Artificial Intelligence

CS 165A

May 30, 2023

Instructor: Prof. Yu-Xiang Wang

Today

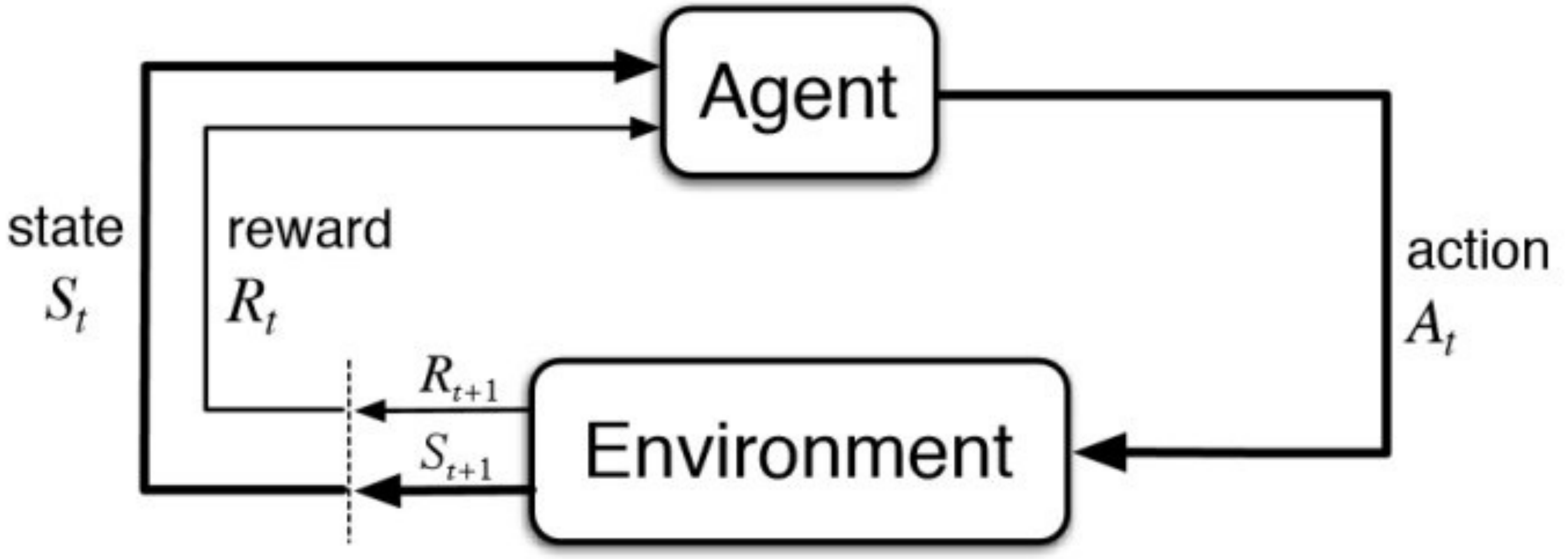
→ Reinforcement Learning

Notes

- Optional HW4 on the website. Discussion this Wednesday.
- Do the quiz on the Ed Stem discussion.
- Don’t wait until the last week for Project 3.
  - You will have everything you need by today for all problems.

Recap: reinforcement learning agents have "online" access to an environment

- State, Action, Reward
- Unknown reward function, unknown state-transitions.
- Agents can "act" and "experiment", rather than only doing offline planning.



Recap: Reinforcement learning

- Differences from MDP inference
  - Unknown transition probabilities
  - Unknown reward function

- Differences from multi-armed bandits / contextual bandits
  - State not fixed / i.i.d., but depends on the action
  - Need planning / dynamic programming

Recap: Three ideas for solving RL

- Idea 1: Model-based approach
  - Estimated the CPTs of MDP by their empirical frequency
  - Plug-in the estimate to Bellman equations for VI / PI
- Idea 2: Model-free approach: Directly estimate V function and Q function
  - Monte Carlo: Run many episodes of the MDP
  - First-visit MC: first time you visit State s, keep all subsequent rewards, then average over many such episodes
- Idea 3: Better model-free approach: Combining MC with VI / PI directly.
  - Temporal difference (TD) learning

Recap: DP + MC = Temporal Difference Learning

- Monte Carlo

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[G_t - V(S_t)\Big],$$

Issue: G_t can only be obtained after the entire episode!

- The idea of TD learning:

$$\mathbb{E}_\pi[G_t] = \mathbb{E}_\pi[R_t|S_t] + \gamma V^\pi(S_{t+1})$$

We only need one step before we can plug-in and estimate the RHS!

- TD-Policy evaluation

Bootstrapping!

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\Big]$$

Recap: TD policy optimization (TD-control)

- SARSA (On-Policy TD-control)
  - Update the Q function by bootstrapping Bellman Equation

$$Q(S, A) \leftarrow Q(S, A) + \alpha \big[R + \gamma Q(S', A') - Q(S, A) \big]$$

  - Choose the next A' using Q, e.g., eps-greedy.

- Q-Learning (Off-policy TD-control)
  - Update the Q function by bootstrapping Bellman Optimality Eq.

$$Q(S, A) \leftarrow Q(S, A) + \alpha \big[R + \gamma \max_a Q(S', a) - Q(S, A) \big]$$

  - Choose the next A' using Q, e.g., eps-greedy, or any other policy.

Remarks:

- These are **proven to converge** asymptotically.
- Much more data-efficient in practice, than MC.
- Regret analysis is still active area of research.

Recap: Advantage of TD over Monte Carlo

- Given a trajectory, a roll-out, of T steps.
  - MC updates the Q function only once
  - TD updates the Q function (and the policy) T times!

Remark: This is the same kind of improvement from Gradient Descent to Stochastic Gradient Descent (SGD).

(Optional reading: Sutton and Barton 9.3 Semi-gradient methods)

This lecture

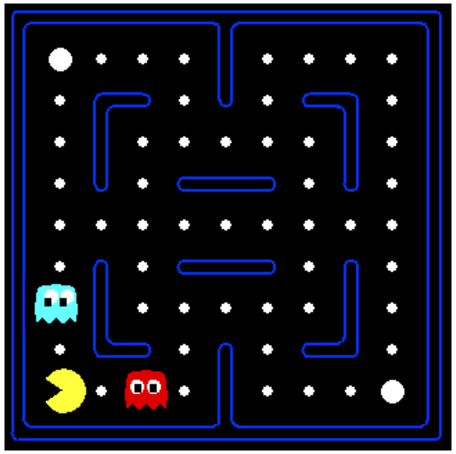
- Features and linear function approximation
- Policy Gradient method (very brief)
- Intro to Logical Agents

The problem of large state-space is still there

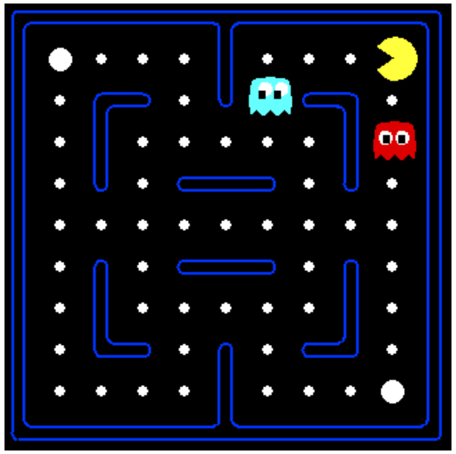
- We need to represent and learn SA parameters in Q-learning and SARSA.
- S is often large
  - 9-puzzle, Tic-Tac-Toe: 9! = 362,800, S^2 = 1.3*10^11
  - PACMAN with 20 by 20 grid. S = O(2^400), S^2 = O(2^800)
- O(S) is not acceptable in some cases.
- Need to think of ways to "generalize"/share information across states.

Example: Pacman

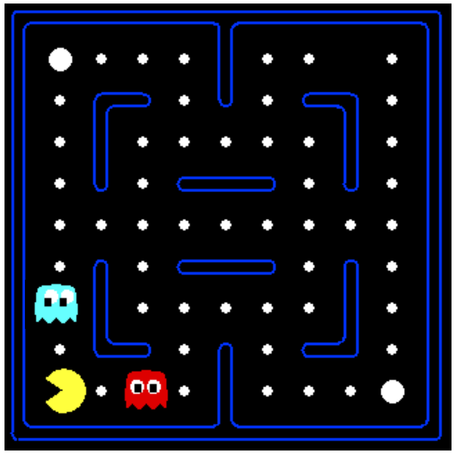
Let’s say we discover through experience that this state is bad:



In naïve q-learning, we know nothing about this state:



Or even this one!



(From Dan Klein and Pieter Abbeel)

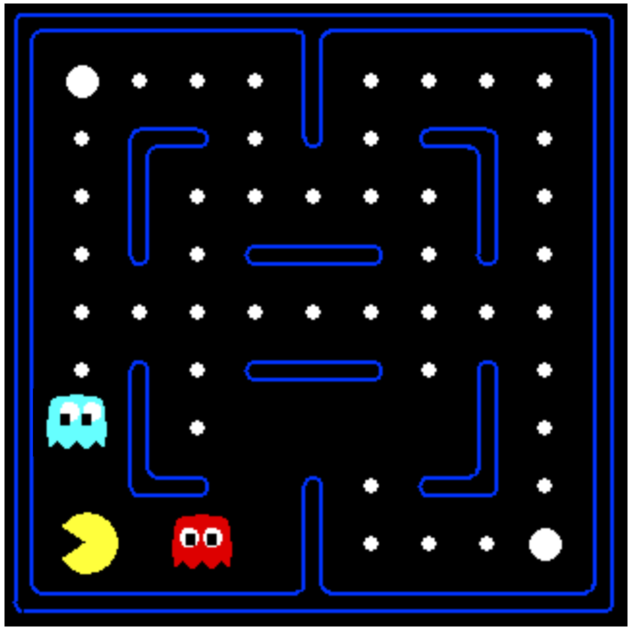
Video of Demo Q-Learning Pacman – Tiny – Watch All

Video of Demo Q-Learning Pacman – Tiny – Silent Train

Video of Demo Q-Learning Pacman – Tricky – Watch All

Why not use an evaluation function? A Feature-Based Representations

- Solution: describe a state using a vector of features (properties)
  - Features are functions from states to real numbers (often 0/1) that capture important properties of the state
  - Example features:
    - Distance to closest ghost
    - Distance to closest dot
    - Number of ghosts
    - 1 / (dist to dot)^2
    - Is Pacman in a tunnel? (0/1)
    - …… etc.
    - Is it the exact state on this slide?
  - Can also describe a q-state (s, a) with features (e.g. action moves closer to food)



Linear Value Functions

- Using a feature representation, we can write a q function (or value function) for any state using a few weights:
  - $V_{\mathbf{w}}(s) = w_1f_1(s) + w_2f_2(s) + \ldots + w_nf_n(s)$
  - $Q_{\mathbf{w}}(s,a) = w_1f_1(s,a) + w_2f_2(s,a) + \ldots + w_nf_n(s,a)$
- Advantage: our experience is summed up in a few powerful numbers
- Disadvantage: states may share features but actually be very different in value!

Updating a linear value function

- Original Q learning rule tries to reduce prediction error at s, a:

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

- Instead, we update the weights to try to reduce the error at s, a:

$$w_i \leftarrow w_i + \alpha \cdot [R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)] \, \partial Q_{\mathbf{w}}(s,a)/\partial w_i$$
$$= w_i + \alpha \cdot [R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)] \, f_i(s,a)$$

Updating a linear value function

- Original Q learning rule tries to reduce prediction error at s, a:

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

- Instead, we update the weights to try to reduce the error at s, a:

$$w_i \leftarrow w_i + \alpha \cdot [R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)] \, \partial Q_{\mathbf{w}}(s,a)/\partial w_i$$
$$= w_i + \alpha \cdot [R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)] \, f_i(s,a)$$

- Qualitative justification:
  - Pleasant surprise: increase weights on positive features, decrease on negative ones
  - Unpleasant surprise: decrease weights on positive features, increase on negative ones

Q-Learning with function approximation

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

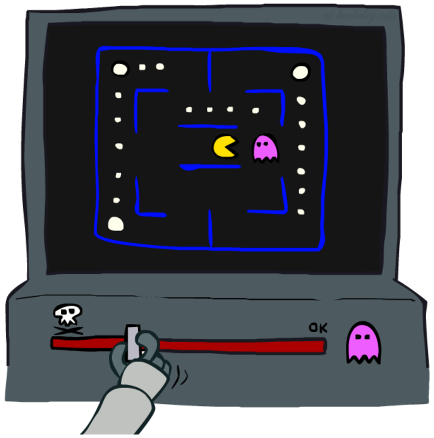
- Q-learning with linear Q-functions:

$$\text{transition } = (s, a, r, s')$$

$$\text{difference} = \left[r + \gamma \max_{a'} Q(s', a')\right] - Q(s,a)$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha\,[\text{difference}] \qquad \text{Exact Q's}$$

$$w_i \leftarrow w_i + \alpha\,[\text{difference}]\, f_i(s,a) \qquad \text{Approximate Q's}$$



- Intuitive interpretation:
  - Adjust weights of active features
  - E.g., if something unexpectedly bad happens, blame the features that were on: disprefer all states with that state's features
- Formal justification: online least squares (Read the textbook!)

PACMAN Q-Learning (Linear function approx.)

So far, in RL algorithms

- Model-based approaches
  - Estimate the MDP parameters.
  - Then use policy-iterations, value iterations.
- Monte Carlo methods:
  - estimating the rewards by empirical averages
- Temporal Difference methods:
  - Combine Monte Carlo methods with Dynamic Programming
- Linear function approximation in Q-learning
  - Similar to SGD
  - Learning heuristic function

*Question: What is the policy class Π of interest in these methods?

Policy gradient

- Let's not worry about states, dynamics, Q function.
  - We might not even observe the true state.
  - Let's specify a class of parametrized policy and hope to compare to the best within this class

- Objective function to maximize: $J(\boldsymbol{\theta}) \doteq v_{\pi_{\boldsymbol{\theta}}}(s_0),$

- Do SGD: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \widehat{\nabla J(\boldsymbol{\theta}_t)},$

- Policy gradient theorem:

$$\nabla J(\boldsymbol{\theta}) = \sum_s d^\pi(s) \sum_a Q^\pi(s,a) \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta})$$

*Note how this theorem is non-trivial… The first two terms depends on π, but we did not take the gradient w.r.t. them.

Stochastic approximation in policy gradients

$$\nabla J(\boldsymbol{\theta}) = \sum_s d^\pi(s) \sum_a Q^\pi(s,a) \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta})$$

- Sample from running policy π

$$(S_1, A_1, R_1), (S_2, A_2, R_2), \ldots, (S_T, A_T, R_T)$$

- Idea: Sample s, then the following is an unbiased estimator (finite horizon episodic case)

$$\sum_{t=1}^{T} \Big(\sum_{\ell=t}^{T} R_\ell \Big) \frac{\nabla_\theta \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}$$
$$= \sum_{t=1}^{T} G_t \nabla_\theta \log(\pi(A_t|S_t,\theta))$$

***Show that this is an unbiased estimator of the gradient.**

The REINFORCE algorithm (Williams, 1987)

REINFORCE, A Monte-Carlo Policy-Gradient Method (episodic)

```
Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$
Repeat forever:
    Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
    For each step of the episode $t = 0, \ldots, T-1$:
        $G \leftarrow$ return from step $t$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\gamma^t G \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t,\boldsymbol{\theta})$
```

- From Sutton and Barto Ch .13.
- Note the γ^t term. This is for the discounted (episodic) case
- Updating the parameter T times for each episode!
- Easy to implement easy to understand from SGD theory.

Elements of State-of-the-Art Reinforcement Learning

- Use a deep neural network to parameterize Q-function
- Use a deep neural network to parameterize the policy \pi
- Run a combination of Q-learning and Policy Gradient.
  - Actor-Critics, A3C, etc…
- Heuristic-based exploration: curiosity, reward shaping, etc..
- Experience replay to generate more data from existing data.
- Multi-agent RL: modeling your opponents

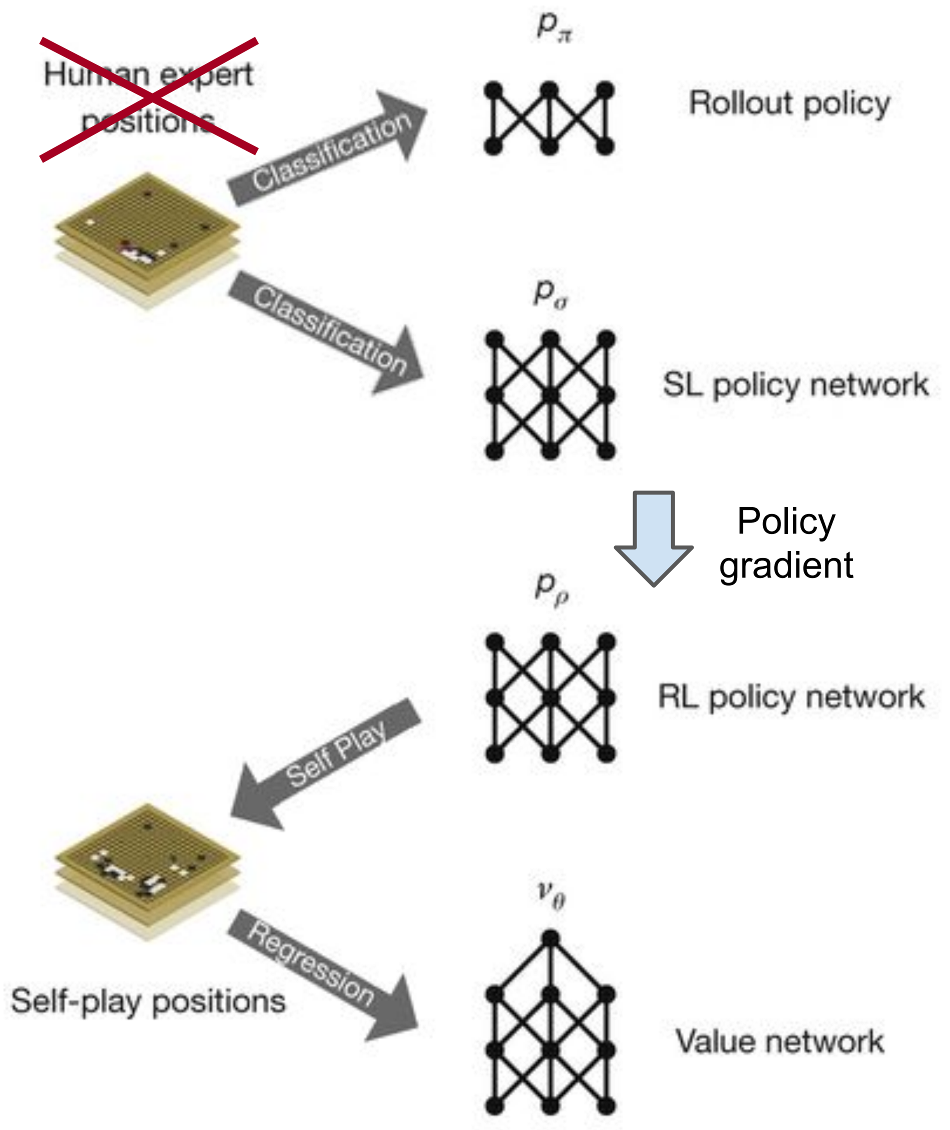
Example of State-of-the-Art RL for Hide-n-Seek

Lowe, Wu et al. (2017)
https://proceedings.neurips.cc/paper/2017/hash/68a9750337a418a86fe06c1991a1d64c-Abstract.html

Alpha-Go and Alpha-Zero

- Parameterize the policy networks with CNN
- ~~Supervised learning initialization~~
- RL using Policy gradient
- Fit Value Network (This is a heuristic function!)
- Monte-Carlo Tree Search

https://www.youtube.com/watch?v=4D5yGiYe8p4



D. Silver et al. Mastering the game of Go with Deep Neural Networks and Tree Search. Nature, vol. 529 issue 7587

D. Silver, et al. "Mastering the game of go without human knowledge." *Nature* 550.7676 (2017): 354-359.

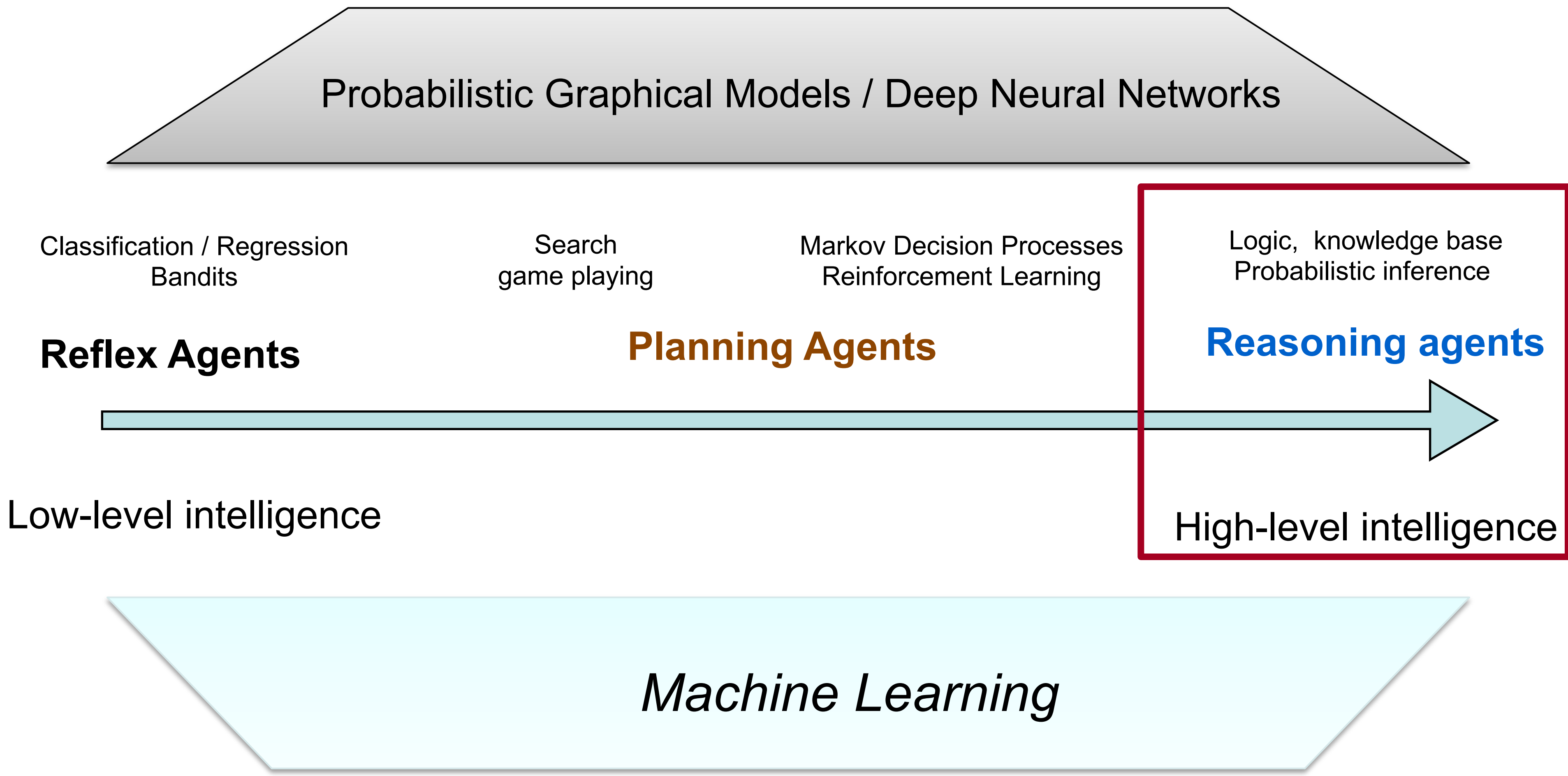
Summary of RL algorithms

- Model-based:
  - Policy iteration / Value iteration
  - Need to estimate the dynamics (MDP parameters)
- Model-free: (no need to "explicitly" estimate dynamics)
  - TD learning: SARSA, Q-learning
  - Function approximation (Share information across states)
- Absolutely model-free (do not even need an MDP model)
  - Policy gradient

Remainder of today’s lecture

- Start Logical Agent
- Logical inference for propositional logic
- What you should do:
  - Read Chapter 7 of AIMA textbook.
  - Start working on Project 3 if you haven’t yet.

High-level intelligence and logical inference

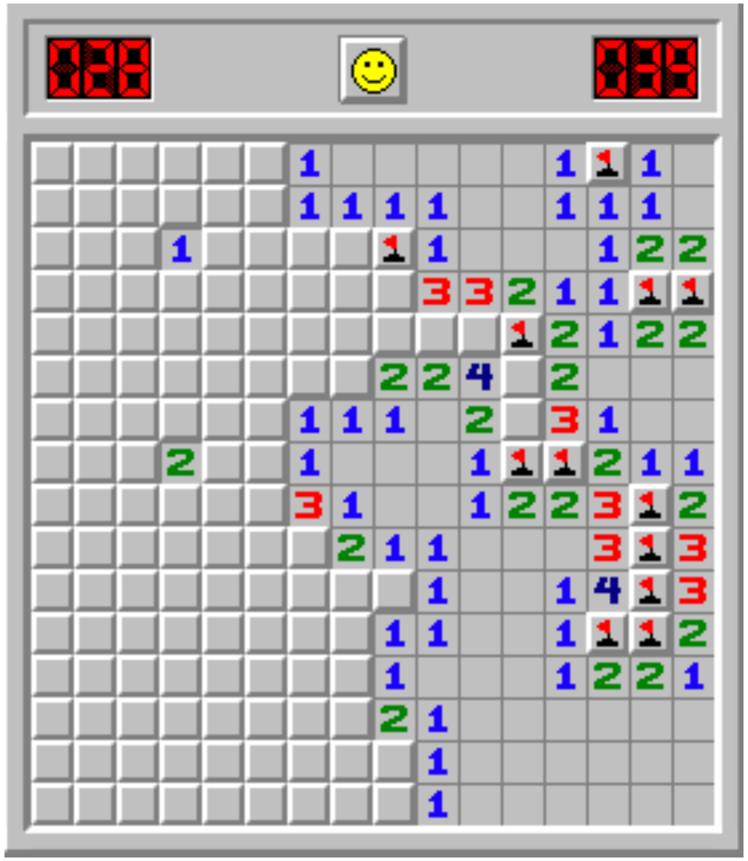


The final lecture series on “logic”

- So far:
  - Reflex agents (classifiers)
  - Problem solving / planning / game solving agents (Search)
  - Planning meets utility-maximizing agents (MDPs)
- They can:
  - Quantify uncertainty
  - Make rational decisions
  - Learn from experience
- What’s missing?
  - Knowledge, reasoning, logical deduction
  - (Arguably PGM does a bit of this, but our focus was to use PGM for modeling the world...)

Why do we care?

- Minesweeper



- Imagine how you would solve this?
- Imagine how an RL agent would solve this?

Knowledge Base:

- Encode the rules.
- Encode the observations so far.

What does a knowledge base do?

- **TELL** operation: add evidence.
- **ASK** operation: check if a tile has a mine under it, or not, or undetermined.

Knowledge and reasoning

- We want powerful methods for
  - Representing *Knowledge* – general methods for representing facts about the world and how to act in world
  - Carrying out *Reasoning* – general methods for deducing additional information and a course of action to achieve goals
  - Focus on *knowledge and reasoning* rather than *states and search*
    - Note that *search* is still critical

- This brings us to the idea of ***logic,*** but….
  - How to define logic formally?
  - How to represent / manipulate knowledge / inference at scale?
  - How to systematically use knowledge / inference by an agent?
  - What are the strengths and limitations of logical agents?

Example

- A certain country is inhabited by people who always tell the truth or always tell lies and who will respond only to yes/no questions.

  A tourist comes to a fork in the road where one branch leads to a restaurant and one does not.

  No sign indicating which branch to take, but there is an inhabitant Mr. X standing on the road.

  With a single yes/no question, can the hungry tourist ask to find the way to the restaurant?

Example (cont.)

- Answer: Is exactly one of the following true:
  1. you always tell the truth
  2. the restaurant is to the left

- Truth Table:

X is truth teller;	restaurant is to left;	response
true;	true;	no
true;	false;	yes
false;	true;	no
false;	false;	yes

Another Example (1 min discussion)

- Bob looks at Alice. Alice looks at George.
  Bob is married. George is unmarried.
  Does a married person ever look at an unmarried one;
  yes, no, cannot be determined?

Another Example (cont.)

- Amarried or ~Amarried
  BlooksA and AlooksG

  BlooksA ^ AlooksG =
  BlooksA ^ AlooksG ^ Amarried or BlooksA ^ AlooksG ^ ~Amarried

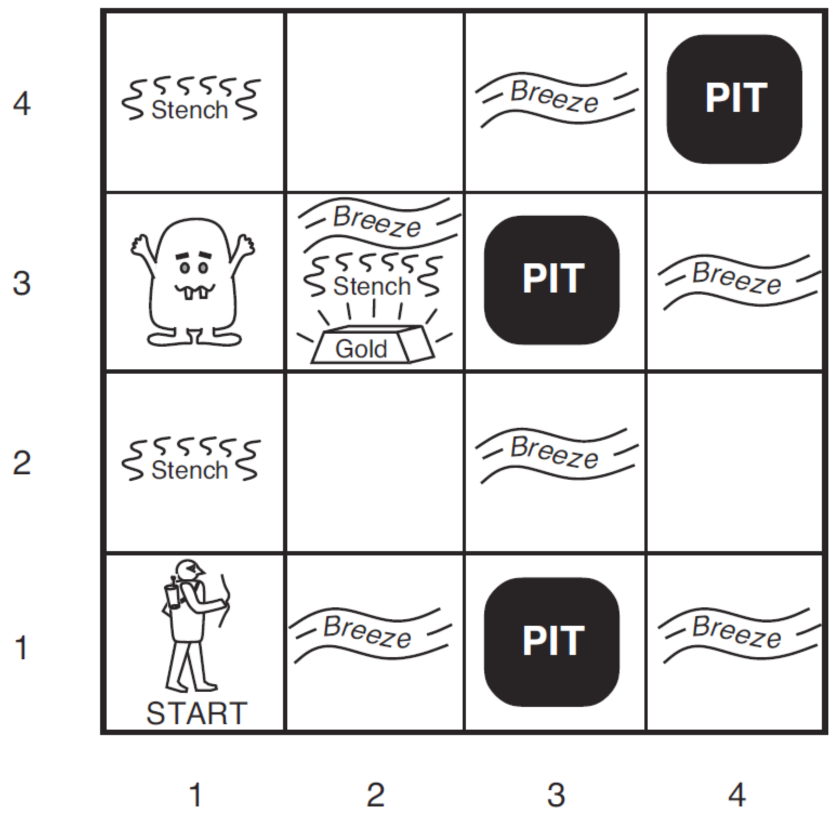
- Case 1: Amarried = true, then BlooksA ^ AlooksG ^ Amarried satisfies conclusion

  Case 2: Amarried = false, then BlooksA ^ AlooksG ^ ~Amarried satisfies conclusion

Wumpus World

- Logical Reasoning as a searc
  problem

  - B_{ij} = breeze felt
  - S_{ij} = stench smelt
  - P_{ij} = pit here
  - W_{ij} = wumpus here
  - G = gold



http://thiagodnf.github.io/wumpus-world-simulator/

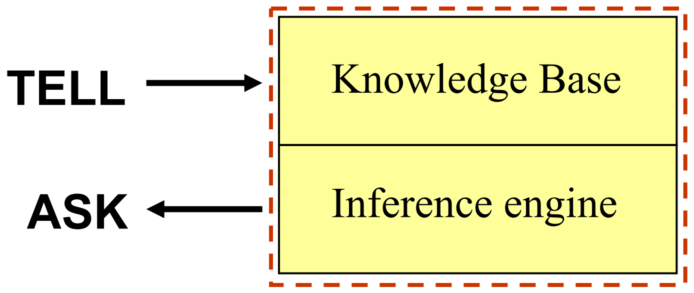
*The agent can only observe blocks that she has visited.

*Cannot observe the state directly. So cannot solve offline with search.

Knowledge-based agents

- A *knowledge-based agent* uses reasoning based on **prior** and **acquired** knowledge in order to achieve its goals
- Two important components:
  - Knowledge Base (KB)
    - Represents facts about the world (the agent's environment)
      - Fact = "sentence" in a particular knowledge representation language (KRL)
    - KB = set of sentences in the KRL
  - Inference Engine – determines what follows from the knowledge base (what the knowledge base *entails*)
    - Inference / deduction
      - Process for deriving new sentences from old ones
        - » *Sound* reasoning from facts to conclusions

KB Agents



True sentences

Domain specific content; facts

Domain independent algorithms; can deduce new facts from the KB

```
function KB-AGENT(percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time

    TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action
```

KB Agents need to TELL, ASK with a language and the KB needs to understand.

- How about using natural languages?
  - Example from Lecture 1.

They ate the pie with ice cream.

They ate the pie with rhubarb.

They ate the pie with paper plates.

They ate the pie with cold milk.

They ate the pie with friends.

They ate the pie with dinner.

They ate the pie with enthusiasm.

They ate the pie with spoons.

They ate the pie with napkins.

Ambiguities!!!

from Dr. Douglas Lenatand Dr. Michael Witbrock

Fundamental Concepts of Logical Language Representation and Concepts

- **Syntax**
  - Grammar / rules to follow for form a well-defined sentence
  - $x + y = 4$ is a valid sentence in "arithmetics", x4y+= is not.
- **Semantics**
  - The meaning of sentences. Truth of each sentence w.r.t. each possible world.
  - Possible World 1: x=3, y=1. Possible World 2: x=1, y=1.
- **Model** (Possible world, a.k.a. "interpretations" in some text)
  - Each model is an assignment of values to variables.
  - Each model fixes the truth value of all sentences.
  - If sentence α is true in Model m, we say: Model m satisfies sentence α, or m is a model of α, or $m \in M(\alpha)$,

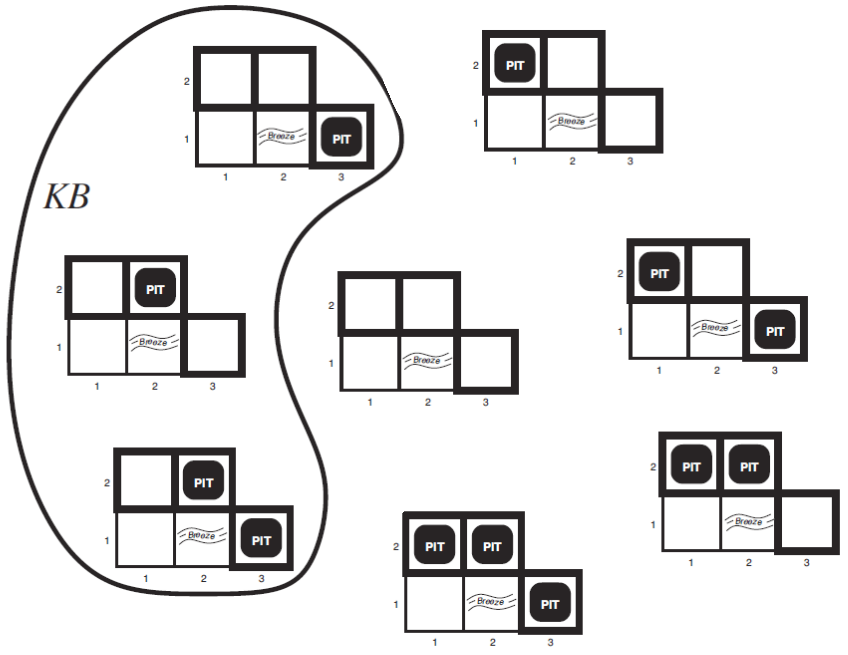
Fundamental Concepts of Logical Language Representation and Concepts

- **Entailment**
  - Sentence β logically follows from Sentence α
  - Denoted by $\alpha \models \beta$
  - α entails β *if an only if* $M(\alpha) \subseteq M(\beta)$
  - If all models of α are also models of β

- **Logical Inference**
  - The procedure of checking whether a sentence is entailed by a given a knowledge base
  - Simplest algorithm for logical inference: **Model checking**
  - Enumerate all models in $\mathrm{M}(\alpha)$, check whether they are in $\mathrm{M}(\beta)$.
  - We will come back to logical inference!

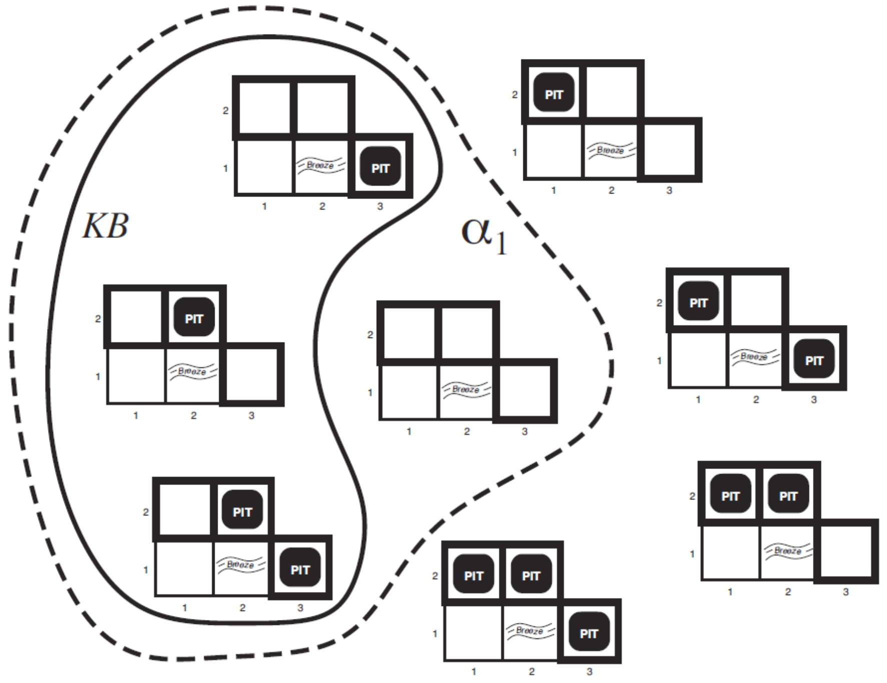
Example: Wumpus World

- Possible Models
  - $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
  - Knowledge base
    - Nothing in [1,1]
    - Breeze in [2,1]



Example: Wumpus World

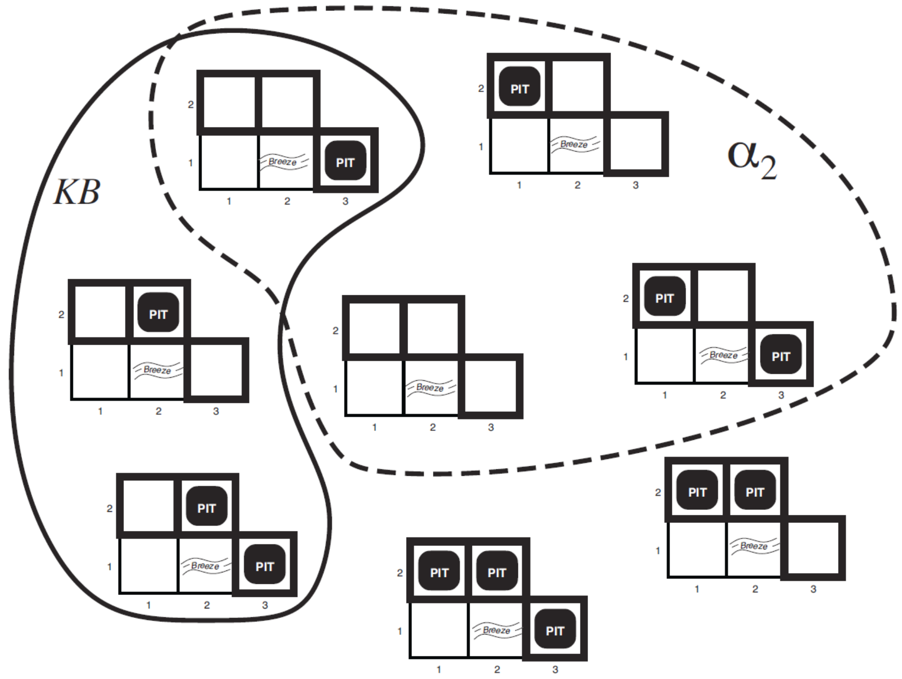
- Possible Models
    - $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
    - Knowledge base
        - Nothing in [1,1]
        - Breeze in [2,1]
    - Query α_1:
        - No pit in [1,2]



*Question: Does KB entails α_1?

Example: Wumpus World

- Possible Models
  - $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
  - Knowledge base
    - Nothing in [1,1]
    - Breeze in [2,1]
  - Query α_2:
    - No pit in [2,2]



*Question: Does KB entails α_2?

Next lecture

- More on logical inference for propositional logic
- First order logic
- Read Chapter 7 and Chapter 8 of AIMA textbook.