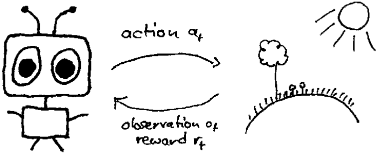
# AIXI Tutorial
# Part II

## Intuitions, Approximations, and the Real World™

John Aslanides and Tom Everitt

July 10, 2018



# Contents

# Why are we here?



- AIXI [1] proposes an answer to the following question:

*What is optimal behavior in general unknown environments?*

# Why are we here?



- AIXI [1] proposes an answer to the following question:

*What is optimal behavior in general unknown environments?*

- In this part we'll give some scaled down examples and conceptual intuitions about what this means.

# Why are we here?



- AIXI [1] proposes an answer to the following question:

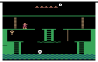
*What is optimal behavior in general unknown environments?*

- In this part we'll give some scaled down examples and conceptual intuitions about what this means.
- These slides can be found at aslanides.io/docs/aixi_tutorial.pdf.

# RL Setting & Notation

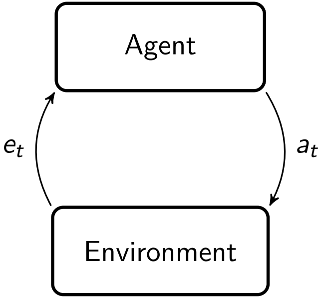


Environment is an **unknown**, **non-ergodic**, **partially observable** MDP.

| Symbol | Description | Example |
|---|---|---|
| $a \in \mathcal{A}$ | Action | $\{\uparrow, \downarrow, \leftarrow, \rightarrow, \ldots\}$, $\mathbb{N}$, ... |
| $o \in \mathcal{O}$ | Observation | $\mathbb{R}^N$, $\mathbb{B}^\star$, , ... |
| $r \in \mathcal{R}$ | Reward | $\mathbb{R}$, $\mathbb{Z}$, ... |
| $e \in \mathcal{E}$ | Percept | $\mathcal{O} \times \mathcal{R}$ (definition) |
| $\mu \in \mathcal{M}$ | Environment | gridworld, robotics, ... |
| $\pi \in \Delta(\mathcal{A})$ | Policy | $\epsilon$-greedy, random, ... |
| $æ_{<t} \in (\mathcal{A} \times \mathcal{E})^\star$ | History | $a_1 o_1 r_1 \ldots a_{t-1} o_{t-1} r_{t-1}$ |

# RL Setting & Notation

Agent and environment interact using the standard RL setup:



# Optimal policy ("Just do the best thing")



- Optimal **state-action value** in environment $\mu$ at time $t$ given history $æ_{<t}$ is given by

$$Q^*_\mu(a_t|æ_{<t}) = \sup_\pi \mathbb{E}_\mu\left[\sum_{k=t}^{\infty} \gamma_k r_k | \pi, æ_{<t} a_t\right]$$

# Optimal policy ("Just do the best thing")



- Optimal **state-action value** in environment $\mu$ at time $t$ given history $æ_{<t}$ is given by

$$Q^*_\mu(a_t|æ_{<t}) = \sup_\pi \mathbb{E}_\mu\left[\sum_{k=t}^{\infty}\gamma_k r_k|\pi, æ_{<t}a_t\right]$$

- Optimal **value**:

$$V^*_\mu\left(æ_{<t}\right) = \max_{a_t\in\mathcal{A}} Q^*_\mu\left(a_t|æ_{<t}\right)$$

# Optimal policy ("Just do the best thing")



- Optimal **state-action value** in environment $\mu$ at time $t$ given history $æ_{<t}$ is given by

$$Q_{\mu}^{*}(a_t|æ_{<t}) = \sup_{\pi} \mathbb{E}_{\mu}\left[\sum_{k=t}^{\infty} \gamma_k r_k | \pi, æ_{<t} a_t\right]$$

- Optimal **value**:

$$V_{\mu}^{*}\left(æ_{<t}\right) = \max_{a_{t \in \mathcal{A}}} Q_{\mu}^{*}\left(a_t|æ_{<t}\right)$$

- Optimal **policy** is greedy, breaking ties at random:

$$\pi_{\mu}^{*}\left(a_t|æ_{<t}\right) = \arg\max_{a} Q_{\mu}^{*}\left(a|æ_{<t}\right)$$

# Optimal value



Optimal **value** in environment $\mu$ at time $t$ given history $æ_{<t}$ is given by

$$V_\mu^*(æ_{<t}) = \lim_{m\to\infty} \max_{a_t} \sum_{e_t} \cdots \max_{a_m} \sum_{e_m} \sum_{k=t}^{t+m} \gamma_k r_k \prod_{j=t}^{k} \mu\left(e_j | æ_{<j} a_j\right).$$

- Likelihood of percepts $e_{t:k}$ given action sequence $a_{1:k}$.

# Optimal value

AIXI Tutorial Part II

John Aslanides and Tom Everitt

Short Recap

Approximations

(Break)

Variants of AIXI

Optimal **value** in environment $\mu$ at time $t$ given history $æ_{<t}$ is given by

$$V^*_\mu(æ_{<t}) = \lim_{m\to\infty} \max_{a_t} \sum_{e_t} \cdots \max_{a_m} \sum_{e_m} \sum_{k=t}^{t+m} \gamma_k r_k \prod_{j=t}^{k} \mu\left(e_j | æ_{<j} a_j\right).$$

- Likelihood of percepts $e_{t:k}$ given action sequence $a_{1:k}$.
- Discounted return realized by the trajectory $e_{t:t+m}$.

# Optimal value

Optimal **value** in environment $\mu$ at time $t$ given history $æ_{<t}$ is given by

$$V^*_\mu(æ_{<t}) = \lim_{m\to\infty} \max_{a_t} \sum_{e_t} \cdots \max_{a_m} \sum_{e_m} \sum_{k=t}^{t+m} \gamma_k r_k \prod_{j=t}^{k} \mu\left(e_j | æ_{<j} a_j\right).$$

- Likelihood of percepts $e_{t:k}$ given action sequence $a_{1:k}$.
- Discounted return realized by the trajectory $e_{t:t+m}$.
- Expectimax up to horizon $m$.

# Optimal value



Optimal **value** *up to horizon m*:

$$V^*_{\mu,m}(æ_{<t}) = \max_{a_t} \sum_{e_t} \cdots \max_{a_m} \sum_{e_m} \sum_{k=t}^{t+m} \gamma_k r_k \prod_{j=t}^{k} \mu\left(e_j | æ_{<j} a_j\right).$$

# Optimal value



Optimal **value** *up to horizon m:*

$$V^*_{\mu,m}(æ_{<t}) = \underbrace{\max_{a_t}\sum_{e_t}\cdots\max_{a_m}\sum_{e_m}\sum_{k=t}^{t+m}\gamma_k r_k}_{\text{"Planning"}}\underbrace{\prod_{j=t}^{k}\mu\left(e_j|æ_{<j}a_j\right)}_{\text{"Learning"}}.$$

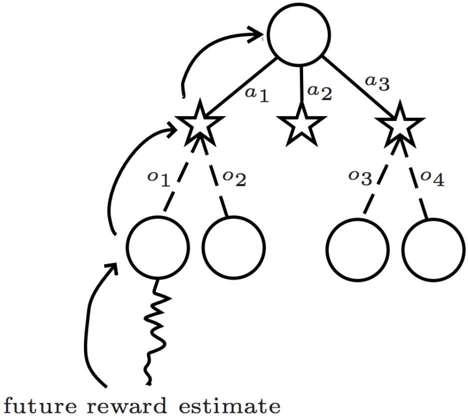
# Planning

- We can approximate the expectimax computation of $V^*_{\mu,m}$ with a variant of **Monte-Carlo Tree Search (MCTS)**.
- Example use: playing Chess, Go, Shogi (**AlphaZero**) [2].

# Planning



- We can approximate the expectimax computation of $V^*_{\mu,m}$ with a variant of **Monte-Carlo Tree Search (MCTS)**.
- Example use: playing Chess, Go, Shogi (**AlphaZero**) [2].



# Planning



- Algorithm: $\boldsymbol{\rho}$**UCT** [3], an extension of **UCT** [4] to histories.

# Planning

- Algorithm: $\rho$**UCT** [3], an extension of **UCT** [4] to histories.
- Idea: Only expand subtrees that show promising rewards and/or high uncertainty.

# Planning



- Algorithm: $\rho$**UCT** [3], an extension of **UCT** [4] to histories.
- Idea: Only expand subtrees that show promising rewards and/or high uncertainty.
- Trade off reward with uncertainty using a tree-based variant of the **UCB** algorithm [5]:

$$a_{\mathrm{UCT}} \in \arg\max_{a \in \mathcal{A}} \left( \underbrace{\hat{Q}\left(a | æ_{<t}\right)}_{\text{Value estimate}} + \underbrace{C \sqrt{\frac{\log T\left(æ_{<t}\right)}{T\left(æ_{<t} a\right)}}}_{\text{Exploration bonus}} \right),$$

where $T\left(\cdot\right)$ is the number of times a sequence has been visited.

# Learning



- Agent doesn't know $\mu$ *a priori*.

# Learning



- Agent doesn't know $\mu$ *a priori*.
- Recall the incomputable Solomonoff model class

$$M\left(e_{<t}|a_{<t}\right) = \sum_{p} 2^{-\ell(p)} \left[\!\left[ p\left(a_{<t}\right) = e_{<t} \right]\!\right]$$

# Learning

- Agent doesn't know $\mu$ *a priori*.
- Recall the incomputable Solomonoff model class

$$M\left(e_{<t}|a_{<t}\right)=\sum_{p}2^{-\ell(p)}\left[\!\left[p\left(a_{<t}\right)=e_{<t}\right]\!\right]$$

- Introduce a finite model class $\mathcal{M}$:

$$\xi\left(e_t|æ_{<t}a_t\right)=\sum_{\nu\in\mathcal{M}}w_\nu\nu\left(e_t|æ_{<t}a_t\right)$$

# Learning



- Agent doesn't know $\mu$ *a priori.*
- Recall the incomputable Solomonoff model class

$$M\left(e_{<t} | a_{<t}\right) = \sum_{p} 2^{-\ell(p)} \left[\!\left[ p\left(a_{<t}\right) = e_{<t} \right]\!\right]$$

- Introduce a finite model class $\mathcal{M}$:

$$\xi\left(e_t | æ_{<t} a_t\right) = \sum_{\nu \in \mathcal{M}} w_\nu \nu\left(e_t | æ_{<t} a_t\right)$$

- Update posterior $w_\nu$ with Bayes rule:

$$w_\nu \leftarrow \frac{\nu\left(e_t\right)}{\xi\left(e_t\right)} w_\nu \ \forall \nu \in \mathcal{M}$$

# Learning



- Agent doesn't know $\mu$ *a priori*.
- Recall the incomputable Solomonoff model class

$$M\left(e_{<t}|a_{<t}\right) = \sum_p 2^{-\ell(p)} \left[\!\left[ p\left(a_{<t}\right) = e_{<t} \right]\!\right]$$

- Introduce a finite model class $\mathcal{M}$:

$$\xi\left(e_t|æ_{<t}a_t\right) = \sum_{\nu\in\mathcal{M}} w_\nu \nu\left(e_t|æ_{<t}a_t\right)$$

- Update posterior $w_\nu$ with Bayes rule:

$$w_\nu \leftarrow \frac{\nu\left(e_t\right)}{\xi\left(e_t\right)} w_\nu \ \forall \nu \in \mathcal{M}$$

- For very small $\mathcal{M}$ we can compute this exactly.

# Learning



- Agent doesn't know $\mu$ *a priori*.
- Recall the incomputable Solomonoff model class

$$M\left(e_{<t}|a_{<t}\right) = \sum_{p} 2^{-\ell(p)} \left[\!\left[ p\left(a_{<t}\right) = e_{<t} \right]\!\right]$$

- Introduce a finite model class $\mathcal{M}$:

$$\xi\left(e_t|æ_{<t}a_t\right) = \sum_{\nu \in \mathcal{M}} w_\nu \nu\left(e_t|æ_{<t}a_t\right)$$

- Update posterior $w_\nu$ with Bayes rule:

$$w_\nu \leftarrow \frac{\nu\left(e_t\right)}{\xi\left(e_t\right)} w_\nu \ \forall \nu \in \mathcal{M}$$

- For very small $\mathcal{M}$ we can compute this exactly.
- Let's look at this with some toy examples.

# Gridworld example



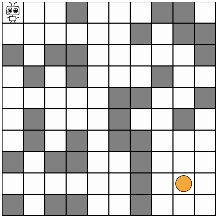
Consider a class of **gridworlds**:

# Gridworld example



Consider a class of **gridworlds**:

- The world is a procedurally generated $N \times N$ maze:

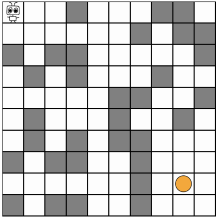


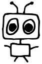
# Gridworld example



Consider a class of **gridworlds**:

- The world is a procedurally generated $N \times N$ maze:



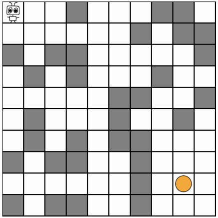
- The agent is a robot  with $\mathcal{A} = \{\leftarrow, \rightarrow, \uparrow, \downarrow, \emptyset\}$.

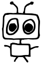
# Gridworld example



Consider a class of **gridworlds**:

- The world is a procedurally generated $N \times N$ maze:

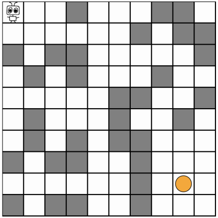


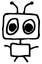
- The agent is a robot  with $\mathcal{A} = \{\leftarrow, \rightarrow, \uparrow, \downarrow, \emptyset\}$.
- The grey tiles  are walls that yield $-5$ reward if hit.

# Gridworld example

Consider a class of **gridworlds**:

- The world is a procedurally generated $N \times N$ maze:



- The agent is a robot  with $\mathcal{A} = \{\leftarrow, \rightarrow, \uparrow, \downarrow, \emptyset\}$.
- The grey tiles  are walls that yield $-5$ reward if hit.
- The white tiles  are empty, but moving costs $-1$.

# Gridworld example

- The orange circle  looks like an empty tile, but randomly dispenses $+100$ each step with some fixed probability $\theta$.

# Gridworld example



- The orange circle  looks like an empty tile, but randomly dispenses $+100$ each step with some fixed probability $\theta$.
- The agent has $\mathcal{O}\left(N^2\right)$ steps to live.

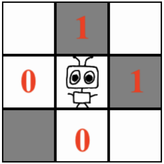
# Gridworld example



- The orange circle  looks like an empty tile, but randomly dispenses $+100$ each step with some fixed probability $\theta$.
- The agent has $\mathcal{O}\left(N^2\right)$ steps to live.
  - e.g. 200 steps on $10 \times 10$ grid.

# Gridworld example

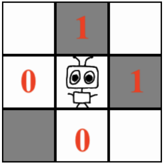
- The orange circle  looks like an empty tile, but randomly dispenses $+100$ each step with some fixed probability $\theta$.
- The agent has $\mathcal{O}\left(N^2\right)$ steps to live.
  - e.g. 200 steps on $10 \times 10$ grid.
- The observations consist of just **four bits**, $\mathcal{O} = \mathbb{B}^4$:



# Gridworld example



- The orange circle  looks like an empty tile, but randomly dispenses $+100$ each step with some fixed probability $\theta$.
- The agent has $\mathcal{O}\left(N^2\right)$ steps to live.
  - e.g. 200 steps on $10 \times 10$ grid.
- The observations consist of just **four bits**, $\mathcal{O} = \mathbb{B}^4$:




- This is a **stochastic** & **partially observable** environment with **simple** & **easy-to-understand** dynamics [3].

# Simple model class

- Let the agent **know**:

# Simple model class



- Let the agent **know**:
  - Maze layout

# Simple model class



- Let the agent **know**:
  - Maze layout
  - Dispenser probability $\theta$

# Simple model class



- Let the agent **know**:
  - Maze layout
  - Dispenser probability $\theta$
  - Environment dynamics.

# Simple model class



- Let the agent **know**:
  - Maze layout
  - Dispenser probability $\theta$
  - Environment dynamics.
- Let it be **uncertain** about *where* the only dispenser is:

$$\mathcal{M} = \{\text{Gridworld with dispenser at } (x, y)\}_{(x,y)}^{(N,N)}$$

# Simple model class



- Let the agent **know**:
  - Maze layout
  - Dispenser probability $\theta$
  - Environment dynamics.
- Let it be **uncertain** about *where* the only dispenser is:

$$\mathcal{M} = \{\text{Gridworld with dispenser at } (x, y)\}_{(x,y)}^{(N,N)}$$

- There are at most $|\mathcal{M}| \leq N^2$ 'legal' dispenser positions.

# Simple model class



- Let the agent **know**:
  - Maze layout
  - Dispenser probability $\theta$
  - Environment dynamics.
- Let it be **uncertain** about *where* the only dispenser is:

$$\mathcal{M} = \{\text{Gridworld with dispenser at } (x, y)\}_{(x,y)}^{(N,N)}$$

- There are at most $|\mathcal{M}| \leq N^2$ 'legal' dispenser positions.
- Let the agent have a uniform prior $w_\nu = |\mathcal{M}|^{-1} \ \forall \nu \in \mathcal{M}$.

# Simple model class



- Let the agent **know**:
  - Maze layout
  - Dispenser probability $\theta$
  - Environment dynamics.
- Let it be **uncertain** about *where* the only dispenser is:

$$\mathcal{M} = \{\text{Gridworld with dispenser at } (x, y)\}_{(x,y)}^{(N,N)}$$

- There are at most $|\mathcal{M}| \leq N^2$ 'legal' dispenser positions.
- Let the agent have a uniform prior $w_\nu = |\mathcal{M}|^{-1} \ \forall \nu \in \mathcal{M}$.
- Each $\nu$ is a complete gridworld simulator, and $\mu \in \mathcal{M}$.

# AIXIjs



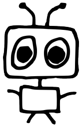
Enough talk. Let's see an

# AIXIjs

Enough talk. Let's see an

## Online web demo

## aslanides.io/aixijs



# Simple model class

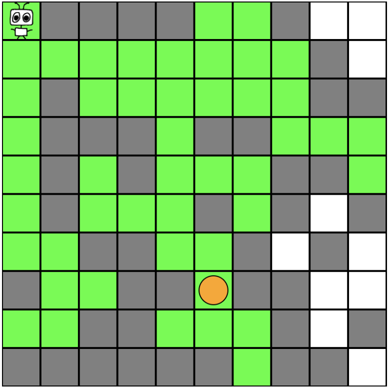


What did we just see?
Let's visualize the agent's uncertainty as it learns.

# Simple model class



What did we just see?
Let's visualize the agent's uncertainty as it learns.

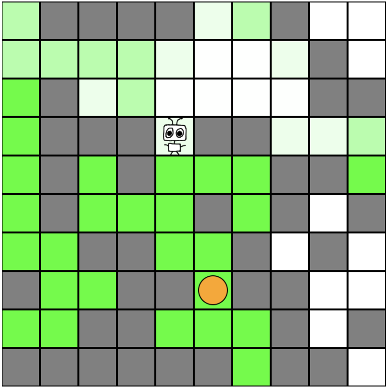


- Initially, the agent has a uniform prior, shown in green.

# Simple model class



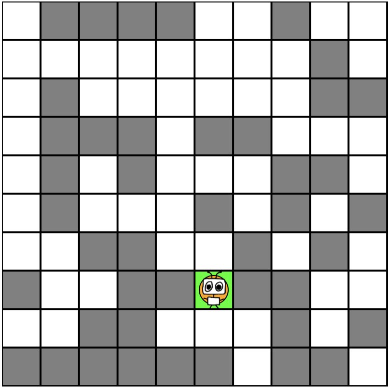
Let's visualize the agent's uncertainty as it learns.



- After exploring a little, the agent's beliefs have changed.
- Lighter green corresponds to less probability mass.

# Simple model class

Let's visualize the agent's uncertainty as it learns.

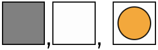


- After discovering the dispenser, the agent's posterior concentrates on $\mu$.
- This concentration is immediate – global 'collapse'.

# A more general model class



The previous model class was limited. Here's a more interesting one.

- Model each tile independently with a categorical/Dirichlet distribution over $\Big\{$  $\Big\}$:

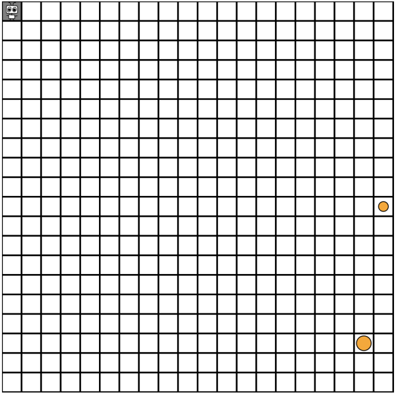
$$\rho\left(e_t|\ldots\right) = \prod_{s' \in \mathrm{ne}(s_t)} \mathrm{Dirichlet}\left(p|\alpha_{s'}\right).$$

- Joint distribution factorizes over the grid.
- The agent learns about state dynamics only **locally**, rather than **globally**.
- Using this model, the agent is **uncertain** about:
  - Maze layout
  - Location, number *and* payout probabilities $\theta_i$ of each dispenser(s).

# A more general model class

What did we just see?

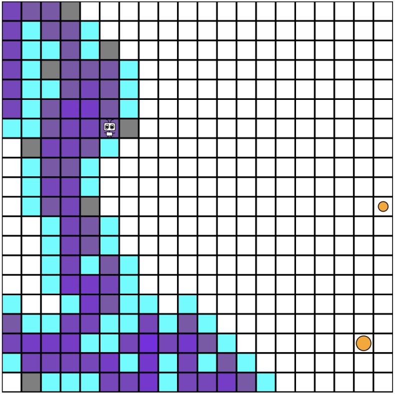
Let's visualize the agent's uncertainty as it learns.



- Initially the agent knows nothing about the layout.
- There are two dispensers, visualized for our benefit.

# A more general model class

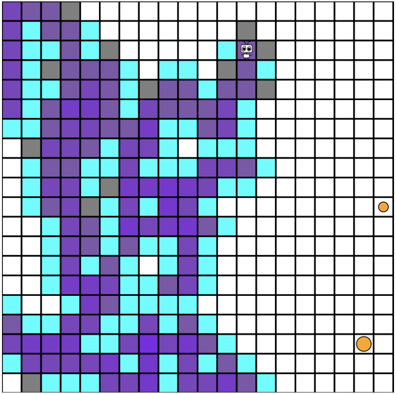
Let's visualize the agent's uncertainty as it learns.



- Tiles that the agent knows are walls are blue .
- Purple tiles  show the agent's belief of $\theta$.

# A more general model class

Let's visualize the agent's uncertainty as it learns.

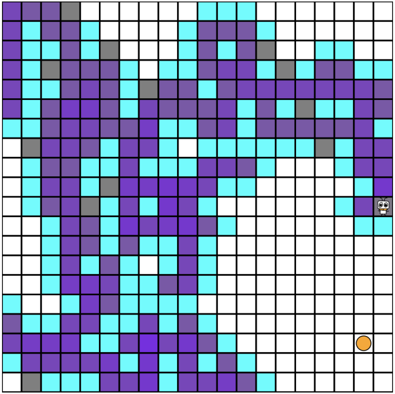


- Note: the smaller  has lower $\theta$ than the larger .
- The agent explores efficiently and learns quickly.

# A more general model class



Let's visualize the agent's uncertainty as it learns.



- Even so, the agent settles for a locally optimal policy.
- Due to its short horizon $m$, it can't see the value in exploring further.

# Exploration/exploitation trade-off

- Here we see the classic exploration/exploitation dilemma.
- Bayesian agents are not immune to this!
- Choices of:
    - model class
    - priors
    - discount function
    - planning horizon

  are all significant!
- Corollary: AI$\xi$ is not **asymptotically optimal**.

# (Aside) An even more general model class



- We've demonstrated Bayesian RL on gridworlds using very domain-oriented model classes.

# (Aside) An even more general model class



- We've demonstrated Bayesian RL on gridworlds using very domain-oriented model classes.
- Is there something more general that is still tractable?

# (Aside) An even more general model class



- We've demonstrated Bayesian RL on gridworlds using very domain-oriented model classes.
- Is there something more general that is still tractable?
- Yes! The **Context-Tree Weighting (CTW)** algorithm:

# (Aside) An even more general model class



- We've demonstrated Bayesian RL on gridworlds using very domain-oriented model classes.
- Is there something more general that is still tractable?
- Yes! The **Context-Tree Weighting (CTW)** algorithm:
  - A data compressor with good theoretical guarantees.

# (Aside) An even more general model class



- We've demonstrated Bayesian RL on gridworlds using very domain-oriented model classes.
- Is there something more general that is still tractable?
- Yes! The **Context-Tree Weighting (CTW)** algorithm:
  - A data compressor with good theoretical guarantees.
  - Mixes over all $< k^{\text{th}}$-order (in bits) Markov models.

# (Aside) An even more general model class



- We've demonstrated Bayesian RL on gridworlds using very domain-oriented model classes.
- Is there something more general that is still tractable?
- Yes! The **Context-Tree Weighting (CTW)** algorithm:
  - A data compressor with good theoretical guarantees.
  - Mixes over all $< k^{\text{th}}$-order (in bits) Markov models.
  - Automatically weights models by complexity (tree depth).

# (Aside) An even more general model class



- We've demonstrated Bayesian RL on gridworlds using very domain-oriented model classes.
- Is there something more general that is still tractable?
- Yes! The **Context-Tree Weighting (CTW)** algorithm:
  - A data compressor with good theoretical guarantees.
  - Mixes over all $< k^{\text{th}}$-order (in bits) Markov models.
  - Automatically weights models by complexity (tree depth).
  - Model updates in time linear in $k$.

# (Aside) An even more general model class

- We've demonstrated Bayesian RL on gridworlds using very domain-oriented model classes.
- Is there something more general that is still tractable?
- Yes! The **Context-Tree Weighting (CTW)** algorithm:
  - A data compressor with good theoretical guarantees.
  - Mixes over all $< k^{\text{th}}$-order (in bits) Markov models.
  - Automatically weights models by complexity (tree depth).
  - Model updates in time linear in $k$.
  - Based on the KT estimator (similar to Beta distribution).

# (Aside) An even more general model class



- We've demonstrated Bayesian RL on gridworlds using very domain-oriented model classes.
- Is there something more general that is still tractable?
- Yes! The **Context-Tree Weighting (CTW)** algorithm:
  - A data compressor with good theoretical guarantees.
  - Mixes over all $< k^{\text{th}}$-order (in bits) Markov models.
  - Automatically weights models by complexity (tree depth).
  - Model updates in time linear in $k$.
  - Based on the KT estimator (similar to Beta distribution).
  - Can model any sequential density up to a finite given context/history length.

# (Aside) An even more general model class



- We've demonstrated Bayesian RL on gridworlds using very domain-oriented model classes.
- Is there something more general that is still tractable?
- Yes! The **Context-Tree Weighting (CTW)** algorithm:
  - A data compressor with good theoretical guarantees.
  - Mixes over all $< k^{\text{th}}$-order (in bits) Markov models.
  - Automatically weights models by complexity (tree depth).
  - Model updates in time linear in $k$.
  - Based on the KT estimator (similar to Beta distribution).
  - Can model any sequential density up to a finite given context/history length.
  - Learns to play PacMan, Tic-Tac-Toe, Kuhn Poker, and Rock/Paper/Scissors *tabula rasa* [3].

# Break Time

Let's take a tea/coffee break!

(See you again in 30 mins)

# Variants of AI$\xi$



We'll discuss various variants of AIXI and their links with 'model-free'/'deep RL' algorithms:

- MDL Agent

# Variants of AI$\xi$



We'll discuss various variants of AIXI and their links with 'model-free'/'deep RL' algorithms:

- MDL Agent
- Thompson Sampling

# Variants of AI$\xi$



We'll discuss various variants of AIXI and their links with 'model-free'/'deep RL' algorithms:

- MDL Agent
- Thompson Sampling
- Knowledge-Seeking Agents

# Variants of AI$\xi$



We'll discuss various variants of AIXI and their links with 'model-free'/'deep RL' algorithms:

- MDL Agent
- Thompson Sampling
- Knowledge-Seeking Agents
- BayesExp

# MDL Agent



- **Minimum Description Length (MDL)** principle: prefer simple models

$$\rho = \arg\min_{\nu \in \mathcal{M}} \left( K(\nu) - \lambda \underbrace{\log \prod_{k=1}^{t} \log \nu\left(e_k \middle| æ_{<k} a_k\right)}_{\text{Log-likelihood}} \right)$$

# MDL Agent

- **Minimum Description Length (MDL)** principle: prefer simple models
- Another take on the 'Occam principle':

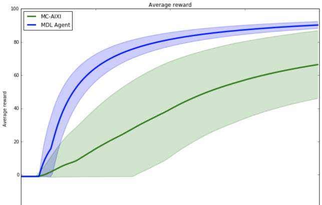
$$\rho = \arg\min_{\nu \in \mathcal{M}} \left( K(\nu) - \lambda \underbrace{\log \prod_{k=1}^{t} \log \nu\left(e_k | æ_{<k} a_k\right)}_{\text{Log-likelihood}} \right)$$

# MDL Agent

- **Minimum Description Length (MDL)** principle: prefer simple models
- Another take on the 'Occam principle':

$$\rho = \arg\min_{\nu \in \mathcal{M}} \left( K(\nu) - \lambda \underbrace{\log \prod_{k=1}^{t} \log \nu \left( e_k | æ_{<k} a_k \right)}_{\text{Log-likelihood}} \right)$$

- In deterministic environments: "use the simplest yet-unfalsified hypothesis"



# Thompson Sampling



- Recall the Bayes-optimal agent (AI$\xi$) maximizes $\xi$-expected return:

$$\begin{aligned} a_{AI\xi} &= \arg\max_a Q^\star_\xi\left(a | æ_{<t}\right) \\ &= \arg\max_a \max_\pi \mathbb{E}^\pi_\xi\left[\sum_{k=t}^{\infty} \gamma_k r_k \,\middle|\, æ_{<t} a\right] \end{aligned}$$

# Thompson Sampling



- Recall the Bayes-optimal agent (AI$\xi$) maximizes $\xi$-expected return:

$$\begin{aligned} a_{AI\xi} &= \arg\max_a Q^\star_\xi\left(a \middle| æ_{<t}\right) \\ &= \arg\max_a \max_\pi \mathbb{E}^\pi_\xi\left[\sum_{k=t}^{\infty} \gamma_k r_k \middle| æ_{<t} a\right] \end{aligned}$$

- A related algorithm is Thompson sampling).

# Thompson Sampling



- Recall the Bayes-optimal agent (AI$\xi$) maximizes $\xi$-expected return:

$$
\begin{aligned}
a_{AI\xi} &= \arg\max_{a} Q^{\star}_{\xi}\left(a | æ_{<t}\right) \\
&= \arg\max_{a} \max_{\pi} \mathbb{E}^{\pi}_{\xi}\left[\sum_{k=t}^{\infty} \gamma_k r_k \middle| æ_{<t} a\right]
\end{aligned}
$$

- A related algorithm is Thompson sampling).
- Idea: Instead of maximizing the $\xi$-expected return:

# Thompson Sampling



- Recall the Bayes-optimal agent (AI$\xi$) maximizes $\xi$-expected return:

$$
\begin{aligned}
a_{AI\xi} &= \arg\max_a Q^\star_\xi\left(a|æ_{<t}\right) \\
&= \arg\max_a \max_\pi \mathbb{E}^\pi_\xi\left[\sum_{k=t}^{\infty} \gamma_k r_k \middle| æ_{<t} a\right]
\end{aligned}
$$

- A related algorithm is Thompson sampling).
- Idea: Instead of maximizing the $\xi$-expected return:
  - maximize the $\rho$-expected return, $\rho$ drawn from $w\left(\cdot|æ_{<t}\right)$.

# Thompson Sampling



- Recall the Bayes-optimal agent (AI$\xi$) maximizes $\xi$-expected return:

$$
\begin{aligned}
a_{AI\xi} &= \arg\max_a Q^\star_\xi\left(a | æ_{<t}\right) \\
&= \arg\max_a \max_\pi \mathbb{E}^\pi_\xi \left[ \sum_{k=t}^{\infty} \gamma_k r_k \middle| æ_{<t} a \right]
\end{aligned}
$$

- A related algorithm is Thompson sampling).
- Idea: Instead of maximizing the $\xi$-expected return:
  - maximize the $\rho$-expected return, $\rho$ drawn from $w\left(\cdot | æ_{<t}\right)$.
  - resample $\rho$ every 'effective horizon' given by discount $\gamma$.

# Thompson Sampling



- Recall the Bayes-optimal agent (AI$\xi$) maximizes $\xi$-expected return:

$$a_{AI\xi} = \arg\max_a Q^{\star}_{\xi}\left(a|æ_{<t}\right)$$
$$= \arg\max_a \max_{\pi} \mathbb{E}^{\pi}_{\xi}\left[\sum_{k=t}^{\infty} \gamma_k r_k \middle| æ_{<t} a\right]$$

- A related algorithm is Thompson sampling).
- Idea: Instead of maximizing the $\xi$-expected return:
  - maximize the $\rho$-expected return, $\rho$ drawn from $w\left(\cdot|æ_{<t}\right)$.
  - resample $\rho$ every 'effective horizon' given by discount $\gamma$.
- Good regret guarantees in finite MDPs [1]

# Thompson Sampling



- Recall the Bayes-optimal agent (AI$\xi$) maximizes $\xi$-expected return:

$$\begin{aligned} a_{AI\xi} &= \arg\max_a Q^\star_\xi\left(a|æ_{<t}\right) \\ &= \arg\max_a \max_\pi \mathbb{E}^\pi_\xi\left[\sum_{k=t}^{\infty} \gamma_k r_k \middle| æ_{<t} a\right] \end{aligned}$$

- A related algorithm is Thompson sampling).
- Idea: Instead of maximizing the $\xi$-expected return:
  - maximize the $\rho$-expected return, $\rho$ drawn from $w\left(\cdot|æ_{<t}\right)$.
  - resample $\rho$ every 'effective horizon' given by discount $\gamma$.
- Good regret guarantees in finite MDPs [1]
- Asymptotically optimal in general environments [2].

# Thompson Sampling



- Recall the Bayes-optimal agent (AI$\xi$) maximizes $\xi$-expected return:

$$a_{AI\xi} = \arg\max_a Q^\star_\xi\left(a|æ_{<t}\right) = \arg\max_a \max_\pi \mathbb{E}^\pi_\xi\left[\sum_{k=t}^{\infty}\gamma_k r_k \middle| æ_{<t}a\right]$$

- A related algorithm is Thompson sampling).
- Idea: Instead of maximizing the $\xi$-expected return:
  - maximize the $\rho$-expected return, $\rho$ drawn from $w\left(\cdot|æ_{<t}\right)$.
  - resample $\rho$ every 'effective horizon' given by discount $\gamma$.
- Good regret guarantees in finite MDPs [1]
- Asymptotically optimal in general environments [2].
- Intuition: 'commits' the agent to a given belief/policy for a significant amount of time,

# Thompson Sampling

- Recall the Bayes-optimal agent (AI$\xi$) maximizes $\xi$-expected return:

$$a_{AI\xi} = \arg\max_a Q^{\star}_{\xi}\left(a | æ_{<t}\right)$$
$$= \arg\max_a \max_\pi \mathbb{E}^{\pi}_{\xi}\left[\sum_{k=t}^{\infty} \gamma_k r_k \middle| æ_{<t} a\right]$$

- A related algorithm is Thompson sampling).
- Idea: Instead of maximizing the $\xi$-expected return:
  - maximize the $\rho$-expected return, $\rho$ drawn from $w\left(\cdot | æ_{<t}\right)$.
  - resample $\rho$ every 'effective horizon' given by discount $\gamma$.
- Good regret guarantees in finite MDPs [1]
- Asymptotically optimal in general environments [2].
- Intuition: 'commits' the agent to a given belief/policy for a significant amount of time,
  - this encourages 'deep' exploration.

# Thompson Sampling



'Deep RL' version: **Deep Exploration via Bootstrapped DQN** [2].

- Idea: Maintain an **ensemble** of value functions $\{Q_k(s,a)\}$.

# Thompson Sampling

'Deep RL' version: **Deep Exploration via Bootstrapped DQN** [2].

- Idea: Maintain an **ensemble** of value functions $\{Q_k(s, a)\}$.
- Train these using e.g. DQN using the statistical bootstrap.

# Thompson Sampling



'Deep RL' version: **Deep Exploration via Bootstrapped DQN** [2].

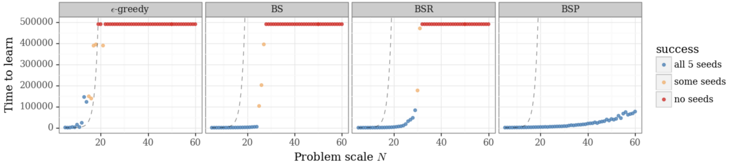
- Idea: Maintain an **ensemble** of value functions $\{Q_k(s, a)\}$.
- Train these using e.g. DQN using the statistical bootstrap.
- Thompson sampling: draw a $Q$-function at random each episode and use a greedy policy.

# Thompson Sampling



'Deep RL' version: **Deep Exploration via Bootstrapped DQN** [2].

- Idea: Maintain an **ensemble** of value functions $\{Q_k(s,a)\}$.
- Train these using e.g. DQN using the statistical bootstrap.
- Thompson sampling: draw a $Q$-function at random each episode and use a greedy policy.
- Exhibits much better exploration properties than many alternatives



# Knowledge-Seeking Agents



- It has long been thought that some form of **intrinsic motivation**, **surprise**, or **curiosity** is necessary for effective exploration and learning [5].

# Knowledge-Seeking Agents



- It has long been thought that some form of **intrinsic motivation**, **surprise**, or **curiosity** is necessary for effective exploration and learning [5].
- **Knowledge-seeking agents (KSA)** take to this to the extreme:

# Knowledge-Seeking Agents



- It has long been thought that some form of **intrinsic motivation**, **surprise**, or **curiosity** is necessary for effective exploration and learning [5].
- **Knowledge-seeking agents (KSA)** take to this to the extreme:
  - Fully unsupervised (no extrinsic rewards)

# Knowledge-Seeking Agents



- It has long been thought that some form of **intrinsic motivation**, **surprise**, or **curiosity** is necessary for effective exploration and learning [5].
- **Knowledge-seeking agents (KSA)** take to this to the extreme:
  - Fully unsupervised (no extrinsic rewards)
  - Utility function depends on agent beliefs about the world

# Knowledge-Seeking Agents

- It has long been thought that some form of **intrinsic motivation**, **surprise**, or **curiosity** is necessary for effective exploration and learning [5].
- **Knowledge-seeking agents (KSA)** take to this to the extreme:
  - Fully unsupervised (no extrinsic rewards)
  - Utility function depends on agent beliefs about the world
  - Exploration $\equiv$ Exploitation

# Knowledge-Seeking Agents

- It has long been thought that some form of **intrinsic motivation**, **surprise**, or **curiosity** is necessary for effective exploration and learning [5].
- **Knowledge-seeking agents (KSA)** take to this to the extreme:
    - Fully unsupervised (no extrinsic rewards)
    - Utility function depends on agent beliefs about the world
    - Exploration $\equiv$ Exploitation
- Two forms:

# Knowledge-Seeking Agents

- It has long been thought that some form of **intrinsic motivation**, **surprise**, or **curiosity** is necessary for effective exploration and learning [5].
- **Knowledge-seeking agents (KSA)** take to this to the extreme:
  - Fully unsupervised (no extrinsic rewards)
  - Utility function depends on agent beliefs about the world
  - Exploration $\equiv$ Exploitation
- Two forms:
  - Shannon KSA ("surprise"):

$$U\left(e_t | æ_{<t} a_t\right) = -\log \xi\left(e_t | æ_{<t} a_t\right)$$

# Knowledge-Seeking Agents



- It has long been thought that some form of **intrinsic motivation**, **surprise**, or **curiosity** is necessary for effective exploration and learning [5].
- **Knowledge-seeking agents (KSA)** take to this to the extreme:
  - Fully unsupervised (no extrinsic rewards)
  - Utility function depends on agent beliefs about the world
  - Exploration $\equiv$ Exploitation
- Two forms:
  - Shannon KSA ("surprise"):

$$U\left(e_t | æ_{<t} a_t\right) = -\log \xi\left(e_t | æ_{<t} a_t\right)$$

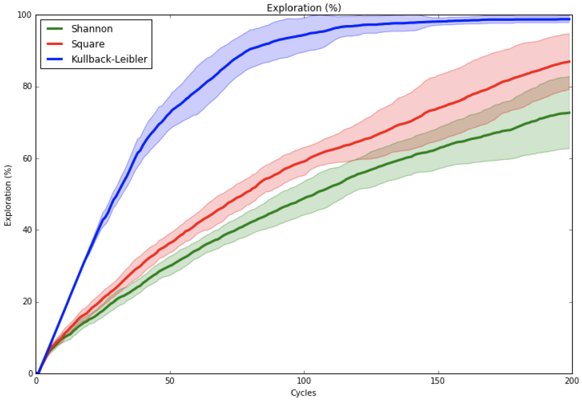
  - Kullback-Leibler KSA ("information gain"):

$$U\left(e_t | æ_{<t} a_t\right) = \mathrm{Ent}\left(w | æ_{<t} a_t\right) - \mathrm{Ent}\left(w | æ_{1:t}\right)$$

# Knowledge-Seeking Agents



Kullback Leibler ("information-seeking") is superior to Shannon & Renyi ("entropy-seeking"):



# Knowledge-Seeking Agents



'Deep RL' version: **Variational Information Maximization for Exploration (VIME)** [1].

- Idea:

# Knowledge-Seeking Agents



'Deep RL' version: **Variational Information Maximization for Exploration (VIME)** [1].

- Idea:
  - Learn a forward dynamics model in tandem with model-free RL

# Knowledge-Seeking Agents

'Deep RL' version: **Variational Information Maximization for Exploration (VIME)** [1].

- Idea:
  - Learn a forward dynamics model in tandem with model-free RL
  - Use a variational approximation to compute the information gain in closed form

# Knowledge-Seeking Agents



'Deep RL' version: **Variational Information Maximization for Exploration (VIME)** [1].

- Idea:
  - Learn a forward dynamics model in tandem with model-free RL
  - Use a variational approximation to compute the information gain in closed form
  - Use this as an 'exploration bonus', or intrinsic reward

# Knowledge-Seeking Agents



'Deep RL' version: **Variational Information Maximization for Exploration (VIME)** [1].

- Idea:
  - Learn a forward dynamics model in tandem with model-free RL
  - Use a variational approximation to compute the information gain in closed form
  - Use this as an 'exploration bonus', or intrinsic reward
- Downside: only works well when learning from 'states', not pixels (wrong loss).

# BayesExp

Combine best of both worlds:

- Bayes-optimal reinforcement learner (AI$\xi$) with

# BayesExp



Combine best of both worlds:

- Bayes-optimal reinforcement learner (AI$\xi$) with
- Information-seeking (KL-KSA).

# BayesExp

Combine best of both worlds:

- Bayes-optimal reinforcement learner (AI$\xi$) with
- Information-seeking (KL-KSA).
- Idea: switch between RL and KSA policies depending on the relative sizes of $V_{KSA}$ and $V_{RL}$.

# Thanks!



Thanks!

# References



Marcus Hutter (2005):
Universal Artificial Intelligence.

David Silver et al. (2017):
Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm.

Joel Veness et al. (2011):
A Monte-Carlo AIXI Approximation.

Levente Kocsis and Csaba Szepesvari (2006):
Bandit based Monte-Carlo Planning.

Peter Auer (2002):
Using Confidence Bounds for Exploitation-Exploration Trade-offs.

# References



Shipra Agrawal and Randy Jia (2017):
Posterior Sampling for Reinforcement Learning: Worst-Case Regret Bounds.

Jan Leike et al. (2016):
Thompson Sampling is Asymptotically Optimal in General Environments.

John Aslanides, Jan Leike, and Marcus Hutter (2017):
Universal Reinforcement Learning Algorithms: Survey and Experiments.

Ian Osband, John Aslanides, and Albin Cassirer (2018):
Randomized Prior Functions for Deep Reinforcement Learning.

Juergen Schmidhuber (2008):
Driven by Compression Progress.

# References



Rein Houthooft et al. (2016):
VIME: Variational Information Maximization for Exploration.