Multi-Scale Deep Reinforcement Learning for Real-Time 3D-Landmark Detection in CT Scans

Florin C. Ghesu[1,2], Bogdan Georgescu[2], Yefeng Zheng[2], Sasa Grbic[2], Andreas Maier[1], Joachim Hornegger[1], and Dorin Comaniciu[2]

Presented By:
Anmol Sharma[3]

[1]Friedrich Alexander-Universitat Erlangen-Nurnberg

[2]Siemens Healthineers

[3]Medical Image Analysis Laboratory
School of Computing Science
Simon Fraser University
Burnaby, Canada

December 9, 2018

Outline

Outline

Reinforcement Learning

In a Nutshell

- Theory and algorithms uniting multiple fields of machine learning, optimal control theory, psychology and neuroscience.

Reinforcement Learning

In a Nutshell

- Theory and algorithms uniting multiple fields of machine learning, optimal control theory, psychology and neuroscience.
- A class of machine learning algorithms that learn by trial-and-error.

Reinforcement Learning

In a Nutshell

- Theory and algorithms uniting multiple fields of machine learning, optimal control theory, psychology and neuroscience.
- A class of machine learning algorithms that learn by trial-and-error.
- RL is a general-purpose framework for decision-making.

Reinforcement Learning

In a Nutshell

- Theory and algorithms uniting multiple fields of machine learning, optimal control theory, psychology and neuroscience.
- A class of machine learning algorithms that learn by trial-and-error.
- RL is a general-purpose framework for decision-making.
- RL is for an agent with the capacity to act.

Reinforcement Learning

In a Nutshell

- Theory and algorithms uniting multiple fields of machine learning, optimal control theory, psychology and neuroscience.
- A class of machine learning algorithms that learn by trial-and-error.
- RL is a general-purpose framework for decision-making.
- RL is for an agent with the capacity to act.
- Each action influences the agents future state.

Reinforcement Learning
In a Nutshell

- Theory and algorithms uniting multiple fields of machine learning, optimal control theory, psychology and neuroscience.
- A class of machine learning algorithms that learn by trial-and-error.
- RL is a general-purpose framework for decision-making.
- RL is for an agent with the capacity to act.
- Each action influences the agents future state.
- Success is measured by a scalar reward signal.

Reinforcement Learning

In a Nutshell

- Theory and algorithms uniting multiple fields of machine learning, optimal control theory, psychology and neuroscience.
- A class of machine learning algorithms that learn by trial-and-error.
- RL is a general-purpose framework for decision-making.
- RL is for an agent with the capacity to act.
- Each action influences the agents future state.
- Success is measured by a scalar reward signal.
- Goal: select actions to maximize future reward.

Reinforcement Learning

High level overview

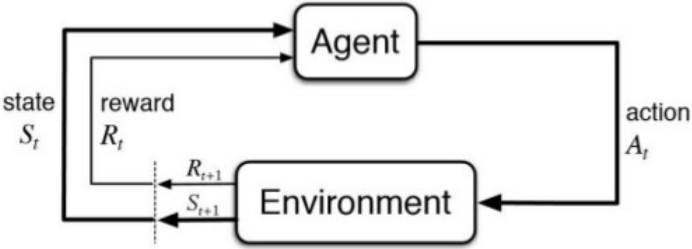

Figure: Agent-Environment interaction

Outline

Reward Function

Mathematical Description

- A reward function is defined inside the environment, which given the **state** and **action**, computes the **reward**.

Reward Function

Mathematical Description

- A reward function is defined inside the environment, which given the **state** and **action**, computes the **reward**.
- Mathematically - $f : (s_t, a_t) \rightarrow \mathbb{R}$, where s_t = Current State, and a_t = Current Action.

Reward Function

Mathematical Description

- A reward function is defined inside the environment, which given the **state** and **action**, computes the **reward**.
- Mathematically - $f : (s_t, a_t) \rightarrow \mathbb{R}$, where s_t = Current State, and a_t = Current Action.
- $r_{t+1} = \mathcal{R}(s_t, a_t)$

Reward Function

Mathematical Description

- A reward function is defined inside the environment, which given the **state** and **action**, computes the **reward**.
- Mathematically - $f : (s_t, a_t) \rightarrow \mathbb{R}$, where s_t = Current State, and a_t = Current Action.
- $r_{t+1} = \mathcal{R}(s_t, a_t)$
- Reward function provides the immediate, intrinsic desirability of environmental states.

Reward Function

Simple Visualization

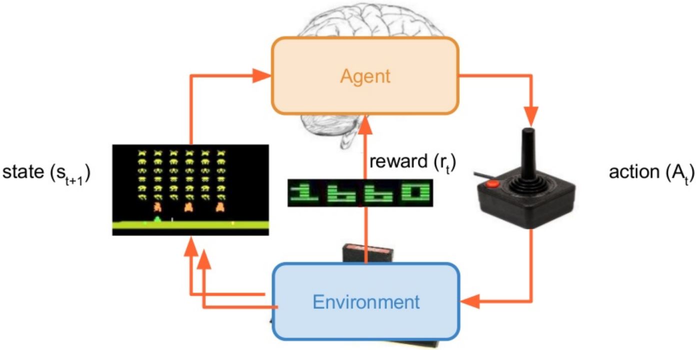

Figure: Atari Example for Reward Function

Value Function

Mathematical Description

- A value function is similar to reward function. but it generates (or predicts) **long-term rewards**.

Value Function

Mathematical Description

- A value function is similar to reward function. but it generates (or predicts) **long-term rewards**.
- It answers the question:

Value Function

Mathematical Description

- A value function is similar to reward function. but it generates (or predicts) **long-term rewards**.
- It answers the question: If I take an action a_t now, how much total reward I get in total in the end?

Value Function

Mathematical Description

- A value function is similar to reward function. but it generates (or predicts) **long-term rewards**.
- It answers the question: If I take an action a_t now, how much total reward I get in total in the end?
- Mathematically - $V : (s_t, a_t) \to \mathbb{R}$, where s_t = Current State, and a_t = Current Action.

Value Function

Mathematical Description

- A value function is similar to reward function. but it generates (or predicts) **long-term rewards**.
- It answers the question: If I take an action a_t now, how much total reward I get in total in the end?
- Mathematically - $V : (s_t, a_t) \rightarrow \mathbb{R}$, where s_t = Current State, and a_t = Current Action.
- $r_{total} = \mathcal{V}(s_t, a_t)$

Value Function

Mathematical Description

- A value function is similar to reward function. but it generates (or predicts) **long-term rewards**.
- It answers the question: If I take an action a_t now, how much total reward I get in total in the end?
- Mathematically - $V : (s_t, a_t) \rightarrow \mathbb{R}$, where s_t = Current State, and a_t = Current Action.
- $r_{total} = \mathcal{V}(s_t, a_t)$
- Value function estimates the worth of an environmental state in the long run.

Value Function

Simple Visualization

Value Function

Simple Visualization

Figure: Mario trying to figure out life (like most grad students)

Value Function

Simple Visualization

Figure: Mario trying to figure out life (like most grad students)

- Mario - "If I take the star, will it help me achieve a better reward by end of this level?"

Value Function

Simple Visualization

Figure: Mario trying to figure out life (like most grad students)

- Mario - "If I take the star, will it help me achieve a better reward by end of this level?"
- Value function $\mathcal{V}(s_t, a_t)$ calculates this total reward quantity, given the current state and action agent takes.

Policy Function

Mathematical Description

- Policy function is a function that takes the decision given a state.

Policy Function

Mathematical Description

- Policy function is a function that takes the decision given a state.
- It answers the question:

Policy Function

Mathematical Description

- Policy function is a function that takes the decision given a state.
- It answers the question: Which action to perform if I'm in the current state s_t?

Policy Function

Mathematical Description

- Policy function is a function that takes the decision given a state.
- It answers the question: Which action to perform if I'm in the current state s_t?
- Mathematically - $\pi : (s_t) \rightarrow \mathbb{A}$, where $\mathbb{A}$ = Action space

Policy Function

Simple Visualization

Policy Function

Simple Visualization

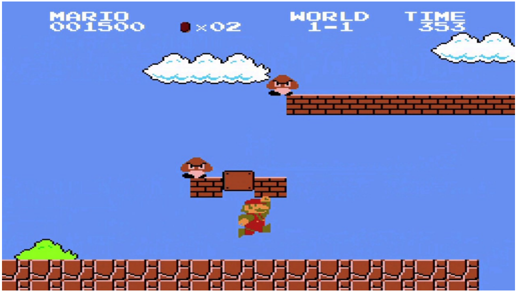

Figure: Mario (still) trying to figure out life

Policy Function

Simple Visualization

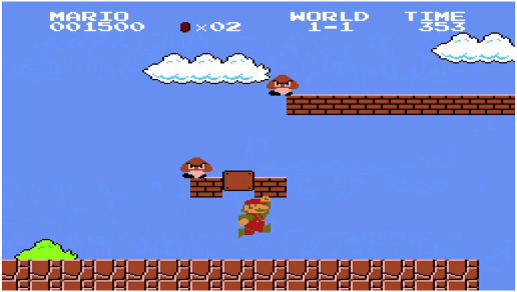

Figure: Mario (still) trying to figure out life

- Mario - "Given that I'm under this brick ceiling, what action should I take?"

Policy Function

Simple Visualization

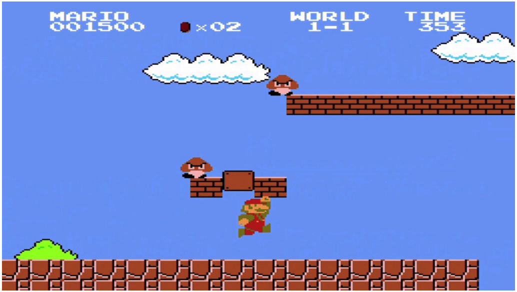

Figure: Mario (still) trying to figure out life

- Mario - "Given that I'm under this brick ceiling, what action should I take?"
- Policy function π computes action a_t, given a current state s_t.

Fundamental Functions

Summary

- Reward Function: $\mathcal{R}(s_t, a_t)$

Fundamental Functions
Summary

- Reward Function: $\mathcal{R}(s_t, a_t)$
  - Computes the reward from environment given the current state, and current action.

Fundamental Functions

Summary

- Reward Function: $\mathcal{R}(s_t, a_t)$
  - Computes the reward from environment given the current state, and current action.
- Value Function: $\mathcal{V}(s_t, a_t)$

Fundamental Functions

Summary

- Reward Function: $\mathcal{R}(s_t, a_t)$
  - Computes the reward from environment given the current state, and current action.
- Value Function: $\mathcal{V}(s_t, a_t)$
  - Computes total expected reward given current state and action taken.

Fundamental Functions

Summary

- Reward Function: $\mathcal{R}(s_t, a_t)$
  - Computes the reward from environment given the current state, and current action.
- Value Function: $\mathcal{V}(s_t, a_t)$
  - Computes total expected reward given current state and action taken.
  - Value function can also be represented as:

Fundamental Functions

Summary

- Reward Function: $\mathcal{R}(s_t, a_t)$
  - Computes the reward from environment given the current state, and current action.
- Value Function: $\mathcal{V}(s_t, a_t)$
  - Computes total expected reward given current state and action taken.
  - Value function can also be represented as:
    - V^{π} = Expected return of reward for a given policy π, starting at any given state s.

Fundamental Functions
Summary

- Reward Function: $\mathcal{R}(s_t, a_t)$
  - Computes the reward from environment given the current state, and current action.
- Value Function: $\mathcal{V}(s_t, a_t)$
  - Computes total expected reward given current state and action taken.
  - Value function can also be represented as:
    - V^{π} = Expected return of reward for a given policy π, starting at any given state s.
    - More on this later.

Fundamental Functions

Summary

- Reward Function: $\mathcal{R}(s_t, a_t)$
  - Computes the reward from environment given the current state, and current action.
- Value Function: $\mathcal{V}(s_t, a_t)$
  - Computes total expected reward given current state and action taken.
  - Value function can also be represented as:
    - V^{π} = Expected return of reward for a given policy π, starting at any given state s.
    - More on this later.
- Policy Function: $\pi(s_t)$

Fundamental Functions

Summary

- Reward Function: $\mathcal{R}(s_t, a_t)$
  - Computes the reward from environment given the current state, and current action.
- Value Function: $\mathcal{V}(s_t, a_t)$
  - Computes total expected reward given current state and action taken.
  - Value function can also be represented as:
    - V^{π} = Expected return of reward for a given policy π, starting at any given state s.
    - More on this later.
- Policy Function: $\pi(s_t)$
  - Predicts the action to take given the current state.

Fundamental Functions

Summary

- Reward Function: $\mathcal{R}(s_t, a_t)$
  - Computes the reward from environment given the current state, and current action.
- Value Function: $\mathcal{V}(s_t, a_t)$
  - Computes total expected reward given current state and action taken.
  - Value function can also be represented as:
    - V^{π} = Expected return of reward for a given policy π, starting at any given state s.
    - More on this later.
- Policy Function: $\pi(s_t)$
  - Predicts the action to take given the current state.

Outline

Markov Decision Process

Mathematical Description

- The agent and the environment is formally represented as a Markov Decision Process (MDP)

Markov Decision Process

Mathematical Description

- The agent and the environment is formally represented as a Markov Decision Process (MDP)
- It is a 5-tuple as follows:

Markov Decision Process

Mathematical Description

- The agent and the environment is formally represented as a Markov Decision Process (MDP)
- It is a 5-tuple as follows:

$$< S, A, T, \boldsymbol{r}, P_0 >$$

Markov Decision Process

Mathematical Description

- The agent and the environment is formally represented as a Markov Decision Process (MDP)
- It is a 5-tuple as follows:

$$< S, A, T, \boldsymbol{r}, P_0 >$$

Where:

Markov Decision Process

Mathematical Description

- The agent and the environment is formally represented as a Markov Decision Process (MDP)
- It is a 5-tuple as follows:

$$< S, A, T, \boldsymbol{r}, P_0 >$$

Where:

- S = All possible agent states

Markov Decision Process

Mathematical Description

- The agent and the environment is formally represented as a Markov Decision Process (MDP)
- It is a 5-tuple as follows:

$$< S, A, T, \boldsymbol{r}, P_0 >$$

Where:

- $S =$ All possible agent states
- $A =$ All possible agent actions

Markov Decision Process

Mathematical Description

- The agent and the environment is formally represented as a Markov Decision Process (MDP)
- It is a 5-tuple as follows:

$$< S, A, T, \boldsymbol{r}, P_0 >$$

Where:

- S = All possible agent states
- A = All possible agent actions
- T = Transition probability from state $s_t \in S$ to $s_{t+1} \in S$ after performing action $a \in A$

Markov Decision Process

Mathematical Description

- The agent and the environment is formally represented as a Markov Decision Process (MDP)
- It is a 5-tuple as follows:

$$< S, A, T, \boldsymbol{r}, P_0 >$$

Where:

- S = All possible agent states
- A = All possible agent actions
- T = Transition probability from state $s_t \in S$ to $s_{t+1} \in S$ after performing action $a \in A$
- $\boldsymbol{r}$ = Reward function that computes a scaler reward when transitioning from state s_t to s_{t+1} using action a_t.

Markov Decision Process

Mathematical Description

- The agent and the environment is formally represented as a Markov Decision Process (MDP)
- It is a 5-tuple as follows:

$$< S, A, T, \boldsymbol{r}, P_0 >$$

Where:

- S = All possible agent states
- A = All possible agent actions
- T = Transition probability from state $s_t \in S$ to $s_{t+1} \in S$ after performing action $a \in A$
- $\boldsymbol{r}$ = Reward function that computes a scaler reward when transitioning from state s_t to s_{t+1} using action a_t.
- P_0 = Probability distribution over initial state.

Markov Decision Process

Mathematical Description

- The agent and the environment is formally represented as a Markov Decision Process (MDP)
- It is a 5-tuple as follows:

$$< S, A, T, \boldsymbol{r}, P_0 >$$

Where:

- S = All possible agent states
- A = All possible agent actions
- T = Transition probability from state $s_t \in S$ to $s_{t+1} \in S$ after performing action $a \in A$
- $\boldsymbol{r}$ = Reward function that computes a scaler reward when transitioning from state s_t to s_{t+1} using action a_t.
- P_0 = Probability distribution over initial state.

Markov Property

Mathematical Description

- Markov Property or Markov Condition is assumed to be applied to the MDP.

Markov Property

Mathematical Description

- Markov Property or Markov Condition is assumed to be applied to the MDP.
- Markov Property is defined to hold if the

Markov Property

Mathematical Description

- Markov Property or Markov Condition is assumed to be applied to the MDP.
- Markov Property is defined to hold if the conditional probability distribution

Markov Property

Mathematical Description

- Markov Property or Markov Condition is assumed to be applied to the MDP.
- Markov Property is defined to hold if the conditional probability distribution of future states of the process

Markov Property

Mathematical Description

- Markov Property or Markov Condition is assumed to be applied to the MDP.
- Markov Property is defined to hold if the conditional probability distribution of future states of the process (conditional on both past and present states)

Markov Property

Mathematical Description

- Markov Property or Markov Condition is assumed to be applied to the MDP.
- Markov Property is defined to hold if the conditional probability distribution of future states of the process (conditional on both past and present states) depends only upon the present state.

Markov Property

Mathematical Description

- Markov Property or Markov Condition is assumed to be applied to the MDP.
- Markov Property is defined to hold if the conditional probability distribution of future states of the process (conditional on both past and present states) depends only upon the present state.
- The MDP $< S, A, T, \boldsymbol{r}, P_0 >$ is assumed to obey the Markov Property, given as:

Markov Property

Mathematical Description

- Markov Property or Markov Condition is assumed to be applied to the MDP.
- Markov Property is defined to hold if the conditional probability distribution of future states of the process (conditional on both past and present states) depends only upon the present state.
- The MDP $< S, A, T, \boldsymbol{r}, P_0 >$ is assumed to obey the Markov Property, given as:

$$P(s_{t+1} = s, r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1} \ldots . s_0, a_0, r_0) =$$

Markov Property

Mathematical Description

- Markov Property or Markov Condition is assumed to be applied to the MDP.
- Markov Property is defined to hold if the conditional probability distribution of future states of the process (conditional on both past and present states) depends only upon the present state.
- The MDP $< S, A, T, \boldsymbol{r}, P_0 >$ is assumed to obey the Markov Property, given as:

$$P(s_{t+1} = s, r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1} \ldots . s_0, a_0, r_0) = P(s_{t+1} = s, r_{t+1} = r | s_t, a_t, r_t)$$

Markov Property

Simple Visualization

Markov Property

Simple Visualization

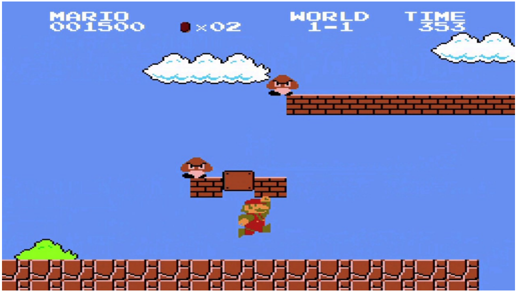

Figure: Super Mario Bros Atari emulator as an environment

Markov Property

Simple Visualization

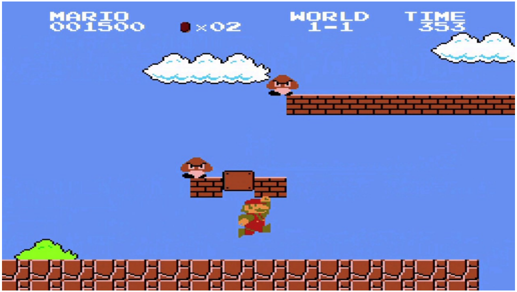

Figure: Super Mario Bros Atari emulator as an environment

- Mario - "My decision to jump and take the mushroom now is independent of whatever I did before this time"

Markov Property

Simple Visualization

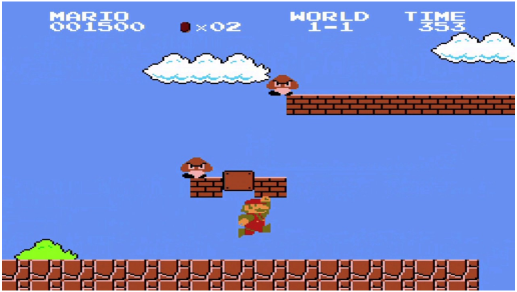

Figure: Super Mario Bros Atari emulator as an environment

- Mario - "My decision to jump and take the mushroom now is independent of whatever I did before this time"
- "It is only dependent on where I am right now".

Markov Property

Simple Visualization

Markov Property

Simple Visualization

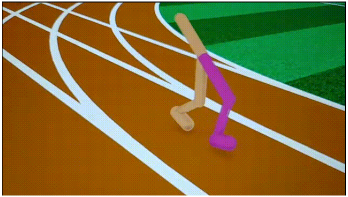

Figure: Physics-based walking environment

Markov Property

Simple Visualization

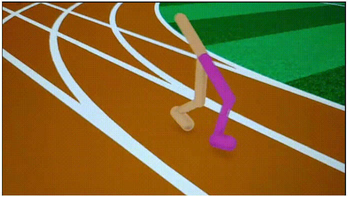

Figure: Physics-based walking environment

- Biped - "My decision to take the next step is independent of my past steps"

Markov Property

Simple Visualization

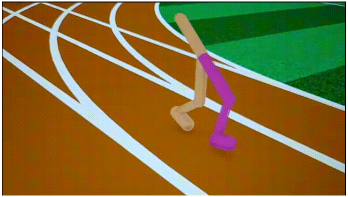

Figure: Physics-based walking environment

- Biped - "My decision to take the next step is independent of my past steps"
- "It is only dependent on where my swing leg is right now".

Outline

How to perform Actions?

Problem Statement

- Given an environment, how do I use it's current state to perform an action?

How to perform Actions?

Problem Statement

- Given an environment, how do I use it's current state to perform an action?
  - Remember, the aim is to maximize the total reward.

How to perform Actions?

Problem Statement

- Given an environment, how do I use it's current state to perform an action?
  - Remember, the aim is to maximize the total reward.
- Use the policy function $\pi(a_{t+1} = a | s_t)$!!
  - Output the probability of taking an action a given a state s_t

How to perform Actions?

Problem Statement

- Given an environment, how do I use it's current state to perform an action?
  - Remember, the aim is to maximize the total reward.
- Use the policy function $\pi(a_{t+1} = a|s_t)$!!
  - Output the probability of taking an action a given a state s_t
  - Remember there are multiple actions that you can take.

How to perform Actions?

Problem Statement

- Given an environment, how do I use it's current state to perform an action?
  - Remember, the aim is to maximize the total reward.
- Use the policy function $\pi(a_{t+1} = a | s_t)$!!
  - Output the probability of taking an action a given a state s_t
  - Remember there are multiple actions that you can take.
  - For mario, its *forward, backwards, jump, run*

How to perform Actions?

Problem Statement

- Given an environment, how do I use it's current state to perform an action?
  - Remember, the aim is to maximize the total reward.
- Use the policy function $\pi(a_{t+1} = a|s_t)$!!
  - Output the probability of taking an action a given a state s_t
  - Remember there are multiple actions that you can take.
  - For mario, its *forward, backwards, jump, run*
- So what's a simple and intuitive way to do it?

How to perform Actions?

Simple Algorithm

1. Sample a state s_0 from P_0 at $t = 0$ # sample the initial position

How to perform Actions?

Simple Algorithm

1. Sample a state s_0 from P_0 at $t = 0$ # sample the initial position
2. $a_0 = \pi(.|s_0)$ # given the current state, find the action with highest probability

How to perform Actions?

Simple Algorithm

1. Sample a state s_0 from P_0 at $t = 0$ # sample the initial position
2. $a_0 = \pi(.|s_0)$ # given the current state, find the action with highest probability
3. Apply a_0 to the environment and obtain s_1 by observing the environment.

How to perform Actions?

Simple Algorithm

1. Sample a state s_0 from P_0 at $t = 0$ # sample the initial position
2. $a_0 = \pi(.|s_0)$ # given the current state, find the action with highest probability
3. Apply a_0 to the environment and obtain s_1 by observing the environment.
4. Calculate the reward we received using $r_0 = \boldsymbol{r}(s_0, a_0, s_1)$

How to perform Actions?

Simple Algorithm

1. Sample a state s_0 from P_0 at $t = 0$ # sample the initial position
2. $a_0 = \pi(.|s_0)$ # given the current state, find the action with highest probability
3. Apply a_0 to the environment and obtain s_1 by observing the environment.
4. Calculate the reward we received using $r_0 = \boldsymbol{r}(s_0, a_0, s_1)$
5. Append to $\tau = (s_0, a_0, r_0)$.

How to perform Actions?

Simple Algorithm

1. Sample a state s_0 from P_0 at $t = 0$ # sample the initial position
2. $a_0 = \pi(.|s_0)$ # given the current state, find the action with highest probability
3. Apply a_0 to the environment and obtain s_1 by observing the environment.
4. Calculate the reward we received using $r_0 = \boldsymbol{r}(s_0, a_0, s_1)$
5. Append to $\tau = (s_0, a_0, r_0)$.
6. Repeat Steps I to V, H times. # H = Horizon

How to perform Actions?

Simple Algorithm

1. Sample a state s_0 from P_0 at $t = 0$ # sample the initial position
2. $a_0 = \pi(.|s_0)$ # given the current state, find the action with highest probability
3. Apply a_0 to the environment and obtain s_1 by observing the environment.
4. Calculate the reward we received using $r_0 = \boldsymbol{r}(s_0, a_0, s_1)$
5. Append to $\tau = (s_0, a_0, r_0)$.
6. Repeat Steps I to V, H times. # H = Horizon
7. Return $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, ...s_{H-1}, a_{H-1}, r_{H-1})$

Ok I performed actions, now what?

Horizon

- The parameter H is called the “horizon”.

Ok I performed actions, now what?

Horizon

- The parameter H is called the "horizon".
  - It a measure of how long should the "episode" last for the current policy to be making the decision, without stopping and learning.

Ok I performed actions, now what?

Horizon

- The parameter H is called the "horizon".
  - It a measure of how long should the "episode" last for the current policy to be making the decision, without stopping and learning.
- If $H =$ finite, the task is called episodic, elif $H =$ infinite, the task is continuous.

Ok I performed actions, now what?

Horizon

- The parameter H is called the "horizon".
  - It a measure of how long should the "episode" last for the current policy to be making the decision, without stopping and learning.
- If $H =$ finite, the task is called episodic, elif $H =$ infinite, the task is continuous.
  - Atari games are episodic. The Agent usually plays until 1) It dies, or 2) The end of round.

Ok I performed actions, now what?

Horizon

- The parameter H is called the "horizon".
  - It a measure of how long should the "episode" last for the current policy to be making the decision, without stopping and learning.
- If $H =$ finite, the task is called episodic, elif $H =$ infinite, the task is continuous.
  - Atari games are episodic. The Agent usually plays until 1) It dies, or 2) The end of round.
  - After one of the above event happens, the learning process begins.

Ok I performed actions, now what?

Horizon

- The parameter H is called the “horizon”.
  - It a measure of how long should the “episode” last for the current policy to be making the decision, without stopping and learning.
- If $H =$ finite, the task is called episodic, elif $H =$ infinite, the task is continuous.
  - Atari games are episodic. The Agent usually plays until 1) It dies, or 2) The end of round.
  - After one of the above event happens, the learning process begins.

Ok I performed actions, now what?

Trajectory and Total Reward

- Recall from the algorithm slide, that we returned a vector $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots . s_{H-1}, a_{H-1}, r_{H-1})$.

Ok I performed actions, now what?

Trajectory and Total Reward

- Recall from the algorithm slide, that we returned a vector $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots .s_{H-1}, a_{H-1}, r_{H-1})$.
- τ is called the "trajectory" of the agent.

Ok I performed actions, now what?

Trajectory and Total Reward

- Recall from the algorithm slide, that we returned a vector $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, ...s_{H-1}, a_{H-1}, r_{H-1})$.
- τ is called the "trajectory" of the agent.
  - Trajectory is basically the path that the agent took in the current episode.

Ok I performed actions, now what?

Trajectory and Total Reward

- Recall from the algorithm slide, that we returned a vector $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, ...s_{H-1}, a_{H-1}, r_{H-1})$.
- τ is called the “trajectory” of the agent.
  - Trajectory is basically the path that the agent took in the current episode.
- Now, given a trajectory τ obtained after the agent performed actions for an episode of H time steps:

Ok I performed actions, now what?

Trajectory and Total Reward

- Recall from the algorithm slide, that we returned a vector $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, ...s_{H-1}, a_{H-1}, r_{H-1})$.
- τ is called the "trajectory" of the agent.
  - Trajectory is basically the path that the agent took in the current episode.
- Now, given a trajectory τ obtained after the agent performed actions for an episode of H time steps:
  - We need to find the total reward that we received after this whole episode.

Ok I performed actions, now what?

Trajectory and Total Reward

- Recall from the algorithm slide, that we returned a vector $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, ...s_{H-1}, a_{H-1}, r_{H-1})$.
- τ is called the "trajectory" of the agent.
  - Trajectory is basically the path that the agent took in the current episode.
- Now, given a trajectory τ obtained after the agent performed actions for an episode of H time steps:
  - We need to find the total reward that we received after this whole episode.
  - It is given as:

Ok I performed actions, now what?

Trajectory and Total Reward

- Recall from the algorithm slide, that we returned a vector $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, ...s_{H-1}, a_{H-1}, r_{H-1})$.
- τ is called the "trajectory" of the agent.
  - Trajectory is basically the path that the agent took in the current episode.
- Now, given a trajectory τ obtained after the agent performed actions for an episode of H time steps:
  - We need to find the total reward that we received after this whole episode.
  - It is given as:

$$R(\tau) = \sum_{i=0}^{H-1} \gamma^i * r_i$$

Ok I performed actions, now what?

Trajectory and Total Reward

- Recall from the algorithm slide, that we returned a vector $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, ...s_{H-1}, a_{H-1}, r_{H-1})$.
- τ is called the "trajectory" of the agent.
  - Trajectory is basically the path that the agent took in the current episode.
- Now, given a trajectory τ obtained after the agent performed actions for an episode of H time steps:
  - We need to find the total reward that we received after this whole episode.
  - It is given as:
    $$R(\tau) = \sum_{i=0}^{H-1} \gamma^i * r_i$$
  - Where γ is the "discount factor".

Ok I performed actions, now what?

Trajectory and Total Reward

- Recall from the algorithm slide, that we returned a vector $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, ...s_{H-1}, a_{H-1}, r_{H-1})$.
- τ is called the "trajectory" of the agent.
  - Trajectory is basically the path that the agent took in the current episode.
- Now, given a trajectory τ obtained after the agent performed actions for an episode of H time steps:
  - We need to find the total reward that we received after this whole episode.
  - It is given as:

$$R(\tau) = \sum_{i=0}^{H-1} \gamma^i * r_i$$

  - Where γ is the "discount factor".
  - What does γ do?

Ok I performed actions, now what?

Trajectory and Total Reward

- Recall from the algorithm slide, that we returned a vector $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, ...s_{H-1}, a_{H-1}, r_{H-1})$.
- τ is called the "trajectory" of the agent.
  - Trajectory is basically the path that the agent took in the current episode.
- Now, given a trajectory τ obtained after the agent performed actions for an episode of H time steps:
  - We need to find the total reward that we received after this whole episode.
  - It is given as:
$$R(\tau) = \sum_{i=0}^{H-1} \gamma^i * r_i$$
  - Where γ is the "discount factor".
  - What does γ do?
    - Larger the value of γ, greater the influence of later rewards on the return.

Ok I performed actions, now what?

Trajectory and Total Reward

- Recall from the algorithm slide, that we returned a vector $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, ...s_{H-1}, a_{H-1}, r_{H-1})$.
- τ is called the "trajectory" of the agent.
  - Trajectory is basically the path that the agent took in the current episode.
- Now, given a trajectory τ obtained after the agent performed actions for an episode of H time steps:
  - We need to find the total reward that we received after this whole episode.
  - It is given as:
    $$R(\tau) = \sum_{i=0}^{H-1} \gamma^i * r_i$$
  - Where γ is the "discount factor".
  - What does γ do?
    - Larger the value of γ, greater the influence of later rewards on the return.
    - In other words, it larger γ favours long term rewards, and the agent becomes willing to forgo short term rewards.

Outline

How to learn the policy function?

Objective Function and Aim

- Objective of the agent is to maximize the total reward $R(\tau)$ that it receives, by tweaking the trajectory τ.

How to learn the policy function?

Objective Function and Aim

- Objective of the agent is to maximize the total reward $R(\tau)$ that it receives, by tweaking the trajectory τ.
- Recall that $R(\tau) = \sum_{i=0}^{H-1} \gamma^i * r_i$

How to learn the policy function?

Objective Function and Aim

- Objective of the agent is to maximize the total reward $R(\tau)$ that it receives, by tweaking the trajectory τ.
- Recall that $R(\tau) = \sum_{i=0}^{H-1} \gamma^i * r_i$
- Expanding r, we get: $R(\tau) = \sum_{i=0}^{H-1} \gamma^i * r(s_t, a_t, s_{t+1})$

How to learn the policy function?

Objective Function and Aim

- Objective of the agent is to maximize the total reward $R(\tau)$ that it receives, by tweaking the trajectory τ.
- Recall that $R(\tau) = \sum_{i=0}^{H-1} \gamma^i * r_i$
- Expanding r, we get: $R(\tau) = \sum_{i=0}^{H-1} \gamma^i * r(s_t, a_t, s_{t+1})$
- Aim: Find a policy π which maximizes a cost function $J(\pi)$

How to learn the policy function?

Objective Function and Aim

- Objective of the agent is to maximize the total reward $R(\tau)$ that it receives, by tweaking the trajectory τ.
- Recall that $R(\tau) = \sum_{i=0}^{H-1} \gamma^i * r_i$
- Expanding r, we get: $R(\tau) = \sum_{i=0}^{H-1} \gamma^i * r(s_t, a_t, s_{t+1})$
- Aim: Find a policy π which maximizes a cost function $J(\pi)$

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{i=0}^{H-1} \gamma^i * r(s_t, a_t, s_{t+1}) \middle| s_0 \right]$$

- In general: $\pi^* = argmax_\pi J(\pi)$

How to learn the policy function?

Representing Policy

- Policy function can be represented as any non-linear function approximator.

How to learn the policy function?

Representing Policy

- Policy function can be represented as any non-linear function approximator.
- Fancy name for an obvious choice: Neural Network.

How to learn the policy function?

Representing Policy

- Policy function can be represented as any non-linear function approximator.
- Fancy name for an obvious choice: Neural Network.
- A Neural Network as a policy function learns its parameters θ so that it can generate trajectories that maximize the reward.

How to learn the policy function?

Representing Policy

- Policy function can be represented as any non-linear function approximator.
- Fancy name for an obvious choice: Neural Network.
- A Neural Network as a policy function learns its parameters θ so that it can generate trajectories that maximize the reward.
- Hence in a setting where a policy is represented by a neural network, the problem becomes:

How to learn the policy function?

Representing Policy

- Policy function can be represented as any non-linear function approximator.
- Fancy name for an obvious choice: Neural Network.
- A Neural Network as a policy function learns its parameters θ so that it can generate trajectories that maximize the reward.
- Hence in a setting where a policy is represented by a neural network, the problem becomes:

$$\theta^* = argmax_\theta J(\theta)$$

How to learn the policy function?

Representing Policy

- Policy function can be represented as any non-linear function approximator.
- Fancy name for an obvious choice: Neural Network.
- A Neural Network as a policy function learns its parameters θ so that it can generate trajectories that maximize the reward.
- Hence in a setting where a policy is represented by a neural network, the problem becomes:

$$\theta^* = argmax_{\theta} J(\theta)$$

- This can be maximized using Gradient Ascent

How to learn the policy function?

Representing Policy

- Policy function can be represented as any non-linear function approximator.
- Fancy name for an obvious choice: Neural Network.
- A Neural Network as a policy function learns its parameters θ so that it can generate trajectories that maximize the reward.
- Hence in a setting where a policy is represented by a neural network, the problem becomes:

$$\theta^* = argmax_\theta J(\theta)$$

- This can be maximized using Gradient Ascent

$$\theta \leftarrow \theta + \alpha_t \nabla J(\theta)$$

Outline

Summary

- That's it!

Summary

- That's it!
- This was the fundamental formal definition of the reinforcement learning problem.

Summary

- That's it!
- This was the fundamental formal definition of the reinforcement learning problem.
- The concepts introduced here are general terminologies, and should be a good starting point to explore state-of-art literature in (Deep) RL.