Lecture 9: Markov Decision Processes
CS486/686 Intro to Artificial Intelligence

2024-6-6

Pascal Poupart
David R. Cheriton School of Computer Science
CIFAR AI Chair at Vector Institute

UNIVERSITY OF WATERLOO

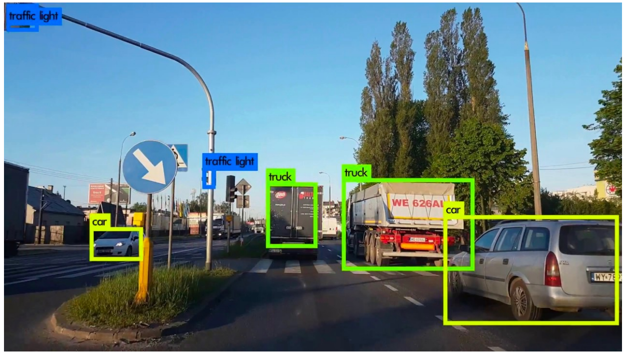
Outline

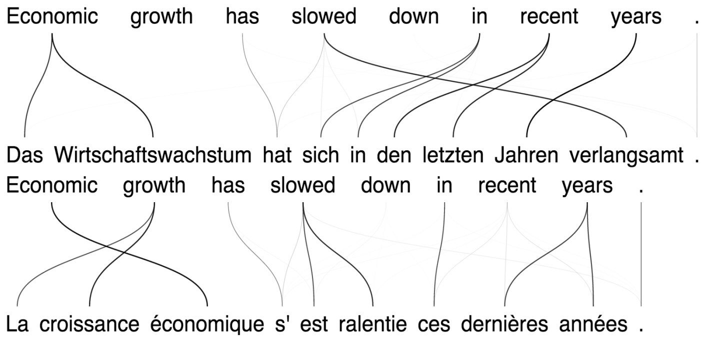
- Course logistics
- Introduction to Reinforcement Learning
- Markov Decision Processes
  - Value Iteration

Machine Learning

- Traditional computer science
  - Program computer for every task

- New paradigm
  - Provide examples to machine
  - Machine learns to accomplish tasks based on examples





UNIVERSITY OF WATERLOO

Machine Learning

- Success mostly due to supervised learning
  - Bottleneck: need lots of labeled data
  - Limitation: mimic data

- Alternatives
  - Unsupervised, semi-supervised, self-supervised learning
  - Transfer learning, domain adaptation, meta-learning
  - Reinforcement Learning

What is Reinforcement Learning?

- Reinforcement learning is also known as
  - Optimal control
  - Approximate dynamic programming
  - Neuro-dynamic programming

- Wikipedia: reinforcement learning is an area of machine learning inspired by behavioural psychology, concerned with how software **agents** ought to take **actions** in an **environment** so as to maximize some notion of cumulative **reward**.

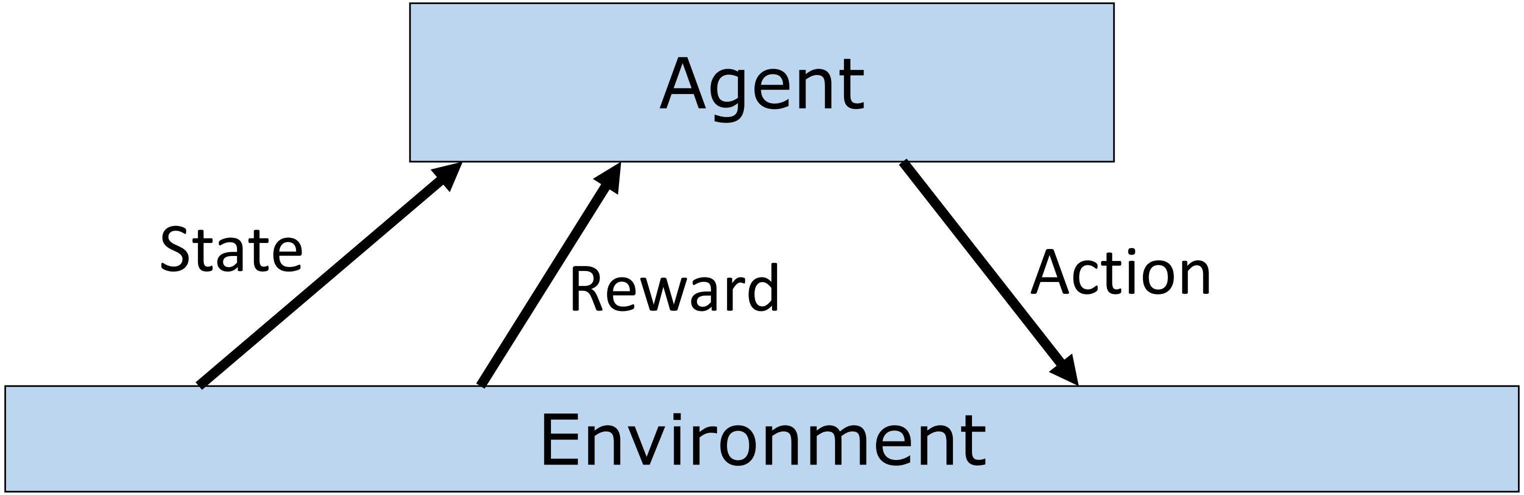
Animal Psychology

- Negative reinforcements
  - Pain and hunger
- Positive reinforcements
  - Pleasure and food

- Reinforcements used to train animals

- Let's do the same with computers

Reinforcement Problem



Goal: Learn to choose actions that maximize rewards

Sample Industrial Use Cases

Less Complex → *More Complex*

Contextual Bandits

Marketing
- ad placement,
- recommender systems

Loyalty programs
- personalized offers

Price management
- airline seat pricing
- cargo shipment pricing
- food pricing

Optimal design
- interface personalization

Bayesian Optimization

Hyperparameter optimization

Troubleshooting
- Customer assistance

Diagnostics
- Fault detection

Design of experiments
- Drug design
- Material design

Sequential decision Making

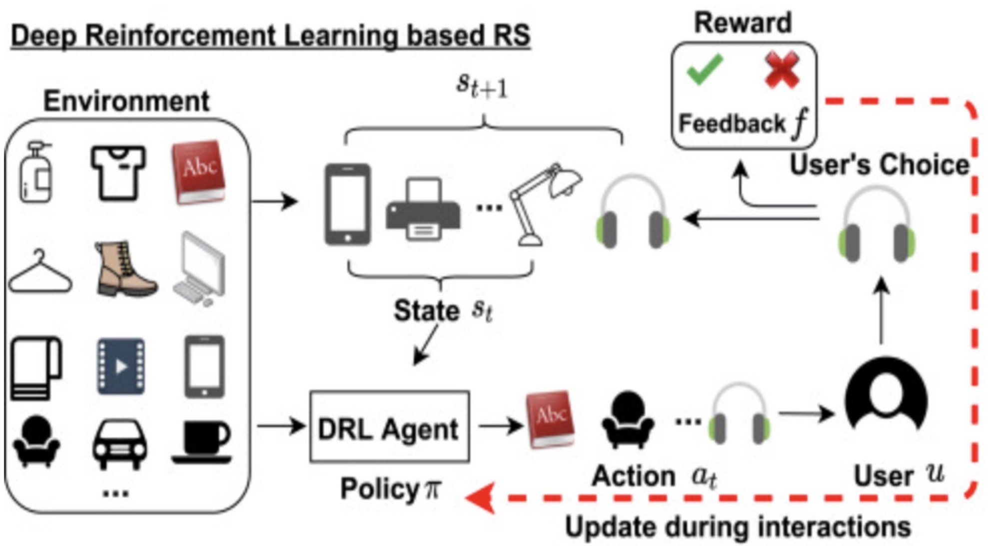
Automated trading
- Stocks, energy

Optimization
- Path planning
- Routing
- Energy consumption

Control
- Robotics
- Autonomous driving

Marketing (Recommender System)

- **Agent:** recommender system
- **Environment:** user
- **State:** context, past recommendations and feedback
- **Action:** recommended item
- **Reward:** value of user feedback



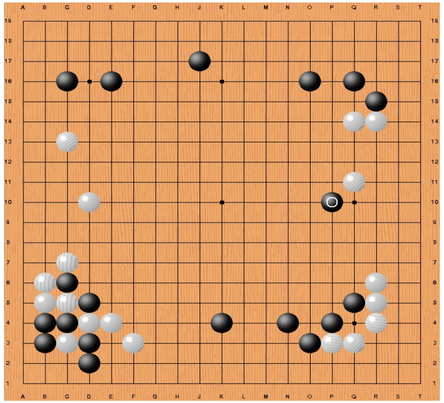
UNIVERSITY OF WATERLOO

Operations Research (vehicle routing)

- **Agent:** vehicle routing system
- **Environment:** stochastic demand
- **State:** vehicle location,
  capacity and depot requests
- **Action:** vehicle route
- **Reward:** - travel costs

Game Playing (Computer Go)

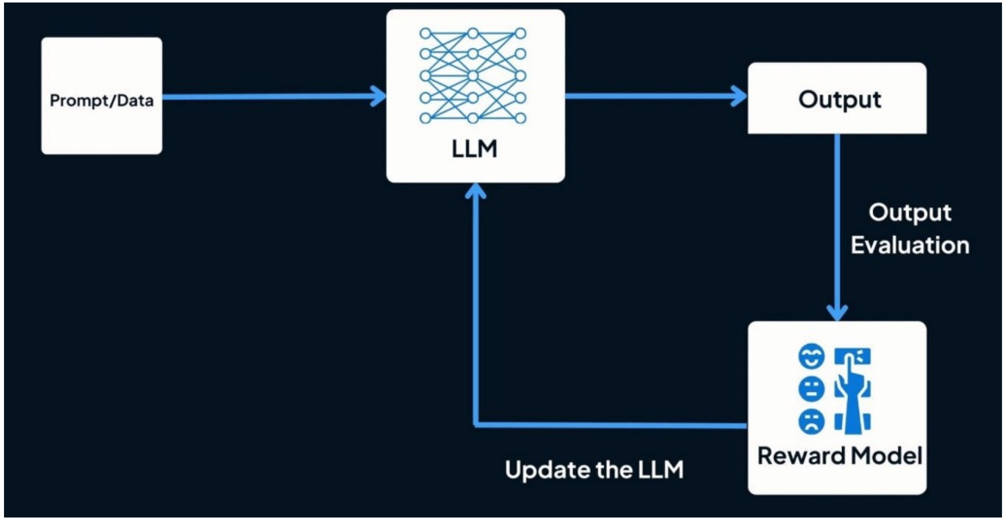
- **Agent:** player
- **Environment:** opponent
- **State:** board configuration
- **Action:** next stone location
- **Reward:** +1 win / -1 loose



- 2016: AlphaGo defeats Lee Sedol (4-1)
  - Game 2 move 37: AlphaGo plays unexpected move (odds 1/10,000)

Conversational Agents (RL from Human Feedback)

- **Agent:** system
- **Environment:** user
- **State:** history of past utterances
- **Action:** system utterance
- **Reward:** task completion, human feedback



Credit: https://www.twine.net/blog/what-is-reinforcement-learning-from-human-feedback-rlhf-and-how-does-it-work/

Computational Finance (Trading)

- **Agent:** trading software
- **Environment:** other traders
- **State:** price history
- **Action:** buy/sell/hold
- **Reward:** amount of profit

Example: how to purchase a large # of shares in a short period of time without affecting the price

Reinforcement Learning

- Comprehensive, but challenging form of machine learning
  - Stochastic environment
  - Incomplete model
  - Interdependent sequence of decisions
  - No supervision
  - Partial and delayed feedback

- Long term goal: continual learning

Markov Decision Process

Components	Formal definition	Inventory management
States	$s \in S$	inventory levels
Actions	$a \in A$	{doNothing, orderWidgets}
Rewards	$r \in \mathbb{R}$	Profit ($)
Transition model	$\Pr(s_t \| s_{t-1}, a_{t-1})$	Stochastic demand
Reward model	$\Pr(r_t \| s_t, a_t)$ $R(s_t, a_t) = \sum_{r_t} r_t \Pr(r_t \| s_t, a_t)$	$R(s_t, a_t) =$ sales – costs – storage
Discount factor	$0 \leq \gamma \leq 1$	$\gamma = 0.999$
Horizon	$h \in \mathbb{N}$ or $h = \infty$	$h = \infty$

Common Assumptions

- Transition model
  - **Markovian**: $\Pr(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \dots) = \Pr(s_{t+1}|s_t, a_t)$
    - Current inventory and order sufficient to predict future inventory
  - **Stationary**: $\Pr(s_{t+1}|s_t, a_t)$ is same for all t
    - Distribution of demand same every day
- Reward model
  - **Stationary**: $R(s_t, a_t) = \sum_t r_t \Pr(r_t|s_t, a_t)$ is same for all t
    - Formula to compute profits is same every day
  - **Exception:** terminal reward is often different
    - In a game: 0 reward at each step and
      +1/-1 reward at the end for winning/losing

Discounted/Average Rewards

- Goal: maximize total rewards $\sum_{t=0}^{h} R(s_t, a_t)$
  Problem: if $h = \infty$, then $\sum_{t=0}^{h} R(s_t, a_t)$ may be infinite

- Solution 1: discounted rewards
  - Discount factor: $0 \leq \gamma < 1$
  - Finite utility: $\sum_t \gamma^t R(s_t, a_t)$ is a geometric sum
  - γ induces an inflation rate of $1/\gamma - 1$ (prefer utility sooner than later)

- Solution 2: average rewards
  - More complicated computationally (beyond scope of this course)

Policy

- Choice of action at each time step
- Formally:
  - Mapping from states to actions: $\pi(s_t) = a_t$
  - Assumption: fully observable states
    - Allows a_t to be chosen only based on current state s_t

Policy Optimization

- **Policy evaluation:** compute expected utility

$$V^{\pi}(s_0) = \sum_{t=0}^{h} \gamma^t \sum_{s_{t+1}} \Pr(s_{t+1}|s_0,\pi)\, R(s_{t+1},\pi(s_{t+1}))$$

- **Optimal policy π^*:** policy with highest expected utility

$$V^{\pi^*}(s_0) \geq V^{\pi}(s_0) \;\; \forall \pi$$

- Several classes of algorithms:
  - Value iteration
  - Policy iteration
  - Linear Programming
  - Search techniques

Value Iteration

- Value when no time left:

$$V_0^*(s_h) = \max_{a_h} R(s_h, a_h)$$

- Value with one time step left:

$$V_1^*(s_{h-1}) = \max_{a_{h-1}} R(s_{h-1}, a_{h-1}) + \gamma \sum_{s_h} \Pr(s_h | s_{h-1}, a_{h-1})\, V_0^*(s_h)$$

- Value with two time steps left:

$$V_2^*(s_{h-2}) = \max_{a_{h-2}} R(s_{h-2}, a_{h-2}) + \gamma \sum_{s_{h-1}} \Pr(s_{h-1} | s_{h-2}, a_{h-2})\, V_1^*(s_{h-1})$$

- …
- Bellman's equation:

$$V_\infty^*(s_t) = \max_{a_t} R(s_t, a_t) + \gamma \sum_{s_{t+1}} \Pr(s_{t+1} | s_t, a_t)\, V_\infty^*(s_{t+1})$$

$$a_t^* = \operatorname*{argmax}_{a_t} R(s_t, a_t) + \gamma \sum_{s_{t+1}} \Pr(s_{t+1} | s_t, a_t)\, V_\infty^*(s_{t+1})$$

Value Iteration

valueIteration(MDP)

$V_0^*(s) \leftarrow \max_a R(s,a) \ \ \forall s$

For $n = 1$ to h do

$V_n^*(s) \leftarrow \max_a R(s,a) + \gamma \sum_{s'} \Pr(s'|s,a)\, V_{n-1}^*(s') \ \ \forall s$

Return V^*

Optimal policy π^*

$n = 0$: $\pi_0^*(s) \leftarrow \operatorname{argmax}_a R(s,a) \ \forall s$

$n > 0$: $\pi_n^*(s) \leftarrow \operatorname{argmax}_a R(s,a) + \gamma \sum_{s'} \Pr(s'|s,a)\, V_{n-1}^*(s') \ \ \forall s$

NB: π^* is non-stationary (i.e., time dependent)

Value Iteration (Matrix Form)

R^a: $|S| \times 1$ column vector of rewards for a

V_n^*: $|S| \times 1$ column vector of state values

T^a: $|S| \times |S|$ matrix of transition probabilities for a

valueIteration(MDP)

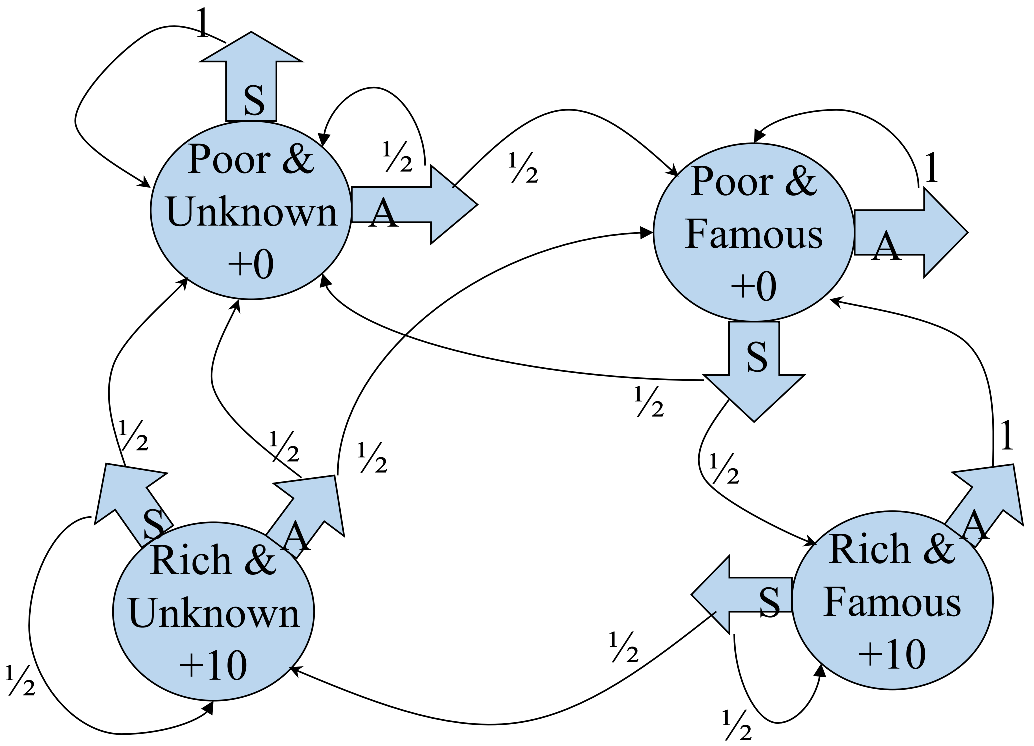
$V_0^* \leftarrow \max_a R^a$

For $t = 1$ to h do

$V_n^* \leftarrow \max_a R^a + \gamma T^a V_{n-1}^*$

Return V^*

A Markov Decision Process

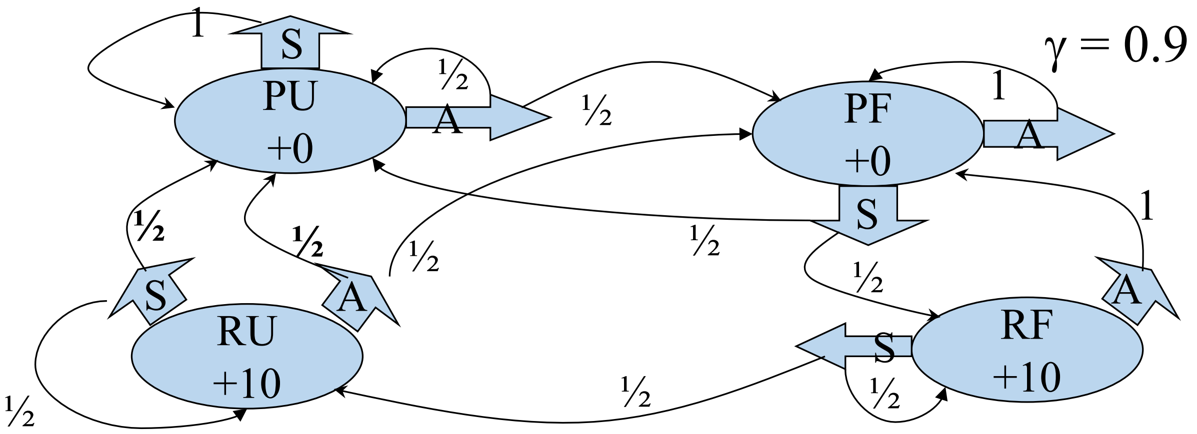


$\gamma = 0.9$

You own a company

In every state you must choose between

Saving money or **A**dvertising



n	$V^*_n(PU)$	$\pi^*_n(PU)$	$V^*_n(PF)$	$\pi^*_n(PF)$	$V^*_n(RU)$	$\pi^*_n(RU)$	$V^*_n(RF)$	$\pi^*_n(RF)$
0	0	A,S	0	A,S	10	A,S	10	A,S
1	0	A,S	4.5	S	14.5	S	19	S
2	2.03	A	8.55	S	16.53	S	25.08	S
3	4.76	A	12.20	S	18.35	S	28.72	S
4	7.63	A	15.07	S	20.40	S	31.18	S
5	10.21	A	17.46	S	22.61	S	33.21	S

Exercise: Value Iteration, No Time Left (RF State)

$$V_0(RF) = \max\{R(RF,A), R(RF,S)\}$$
$$= \max\{10, 10\}$$
$$= 10$$

$$\pi_0(RF) = \text{argmax}\{R(RF,A), R(RF,S)\}$$
$$= \{A, S\}$$

Exercise: Value Iteration, One Time Step Left (RF State)

$$V_1(RF) = \max_a R(RF, a) + \gamma \sum_{s'} P(s' | RF, a) V_0(s')$$

$$= \max \{10 + 0.9 \times 1 \times 0, \quad 10 + 0.9(0.5 \times 10 + 0.5 \times 10)\}$$

$$= \max \{10, 19\}$$

$$= \underline{19}$$

$$\pi_1(RF) = \operatorname{argmax} \{10, 19\}$$

$$= S$$

Horizon Effect

- Finite h:
  - Non-stationary optimal policy
  - Best action different at each time step
  - Intuition: best action varies with the amount of time left

- Infinite h:
  - Stationary optimal policy
  - Same best action at each time step
  - Intuition: same (infinite) amount of time left at each time step
  - **Problem**: value iteration does infinite # of iterations

Infinite Horizon

- Assuming a discount factor γ, after n time steps, rewards are scaled down by γ^n
- For large enough n, rewards become insignificant since $\gamma^n \to 0$

- Solution #1:
  - pick large enough n and run value iteration for n steps
  - Execute policy π_n found at the n^{th} iteration

$$||V_n - V_{n-1}||_\infty = \max_s |V_n(s) - V_{n-1}(s)|$$

- Solution #2:
  - Continue iterating until $||V_n - V_{n-1}||_\infty \leq \epsilon$ (ϵ is called tolerance)
  - Execute policy π_n found at the n^{th} iteration

UNIVERSITY OF WATERLOO