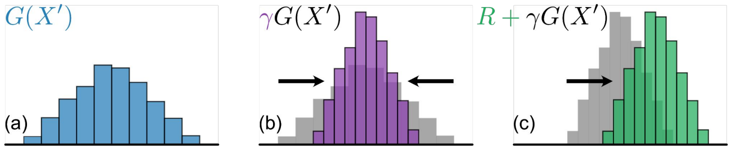
www.distributional-rl.org

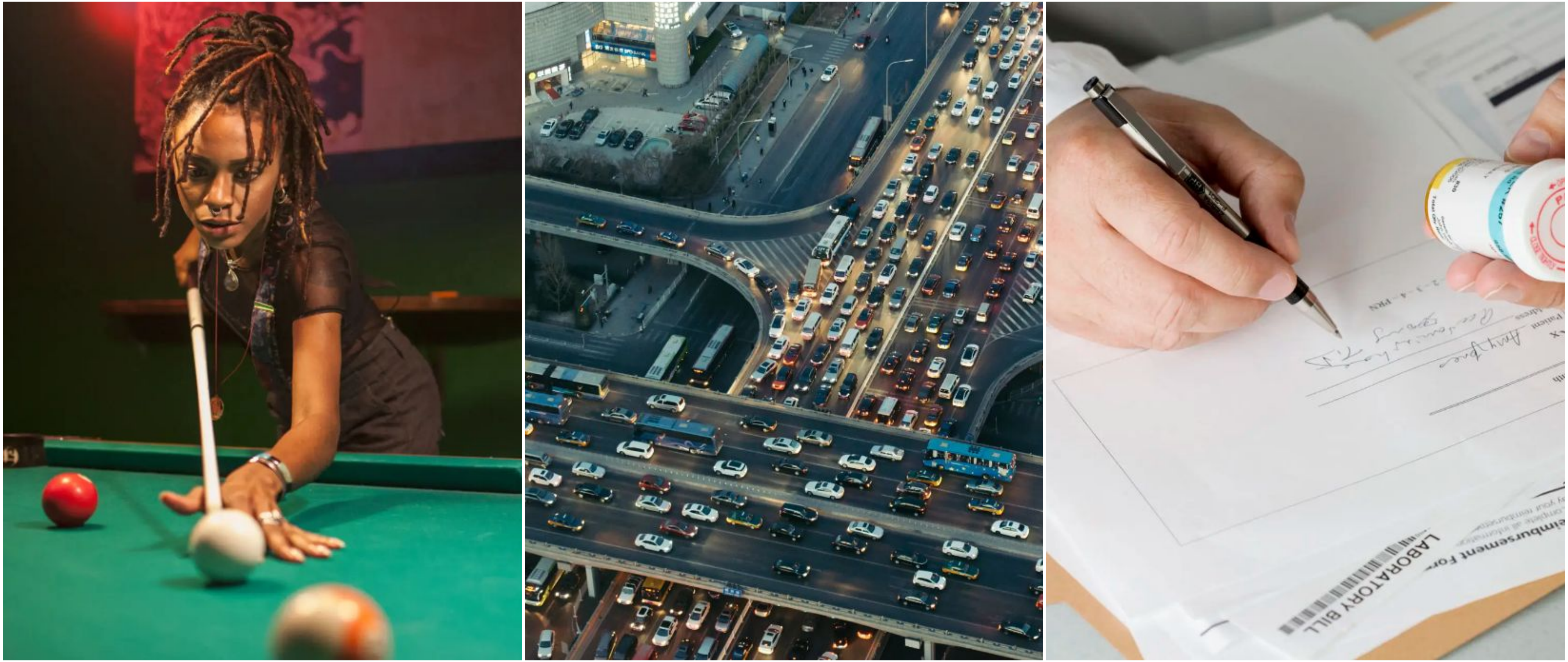
6.7920 Reinforcement Learning Foundations and Methods

Distributional Reinforcement Learning

Marc G. Bellemare

Mila



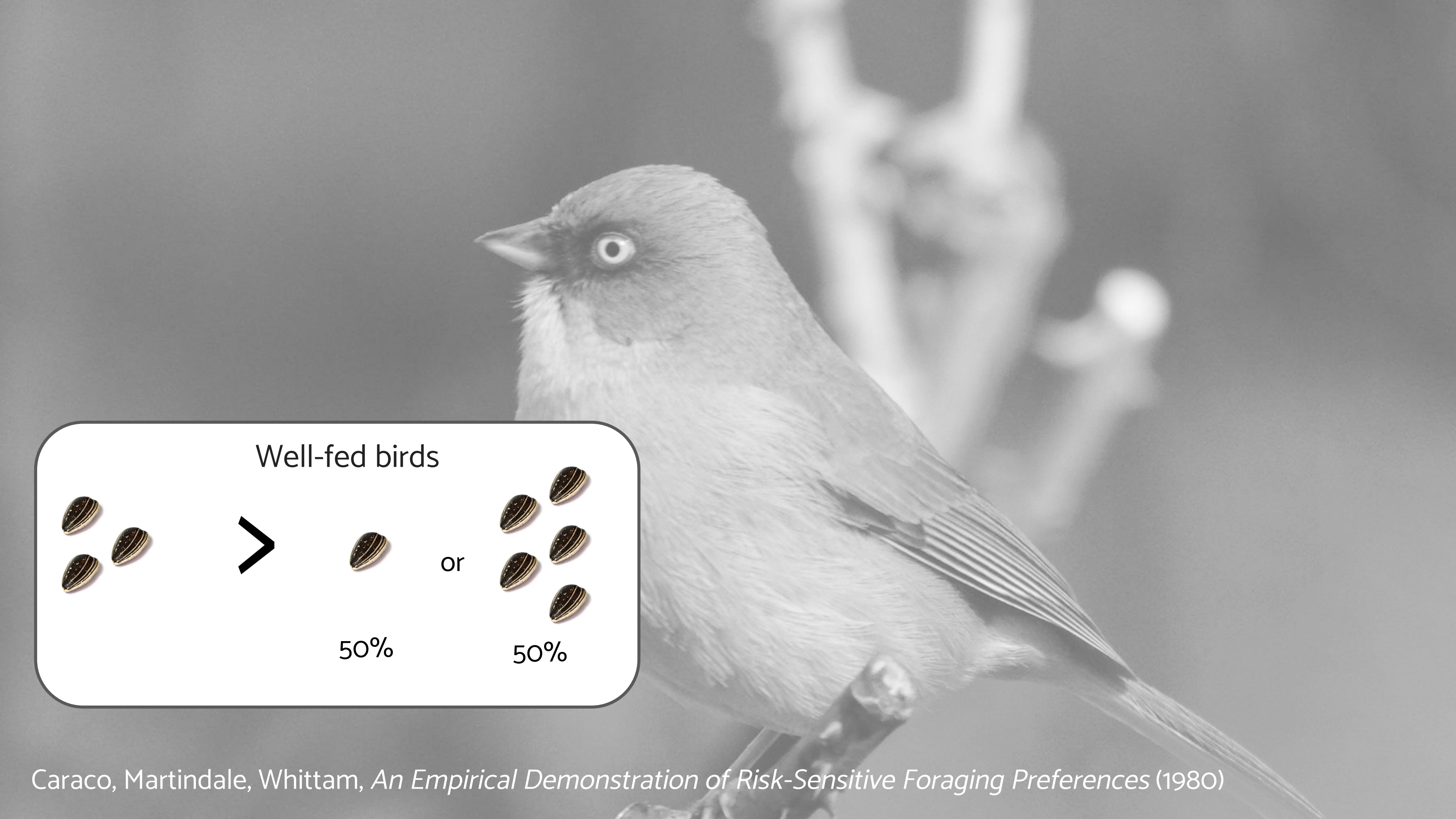


Randomness is an integral part of complex systems

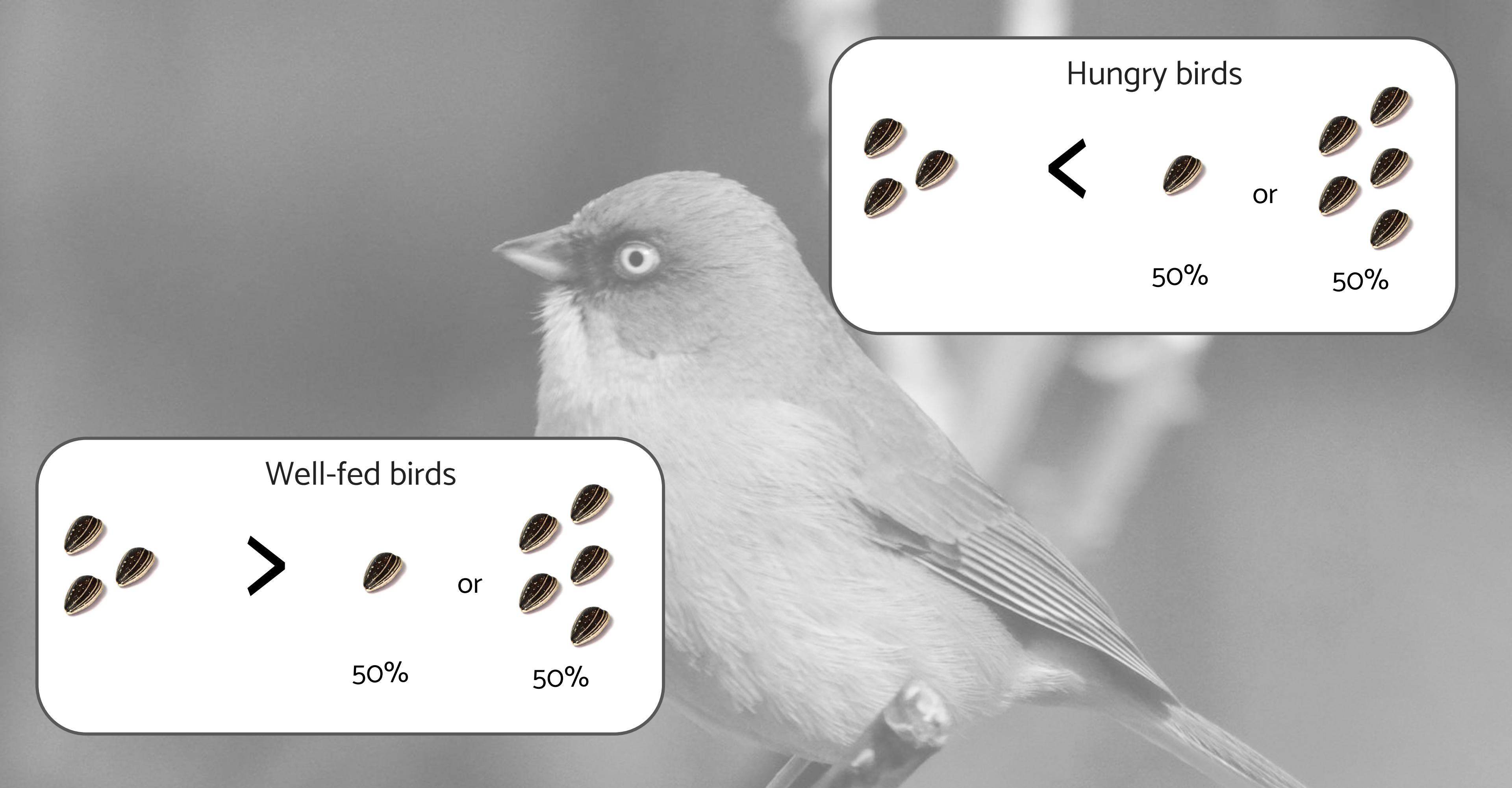
The general idea, and this is fairly unanimously accepted, is to use ***some average of the possible outcomes as a measure of the value of a policy****.*

Bellman, 1957

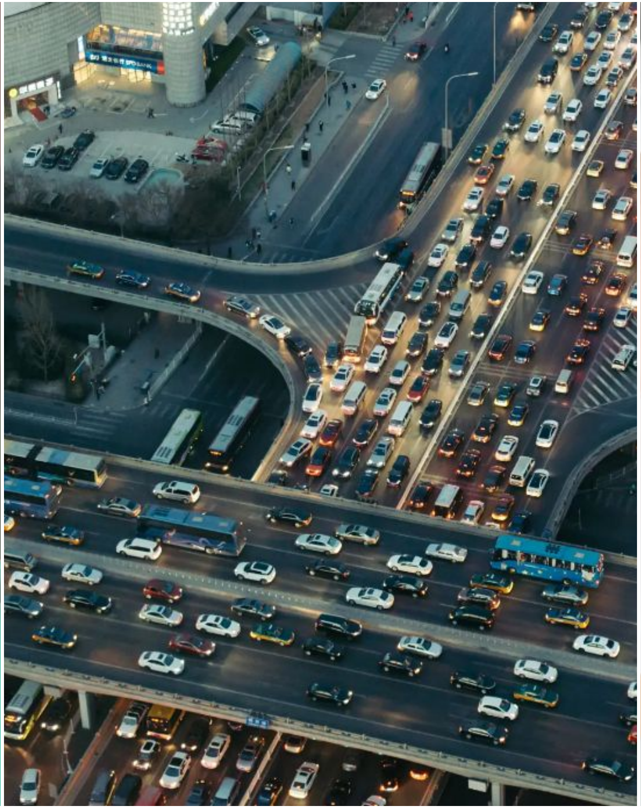
Caraco, Martindale, Whittam, *An Empirical Demonstration of Risk-Sensitive Foraging Preferences* (1980)

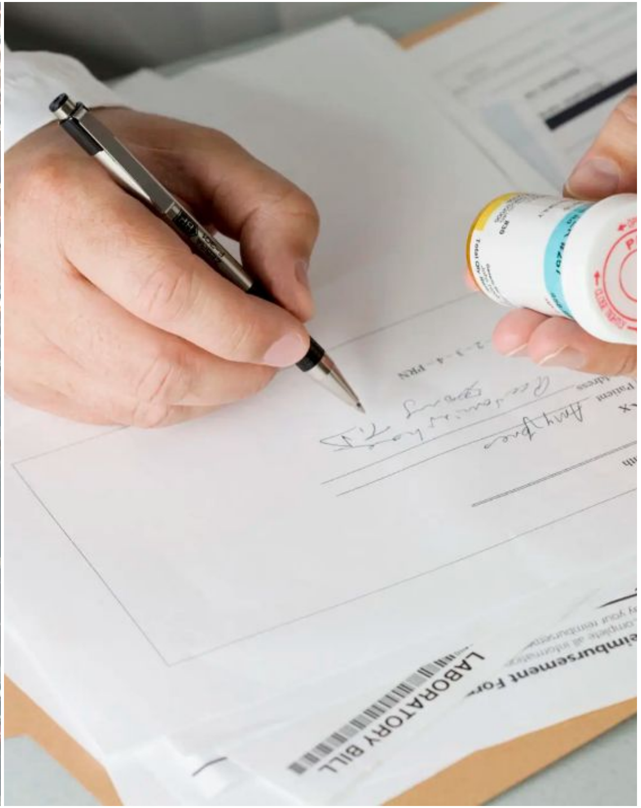


Caraco, Martindale, Whittam, *An Empirical Demonstration of Risk-Sensitive Foraging Preferences* (1980)



Caraco, Martindale, Whittam, *An Empirical Demonstration of Risk-Sensitive Foraging Preferences* (1980)

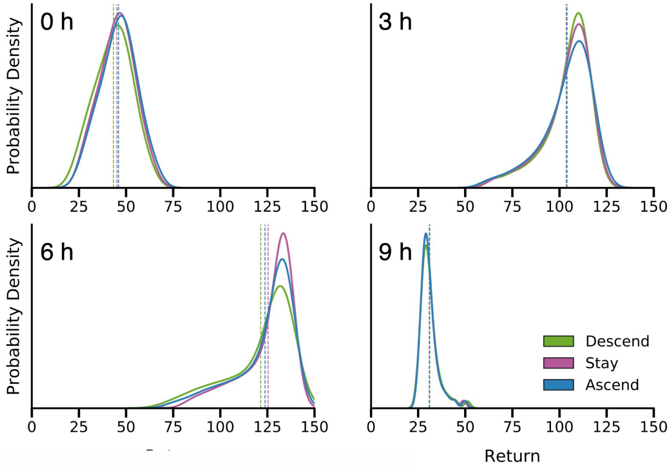


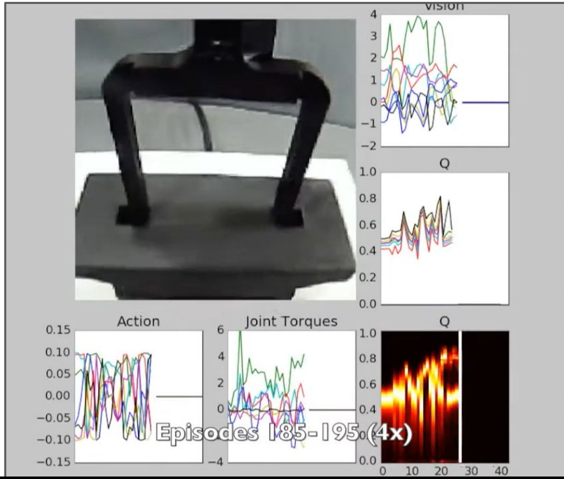


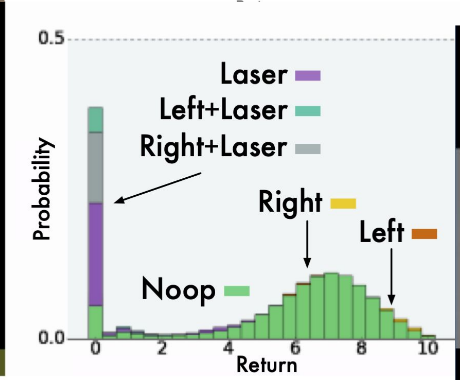
Randomness is an integral part of complex systems,
and is not well-characterized by expected values

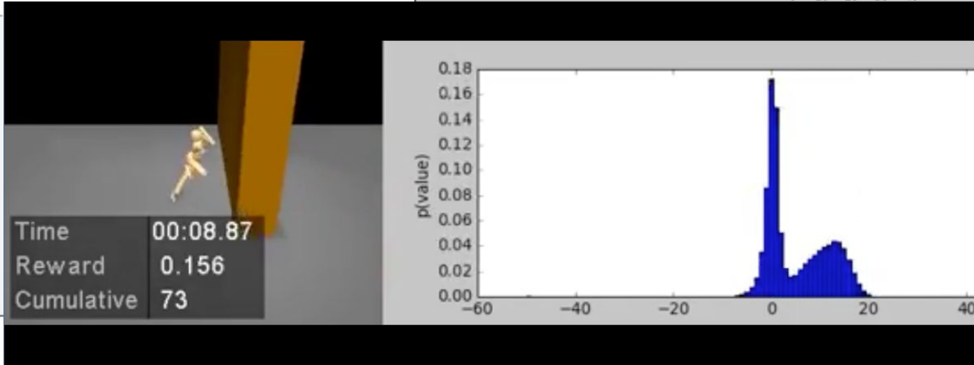
Distributional reinforcement learning is …

the study and design of RL algorithms that treat randomness as the key quantity of interest







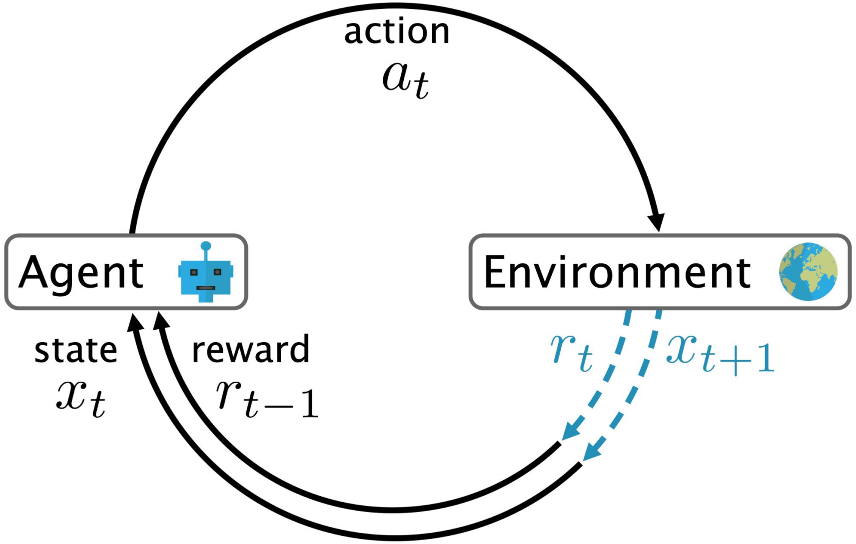


Mathematical framework

Making it practical

Risk-sensitive control

Markov decision processes



The generative equations

$$X_0 \sim \xi_0$$

Initial state distribution

$$A_t \sim \pi(\cdot \,|\, X_t)$$

Policy

$$R_t \sim P_{\mathcal{R}}(\cdot \,|\, X_t, A_t)$$

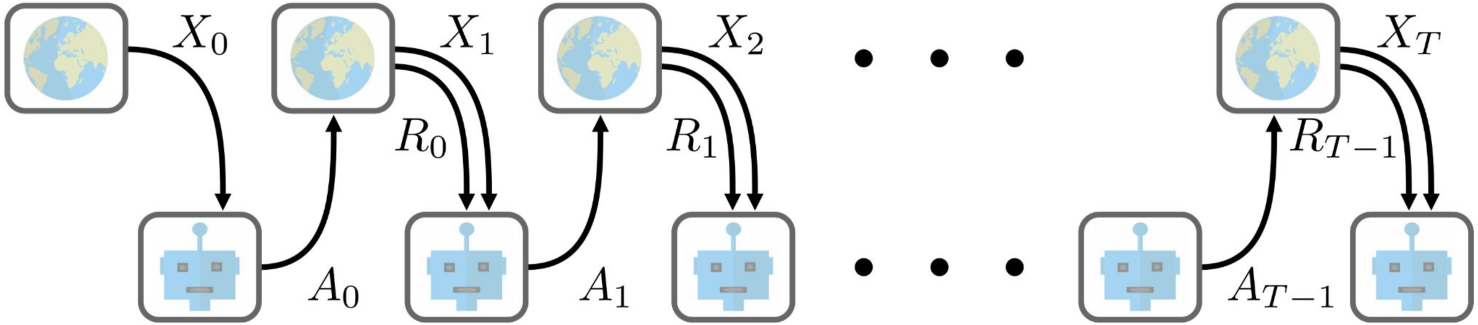
Reward distribution

$$X_{t+1} \sim P_{\mathcal{X}}(\cdot \,|\, X_t, A_t)$$

Transition kernel

Sometimes:

$$R_t, X_{t+1} \sim P(\cdot \,|\, X_t, A_t)$$



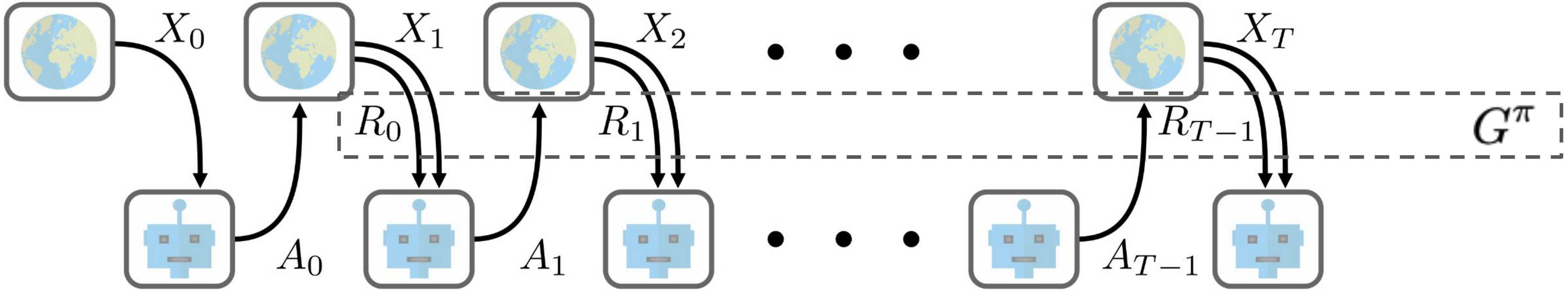
$$\left.\begin{aligned} X_0 &= x \\ A_t &\sim \pi(\cdot \mid X_t) \\ R_t &\sim P_{\mathcal{R}}(\cdot \mid X_t, A_t) \\ X_{t+1} &\sim P_{\mathcal{X}}(\cdot \mid X_t, A_t) \end{aligned}\right\} \mathbb{P}_\pi(\cdot \mid X_0 = x)$$

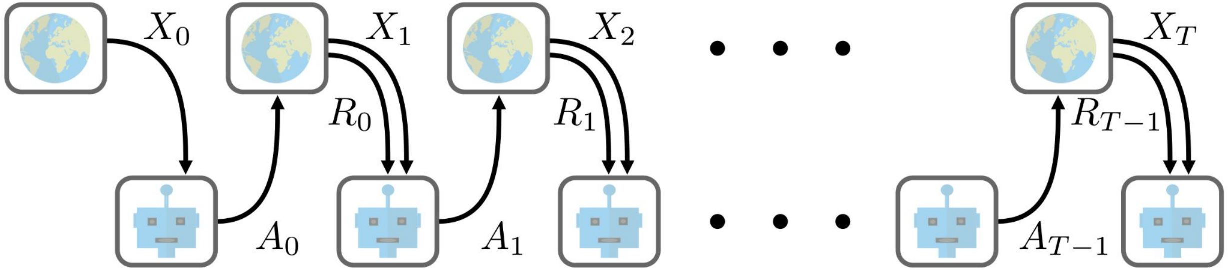
Joint distribution

The random return

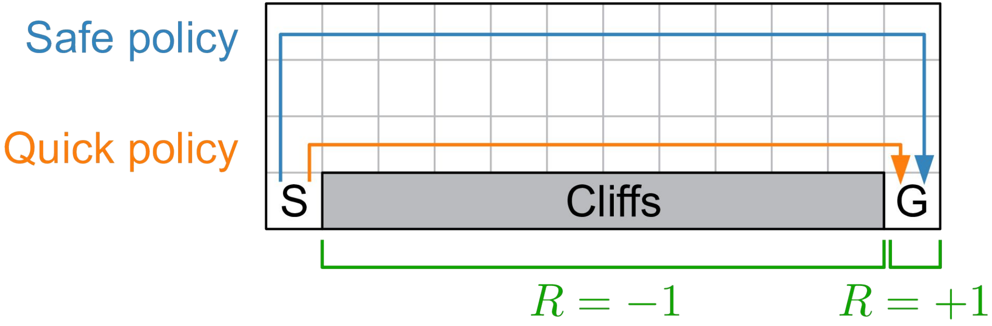
$$G^\pi(x) = \sum_{t=0}^{\infty} \gamma^t R_t, \quad X_0 = x$$

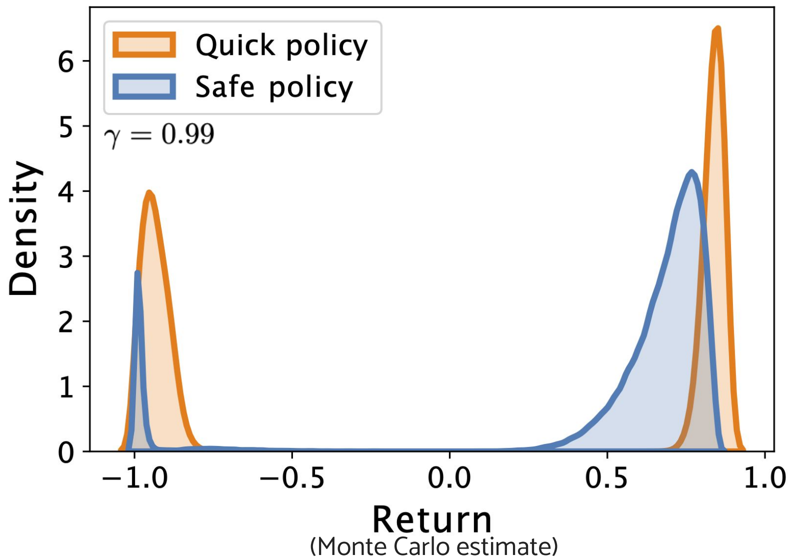
$\gamma \in [0, 1]$





$$G^{\pi}(x) = \sum_{t=0}^{\infty} \gamma^t R_t$$





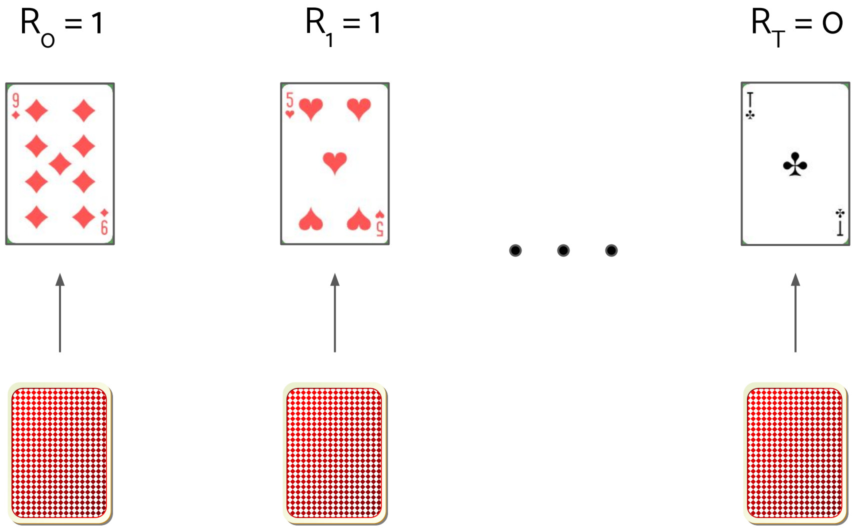
Immediate questions

How do the parameters of an MDP affect the distribution of random returns?

Can we adapt the language of reinforcement learning to distributions?

How can we learn to predict the random return (from data)?

Example 1: One-card solitaire

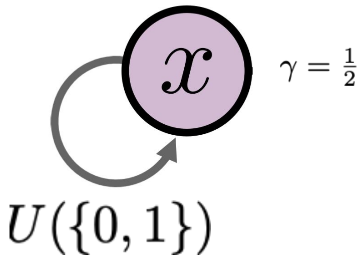


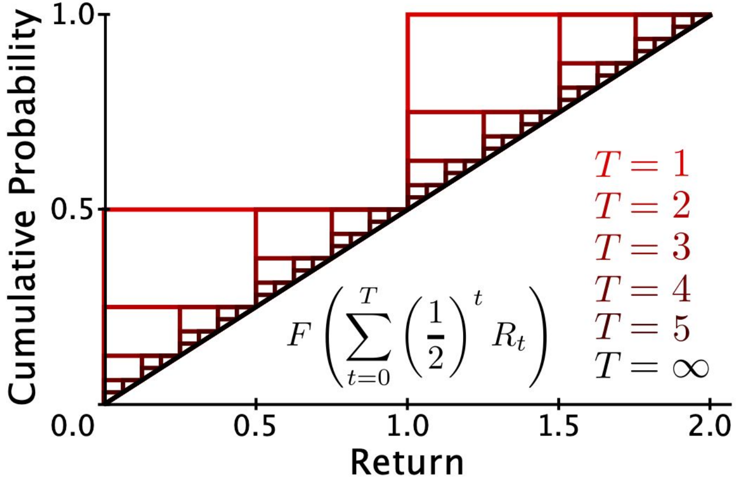
What is the probability distribution of G^{π} when sampling ...

- With replacement?
- Without replacement? (offline)

$\gamma = 1$

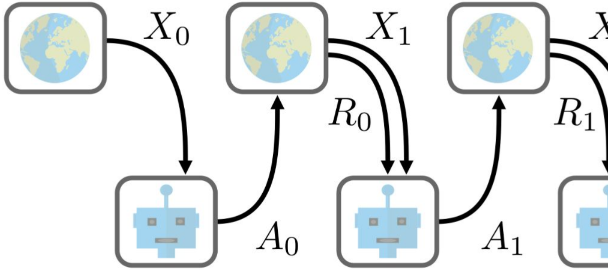
Example 2: Bernoulli rewards, 1/2 discount





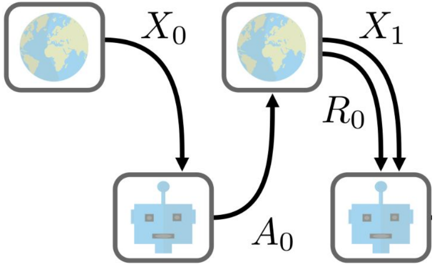
Value function:

$$V^\pi(x) = \mathbf{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_t \,|\, X_0 = x\right]$$



Bellman equation:

$$V^\pi(x) = \mathbf{E}_\pi\left[R + \gamma V^\pi(X') \,|\, X = x\right]$$



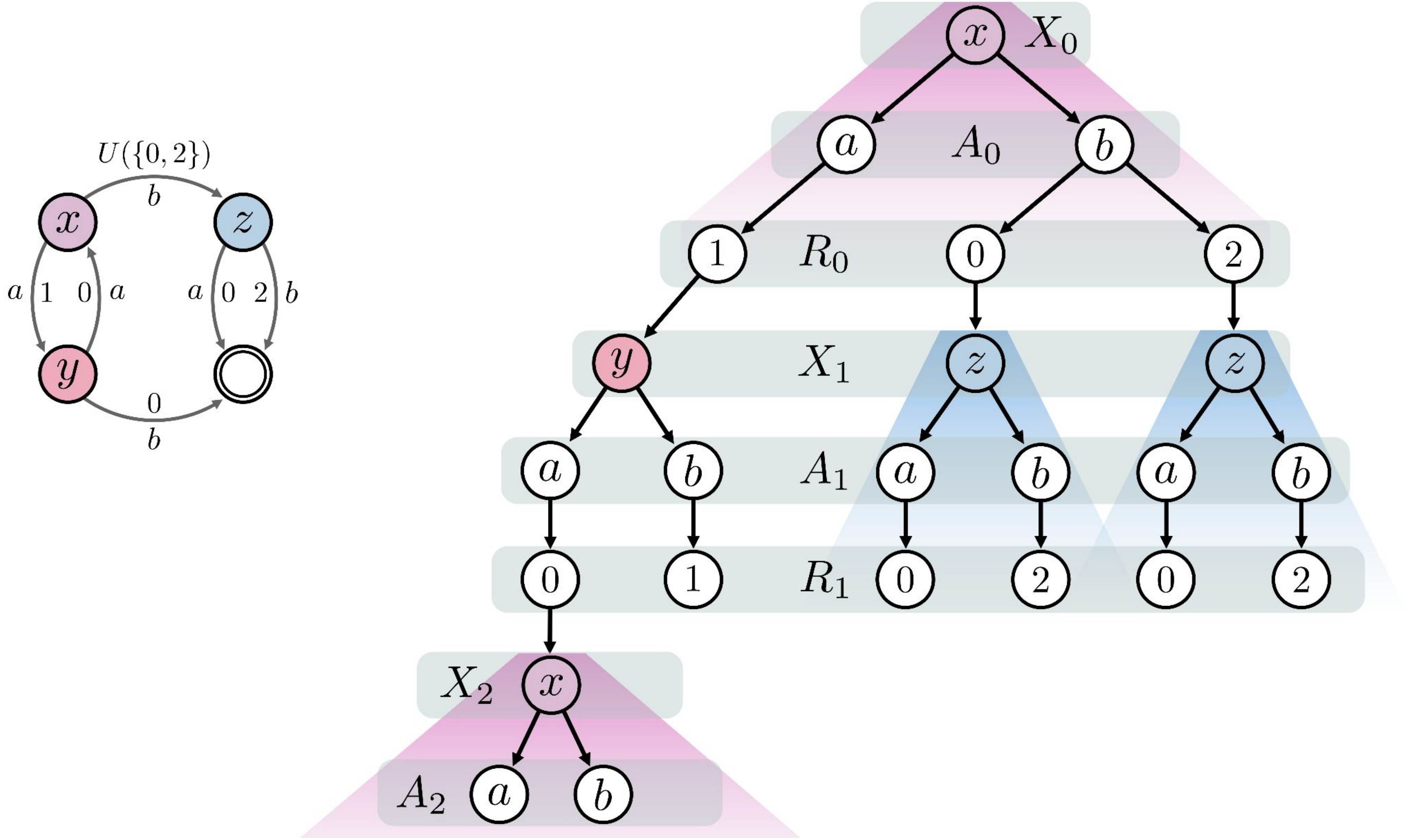
The value function is given *recursively*
in terms of immediate reward & next state

The random-variable Bellman equation

$$G^\pi(x) = \sum_{t=0}^{\infty} \gamma^t R_t, \quad X_0 = x$$

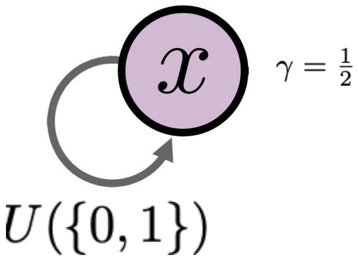
$$G^\pi(x) \overset{\mathcal{D}}{=} R + \gamma G^\pi(X'), \quad X = x$$

Equality in distribution



U({0,2})
b
x
z
a 1 0 a
a 0 2 b
y
0
b
x X_0
a A_0 b
1 R_0 0 2
y X_1 z z
a b A_1 a b a b
0 1 R_1 0 2 0 2
X_2 x
A_2 a b

Why equality in distribution matters

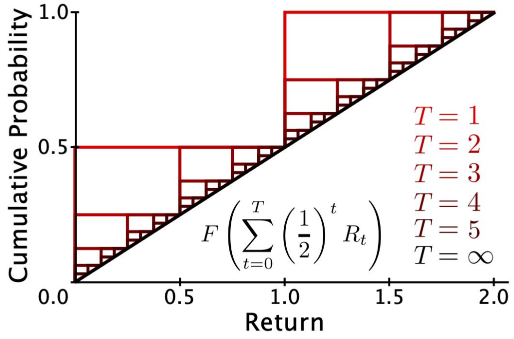


$$G(x) = R + \gamma G(x)$$

$$G(x) \overset{\mathcal{D}}{=} R + \gamma G(x)$$

An alternative? The CDF Bellman equation

$$F_{G^\pi(x)}(z) = \mathbf{E}_\pi \left[F_{G^\pi(X')}\left(\frac{z - R}{\gamma}\right) \middle| X = x \right]$$



Even better: Return-distribution functions

Random variables are convenient – but sometimes hard to work with

We really only care about the relationship between their *distributions*

Idea: express random-variable equation directly in terms of distributions

$$G^\pi(x) \sim \eta^\pi(x)$$

Collection of return distributions

The distributional Bellman equation

$$G^\pi(x) \overset{\mathcal{D}}{=} R + \gamma G^\pi(X'), \quad X = x$$

$$\downarrow$$

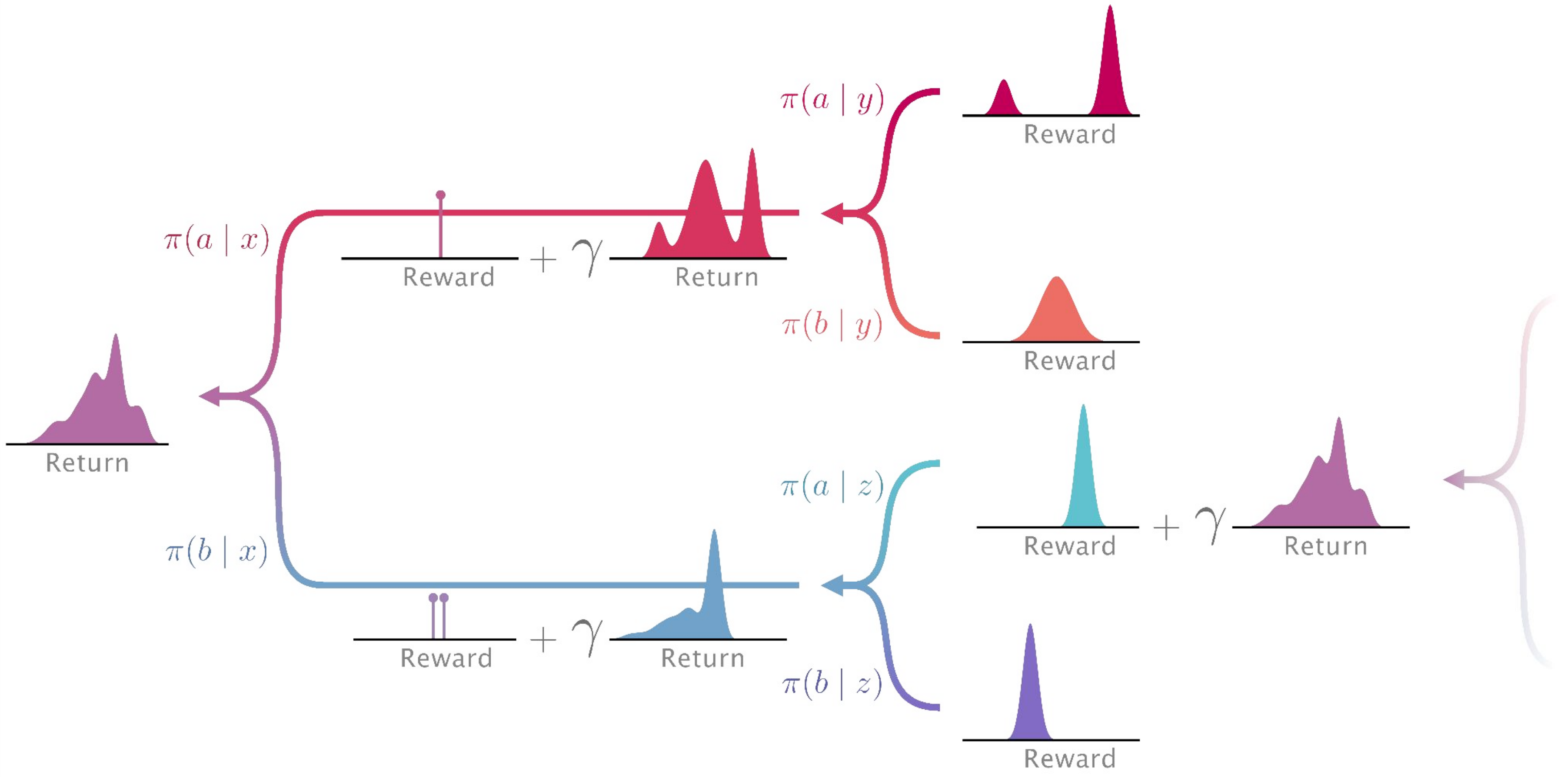
$$\eta^\pi(x) = \mathbf{E}_\pi\left[(\mathrm{b}_{R,\gamma})_\# \eta^\pi(X') \,|\, X = x\right]$$

Key operations

$$\eta^\pi(x) = \mathbf{E}_\pi \left[\boxed{(\mathrm{b}_{R,\gamma})_\#} \eta^\pi(X') \,|\, X = x \right]$$ Scaling (a) and shifting (b)

$$\eta^\pi(x) = \mathbf{E}_\pi \left[(\mathrm{b}_{R,\gamma})_\# \boxed{\eta^\pi(X')} \,|\, X = x \right]$$ Indexing (c)

$$\eta^\pi(x) = \boxed{\mathbf{E}_\pi} \left[(\mathrm{b}_{R,\gamma})_\# \eta^\pi(X') \,|\, X = x \right]$$ Mixing (d)



$\pi(a \mid y)$
Reward
$\pi(a \mid x)$
Reward
$+ \gamma$
Return
$\pi(b \mid y)$
Reward
Return
$\pi(a \mid z)$
Reward
$+ \gamma$
Return
$\pi(b \mid x)$
Reward
$+ \gamma$
Return
$\pi(b \mid z)$
Reward

Solving the (standard) Bellman equation

The value function:

$$V^\pi(x) = \mathbf{E}\left[G^\pi(x)\right]$$

Bellman operator:

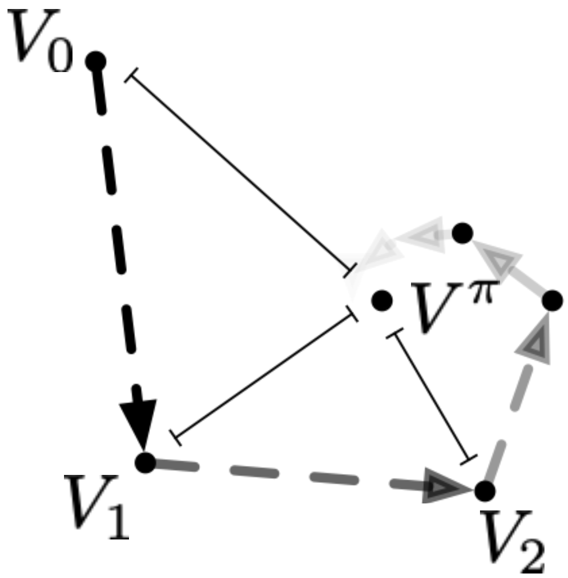
$$(T^\pi V)(x) = \mathbf{E}_\pi\left[R + \gamma V(X') \,|\, X = x\right]$$

Dynamic programming:

$$V_{k+1}(x) = (T^\pi V_k)(x)$$

Fundamental contractive result:

$$\|V_{k+1} - V^\pi\|_\infty \leq \gamma \|V_k - V^\pi\|_\infty$$



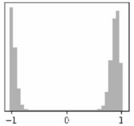
The distributional Bellman operator

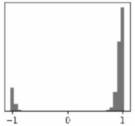
Transform a collection of probability distributions into a new collection:

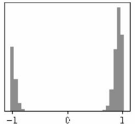
$$(\mathcal{T}^\pi \eta)(x) = \mathbf{E}_\pi \left[(\mathrm{b}_{R,\gamma})_\# \eta(X') \,|\, X = x \right]$$

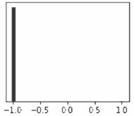
More explicitly:

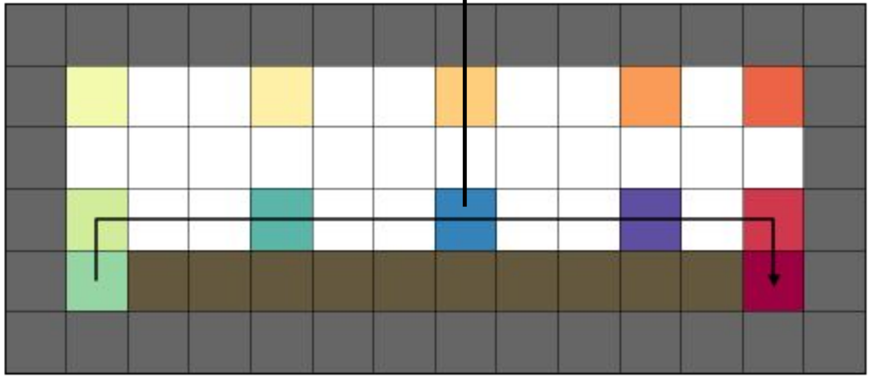
$$(\mathcal{T}^\pi \eta)(x) = \sum_{a \in \mathcal{A}} \sum_{r \in \mathcal{R}} \sum_{x' \in \mathcal{X}} \mathbb{P}_\pi(A = a, R = r, X' = x' \,|\, X = x) \eta(x')$$

$(\mathcal{T}^\pi \eta)(x)$ = P(up | x) 

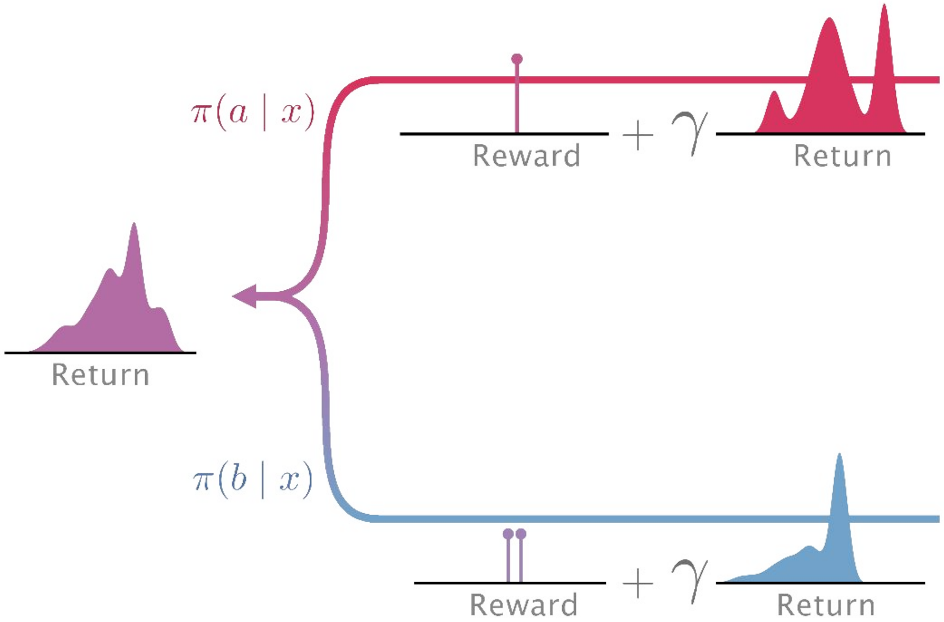
+ P(right | x) 

+ P(left | x) 

+ P(down | x) 



$$\eta^\pi(x) = (\mathcal{T}^\pi \eta^\pi)(x)$$



Solving the distributional Bellman equation

Random-variable Bellman equation:

$$G^\pi(x) \overset{\mathcal{D}}{=} (\mathcal{T}^\pi G^\pi)(x)$$

Random-variable operator:

$$(\mathcal{T}^\pi G)(x) \overset{\mathcal{D}}{=} R + \gamma G(X'), \quad X = x$$

Distributional dynamic programming:

$$G_{k+1}(x) \overset{\mathcal{D}}{=} (\mathcal{T}^\pi G_k)(x)$$

Contraction in *Wasserstein distance*:

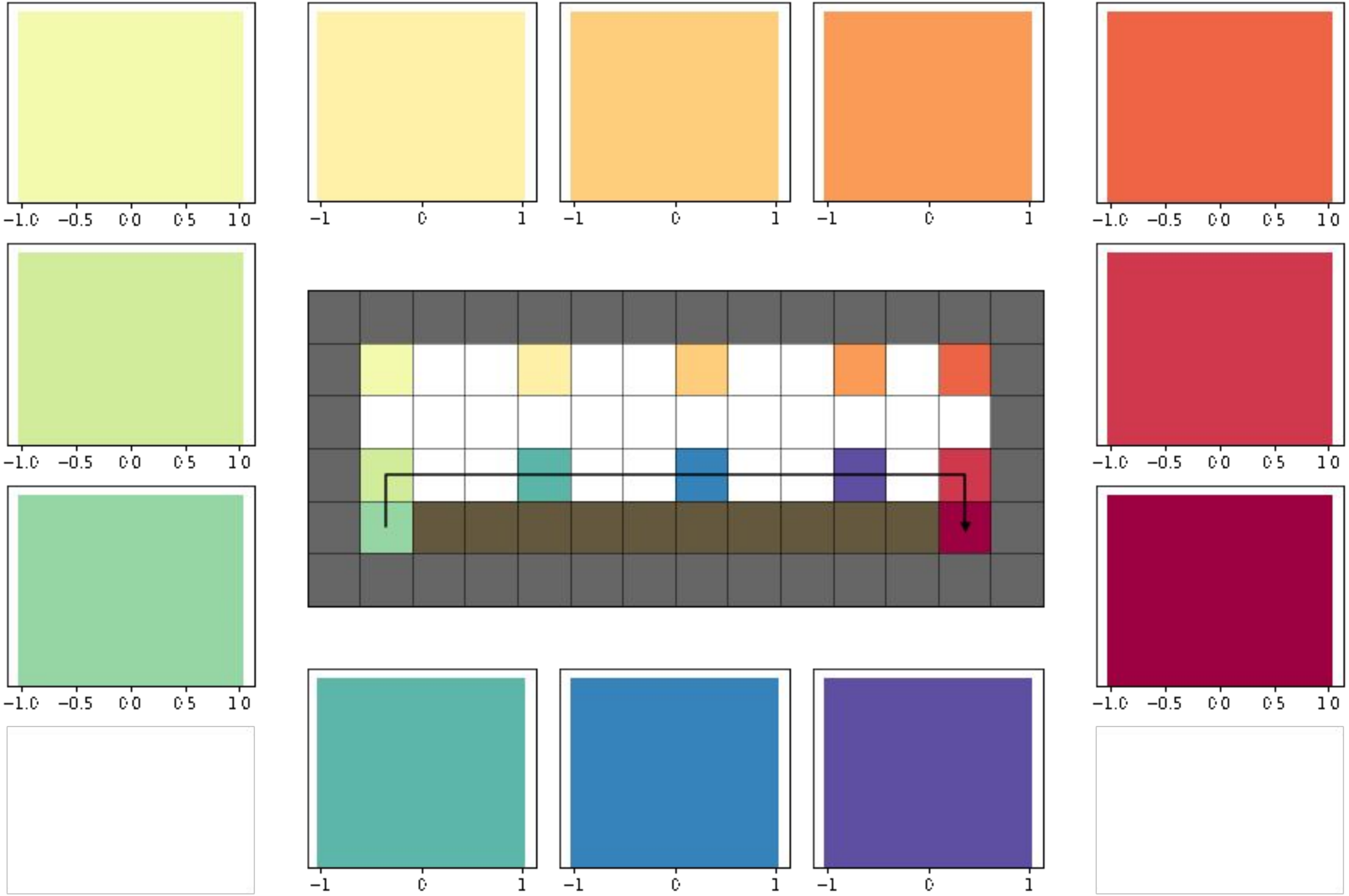
$$\sup_{x \in \mathcal{X}} w_p\big(G_{k+1}(x), G^\pi(x)\big) \leq \gamma \sup_{x \in \mathcal{X}} w_p\big(G_k(x), G^\pi(x)\big)$$

$$p \in [1, \infty]$$

Hence:

$$\lim_{k \to \infty} G_k(x) \overset{\mathcal{D}}{=} G^\pi(x)$$

(or equivalently, with $\eta^\pi(x)$)


−1.0 −0.5 0.0 0.5 1.0
−1 0 1
−1 0 1
−1 0 1
−1.0 −0.5 0.0 0.5 1.0
−1.0 −0.5 0.0 0.5 1.0
−1.0 −0.5 0.0 0.5 1.0
−1.0 −0.5 0.0 0.5 1.0
−1.0 −0.5 0.0 0.5 1.0
−1 0 1
−1 0 1
−1 0 1

Mathematical framework

Making it practical

Risk-sensitive control

Can we **compute** the return-distribution function?

A few challenges

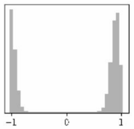
Probability distributions are **infinite-dimensional**; need to worry about

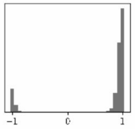
- Memory
- Efficient backups
- Estimating from samples
- NP-hardness (Mannor and Tsitsiklis 2011, 2013, BDR 2022)

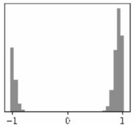
Given a **finite-memory approximation** scheme...

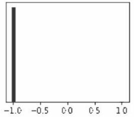
- How accurate is it?
- Can it be computed efficiently?
- Does it result in a contractive map?

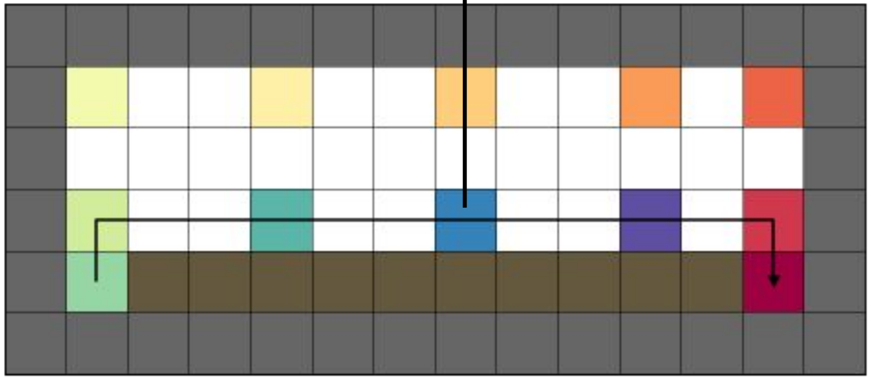
The “growing support” problem

$(\mathcal{T}^\pi\eta)(x)$ = P(up | x) 

+ P(right | x) 

+ P(left | x) 

+ P(down | x) 



If each next-state distribution takes on *m* values, then $(\mathcal{T}^\pi\eta)(x)$ may take on *4m* values

The empirical representation

Represent a distribution

$$\eta(x) = \sum_{i=1}^{m(x)} p_i(x)\delta_{\theta_i(x)}$$

Using 1 + 2 m(x) parameters for state *x*?

Could grow exponentially!

An alternative: the categorical representation

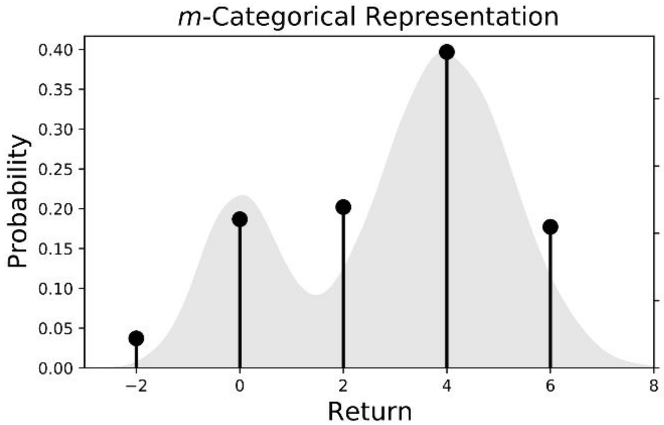
Fix number of particles to *m*

Keep them in **fixed locations**

$$\theta_i = V_{\text{MIN}} + \underbrace{\frac{V_{\text{MAX}} - V_{\text{MIN}}}{m-1}}_{\varsigma_m}(i-1)$$

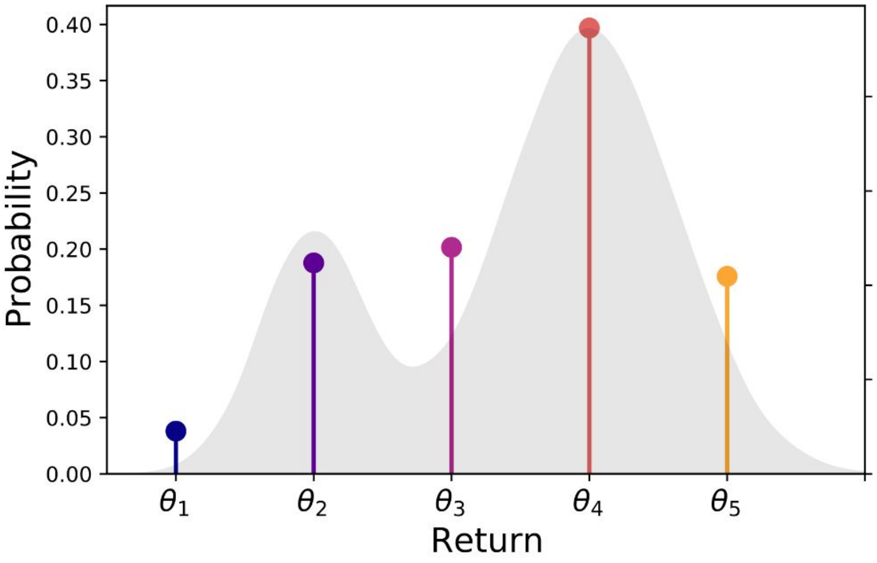
Parametrize probability at each location:

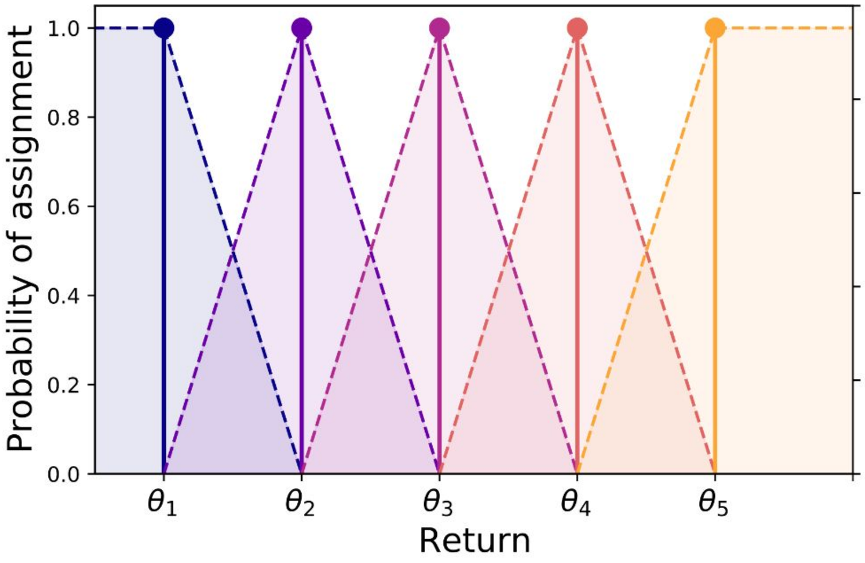
$$\eta(x) = \sum_{i=1}^{m} p_i(x)\delta_{\theta_i}$$

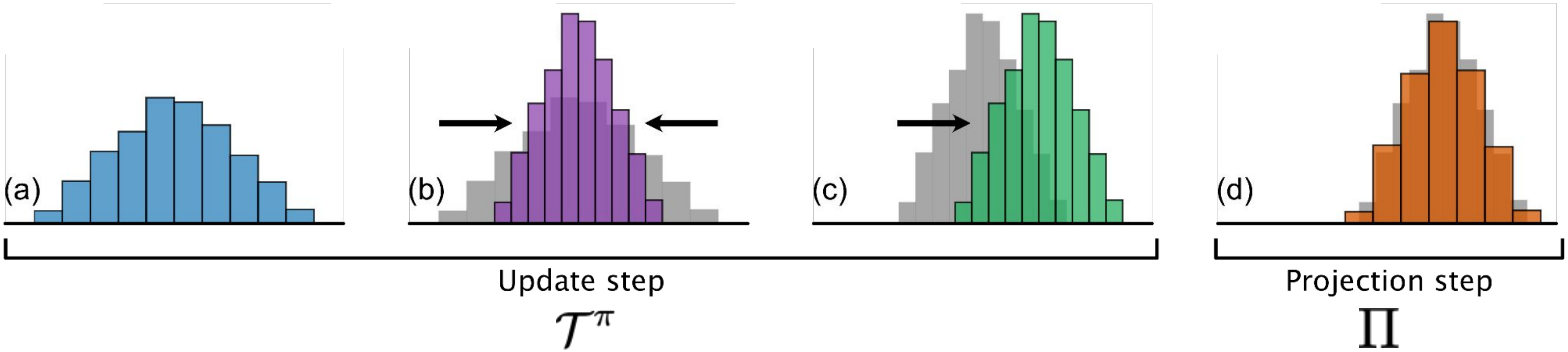


Solving the support problem with the categorical projection

“Assign probability mass in proportion to the distance to the nearest locations”







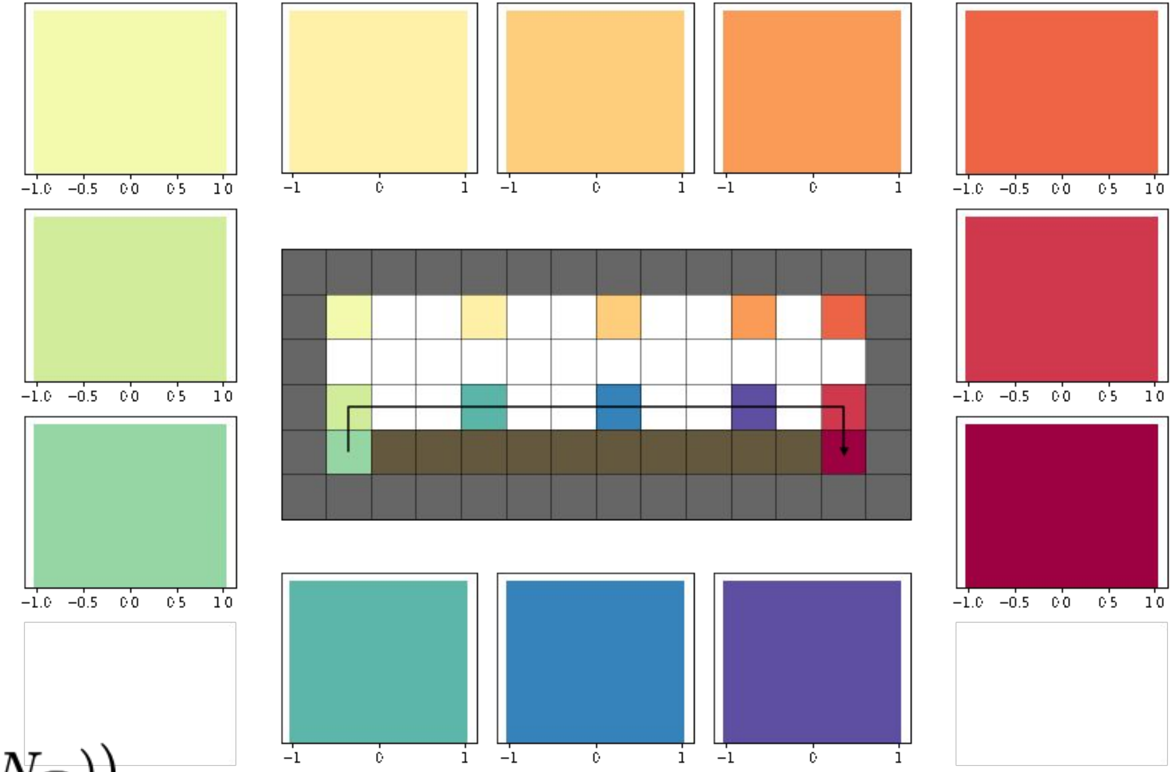
Categorical dynamic programming:

$$G_{k+1}(x) \stackrel{\mathcal{D}}{=} \left(\Pi \mathcal{T}^\pi G_k\right)(x)$$

Categorical dynamic programming

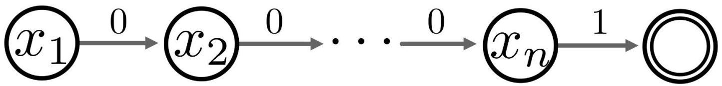
Pick initial G_0
For *k* = 0, 1, ...

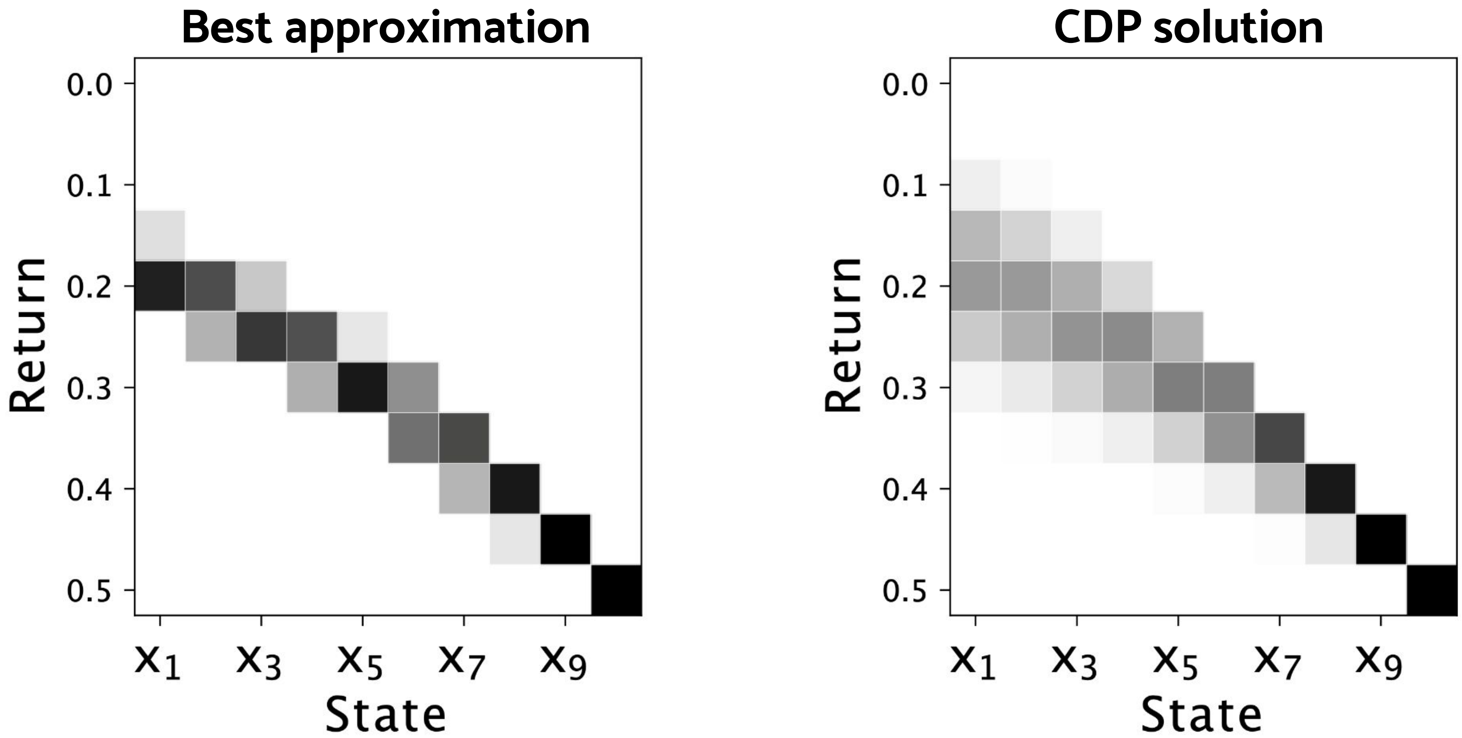
$$\tilde{G}(x) \leftarrow (\mathcal{T}^\pi G_k)(x)$$
$$G_{k+1}(x) \leftarrow \Pi \tilde{G}(x) \quad \forall x$$



Per iteration: $O\big(mN_{\mathcal{X}}N_{\mathcal{A}}(N_{\mathcal{X}} + N_{\mathcal{R}})\big)$

Diffusion error (due to projection)





How good is this approximation?

Measured in Cramér (l_2) distance

Convergence rate

If $K \geq \dfrac{\log\left(\frac{1}{\varepsilon}\right) + \log \bar{\ell}_2(G_0, G^\pi_{\mathcal{C}})}{c \log\left(\frac{1}{\gamma}\right)}$

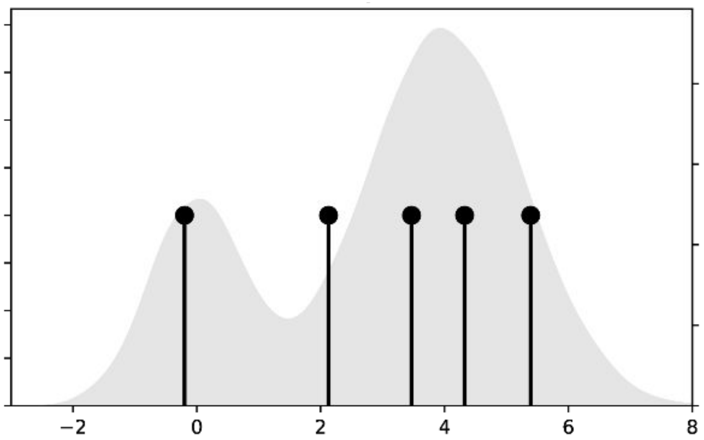
Then $\bar{\ell}_2(G_K, G^\pi_{\mathcal{C}}) < \varepsilon$

Approximation error

$$\bar{\ell}_2(G^\pi_{\mathcal{C}}, G^\pi) \leq \frac{V_{\text{MAX}} - V_{\text{MIN}}}{(m-1)\left(1 - \gamma^{\frac{1}{2}}\right)}$$

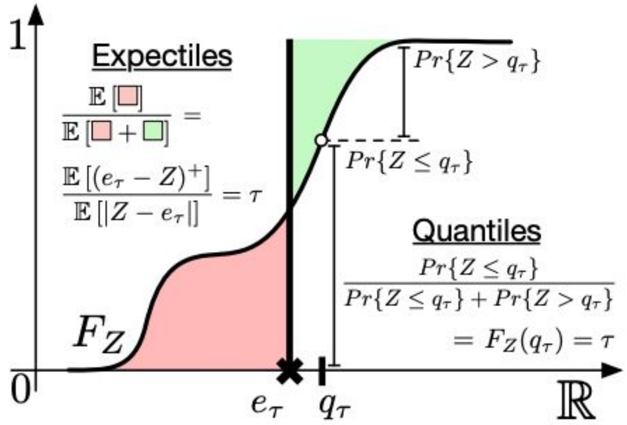
A few other effective representations

Quantiles



- **"Transpose"** of categorical representation
- Projection is contraction in Wasserstein distance
- Learning from samples requires care
- Ignores distribution tails

Expectiles



- **Smooth version** of quantiles, handles tails
- Parameters no longer locations or probabilities
- Requires additional *imputation* machinery
- Convergence still not completely understood

Temporal-difference learning

- Start with some initial value function estimate V
- Given a sample transition (x, a, r, x'),

$$V(x) \leftarrow (1-\alpha)V(x) + \alpha(r + \gamma V(x'))$$

- TD learns one state at a time;
- By **incrementally updating** its value function.

Categorical temporal-difference learning

The distributional equivalent of TD learning

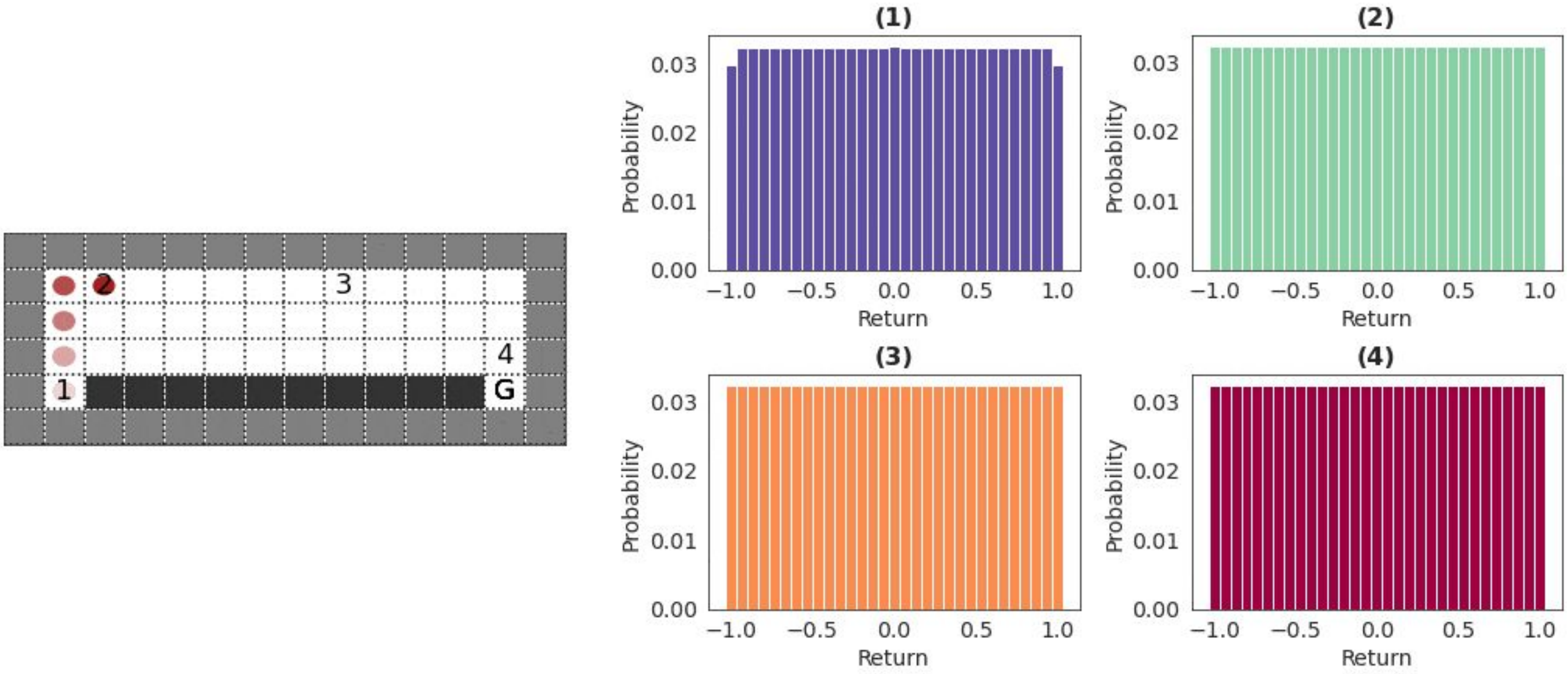
Combines TD update rule with distributional “ideas”:

m particles, projection, pushforward

$$\eta(x) \leftarrow (1-\alpha)\eta(x) + \alpha\Big((\Pi_C(\mathrm{b}_{r,\gamma})_\# \eta(x')\Big)$$

Instead of full update, take a small step “towards” categorical target

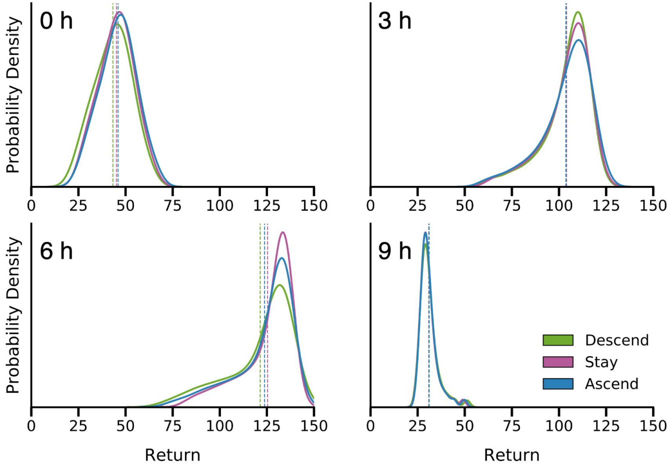
Keep in mind: **can’t add** probability distributions, only mix


1
2
3
4
G
(1)
(2)
(3)
(4)
Probability
Return
0.00
0.01
0.02
0.03
−1.0
−0.5
0.0
0.5
1.0

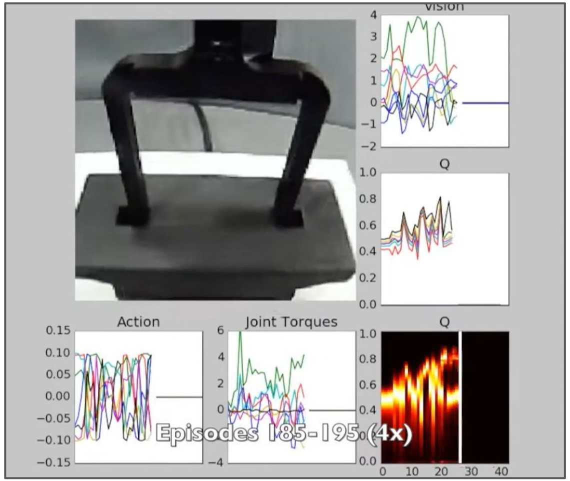
Mathematical framework

Making it practical

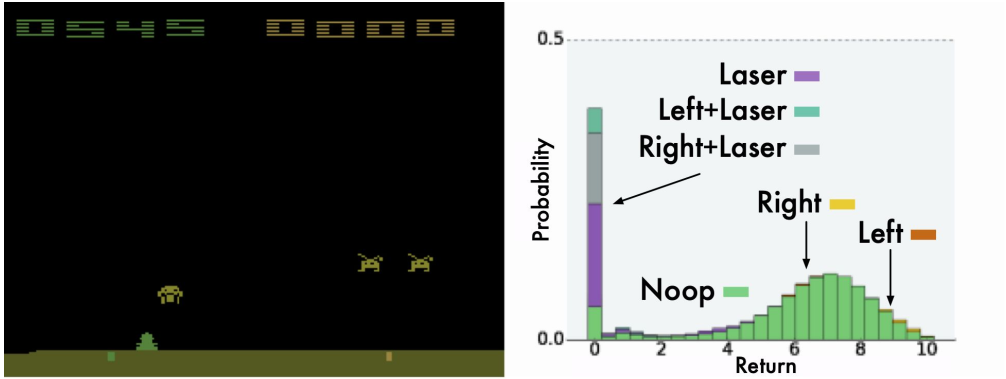
Risk-sensitive control



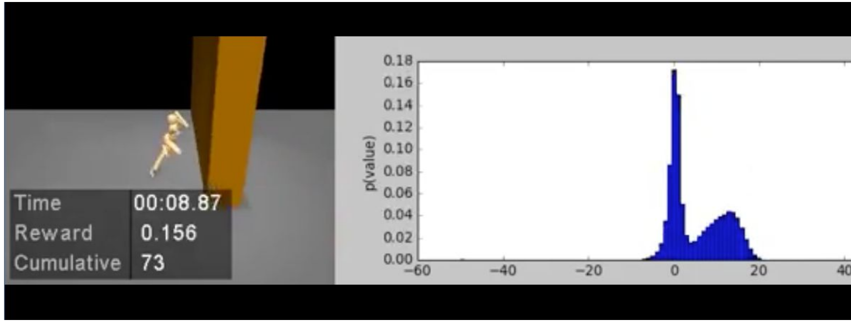
0 h
3 h
6 h
9 h
Probability Density
Return
0 25 50 75 100 125 150
Descend
Stay
Ascend



Vision
Q
Action
Joint Torques
Q
Episodes 185-195 (4x)



Laser
Left+Laser
Right+Laser
Right
Left
Noop
Probability
Return
0.5
0.0
0 2 4 6 8 10



Time 00:08.87
Reward 0.156
Cumulative 73
p(value)

Risk-sensitive control

Find π maximizing

$$\rho\Big(\sum_{t=0}^{\infty}\gamma^t R_t\Big)$$

Where ϱ is a **risk measure**:

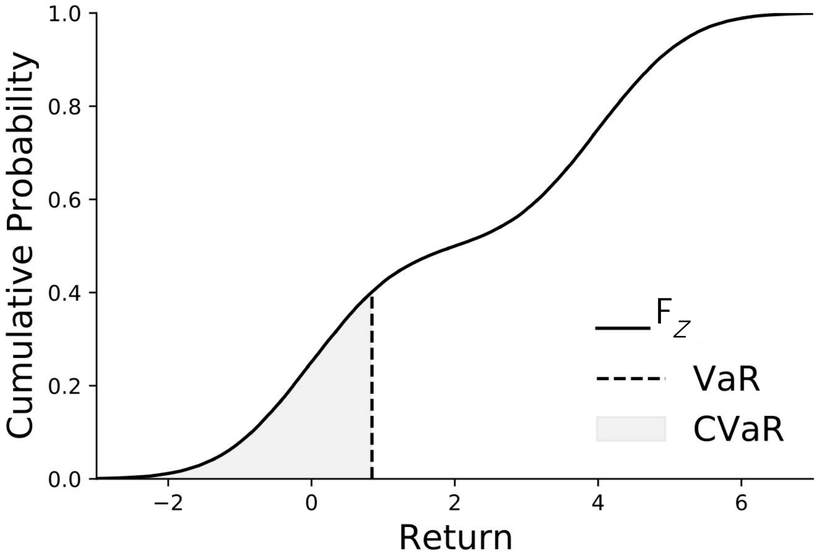
$$\rho : \mathscr{P}(\mathbb{R}) \to \mathbb{R}$$

Risk measures in RL

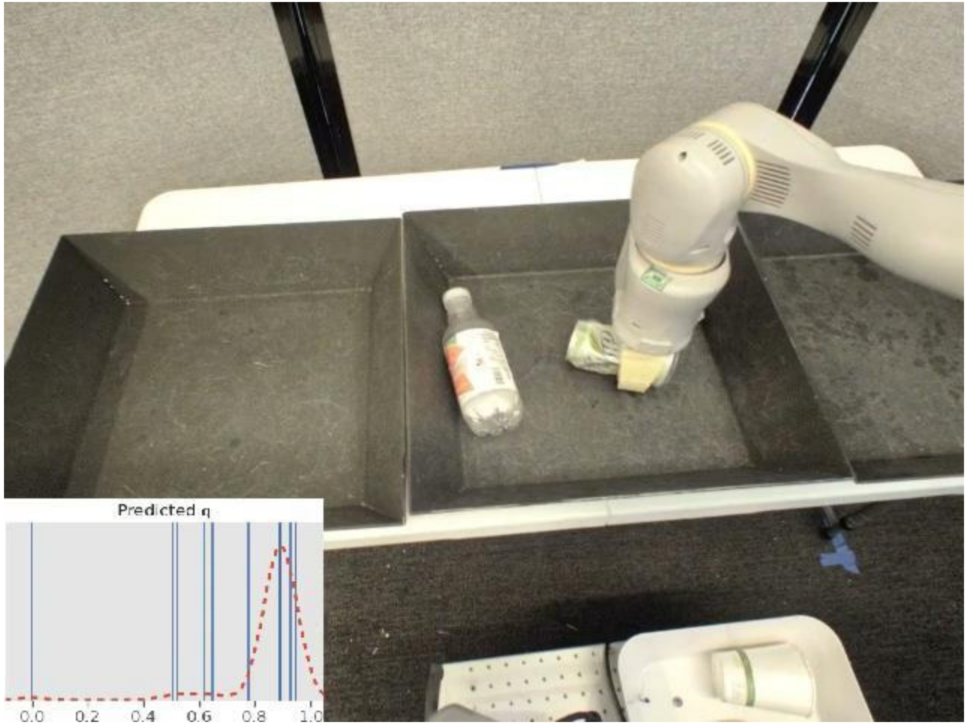
- Mean-variance criterion

$$\rho(G) = \mathbb{E}[G] - \lambda \mathbb{V}(G), \lambda \in [0, \infty)$$

- Value-at-risk
- Conditional value-at-risk
- And also:
  - Entropic risk
  - Entropic value-at-risk
  - Risk distortion metrics
  - etc.



Risk-sensitive grasping (Q2-Opt)



- **Task:**
  Object grasping from vision
  Discount factor + per-step penalty
  Large dataset of offline grasps (QT-Opt)

- **Non-trivial distributions** arise from …
  actuation noise,
  environment dynamics,
  limited sensors,
  function approximation,
  policy nonstationarity

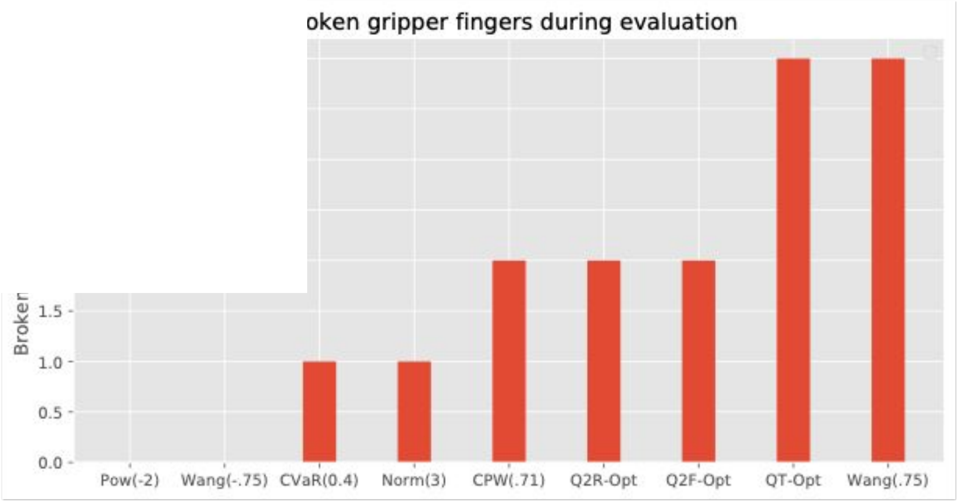
Bodnar et al., RSS 2020

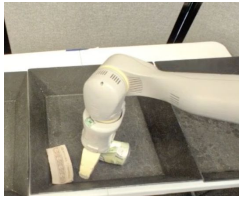
Model	Grasp Success Rate
QT-Opt	70.00%
Q2R-Opt	79.50%
Q2F-Opt	**82.00%**

Distributional

Risk-seekir

Risk-avers





Model	Grasp Success Rate
QT-Opt	70.00%
Q2R-Opt	79.50%
Q2F-Opt	**82.00%**

Bodnar et al. (2020)

Wurman et al. (2022)

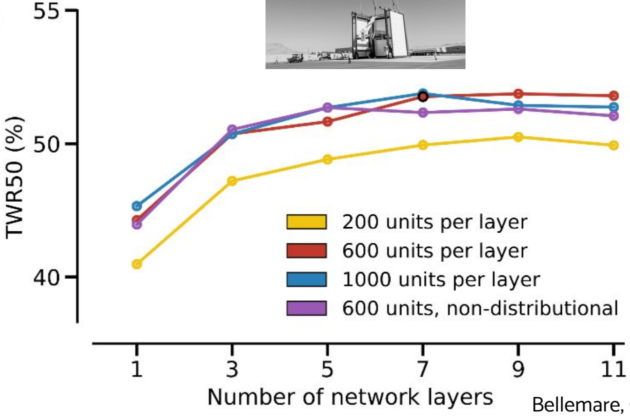
Agent	Normal	Hard	Unseen
Our approach	**80%**	**80%**	**50%**
No random watcher data	**80%**	70%	20%
Only lift data	0%	0%	0%
Non-distributional RL	30%	20%	10%

(a) `lift_green`

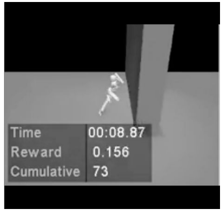
Agent	Normal	Hard	Unseen
Our approach	**60%**	**40%**	**40%**
No random watcher data	50%	30%	30%
Only stacking data	0%	10%	0%
Non-distributional RL	20%	0%	0%

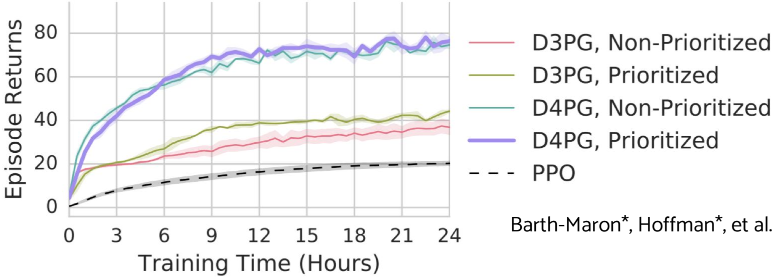
(b) `stack_green_on_red`.

Cabi et al. (2020)

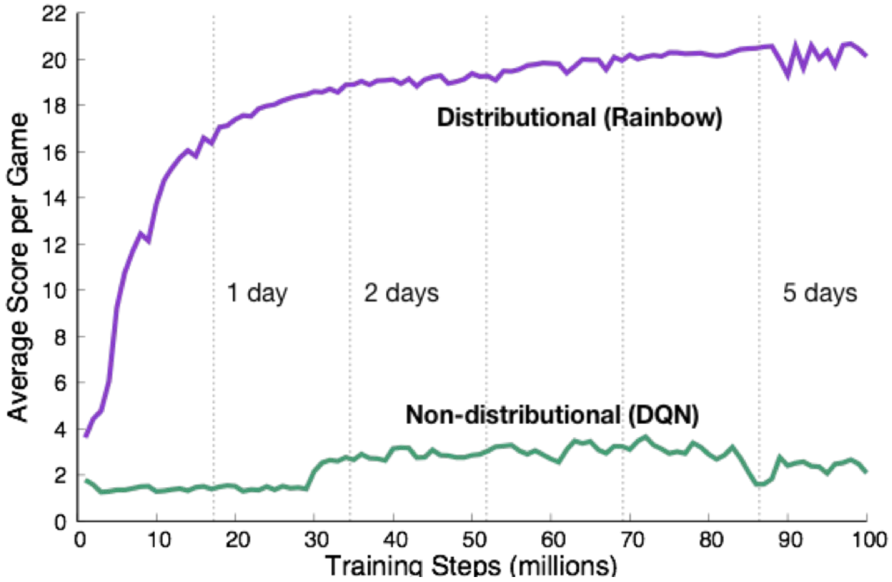


Bellemare, Candido, et al. (2020)



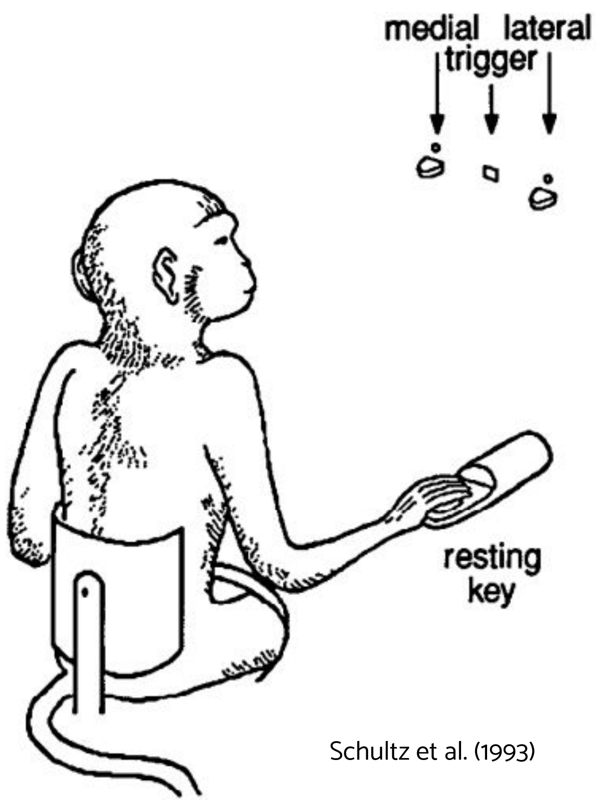


Barth-Maron*, Hoffman*, et al. (2018)

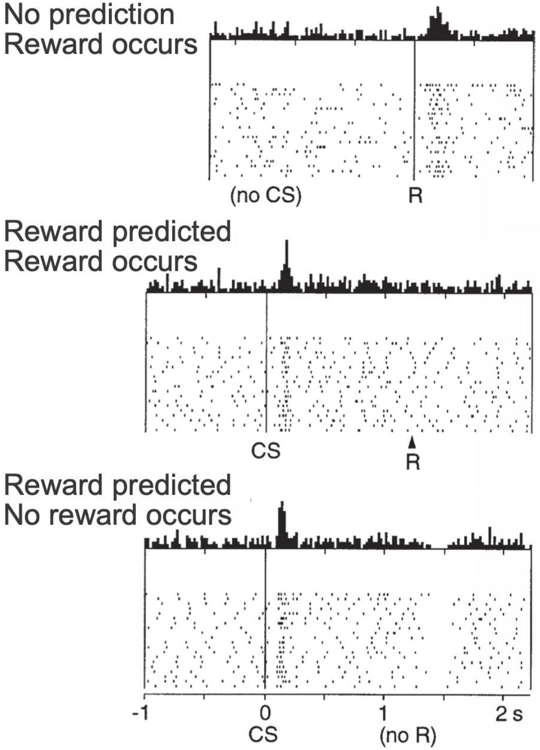


Bard*, Foerster*, et al. (2020)

#3: Understanding the brain

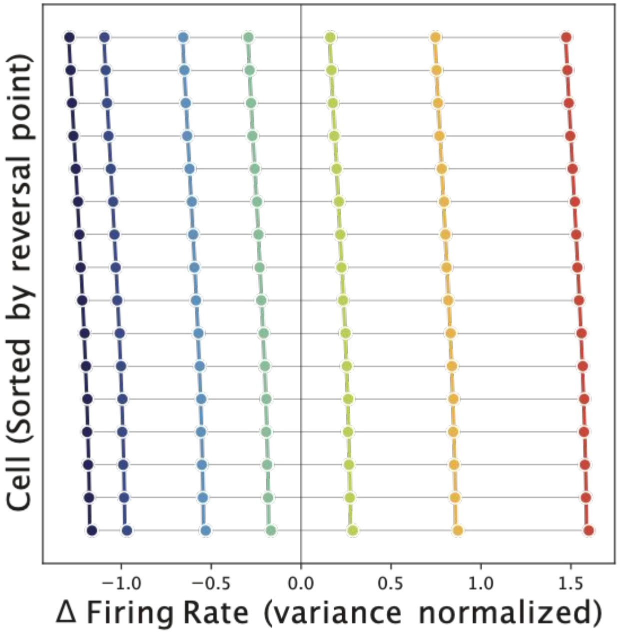


Schultz et al. (1993)

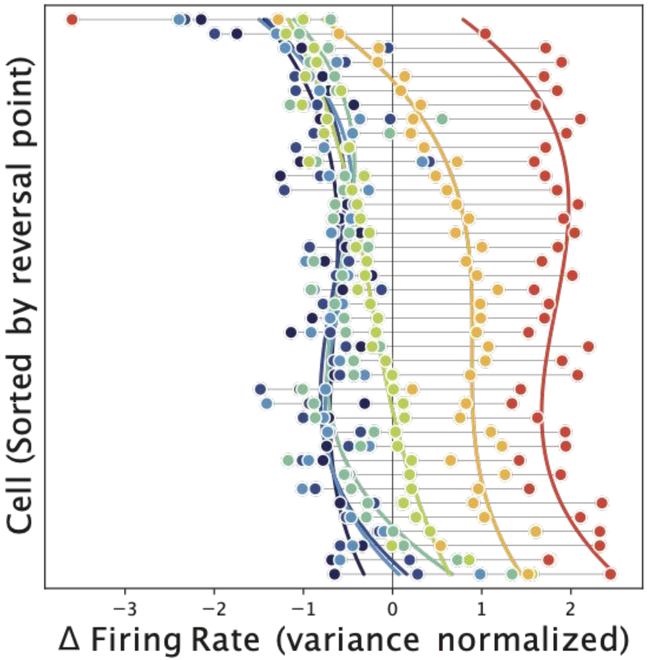


Schultz et al. (1997)

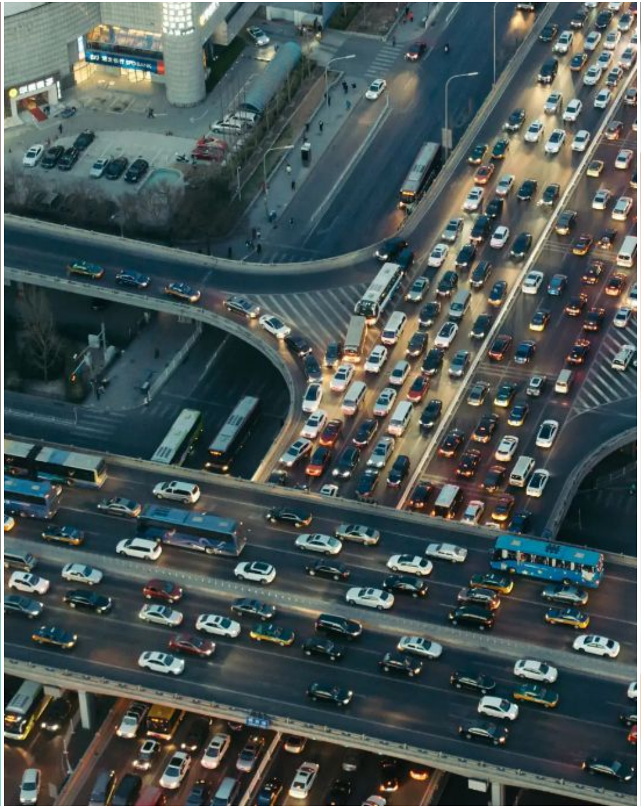
TD model

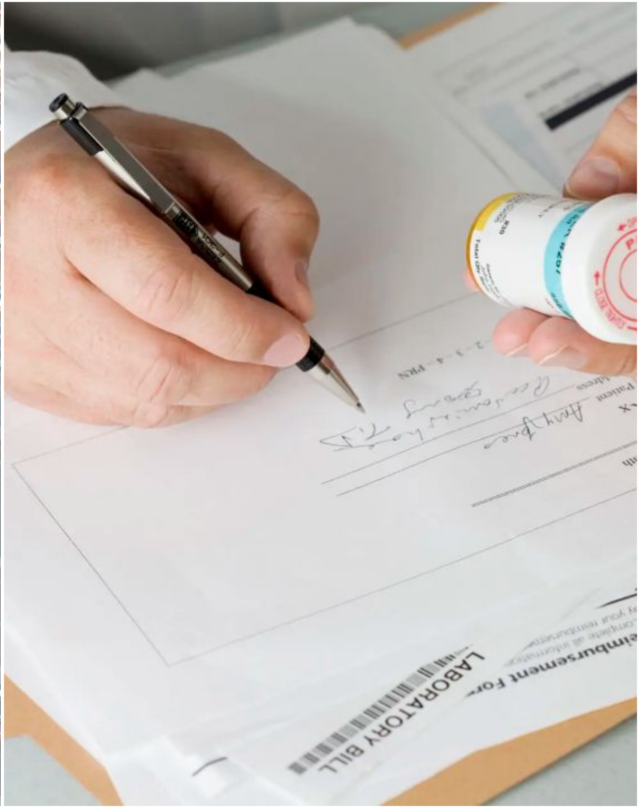


Actual data



Dabney et al. (2020)
Data from Eshel et al. (2015)





Randomness is an integral part of complex systems,
and is captured by distributional RL

www.distributional-rl.org

6.7920 Reinforcement Learning Foundations and Methods

Distributional Reinforcement Learning

Marc G. Bellemare