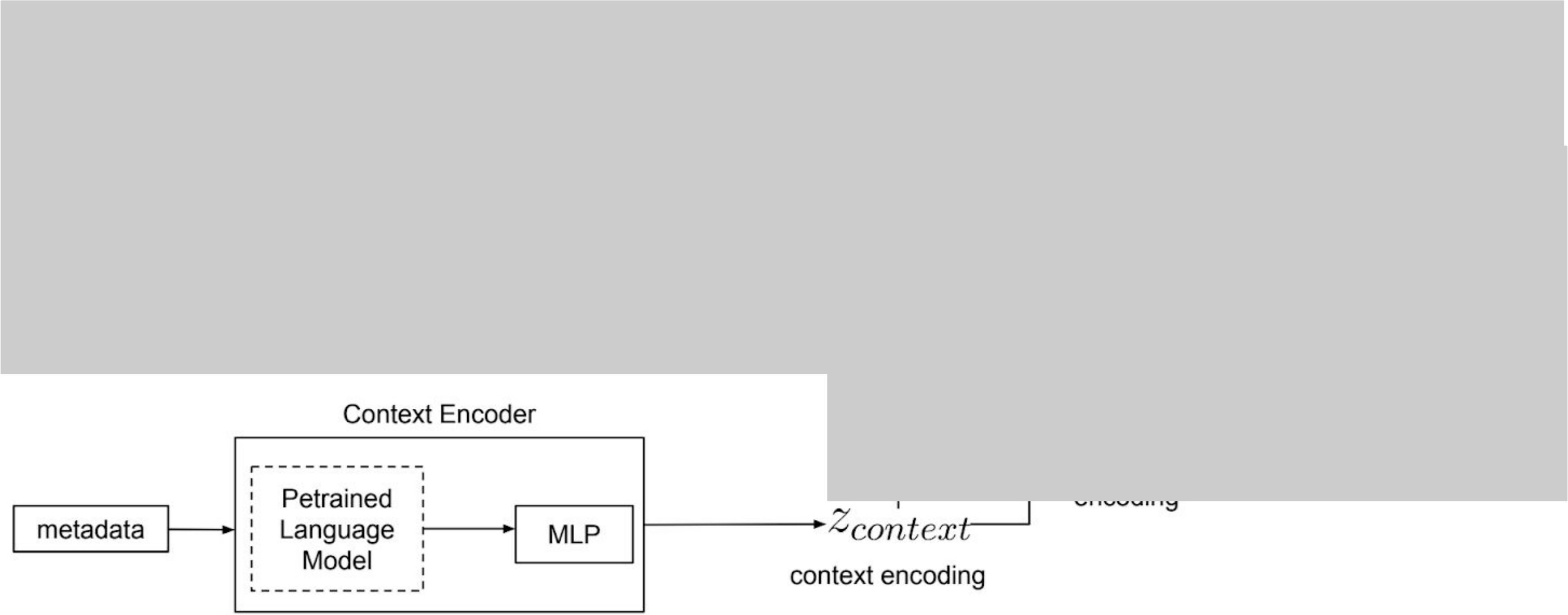


Image taken from [14]: [2102.06177] Multi-Task Reinforcement Learning with Context-based Representations

Multi-Task Reinforcement Learning with Context-based Representations

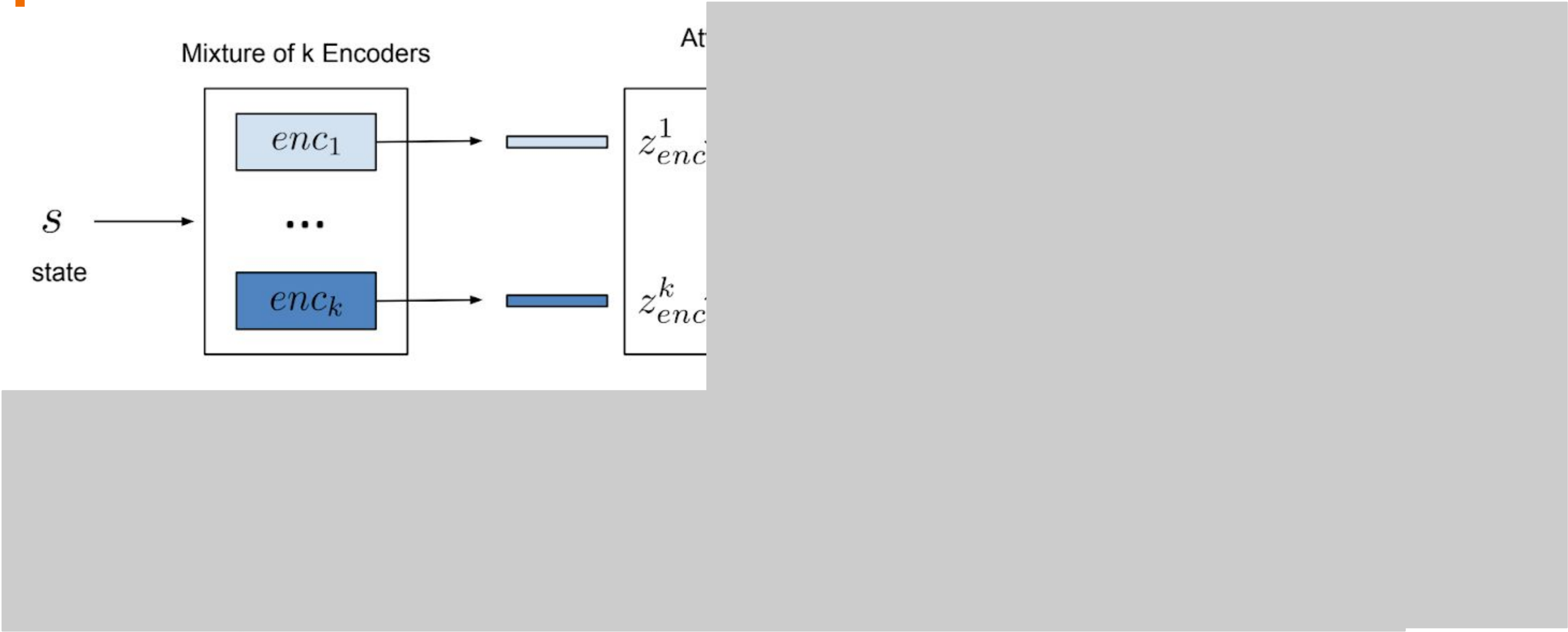


Image taken from [14]: [2102.06177] Multi-Task Reinforcement Learning with Context-based Representations

Multi-Task Reinforcement Learning with Context-based Representations

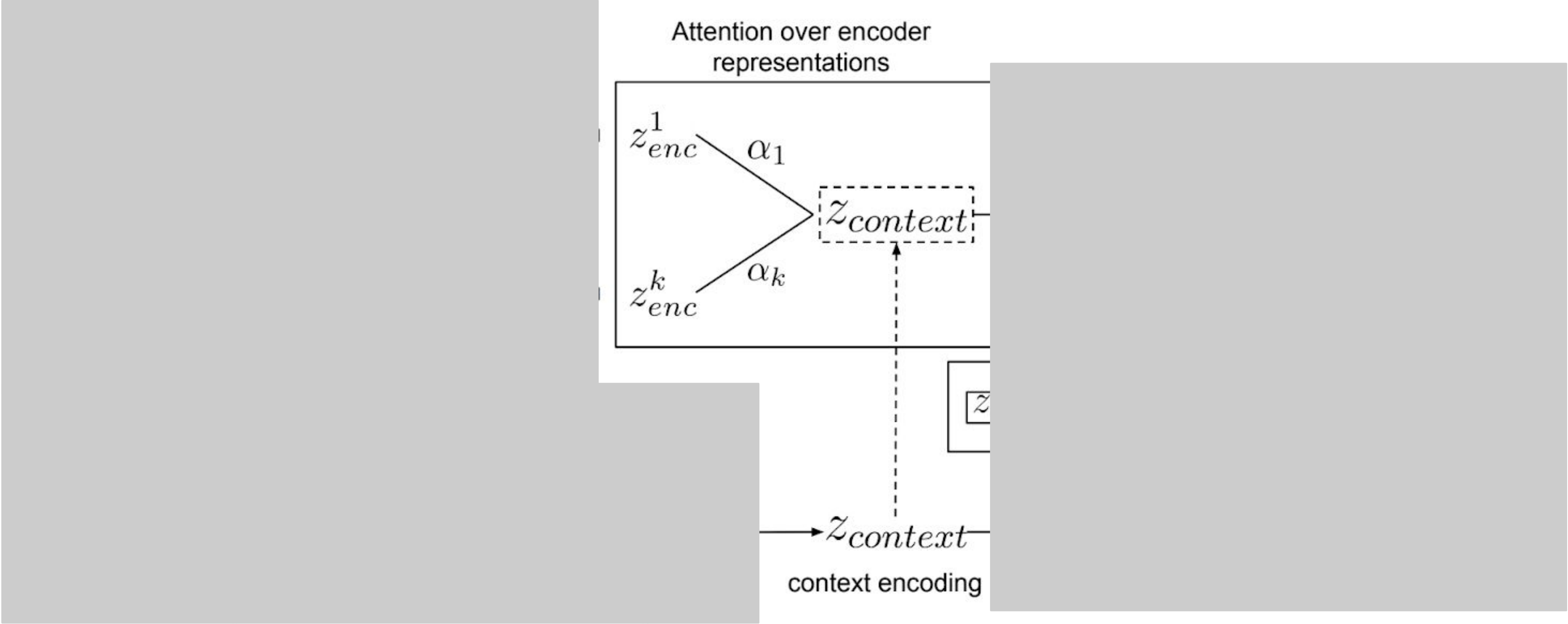


Image taken from [14]: [2102.06177] Multi-Task Reinforcement Learning with Context-based Representations

Multi-Task Reinforcement Learning with Context-based Representations

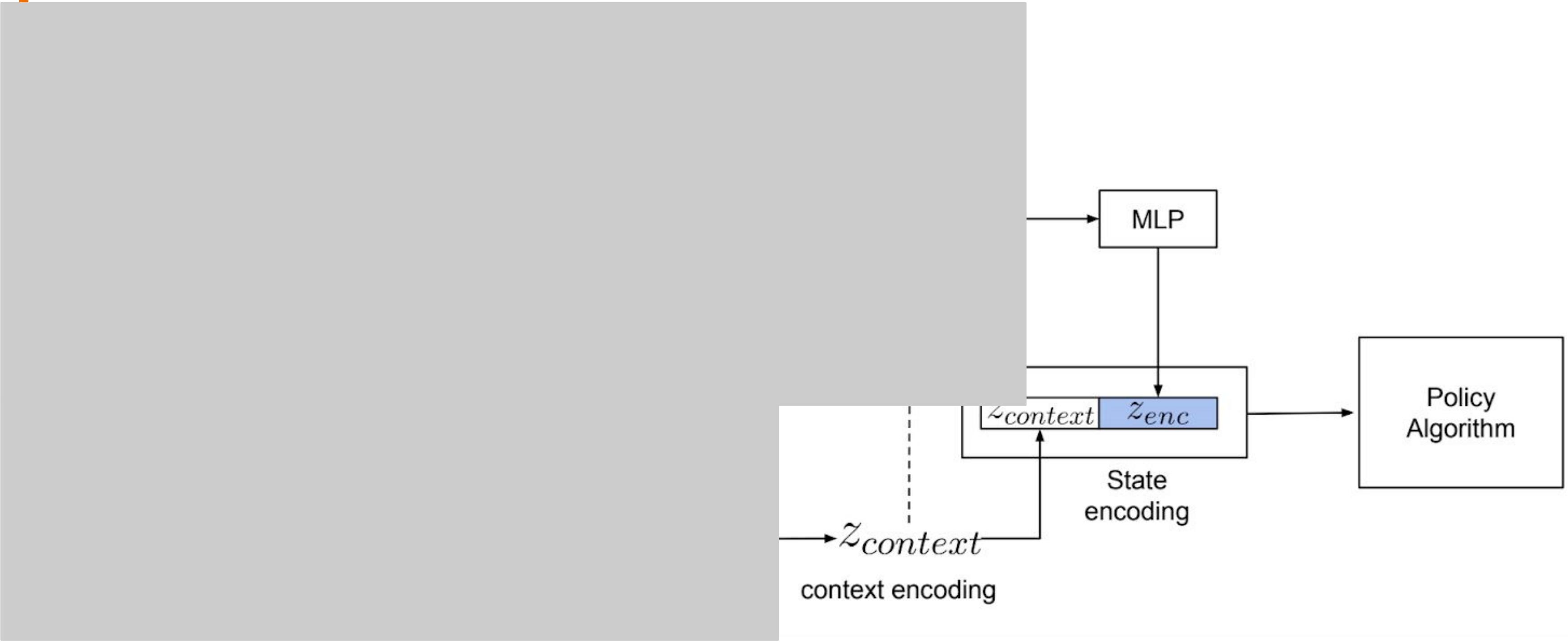


Image taken from [14]: [2102.06177] Multi-Task Reinforcement Learning with Context-based Representations

Multi-Task Reinforcement Learning with Context-based Representations

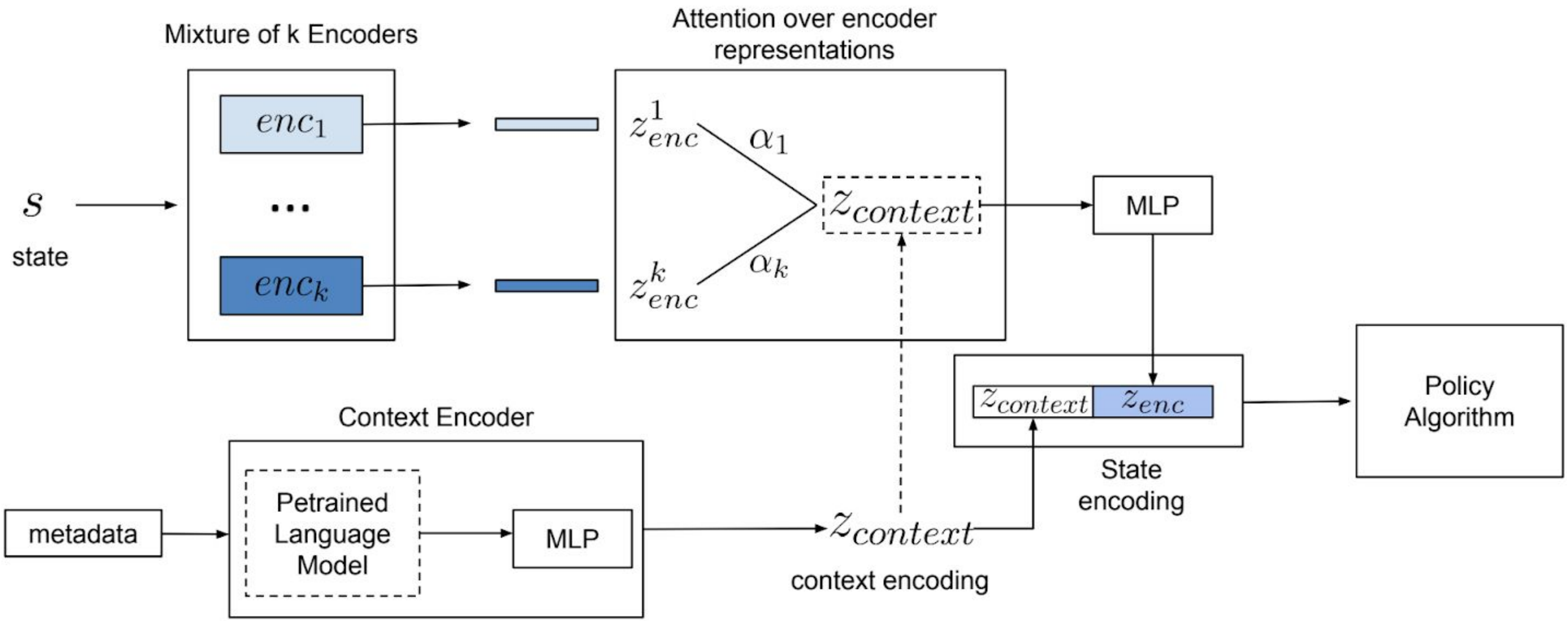


Image taken from [14]: [2102.06177] Multi-Task Reinforcement Learning with Context-based Representations

Recap

- Case I: No relation between tasks.
- Case II: State and action spaces are shared.
  - Sharing knowledge via distillation.
  - Sharing parameters.
  - Can lead to negative interference.
  - Can lead to loss imbalance

Recap

- Case III: State & action spaces are shared and transition dynamics are related.

- Case IV: State & action spaces are shared and common objects across tasks

- Case V: State & action spaces are shared and task metadata is available.

Disclaimer Again

- This is not an exhaustive literature survey on multi task RL.
- We will look at some research papers and setups but there are a lot of other important works.
- The focus will be on providing the motivation/intuition behind the different setups.
- Did not discuss many interesting and related topics: Hierarchical Reinforcement Learning, Curriculum Learning, Meta Learning, etc

Environments for Multi-task Reinforcement Learning

1. [Metaworld: An open source robotics benchmark for meta- and multi-task reinforcement learning]

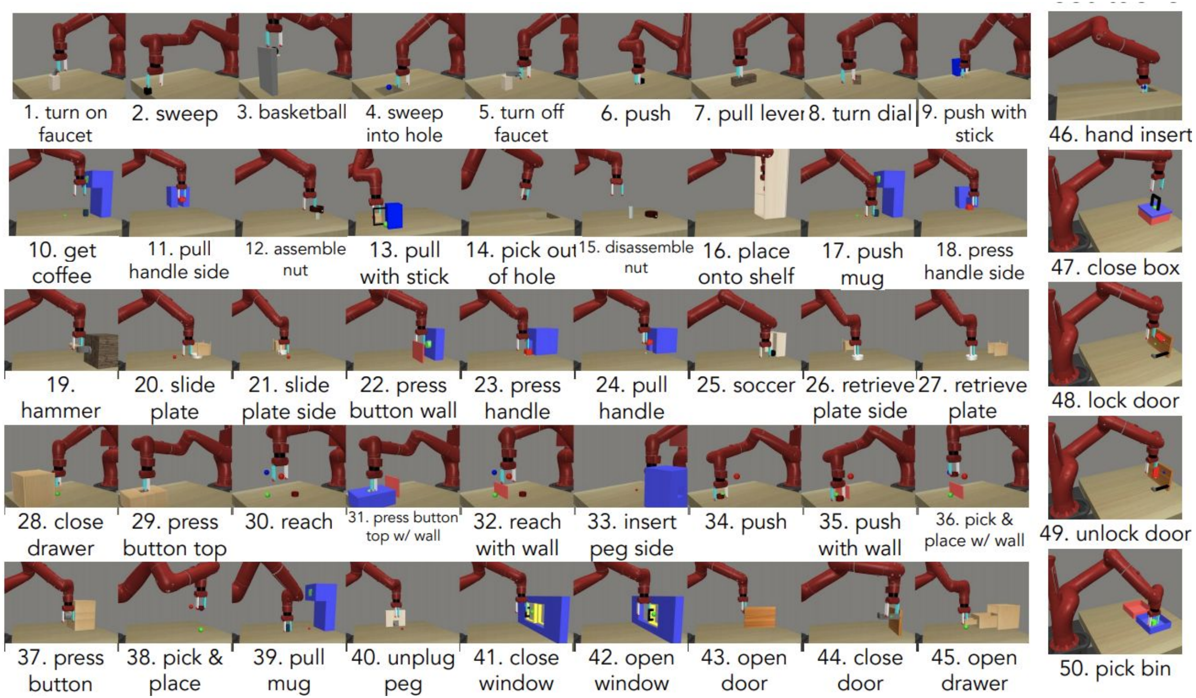


Image taken from [12]: [1910.10897] Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning

Environments for Multi-task Reinforcement Learning

1. Variations of single task RL environments:
    a. GridWorld, Mazes

Four different examples of GridWorld tasks

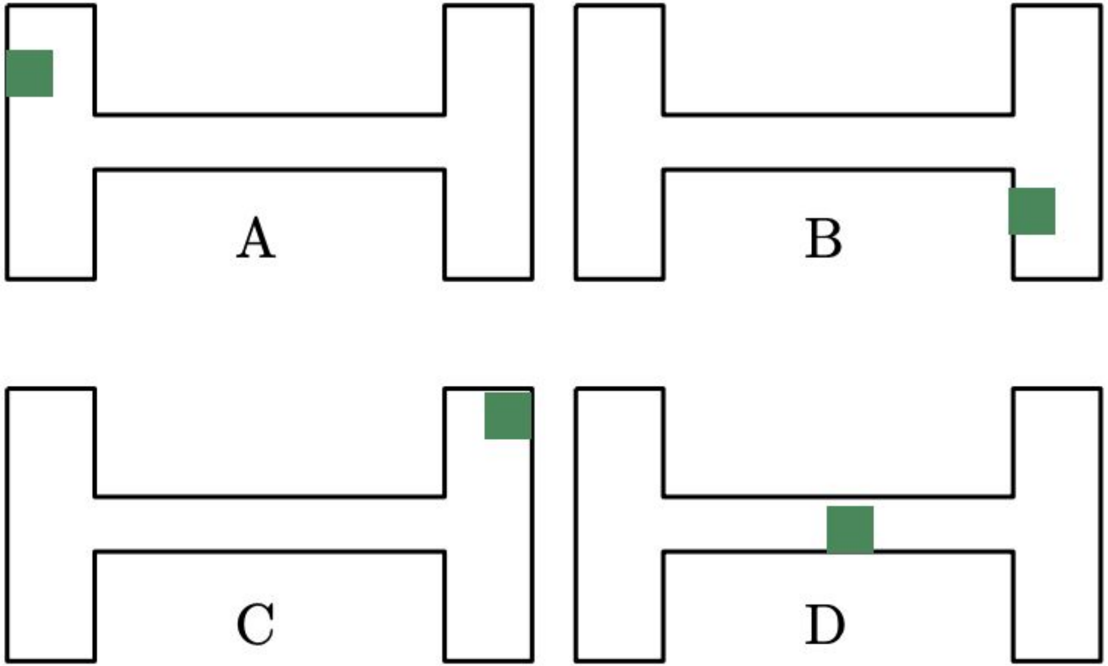


Image taken from [1707.04175] Distral: Robust Multitask Reinforcement Learning

Environments for Multi-task Reinforcement Learning

1. Variations of single task RL environments:
    a. GridWorld, Mazes
    b. Mujoco environments

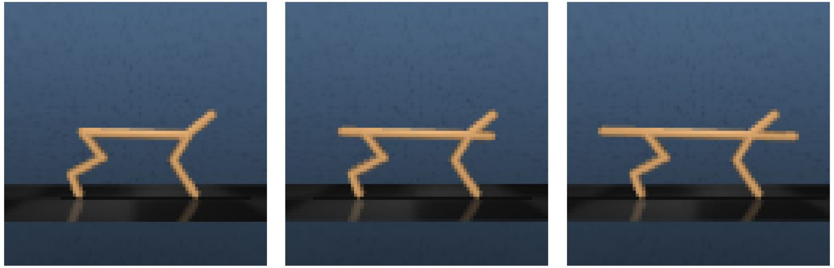


Image taken from [2007.07206] Learning Robust State Abstractions for Hidden-Parameter Block MDPs

Environments for Multi-task Reinforcement Learning

1. MTEnv: MultiTask Environments for Reinforcement Learning

    a. Collection of multi-task environments, including wrapper for some existing multi-task environments, like MetaWorld.

    b. Makes it easy to create multi-task environments from single task environments.

Environments for Multi-task Reinforcement Learning

1. Metaworld: An open source robotics benchmark for meta- and multi-task reinforcement learning

2. Variations of single task RL environments

3. MTEnv: MultiTask Environments for Reinforcement Learning

Startup code for Multi-task Reinforcement Learning

1. Plenty of useful repositories for single task RL
   a. Spinning Up in Deep RL!
   b. PFRL: a PyTorch-based deep reinforcement learning library
   c. RLlib: Scalable Reinforcement Learning

2. MTRL: Multi Task RL Baselines

3. garage: A toolkit for reproducible reinforcement learning research.

Where do I go from here?

Single Task Reinforcement Learning

- ShangtongZhang/reinforcement-learning-an-introduction: Python Implementation of Reinforcement Learning: An Introduction
- Welcome to Spinning Up in Deep RL! — Spinning Up documentation
- pfnet/pfrl: PFRL: a PyTorch-based deep reinforcement learning library
- tensorflow/agents: TF-Agents: A reliable, scalable and easy to use TensorFlow library for Contextual Bandits and Reinforcement Learning.
- RLlib: Scalable Reinforcement Learning
- thu-ml/tianshou: An elegant PyTorch deep reinforcement learning platform.
- rlworkgroup/garage: A toolkit for reproducible reinforcement learning research.

Where do I go from here?

Multi Task Reinforcement Learning

- rlworkgroup/metaworld: An open source robotics benchmark for meta- and multi-task reinforcement learning

- facebookresearch/mtenv: MultiTask Environments for Reinforcement Learning.

- facebookresearch/mtrl: Multi Task RL Baselines

- rlworkgroup/garage: A toolkit for reproducible reinforcement learning research.

Acknowledgement

- Olivier Delalleau
- Sanket Mehta
- Amy Zhang
- Khimya Khetarpal
- Joelle Pineau

Thank you

@shagunsodhani
Facebook AI Research

References

[1]: Alexa Prize Socialbot Grand Challenge 4

[2]: How Customer Service Chatbots Are Redefining Support w/ AI [2019]

[3]: [2004.13637] Recipes for building an open-domain chatbot

[4]: Sutton & Barto Book: Reinforcement Learning: An Introduction

[5]: [1707.04175] Distral: Robust Multitask Reinforcement Learning

[6]: [1810.04650] Multi-Task Learning as Multi-Objective Optimization

[7]: [1711.02257] GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks

[8]: [1812.02224] Adapting Auxiliary Losses Using Gradient Similarity

References

[9]: [2001.06782] Gradient Surgery for Multi-Task Learning

[10]: [1308.3513] Hidden Parameter Markov Decision Processes: A Semiparametric Regression Approach for Discovering Latent Task Parametrizations

[11]: [2007.07206] Learning Robust State Abstractions for Hidden-Parameter Block MDPs

[12]: [1910.10897] Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning

[13]: [2003.13661] Multi-Task Reinforcement Learning with Soft Modularization

[14]: [2102.06177] Multi-Task Reinforcement Learning with Context-based Representations