Reinforcement Learning

Prabuchandran K.J.

Assistant Professor, IIT Dharwad

17 January 2020

Outline

- Decision making under certainty

Outline

- Decision making under certainty
  - Monte Hall Problem
  - Bayes Decision Making
- Reinforcement Learning Overview

Outline

- Decision making under certainty
  - Monte Hall Problem
  - Bayes Decision Making
- Reinforcement Learning Overview
- Bandits

Outline

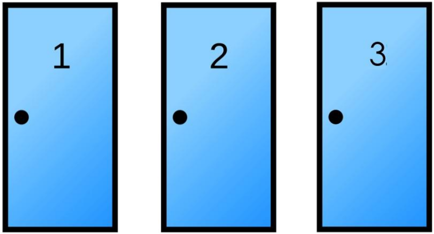
- Decision making under certainty
  - Monte Hall Problem
  - Bayes Decision Making
- Reinforcement Learning Overview
- Bandits
- Markov Decision Process Framework

Monty Hall Problem

You're on a game show where you're presented with three doors.

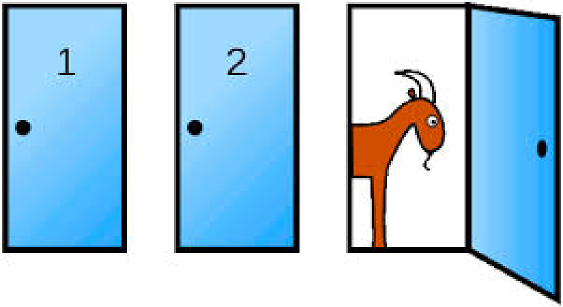
Behind two of the doors, there are goats.

Behind one of the doors is a shiny new car.



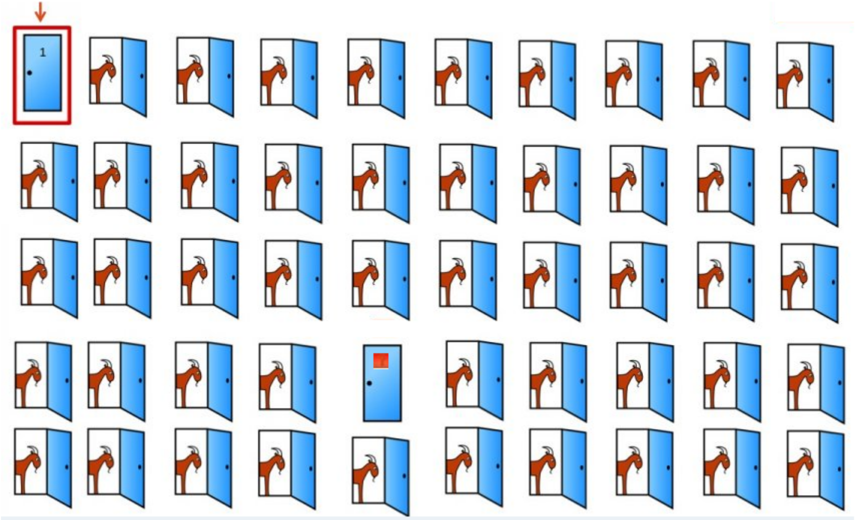
Monty Hall Problem

Problem Description



Monty Hall Problem

Solution Intuition



1000 doors

Monty Hall Problem

Solution

- Define all possible outcomes of the experiment.

Monty Hall Problem

Solution

- Define all possible outcomes of the experiment.
- First component denotes what you chose and second component denotes what the host has shown you.

$$\Omega = \{G_1G_2, G_2G_1, TG_1, TG_2\}$$

Monty Hall Problem

Solution

- Define all possible outcomes of the experiment.
- First component denotes what you chose and second component denotes what the host has shown you.

$$\Omega = \{G_1G_2, G_2G_1, TG_1, TG_2\}$$

- $\mathbf{P}(G_1G_2) = \frac{1}{3},\ \mathbf{P}(G_2G_1) = \frac{1}{3},\ \mathbf{P}(TG_1) = \frac{1}{6},\ \mathbf{P}(TG_2) = \frac{1}{6}$

Monty Hall Problem

Outcomes favorable if you switch

- $W = \{G_1G_2, G_2G_1\}$

Monty Hall Problem

Outcomes favorable if you switch

- $W = \{G_1G_2, G_2G_1\}$
- What is the probability that W happens ?

Monty Hall Problem

Outcomes favorable if you switch

- $W = \{G_1G_2, G_2G_1\}$
- What is the probability that W happens ?

  $\mathbf{P}(W) = \mathbf{P}(G_1G_2) + \mathbf{P}(G_2G_1) = \frac{1}{3} + \frac{1}{3} = \frac{2}{3}$

Monty Hall Problem

Outcomes favorable if you do not switch

- $\bar{W} = \{TG_1, TG_2\}$

Monty Hall Problem

Outcomes favorable if you do not switch

- $\bar{W} = \{TG_1, TG_2\}$
- What is the probability that $\bar{W}$ happens ?

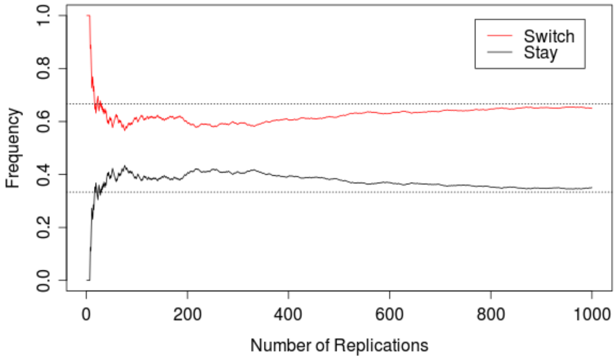
Monty Hall Problem

Outcomes favorable if you do not switch

- $\bar{W} = \{TG_1, TG_2\}$
- What is the probability that $\bar{W}$ happens ?

  $\mathbf{P}(\bar{W}) = \mathbf{P}(TG_1) + \mathbf{P}(TG_2) = \frac{1}{6} + \frac{1}{6} = \frac{1}{3}$

Monty Hall Problem



Cumulative results of the first 1000 trials comparing the two different strategies

Monty Hall Problem

Conclusion

- Probability of winning if you switch $= \frac{2}{3}$
- Probability of winning if you do not switch $= \frac{1}{3}$

Monty Hall Problem

Conclusion

- Probability of winning if you switch $= \frac{2}{3}$
- Probability of winning if you do not switch $= \frac{1}{3}$

We have seen one simple instance of decision making problem under uncertainity.

Bayesian Decision Making

Loan Lending Problem

Could you lend me a loan?

Prediction Problem

- Predict when a customer asks for a loan whether he will default or return? Minimize error

Prediction Problem

- Predict when a customer asks for a loan whether he will default or return? Minimize error

- What is predicton/misclassification error?

Prediction Problem

- Predict when a customer asks for a loan whether he will default or return? Minimize error

- What is predicton/misclassification error?

customer returns when we predict he will default

or customer defaults when we predict he will return

Prediction Problem

- Predict when a customer asks for a loan whether he will default or return? Minimize error

- What is predicton/misclassification error?

customer returns when we predict he will default

or customer defaults when we predict he will return

- Known information

P(Default)	0.2
P(Return)	0.8

Table: Prior

Prediction Strategies

- Predict the customer will always ‘Default’

Prediction Strategies

- Predict the customer will always ‘Default’
- Predict the customer will always ‘Return’

Prediction Strategies

- Predict the customer will always ‘Default’
- Predict the customer will always ‘Return’
- Toss your lucky coin with bias for head p if head - predict he defaults and tail - predict he returns

Prediction Error in Strategies

- 'Say always default' : $P(\text{error}) = P(\text{whenever return happens}) = 0.8$

Prediction Error in Strategies

- 'Say always default' : $P(\text{error}) = P(\text{whenever return happens}) = 0.8$
- 'Say always return' : $P(\text{error}) = P(\text{whenever default happens}) = 0.2$

Prediction Error in Strategies

- 'Say always default' : $P(\text{error}) = P(\text{whenever return happens}) = 0.8$
- 'Say always return' : $P(\text{error}) = P(\text{whenever default happens}) = 0.2$
- Mixed Strategy - $P(\text{error}) = p * 0.8 + (1 - p) * 0.2 = 0.2 + 0.6p$

How to incorporate information?

- Suppose we have additional information

How to incorporate information?

- Suppose we have additional information
- We call a loan High risk H - if Loan amount $>$ 50% annual salary
- We call a loan Low risk L - if Loan amount $<$ 50% annual salary
- Additional Information

$P(H\|D)$	2/3
$P(H\|R)$	1/10

Table: Likelihood

Posterior Probability Computation

- $P(D|H)$ (posterior) $= \frac{P(H|D) \text{ (Likelihood) } *P(D) \text{ (Prior)}}{P(H) \text{ (Evidence)}}$

Posterior Probability Computation

- $P(D|H)$ (posterior) $= \frac{P(H|D) \text{ (Likelihood) } \; *P(D) \text{ (Prior)}}{P(H) \text{ (Evidence)}}$

$$P(H|D)P(D) = 2/3 * 2/10 = 4/30$$

$$P(H) = P(H|D)P(D) + P(H|R)P(R)$$

$$= 2/3 * 2/10 + 1/10 * 8/10 = 64/300$$

$$P(D|H) = 4/30 * 300/64 = 5/8 = 0.625$$

- Similarly we can compute $P(D|L) = 0.08$

Bayes Prediction Rule

- How to predict with new information?

Bayes Prediction Rule

- How to predict with new information?

- Since $P(D|H) > P(R|H)$, predict default for high risk loans

Bayes Prediction Rule

- How to predict with new information?

- Since $P(D|H) > P(R|H)$, predict default for high risk loans

- Since $P(R|L) > P(D|L)$, predict return for low risk loans

Bayes Prediction Rule

- How to predict with new information?
- Since $P(D|H) > P(R|H)$, predict default for high risk loans
- Since $P(R|L) > P(D|L)$, predict return for low risk loans
- What is misclassification error?

Bayes Prediction Rule

- How to predict with new information?
- Since $P(D|H) > P(R|H)$, predict default for high risk loans
- Since $P(R|L) > P(D|L)$, predict return for low risk loans
- What is misclassification error?

$$
\begin{aligned}
P(\text{error}) &= P(\text{person defaults and we predict return}) \\
&\quad + P(\text{person returns and we predict default}) \\
&= P(D)P(L|D) + P(R)P(H|R) = 0.15
\end{aligned}
$$

Bayes Prediction Rule

- How to predict with new information?
- Since $P(D|H) > P(R|H)$, predict default for high risk loans
- Since $P(R|L) > P(D|L)$, predict return for low risk loans
- What is misclassification error?

$$\begin{aligned} P(\text{error}) &= P(\text{person defaults and we predict return}) \\ &+ P(\text{person returns and we predict default}) \\ &= P(D)P(L|D) + P(R)P(H|R) = 0.15 \end{aligned}$$

Bayes Decision Making Summary

- In general, there are more than one feature

Bayes Decision Making Summary

- In general, there are more than one feature
- Likelihood information is given as density/distribution over features under a given class

Bayes Decision Making Summary

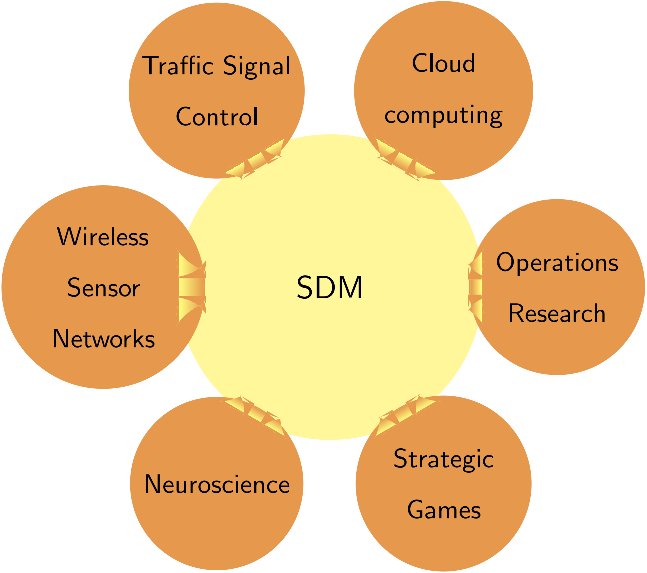
- In general, there are more than one feature
- Likelihood information is given as density/distribution over features under a given class
- Bayes rule predicts the class which has high posterior probability given that feature

Bayes Decision Making Summary

- In general, there are more than one feature
- Likelihood information is given as density/distribution over features under a given class
- Bayes rule predicts the class which has high posterior probability given that feature
- Bayes rule is optimal and achieves the least classification error

Reinforcement Learning Overview

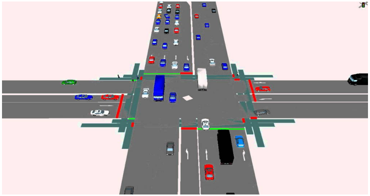
Sequential Decision Making (SDM) Problems



Examples of SDM Problems

Strategic Games

Robo Soccer



Traffic Signal Control

Inventory Management

Common Features of SDM Problems

- Long-term goal that needs to be achieved
- Uncertainity in the evolution of configuration (state)
- Decisions (actions) need to be taken in stages
- Simple feedback signal (reward/cost) - how good is the action for the given state
- Available information or experience - state, action and reward

Solving SDM Problems

How do we model?

Solving SDM Problems

How do we model?

Markov Decision Process

Solving SDM Problems

How do we model?

Markov Decision Process

How to learn from experience?

Solving SDM Problems

How do we model?

Markov Decision Process

How to learn from experience?

Reinforcement Learning and Stochastic Optimization algorithms

Solving SDM Problems

How do we model?

Markov Decision Process

How to learn from experience?

Reinforcement Learning and Stochastic Optimization algorithms

How to analyse these algorithms?

Solving SDM Problems

How do we model?

Markov Decision Process

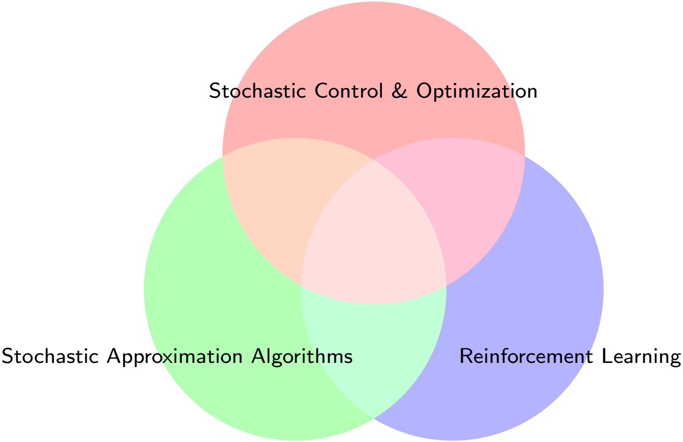
How to learn from experience?

Reinforcement Learning and Stochastic Optimization algorithms

How to analyse these algorithms?

Stochastic Approximation framework

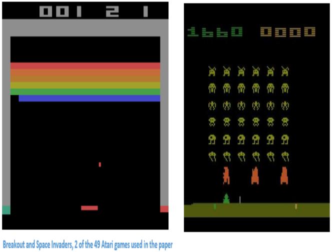
Solution to SDM problems



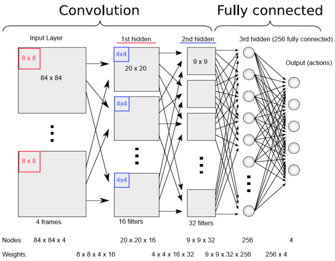
Successful Reinforcement Learning (RL) solutions

- Q-learning / Deep Q-Networks (DQN)
- Actor-Critic methods (AC) / Policy gradient methods
- Upper Confidence Tree (UCT) / Monte-Carlo Tree Search algorithm

Deep Q-Networks

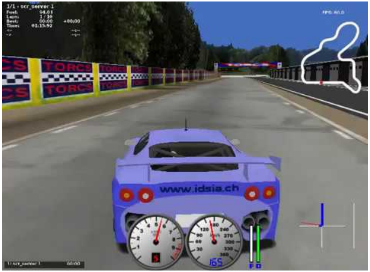


Break out and Space Invaders

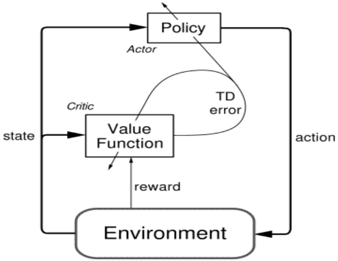


Deep Q-Network

Deep Deterministic Policy Gradient

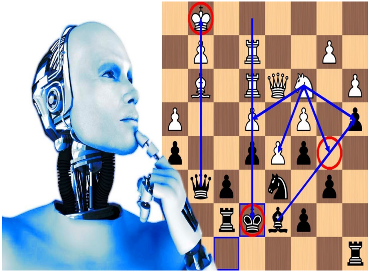


TORCS car simulation

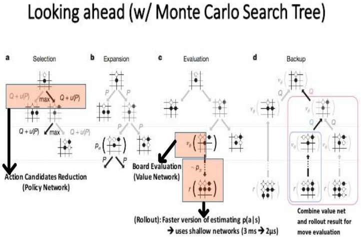


Actor-Critic Network

Upper Confidence Tree (UCT)



Alpha Zero



UCT algorithm

Reason for Success

Rule Based Methods

Reason for Success

Rule Based Methods

$$\Downarrow$$

Feature Based Methods

Reason for Success

Rule Based Methods

$$\Downarrow$$

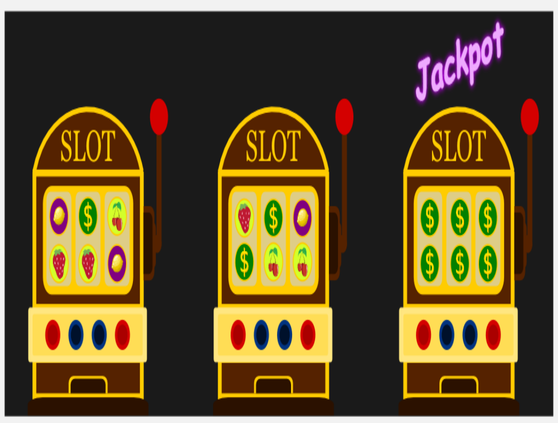
Feature Based Methods

$$\Downarrow$$

Automatic Feature Based Methods

Stateless Reinforcement Learning

Multi Arm Bandit (MAB) Problem



Choose the slot machine

Stochastic Bandits Problem Setting

- K choices or K arms $\{1, 2, \ldots, K\}$

Stochastic Bandits Problem Setting

- K choices or K arms $\{1, 2, \ldots, K\}$
- Each choice has average reward $Q(a)$ based on the arm chosen

Stochastic Bandits Problem Setting

- K choices or K arms $\{1, 2, \ldots, K\}$
- Each choice has average reward $Q(a)$ based on the arm chosen
- Rewards underlying distribution is $P(a)$ for arm a

Stochastic Bandits Problem Setting

- K choices or K arms $\{1, 2, \ldots, K\}$
- Each choice has average reward $Q(a)$ based on the arm chosen
- Rewards underlying distribution is $P(a)$ for arm a
- Collect as much reward in T rounds.

Simple Solution

- Determine $a^* = \max_a Q(a)$

Simple Solution

- Determine $a^* = \max_a Q(a)$
- Play the optimal arm a^* in all T rounds

Simple Solution

- Determine $a^* = \max_a Q(a)$
- Play the optimal arm a^* in all T rounds
- What is the total reward collected?

Simple Solution

- Determine $a^* = \max_a Q(a)$
- Play the optimal arm a^* in all T rounds
- What is the total reward collected? $Q(a^*)T$

Simple Solution

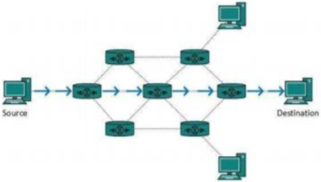
- Determine $a^* = \max_a Q(a)$
- Play the optimal arm a^* in all T rounds
- What is the total reward collected? $Q(a^*)T$
- Problem: Average reward unknown for all arms and has to be learnt

Revisit: Stochastic Bandits Problem Setting

- K choices or K arms $\{1, 2, \ldots, K\}$
- Each choice has average reward $Q(a)$ based on the arm chosen
- Rewards underlying distribution is $P(a)$ for arm a
- Reward Distribution or average reward unknown
- Collect as much reward on an average in T rounds.

Application of Bandits

Clinical Trials

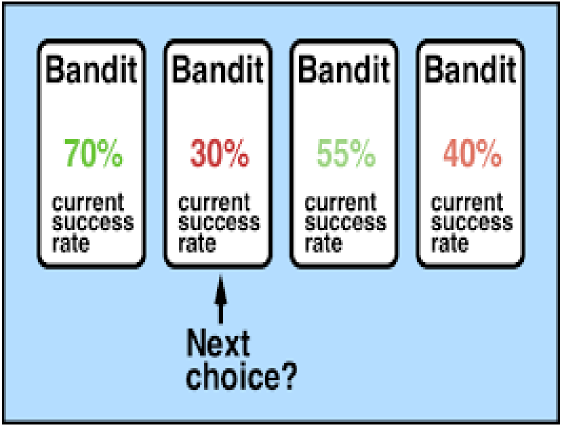


Network Routing

Online Ad Placement

Game Design

Explore vs Exploit Dilemma



Current Success Estimates

Bernoulli Bandit

- 2 Arms/Coins

Bernoulli Bandit

- 2 Arms/Coins
- Each coin has probability of head $Q(1)$ and $Q(2)$

Bernoulli Bandit

- 2 Arms/Coins
- Each coin has probability of head $Q(1)$ and $Q(2)$
- Head/Success yields reward 1 and Tail/Failure yields 0

Bernoulli Bandit

- 2 Arms/Coins
- Each coin has probability of head $Q(1)$ and $Q(2)$
- Head/Success yields reward 1 and Tail/Failure yields 0
- Average reward of coin 1 and coin 2 - $Q(1)$ and $Q(2)$ probabilities and they are unknown

Bernoulli Bandit

- 2 Arms/Coins
- Each coin has probability of head $Q(1)$ and $Q(2)$
- Head/Success yields reward 1 and Tail/Failure yields 0
- Average reward of coin 1 and coin 2 - $Q(1)$ and $Q(2)$ probabilities and they are unknown
- In T rounds, choose either coin 1 or coin 2 at evey time $t \in \{1, 2, \ldots, T\}$

Bernoulli Bandit

- 2 Arms/Coins
- Each coin has probability of head $Q(1)$ and $Q(2)$
- Head/Success yields reward 1 and Tail/Failure yields 0
- Average reward of coin 1 and coin 2 - $Q(1)$ and $Q(2)$ probabilities and they are unknown
- In T rounds, choose either coin 1 or coin 2 at evey time $t \in \{1, 2, \ldots, T\}$
- Goal is to maximize expected total reward ($\mathcal{TR}$) collected

$$\mathcal{TR} = \mathbb{E}[R_1 + R_2 + \ldots + R_T]$$

Greedy Algorithm

- Keep track of the proportion of heads

Greedy Algorithm

- Keep track of the proportion of heads

- Head estimates are $\hat{Q}_t(1)$ and $\hat{Q}_t(2)$

Greedy Algorithm

- Keep track of the proportion of heads

- Head estimates are $\hat{Q}_t(1)$ and $\hat{Q}_t(2)$

$$\hat{Q}_t(a) = \frac{\sum_{k=1}^{t} R_k * I_{a_k}(a)}{k}, \quad a = 1, 2$$

- Pick $a_{t+1} = \arg\max\{\hat{Q}_t(1), \hat{Q}_t(2)\}$

Greedy Algorithm

- Keep track of the proportion of heads
- Head estimates are $\hat{Q}_t(1)$ and $\hat{Q}_t(2)$

$$\hat{Q}_t(a) = \frac{\sum_{k=1}^{t} R_k * I_{a_k}(a)}{k}, \quad a = 1, 2$$

- Pick $a_{t+1} = \arg\max\{\hat{Q}_t(1), \hat{Q}_t(2)\}$
- Greedy picks suboptimal arm $O(T)$ rounds

UCB Algorithm

- Keep track of the proportion of heads for each coin

UCB Algorithm

- Keep track of the proportion of heads for each coin
- Keep track of the number of plays of each coin $N_1(t), N_2(t)$

UCB Algorithm

- Keep track of the proportion of heads for each coin
- Keep track of the number of plays of each coin $N_1(t), N_2(t)$
- Head estimates are $\hat{Q}_t(1)$ and $\hat{Q}_t(2)$

UCB Algorithm

- Keep track of the proportion of heads for each coin
- Keep track of the number of plays of each coin $N_1(t), N_2(t)$
- Head estimates are $\hat{Q}_t(1)$ and $\hat{Q}_t(2)$
- Pick $a_{t+1} = \arg\max\{\hat{Q}_t(1) + \text{bonus}(1), \hat{Q}_t(2) + \text{bonus}(2)\}$, where $\text{bonus}(i) = \sqrt{\frac{2\ln t}{N_i(t)}},\ i = 1, 2$

UCB Algorithm

- Keep track of the proportion of heads for each coin
- Keep track of the number of plays of each coin $N_1(t), N_2(t)$
- Head estimates are $\hat{Q}_t(1)$ and $\hat{Q}_t(2)$
- Pick $a_{t+1} = \arg\max\{\hat{Q}_t(1) + \text{bonus}(1), \hat{Q}_t(2) + \text{bonus}(2)\}$, where $\text{bonus}(i) = \sqrt{\frac{2\ln t}{N_i(t)}},\ i = 1, 2$
- UCB picks suboptimal arm $O(\lg T)$ rounds

Bayesian Treatment

- Treat each coin's success probabilities as random variables

Bayesian Treatment

- Treat each coin's success probabilities as random variables
- For each of the random variables assume an underlying distribution

Bayesian Treatment

- Treat each coin's success probabilities as random variables
- For each of the random variables assume an underlying distribution
- Based on reward observations update success probabilities distributions

Exploring choice of distribution

- Which distribution can we use?

Exploring choice of distribution

- Which distribution can we use? Uniform distribution

Exploring choice of distribution

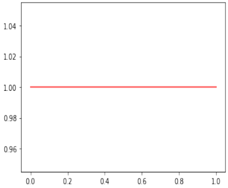
- Which distribution can we use? Uniform distribution
- This is same as $beta(\alpha, \beta)$ distribution with $\alpha = \beta = 1$

Beta distribution

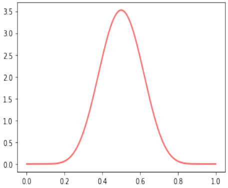
$$f(p;\alpha,\beta) = \frac{p^{\alpha-1}(1-p)^{\beta-1}}{B(\alpha,\beta)}, \quad 0 \le p \le 1$$

$B(\alpha,\beta)$ - Beta function, $\alpha,\beta > 0$

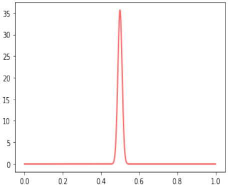
Beta density for different α and β



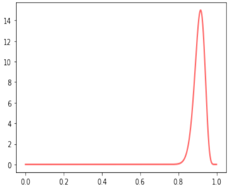
$\alpha = \beta = 1$



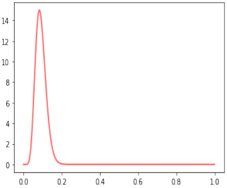
$\alpha = \beta = 100$



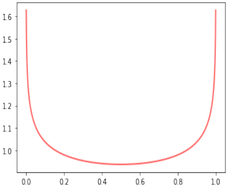
$\alpha = \beta = 1000$



$\alpha = 100, \beta = 10$



$\alpha = 10, \beta = 100$



$\alpha = \beta = 0.9$

Bayesian Update : Prior to Posterior

- Suppose we toss a coin and we obtain head then how do we update distribution?

Bayesian Update : Prior to Posterior

- Suppose we toss a coin and we obtain head then how do we update distribution?
- Should we increase α or β?

Bayesian Update : Prior to Posterior

- Suppose we toss a coin and we obtain head then how do we update distribution?
- Should we increase α or β?
- What is α_{new} and β_{new}?

Bayesian Update : Prior to Posterior

- Suppose we toss a coin and we obtain head then how do we update distribution?
- Should we increase α or β?
- What is α_{new} and β_{new}?

$$\alpha_{new} = \alpha_{old} + 1 \quad \text{and} \quad \beta_{new} = \beta_{old}$$

Thompson sampling

- For coin 1 maintain an uncertainity model $beta(\alpha_1, \beta_1)$
- Similarly for coin 2 maintain an uncertainity model $beta(\alpha_2, \beta_2)$
- Whenever head or tail comes from the coin, we know how to update the parameters
- If head comes increase α and if tail then increase β

Thompson sampling ctd

- How to choose which coin to choose at time t?

Thompson sampling ctd

- How to choose which coin to choose at time t?

- Sample values from coin1 as well as coin2 based on the current model

$$\hat{Q}(1) \sim beta(\alpha_1, \beta_1)$$

$$\hat{Q}(2) \sim beta(\alpha_2, \beta_2)$$

Thompson sampling ctd

- How to choose which coin to choose at time t?
- Sample values from coin1 as well as coin2 based on the current model

$$\hat{Q}(1) \sim beta(\alpha_1, \beta_1)$$

$$\hat{Q}(2) \sim beta(\alpha_2, \beta_2)$$

- $a_{t+1} = \arg\max\{\hat{Q}(1), \hat{Q}(2)\}$

Thompson sampling ctd

- How to choose which coin to choose at time t?
- Sample values from coin1 as well as coin2 based on the current model

$$\hat{Q}(1) \sim beta(\alpha_1, \beta_1)$$

$$\hat{Q}(2) \sim beta(\alpha_2, \beta_2)$$

- $a_{t+1} = \arg\max\{\hat{Q}(1), \hat{Q}(2)\}$
- Thompson sampling picks suboptimal arms $O(\lg T)$ rounds

Thompson sampling Summary

- Maintain uncertainty models for each of the arms

Thompson sampling Summary

- Maintain uncertainty models for each of the arms
- Sample rewards from the uncertainty model for each of the arms

Thompson sampling Summary

- Maintain uncertainty models for each of the arms
- Sample rewards from the uncertainty model for each of the arms
- Pull the arm that has the highest sample reward

Thompson sampling Summary

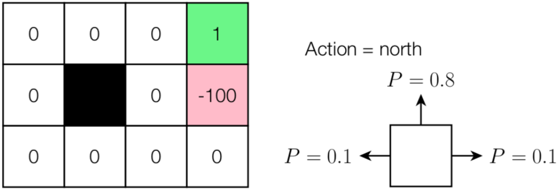
- Maintain uncertainty models for each of the arms
- Sample rewards from the uncertainty model for each of the arms
- Pull the arm that has the highest sample reward
- Update the uncertainty model of the pulled arm in the Bayesian way

Thompson sampling Summary

- Maintain uncertainty models for each of the arms
- Sample rewards from the uncertainty model for each of the arms
- Pull the arm that has the highest sample reward
- Update the uncertainty model of the pulled arm in the Bayesian way
- Thompson sampling achieves $O(\lg T)$ regret

Markov Decision Process

Example Simple MDP



Grid World with discount factor $\gamma = 0.9$

Grid World Setup

- Simple grid world with a ‘goal state’ with reward and a ‘bad state’ with reward -100
- Actions move in the desired direction with probabilty 0.8
- Taking an action that would bump into a wall leaves agent where it is

Grid World MDP

0	0	0	1
0		0	-100
0	0	0	0

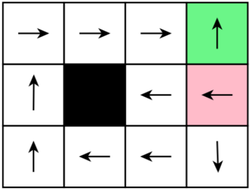
Reward Function

Grid World MDP

5.470	6.313	7.190	8.669
4.802		3.347	-96.67
4.161	3.654	3.222	1.526

Value Function

Grid World MDP



Optimal Policy

A Finite Markov Decision Process (MDP)

- State Space, $S = \{1, 2, \ldots, N\}$

A Finite Markov Decision Process (MDP)

- State Space, $S = \{1, 2, \ldots, N\}$
- Action Space, $A = \{1, 2, \ldots, M\}$

A Finite Markov Decision Process (MDP)

- State Space, $S = \{1, 2, \ldots, N\}$
- Action Space, $A = \{1, 2, \ldots, M\}$
- Probability transition kernel, $P_{i,j}(a)$

$$Pr\{s_{n+1} = j | s_n = i, a_n = a\} = P_{i,j}(a)$$

s_n, s_{n+1} - state at time n and $n + 1$, a_n - action at time n.

A Finite Markov Decision Process (MDP)

- State Space, $S = \{1, 2, \ldots, N\}$

- Action Space, $A = \{1, 2, \ldots, M\}$

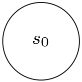
- Probability transition kernel, $P_{i,j}(a)$

$$Pr\{s_{n+1} = j | s_n = i, a_n = a\} = P_{i,j}(a)$$

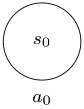
  s_n, s_{n+1} - state at time n and $n+1$, a_n - action at time n.

- Cost function, $k(i, a, j)$

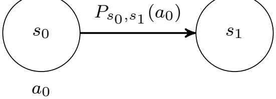
Markov Decision Process (MDP) Evolution



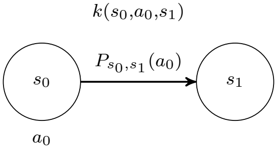
Markov Decision Process (MDP) Evolution



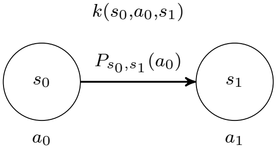
Markov Decision Process (MDP) Evolution



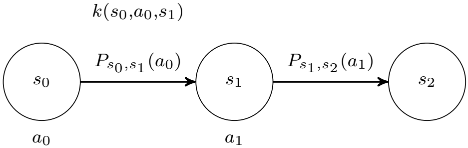
Markov Decision Process (MDP) Evolution



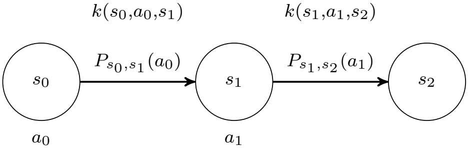
Markov Decision Process (MDP) Evolution



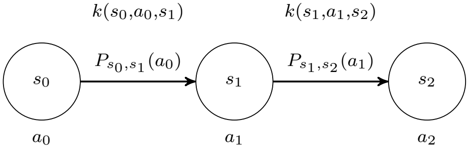
Markov Decision Process (MDP) Evolution



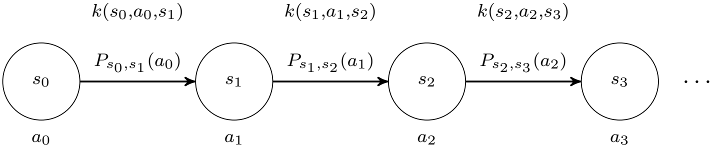
Markov Decision Process (MDP) Evolution



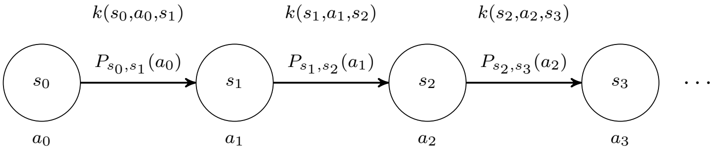
Markov Decision Process (MDP) Evolution



Markov Decision Process (MDP) Evolution



Markov Decision Process (MDP) Evolution



Goal : Find the optimal sequence of actions to minimize a long-term objective

Long-term objective

Total cost incurred in an infinite horizon

$$\mathbb{E}[\underbrace{k(s_0, a_0, s_1)}_{C_1} + \underbrace{k(s_1, a_1, s_2)}_{C_2} + \underbrace{k(s_2, a_2, s_3)}_{C_3} + \cdots]$$

Long-term objective

Total cost incurred in an infinite horizon

$$\mathbb{E}[\underbrace{k(s_0, a_0, s_1)}_{C_1} + \underbrace{k(s_1, a_1, s_2)}_{C_2} + \underbrace{k(s_2, a_2, s_3)}_{C_3} + \cdots]$$

Goal: Find $\{a_0^*, a_1^*, a_2^*, \cdots\}$ that minimize total cost

Long-term objective

Discounted cost incurred in an infinite horizon

$$\mathbb{E}[k(s_0, a_0, s_1) + \gamma\ k(s_1, a_1, s_2) + \gamma^2\ k(s_2, a_2, s_3) + \cdots]$$

Long-term objective

Discounted cost incurred in an infinite horizon

$$\mathbb{E}[k(s_0, a_0, s_1) + \gamma\ k(s_1, a_1, s_2) + \gamma^2\ k(s_2, a_2, s_3) + \cdots]$$

Goal: Find $\{a_0^*, a_1^*, a_2^*, \cdots\}$ that minimize long-run discounted cost

Long-term objective

Average cost incurred in an infinite horizon

$$\lim_{T\to\infty} \mathbb{E}[\frac{k(s_0, a_0, s_1) + k(s_1, a_1, s_2) + \cdots + k(s_{T-1}, a_{T-1}, s_T)}{T}]$$

Long-term objective

Average cost incurred in an infinite horizon

$$\lim_{T \to \infty} \mathbb{E}[\frac{k(s_0, a_0, s_1) + k(s_1, a_1, s_2) + \cdots + k(s_{T-1}, a_{T-1}, s_T)}{T}]$$

Goal: Find $\{a_0^*, a_1^*, a_2^*, \cdots\}$ that minimize long-run average cost

Policy: State Dependent Actions

Stationary Deterministic Policy (SDP)

State	Action
1	$\mu(1)$
2	$\mu(2)$
$\vdots$	$\vdots$
N	$\mu(N)$

Policy $\mu\colon S \to A$

Long term dependencies

How good is a policy ?

Value function

Different Starting states

$$1 \rightarrow \mathbb{E}[k(1, \mu(1), s_1) + \gamma\ k(s_1, \mu(s_1), s_2) + \gamma^2\ k(s_2, \mu(s_2), s_3) \cdots]$$

Long term dependencies

How good is a policy ?

Value function

Different Starting states

$$i \to \mathbb{E}[k(i,\mu(i),s_1) + \gamma\ k(s_1,\mu(s_1),s_2) + \gamma^2\ k(s_2,\mu(s_2),s_3)\cdots]$$

Discounted Cost Value Function

State	Value
1	$V^{\mu}(1)$
2	$V^{\mu}(2)$
$\vdots$	$\vdots$
N	$V^{\mu}(N)$

Value function $V^{\mu}: S \to \mathbb{R}$

$$V^{\mu}(i) = \sum_{n=0}^{\infty} \mathbb{E}[\gamma^n k(s_n, a_n, s_{n+1}) | s_0 = i, \mu], \tag{1}$$

Bellman Equation for a Fixed Policy μ

$$V^{\mu}(i) = \sum_{j=1}^{N} P_{ij}(\mu(i))[k(i,\mu(i),j) + \gamma V^{\mu}(j)] \tag{2}$$

$$V^{\mu} = k^{\mu} + \gamma P^{\mu} V^{\mu}, \tag{3}$$

P^{μ} - transition probabilities between states under μ,

k^{μ} - vector of single-stage costs

Bellman Equation for a Fixed Policy μ

$$V^{\mu}(i) = \sum_{j=1}^{N} P_{ij}(\mu(i))[k(i,\mu(i),j) + \gamma V^{\mu}(j)] \tag{2}$$

$$V^{\mu} = k^{\mu} + \gamma P^{\mu} V^{\mu}, \tag{3}$$

P^{μ} - transition probabilities between states under μ,

k^{μ} - vector of single-stage costs

Long term cost = Expected immediate cost + Expected future cost

Goal for Long-run Discounted Cost Objective

- Optimal value function V^*

$$V^*(i) = \min_{\mu \in \Pi} V_\mu(i), \ \forall i \in S, \tag{4}$$

Π - set of all SDPs

Goal for Long-run Discounted Cost Objective

- Optimal value function V^*

$$V^*(i) = \min_{\mu \in \Pi} V_\mu(i), \ \forall i \in S, \tag{4}$$

Π - set of all SDPs

- Optimal SDP μ^*

$$\mu^*(i) = \arg\min_{a \in A} \sum_{j \in S} P_{ij}(a) \ \left[k(i,a,j) + \gamma V^*(j)\right] \ \forall i \in S. \tag{5}$$

Summary

Goal Find optimal sequence of actions to minimize the given objective

Summary

Goal Find optimal sequence of actions to minimize the given objective

1. Policy representation of actions

Summary

Goal Find optimal sequence of actions to minimize the given objective

1. Policy representation of actions

2. Associate value function V^{μ} for policy μ

Summary

Goal Find optimal sequence of actions to minimize the given objective

1. Policy representation of actions
2. Associate value function V^μ for policy μ
3. Finite number of stationary deterministic policies M^N

Summary

Goal Find optimal sequence of actions to minimize the given objective

1. Policy representation of actions
2. Associate value function V^{μ} for policy μ
3. Finite number of stationary deterministic policies M^N
4. Find optimal value function V^* min of all value function vectors V^{μ}s

Summary

Goal Find optimal sequence of actions to minimize the given objective

1. Policy representation of actions
2. Associate value function V^{μ} for policy μ
3. Finite number of stationary deterministic policies M^N
4. Find optimal value function V^* min of all value function vectors V^{μ}s
5. Find optimal stationary deterministic policy μ^* from V^*

Summary Ctd.

$\boxed{\text{Markov Decision Process (MDP)}}$

Summary Ctd.

Markov Decision Process (MDP)

Model based Methods

Dynamic Programming (DP)

Value Iteration

Policy Iteration

Summary Ctd.

Markov Decision Process (MDP)

Model based Methods

Dynamic Programming (DP)

Value Iteration

Policy Iteration

Model Information not known
Too many states/actions

Summary Ctd.

Markov Decision Process (MDP)

Model based Methods

Dynamic Programming (DP)

- Value Iteration
- Policy Iteration

Model Information not known
Too many states/actions

Model free Methods

Reinforcement Learning (RL)

Summary Ctd.

Markov Decision Process (MDP)

Model based Methods

Dynamic Programming (DP)

Value Iteration

Policy Iteration

Model Information not known
Too many states/actions

Model free Methods

Reinforcement Learning (RL)

Q-learning

Actor Critic methods

Summary Ctd.

Markov Decision Process (MDP)

Model based Methods

Dynamic Programming (DP)

Value Iteration

Policy Iteration

Model Information not known
Too many states/actions

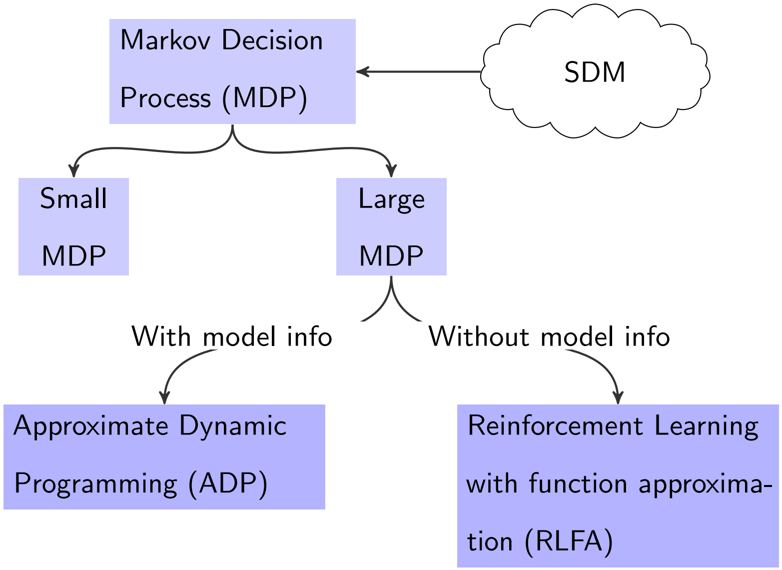
Model free Methods

Reinforcement Learning (RL)

Q-learning

Actor Critic methods

Value Function Approximation


Markov Decision Process (MDP)
SDM
Small MDP
Large MDP
With model info
Without model info
Approximate Dynamic Programming (ADP)
Reinforcement Learning with function approximation (RLFA)

Questions

Thank you !

References I