AI 3000 (CS 5500) : Reinforcement Learning

Easwar Subramanian

TCS Innovation Labs, Hyderabad

Email : cs5500.2020@iith.ac.in

August 03, 2024

Overview

TCS Research & Innovation

Introduction

Machine Learning

" Machine learning is about developing bots that has the ability to automatically learn and improve from experience without being explicitly programmed "

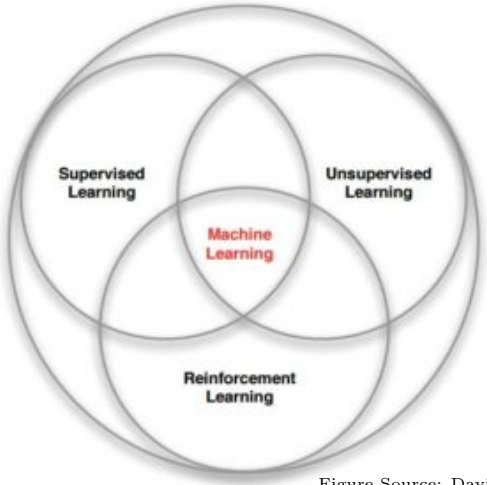
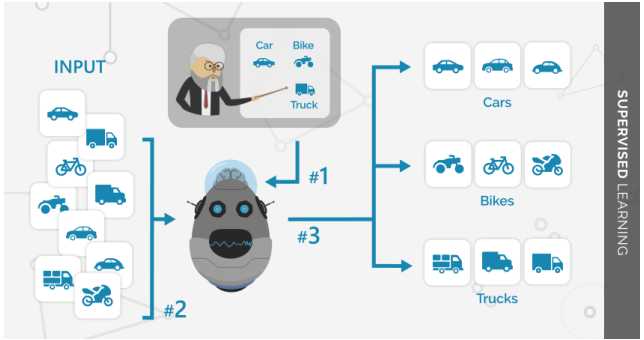


Figure Source: David Silver's RL course

Supervised Learning

- **Data** : $(x, y) \rightarrow x$ is data and y is label
- **Goal**: Learn a function f to map $y = f(x)$
- **Problems** : Classification or Regression

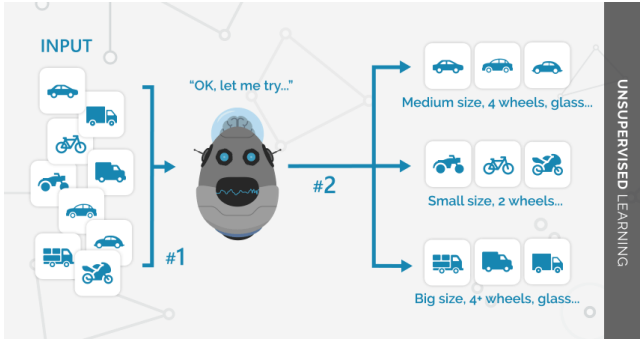


Classification

Figure Source: Aura Portal - AI/ML Blog

Unsupervised Learning

- **Data** : $(x) \rightarrow$ Only data; No label
- **Goal**: Learn underlying structure
- **Techniques** : Clustering

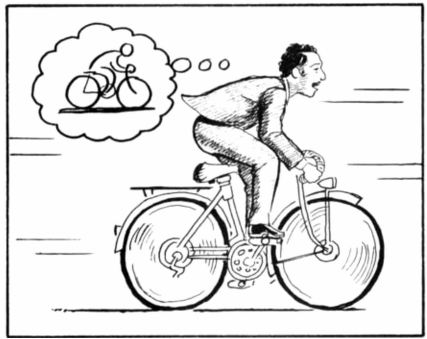


Clustering

Figure Source: Aura Portal - AI/ML Blog

Reinforcement Learning

- **Data** : Agent interacts with environment to collect data
- **Goal** : Agent learns to interact with environment to maximize an utility
- **Examples** : Learn a task, Navigation

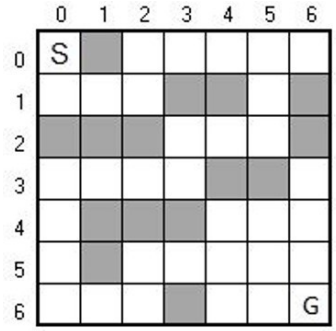


Learn to cycle (task)

Figure Source: worldmodels.github.io

Example : Navigation

- **Task :** Start from square S and reach square G in as less moves as possible



Navigation in grid world

- One has to make **sequence** of moves (actions)
- Action chosen **determine** which squares (states) would be visited subsequently
- Reaching the **goal state** will fetch a reward; Visiting intermediate squares (states) may or may not fetch reward

Figure Source: Genevieve Hayes : Medium Post

Sequential Decision Making

TCS Research & Innovation

Supervised or Unsupervised Setting

- System is making a isolated decision; i.e., classification, regression or clustering;
- Decision does not affect future observations

Reinforcement Learning

- Generally, the agent makes a sequence of decisions (or actions)
- Actions affect future observations
- Actions taken have consequences

Types of Learning : Summary

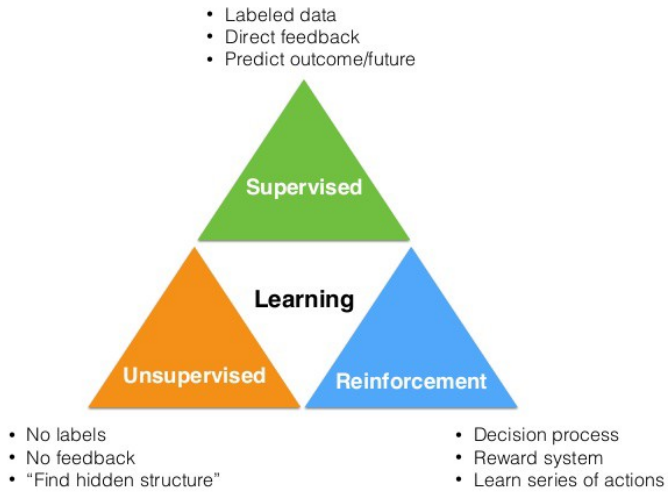
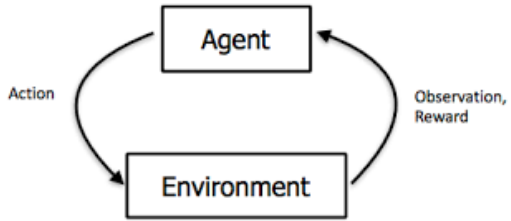


Figure Source: Saggie

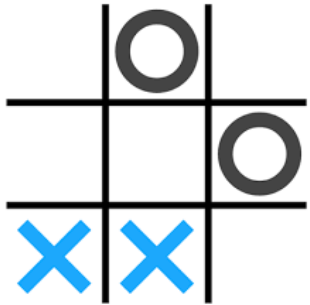
RL : Framework, Components and Challenges

Reinforcement Learning : Framework



- Observations are non i.i.d and are sequential in nature
- Agent's action (may) affect the subsequent observations seen
- There is no supervisor; Only reward signal (feedback)
- Reward or feedback can be delayed

Example : Tic-Tac-Toe



- Observations : Board position
- Actions : Moves
- Reward : Win or Loss

Example : Robotics

- Observations : Image from in-built camera
- Actions : Motor current for movement
- Reward : Task success measure

Example : Inventory Control

- Observations : Stock levels
- Actions : What to purchase
- Reward : Profit

Components of RL : Agent and Environment

TCS Research & Innovation

Agent

- Executes action upon receiving observation
- For taking an action the agent receives an appropriate reward

Environment

- An **external system** that an agent can perceive and act on.
- Receives action from agent and in response emits appropriate reward and (next) observation

Components of RL : State and Reward

TCS Research & Innovation

State

- State can be viewed as a summary or an abstraction of the past history of the system
  - ★ For example, in Tic-Tac-Toe, the state could be raw image or vector representation of the board

Reward

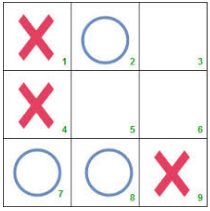
- Reward is a scalar feedback signal
- Indicates how well agent acted at a certain time
- The agent's aim is to maximise cumulative reward

Reinforcement Learning : Challenges

- Delayed Feedback
- Credit Assignment Problem
- Stochastic Environment
- Definition of Reward Function
- Data Collection Problem

Historical Notes

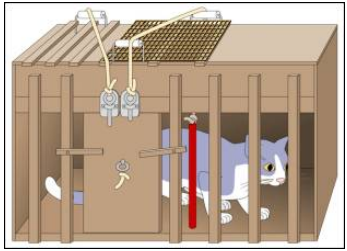
Learning by Trial and Error



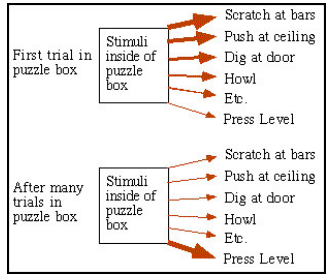
Tic-Tac-Toe

- Random movements by agent is akin to exploration
- Exploration can help the agent place 'X' in square number 5
- Reward obtained from placing 'X' in square number 5 can now be remembered in terms of updating the policy or value function

Thondrike's Cat : Psychophysical Experiment



Thondrike's cat



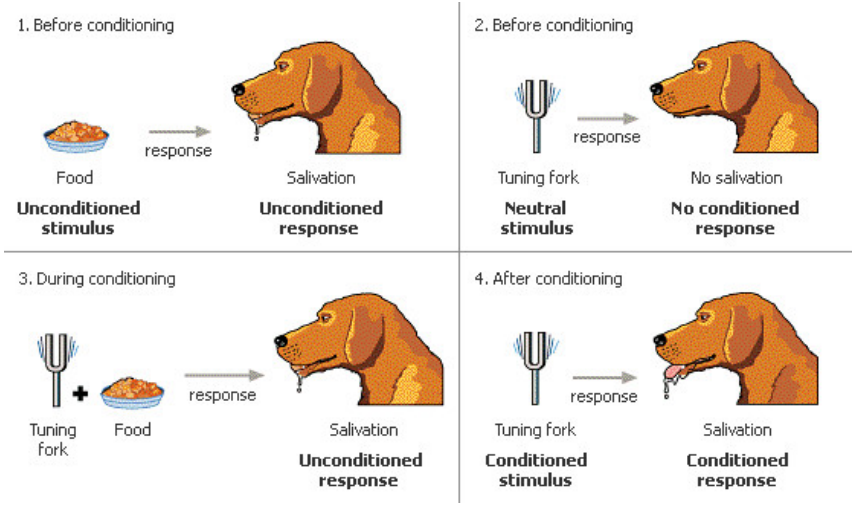
Law of Effect

Law of Effect (1898)

Any behaviour that is followed by pleasant consequences is likely to be repeated, and any behaviour followed by unpleasant consequences is likely to be stopped

Figure Source: Oscar Education : Blogpost

Pavlov's Dog



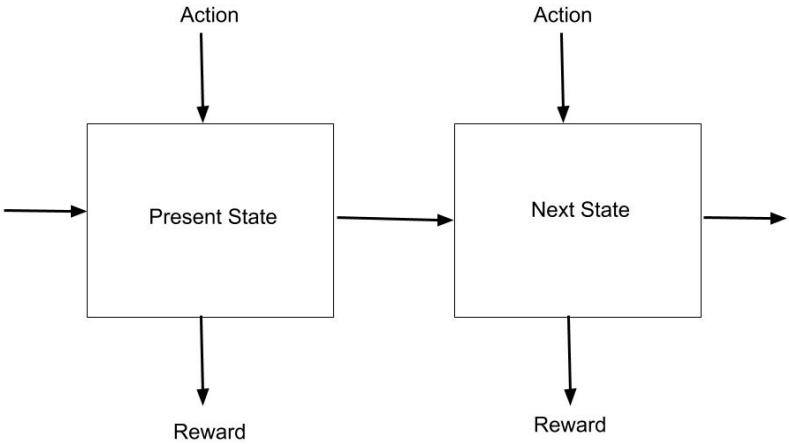
Pavlov's Dog

Figure Source: https://www.age-of-the-sage.org/psychology/pavlov.html

Connections to Temporal Difference

- Ivav Pavlov laid the ground for classical conditioning (1901)
- First theory that incorporated time into the learning procedure
- **Rescorla-Wagner** (RW) (1972) model is a formal model to explain Pavlovian conditioning
- Temporal-Difference (TD) learning, that extends RW model, is an approach to learning how to predict a quantity that depends on future values of a given signal (Sutton, 1984)
- TD learning forms the basis of almost all RL algorithms that we see today

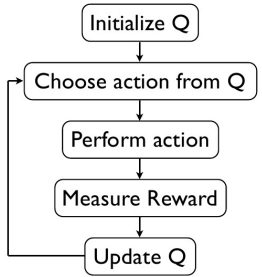
Connections to Optimal Control



Connections to Optimal Control

- Outcomes are partly random and partly under the control of the decision maker
- Markov Decision Process (MDP) (Bellman, 1957) is used as a framework to model and solve sequential decision problem
- People working in control theory have contributed to optimal sequential decision making

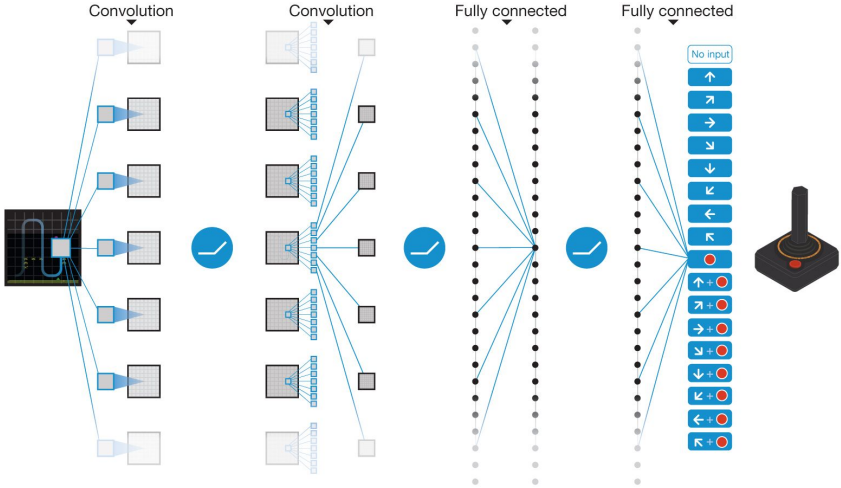
Modern Reinforcement Learning



- The temporal difference (TD) thread and the optimal control thread were bought together by Watkins (1989) when he proposed the famous **Q-learning algorithm**
- Gerald Tesauro (1992) employed TD learning to play **backgammon**; The developed software agent was able to beat experts

Figure Source: https://www.linuxjournal.com/article/11038

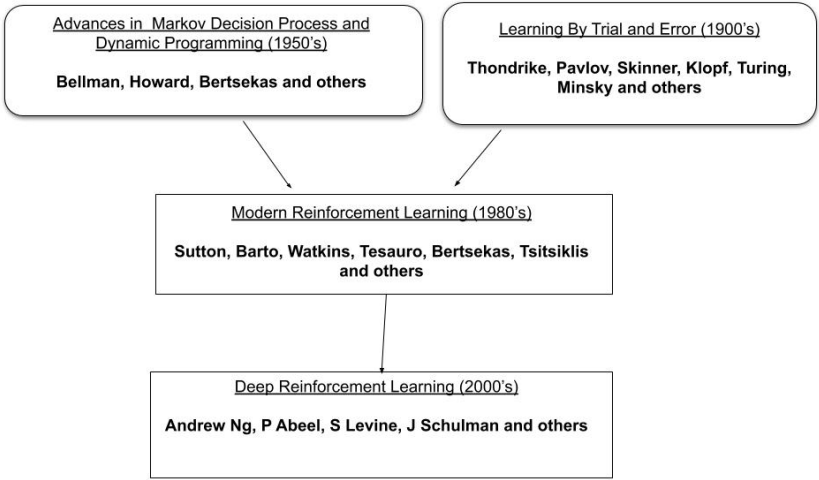
Era of Deep (Reinforcement) Learning



Deep Neural Net for Atari Games

 Figure Source: Minh et.al, 2015

Reinforcement Learning : History



Motivation and Success Stories

Motivation

TCS Research & Innovation

Why study Reinforcement Learning (RL) now ?

- Advances in computational capability
- Advances in deep learning
- Advances in reinforcement learning
  - ★ Subject matter of this course !

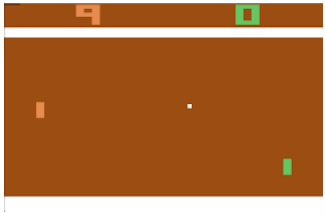
Slide Credit: Sergey Levine course on Deep RL at UCB

Sucess Stories

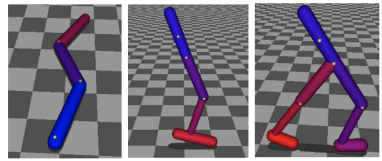
(a) Ng et al 2004

(b) Kohl et al 2004

Sucess Stories



(c) Minh et al 2013



(d) Schulman et al 2016

(d) Silver et al. 2016

Towards Intelligent Systems

- Things that we can all do (Walking) (Evolution, may be)
- Things that we learn (driving a bicycle, car etc)
- We learn a huge variety of things (music, sport, arts etc)

We are still far from building a 'reasonable' intelligent system

- We are taking baby steps towards the goal of building intelligent systems
- **Reinforcement Learning (RL) is one of the important paradigm towards that goal**

Slide Credit: Sergey Levine course on Deep RL at UCB

Course Logistics

TCS Research & Innovation

Modern Reinforcement Learning

- Markov Decision Process
- Dynamic Programming and Bellman Optimality Principle
- Value and Policy Iteration
- Convergence Properties of Value and Policy Iteration
- Model Free Prediction
- Model Free Control : Q-Learning and SARSA

TCS Research & Innovation

Deep Reinforcement Learning

- Deep Q-Learning and Variants
- Policy Gradient Approaches
- Variance Reduction in Policy Gradient Methods
- Actor Crtic Algorithms
- Deterministic Policy Gradients
- Advanced Policy Gradient Methods : TRPO and PPO

Course Prerequisites

- **Prerequisites**
  - ★ Probability
  - ★ Linear Algebra
  - ★ Machine Learning
  - ★ Deep Learning

- **Programming Prerequisites**
  - ★ Good Proficiency in Python
  - ★ Tensorflow / Theano / PyTorch / Keras
  - ★ Other Associated Python Libraries

Venue and Timing

- **Mode**
  - In class lectures at LHC-3 (possibly recorded for MDS students)
- **Timing**
  - Saturday - 10.00 AM to 1.00 PM (??)
- **Course Co-ordinator**
  - Prof. Konda Reddy

Course Evaluation

- **Assignments** : Three or Four in Total (30 %)
- **Exams** : Two in Total (70 %)

Details will be in Piazza

Course Material : Books

- Reinforcement Learning : Sutton and Barto
- Reinforcement Learning and Optimal Control, Bertsekas and Tsitsiklis
- Dynamic Programming and Optimal Control (I and II) by Bertsekas

Course Material : Online Material

- David Silver's course on Reinforcement Learning
- Stanford course on Deep RL (Sergey Levine)
- Deep RL BootCamp (Pieter Abeel)
- John Schulman's lectures on Policy Gradient Methods
- ... and many others

Course Material : From India

- Prof. B. Ravindran's Course on RL (NPTEL)
- Dr. Abir Das's Course on RL (IIT KGP)
- Reinforcement Learning via Stochastic Approximation, Mathukumalli Vidyasagar, Lecture Notes, 2022 (Link to online version available in Piazza)

Attribution and Disclaimer

- Most concepts, ideas and figures, that form part of course lectures, are from several sources from across web; Most of them are listed as course material
- Care is taken to provide appropriate attribution; Omissions, if any, are regretted and unintentional
- Material prepared only for learning / teaching purpose
- Original authorship / copyright rests with the respective authors / publishers

Enjoy Learning !

Markov Decision Process

Easwar Subramanian

TCS Innovation Labs, Hyderabad

Email : cs5500.2020@iith.ac.in

August 10, 2024

Administrivia

- Please consult Prof. Konda Reddy, for all queries related to registration and other administrative issues
- If need be, register for CS 5500 instead of AI 3000 (relevant for MDS / CS students)
- The Piazza course page is ready; Enrollments are to be done
- Tentative schedule for assignments and exams are in Google sheet

Overview

Review

Types of Learning : Summary

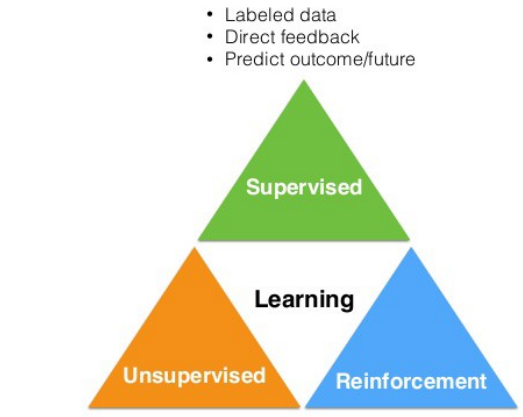
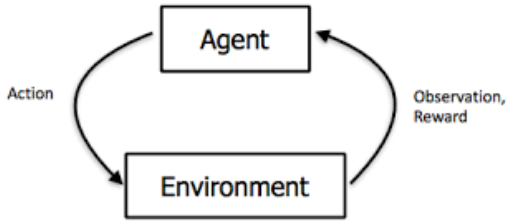


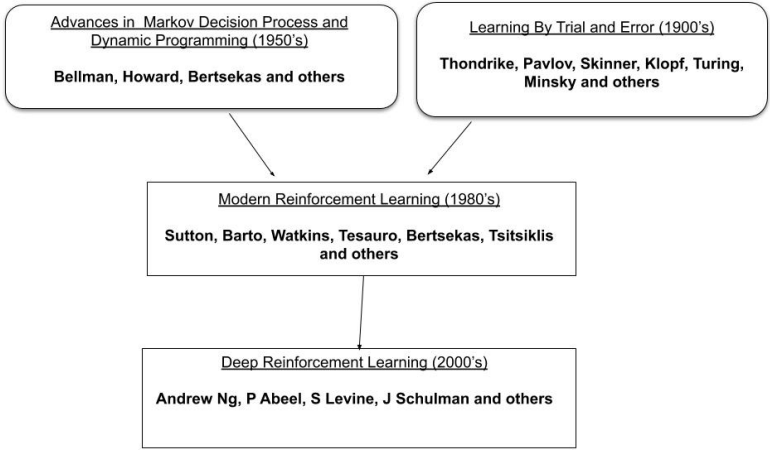
Figure Source: Saggie

Characteristics of Reinforcement Learning

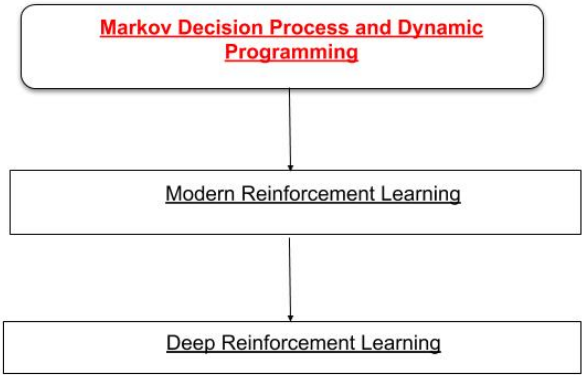


- Observations are non i.i.d and are sequential in nature
- Agent's action (may) affect the subsequent observation seen
- There is no supervisor; Only reward signal (feedback)
- Reward or feedback can be delayed

Reinforcement Learning : History



Course Setup



Mathematical Framework for Decision Making

RL Framework : Notations

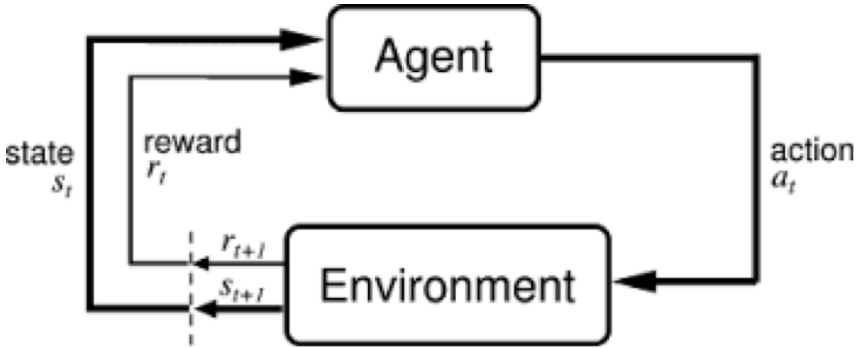


 Figure Source: Sutton and Barto

Markov Decision Process

- Markov Decision Process (MDP) provides a mathematical framework for modeling decision making process
- Can formally describe the working of the environment and agent in the RL setting
- Can handle huge variety of interesting settings
  - ★ Multi-arm Bandits - Single state MDPs
  - ★ Optimal Control - Continuous MDPs
- Core problem in solving an MDP is to find an 'optimal' policy (or behaviour) for the decision maker (agent) in order to maximize the total future reward

Markov Chains

TCS Research & Innovation

Random Variable (Non-mathematical definition)

A random variable is a variable whose value depend on the outcome of a random phenomenon

- Outcome of a coin toss
- Outcome of roll of a dice

Stochastic Process

A stochastic or random process, denoted by $\{s_t\}_{t \in T}$, can be defined as a collection of random variables that is indexed by some mathematical set T

- Index set has the interpretation of time
- The set T is, typically, $\mathbb{N}$ or $\mathbb{R}$

Notations

- Typically, in optimal control problems, the index set is continuous (say $\mathbb{R}$)
- Throughout this course (RL), the index set is always discrete (say $\mathbb{N}$)
- Let $\{s_t\}_{t \in T}$ be a stochastic process
- Let s_t be the state at time t of the stochastic process $\{s_t\}_{t \in T}$

TCS Research & Innovation

Markov Property

A state s_t of a stochastic process $\{s_t\}_{t \in T}$ is said to have Markov property if

$$P(s_{t+1}|s_t) = P(s_{t+1}|s_1, \cdots, s_t)$$

The state s_t at time t captures all relevant information from history and is a sufficient statistic of the future

TCS Research & Innovation

State Transition Probability

For a Markov state s and a successor state s', the state transition probability is defined by

$$\mathcal{P}_{ss'} = P(s_{t+1} = s' | s_t = s)$$

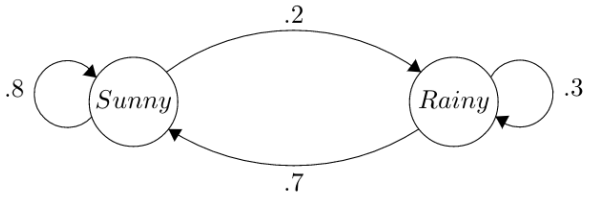
State transition matrix $\mathcal{P}$ then denotes the transition probabilities from all states s to all successor states s' (with each row summing to 1)

$$\mathcal{P} = \begin{bmatrix} \mathcal{P}_{11} & \mathcal{P}_{12} & \cdots & \mathcal{P}_{1n} \\ \vdots & & & \\ \mathcal{P}_{n1} & \mathcal{P}_{n2} & \cdots & \mathcal{P}_{nn} \end{bmatrix}$$

Markov Chain

A stochastic process $\{s_t\}_{t \in T}$ is a **Markov process** or **Markov Chain** if the sequence of random states satisfy the Markov property. It is represented by tuple $< \mathcal{S}, \mathcal{P} >$ where $\mathcal{S}$ denote the set of states and $\mathcal{P}$ denote the state transition probablity

Example 1 : Simple Two State Markov Chain

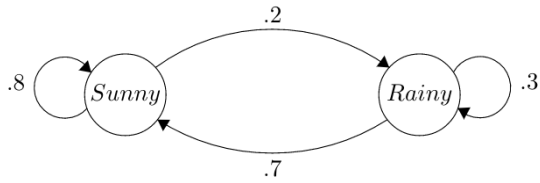


- **State** $\mathcal{S} = \{\text{Sunny}, \text{Rainy}\}$
- **Transition Probability Matrix**

$$\mathcal{P} = \begin{bmatrix} .8 & .2 \\ .7 & .3 \end{bmatrix}$$

Figure Source: https://bookdown.org/probability

Markov Chain : Example Revisited



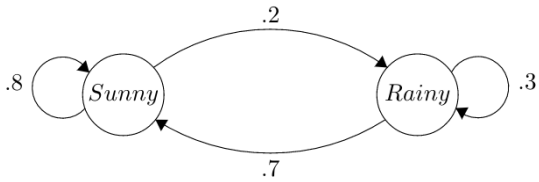
State $\mathcal{S} = \{\text{Sunny}, \text{Rainy}\}$ and **Transition Probability Matrix**

$$\mathcal{P} = \begin{bmatrix} .8 & .2 \\ .7 & .3 \end{bmatrix}$$

- Probability that tomorrow will be *'Rainy'* given today is *'Sunny'* $= 0.2$

Figure Source: https://bookdown.org/probability

Multi-Step Transitions



- Probability that day-after-tomorrow will be *'Rainy'* given today is *'Sunny'* is given by $0.2 * 0.3 + 0.8 * 0.2 = 0.22$

In general, if one step transition matrix is given by,

$$\mathcal{P} = \begin{bmatrix} P_{ss} & P_{sr} \\ P_{rs} & P_{rr} \end{bmatrix}$$

then the two step transition matrix is given by,

$$\mathcal{P}_{(2)} = \begin{bmatrix} P_{ss} * P_{ss} + P_{sr} * P_{rs} & P_{ss} * P_{sr} + P_{sr} * P_{rr} \\ P_{rr} * P_{rs} + P_{rs} * P_{ss} & P_{rr} * P_{rr} + P_{rs} * P_{sr} \end{bmatrix} = P^2$$

Figure Source: https://bookdown.org/probability

Multi-Step Transitions

In general, n-step transition matrix is given by,

$$P_{(n)} = P^n$$

Assumption

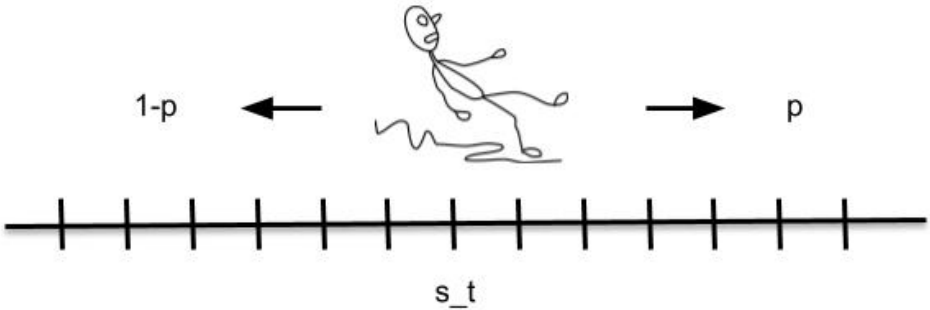
We made an important assumption in arriving at the above expression. That the one-step transition matrix stays constant through time or independent of time

- Markov chains generated using such transition matrices are called homogeneous Markov chains
- For much of this course, we will consider homogeneous Markov chains, for which the transition probabilities depend on the length of time interval $[t_1, t_2]$ but not on the exact time instants

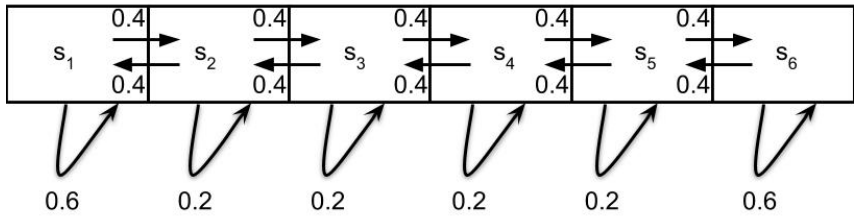
Example 2 : One dimensional random walk

A walker flips a coin every time slot to decide which 'way' to go.

$$s_{t+1} = \begin{cases} s_t + 1 & \text{with probability } p \\ s_t - 1 & \text{with probability } 1 - p \end{cases}$$



Example 3 : Simple Grid World



- $\mathcal{S} = \{s_1, s_2, s_3, s_4, s_5, s_6, s_6\}$
- $\mathcal{P}$ as shown above
- Example Markov Chains with s_2 as start state
  - ★ $\{s_2, s_3, s_2, s_1, s_2, \cdots\}$
  - ★ $\{s_2, s_2, s_3, s_4, s_3, \cdots\}$

Example 4 : Dice roll experiment

Let $\{s_t\}_{t \in T}$ model the stochastic process representing the cumulative sum of a fair six-sided die rolls

Example 5 : Natural Language Processing

Let $\{s_t\}_{t \in T}$ model the stochastic process that keeps track of the chain of letters in a sentence. Consider an example

`Tomorrow is a sunny day`

- We normally don't ask the question what is probability of character '`a`' appearing given previous character is '`d`'
- Sentence formation is typically **non-Markovian**

Notion of Absorbing State

TCS Research & Innovation

Absorbing State

A state $s \in \mathcal{S}$ is called **absorbing** state if it is impossible to leave the state. That is,

$$P_{ss'} = \left\{ \begin{array}{ll} 1, & \text{if } s = s' \\ 0, & \text{otherwise} \end{array} \right\}$$

Markov Reward Process

TCS Research & Innovation

Markov Reward Process

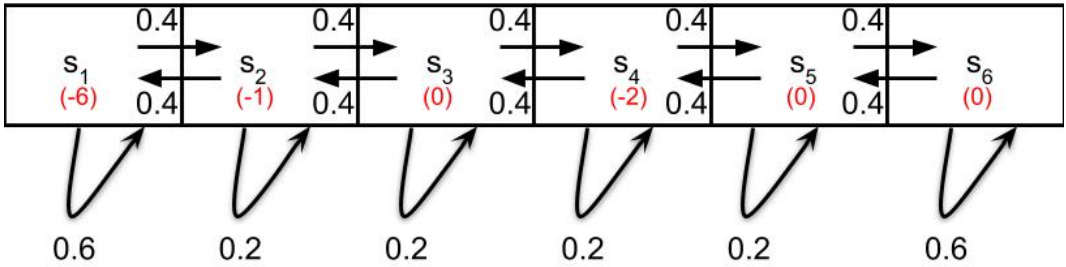
A Markov reward process is a tuple $< \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma >$ is a Markov chain with values

- $\mathcal{S}$: (Finite) set of states
- $\mathcal{P}$: State transition probablity
- $\mathcal{R}$: Reward for being in state s_t is given by a deterministic function $\mathcal{R}$

$$r_{t+1} = \mathcal{R}(s_t)$$

- γ : Discount factor such that $\gamma \in [0, 1]$

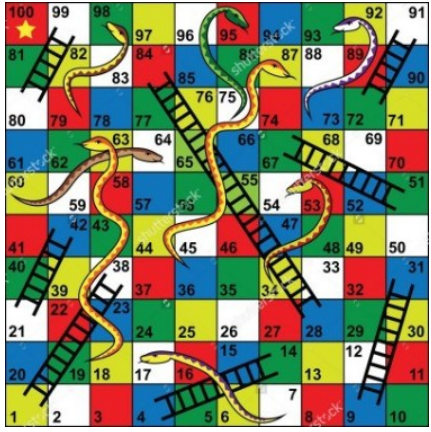
Simple Grid World : Revisited



- For the Markov chain $\{s_2, s_3, s_2, s_1, s_2, \cdots\}$ the corresponding reward sequence is $\{-1, 0, -1, -6, -1, \cdots\}$

No notion of action

Example : Snakes and Ladders



Example : Snakes and Ladders

- **States** $\mathcal{S}$: $\{s_1, s_2, \cdots, s_{100}\}$
- **Transition Probability** $\mathcal{P}$:
  - ★ What is the probability to move from state 2 to 6 in one step ?
  - ★ What are the states that can be visited in one-step from state 2 ?
  - ★ What is the probability to move from state 2 to 4 ?
  - ★ Can we transition from state 15 to 7 in one step ?

  Question : Is transition matrix independent of time ?

Question : Can we formulate the game of Snake and Ladders as a MRP ?

Need to define suitable reward function and discounting factor

On Rewards : Total Return

- At each time step t, there is a reward r_{t+1} associated with being in state s_t
- Ideally, we would like the agent to pick such trajectories in which the cumulative reward accumulated by traversing such a path is high

Question : How can we formalize this ?

Answer : If the reward sequence is given by $\{r_{t+1}, r_{t+2}, r_{t+3}, \cdots\}$, then, we want to maximize the sum

$$r_{t+1} + r_{t+2} + r_{t+3} + \cdots$$

Define G_t to be

$$G_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots = \sum_{k=0}^{\infty} r_{t+k+1}$$

The goal of the agent is to pick such paths that maximize G_t

Total (Discounted) Return

Recall that,

$$G_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots = \sum_{k=0}^{\infty} r_{t+k+1}$$

- In the case that the underlying stochastic process has infinite terms the above summation could be divergent

Therefore, we introduce discount factor $\gamma \in [0, 1]$ and redefine G_t as

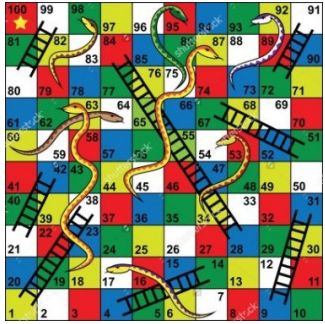
$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

- G_t is the total discounted return starting from time t
- If $\gamma < 1$ then the infinite sum has a finite value if the reward sequence is bounded
- γ close to 0 the agent is concerned only with immediate reward(s) (myopic)
- γ close to 1 the agent considers future reward more strongly (far-sighted)

Few Remarks on Discounting

- Mathematically convienient to discount rewards
- Avoids infinite returns in cyclic and infinite horizon setting
- Discount rate determines the present value of future reward
- Offers trade-off between being 'myopic' and 'far-sighted' reward
- In finite MDPs, it is sometimes possible to use undiscounted reward (i.e. $\gamma = 1$), for example, if all sequences terminate

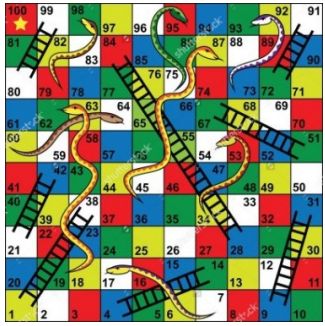
Snakes and Ladders : Revisited



Question : What can be a suitable reward function and discount factor to describe 'Snake and Ladders' as a Markov reward process ?

- **Goal** : From any given state reach s_{100} in as few steps as possible
- **Reward** $\mathcal{R}$: $\mathcal{R}(s) = -1$ for $s \in s_1, \cdots, s_{99}$ and for $R(s_{100}) = 0$
- **Discount Factor** $\gamma = 1$

Snakes and Ladders : Revisited



Question : Are all intermediate states equally 'valuable ' just because they have equal reward ?

Value Function

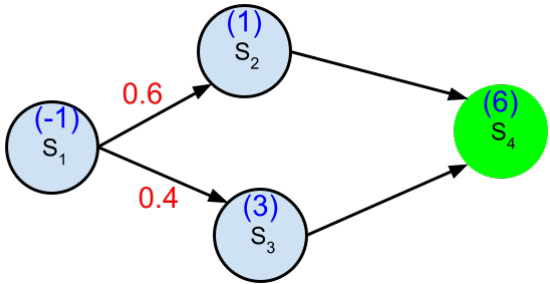
The value function $V(s)$ gives the long-term value of state $s \in \mathcal{S}$

$$V(s) = \mathbb{E}\left(G_t | s_t = s\right) = \mathbb{E}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right)$$

- Value function $V(s)$ determines the value of being in state s
- $V(s)$ measures the potential future rewards we may get from being in state s
- $V(s)$ is independent of t

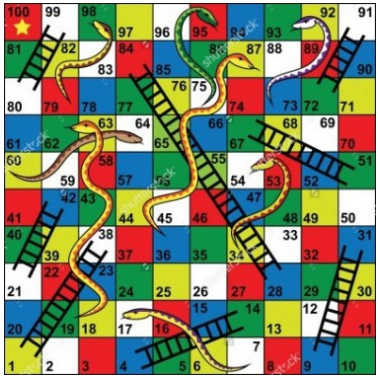
Value Function Computation : Example

Consider the following MRP. Assume $\gamma = 1$



- $V(s_1) = 6.8$
- $V(s_2) = 1 + \gamma * 6 = 7$
- $V(s_3) = 3 + \gamma * 6 = 9$
- $V(s_4) = 6$

Example : Snakes and Ladders



Question : How can we evaluate the value of each state in a large MRP such as 'Snakes and Ladders ' ?

Decomposition of Value Function

Let s and s' be successor states at time steps t and $t+1$, the value function can be decomposed into sum of two parts

- Immediate reward r_{t+1}
- Discounted value of next state s' (i.e. $\gamma V(s')$)

$$\begin{aligned} V(s) = \mathbb{E}\left(G_t | s_t = s\right) &= \mathbb{E}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right) \\ &= \mathbb{E}\left(r_{t+1} + \gamma V(s_{t+1}) | s_t = s\right) \end{aligned}$$

Decomposition of Value Function

Recall that,

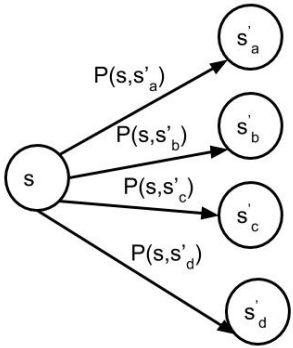
$$G_t = \left(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots\right) = \sum_{k=0}^{\infty} \left(\gamma^k r_{t+k+1}\right)$$

$$\begin{aligned}
V(s) &= \mathbb{E}\left(G_t | s_t = s\right) = \mathbb{E}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right) \\
&= \mathbb{E}\left(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s\right) \\
&= \mathbb{E}(r_{t+1} | s_t = s) + \sum_{k=1}^{\infty} \gamma^k \, \mathbb{E}\left(r_{t+k+1} | s_t = s\right) \\
&= \mathbb{E}(r_{t+1} | s_t = s) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s) \sum_{k=0}^{\infty} \gamma^k \, \mathbb{E}\left(r_{t+k+1} | s_t = s, s_{t+1} = s'\right) \\
&= \mathbb{E}(r_{t+1} | s_t = s) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s) \sum_{k=0}^{\infty} \gamma^k \, \mathbb{E}\left(r_{t+k+1} | s_{t+1} = s'\right) \quad \text{(Markov property)} \\
&= \mathbb{E}(r_{t+1} + \gamma V(s_{t+1}) | s_t = s)
\end{aligned}$$

Value Function : Evaluation

We have

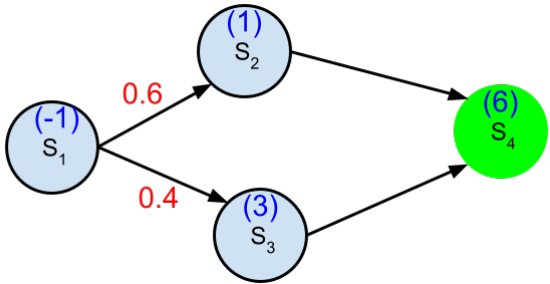
$$V(s) = \mathbb{E}(r_{t+1} + \gamma V(s_{t+1}) | s_t = s)$$



$$V(s) = \mathcal{R}(s) + \gamma \left[\mathcal{P}_{ss'_a} V(s'_a) + \mathcal{P}_{ss'_b} V(s'_b) + \mathcal{P}_{ss'_c} V(s'_c) + \mathcal{P}_{ss'_d} V(s'_d) \right]$$

Value Function Computation : Example

Consider the following MRP. Assume $\gamma = 1$



- $V(s_4) = 6$
- $V(s_3) = 3 + \gamma * 6 = 9$
- $V(s_2) = 1 + \gamma * 6 = 7$
- $V(s_1) = -1 + \gamma * (0.6 * 7 + 0.4 * 9) = 6.8$

Bellman Equation for Markov Reward Process

$$V(s) = \mathbb{E}(r_{t+1} + \gamma V(s_{t+1}) | s_t = s)$$

For any $s' \in \mathcal{S}$ a successor state of s with transition probability $\mathcal{P}_{ss'}$, we can rewrite the above equation as (using definition of Expectation)

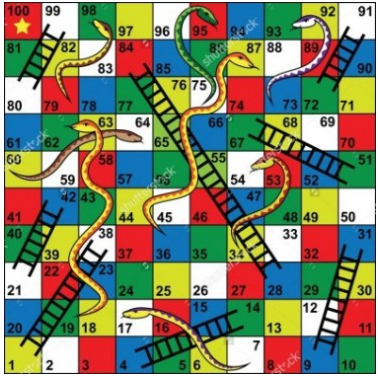
$$\boxed{V(s) = \mathbb{E}(r_{t+1} | s_t = s) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} V(s')}$$

This is the **Bellman Equation** for value functions

Snakes and Ladders

Question : How can we evaluate the value of (all) states using the value function decomposition ?

$$V(s) = \mathbb{E}(r_{t+1}|s_t = s) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} V(s')$$



Bellman Equation in Matrix Form

Let $\mathcal{S} = \{1, 2, \cdots, n\}$ and $\mathcal{P}$ be known. Then one can write the Bellman equation can as,

$$V = \mathcal{R} + \gamma \mathcal{P} V$$

where

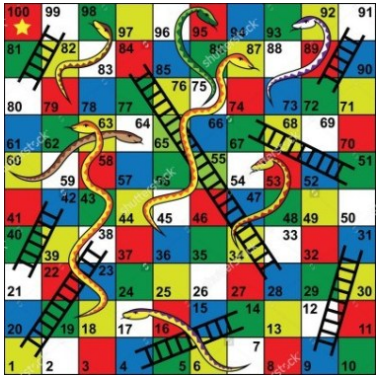
$$\begin{bmatrix} V(1) \\ V(2) \\ \vdots \\ V(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}(1) \\ \mathcal{R}(2) \\ \vdots \\ \mathcal{R}(n) \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \mathcal{P}_{12} & \cdots & \mathcal{P}_{1n} \\ \mathcal{P}_{21} & \mathcal{P}_{22} & \cdots & \mathcal{P}_{2n} \\ \vdots & & & \\ \mathcal{P}_{n1} & \mathcal{P}_{n2} & \cdots & \mathcal{P}_{nn} \end{bmatrix} \times \begin{bmatrix} V(1) \\ V(2) \\ \vdots \\ V(n) \end{bmatrix}$$

Solving for V, we get,

$$\boxed{V = (I - \gamma \mathcal{P})^{-1} \mathcal{R}}$$

The discount factor should be $\gamma < 1$ for the inverse to exist

Example : Snakes and Ladders



- We can now compute the value of states in such 'large' MRP using the matrix form of Bellman equation
- Value function computed for a particular state provides the **expected** number of plays to reach the goal state s_{100} from that state

Markov Decision Process

Markov Decision Process

Markov decision process is a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ where

- $\mathcal{S}$: (Finite) set of states
- $\mathcal{A}$: (Finite) set of actions
- $\mathcal{P}$: State transition probability

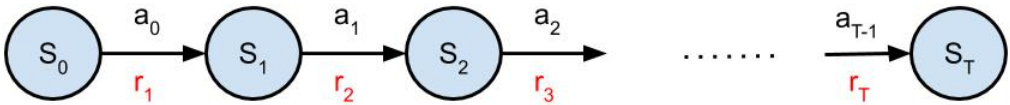
$$\mathcal{P}^{a}_{ss'} = \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a), a_t \in \mathcal{A}$$

- $\mathcal{R}$: Reward for taking action a_t at state s_t and transitioning to state s_{t+1} is given by the deterministic function $\mathcal{R}$

$$r_{t+1} = \mathcal{R}(s_t, a_t, s_{t+1})$$

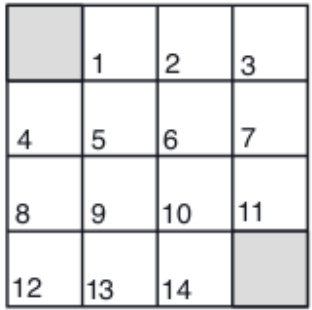
- γ : Discount factor such that $\gamma \in [0, 1]$

Wealth Management Problem



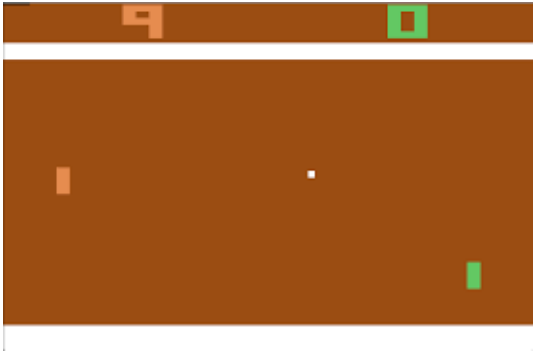
- States $\mathcal{S}$: Current value of the portfolio and current valuation of instruments in the portfolio
- Actions $\mathcal{A}$: Buy / Sell instruments of the portfolio
- Reward $\mathcal{R}$: Return on portfolio compared to previous decision epoch

Navigation Problem



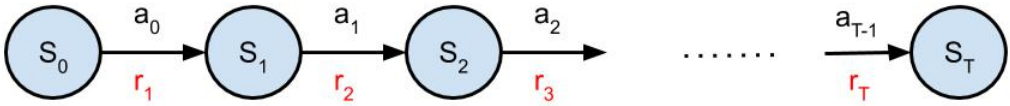
- States $\mathcal{S}$: Squares of the grid
- Actions $\mathcal{A}$: Any of the four directions possible
- Reward $\mathcal{R}$: -1 for every move made until reaching goal state

Example : Atari Games



- States $\mathcal{S}$: Possible set of all (Atari) images
- Actions $\mathcal{A}$: Move the paddle up or down
- Reward $\mathcal{R}$: +1 for making the opponent miss the ball; -1 if the agent miss the ball; 0 otherwise;

Flow Diagram



- The goal is to choose a sequence of actions such that the expected total discounted future reward $\mathbb{E}(G_t|s_t = s)$ is maximized where

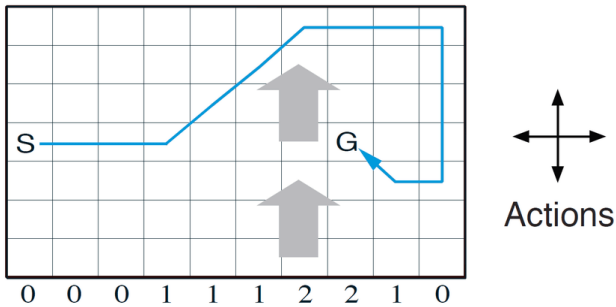
$$G_t = \sum_{k=0}^{\infty} \left(\gamma^k r_{t+k+1}\right)$$

Windy Grid World : Stochastic Environment

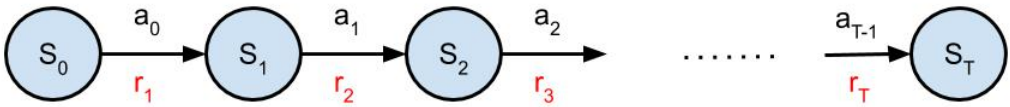
Recall given an MDP $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$, we have the state transition probability $\mathcal{P}$ defined as

$$\mathcal{P}^{a}_{ss'} = \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a), a_t \in \mathcal{A}$$

- In general, note that even after choosing action a at state s (as prescribed by the policy) the next state s' need not be a fixed state

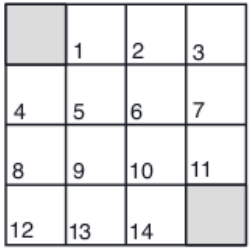


Finite and Infinite Horizon MDPs



- If T is fixed and finite, the resultant MDP is a finite horizon MDP
  - ★ Wealth management problem
- If T is infinite, the resultant MDP is infinite horizon MDP
  - ★ Certain Atari games
- When $|\mathcal{S}|$ is finite, the MDP is called finite state MDPs

Grid World Example



Question : Is Grid world finite / infinite horizon problem ? Why ?

(Stochastic shortest path MDPs)

- For finite horizon MDPs and stochastic shortest path MDPs, one can use $\gamma = 1$

Notion of Policy and Optimal Policies

Easwar Subramanian

TCS Innovation Labs, Hyderabad

cs5500.2020@iith.ac.in

August 17, 2024

Overview

TCS Research & Innovation

Review

Markov Reward Process

A Markov reward process is a tuple $< \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma >$ is a Markov chain with values

- $\mathcal{S}$: (Finite) set of states
- $\mathcal{P}$: State transition probablity
- $\mathcal{R}$: Reward for being in state s_t is given by a deterministic function $\mathcal{R}$

$$r_{t+1} = \mathcal{R}(s_t)$$

- γ : Discount factor such that $\gamma \in [0, 1]$

- In general, the reward function can also be an expectation $\mathcal{R}(s_t = s) = \mathbb{E}[r_{t+1}|s_t = s]$

No notion of action !

Value Function

The value function $V(s)$ gives the long-term value of state $s \in \mathcal{S}$

$$V(s) = \mathbb{E}\left(G_t | s_t = s\right) = \mathbb{E}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right)$$

- Value function $V(s)$ determines the value of being in state s
- $V(s)$ measures the potential future rewards we may get from being in state s
- Observe that $V(s)$ is independent of t

Decomposition of Value Function

Let s and s' be successor states at time steps t and $t+1$, the value function can be decomposed into sum of two parts

- Immediate reward r_{t+1}
- Discounted value of next state s' (i.e. $\gamma V(s')$)

$$\begin{aligned} V(s) = \mathbb{E}\left(G_t | s_t = s\right) &= \mathbb{E}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right) \\ &= \mathbb{E}\left(r_{t+1} + \gamma V(s_{t+1}) | s_t = s\right) \end{aligned}$$

Bellman equation for value functions

$$\boxed{V(s) = \mathbb{E}(r_{t+1} | s_t = s) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} V(s')}$$

Bellman Equation in Matrix Form

We have $\mathcal{S} = \{1, 2, \cdots, n\}$ and let $\mathcal{P}$, $\mathcal{R}$ be known. Then one can write the Bellman equation can as,

$$V = \mathcal{R} + \gamma \mathcal{P} V$$

where

$$\begin{bmatrix} V(1) \\ V(2) \\ \vdots \\ V(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}(1) \\ \mathcal{R}(2) \\ \vdots \\ \mathcal{R}(n) \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \mathcal{P}_{12} & \cdots & \mathcal{P}_{1n} \\ \mathcal{P}_{21} & \mathcal{P}_{22} & \cdots & \mathcal{P}_{2n} \\ \vdots & & & \\ \mathcal{P}_{n1} & \mathcal{P}_{n2} & \cdots & \mathcal{P}_{nn} \end{bmatrix} \times \begin{bmatrix} V(1) \\ V(2) \\ \vdots \\ V(n) \end{bmatrix}$$

Solving for V, we get,

$$\boxed{V = (I - \gamma \mathcal{P})^{-1} \mathcal{R}}$$

The discount factor should be $\gamma < 1$ for the inverse to exist

Markov Decision Process

Markov decision process is a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ where

- $\mathcal{S}$: (Finite) set of states
- $\mathcal{A}$: (Finite) set of actions
- $\mathcal{P}$: State transition probability

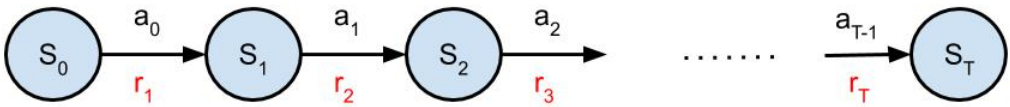
$$\mathcal{P}^a_{ss'} = \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a), a_t \in \mathcal{A}$$

- $\mathcal{R}$: Reward for taking action a_t at state s_t and transitioning to state s_{t+1} is given by the deterministic function $\mathcal{R}$

$$r_{t+1} = \mathcal{R}(s_t, a_t, s_{t+1})$$

- γ : Discount factor such that $\gamma \in [0, 1]$

Finite and Infinite Horizon MDPs



Depending on time horizon, a Markov decision process can be

- Finite horizon
- Infinite horizon
- Indefinite horizon (SSP)

For finite and (certain) indefinite MDPs with at least absorbing state, we can take the discount factor to be 1

Policy

Policy

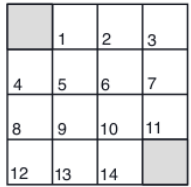
Let π denote a policy that maps state space $\mathcal{S}$ to action space $\mathcal{A}$

Policy

- Deterministic policy: $a = \pi(s), s \in \mathcal{S}, a \in \mathcal{A}$
- Stochastic policy $\pi(a|s) = P[a_t = a|s_t = s]$

Grid World : Revisited

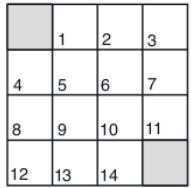
Consider a 4×4 grid world problem



- $\mathcal{S}$: $\{1, 2, \cdots, 14\}$ (non-terminal) + 2 terminal states (shaded)
- $\mathcal{A}$: {Right, Left, Up, Down}

Figure Source: David Silver's RL course

Grid World : Deterministic Policy



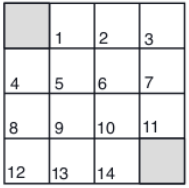
- $\mathcal{S}$: $\{1, 2, \cdots, 14\}$ (non-terminal) + 2 terminal states (shaded)
- $\mathcal{A}$: {Right, Left, Up, Down}
- **Deterministic policy** :

$$\pi(s) = \left\{ \begin{array}{ll} \text{Down}, & \text{if } s = \{3, 7, 11\} \\ \text{Right}, & \text{Otherwise} \end{array} \right\}$$

- Example sequences : $\{\{8, 9, 10, 11, G\}, \{2, 3, 7, 11, G\}\}$

Figure Source: David Silver's RL course

Grid World : Stochastic Policy



- $\mathcal{S}$: $\{1, 2, \cdots, 14\}$ (non-terminal) + 2 terminal states (shaded)
- $\mathcal{A}$: {Right, Left, Up, Down}
- **Stochastic policy** : $\pi(a|s)$ could be a uniform random action between all available actions at state s
- Example sequences : $\{\{8, 4, 8, 9, 13, \cdots, \}, \{2, 6, 5, 9, 13, \cdots, \}\}$

Figure Source: David Silver's RL course

Stochastic Policy : Rock Scissors Paper

- Two player game of rock-paper-scissors
  - ★ Scissors beats paper
  - ★ Rock beats scissors
  - ★ Paper beats rock
- Consider policies for iterated rock-paper-scissors
  - ★ A deterministic policy is easily exploited
  - ★ A uniform random policy is optimal (i.e. Nash equilibrium)

Policy Evaluation

Value Functions with Policy

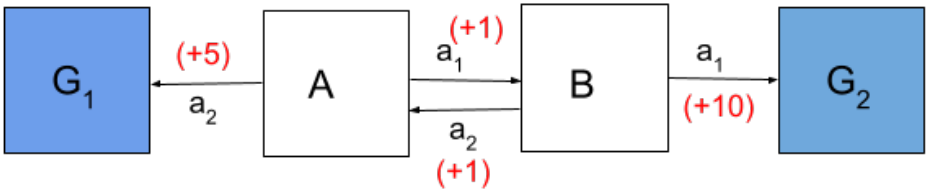
Given a MDP and a policy π, we define the value of a policy as follows :

State-value function

The value function $V^{\pi}(s)$ in state s is the expected (discounted) total return starting from state s and then following the policy π

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left(\sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} | s_t = s\right)$$

Value Function Computation : Example



- States $\mathcal{S} = \{A, B, G_1, G_2\}$; States G_1 and G_2 are terminal states
- Two actions $\mathcal{A} = \{a_1, a_2\}$
- Value of states $\{A, B\}$ using forward policy π_f is given by, $V_{\pi_f}(A) = 11$, $V_{\pi_f}(B) = 10$
- Value of states $\{A, B\}$ using backward policy π_b is given by, $V_{\pi_b}(B) = 6$, $V_{\pi_b}(A) = 5$

Decomposition of State Value Function

The state-value function can be decomposed into immediate reward plus discounted value of successor state

$$V^{\pi}(s) = \mathbb{E}_{\pi}(r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s)$$

Expanding the expectation, with $\mathcal{R}^a_{ss'} = \mathcal{R}(s, a, s')$ we get,

$$\mathbb{E}_{\pi}[r_{t+1} | s_t = s] = \sum_a \pi(a|s) \sum_{s'} \mathcal{P}^a_{ss'} \mathcal{R}^a_{ss'}$$

and

$$\mathbb{E}_{\pi}[\gamma V^{\pi}(s_{t+1}) | s_t = s] = \sum_a \pi(a|s) \sum_{s'} \mathcal{P}^a_{ss'} \gamma V^{\pi}(s')$$

Hence,

$$\boxed{V^{\pi}(s) = \sum_a \pi(a|s) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s')\right]}$$

The above equation is called the Bellman Evaluation operator

Matrix Formulation of Bellman Evaluation Equation

$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s')\right]$$

Denote,

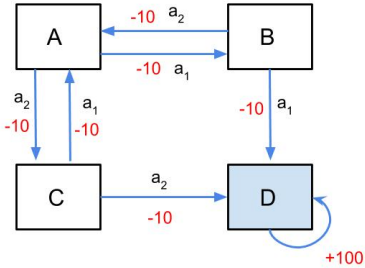
$$\begin{aligned} \mathcal{P}^{\pi}(s'|s) &= \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}^{a}_{ss'} \\ \mathcal{R}^{\pi}(s) &= \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s'} \mathcal{P}^{a}_{ss'} \mathcal{R}^{a}_{ss'} = \mathbb{E}(r_{t+1}|s_t = s) \end{aligned}$$

Using $\mathcal{P}^{\pi}$ and $\mathcal{R}^{\pi}$, for finite state MDP, one can rewrite the Bellman evaluation equation as

$$V^{\pi} = \mathcal{R}^{\pi} + \gamma \mathcal{P}^{\pi} V^{\pi} \implies V^{\pi} = (I - \gamma \mathcal{P}^{\pi})^{-1} \mathcal{R}^{\pi}$$

Remark : Bellman Evaluation Equation for $V^{\pi}(s)$ is a system of $n = |\mathcal{S}|$ (linear) equations with n variables and can be solved if the model is known

Value Function Computation : Example



- States $\mathcal{S} = \{A, B, C, D\}$; State D is terminal state
- Two actions $\mathcal{A} = \{a_1, a_2\}$
- Stochastic Environment with action chosen succeeding 90% and failing 10%
- Upon failure, agent moves in the direction suggested by the other action

Value Function Computation : Example

- Consider a deterministic policy (π_1) that chooses action a_1 in all states
- Transition matrix corresponding to policy π_1 is given by

$$\begin{bmatrix} & A & B & C & D \\ A & 0 & 0.9 & 0.1 & 0 \\ B & 0.1 & 0 & 0 & 0.9 \\ C & 0.9 & 0 & 0 & 0.1 \\ D & 0 & 0 & 0 & 1 \end{bmatrix}$$

- Value of the states under the policy π_1 is given by,
  - $V^{\pi_1}(D) = 100$
  - $V^{\pi_1}(A) = 0.9 * [-10 + V^{\pi_1}(B)] + 0.1 * [-10 + V^{\pi_1}(C)]$
  - $V^{\pi_1}(B) = 0.9 * [-10 + V^{\pi_1}(D)] + 0.1 * [-10 + V^{\pi_1}(A)]$
  - $V^{\pi_1}(C) = 0.9 * [-10 + V^{\pi_1}(A)] + 0.1 * [-10 + V^{\pi_1}(D)]$
- $V^{\pi_1} = \{75.61, 87.56, 68.05, 100\}$;

Value Function Computation : Example

- Consider a deterministic policy (π_2) that chooses action a_2 in all states
- Transition matrix corresponding to policy π_2 is given by

$$\begin{bmatrix} & A & B & C & D \\ A & 0 & 0.1 & 0.9 & 0 \\ B & 0.9 & 0 & 0 & 0.1 \\ C & 0.1 & 0 & 0 & 0.9 \\ D & 0 & 0 & 0 & 1 \end{bmatrix}$$

- Value of the states under the policy π_2 is given by,
  - $V^{\pi_2}(D) = 100$
  - $V^{\pi_2}(A) = 0.9 * [-10 + V^{\pi_2}(C)] + 0.1 * [-10 + V^{\pi_2}(D)]$
  - $V^{\pi_2}(B) = 0.9 * [-10 + V^{\pi_2}(A)] + 0.1 * [-10 + V^{\pi_2}(D)]$
  - $V^{\pi_2}(C) = 0.9 * [-10 + V^{\pi_2}(D)] + 0.1 * [-10 + V^{\pi_2}(A)]$
- $V^{\pi_2} = \{75.61, 68.05, 87.56, 100\}$;

MDP + Policy = MRP

- MDP + policy = Markov Reward Process.
- Given a MDP $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ and a policy π
- The MRP is given by $(\mathcal{S}, \mathcal{P}^{\pi}, \mathcal{R}^{\pi}, \gamma)$

Action Value Function

Action Value Function

Action-value function

The action-value function $Q(s,a)$ under policy π is the expected return starting from state s and taking action a and then following the policy π

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left(\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+1} | s_t = s, a_t = a\right)$$

The action-value function can similarly be decomposed as

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}(r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) | s_t = s, a_t = a)$$

Expanding the expectation we have $Q^{\pi}(s,a)$ to be

$$Q^{\pi}(s,a) = \sum_{s'} \mathcal{P}^{a}_{ss'}\left[\mathcal{R}^{a}_{ss'} + \gamma \sum_{a'} \pi(a'|s') Q^{\pi}(s',a')\right]$$

Relationship between $V^{\pi}(\cdot)$ and $Q^{\pi}(\cdot)$

Using definitions of $V^{\pi}(s)$ and $Q^{\pi}(s,a)$, we can arrive at the following relationships

$$V^{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) Q^{\pi}(s,a)$$

$$Q^{\pi}(s,a) = \sum_{s' \in \mathcal{S}} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s') \right]$$

Optimality in Policies

Optimal Policy

Define a partial ordering over policies

$$\pi \geq \pi', \quad \text{if} \quad V^{\pi}(s) \geq V^{\pi'}(s), \quad \forall s \in \mathcal{S}$$

Theorem

- There exists an optimal policy π_* that is better than or equal to all other policies.
- All optimal policies achieve the optimal value function, $V_*(s) = V^{\pi_*}(s)$
- All optimal policies achieve the optimal action-value function, $Q_*(s,a) = Q^{\pi_*}(s,a)$

Grid World Problem

Consider a 4×4 grid world problem

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$R_t = -1$
on all transitions

- $\mathcal{S}$: $\{1, 2, \cdots, 14\}$ (non-terminal) + 2 terminal states (shaded)
- $\mathcal{A}$: {East, West, North, South}
- $\mathcal{P}$: Upon choosing an action from $\mathcal{A}$, state transitions are deterministic; except the actions that would take the agent off the grid in fact leave the state unchanged
- $\mathcal{R}$: Reward is -1 on all transitions until the terminal state is reached

Grid World Problem

actions

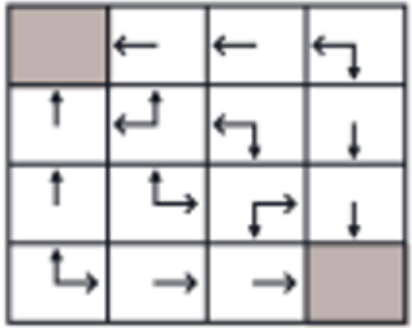
	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$R_t = -1$
on all transitions

Goal : Reach any of the goal state in as minimum plays as possible

Question : What could be an optimal policy to achieve the above objective ?

Grid World Problem : Optimal Policies



Question : How many optimal policies are there ?

Answer : There are infinite optimal policies (including some deterministic ones)

Solution to an MDP

Solving an MDP means finding a policy π_* as follows

$$\pi_* = \arg\max_{\pi} \left[\mathbb{E}_{\pi} \left(\sum_{t=0}^{\infty} \gamma^t r_{t+1} \right) \right]$$

is **maximum**

- Denote optimal value function $V_*(s) = V^{\pi_*}(s)$
- Denote optimal action value function $Q_*(s, a) = Q^{\pi_*}(s, a)$
- The main goal in RL or solving an MDP means finding an **optimal value function** V_* or **optimal action value function** Q_* or **optimal policy** π_*

Finding an Optimal Policy

Question : Suppose we are given $Q_*(s, a)$. Can we find an optimal policy ?

Answer : An optimal policy can be found by maximising over $Q_*(s, a)$

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_{a \in \mathcal{A}} Q_*(s, a) \\ 0 & \text{Otherwise} \end{cases}$$

- If we know $Q_*(s, a)$, we immediately have an optimal policy
- There is always a deterministic optimal policy for any MDP

Greedy (Optimal) Policy

Greedy policy with respect to optimal (action) value function is an optimal policy

An optimal policy can be found by maximising over $Q_*(s, a)$

$$\pi_*(s) = \begin{cases} 1 & \text{if } a = \arg\max_{a \in \mathcal{A}} Q_*(s, a) \\ 0 & \text{Otherwise} \end{cases}$$

Greedy Policy

For a given $Q^{\pi}(\cdot,\cdot)$, define $\pi'(s)$ as follows

$$\pi'(s) = \text{greedy}(Q) = \begin{cases} 1 & \text{if } a = \arg\max_{a \in \mathcal{A}} Q^{\pi}(s,a) \\ 0 & \text{Otherwise} \end{cases}$$

For a given $V^{\pi}(\cdot)$, define $\pi'(s)$ as follows

$$\pi'(s) = \text{greedy}(V) = \begin{cases} 1 & \text{if } a = \arg\max_{a \in \mathcal{A}} \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^{a}_{ss'} \left(\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s')\right)\right] \\ 0 & \text{Otherwise} \end{cases}$$

Relationship between $V_*(\cdot)$ and $Q_*(\cdot, \cdot)$

Question : Suppose we are given $Q_*(s, a), \forall s \in \mathcal{S}$. Can we find $V_*(s)$?

$$V_*(s) = \max_a Q_*(s, a)$$

Question : Suppose we are given $V_*(s), \forall s \in \mathcal{S}$. Can we find $Q_*(s, a)$?

$$Q_*(s, a) = \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V_*(s') \right) \right]$$

Exact Methods

Policy Iteration

Question : Is there a way to arrive at π_* starting from an arbitrary policy π ?

Answer : Policy Iteration

- Evaluate the policy π
  - ★ Compute $V^\pi(s) = \mathbb{E}_\pi(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s)$
- Improve the policy π

$$\pi'(s) = \text{greedy}(V^\pi(s))$$

$$\pi_0 \xrightarrow{\text{E}} V^{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} V^{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi^* \xrightarrow{\text{E}} V^*,$$

Policy Evaluation

- **Problem** : Evaluate a given policy π
- Compute $V^{\pi}(s) = \mathbb{E}_{\pi}(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s)$
- **Solution 1** : Solve a system of linear equations using any solver
- **Solution 2** : Iterative application of Bellman Evaluation Equation
- Iterative update rule :

$$V^{\pi}_{k+1}(s) \leftarrow \sum_{a} \pi(a|s) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}_{k}(s')\right]$$

- The sequence of value functions $\{V^{\pi}_1, V^{\pi}_2, \cdots, \}$ converge to V^{π}

Policy Improvement

Suppose we know V^π. How to improve policy π ?

The answer lies in the definition of action value function $Q^\pi(s,a)$. Recall that,

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_\pi\left(\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t=s,a_t=a\right) \\ &= \mathbb{E}(r_{t+1}+\gamma V^\pi(s_{t+1})|s_t=s,a_t=a) \\ &= \sum_{s'\in\mathcal{S}}\mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'}+\gamma V^\pi(s')\right] \end{aligned}$$

- If $Q^\pi(s,a) > V^\pi(s) \implies$ Better to select action a in state s and thereafter follow the policy π
- This is a special case of the policy improvement theorem

Policy Iteration : Algorithm

Algorithm Policy Iteration

1: Start with an initial policy π_1
2: **for** $i = 1, 2, \cdots, N$ **do**
3: Evaluate $V^{\pi_i}(s) \quad \forall s \in \mathcal{S}$. That is,
4: **for** $k = 1, 2, \cdots, K$ **do**
5: For all $s \in \mathcal{S}$ calculate

$$V_{k+1}^{\pi_i}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} \mathcal{P}_{ss'}^a \left[\mathcal{R}_{ss'}^a + \gamma V_k^{\pi_i}(s')\right]$$

6: **end for**
7: Perform policy Improvement

$$\pi_{i+1} = \text{greedy}(V^{\pi_i})$$

8: **end for**

Value Iteration

Question : Is there a way to arrive at V_* starting from an arbitrary value function V_0 ?

Answer : Value Iteration

Optimality Equation for State Value Function

Recall the Bellman Evaluation Equation for an MDP with policy π

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s')\right]$$

Question : Can we have a recursive formulation for $V_*(s)$?

$$V_*(s) = \max_{a} Q_*(s,a) = \max_{a} \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^{a}_{ss'} \left(\mathcal{R}^{a}_{ss'} + \gamma V_*(s')\right)\right]$$

Value Iteration : Idea

- Suppose we know the value $V_*(s')$
- Then the solution $V_*(s)$ can be found by one step look ahead

$$V_*(s) \leftarrow \max_a \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V_*(s') \right) \right]$$

- Idea of value iteration is to perform the above updates iteratively

Value Iteration : Algorithm

Algorithm Value Iteration

1: Start with an initial value function $V_1(\cdot)$;
2: **for** $k = 1, 2, \cdots, K$ **do**
3: **for** $s \in \mathcal{S}$ **do**
4: Calculate

$$V_{k+1}(s) \leftarrow \max_a \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V_k(s') \right) \right]$$

5: **end for**
6: **end for**

Exact Methods : Value and Policy Iteration

Easwar Subramanian

TCS Innovation Labs, Hyderabad

Email : cs5500.2020@iith.ac.in

August 24, 2024

Overview

Review

Optimal Policy

Define a partial ordering over policies

$$\pi \geq \pi', \quad \text{if} \quad V^{\pi}(s) \geq V^{\pi'}(s), \quad \forall s \in \mathcal{S}$$

Theorem

- There exists an optimal policy π_* that is better than or equal to all other policies.
- All optimal policies achieve the optimal value function, $V_*(s) = V^{\pi_*}(s)$
- All optimal policies achieve the optimal action-value function, $Q_*(s,a) = Q^{\pi_*}(s,a)$

Solution to an MDP

Solving an MDP means finding a policy π_* as follows

$$\pi_* = \arg\max_{\pi} \left[\mathbb{E}_{\pi} \left(\sum_{t=0}^{\infty} \gamma^t r_{t+1} \right) \right]$$

is **maximum**

- Denote optimal value function $V_*(s) = V^{\pi_*}(s)$
- Denote optimal action value function $Q_*(s, a) = Q^{\pi_*}(s, a)$
- The main goal in RL or solving an MDP means finding an **optimal value function** V_* or **optimal action value function** Q_* or **optimal policy** π_*

Value Iteration and its Convergence

Value Iteration

Question : Is there a way to arrive at V_* starting from an arbitrary value function V_0 ?

Answer : Value Iteration

Bellman Evaluation Equation

$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s')\right]$$

- For a MDP with $\mathcal{S} = n$, Bellman Evaluation Equation for $V^{\pi}(s)$ is a system of $n = |\mathcal{S}|$ (linear) equations with n variables and can be solved if the model is known

Denote,

$$\begin{aligned} \mathcal{P}^{\pi}(s'|s) &= \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}^{a}_{ss'} \\ \mathcal{R}^{\pi}(s) &= \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s'} \mathcal{P}^{a}_{ss'} \mathcal{R}^{a}_{ss'} = \mathbb{E}(r_{t+1}|s_t = s) \end{aligned}$$

$$V^{\pi} = \mathcal{R}^{\pi} + \gamma \mathcal{P}^{\pi} V^{\pi} \implies V^{\pi} = (I - \gamma \mathcal{P}^{\pi})^{-1} \mathcal{R}^{\pi}$$

Optimality Equation for State Value Function

Question : Can we have a recursive formulation for $V_*(s)$?

$$V_*(s) = \max_a Q_*(s,a) = \max_a \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V_*(s') \right) \right]$$

Question : These are also a system of equations with $n = |\mathcal{S}|$ with n variables. Can we solve them ?

Answer : Optimality equations are non-linear system of equations with n unknowns and n non-linear constraints (i.e., the max operator).

Solving the Bellman Optimality Equation

- Bellman optimality equations are non-linear
- In general, there are no closed form solutions
- Iterative methods are typically used

Principle of Optimality

The tail of an optimal policy must be optimal

- Any optimal policy can be subdivided into two components; an optimal first action, followed by an optimal policy from successor state s'.

Solution Methodology : Dynamic Programming

Bellman optimality equation :

$$V_*(s) = \max_a \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V_*(s') \right) \right]$$

Optimal Substructure : Optimal solution can be constructed from optimal solutions to subproblems

Overlapping Subproblems : Problem can be broken down into subproblems and can be reused several times

- Markov Decision Processes, generally, satisfy both these characterstics
- Dynamic Programming is a popular solution method for problems having such properties

Value Iteration : Idea

- Suppose we know the value $V_*(s')$
- Then the solution $V_*(s)$ can be found by one step look ahead

$$V_*(s) \leftarrow \max_a \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V_*(s') \right) \right]$$

- Idea of value iteration is to perform the above updates iteratively

Value Iteration : Algorithm

Algorithm Value Iteration

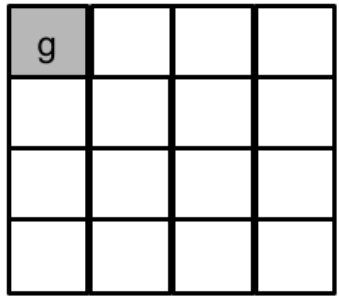
1: Start with an initial value function $V_1(\cdot)$;
2: **for** $k = 1, 2, \cdots, K$ **do**
3: **for** $s \in \mathcal{S}$ **do**
4: Calculate

$$V_{k+1}(s) \leftarrow \max_a \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V_k(s') \right) \right]$$

5: **end for**
6: **end for**

Value Iteration : Example

No noise and discount factor $\gamma = 1$



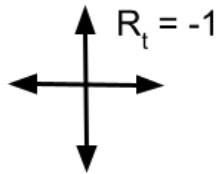
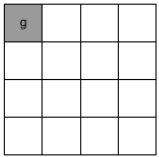


Figure Source: David Silver's UCL Course

Value Iteration : Example

$$V_{k+1}(s) \leftarrow \max_a \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V_k(s') \right) \right]$$



Problem

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

V_1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1

V_2

0	-1	-2	-2
-1	-2	-2	-2
-2	-2	-2	-2
-2	-2	-2	-2

V_3

0	-1	-2	-3
-1	-2	-3	-3
-2	-3	-3	-3
-3	-3	-3	-3

V_4

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-4
-3	-4	-4	-4

V_5

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-5

V_6

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-6

V_7

Value Iteration : Remarks

- The sequence of value functions $\{V_1, V_2, \cdots, \}$ converge
- It converges to V_*
- Convergence is independent of the choice of V_0.
- Intermediate value functions need not correspond to a policy in the sense of satisfying the Bellman Evaluation Equation
- However, for any k, one can come up with a greedy policy as follows

$$\pi_{k+1}(s) \leftarrow \text{greedy} V_k(s)$$

- The crux of proving the above statements lie in **Banach Fixed Point Theorem / Contraction Mapping Theorem**

Optimality Equation for Action-Value Function

There is a recursive formulation for $Q_*(\cdot, \cdot)$

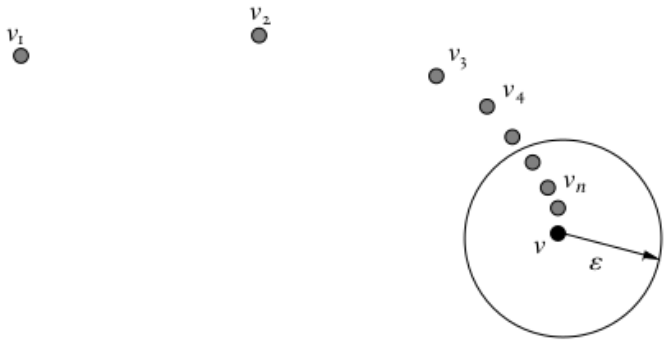
$$Q_*(s,a) = \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q_*(s', a') \right) \right]$$

One could similarly conceive an iterative algorithm to compute optimal Q_* using the above recursive formulation !!

Convergence

Let $\mathcal{V}$ be a vector space. A sequence of vectors $\{v_n\} \in \mathcal{V}$ (with $n \in \mathbb{N}$) is said to converge to v if and only if

$$\lim_{n \to \infty} \|v_n - v\| = 0$$



TCS Research & Innovation

Cauchy Sequence

A sequence of vectors $\{v_n\} \in \mathcal{V}$ (with $n \in \mathbb{N}$) is said to be a Cauchy sequence, if and only if, for each $\varepsilon > 0$, there exists an N_ε such that $\|v_n - v_m\| \leq \varepsilon$ for any $n, m > N_\varepsilon$

ustration

TCS TATA CONSULTANCY SERVICES

Notion of Completeness

TCS Research & Innovation

Completeness

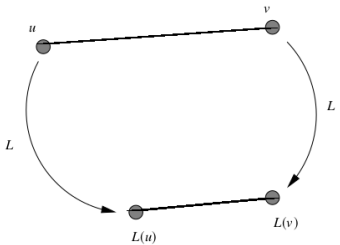
A **normed vector space** $(\mathcal{V}, \|\cdot\|)$ is complete, if and only if, every Cauchy sequence in $\mathcal{V}$ converges to a point in $\mathcal{V}$

Contractions

Let $(\mathcal{V}, \|\cdot\|)$ be a normed vector space and and let $L : \mathcal{V} \to \mathcal{V}$. We say that L is a contraction, or a contraction mapping, if there is a real number $\gamma \in [0, 1)$, such that

$$\|L(v) - L(u)\| \leq \gamma \|v - u\|$$

for all v and u in $\mathcal{V}$, where the term γ is called a Lipschitz coefficient for L.



Notion of Fixed Point

Fixed Point

A vector $v \in \mathcal{V}$ is a fixed point of the map $L : \mathcal{V} \to \mathcal{V}$ if $L(v) = v$

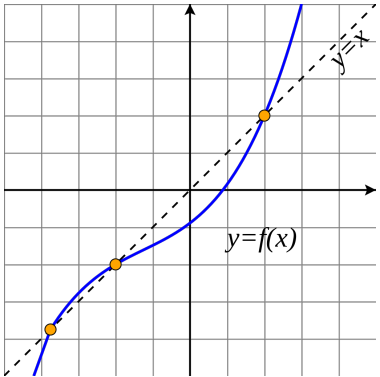
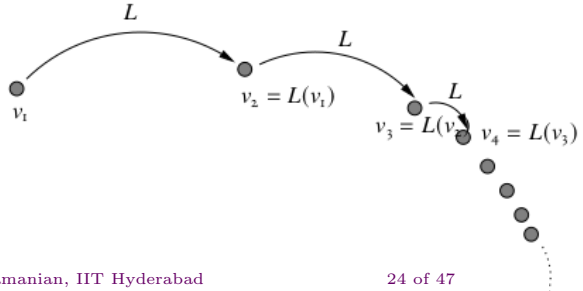


Figure: Fixed Point : Illustration

Banach Fixed Point Theorem

Theorem

Let $< \mathcal{V}, \|\cdot\| >$ be a complete normed vector space and let $L : \mathcal{V} \to \mathcal{V}$ be a γ-contraction mapping. Then iterative application of L converges to a unique fixed point in $\mathcal{V}$ independent of the starting point



Value Function Space

- $\mathcal{S}$ is a discrete state space with $|\mathcal{S}| = n$
- $\mathcal{A}_s \subseteq \mathcal{A}$ be the non-empty subset of actions allowed from state s
- $\mathcal{V}$ be a vector space of set of all bounded real valued functions from $\mathcal{S}$ to $\mathbb{R}$
- Measure the distance between state value functions $u, v \in \mathcal{V}$ using the max-norm defined as follows

$$\|u - v\| = \|u - v\|_\infty = \max_{s \in \mathcal{S}} |u(s) - v(s)| \quad s \in \mathcal{S}; u, v \in \mathcal{V}$$

  - ★ Largest distance between state values
- The space $\mathcal{V}$ is complete

Bellman Evaluation Operator

$$V_{k+1}^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s'} \mathcal{P}_{ss'}^{a} \left[\mathcal{R}_{ss'}^{a} + \gamma V_{k}^{\pi}(s') \right]$$

Denote,

$$\begin{aligned} \mathcal{P}^{\pi}(s'|s) &= \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}_{ss'}^{a} \\ \mathcal{R}^{\pi}(s) &= \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s'} \mathcal{P}_{ss'}^{a} \mathcal{R}_{ss'}^{a} = \mathbb{E}(r_{t+1}|s_t = s) \end{aligned}$$

Then, we can write,

$$V^{\pi} = \mathcal{R}^{\pi} + \gamma \mathcal{P}^{\pi} V^{\pi} \quad \text{(or) } V_{k+1} = \mathcal{R}^{\pi} + \gamma \mathcal{P}^{\pi} V_{k}$$

Define **Bellman Evaluation Operator** ($\mathcal{L}^{\pi} : \mathcal{V} \rightarrow \mathcal{V}$) as,

$$L^{\pi}(v) = \mathcal{R}^{\pi} + \gamma \mathcal{P}^{\pi} v$$

Bellman Optimality Operator

$$V_{k+1}(s) = \max_a \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V_k(s') \right) \right]$$

Denote,

$$\mathcal{R}^a(s) \quad = \quad \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \mathcal{R}^a_{ss'}$$

Then, we can write,

$$V_{k+1} = \max_{a \in \mathcal{A}} \left[\mathcal{R}^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} V_k \right]$$

Definte **Bellman Optimality Operator** : $(\mathcal{L} : \mathcal{V} \to \mathcal{V})$ as

$$L(v) = \max_{a \in \mathcal{A}} \left[\mathcal{R}^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} v \right]$$

Remark : Note that since value functions are a mapping from state space to real numbers one can also think of $\mathcal{L}^\pi$ and $\mathcal{L}$ as mappings from $\mathbb{R}^d \to \mathbb{R}^d$

Fixed Points of Maps $\mathcal{L}^\pi$ and $\mathcal{L}$

We can see that V^π is a fixed point of function $\mathcal{L}^\pi$

$$\mathcal{L}^\pi V^\pi = V^\pi$$

and V_* is a fixed point of operator $\mathcal{L}$

$$\mathcal{L} V_* = V_*$$

Bellman Evaluation Operator is a Contraction

Recall that Bellman evaluation operator is given by $L^\pi : \mathcal{V} \to \mathcal{V}$

$$L^\pi(v) = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v$$

- This operator is γ contraction. i.e., it makes value fuctions closer by at least γ.

Proof.

For any two value functions u and v in the space $\mathcal{V}$, we have,

$$\begin{aligned} \|L^\pi(u) - L^\pi(v)\|_\infty &= \|(\mathcal{R}^\pi + \gamma \mathcal{P}^\pi u) - (\mathcal{R}^\pi + \gamma \mathcal{P}^\pi v)\|_\infty \\ &= \|\gamma \mathcal{P}^\pi (u - v)\|_\infty (\leq \gamma \|P^\pi\|_\infty \|(u - v)\|_\infty = \gamma \|(u - v)\|_\infty) \\ &\leq \|\gamma \mathcal{P}^\pi \|u - v\|_\infty\|_\infty \\ &\leq \gamma \|u - v\|_\infty \end{aligned}$$

(We used for every $x \in \mathbb{R}^n$, and A is a $m \times n$ matrix, $\|Ax\|_\infty \leq \|A\|_\infty \|x\|_\infty$) □

Convergence of Bellman Updates

- Banach fixed-point theorem guarantees that iteratively applying evaluation operator $\mathcal{L}^\pi$ to any function $V \in \mathcal{V}$ will converge to a unique function $V^\pi \in V$
- Similarly, the Bellman optimality operator ($\mathcal{L} : \mathcal{V} \to \mathcal{V}$)

$$L(v) = \max_{a \in \mathcal{A}} [\mathcal{R}^a + \gamma \mathcal{P}^a v] \quad (\text{ A similar argument as } L^\pi)$$

  is also a γ contraction and hence iteratively applying optimality operator $\mathcal{L}$ to any funciton $V \in \mathcal{V}$ will converge to a unique function $V_* \in V$
- Does $V_* = \max_\pi V^\pi(\cdot)$? (Yes, it does)

Policy Iteration

Policy Iteration

Question : Is there a way to arrive at π_* starting from an arbitrary policy π ?

Answer : Policy Iteration

- Evaluate the policy π
  - ★ Compute $V^\pi(s) = \mathbb{E}_\pi(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s)$
- Improve the policy π

$$\pi'(s) = \text{greedy}(V^\pi(s))$$

$$\pi_0 \xrightarrow{\text{E}} V^{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} V^{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi^* \xrightarrow{\text{E}} V^*,$$

Policy Evaluation

- **Problem** : Evaluate a given policy π
- Compute $V^{\pi}(s) = \mathbb{E}_{\pi}(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s)$
- **Solution 1** : Solve a system of linear equations using any solver

- **Solution 2** : Iterative application of Bellman Evaluation Equation
- Iterative update rule :

$$V^{\pi}_{k+1}(s) \leftarrow \sum_{a} \pi(a|s) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}_{k}(s')\right]$$

- The sequence of value functions $\{V^{\pi}_1, V^{\pi}_2, \cdots, \}$ converge to V^{π}

Policy Improvement

Suppose we know V^π. How to improve policy π ?

The answer lies in the definition of action value function $Q^\pi(s,a)$. Recall that,

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_\pi\left(\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t=s,a_t=a\right) \\ &= \mathbb{E}(r_{t+1}+\gamma V^\pi(s_{t+1})|s_t=s,a_t=a) \\ &= \sum_{s'\in\mathcal{S}}\mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'}+\gamma V^\pi(s')\right] \end{aligned}$$

- If $Q^\pi(s,a) > V^\pi(s) \implies$ Better to select action a in state s and thereafter follow the policy π
- This is a special case of the policy improvement theorem

Policy Improvement Theorem

Theorem

Let π and π' be any pair of deterministic policies such that, for all $s \in \mathcal{S}$,

$$Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s).$$

Then $V^{\pi'}(s) \geq V^{\pi}(s)$ for all $s \in \mathcal{S}$

Proof.

$$\begin{aligned}
V^{\pi}(s) &\leq Q^{\pi}(s, \pi'(s)) = \mathbb{E}_{\pi'}(r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s) \\
&\leq \mathbb{E}_{\pi'}(r_{t+1} + \gamma Q^{\pi}(s_{t+1}, \pi'(s_{t+1})) | s_t = s) \\
&= \mathbb{E}_{\pi'}(r_{t+1} + \gamma r_{t+2} + \gamma^2 V^{\pi}(s_{t+2}) | s_t = s) \\
&\leq \mathbb{E}_{\pi'}(r_{t+1} + \gamma r_{t+2} + \gamma^2 Q^{\pi}(s_{t+2}, \pi'(s_{t+2})) | s_t = s) \\
&\leq \mathbb{E}_{\pi'}(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s) = V^{\pi'}(s)
\end{aligned}$$

□

Policy Improvement

- Now consider the greedy policy $\pi' = \text{greedy}(V^\pi)$.
- Then, $\pi' \geq \pi$. That is, $V^{\pi'}(s) \geq V^\pi(s)$ for all $s \in \mathcal{S}$.
  - ★ By defintion of π', at state s, the action chosen by policy π' is given by the greedy operator
  $$\pi'(s) = \arg\max_a Q^\pi(s, a)$$
  - ★ This improves the value from any state s over one step
  $$Q^\pi(s, \pi'(s)) = \max_a Q^\pi(s, a) \geq Q^\pi(s, \pi(s)) = V^\pi(s)$$
  - ★ It therefore improves the value function, $V^{\pi'}(s) \geq V^\pi(s)$
- Policy π' is at least as good as policy π

Figure Source: Refer to David Silver Lecture 3 slides for a more detailed proof

Policy Improvement

- If improvements stop,

$$Q^{\pi}(s, \pi'(s)) = \max_{a} Q^{\pi}(s, a) = Q^{\pi}(s, \pi(s)) = V^{\pi}(s)$$

- Bellman optimality equation is satisfied as,

$$V^{\pi}(s) = \max_{a} Q^{\pi}(s, a)$$

- The policy π for which the improvement stops is the optimal policy.

$$V^{\pi}(s) = V_{*}(s) \quad \forall s \in \mathcal{S}$$

Policy Iteration : Algorithm

Algorithm Policy Iteration

1: Start with an initial policy π_1
2: **for** $i = 1, 2, \cdots, N$ **do**
3: Evaluate $V^{\pi_i}(s) \quad \forall s \in \mathcal{S}$. That is,
4: **for** $k = 1, 2, \cdots, K$ **do**
5: For all $s \in \mathcal{S}$ calculate

$$V_{k+1}^{\pi_i}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} \mathcal{P}_{ss'}^a \left[\mathcal{R}_{ss'}^a + \gamma V_k^{\pi_i}(s')\right]$$

6: **end for**
7: Perform policy Improvement

$$\pi_{i+1} = \text{greedy}(V^{\pi_i})$$

8: **end for**

Policy Iteration : Example

Update Rule :

$$V_{k+1}^{\pi_i}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} \mathcal{P}_{ss'}^a \left[\mathcal{R}_{ss'}^a + \gamma V_k^{\pi_i}(s')\right]$$

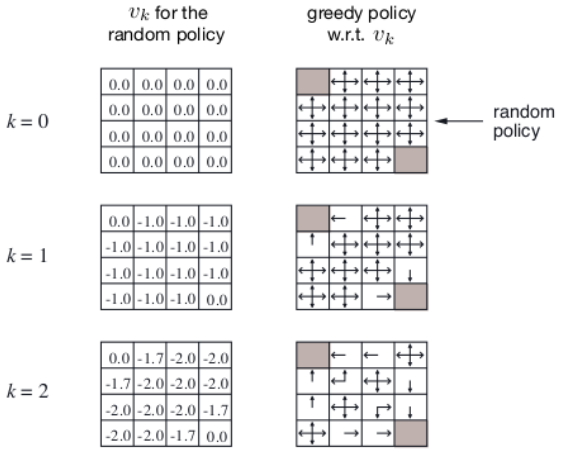


Figure Source: David Silver's UCL course

Policy Iteration : Example

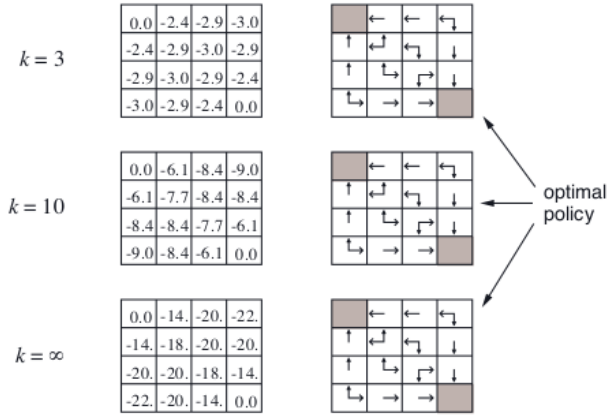
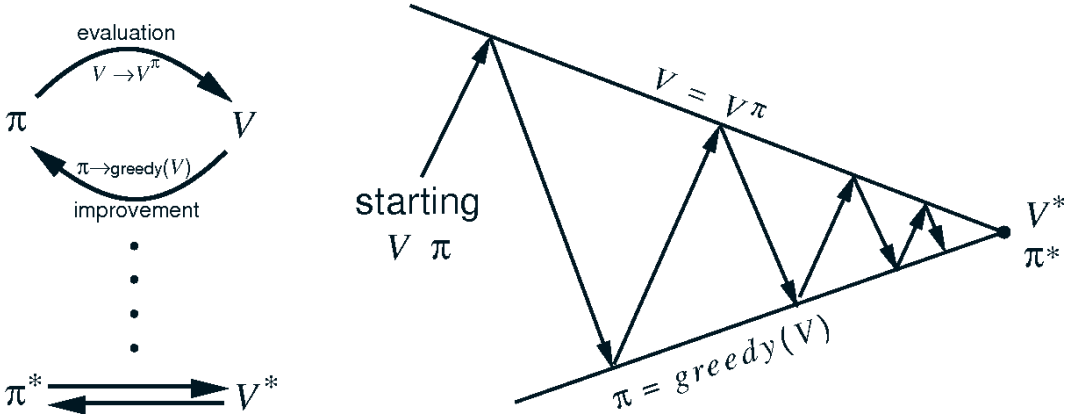


Figure Source: David Silver's UCL course

Policy Iteration : Schematic Representation



- The sequence $\{\pi_1, \pi_2, \cdots, \}$ is guaranteed to converge.
- At convergence, both current policy and the value function associated with the policy are optimal.

Figure Source: David Silver's UCL course

Modified Policy Iteration

Can we computationally simplify policy iteration process ?

- We need not wait for policy evaluation to converge to V^π
- We can have a stopping criterion like ϵ-convergence of value function evaluation or K iterations of policy evaluation
- Extreme case of $K = 1$ is **value iteration**. We update the policy every iteration

Possible Extensions and Remarks

Asynchronous Dynamic Programming

- Updates to states are done individually, in any order
- For each selected state, apply the appropriate backup
- Can significantly reduce computation
- Convergence guarantees exist, if all states are selected sufficient number of times

Real Time Dynamic Programming

- Idea : update only states that are relevant to agent
- After each time step, we get s_t, a_t, r_{t+1}
- Perform the following update

$$V(s_t) \leftarrow \max_a \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{s_t s'} \left(\mathcal{R}^a_{s_t s'} + \gamma V(s') \right) \right]$$

MDP and RL setting

- **MDP Setting** : The agent has knowledge of the state transition matrices $\mathcal{P}^a_{ss'}$ and the reward function $\mathcal{R}$
- **RL Setting** : The agent does not have knowledge of the state transition matrices $\mathcal{P}^a_{ss'}$ and the reward function $\mathcal{R}$
  - ★ The goal in both cases are same; Determine optimal sequence of actions such that the total discounted future reward is maximum.
  - ★ Although, this course would assume Markovian structure to state transitions, in many (sequential) decision making problems we may have to consider the history as well.

Prediction and Control using Dynamic Programming

- Dynamic Programming assumes full knowledge of MDP
- Used for both **prediction** and **control** in an MDP
- Prediction
  - Input MDP $(< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >)$ and policy π
  - Output : $V^{\pi}(\cdot)$
- Control
  - Input MDP $(< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >)$
  - Output : Optimal value function $V_*(\cdot)$ or optimal policy π_*

Model Free Prediction

Easwar Subramanian

TCS Innovation Labs, Hyderabad

Email : cs5500.2020@iith.ac.in

August 31, 2024

Overview

TCS Research & Innovation

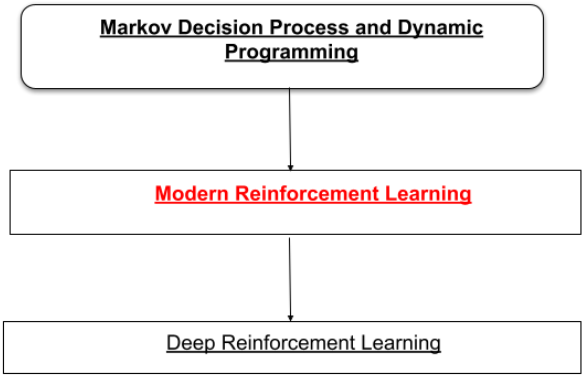
MDP and RL setting

- **MDP Setting** : The agent has knowledge of the state transition matrices $\mathcal{P}^a_{ss'}$ and the reward function $\mathcal{R}$
- **RL Setting** : The agent does not have knowledge of the state transition matrices $\mathcal{P}^a_{ss'}$ and the reward function $\mathcal{R}$

Prediction and Control using Dynamic Programming

- Dynamic Programming assumes full knowledge of MDP
- Used for both **prediction** and **control** in an MDP
- Prediction
  - Input MDP $(< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >)$ and policy π
  - Output : $V^{\pi}(\cdot)$
- Control
  - Input MDP $(< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >)$
  - Output : Optimal value function $V_*(\cdot)$ or optimal policy π_*

Course Setup



DP Algorithms : A Closer Look

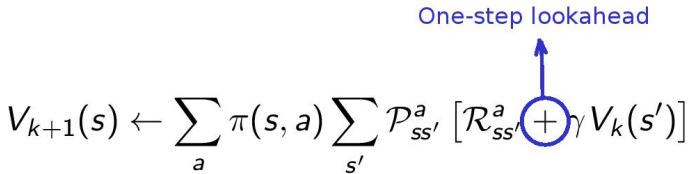
DP Algorithms : Terminology

Let us consider the policy evaluation formula

$$V_{k+1}(s) \leftarrow \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V_k(s') \right]$$

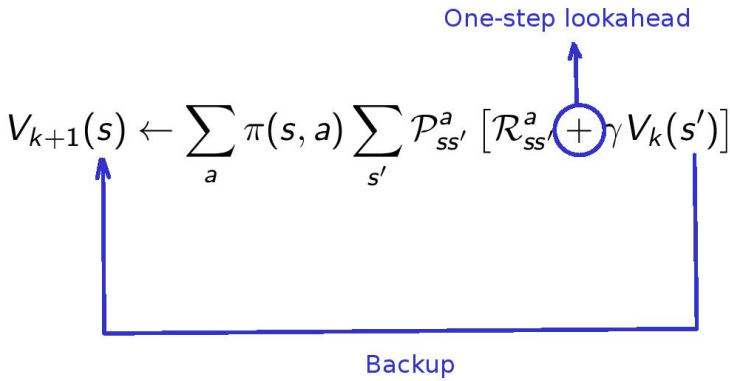
DP Algorithms : Terminology

Let us consider the policy evaluation formula



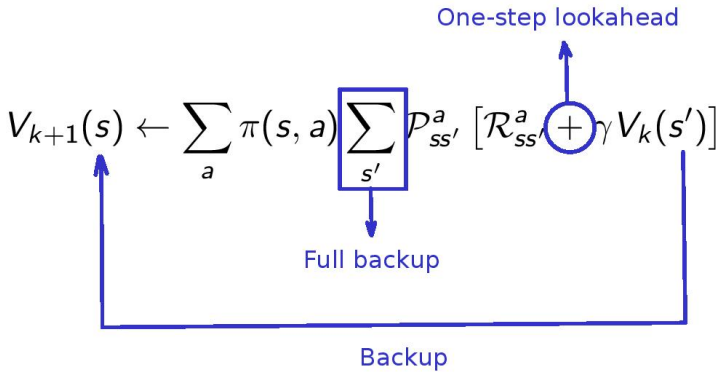
DP Algorithms : Terminology

Let us consider the policy evaluation formula



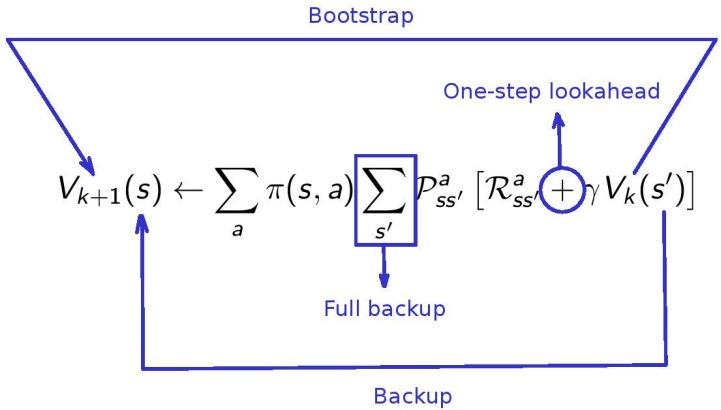
DP Algorithms : Terminology

Let us consider the policy evaluation formula

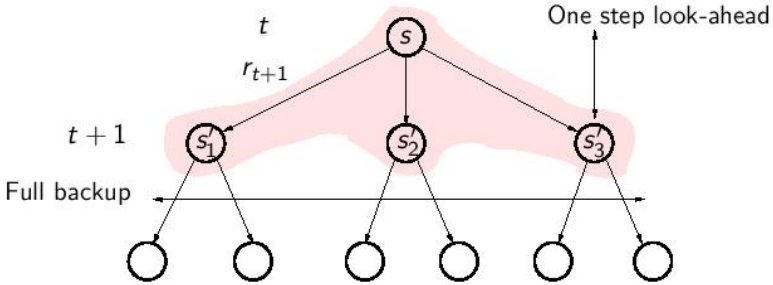


DP Algorithms : Terminology

Let us consider the policy evaluation formula



DP Algorithms : Schematic View



$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s') \right]$$

$$V_{k+1}(s) \leftarrow \sum_{a} \pi(a|s) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V_{k}(s') \right]$$

Drawbacks of DP Algorithms

- Requires full prior knowledge of the dynamics of the environment
- Can be implemented only on small or medium sized discrete state spaces
  - ★ For large problems, DP suffers from Bellman's curse of dimensionality
- DP uses full width back-ups
  - ★ Every successor state and action is considered

Model Free Prediction: Monte Carlo Methods

Model Free Prediction : Key Idea

$$
\begin{aligned}
V^{\pi}(s) &\overset{\text{def}}{=} \mathbb{E}_{\pi}(G_t|s_t = s) = \mathbb{E}_{\pi}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s\right) \\
&= \mathbb{E}_{\pi}\left[r_{t+1} + \gamma V^{\pi}(s_{t+1})|s_t = s\right]
\end{aligned}
$$

How can we estimate the expectations?
Use samples!

Monte Carlo Policy Evaluation

- Goal : Evaluate $V^{\pi}(s)$ using experiences (or trajectories) under policy π

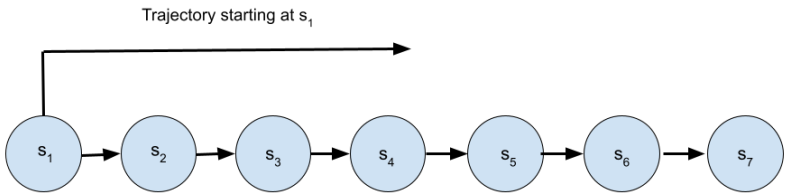
$$s_0, a_0, r_1, s_1, a_1, r_2, s_3, \cdots, s_T$$

- Recall that

$$V^{\pi}(s) = \mathbb{E}_{\pi}(G_t | s_t = s) = \mathbb{E}_{\pi}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right)$$

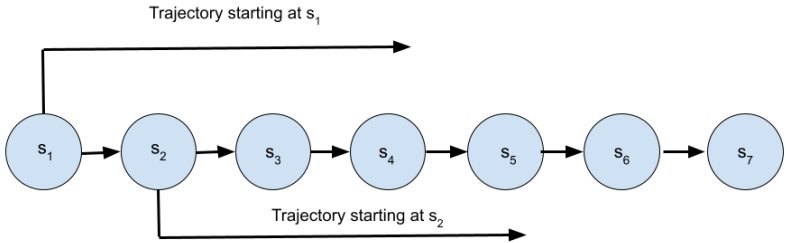
- The idea is to calculate **sample** mean return (G_t) starting from state s instead of expected mean return

Monte Carlo Evalution : Schematics



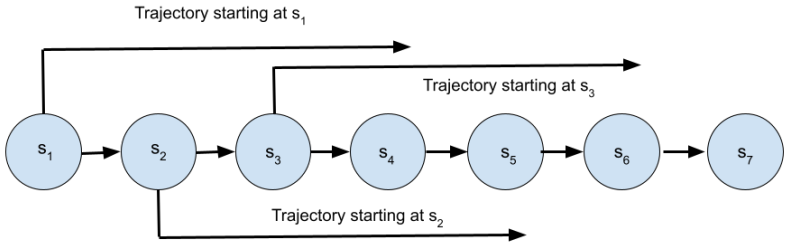
- Use G_1 to update $V^{\pi}(s_1)$

Monte Carlo Evalution : Schematics



- Use G_1 to update $V^\pi(s_1)$
- Use G_2 to update $V^\pi(s_2)$

Monte Carlo Evalution : Schematics



- Use G_1 to update $V^\pi(s_1)$
- Use G_2 to update $V^\pi(s_2)$
- Use G_3 to update $V^\pi(s_3)$

First-visit Monte Carlo Policy Evaluation

- To evaluate $V^\pi(s)$ for some given state s, repeat over several episodes
  - ★ The first time t that $s_t = s$ in the episode
    1. Increment counter for number of visits to s: $N(s) \leftarrow N(s) + 1$
    2. Increment running sum of total returns with return from current episode: $S(s) \leftarrow S(s) + G_t$
- Monte Carlo estimate of value function $V(s) \leftarrow S(s)/N(s)$

By the law of large numbers $V(s) \rightarrow V^\pi(s)$ as number of episodes increases

Monte Carlo Method : Example

- Consider an MDP with two states $\mathcal{S} = \{A, B\}$ with $\gamma = 1$
- $\mathcal{P}$ and $\mathcal{R}$ are unknown
- Consider a policy π that gives rise to following state-reward sequence
  - ★ $A(+3), A(+2), B(-4), A(+4), B(-3)$
  - ★ $B(-2), A(+3), B(-3)$
- What is $V^{\pi}(A)$ and $V^{\pi}(B)$ if we use first visit MC ?
- First visit MC : $V(A) = \frac{1}{2}(2+0) = 1$; $V(B) = \frac{1}{2}(-3-2) = -5/2$

Every-visit Monte Carlo Policy Evaluation

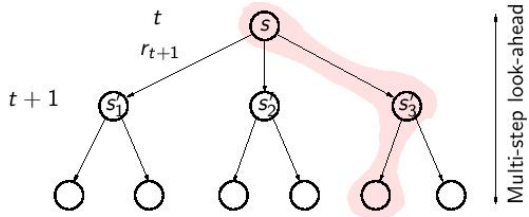
- To evaluate $V^\pi(s)$ for some given state s, repeat over several episodes
  - ★ Every time t that $s_t = s$ in the episode
    1. Increment counter for number of visits to s: $N(s) \leftarrow N(s) + 1$
    2. Increment running sum of total returns with return from current episode: $S(s) \leftarrow S(s) + G_t$
- Monte Carlo estimate of value function $V(s) \leftarrow S(s)/N(s)$

By the law of large numbers $V(s) \rightarrow V^\pi(s)$ as number of episodes increases

Monte Carlo Method : Example

- Consider an MDP with two states $\mathcal{S} = \{A, B\}$ with $\gamma = 1$
- $\mathcal{P}$ and $\mathcal{R}$ are unknown
- Consider a policy π that gives rise to following state-reward sequence
  - ★ $A(+3), A(+2), B(-4), A(+4), B(-3)$
  - ★ $B(-2), A(+3), B(-3)$
- What is $V^\pi(A)$ and $V^\pi(B)$ if we use first visit MC and every visit MC respectively ?
- First visit MC : $V(A) = \frac{1}{2}(2+0) = 1$; $V(B) = \frac{1}{2}(-3-2) = -5/2$
- Every visit MC : $V(A) = \frac{1}{4}(2-1+1+0) = 1/2$; $V(B) = \frac{1}{4}(-3-3-3-2) = -11/4$

Monte Carlo Algorithms: A Schematic View



- Uses experience, rather than model
- Uses only experience; does not bootstrap
- Needs complete sequences; suitable only for episodic tasks
- Suited for off-line learning
- Time required for one estimate does not depend on total number of states
- Estimates for each state are independent

Model Free Prediction : Temporal Difference

Temporal Difference : Key Idea

$$V^{\pi}(s) \stackrel{\text{def}}{=} \mathbb{E}_{\pi}(G_t|s_t = s) = \mathbb{E}_{\pi}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s\right)$$
$$= \mathbb{E}_{\pi}\left[r_{t+1} + \gamma V^{\pi}(s_{t+1})|s_t = s\right]$$

- Estimate expectation from experience using the recursive decomposition formulation of the value function

Incremental Calculation of Mean

$$
\begin{aligned}
\mu_{k+1} &\stackrel{\text{def}}{=} \frac{1}{k+1}\sum_{i=1}^{k+1} x_i \\
&= \frac{1}{k+1}\sum_{i=1}^{k} x_i + \frac{1}{k+1}x_{k+1} \\
&= \frac{k}{k+1}\left(\frac{1}{k}\sum_{i=1}^{k} x_i\right) + \frac{1}{k+1}x_{k+1} \\
&= \frac{k}{k+1}\mu_k + \frac{1}{k+1}x_{k+1} \\
&= \mu_k + \frac{1}{k+1}(x_{k+1} - \mu_k)
\end{aligned}
$$

$$\text{Update} = \text{learning rate} \times (\text{Target} - \text{Previous Value})$$

General Form of Update Rule

The general form for the update rule that is present in the incremental calculation is,

$$\text{New Estimate} \leftarrow \text{Old Estimate} + \text{Learning Rate}(\text{Target - Old Estimate})$$

- The expression (Target - Old Estimate) is an error of the estimate
- The error is reduced by taking steps towards the "Target"
- The target is persumed to indicate a desirable direction to move
- In the incremental calculation of mean, the term x_{k+1} is the target

One-Step TD

- We wish to approximate

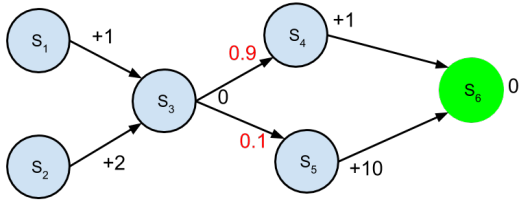
$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s\right]$$

- Approximate the expectation by a sample mean
  - ★ If the *transition* (s_t, r_{t+1}, s_{t+1}) is observed at time t under π, then

$$V(s_t) \leftarrow V(s_t) \; + \; \alpha_t[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

  - ★ Samples come from different visits to the state s, either from same or different trajectories
  - ★ Compute the sample mean incrementally

TD vs MC : Example



(1) $s_1 \xrightarrow{1} s_3 \xrightarrow{0} s_4 \xrightarrow{1} s_6$

(2) $s_1 \xrightarrow{1} s_3 \xrightarrow{0} s_5 \xrightarrow{10} s_6$

(3) $s_1 \xrightarrow{1} s_3 \xrightarrow{0} s_4 \xrightarrow{1} s_6$

(4) $s_1 \xrightarrow{1} s_3 \xrightarrow{0} s_4 \xrightarrow{1} s_6$

(5) $s_2 \xrightarrow{2} s_3 \xrightarrow{0} s_5 \xrightarrow{10} s_6$

TD vs MC : Example

- True value of each state is given by
  $V(s_6) = 0, V(s_5) = 10, V(s_4) = 1, V(s_3) = 1.9, V(s_2) = 3.9$ and $V(s_1) = 2.9$
- Evaluate $V(s_1)$ and $V(s_2)$ using MC $V(s_1) = 4.25$ and $V(s_2) = 12$
- Evaluate $V(s_1)$ and $V(s_2)$ using TD
  - ★ First trajectory $(s_1 \xrightarrow{1} s_3 \xrightarrow{0} s_4 \xrightarrow{1} s_6)$
    $V(s_1) = 1$; $V(s_3) = 0$; $V(s_4) = 1$; $V(s_6) = 0$
  - ★ Second trajectory $(s_1 \xrightarrow{1} s_3 \xrightarrow{0} s_5 \xrightarrow{10} s_6)$
    $V(s_1) = 1$; $V(s_3) = 0$; $V(s_5) = 10$; $V(s_6) = 0$
  - ★ Third trajectory $(s_1 \xrightarrow{1} s_3 \xrightarrow{0} s_4 \xrightarrow{1} s_6)$
    $V(s_1) = 1$; $V(s_3) = 0.33$; $V(s_4) = 1$; $V(s_6) = 0$
  - ★ Fourth trajectory $(s_1 \xrightarrow{1} s_3 \xrightarrow{0} s_4 \xrightarrow{1} s_6)$
    $V(s_1) = 1.08$; $V(s_3) = 0.5$; $V(s_4) = 1$; $V(s_6) = 0$
  - ★ Fifth trajectory $(s_2 \xrightarrow{2} s_3 \xrightarrow{0} s_5 \xrightarrow{10} s_6)$
    $V(s_2) = 2.5$; $V(s_3) = 2.4$; $V(s_5) = 10$; $V(s_6) = 0$

One-Step TD : TD(0) Algorithm

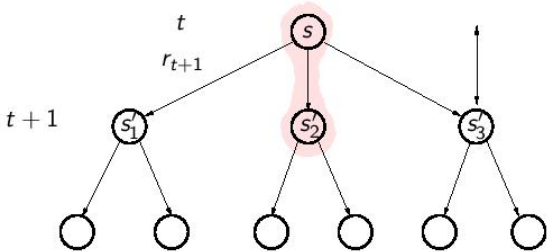
Algorithm TD(0) : Algorithm

Initialize $V(s)$ arbitrarily (say, $V(s) = 0 \quad \forall s \in \mathcal{S}$);
for $k = 1, 2, \cdots, K$ **do**
    Let s be a start state for episode k
    for For each step in the k-th episode **do**
        Take action a recommended by policy π from state s
        Collect reward r and reach next state s'
        Perform the following TD update

$$V(s) = V(s) + \alpha[r + \gamma V(s') - V(s)]$$

        Assign $s \leftarrow s'$
    end for
end for

TD Algorithms: A Schematic View



- Uses experience without model like MC
- Bootstraps like DP
- Can work with partial sequences
- Suited for online learning

Schematic View of Various Algorithms

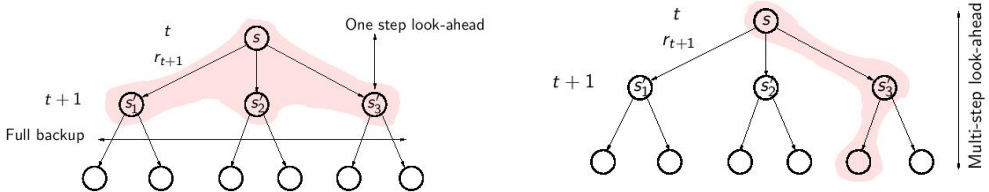
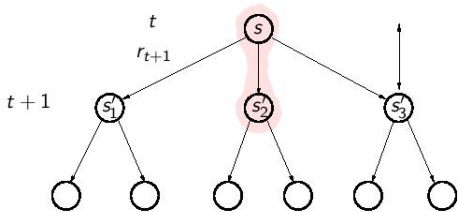


Figure: DP Algorithm and MC Algorithm



Convergence of TD Algorithms

- For any fixed policy π, the TD(0) algorithm described above converges (asymptotically) to V^π under some conditions on the choice of α (Robbins Monroe Condition)
  - ★ $\sum \alpha_t = \infty$
  - ★ $\sum \alpha_t^2 < \infty$

- *Generally*, TD methods have usually been found to converge faster than MC methods on certain class of tasks

Connections between MC Error and TD Error

- The term $\delta_t = [r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$ is called the (one step) **TD error**
- The term $G_t - V(s_t)$ is called the **MC error**
- If a trajectory has T time steps then

$$G_t - V(s_t) = \sum_{k=0}^{T-t-1} \gamma^k \delta_{t+k}$$

(Exercise : Prove it !)

Bias and Variance in Prediction Methods

Bias in MC Algorithms

- Consider a statistical model represented by parameter θ giving rise to distribution of observed data $P(x|\theta)$
- Let $\hat{\theta}$ be an estimator of θ, that depends on observations from $P(x|\theta)$
- Bias of the estimator $\hat{\theta}$ (with true value θ) is given by $\mathbb{E}_{x|\theta}[\hat{\theta}] - \theta$
- Since

$$V^{\pi}(s) \stackrel{\text{def}}{=} \mathbb{E}_{\pi}(G_t|s_t = s) = \mathbb{E}_{\pi}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s\right),$$

  the return G_t is an unbiased estimator of $V^{\pi}(s)$
- Since each total discounted return is sampled from independent trajectory, they are i.i.d., and hence the FVMC estimate of $V^{\pi}(s)$ has no bias

Variance in MC Algorithms

- Consider a sequence of estimates $V_k(s)$ for a particular state s computed as

$$V_k(s) = \frac{1}{k} \sum_{i=1}^{k} G_i$$

- We then have, $\text{Var}(V_k(s)) \propto \frac{1}{k}$ since,

$$\begin{aligned} \text{Var}(V_k(s)) &= \text{Var}\left(\frac{1}{k}\sum_{i=1}^{k} G_i\right) = \frac{1}{k^2}\text{Var}\left(\sum_{i=1}^{k} G_i\right) = \frac{1}{k^2} k \text{Var}\left(G\right) \\ &= \frac{1}{k}\text{Var}\left(G\right) \end{aligned} \tag{1}$$

- Hence MC methods tend to have high variance as returns are a function of multi-step sequence of random actions, states and rewards

Monte Carlo Methods : Bias and Variance

- Both first visit MC and every visit MC converge to V^π as number of trajectories go to infinity
- In first visit MC this is easy to see as each return sample is independent of the another
- By the law of large numbers the sequence of averages of these estimates converges to their expected value
- FVMC average is itself an unbiased estimate, and the standard deviation of its error falls as $\sqrt{1/k}$ where k is the number of returns averaged
- The convergence of every visit MC is less straight forward to see but it also converges at a quadratic rate to V^π

In both MC methods, it is possible that we may leave out computing $V^\pi(s)$ for some $s \in \mathcal{S}$ because the state s was never visited by any of the trajectories

Bias in TD Algorithms

- In TD methods, value estimates are computed using,

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s\right]$$

  with bootstrapping being used
- Bias exists in TD target $r_{t+1} + \gamma V^{\pi}(s_{t+1})$ as $V^{\pi}(s_{t+1})$ is an **estimate** of value function and not the true value function
- There is a lot of bias in the beginning of training where the estimate of V^{π} is far from true V^{π} and the bias reduces with training

Bias/Variance Trade-Off in TD Algorithms

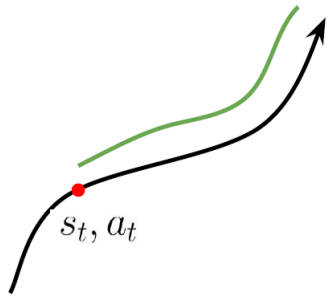
Monte Carlo Algorithms

- No Bias
  - ★ Sample average is an unbiased estimate of the expectation
- High Variance
  - ★ Returns are a function of multi-step sequence of random actions, states and rewards

Temporal Difference Algorithms

- Some Bias
  - ★ TD target $r_{t+1} + \gamma V(s_{t+1})$ is a biased estimate of $V(s)$
- Low Variance
  - ★ TD target only has one random action, reward and next state

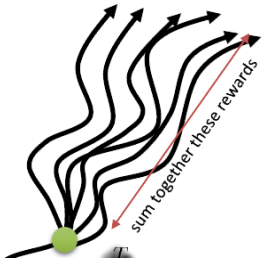
Bias and Variance of MC and TD Estimators : Schematics

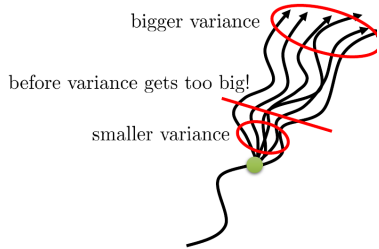


The rewards are counted along the green curve and hence the green curve represents the summation in the defintion of V^{π}

Figure Source:
Jie-Han-Chen:SlideShare

Bias and Variance of MC and TD Estimators : Schematics





- In the MC method we need to wait till the end of the trajectory to make an update; It means lots of future rewards need to be summed. This makes the estimate have high variance
- In the TD, we cut off the path only until next state (in one step TD) and hence the estimate has low variance. Because of bootstrapping, the TD estimate has high bias

Figure Source: UCB : Sergey Levine

Properties of Different Policy Evaluation Algorithms

	DP Algorithms	MC Algorithms	TD Algorithms
Model Free	No	Yes	Yes
Non Episodic Domains	Yes	No	Yes
Non Markovian Domains	No	Yes	No
Bias	Not Applicable	Unbiased	Some Bias
Variance	Not Applicable	High Variance	Low Variance

Multi-Step Temporal Difference

Multi-step TD

- One-step TD

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_\pi \left[r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s \right] \\ V(s_t) &\leftarrow V(s_t) + \alpha_t [r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \end{aligned}$$

- Two-step TD

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_\pi \left[r_{t+1} + \gamma r_{t+2} + \gamma^2 V^\pi(s_{t+2}) | s_t = s \right] \\ V(s_t) &\leftarrow V(s_t) + \alpha_t [r_{t+1} + \gamma r_{t+2} + \gamma^2 V(s_{t+2}) - V(s_t)] \end{aligned}$$

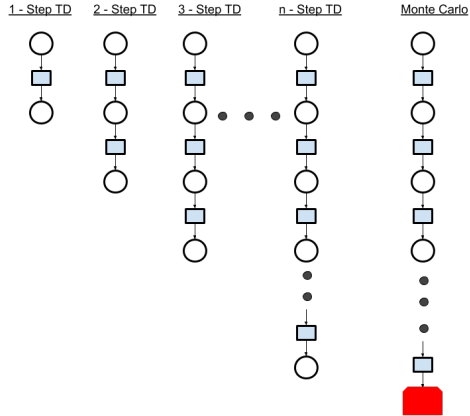
- More generally, define the n-step return

$$G_t^{(n)} \stackrel{\text{def}}{=} r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n})$$

- n-step TD

$$V(s_t) \leftarrow V(s_t) + \alpha_t [G_t^{(n)} - V(s_t)]$$

Why Multi-Step TD ?



- Multi-step TD methods tend to have less bias compared to 1-step TD method

λ-Return

- What if we average some or all the n-step returns?

Example : Target could be

$$\frac{1}{2}G_t^{(1)} + \frac{1}{2}G_t^{(4)}$$

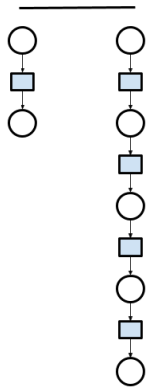


 Figure Source: Sutton and Barto

λ-Return

- Any set of returns can be averaged, even an infinite set, as long as the weights on the component returns are positive and sum to 1
- Choose $\lambda \in [0, 1]$, and define the λ-return at time t as

$$G_t^\lambda \stackrel{\text{def}}{=} (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$$

- λ-return algorithm: use λ-return as the target

$$V(s_t) \leftarrow V(s_t) + \alpha_t [G_t^\lambda - V(s_t)]$$

- If the episode ends at $T > t$, define $G_t^{(n)}$ to be G_t for all $n > T - t$. Then

$$G_t^\lambda = (1-\lambda) \sum_{n=1}^{T-t} \lambda^{n-1} G_t^{(n)} + \lambda^{T-t} G_t$$

- $\lambda = 0$ gives 1-step TD, $\lambda = 1$ gives MC update

λ-Return

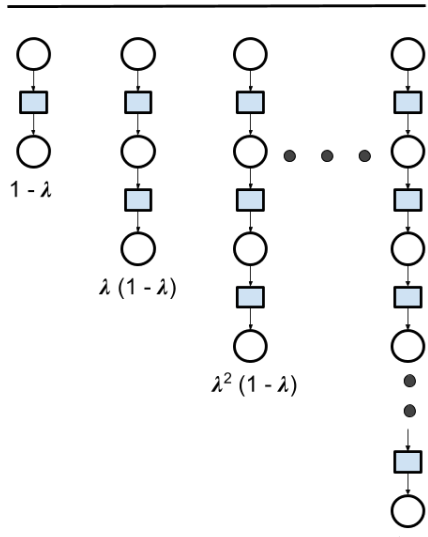
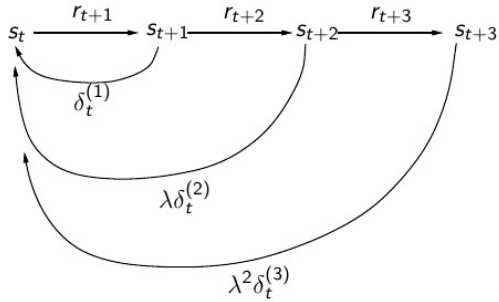


Figure Source: Sutton and Barto

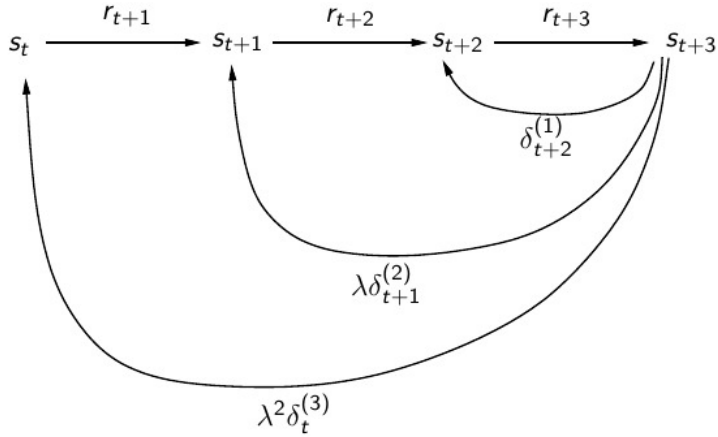
Forward View

$$G_t^\lambda - V(s_t) = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}[G_t^{(n)} - V(s_t)] = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}\delta_t^{(n)}$$



- Not suitable for online implementation;

A Possible Online Implementation



- Requires storing all rewards from the episode

A Rearrangement of n-step TD Errors

$$
\begin{aligned}
\delta_t^{(n)} &\stackrel{\text{def}}{=} r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n}) - V(s_t) \\
&= \gamma^0 [r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \\
&\quad + \gamma^1 [r_{t+2} + \gamma V(s_{t+2}) - V(s_{t+1})] \\
&\vdots \\
&\quad + \gamma^{n-1} [r_{t+n} + \gamma V(s_{t+n}) - V(s_{t+n-1})] \\
&= \sum_{i=t}^{t+n-1} \gamma^{i-t} \delta_i^{(1)}
\end{aligned}
$$

$$
\begin{aligned}
G_t^\lambda - V(s_t) &= (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} \delta_t^{(n)} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} \sum_{i=t}^{t+n-1} \gamma^{i-t} \delta_i^{(1)} \\
&= \sum_{i=0}^{\infty} (\lambda\gamma)^i \delta_{t+i}^{(1)}
\end{aligned}
$$

Online Implementation

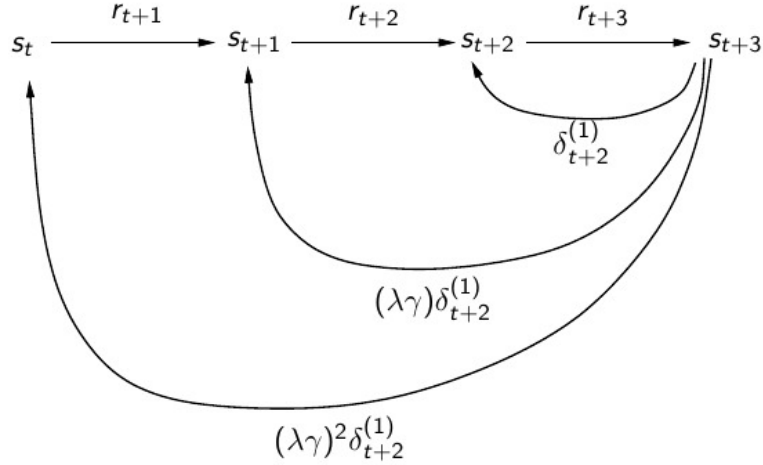
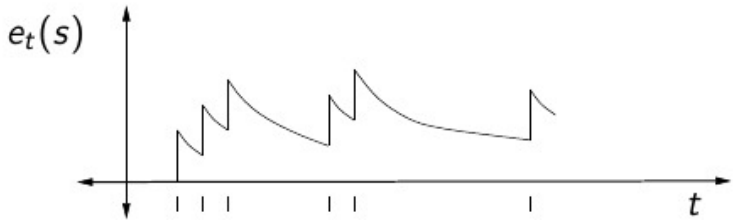


Figure: previously encountered states

Eligibility Traces

- The eligibility trace of a state $s \in \mathcal{S}$ at time t is defined recursively by

$$
\begin{aligned}
e_0(s) &= 0 \\
e_t(s) &= \begin{cases} (\lambda\gamma)e_{t-1}(s), & s_t \neq s \\ (\lambda\gamma)e_{t-1}(s) + 1, & s_t = s \end{cases}
\end{aligned}
$$



Algorithm : TD(λ)

Algorithm TD(λ) : Algorithm

Initialize $e(s) = 0$ for all s, $V(s)$ arbitrarily
for For each episode **do**
    Let s be a start state for episode k
    for For each step of the episode **do**
        Take action a recommended by policy π from state s
        Collect reward r and reach next state s'
        Form the one-step TD error $\delta \leftarrow r + \gamma V(s') - V(s)$
        Increment eligibility trace of state s, $e(s) \leftarrow e(s) + 1$
        for For all states $S \in \mathcal{S}$ **do**
            Update $V(S)$: $V(S) \leftarrow V(S) + \alpha e(S)\delta$
            Update eligibility trace: $e(S) \leftarrow \lambda\gamma e(S)$
        end for
        Move to next state: $s \leftarrow s'$
    end for
end for

Few More Concepts on Model Free Prediction

Certainity Equivalence Estimate

- Model based policy evaluation with model estimted from samples
- Given an experience quadruple (s, a, r, s'), we can estimate
  - ★ Compute MLE for model using (s, a, s')

$$\hat{P}(s'|s,a) = \frac{1}{N(s,a)} \sum_{k=1}^{K} \sum_{t=1}^{L_k - 1} \mathbb{1}(s_{k,t} = s, a_{k,t} = a, s_{k,t+1} = s')$$

  - ★ Compute MLE estimate of the reward function

$$\hat{R}(s,a,s') = \frac{1}{N(s,a,s')} \sum_{k=1}^{K} \sum_{t=1}^{L_k - 1} \mathbb{1}(s_{k,t} = s, a_{k,t} = a, s_{k,t+1} = s')\ r_{t,k}$$

- Once MLE estimates of $\hat{P}(s'|s,a)$ and $\hat{R}(s,a,s')$ use a known policy evaluation method to compute a MLE based estimate for V^{π}.

Off-Policy Learning

TCS Research & Innovation

- Can we estimate the value V^π or Q^π of a *target policy* π ...
- While following a *behavior policy* μ?
  - ★ Trajectories or transitions are sampled from μ
  - ★ Expected values have to be estimated w.r.t. π
- Possible benefits:
  - ★ Learn by observing other agents
  - ★ Re-use previous experience generated from earlier policies
  - ★ Learn about optimal policy while following exploratory policy
  - ★ Learn about multiple policies while following one policy

Importance Sampling : Review

Let $P(x)$ be the target distribution and $Q(x)$ be the behaviour distribution for some random variable x

$$\begin{aligned}\mathbb{E}_{X\sim P}[f(X)] &= \sum_i P(x_i)f(x_i) \\ &= \sum_i Q(x_i)\left[\frac{P(x_i)}{Q(x_i)}f(x_i)\right] \\ &= \mathbb{E}_{X\sim Q}\left[\frac{P(X)}{Q(X)}f(X)\right]\end{aligned}$$

- We have samples of X drawn from Q, but we wish to estimate expectation under P

$$\mathbb{E}_{X\sim P}[f(X)] \simeq \frac{1}{n}\sum_{i=1}^{n}\left[\frac{P(X_i)}{Q(X_i)}f(X_i)\right],\ X_i \sim Q$$

- Caveat: Q should not assign zero probability to any outcome that is assigned non-zero probability by P

Importance Sampling : Review

The ratio $\frac{P(x)}{Q(x)}$ is the importance sampling (IS) weight for x.

What about the variance of IS estimator $\widehat{\mu}_Q \approx \frac{1}{N}\sum_{x_i \in D}\left[\frac{P(x_i)}{Q(x_i)} f(x_i)\right]$?

$$\begin{aligned}
\text{var}_Q\left[\widehat{\mu}_Q\right] &= \text{var}_Q\left[\frac{P(x)}{Q(x)} f(x)\right] \\
&= \left[\mathbb{E}_Q\left[\left(\frac{P(x)}{Q(x)} f(x)\right)^2\right] - \mathbb{E}_Q\left(\left[\frac{P(x)}{Q(x)} f(x)\right]\right)^2\right] \\
&= \mathbb{E}_P\left[\left(\frac{P(x)}{Q(x)} f(x)^2\right)\right] - \mathbb{E}_P\left(f(x)\right)^2
\end{aligned}$$

If the likelihood ratio $\frac{P(x)}{Q(x)}$ is large, the variance of the estimator explodes

Off-Policy TD using Importance Sampling

- Evaluate target policy π using TD targets $r_{t+1} + \gamma V(s_{t+1})$ generated from μ
- Weigh each TD target by the importance sampling factor $\pi(s_t, a_t)/\mu(s_t, a_t)$

$$V(s_t) \leftarrow V(s_t) + \alpha_t \left[\frac{\pi(s_t|a_t)}{\mu(s_t|a_t)} (r_{t+1} + \gamma V(s_{t+1})) - V(s_t) \right]$$

- π may be deterministic and greedy, μ should be stochastic and exploratory
- The case $\mu = \pi$ is called *on-policy* learning
- On Policy Learning : Learn about policy π from experience sampled from π
- Off Policy Learning : Learn about policy π from experience sampled from μ

Model Free Control

Easwar Subramanian

TCS Innovation Labs, Hyderabad

Email : cs5500.2020@iith.ac.in

September 14, 2024

Overview

TCS Research & Innovation

Towards Model Free Control

Problem and Motivation

TCS Research & Innovation

- **Goal** : How can we learn a good policy?
- **Motivation** : Many real world applications can be modelled as MDP
  - ★ Games like Backgammon and Go
  - ★ Robot Locomotion
  - ★ Inventory or supply chain management
- For almost all these problems, model is unknown or computationally infeasible; but sampling experiences is possible
- Learning better policies through experiences is model free control

Towards Model Free Control

DP algorithms for control

- Value Iteration
- Policy Iteration

Question : Can we do model free control with value iteration ?

$$V_{k+1}(s) \leftarrow \max_a \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V_k(s') \right) \right]$$

- Value iteration may not come in handy because it requires knowledge of model; so not suitable

Question : How about policy iteration (PI) ? ?

- PI is a two step provess
  - ★ Policy evaluation
  - ★ Policy improvement

Policy Iteration : Recap

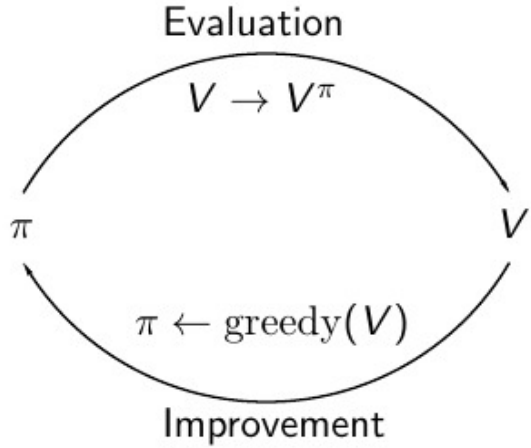


 Figure Source: Sutton and Barto

On Policy Improvement From Samples

- (Greedy) Policy improvement

$$\pi(s) \quad = \quad \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^\pi(s')\right]$$

- Generally, model free control is not done with V as greedy policy improvement over V requires the knowledge of the model
- (Greedy) policy improvement over Q is model free

$$\pi(s) = \arg\max_a Q^\pi(s, a)$$

- For model-free policy improvement, we use Q^π, not V^π

Core Idea behind Model Free Control

- Initialize a policy π
- Repeat
  - ★ Policy Evaluation : Find Q^{π}
  - ★ Policy Improvement : Get an improved policy from evaluation of Q^{π}

Monte Carlo Control

Policy Evaluation : Action Value Function

- We now need to evaluate Q^π instead of V^π
- Recall that the state-action value function of a policy π is given by,

$$\begin{aligned} Q^\pi(s,a) &\stackrel{\text{def}}{=} \mathbb{E}_\pi(G_t|s_t=s,a_t=a) \\ &= \mathbb{E}_\pi\left(\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t=s,a_t=a\right) \\ &= \mathbb{E}_\pi(r_{t+1}+\gamma Q^\pi(s_{t+1},a_{t+1})|s_t=s,a_t=a) \end{aligned}$$

- We can use MC or TD methods to evaluate Q^π using samples

First Visit Monte Carlo : Action Value Function

- To evaluate $Q^{\pi}(s,a)$ for some given state s and action a, repeat over several episodes
  - ★ The first time t that $s_t = s$ and $\pi(s) = a$ in the episode
    1. Increment counter for number of visits to s: $N(s,a) \leftarrow N(s,a) + 1$
    2. Increment running sum of total returns with return from current episode: $S(s,a) \leftarrow S(s,a) + G_t$
- Monte Carlo estimate of value function $Q(s,a) \leftarrow S(s,a)/N(s,a)$

The main drawback of this algorithm is

- Many state action pairs may never be visited
- If policy π is deterministic, things get even worse

Exploring Starts Assumption

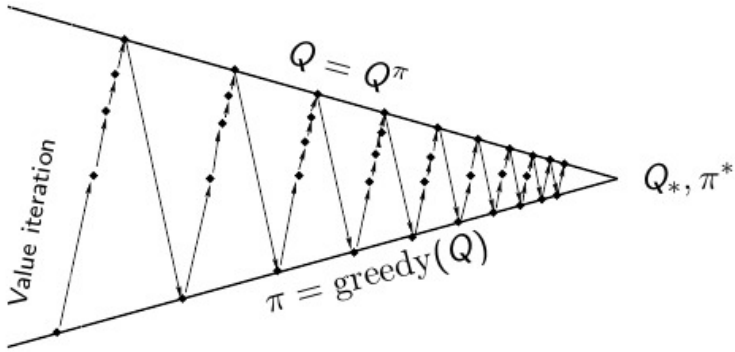
Exploring Starts (ES) Assumption

- First step of each episode start at a state-action pair, and that every such pair has non-zero probability of being selected at start
  - ★ Assumption guarantees that all state-action pairs will be visited an infinite number of times in the limit of an infinite number of episodes

Not a realistic assumption at all !! But let's assume it for a while

- With ES assumption, first or every visit MC algorithm will evaluate Q^{π}

Policy Iteration with Action Value Function



- Monte Carlo Policy Evaluation, $Q = Q^\pi$
- Greedy policy improvement , $\pi' = \arg\max_a Q^\pi(s, a)$

Algorithm Monte Carlo Control with ES

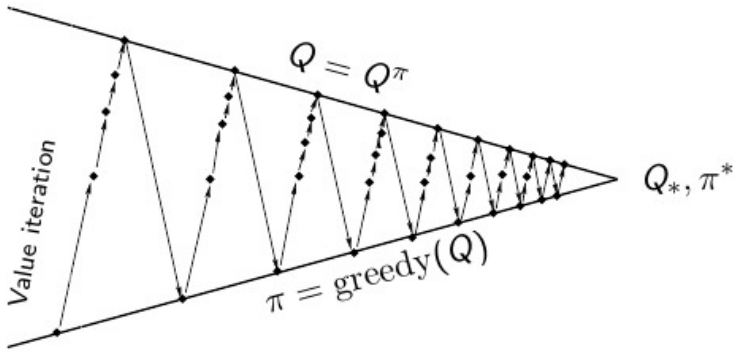
1: Start with an initial policy π_1 and initial estimate for Q^{π_1}
2: **for** $k = 1, 2, \cdots, K$ **do**
3: Generate an episode using ES and policy π_k
4: Policy Evaluation Step :
For each pair (s, a) **appearing in the episode evaluate** $Q^{\pi_k}(s, a)$ **using FVMC**
5: Policy Improvement Step :
For every $s \in \mathcal{S}$

$$\pi_{k+1}(s) = \arg\max_a Q^{\pi_k}(s, a)$$

6: **end for**

- Convergence of policy evaluation to Q^π is assured only under the ES assumption
- Once ES assumption is made, to understand convergence to Q_* and π_* one can use the same kind of arguments as we had in the policy iteration algorithm in the DP setting

Policy Iteration with Action Value Function



- Is it good to be always greedy ?
- Should we patiently wait until policy evaluation step converges ?

On Greedy Action Selection

- There are two doors in front of you
- You open the left door and get reward 0 i.e. $V(\text{left}) = 0$
- You open the right door and get reward 1 $V(\text{right}) = 1$
- You open the right door and get reward 3 $V(\text{right}) = 2$
- You open the right door and get reward 2 $V(\text{right}) = 3$
- Are we sure that right door is the best door ?

ε-Greedy Exploration

- Simplest idea for ensuring continual exploration
- All m actions are tried with non-zero probability every time
  - ★ With probability $1 - \varepsilon$, choose the greedy action
  - ★ With probability ε, choose an action uniformly at random

$$\begin{aligned} \pi(a|s) &= \frac{\varepsilon}{m} + 1 - \varepsilon, \text{ if } a = \arg\max_{a'} Q(s, a'), \\ &= \frac{\varepsilon}{m}, \text{ otherwise} \end{aligned}$$

ε-Greedy Policy Improvement

For any policy ε-greedy policy π, the ε-greedy policy π' w.r.t. Q^π is an improvement over π, that is, $V^{\pi'}(s) \geq V^\pi(s)$

$\varepsilon-$ Greedy Policy Improvement

$$
\begin{aligned}
Q^{\pi}(s,\pi'(s)) &= \sum_{a\in\mathcal{A}} \pi'(a|s) Q^{\pi}(s,a) \\
&= \frac{\varepsilon}{m} \sum_{a\in\mathcal{A}} Q^{\pi}(s,a) + (1-\varepsilon) \max_a Q^{\pi}(s,a) \\
&= \frac{\varepsilon}{m} \sum_{a\in\mathcal{A}} Q^{\pi}(s,a) + (1-\varepsilon) \frac{1-\varepsilon}{1-\varepsilon} \max_a Q^{\pi}(s,a) \\
&= \frac{\varepsilon}{m} \sum_{a\in\mathcal{A}} Q^{\pi}(s,a) + (1-\varepsilon) \sum_a \frac{\pi(a|s) - \frac{\varepsilon}{m}}{1-\varepsilon} \max_a Q^{\pi}(s,a) \\
&\geq \frac{\varepsilon}{m} \sum_{a\in\mathcal{A}} Q^{\pi}(s,a) + (1-\varepsilon) \sum_a \frac{\pi(a|s) - \frac{\varepsilon}{m}}{1-\varepsilon} Q^{\pi}(s,a) \\
&= \sum_{a\in\mathcal{A}} \pi(a|s) Q^{\pi}(s,a) = V^{\pi}(s) \qquad (1)
\end{aligned}
$$

Therefore, $V^{\pi'}(s) \geq V^{\pi}(s)$ from the policy improvement theorem

GLIE

Definition

G*reedy in the* **L***imit with* **I***nfinite* **E***xploration*

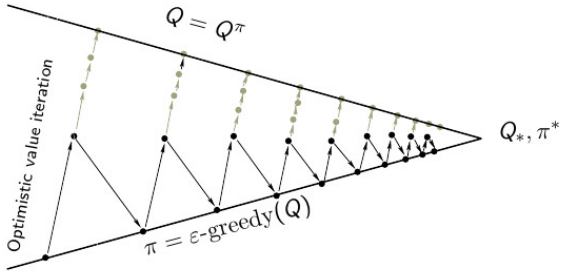
- All state-action pairs are visited infinitely often
- The policy converges to a purely greedy policy

$$\lim_{k\to\infty} \pi_k(a|s) = \mathbf{1}_{a=\arg\max_{a'} Q_k(s,a)}$$

- ε-greedy is GLIE if ε decays to 0 asymptotically, for example,

$$\varepsilon_k = \frac{1}{k}$$

Optimistic GLIE Policy Iteration



Every episode

- Monte Carlo Policy Evaluation $Q \approx Q^\pi$
- Policy improvement using ϵ- greedy with ε decay

Algorithm Monte Carlo Control : GLIE

Initalize Q(s,a) = 0, set $\varepsilon = 1$;
Create an ε-greedy initial policy π_1;
for $k = 1, 2, \cdots, K$ **do**
    Sample a trajectory from policy π_k
    for For each state action (s_t, a_t) pair in the trajectory **do**
        Compute the total discounted return G_t starting from (s_t, a_t)

$$N(s_t, a_t) = N(s_t, a_t) + 1$$

$$Q(s_t, a_t) = Q(s_t, a_t) + \frac{1}{N(s_t, a_t)}(G_t - Q(s_t, a_t))$$

    end for
    Set $\epsilon \leftarrow \frac{1}{k}$ and perform the policy improvement step as

$$\pi_{k+1} = \epsilon\text{-greedy}(\pi_k)$$

end for

TD Control

TD Control

- Natural idea : Use TD instead of MC in policy iteration framework
- Apply TD to evaluate $Q(s, a)$ in the evaluation step
- Use ε-greedy policy improvement in the update step

TD Evaluation of Q Function

- State-action value function of a policy π:

$$\begin{aligned} Q^{\pi}(s,a) &\stackrel{\text{def}}{=} \mathbb{E}_{\pi}(G_t | s_t = s, a_t = a) \\ &= \mathbb{E}_{\pi}(r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) | s_t = s, a_t = a) \end{aligned}$$

- Iterative DP policy evaluation:

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma \sum_{a'} (\pi(s',a') Q_k(s',a')) \right]$$

$$Q_k \rightarrow Q^{\pi}$$

- TD approximation: Given the transition $(s_t, a_t, r_{t+1}, s_{t+1})$, sample $a' \sim \pi(s_{t+1}, \cdot)$, and update

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t [r_{t+1} + \gamma Q(s_{t+1}, a') - Q(s_t, a_t)]$$

TD Evaluation of Q Function : SARSA

- TD approximation: Given the transition $(s_t, a_t, r_{t+1}, s_{t+1})$, sample $a' \sim \pi(s_{t+1}, \cdot)$, and perform the following update

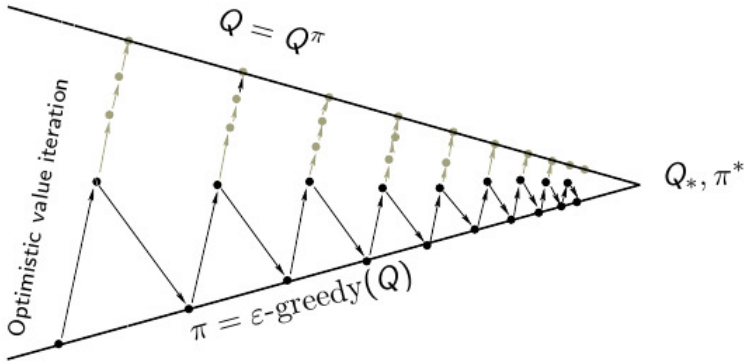
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t[r_{t+1} + \gamma Q(s_{t+1}, a') - Q(s_t, a_t)]$$

- On-policy version (SARSA): $a_t \sim \pi(s_t, \cdot)$
- Off-policy version: $a_t \sim \mu(s_t, \cdot)$;
  - ★ Need to multiply the term inside square brackets with suitable importance sampling factor

TD Evaluation : Convergence

- On Policy and off-policy version covnerges to Q^π
  - ★ Convergence takes place under similar conditions as TD methods for V^π
    - State and action spaces are finite
    - All state-action pairs are visited infinitely often
    - Robbins-Monroe condition: $\sum_t \alpha_t = \infty$, $\sum_t \alpha_t^2 < \infty$

Optimistic Policy Iteration



Along every episode, we interleave one step of policy evaluation followed ϵ-greedy policy improvement

SARSA: On-Policy Control

- Policy is always ε-greedy with ε decay
- Given a trajectory segment $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$ generated by the ε-greedy policy, update

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Algorithm SARSA

Initialize $Q(s, a)$ arbitrarily, with Q at terminal states set to zero
for Repeat for each episode **do**
    Initialize s, choose action a at s using ϵ-greedy over Q
    for Repeat for each step in the episode **do**
        Take action a, observe reward r and next state s'
        Choose action a' for state s' using ϵ-greedy over Q

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)], s \leftarrow s', a \leftarrow a'$$

    end for
end for

Learning Optimal State-Action Value Function

- Optimal Q function:

$$Q_*(s,a) \stackrel{\text{def}}{=} \max_\pi Q^\pi(s,a) = Q^{\pi^*}(s,a)$$

- Bellman optimality equation:

$$Q_*(s,a) \quad = \quad \mathbb{E}\left[r_{t+1} + \gamma \max_{a'} Q_*(s_{t+1}, a') | s_t = s, a_t = a\right]$$

- Iterative DP approximation

$$Q_{k+1}(s,a) \quad \leftarrow \quad \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q_k(s', a')\right]$$

$$Q_k \rightarrow Q_*$$

Q-Learning: Off-Policy Control

- Policy is always ε-greedy with ε decay
- Given a trajectory segment $(s_t, a_t, r_{t+1}, s_{t+1})$ generated by the ε-greedy policy, update

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)]$$

Algorithm Q-Learning

1: Initialize $Q(s, a)$ arbitrarily, with Q at terminal states set to zero
2: **for** Repeat for each episode **do**
3: Initialize s, choose action a at s using ϵ-greedy over Q
4: **for** Repeat for each step in the episode **do**
5: Take action a, observe reward r and next state s'
6: Choose target to update $Q(s, a)$ by being greedy at s' as shown below
7:

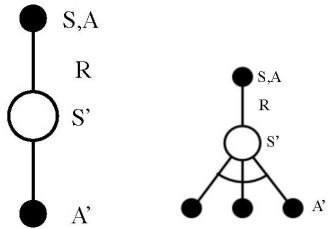
$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)], s \leftarrow s'$$

8: **end for**
9: **end for**

SARSA and Q-Learning : Backup diagram

- Q-learning is an off-policy algorithm
  - ★ Target policy is greedy w.r.t to $Q(s, a)$,
  - ★ Behaviour policy is ε-greedy w.r.t to $Q(s, a)$

Backup Diagrams for SARSA and Q-Learning



TCS Research & Innovation

Summary and Closing Remarks

Summary

- MC-based evaluation of V^π (also possible for Q^π)
- GLIE Monte-Carlo control converges to optimal action value function
- TD-based approximate evaluation of V^π, Q^π
  - ★ 1-step TD, n-step TD, TD(λ), SARSA
  - ★ Convergence guarantees under infinite exploration, and Robbins-Monroe condition
- TD-based control
  - ★ On-policy control with SARSA (also possible: n-step SARSA, SARSA(λ))
  - ★ Off-policy control with Q-learning
  - ★ Based on optimistic policy iteration, and GLIE

MC Vs TD Control

- TD methods have several advantages over MC methods
  - ★ Lower variance
  - ★ Online
  - ★ Partial sequences

Schematic View of MC and TD Algorithms

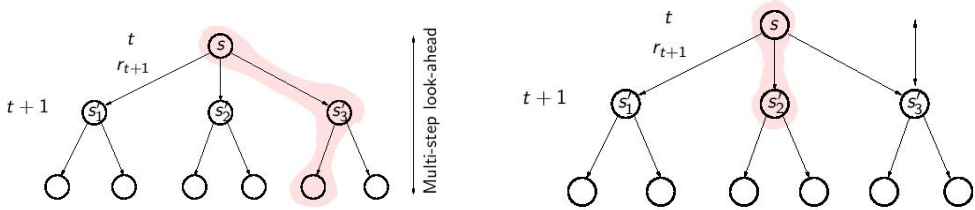
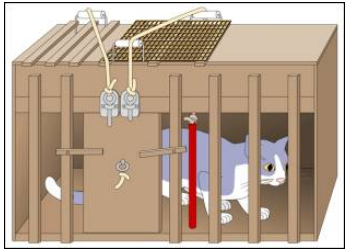
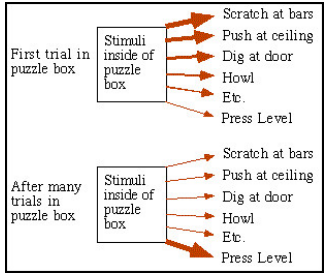


Figure: MC Algorithm and TD Algorithms

Thondrike's Cat and Exploration



Thondrike's cat

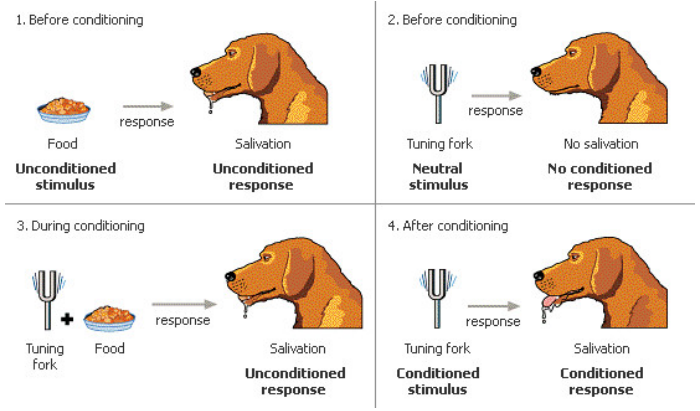


Actions by cat

ϵ-greedy strategy helps to explore !!

Figure Source: Oscar Education : Blogpost

Pavlov's Dog and Temporal Difference

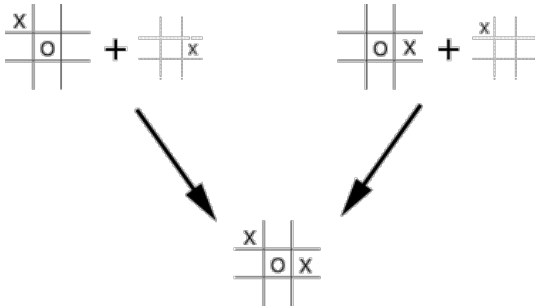


Pavlov's Dog

$$V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$$

Figure Source: https://www.age-of-the-sage.org/psychology/pavlov.html

Afterstates



- Tic-Tac-Toe : States : Board positions and moves are actions
- A conventional action-value function ($Q(s, a)$) would map or learn about the two state action pairs on the top row separately
- An afterstate value function would immediately evaluate both equally
- Any learning about the position-move pair on the left would immediately transfer to the pair on the right

What Next ?

All methods discussed under model free methods are in the tabular setting

- Next: richer ways to represent value functions
- Needed for very large (or continuous) state spaces
- What if the action space is large (or continuous)?

Over to Deep RL !!