Reinforcement Learning and Large Deviations

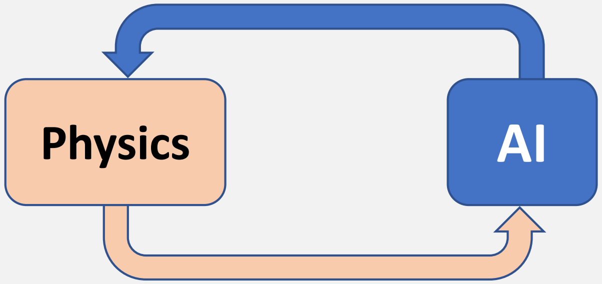
Jacob H. Adamczyk

17 October 2022

Applied Physics PhD Qualifying Exam

Introduction

- Reinforcement Learning (RL), a subset of AI, has had great success in the past decade.
- Large Deviations (LD) theory, an analytical framework for studying non-equilibrium stat. mech. (NESM), gives a new way to describe the RL problem
- In applying such physics-based analysis to RL, we aim to gain insight on the RL problem



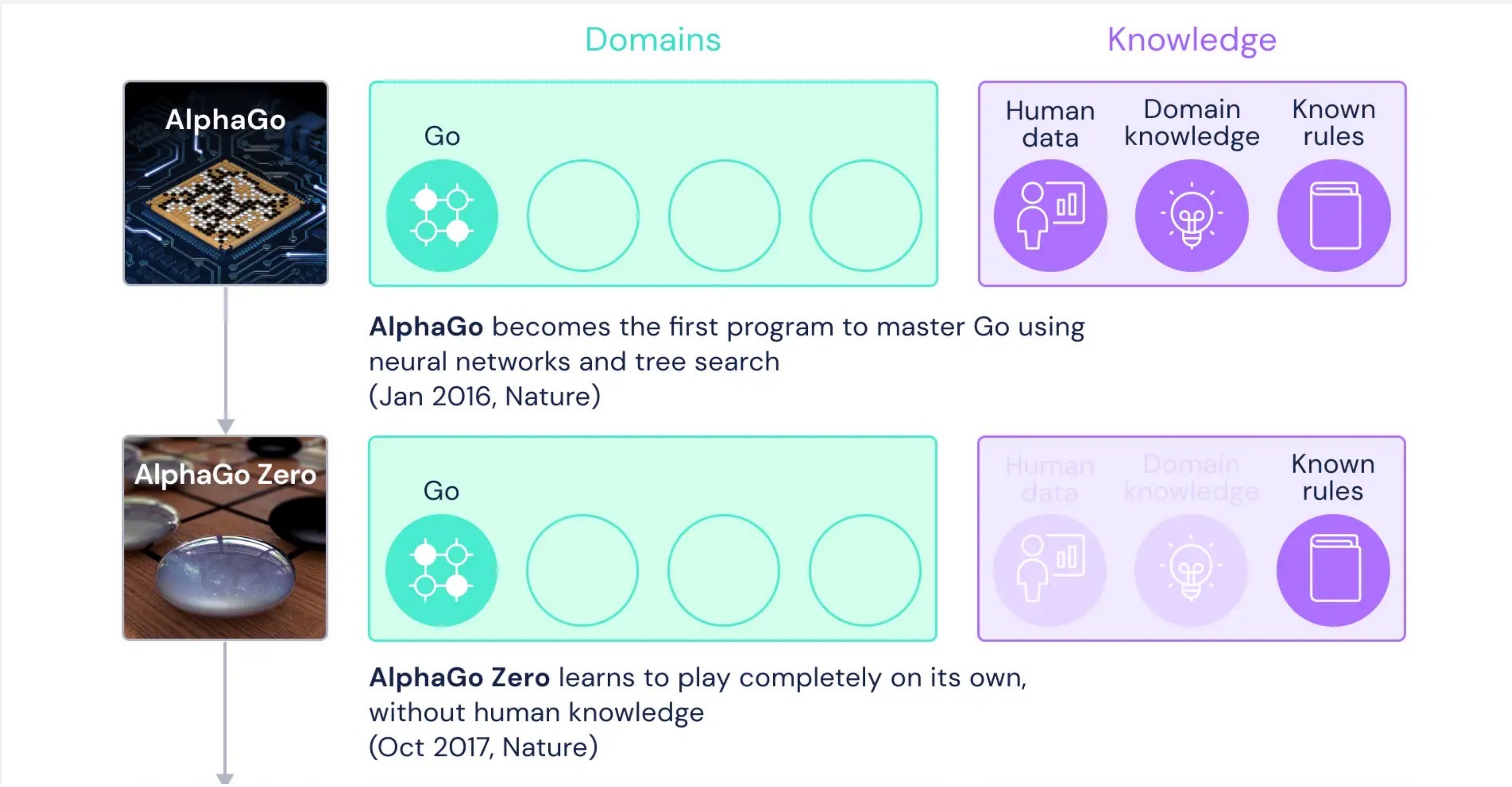
Overview

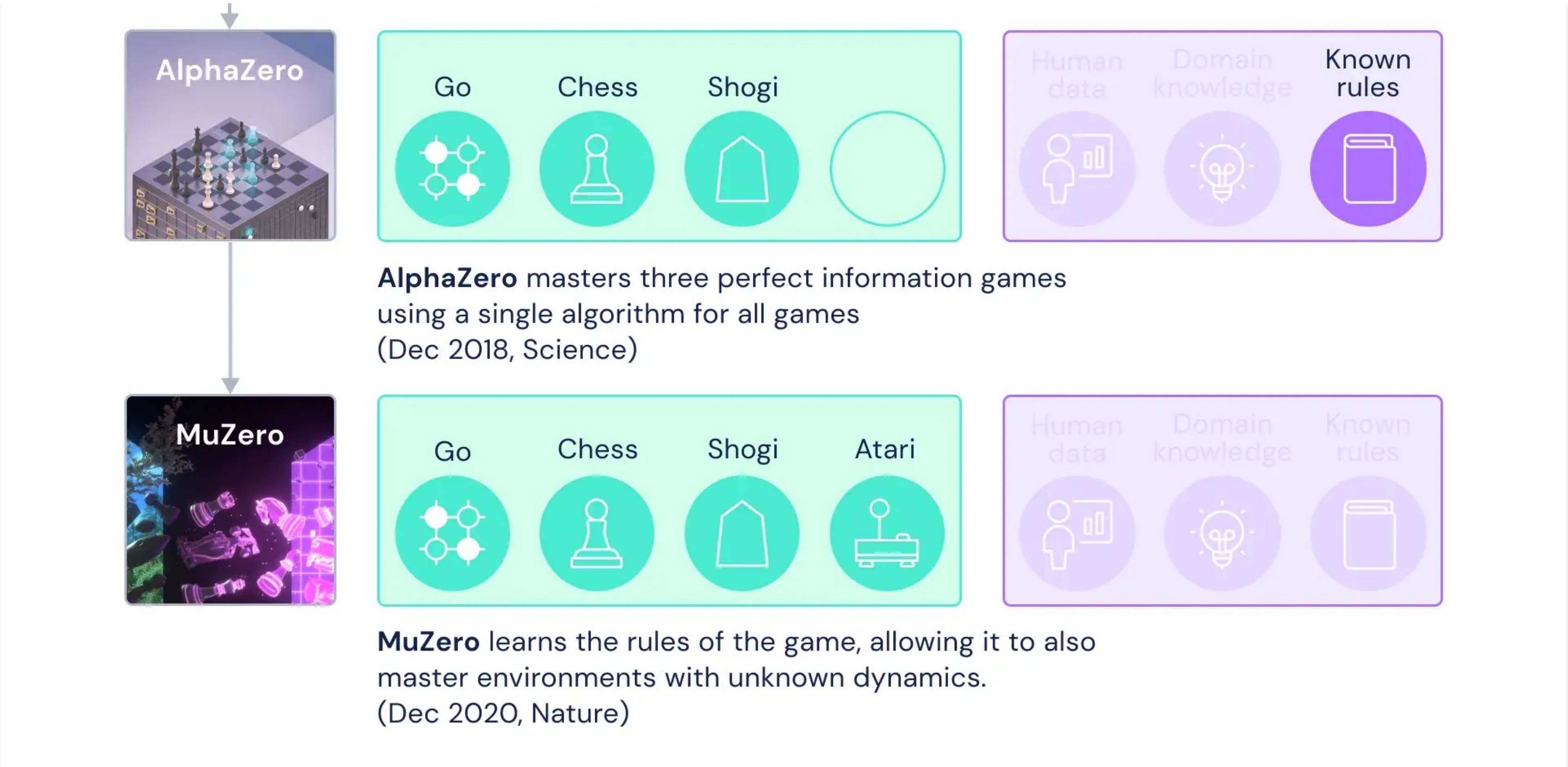
I. Introduce Reinforcement Learning
   a. Markov Decision Process model
   b. Solution methods
   c. Extensions of “standard” RL

II. Introduce Large Deviations
   a. Rate function, cumulant generating function
   b. Show equivalence to RL problem

III. Exhibit connections and applications

Reinforcement Learning





RL success over past ~5 years due to the advent of deep neural networks

From DeepMind’s blog: https://www.deepmind.com/blog/muzero-mastering-go-chess-shogi-and-atari-without-rules

Motivation for RL

Why should physicists care about RL?

➢ *RL is more than games or robotics*

- RL problem can be mapped to NESM
- RL as a direct tool for physics problems
- RL can give insight to Perron-Frobenius theory (well-employed in modeling)

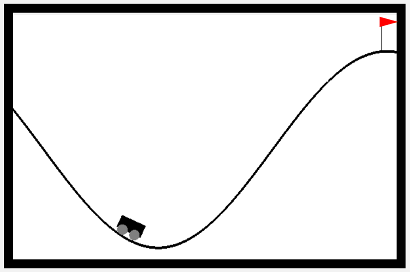
Goal of Thesis

To further develop and exploit this newfound bridge:

RL $\longleftrightarrow$ **NESM**

Reinforcement Learning

- Reinforcement Learning (RL) is a paradigm created to solve decision-making problems



Basic Idea:

- An agent interacts with the **environment**
- Positive behaviors are reinforced relative to undesirable behaviors
  - Reinforcement is implemented via a **reward function**
- After many interaction-reinforcement cycles, the agent should learn to “successfully” interact with the environment

Reinforcement Learning

Two immediate questions arise:

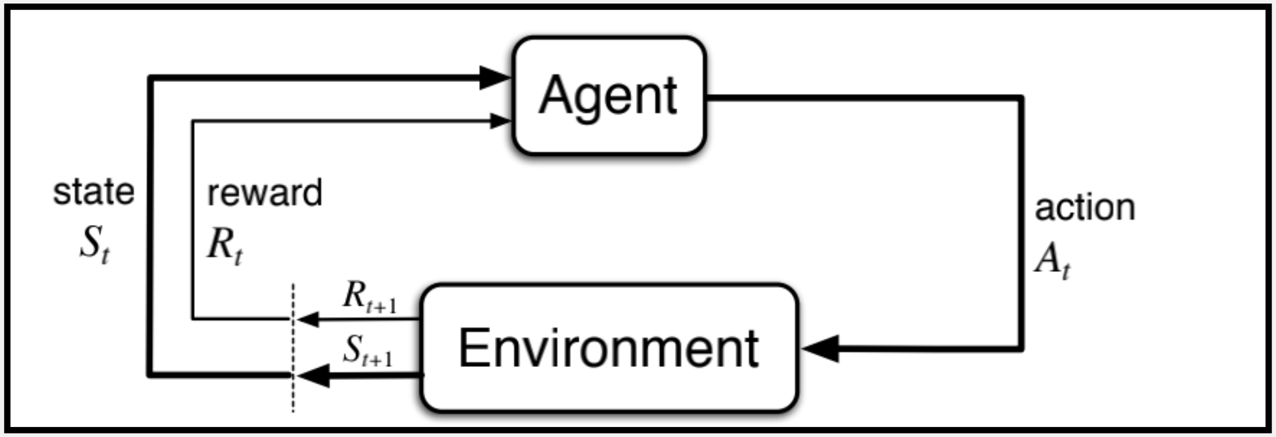
How do we model the problem?

&

How do we derive solutions?

Markov Decision Process

- The current **state** (s) and **action** (a) are used to label the agent's steps in a **trajectory**: $\tau = (s_1, a_1, s_2, a_2, \dots)$
- We model the transition **dynamics** as having the Markov property;
  - Dynamics (p) can be either stochastic or deterministic
- Want "good" **policy** $\pi(a|s)$ which chooses action at each state



"Reinforcement Learning: An Introduction", Book by Andrew Barto and Richard S. Sutton. 2018

Solving the RL Problem

Following a policy $\pi(a|s)$, what is the value of starting in state s and taking an action a?

How much is a policy worth?

<u>Policy Evaluation:</u>

$$Q^{\pi}(s,a) = r(s,a) + \overbrace{\gamma \mathbb{E}_{s' \sim p} \mathbb{E}_{a' \sim \pi} Q^{\pi}(s',a')}^{\text{Discounted future expected value}}$$

Value function

Reward for current state and action

"Discount factor" $\gamma \in (0,1)$

Expectation over next possible states

Expectation over next possible actions

Solving the RL Problem

What does the solution look like?

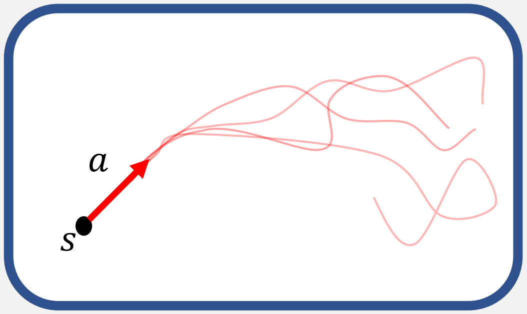
We need to know the decision-making strategy (policy) which attains the highest expected value

Formulate the following objective function:

$$J(\pi) = \mathbb{E}_{\tau \sim p,\pi} \sum_{t=1}^{\infty} \gamma^t r(s_t, a_t)$$

Correspondingly, our optimization problem is:

$$\pi^* = \operatorname*{argmax}_{\pi} J(\pi)$$



Trajectories induced by π, p; given an initial state and action

Solving the RL Problem

The (traditional) way of solving the RL problem is via the Bellman optimality equation:

Take actions in a "greedy" fashion

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim p}\left(\max_{a'} Q^*(s',a')\right)$$

Once we have Q^*, we can calculate the optimal policy:

$$\pi^*(a|s) = \operatorname*{argmax}_{a} Q^*(s,a)$$

"Greedy" policy

Solving the RL problem

<u>How to solve this nonlinear functional equation?</u>

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim p}\,(\max_{a'} Q^*(s',a'))$$

In the simplest case (model-based, tabular/discrete), one can iterate the Bellman *backup* equation:

$$Q^{(k+1)}(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim p}\,(\max_{a'} Q^{(k)}(s',a'))$$

until convergence, using an arbitrary initialization, $Q^{(0)}(s,a)$

Variants on “Standard” RL

Next, introduce two variations on the standard problem set up:

Entropy-Regularized RL

Average-Reward RL

Variant 1: Entropy-Regularized RL

Entropy-Regularized RL

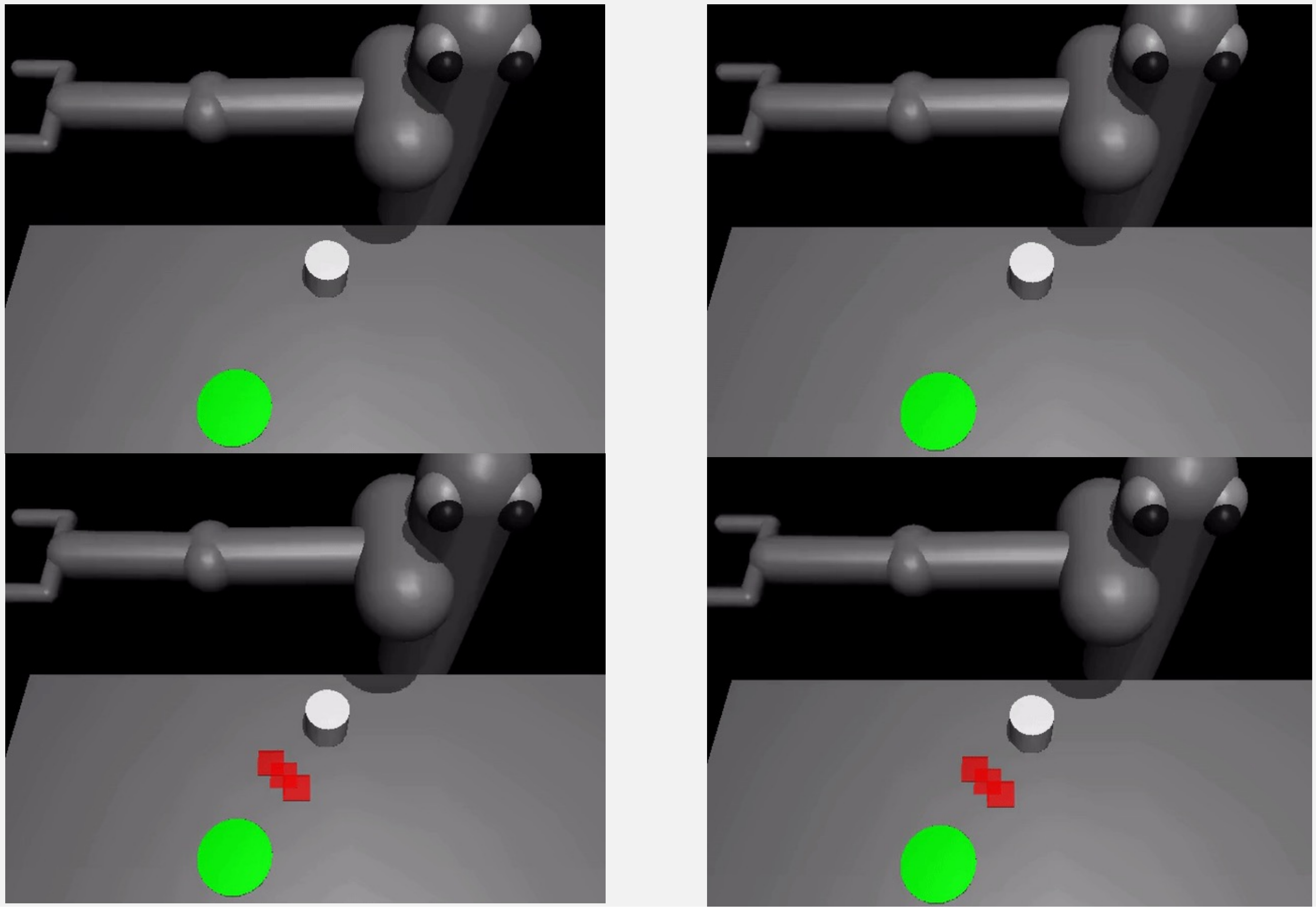
Energy minimization $\rightarrow$ Free energy minimization

- Robust to perturbations
- Less likely to get trapped in local minima
- More exploratory

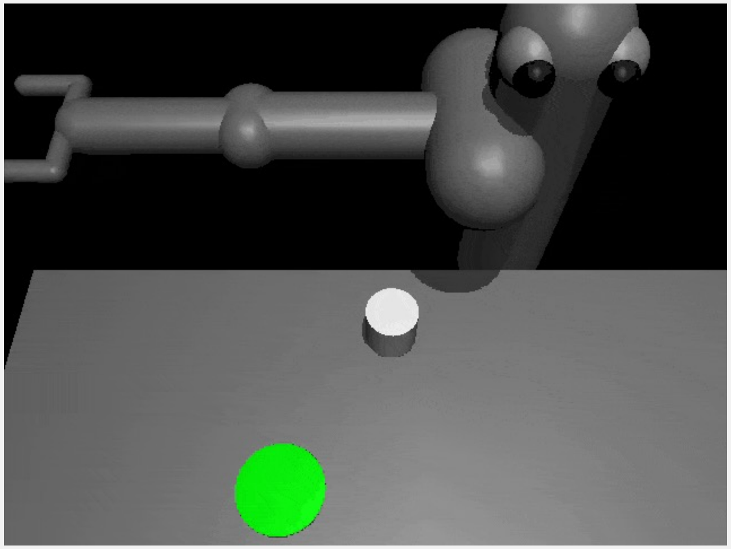
Standard RL | Entropy-Regularized RL

Trained and evaluated **without** obstacle:

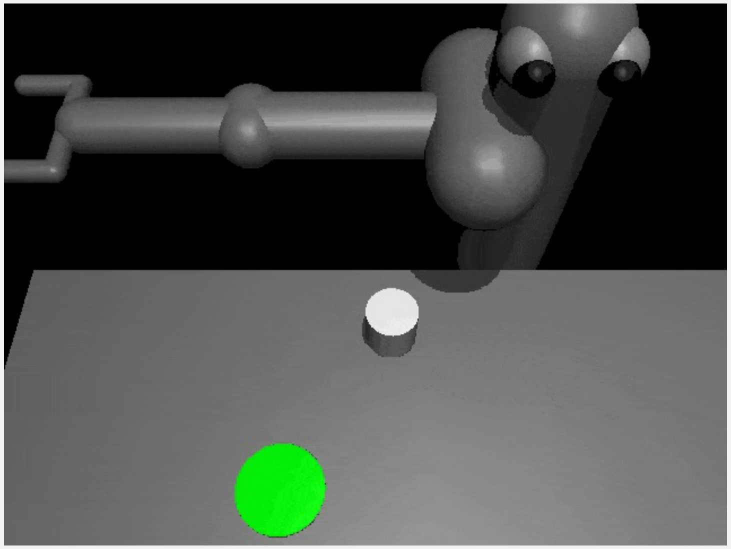
Trained **without** obstacle, evaluated with obstacle:



Standard RL



Entropy-Regularized RL



https://bair.berkeley.edu/blog/2021/03/10/maxent-robust-rl/

Entropy-Regularized RL

Entropy regularization "softens" the Bellman optimality equation:

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim p} \underbrace{\frac{1}{\beta} \log \mathbb{E}_{a' \sim \pi_0} \exp \beta Q^*(s',a')}_{\text{Previously } \max_a Q^*(s',a')}$$

*Note that as $\beta \to \infty$, the original objective is recovered

Entropy-Regularized RL

Rather than maximizing rewards alone, we can penalize based on an "information" or entropy cost based on how far away the optimal policy is from a reference policy, π_0

- Update the objective with an entropic cost

$$J(\pi) = \mathbb{E}_\tau\left[\sum_{t=1}^{\infty} \gamma^t \left(r(s_t, a_t) + \frac{1}{\beta} \log \frac{\pi(a_t|s_t)}{\pi_0(a_t|s_t)} \right) \right]$$

"Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review" Sergey Levine. 2018. Arxiv:abs/1805.00909

Variant 2: Average-Reward RL

Average-Reward RL

Common wisdom for choosing γ:

"Make γ as close to 1 as possible!!!"

Average-Reward RL

- Prefer long-term goals equally to short-term goals
- Total energy of a trajectory matters
  - Timestep shouldn't influence E (time-homogeneity)
- γ is an unnecessary hyperparameter
- Historically γ was introduced to guarantee convergence

Average-Reward RL

Rather than (artificially) discounting, consider average reward accumulated:

$$J(\pi) = \lim_{N\to\infty} \mathbb{E}_{\tau\sim p,\pi} \frac{1}{N}\sum_{t=1}^{N} r(s_t, a_t)$$

Our goal is to maximize this reward **rate** $J(\pi)$ by choosing a good π.

- We shall also assume deterministic dynamics hereon

Mahadevan, S. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Mach Learn* **22**, 159–195 (1996)

Wan Y., Naik A., **Sutton R**. Learning and Planning in Average-Reward Markov Decision Processes. (2020)

Average-Reward and Entropy-Regularized RL

Can we merge the two flavors?

- Previously not done, although both have their own benefits: softening + physicality

Robustness, Flexibility

Physical interpretation, Remove hyperparam.

This turns out to be the natural problem formulation to approach with non-equilibrium statistical mechanics (NESM)

Average-Reward <u>and</u> Entropy-Regularized RL

We get all the previous benefits of both variants, and moreover:

- Another hyperparameter can be reduced
  - LDT gives meaning to β – a control parameter to set average energy
- Can use known tools from NESM
  - Cloning algorithm (importance sampling)
  - Donsker-Varadhan variational form
  - Large deviations results

Statistical Mechanics

Basic Outline:

1. (Unweighted) trajectory distribution (following some π_0)
2. Want to probe rare events (lower avg. $E(\tau)$):
   - Control dynamics s.t. $\langle E(\tau) \rangle_c \ll \langle E(\tau) \rangle_0$
3. Equivalence of ensembles:

$$X_t | \mathcal{A}_T \cong Y_t$$

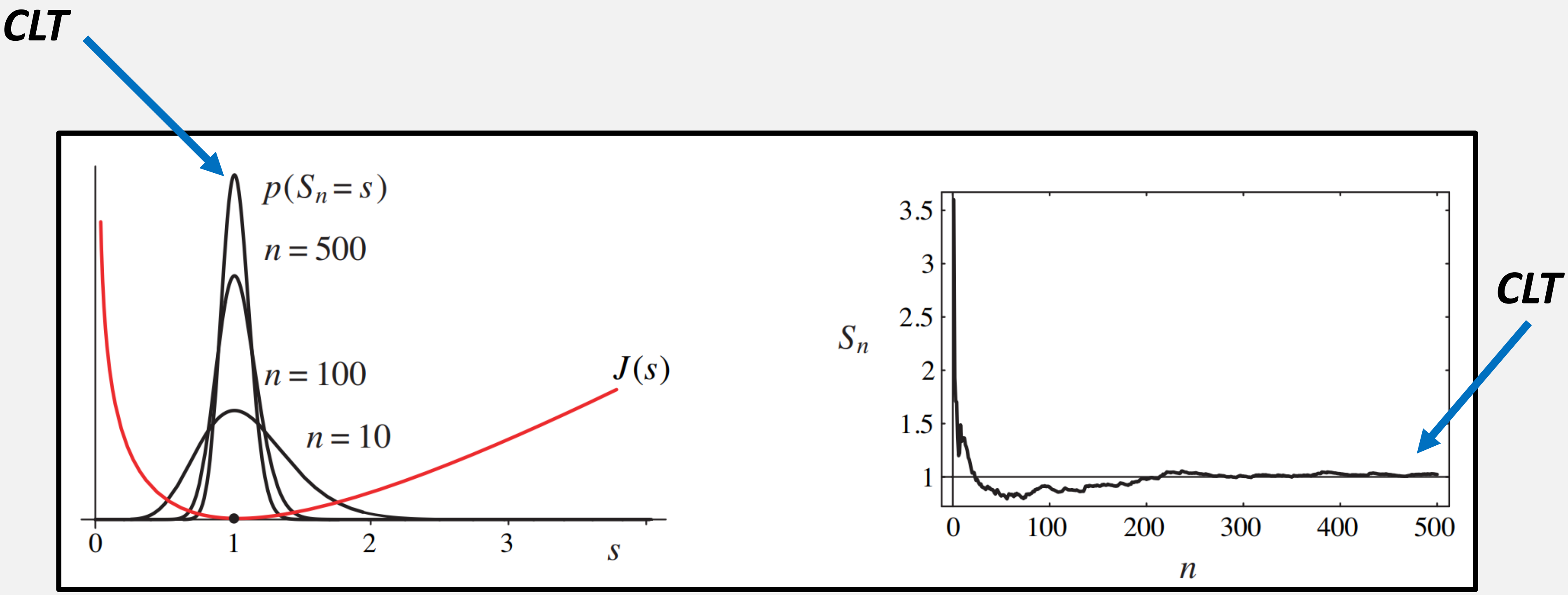
M.C. conditioned (on rare event)

Unconditioned M.C.

Large Deviations Theory

1. Working in the unconstrained (biased) ensemble is easier
   - c.f. canonical vs microcanonical
2. Biased trajectory probability:
   - $P_0(\tau) \to \frac{1}{Z} P_0(\tau) e^{-\beta E(\tau)}$ → Give higher weight to trajectories w/ lower E
   - β corresponds to a choice of $\langle E(\tau) \rangle_c = -\frac{\partial \log Z}{\partial \beta}$
   - Define the free energy $-\beta F \doteq \log Z$
3. Legendre-Fenchel transform of $F(\beta) \to I(E)$
   - Can also use F as the cumulant generating function
4. **The optimal policy and reward rate ($\boldsymbol{\pi}^*$, $\boldsymbol{F(\beta)}$) are dominant e.val and e.vec of tilted generator** (2.)

LDT Example


CLT
p(Sₙ = s)
n = 500
n = 100
n = 10
J(s)
0
1
2
3
s
CLT
3.5
3
2.5
2
1.5
1
Sₙ
0
100
200
300
400
500
n

Connections: NESM → RL

- Can use cloning method to find $\theta(\beta)$ to solve distributional RL
- Can invent algorithms for solving the avg. rwd. entropy-reg RL
    - $\log(u) - \chi$
    - $u - \theta$
- Bogoliubov inequality over trajectories (rather than config.)
- Connection to Jarzynski relation in the quenched limit

Connections: RL → NESM

- Policy Improvement Theorem for driven matrix
- Iterated Bogoliubov (can improve the bound)
- Can use RL techniques (FA's) to solve big LD/spectral problems[1]
- Reward Shaping (changing the energy landscape in a way that leaves the NESM quantities invariant

[1] *Ariel Barr, Willem Gispen, Austen Lamacraft Proceedings of The First Mathematical and Scientific Machine Learning Conference*, PMLR 107:635-653, 2020.

Applications

In Physics:

- Quantum entanglement cooling
- Bogoliubov inequality for trajectories
- Biological Networks
- Spectral problems (ground state)
- New algorithms inspired by RL (driven dynamics improvement)

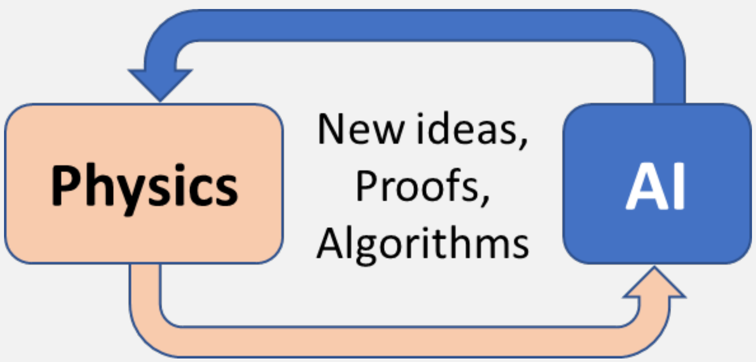
Applications

In Reinforcement Learning:

- Compositionality
- Reward Shaping
- Gauge invariance in RL
- New RL algorithms inspired by NESM
- Distributional RL

Conclusions

There is a connection between ML and physics that can be further investigated and exploited; hopefully in a positive-feedback loop style

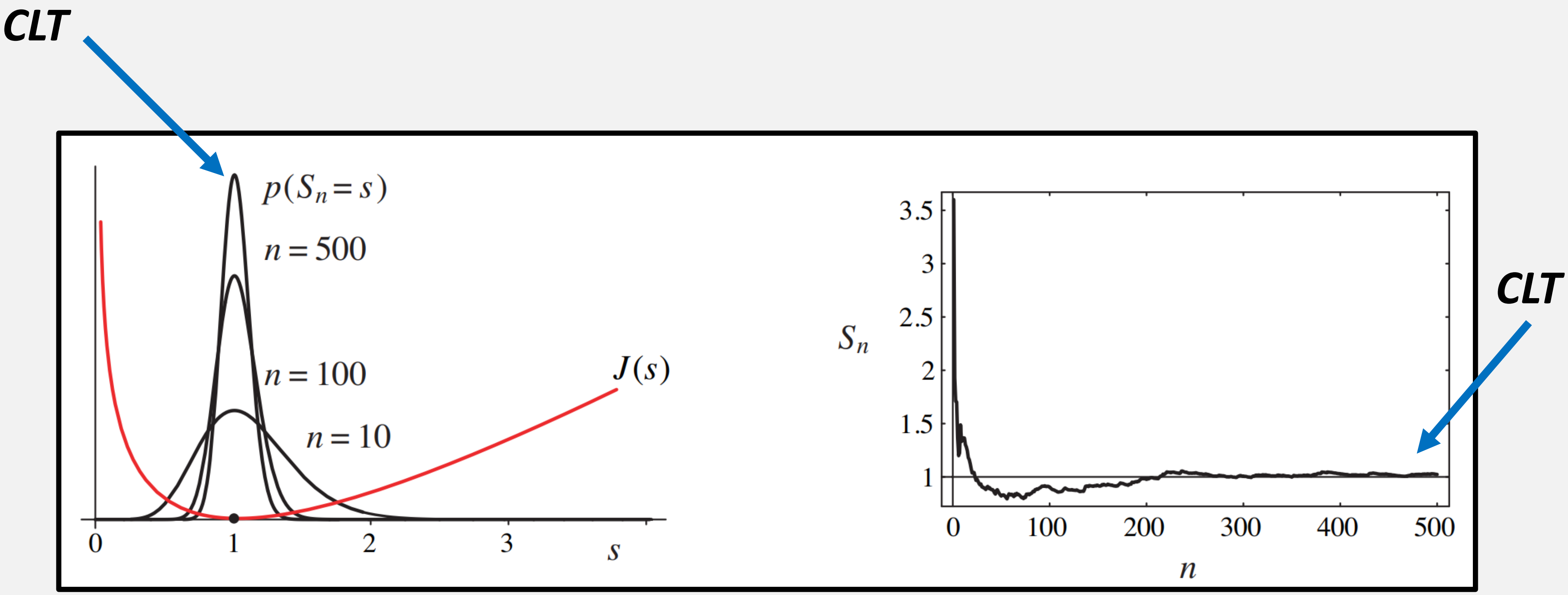


Extra Slides

LDT Example

1. Start with a random variable:
   - $p(X_i = x) = \frac{1}{\mu} e^{-x/\mu}$
   - i.e. $X_i \sim \mathcal{E}(\mu)$
2. Choose a “time-integrated observable”:
   - Sample Mean, $S_n = \frac{1}{n}\sum_{i=1}^{n} X_i$
3. Calculate scaled-cumulant generating function (scgf):
   - $\theta(\beta) = \lim_{n\to\infty} \frac{1}{n} \log\langle e^{-n\beta S_n}\rangle$
4. Obtain the LDT rate function $I(s)$ as the Legendre-Fenchel transform:
   - $P(S_n = s) \sim e^{-nI(s)}$

LDT Example


CLT
CLT
$p(S_n = s)$
$n = 500$
$n = 100$
$n = 10$
$J(s)$
0
1
2
3
s
S_n
3.5
3
2.5
2
1.5
1
0
100
200
300
400
500
n