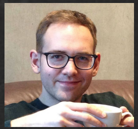
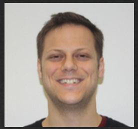
Distributional Reinforcement Learning

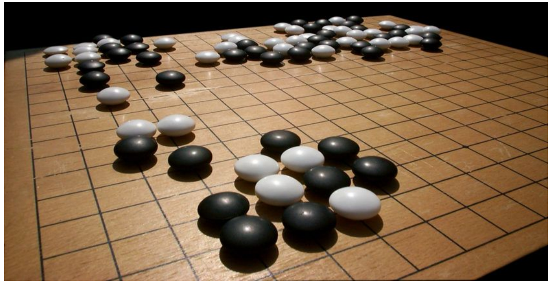
Rémi Munos

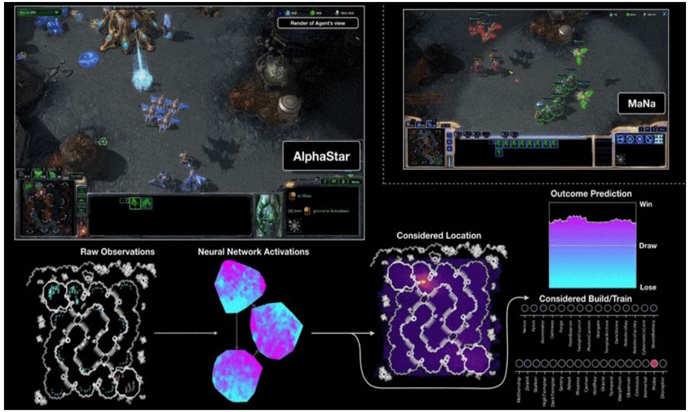
Marc Bellemare, Will Dabney, Georg Ostrovski, Mark Rowland

DeepMindParis

Deep RL is already a successful *empirical* research domain

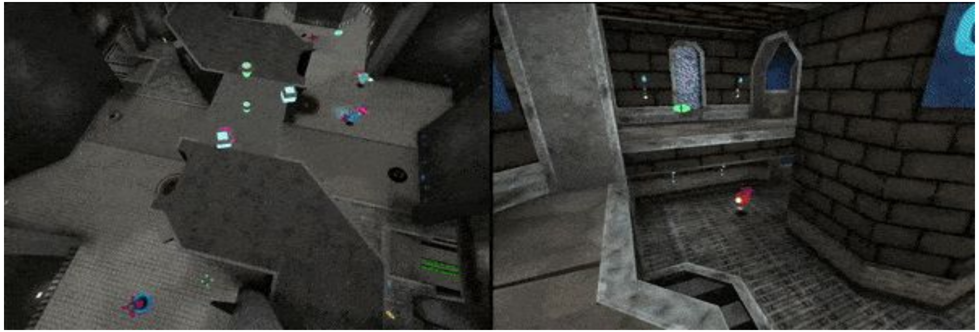


Computer go



StarCraft

DMLab30



Capture the Flag

Atari 57 games

Can we make it a *fundamental* research domain?

Related theoretical works:

- **RL side**: bandits, convergence of Q-learning, sample complexity, linear TD, Approximate DP, ...
- **Deep learning side**: VC-dim, convergence, stability, robustness against adversarial attacks, ...

Nice theoretical results, but how much do they tell us about deepRL?

Can we make it a *fundamental* research domain?

Related fundamental works:

- **RL side**: bandits, convergence of Q-learning, sample complexity, linear TD, Approximate DP, ...
- **Deep learning side**: VC-dim, convergence, stability, robustness against adversarial attacks, ...

Nice theoretical results, but how much do they tell us about deepRL?

What is specific about RL when combined with deep learning?

Distributional-RL

Shows interesting interactions between RL and deep-learning

Outline:

- Introduction to deep reinforcement learning
- The idea of distributional-RL
- Elements of theory
- Represents distributions in a neural net
- Numerical results on Atari
- Discussion about how/why this ‘works’

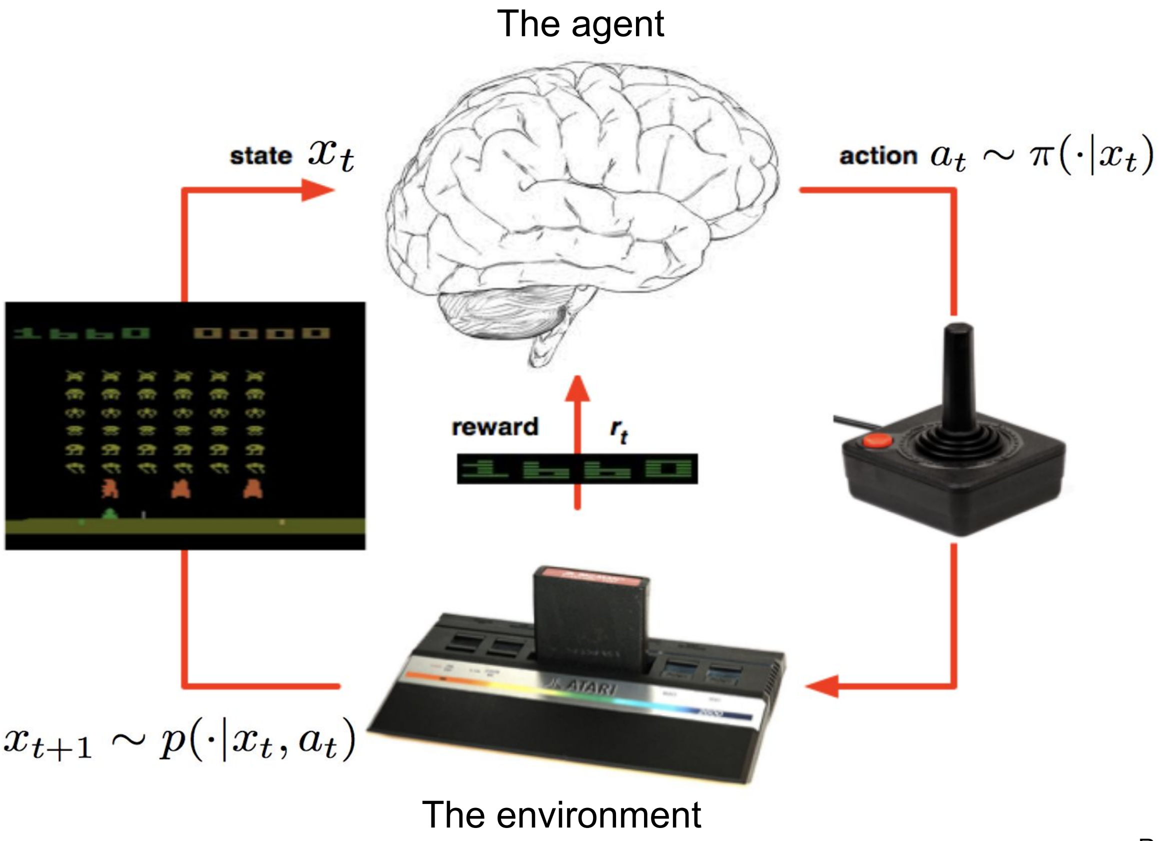
Reinforcement Learning (RL)

Learn to make good decisions

Learn from one's own experience
(by trial and error)

No supervision. Learn from rewards

The RL agent in its environment



2 core ingredients of RL

Credit assignment problem:

which actions are responsible for a reward?
$\longrightarrow$ Value-based methods ([Bellman 1957]'s Dynamic programming)
$\longrightarrow$ Policy-based methods ([Pontryagin 1957]'s Maximum principle)

Representation problem:

how to represent functions, models and policies?
$\longrightarrow$ use deep learning!
$\longrightarrow$ ***DeepRL***

Bellman's dynamic programming

- Define the value function Q^π of a policy $\pi(a|x)$:

$$Q^\pi(x,a) = \mathbb{E}\Big[\sum_{t\geq 0} \gamma^t r_t \Big| x, a, \pi\Big],$$

and the optimal value function:

$$Q^*(x,a) = \max_\pi Q^\pi(x,a).$$

(expected sum of future rewards if the agent plays optimally).

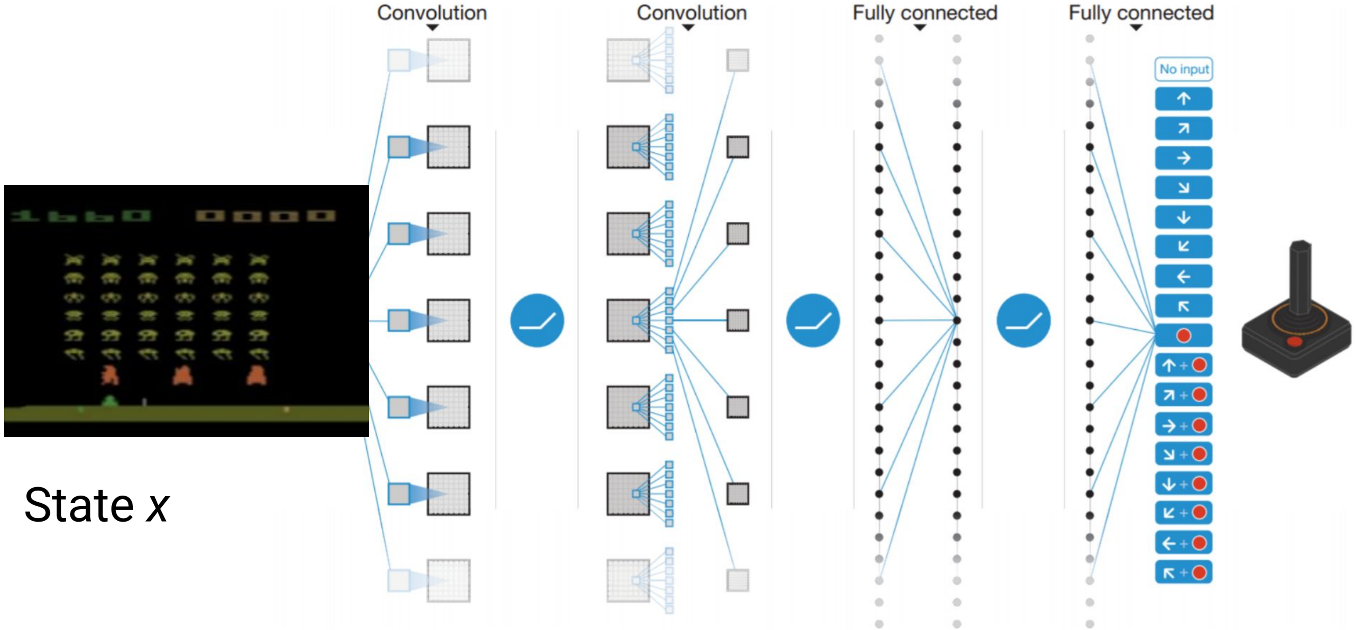
- Bellman equations:

$$Q^\pi(x,a) = r(x,a) + \gamma \mathbb{E}_{\substack{x' \sim p(\cdot|x,a) \\ a' \sim \pi(\cdot|x')}} \big[Q^\pi(x',a')\big]$$

$$Q^*(x,a) = r(x,a) + \gamma \mathbb{E}_{x' \sim p(\cdot|x,a)} \big[\max_{a'} Q^*(x',a')\big]$$

- Optimal policy $\pi^*(x) = \arg\max_a Q^*(x,a)$

Use a neural net for approximating the value function



State x

Q-value $Q_w(x,a)$

Weights w

Represent Q using a neural network

- How to train $Q_w(x,a)$? We don't have supervised values.

$$Q_w(x,a) \approx r(x,a) + \gamma \mathbb{E}_{x'}\left[\max_{a'} Q_w(x',a') \middle| x, a \right]$$

- After a transition $x_t, a_t \rightarrow x_{t+1}$,

$$\text{train } Q_w(x_t, a_t) \text{ to predict } \underbrace{r_t + \gamma \max_a Q_w(x_{t+1}, a)}_{\text{target values}}$$

- Minimize loss $\Big(\underbrace{r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)}_{\text{temporal difference } \delta_t} \Big)^2$.

- At the end of learning, $\mathbb{E}[\delta_t] = 0$.

Deep Q-Networks (DQN) [Mnih et al. 2013, 2015]

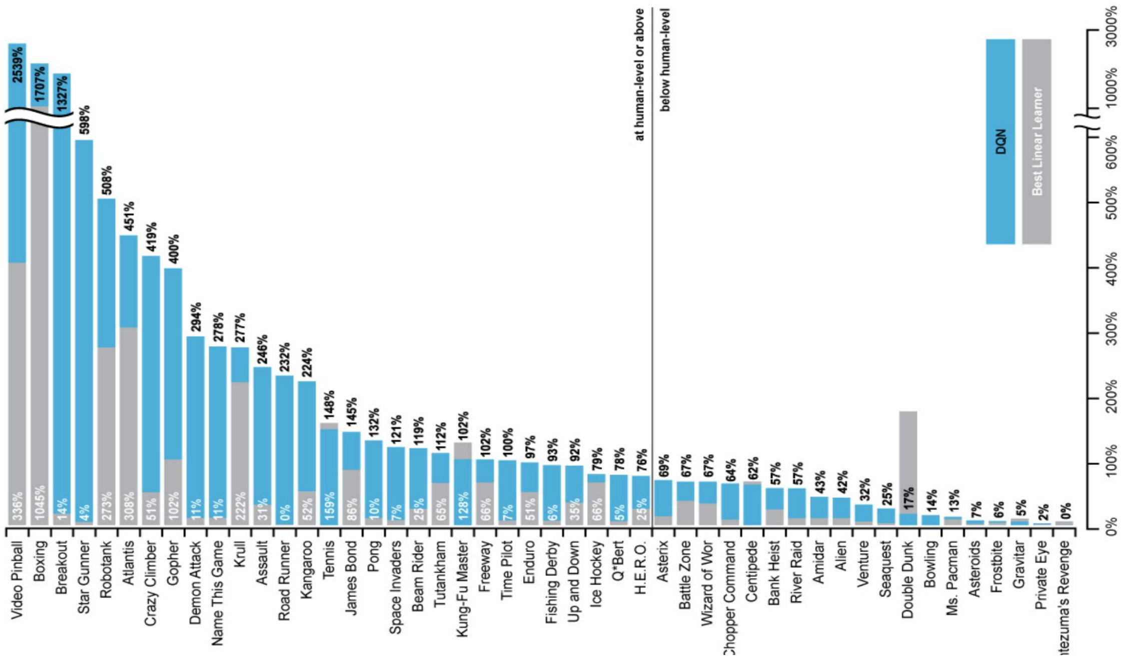
Problems: (1) data is not iid, (2) target values change
Idea: be as close as possible to supervised learning

1. Dissociate acting from learning:
   - Interact with the environments by following behavior policy
   - Store transition samples x_t, a_t, x_{t+1}, r_t into a memory replay
   - Train by replaying iid from memory
2. Use target network fixed for a while

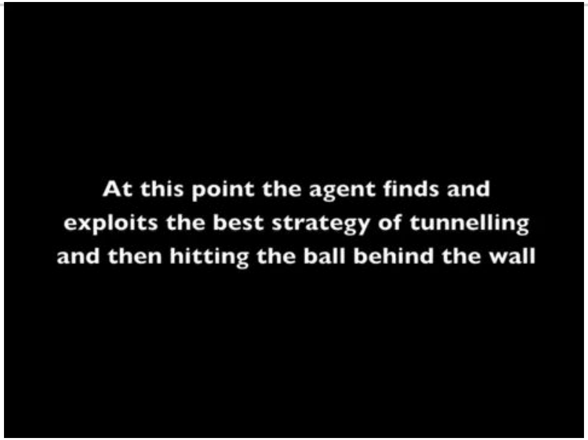
$$\text{loss} \; = \Big(r_t + \gamma \max_a Q_{w_{target}}(x_{t+1}, a) - Q_w(x_t, a_t) \Big)^2$$

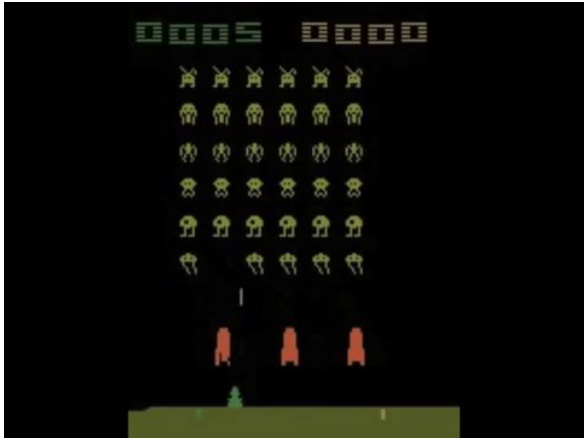
Properties: DQN is off-policy, and uses 1-step bootstrapping.

DQN Results in Atari



The same algorithm learns to play 57 games





Improvements since Nature DQN

- Double DQN: Remove upward bias caused by $\max_a Q(s, a, \mathbf{w})$
  - Current Q-network $\mathbf{w}$ is used to select actions
  - Older Q-network $\mathbf{w}_{\text{target}}$ is used to evaluate actions

$$Q(s, a) \leftarrow r(s, a) + \gamma Q(s', \underset{a'}{\operatorname{argmax}}\ Q(s', a', \mathbf{w}), \mathbf{w}^-)$$

[van Hasselt et al., 2015]

- Prioritised replay: Weight experience according to surprise
  - Store experience in priority queue according to DQN error

$$\left| r + \gamma \ \max a' Q(s', a', \mathbf{w}^-) - Q(s, a, w) \right|$$

[Schaul et al., 2015]

- Duelling network: Split Q-network into two channels
  - Action-independent value function $V(s, v)$
  - Action-dependent advantage function $A(s, a, \mathbf{w})$

[Wang et al., 2015]

$$Q(s, a) = V(s, v) + A(s, a, \mathbf{w})$$

Other improvements

- Persistent DQN: Repeat same action at next state if next state is very similar to previous state. Update $Q(s,a)$

[Bellemare et al., 2015]

$$Q(s,a) \leftarrow r(s,a) + \gamma\Big[\beta \max_{a'} Q(s',a') + (1-\beta)Q(s',a)\Big].$$

- Multi-steps updates: Propagate information over several steps:

$$Q(s,a) \leftarrow \sum_{t=0}^{n-1} \gamma^t r_t + \gamma^n \max_{a'} Q(s_n,a').$$

[Hessel et al., 2017]

Faster propagation of information but this is an on-policy algorithm (i.e. actions are greedy w.r.t. current Q).

- Retrace & vtrace algorithms: multi-steps off-policy learning:

$$Q(s,a) \leftarrow \sum_{t\geq 0} \gamma^t (c_1 \dots c_t)\Big(r_t + \gamma \max_{a'} Q(s_t,a') - Q(s_t,a_t)\Big),$$

[Munos et al., 2016]

where $c_t = \min\left(1, \frac{\pi(a_t|x_t)}{\mu(a_t|s_t)}\right)$.

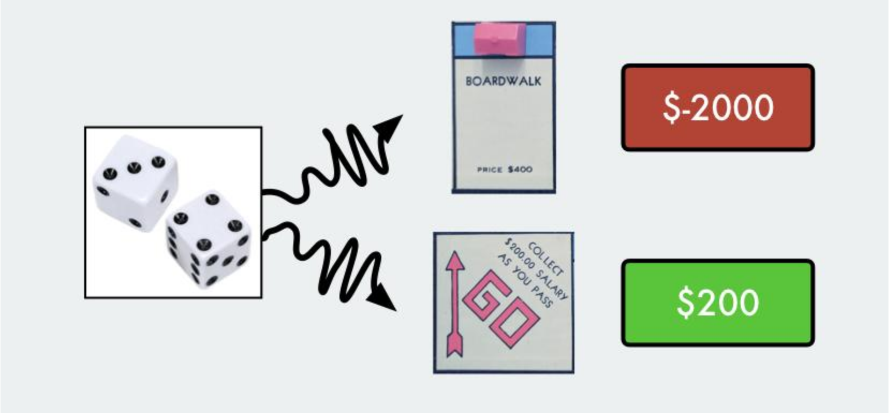
Distributional-RL

- Introduction
- Elements of theory
- Neural net representations
- Experiments on Atari
- Conclusion

Intro to distributional RL

Expected immediate reward

$$\mathbb{E}[R(x)] = \frac{1}{36} \times (-2000) + \frac{35}{36} \times (200) = 138.88$$



Random variable reward:

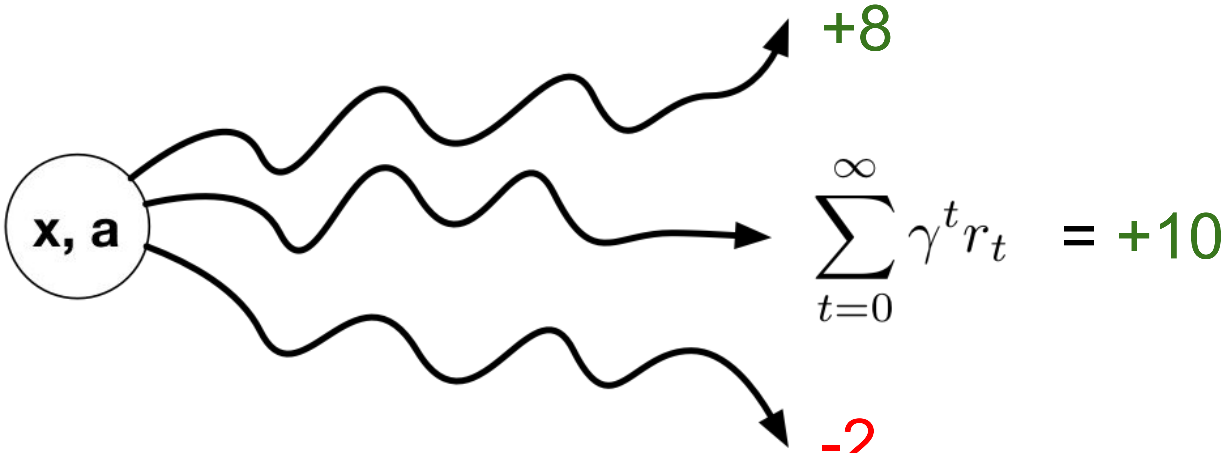
$$R(x) = \begin{cases} -2000 \text{ w.p. } 1/36 \\ 200 \text{ w.p. } 35/36 \end{cases}$$

The return = sum of future discounted rewards

$$R(x_0) + \gamma R(x_1) + \gamma^2 R(x_2) \dots$$

- Returns are often complex, multimodal
- Modelling the expected return hides this intrinsic randomness
- Model all possible returns!

The r.v. Return $Z^\pi(x,a) = \sum_{t\geq 0} \gamma^t r(x_t, a_t)\Big|_{x_0=x, a_0=a, \pi}$



Captures intrinsic randomness from:

- Immediate rewards
- Stochastic dynamics
- Possibly stochastic policy

The expected Return

The value function $Q^\pi(x,a) = \mathbb{E}[Z^\pi(x,a)]$

Satisfies the Bellman equation

$$Q^\pi(x,a) = \mathbb{E}[r(x,a) + \gamma Q^\pi(x',a')]$$

$$\text{where } x' \sim p(\cdot|x,a) \text{ and } a' \sim \pi(\cdot|x')$$

Distributional Bellman equation?

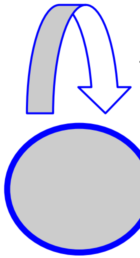
We would like to write a Bellman equation for the distributions:

$$Z^\pi(x,a) \stackrel{D}{=} R(x,a) + \gamma Z^\pi(x',a')$$

$$\text{where } x' \sim p(\cdot|x,a) \text{ and } a' \sim \pi(\cdot|x')$$

Does this equation make sense?

Example



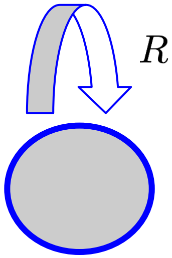
$$R = \begin{cases} 1 & \text{w.p. } 1/2 \\ 0 & \text{w.p. } 1/2 \end{cases}$$

Reward = Bernoulli (½), discount factor γ = ½

Bellman equation: $V = \frac{1}{2} + \frac{1}{2} V$, thus V = 1

Return $Z = \sum_{t \geq 0} 2^{-t} R_t$ Distribution?

Example



$$R = \begin{cases} 1 & \text{w.p. } 1/2 \\ 0 & \text{w.p. } 1/2 \end{cases}$$

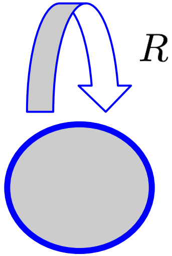
Reward = Bernoulli (½), discount factor γ = ½

Bellman equation: $V = \frac{1}{2} + \frac{1}{2}V$, thus V = 1

Return $Z = \sum_{t \geq 0} 2^{-t} R_t$ Distribution? $\mathcal{U}([0, 2])$

(rewards = binary expansion of a real number)

Example



$$R = \begin{cases} 1 & \text{w.p. } 1/2 \\ 0 & \text{w.p. } 1/2 \end{cases}$$

Reward = Bernoulli (½), discount factor γ = ½

Bellman equation: $V = \frac{1}{2} + \frac{1}{2}V$, thus V = 1

Return $Z = \sum_{t \geq 0} 2^{-t} R_t$ Distribution? $\mathcal{U}([0, 2])$

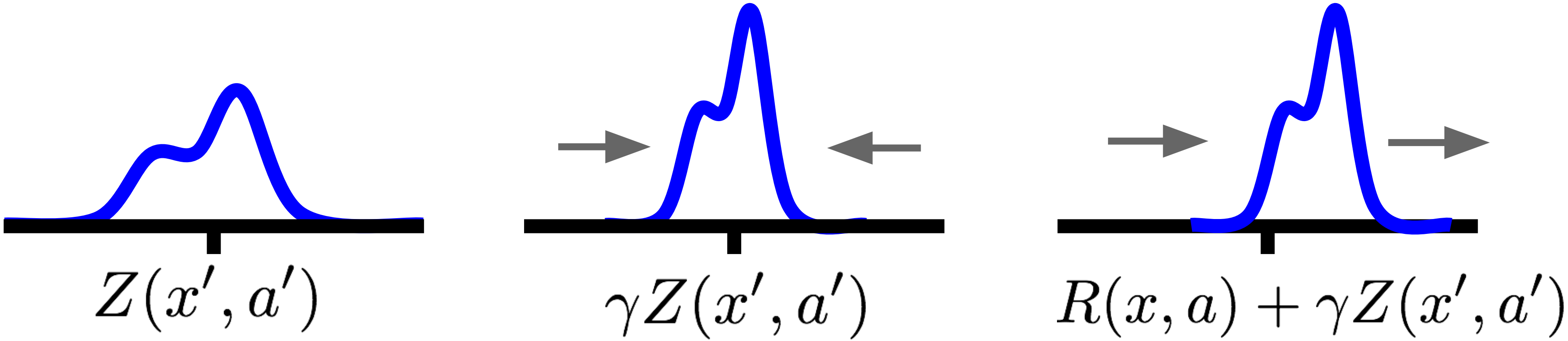
Distributional Bellman equation: $Z = \mathcal{B}(\frac{1}{2}) + \frac{1}{2}Z$

In terms of distribution:

$$\eta(z) = \frac{1}{2}\big(\delta(0) + \delta(1)\big) * 2\eta(2z)$$
$$= \eta(2z) + \eta(2(z-1))$$

Distributional Bellman operator

$$T^\pi Z(x,a) = R(x,a) + \gamma Z(x',a')$$



Does there exists a fixed point?

Properties

Theorem [Rowland et al., 2018]

$\boxed{T^\pi \text{ is a contraction in Cramer metric}}$

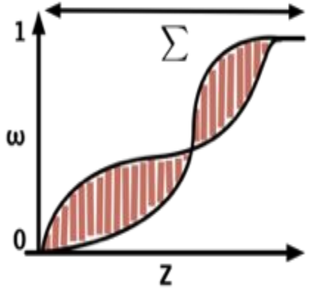
$$\ell_2(X,Y) = \left(\int_{\mathbb{R}} (F_X(t) - F_Y(t))^2 dt\right)^{1/2}$$

Theorem [Bellemare et al., 2017]

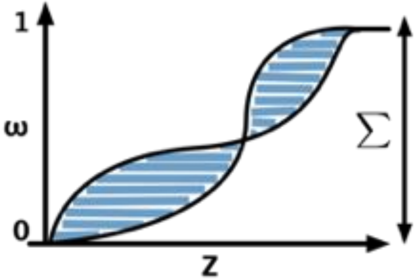
$\boxed{T^\pi \text{ is a contraction in Wasserstein metric,}}$

$$w_p(X,Y) = \left(\int_{\mathbb{R}} (F_X^{-1}(t) - F_Y^{-1}(t))^p dt\right)^{1/p}$$

(but not in KL neither in total variation)
Intuition: the size of the support shrinks.



Cramer



Wasserstein

Distributional dynamic programming

For a given policy π, the distributional Bellman operator

$$T^{\pi} Z(x, a) = R(x, a) + \gamma Z(x', a')$$

Is a contraction mapping, thus has a unique fixed point, which is Z^{π}

And the iterate $Z \leftarrow T^{\pi} Z$ converges to Z^{π}

The control case

Define the distributional Bellman optimality operator

$$TZ(x,a) \stackrel{D}{=} r(x,a) + \gamma Z(x', \pi_Z(x'))$$

$$\text{where } x' \sim p(\cdot|x,a) \text{ and } \pi_Z(x') = \arg\max_{a'} \mathbb{E}[Z(x',a')]$$

Is this operator a contraction mapping?

The control case

Define the distributional Bellman optimality operator

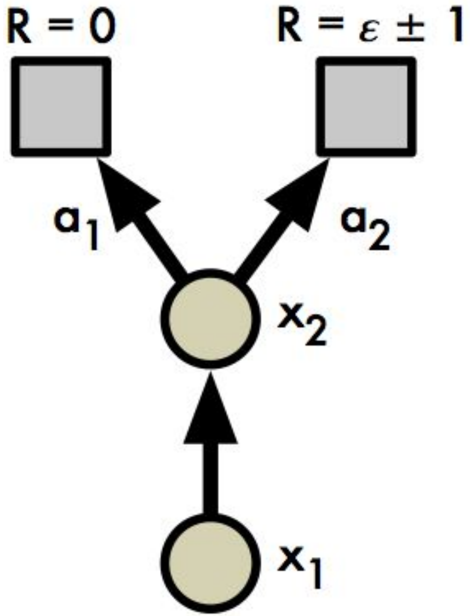
$$TZ(x,a) \stackrel{D}{=} r(x,a) + \gamma Z(x', \pi_Z(x'))$$

$$\text{where } x' \sim p(\cdot|x,a) \text{ and } \pi_Z(x') = \arg\max_{a'} \mathbb{E}[Z(x',a')]$$

Is this operator a contraction mapping?

No! (it's not even continuous)

The dist. opt. Bellman operator is not smooth



Consider distributions Z_ϵ

If $\varepsilon > 0$ we back up a bimodal distribution

If $\varepsilon < 0$ we back up a Dirac in 0

Thus the map $Z_\epsilon \mapsto TZ_\epsilon$ is not continuous

Distributional Bellman optimality operator

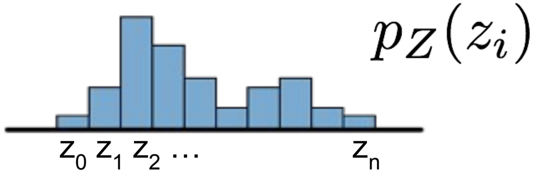
Theorem [Bellemare et al., 2017]

if the optimal policy is unique, then the iterates $Z_{k+1} \leftarrow TZ_k$ converge to Z^{π^*}

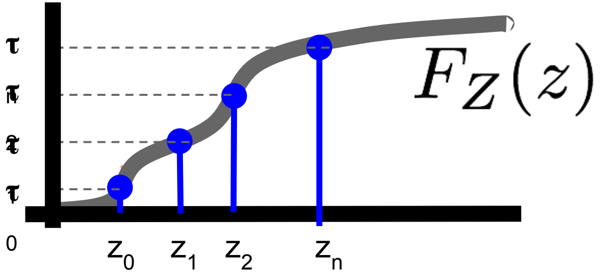
Intuition: The distributional Bellman operator preserves the mean, thus the mean will converge to the optimal policy π^* eventually. If the policy is unique, we revert to iterating T^{π^*}, which is a contraction.

How to represent distributions?

- Categorical



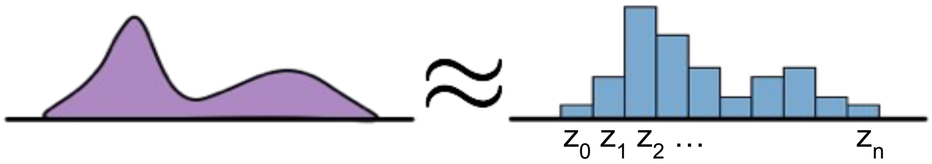
- Inverse CDF for specific quantile levels



- Parametric inverse CDF

$$\tau \mapsto F_Z^{-1}(\tau)$$

Categorical distributions

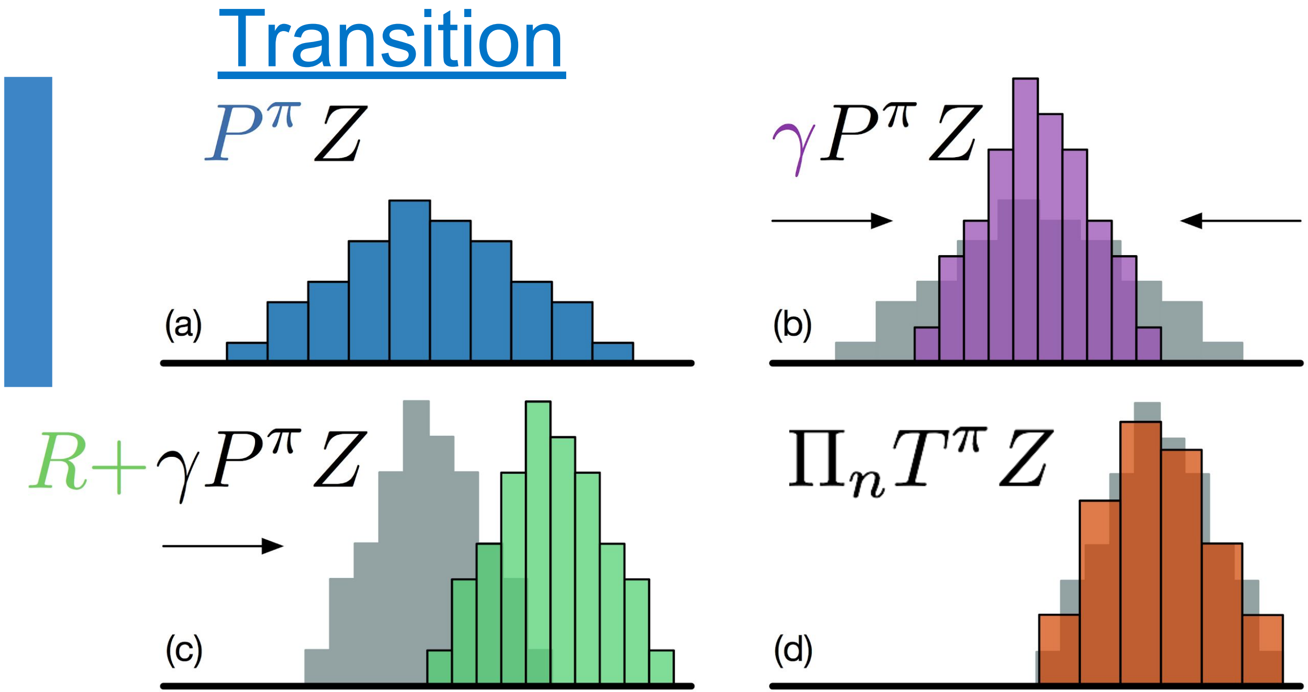


Distributions supported on a finite support $\{z_1, \dots, z_n\}$

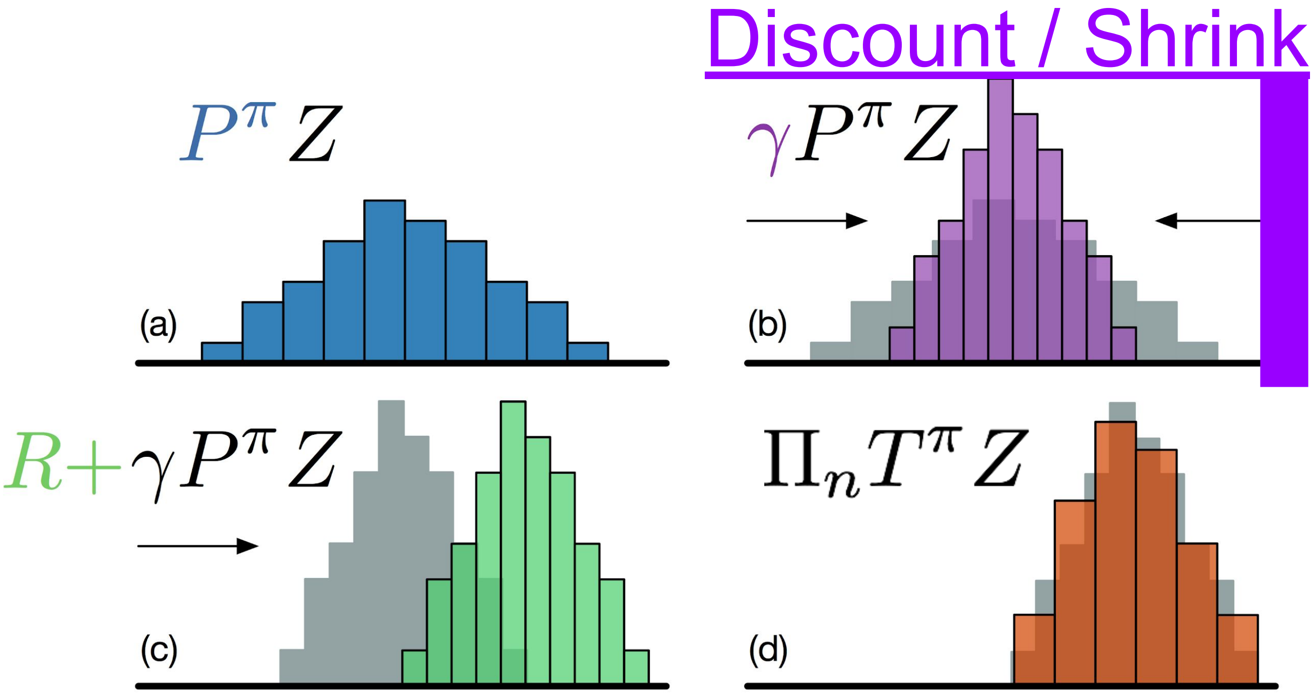
Discrete distribution $\{p_i(x,a)\}_{1 \leq i \leq n}$

$$Z(x,a) = \sum_i p_i(x,a)\delta_{z_i}$$

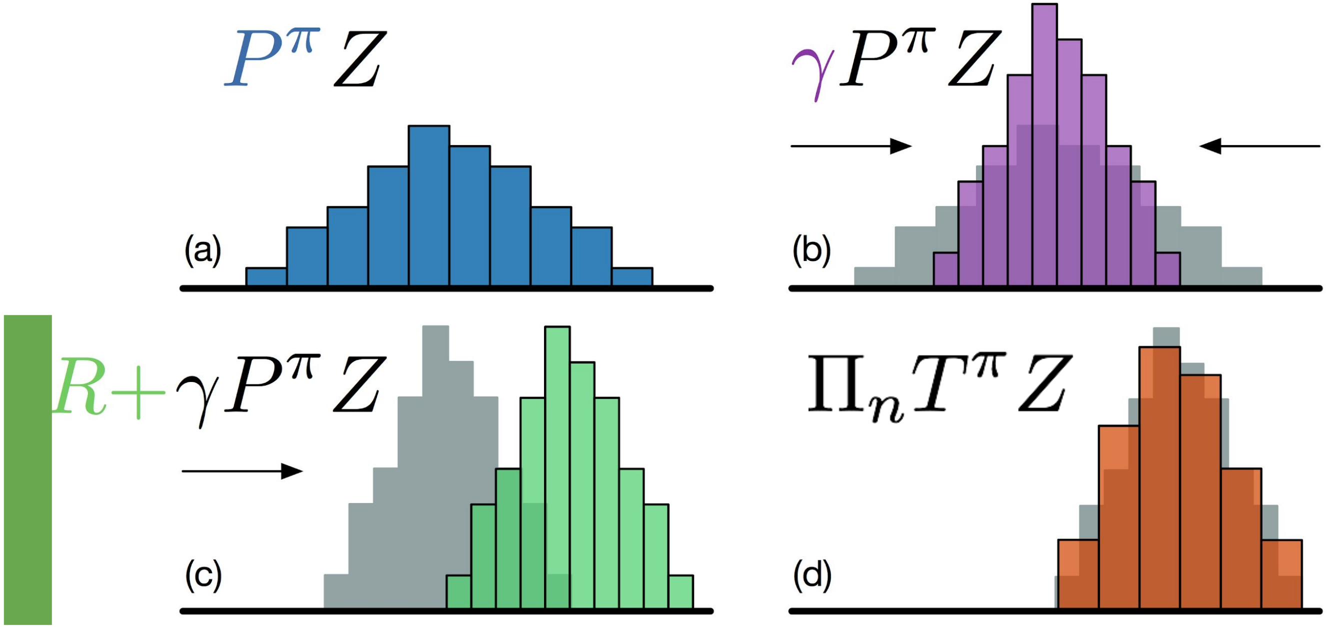
Projected Distributional Bellman Update



Projected Distributional Bellman Update

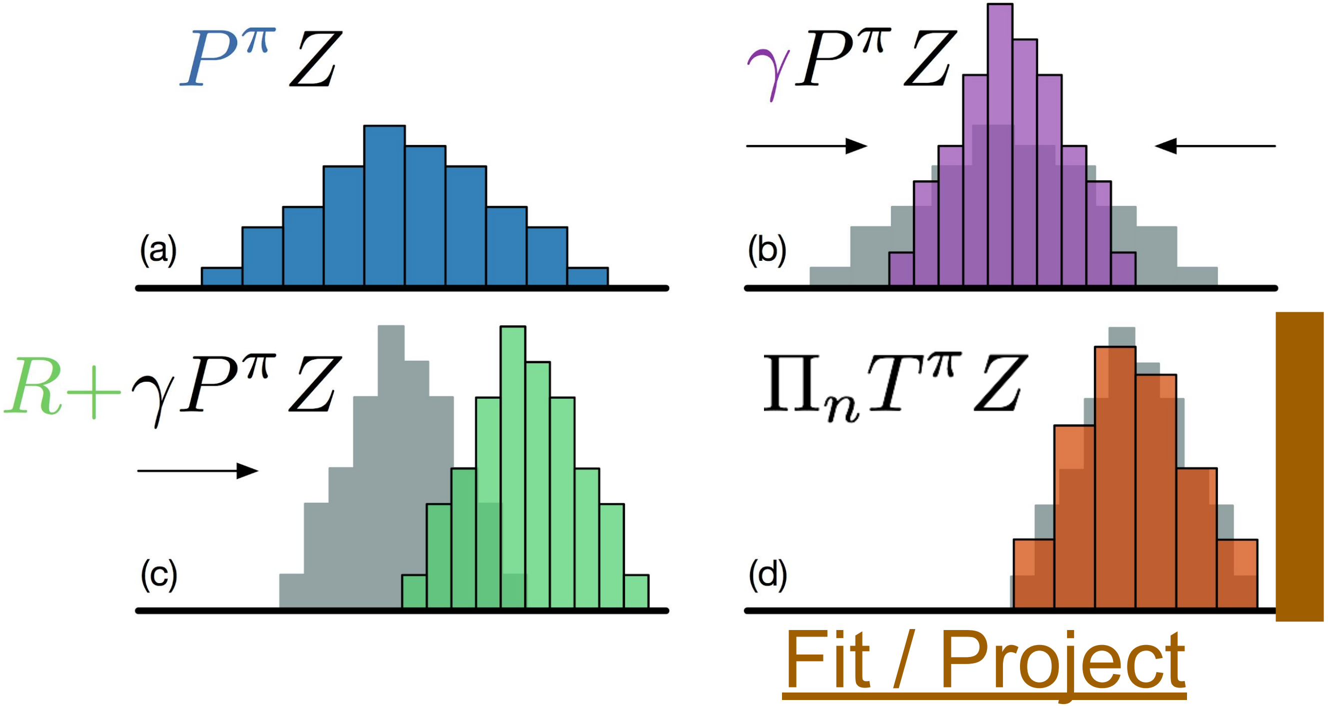


Projected Distributional Bellman Update



Reward / Shift

Projected Distributional Bellman Update



Projected distributional Bellman operator

Let Π_n be the projection onto the support (piecewise linear interpolation)

Theorem: $\boxed{\Pi_n T^\pi \text{ is a contraction (in Cramer distance)}}$

Intuition: Π_n is a non-expansion (in Cramer distance).

Its fixed point Z_n can be computed by value iteration $Z \leftarrow \Pi_n T^\pi Z$

Theorem: $$\boxed{\ell_2^2(Z_n, Z^\pi) \leq \frac{1}{(1-\gamma)} \max_{1 \leq i < n} |z_{i+1} - z_i|}$$ [Rowland et al., 2018]

Projected distributional Bellman operator

Policy iteration: iterate

- Policy evaluation: $Z_k = \Pi_n T^{\pi_k} Z_k$

- Policy improvement: $\pi_{k+1}(x) = \arg\max_a \mathbb{E}[Z^{\pi_k}(x, a)]$

Theorem: Assume there is a unique optimal policy.
Z_k converges to $Z_n^{\pi^*}$, whose greedy policy is optimal.

Distributional Q-learning

Observe transition samples $x_t, a_t \xrightarrow{r_t} x_{t+1}$

Update:

$$Z(x_t, a_t) = (1 - \alpha_t) Z(x_t, a_t) + \alpha_t \Pi_C (r_t + \gamma Z(x_{t+1}, \pi_Z(x_{t+1}))$$

Theorem

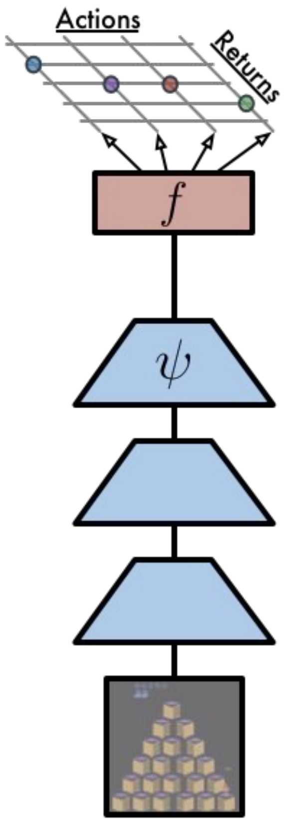
Under the same assumption as for Q-learning,
assume there is a unique optimal policy π^*,
then $Z \to Z_n^{\pi^*}$ and the resulting policy is optimal.

[Rowland et al., 2018]

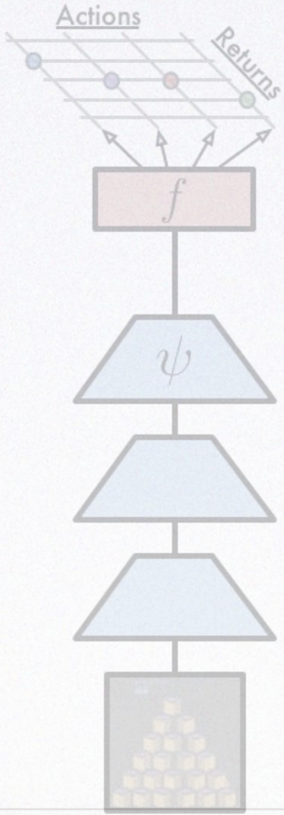
DeepRL implementation

DQN

[Mnih et al., 2013]

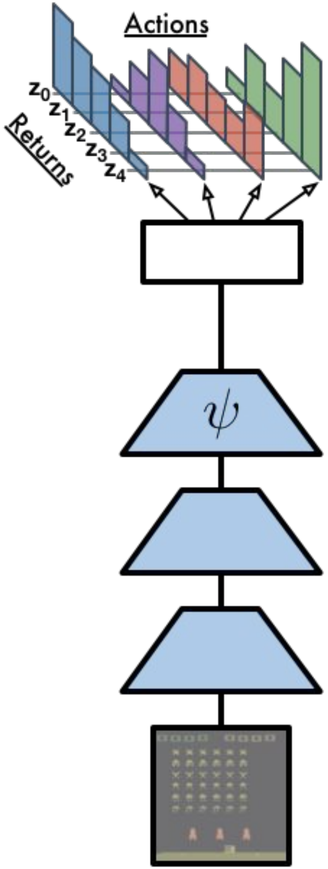


DQN

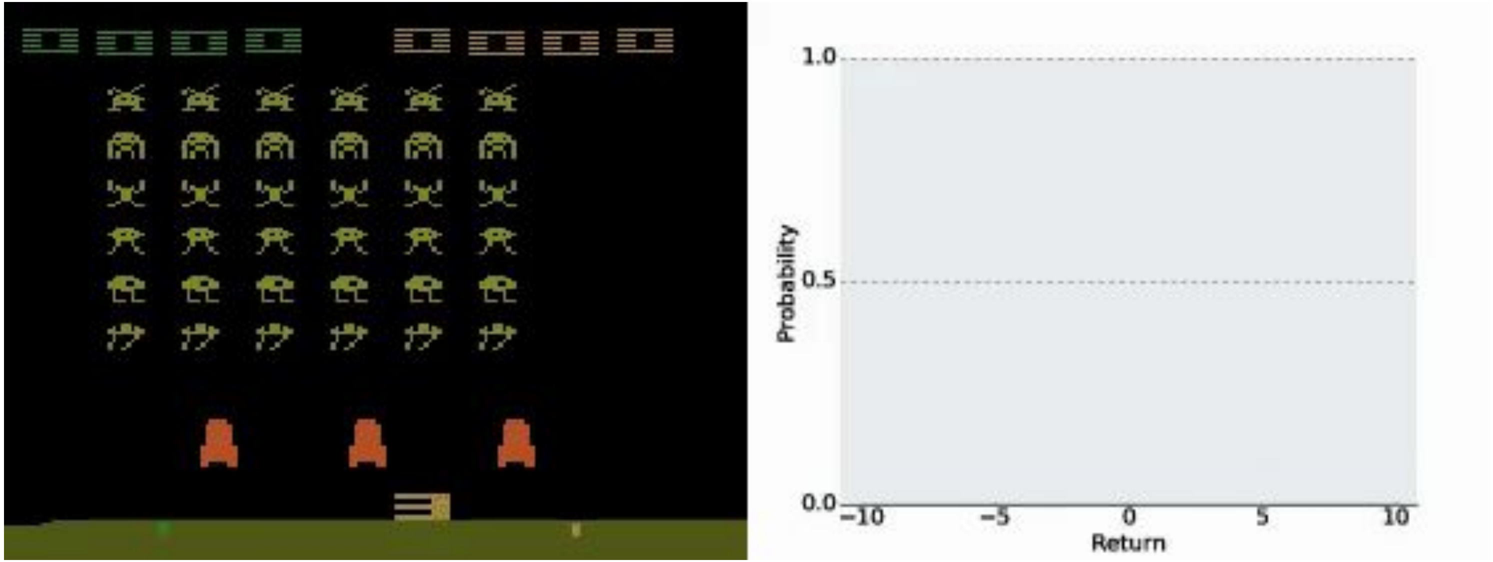


Categorical DQN

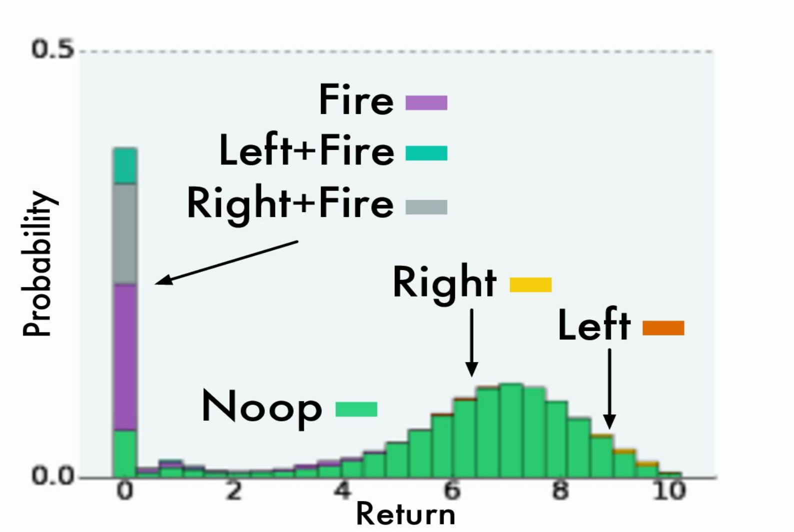
[Bellemare et al., 2017]

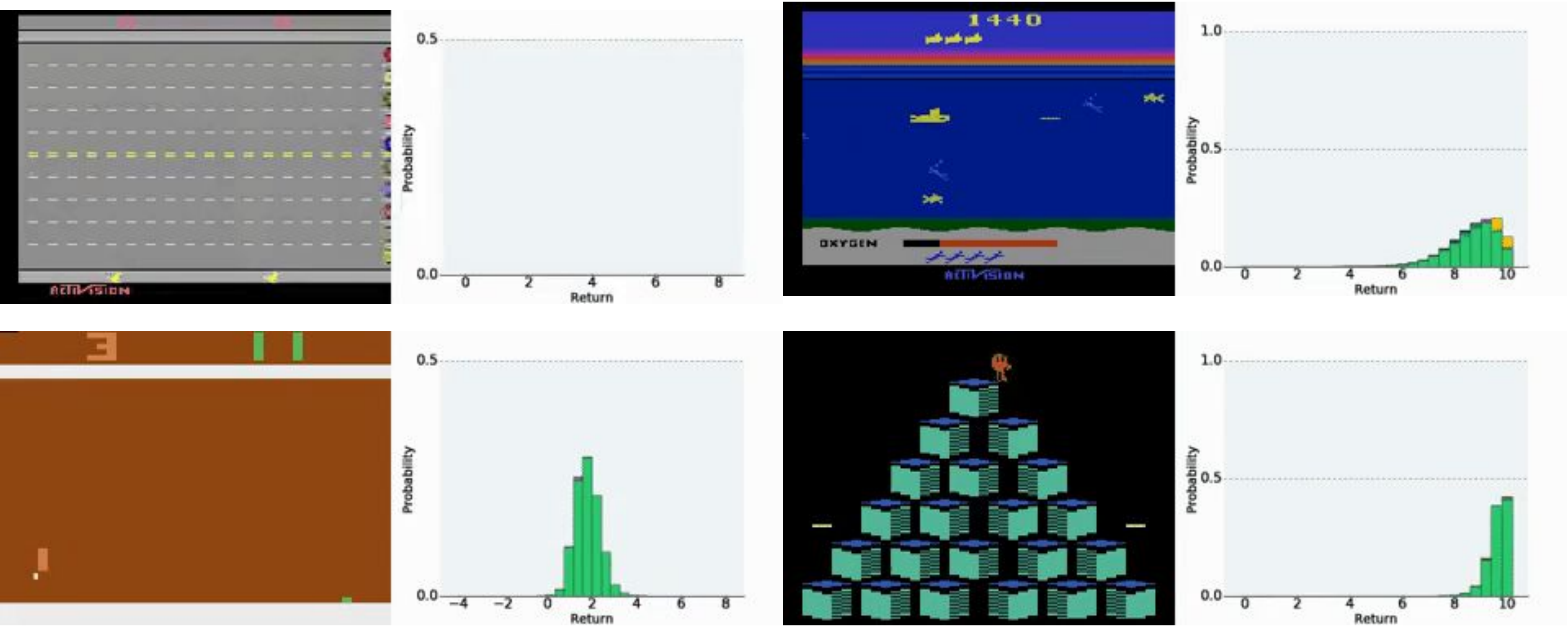


Categorical DQN

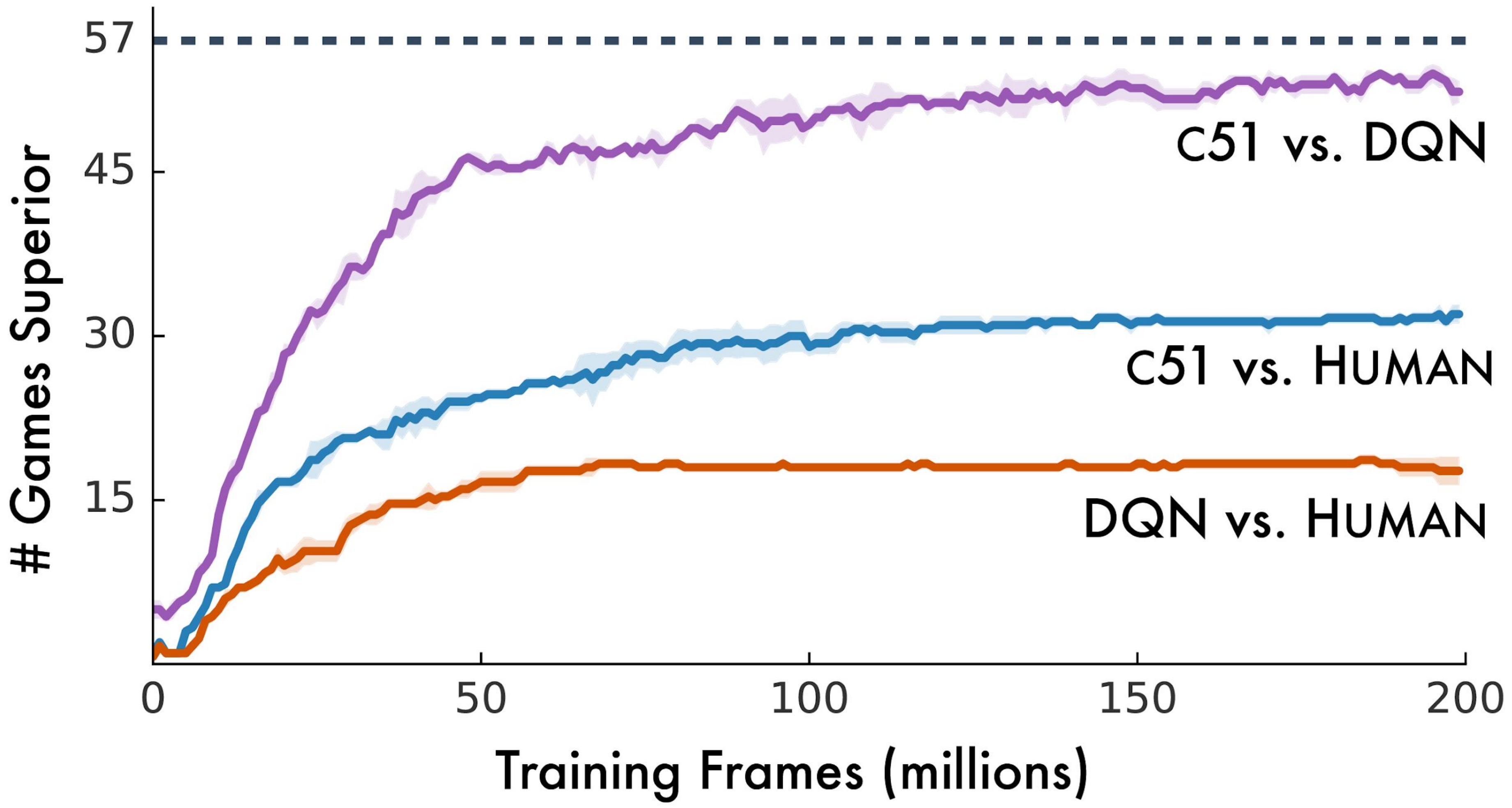


Randomness from future choices





ACTIVISION
0.5
0.0
Probability
0 2 4 6 8
Return
1440
OXYGEN
ACTIVISION
1.0
0.5
0.0
Probability
0 2 4 6 8 10
Return
3
0.5
0.0
Probability
−4 −2 0 2 4 6 8
Return
1.0
0.5
0.0
Probability
0 2 4 6 8 10
Return

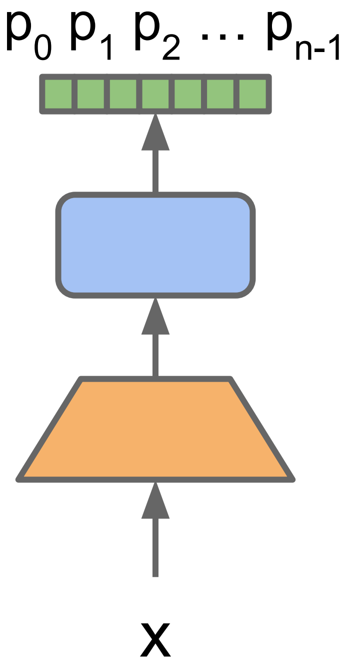


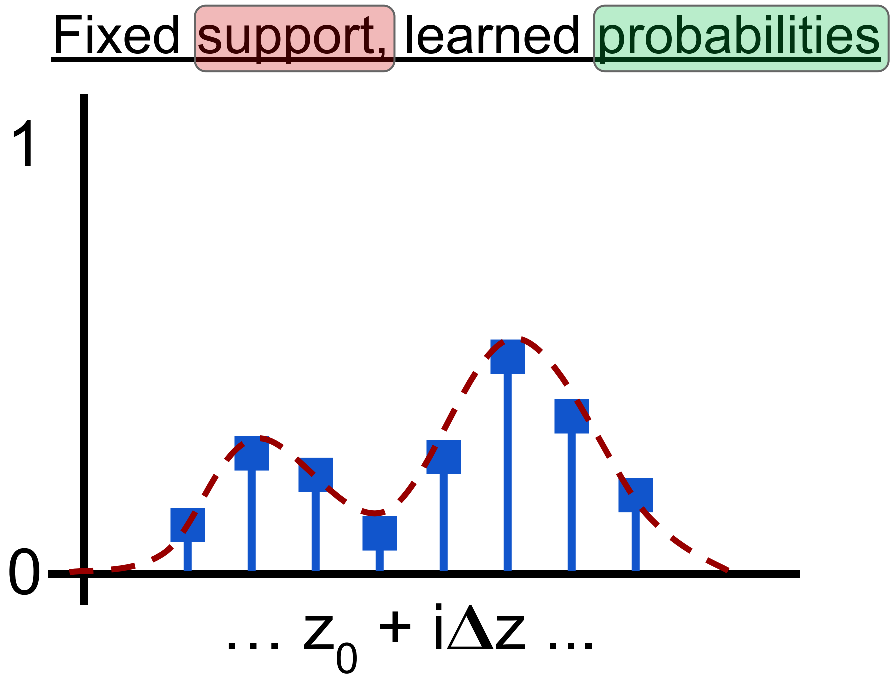
57
45
30
15
Games Superior
C51 vs. DQN
C51 vs. HUMAN
DQN vs. HUMAN
0
50
100
150
200
Training Frames (millions)

Results on 57 games Atari 2600

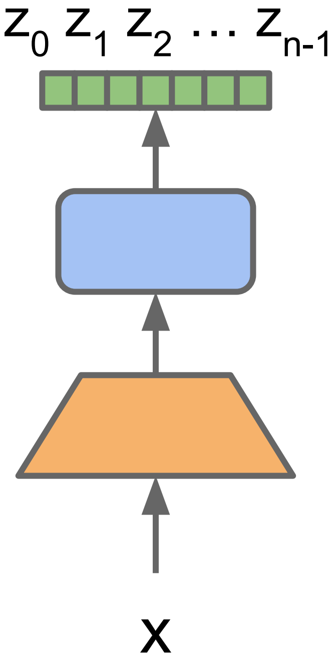
	Mean	Median	>human
DQN	228%	79%	24
Double DQN	307%	118%	33
Dueling	373%	151%	37
Prio. Duel.	592%	172%	39
C51	701%	178%	40

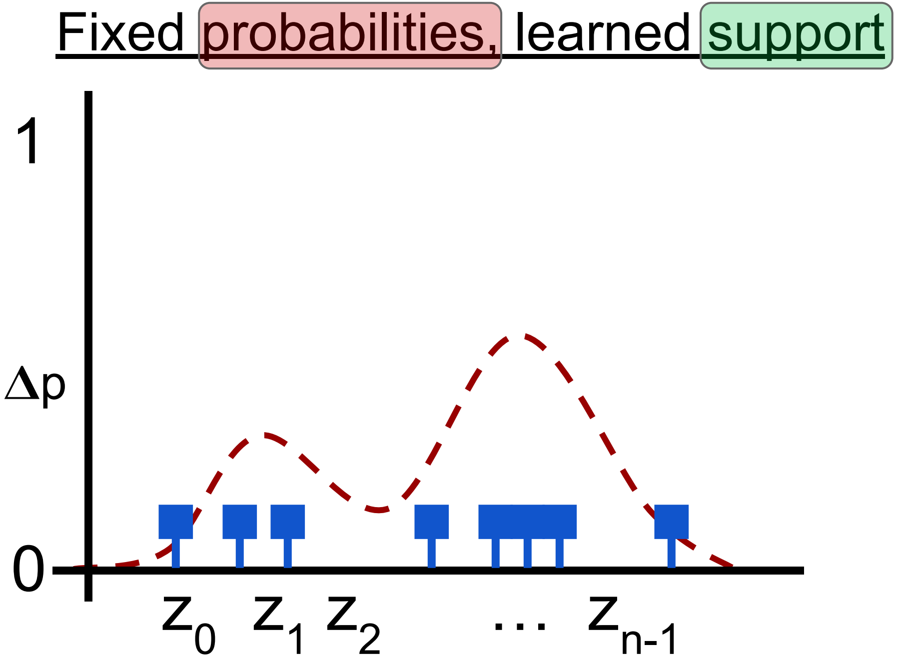
Categorical representation



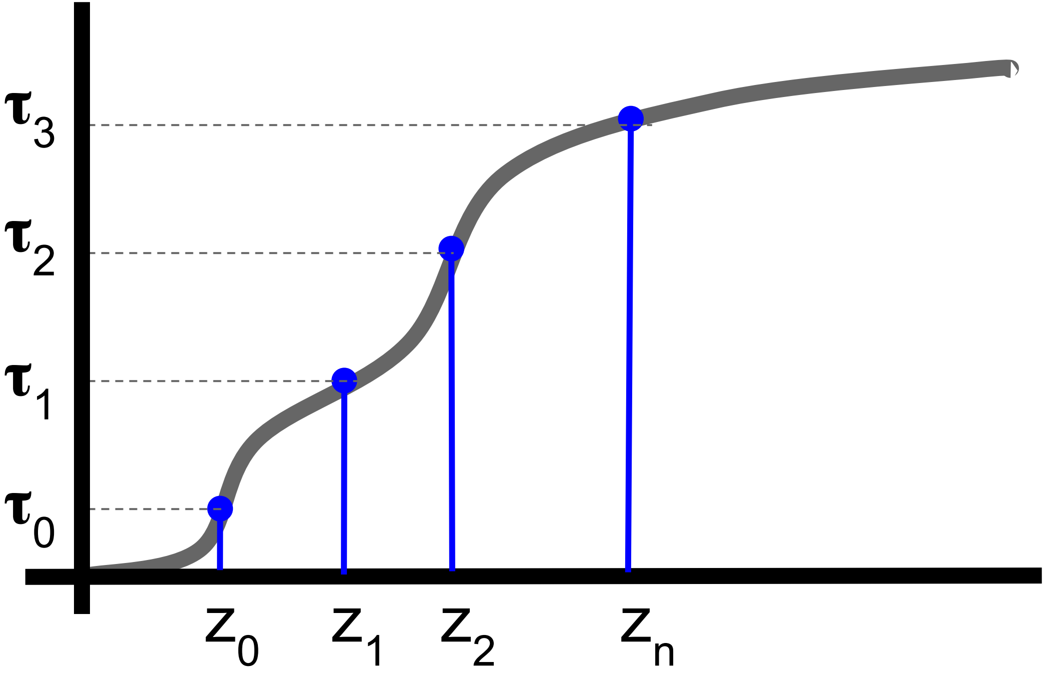


Quantile Regression Networks

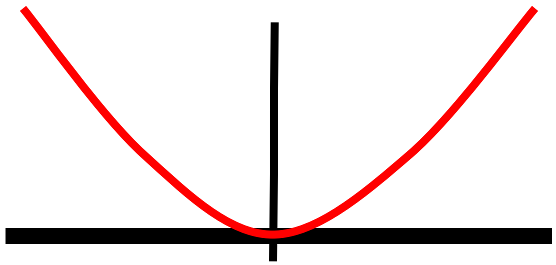




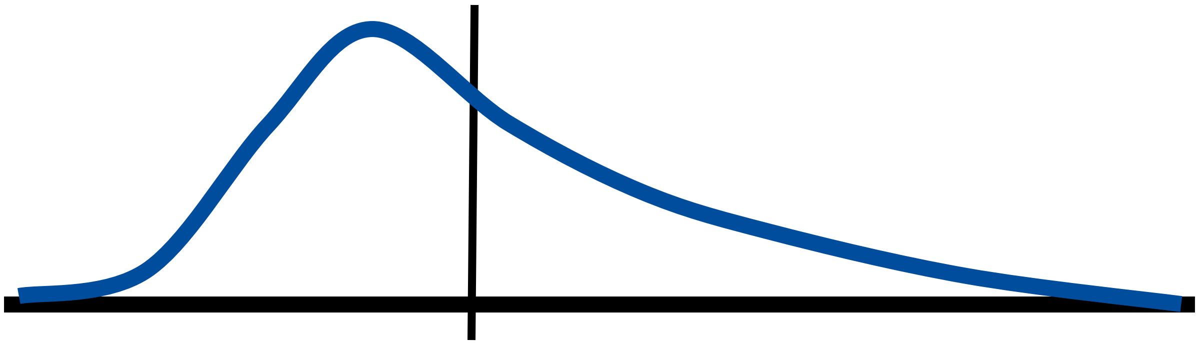
Inverse CDF learnt by Quantile Regression



l2-regression

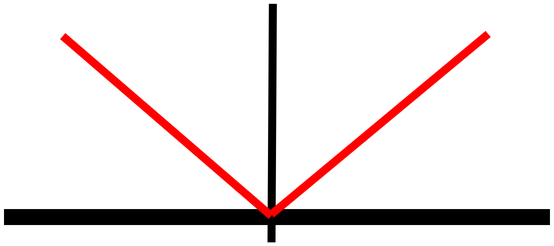


$$loss = x^2$$

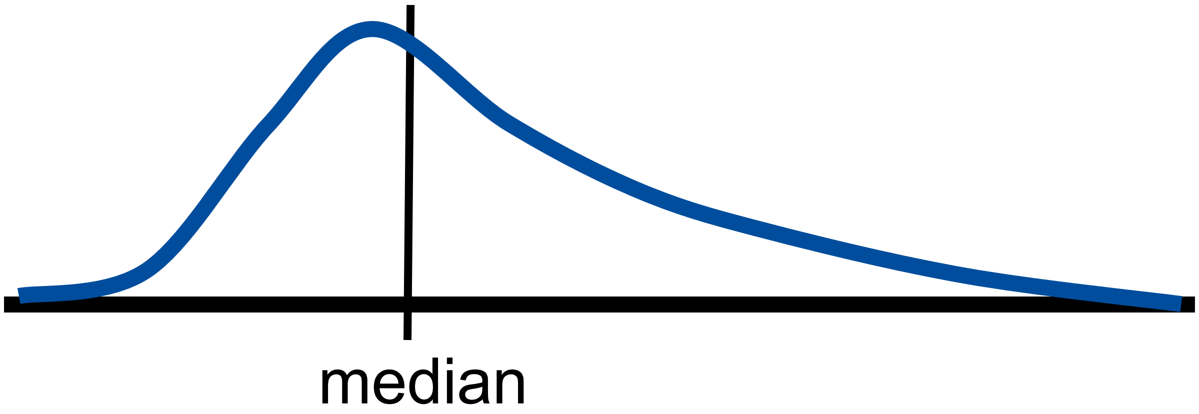


mean

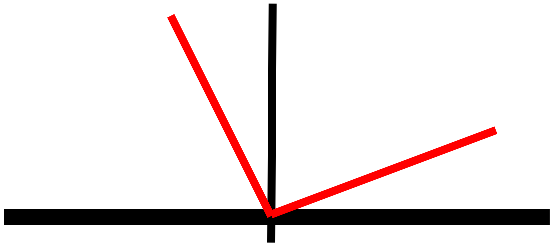
l1-regression



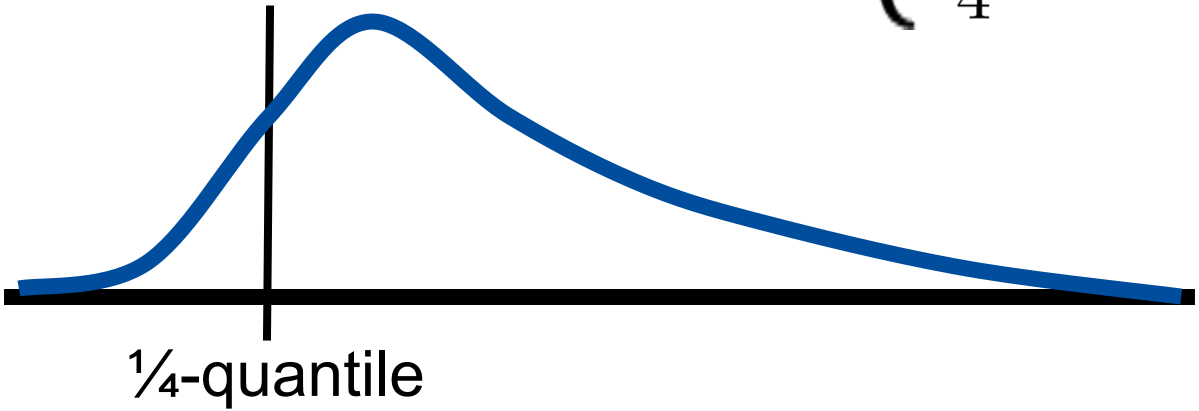
$$loss = |x|$$



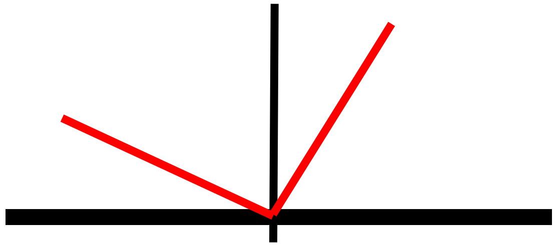
¼-quantile-regression



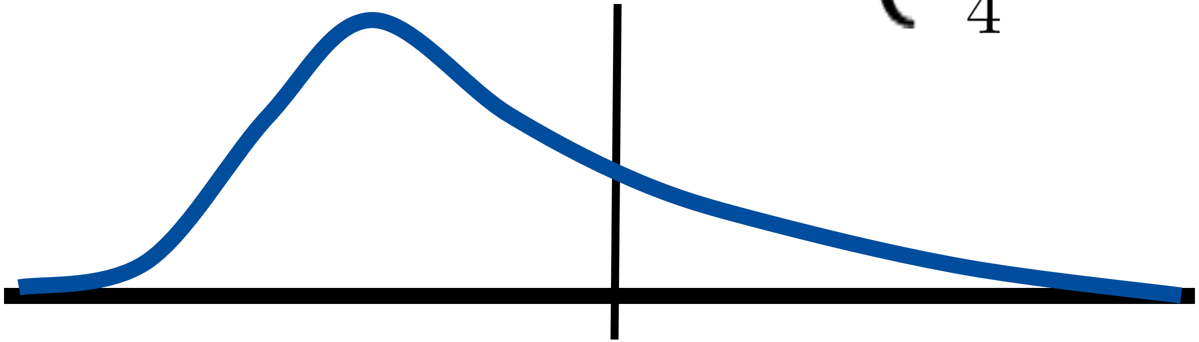
$$loss = \begin{cases} \frac{1}{4}x, & \text{for } x \geq 0 \\ -\frac{3}{4}x, & \text{for } x < 0 \end{cases}$$



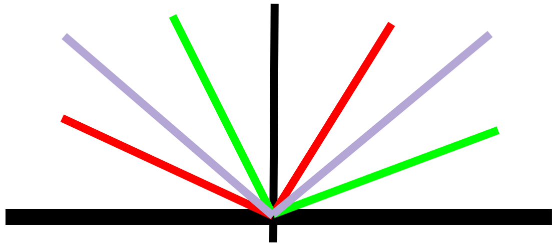
¾-quantile-regression



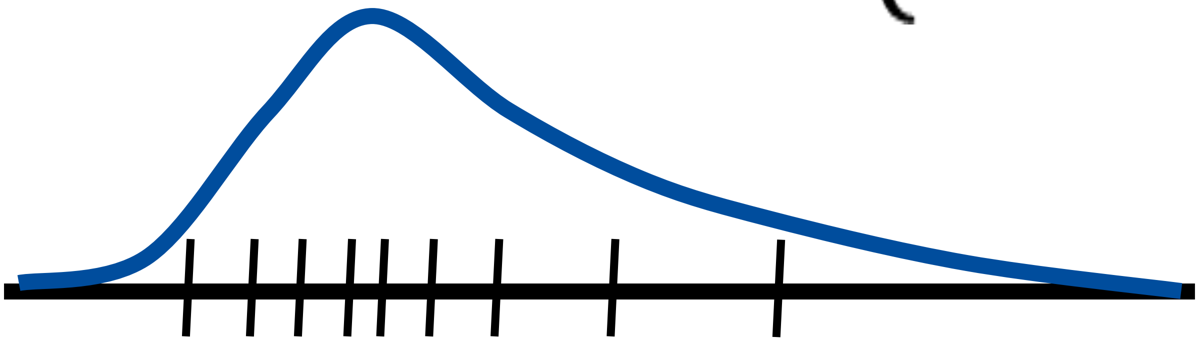
$$loss = \begin{cases} \frac{3}{4}x, & \text{for } x \geq 0 \\ -\frac{1}{4}x, & \text{for } x \geq 0 \end{cases}$$



many-quantiles-regression

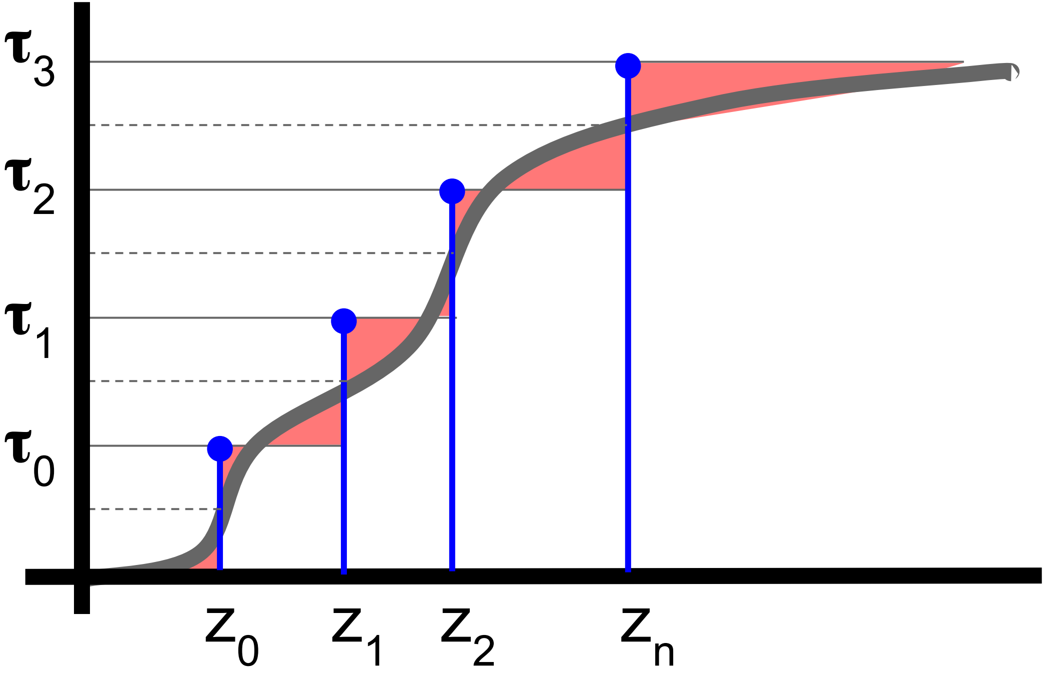


$$loss = \begin{cases} \tau x, & \text{for } x \geq 0 \\ (\tau - 1)x, & \text{for } x \geq 0 \end{cases}$$



Quantile Regression = projection in Wasserstein!

(on a uniform grid)



QR distributional Bellman operator

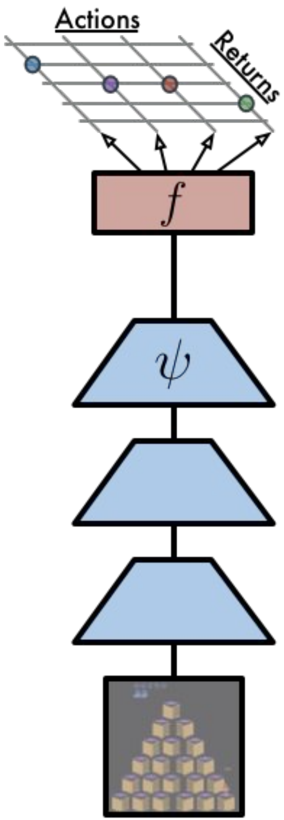
Theorem: $\boxed{\Pi_{QR} T^{\pi} \text{ is a contraction (in Wasserstein)}}$ [Dabney et al., 2018]

Intuition: quantile regression = projection in Wasserstein

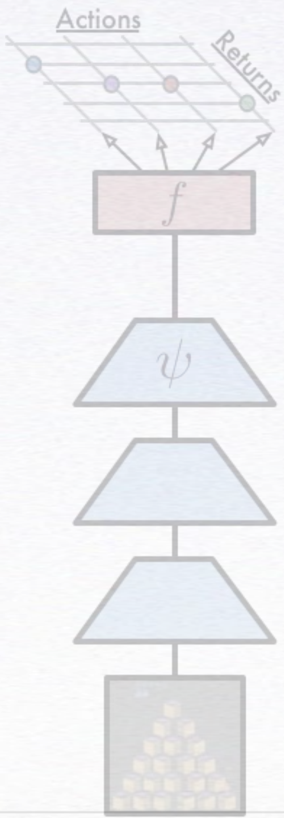
Reminder:

- T^{π} is a contraction (both in Cramer and Wasserstein)
- $\Pi_n T^{\pi}$ is a contraction (in Cramer)

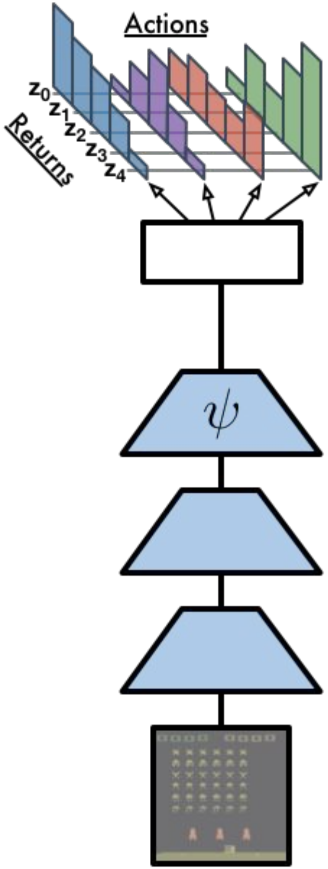
DQN



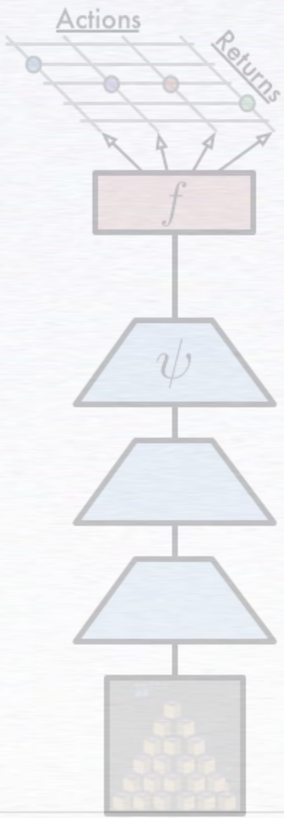
DQN



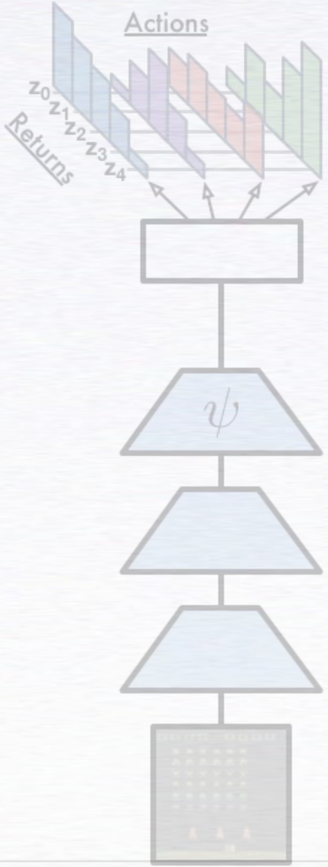
C51



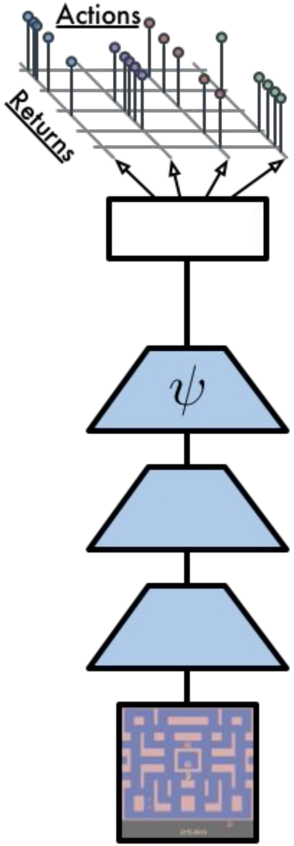
DQN



C51



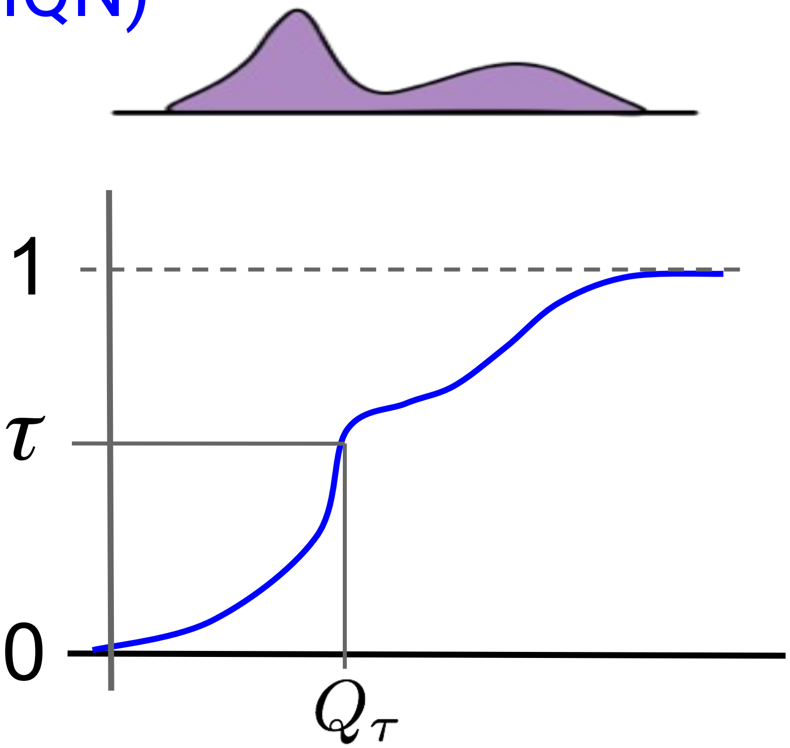
QR-DQN



Quantile-Regression DQN

	Mean	Median
DQN	228%	79%
Double DQN	307%	118%
Dueling	373%	151%
Prio. Duel.	592%	172%
C51	701%	178%
QR-DQN	864%	193%

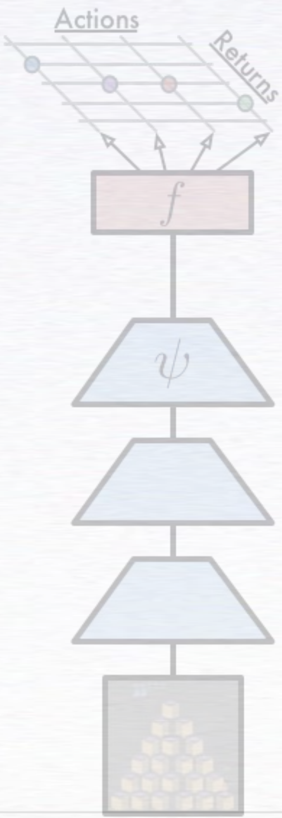
Implicit Quantile Networks (IQN)



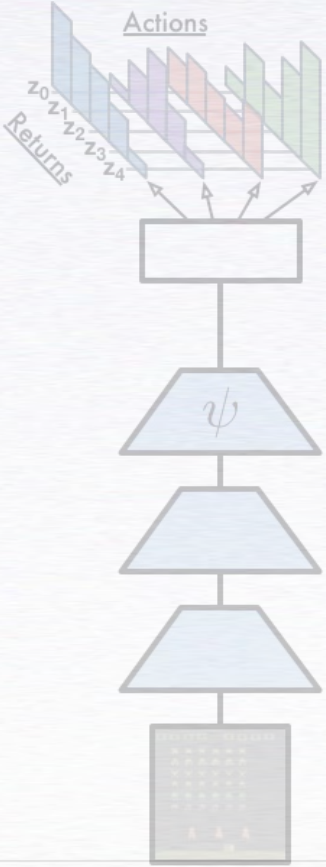
Learn a parametric inverse CDF

$$\tau \mapsto F_Z^{-1}(\tau)$$

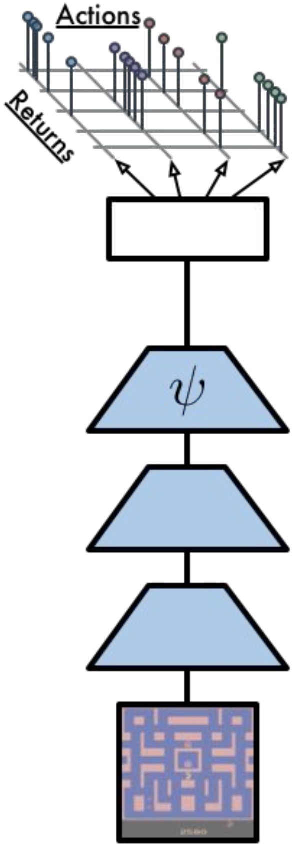
DQN



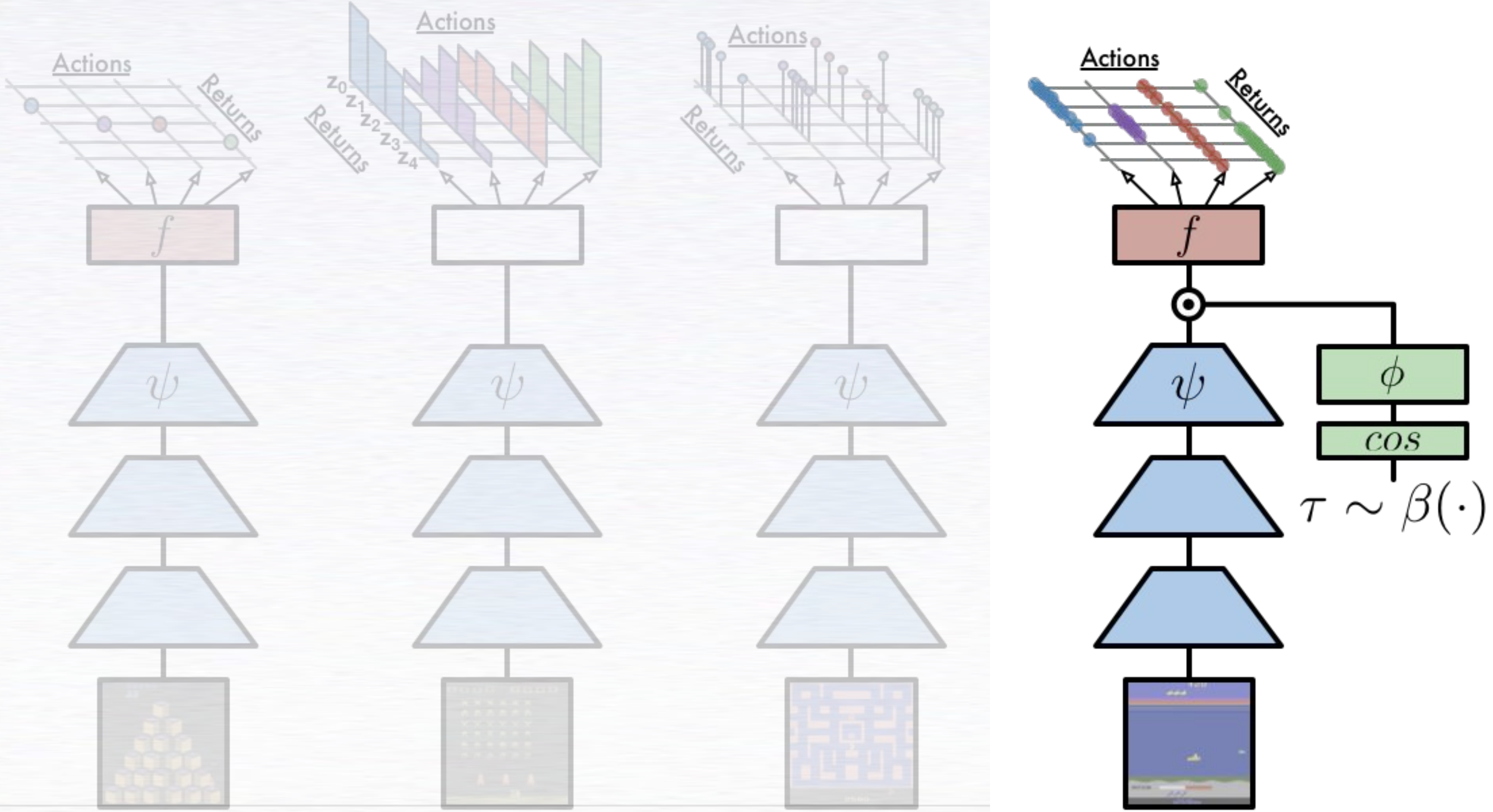
C51



QR-DQN



DQN C51 QR-DQN **IQN**

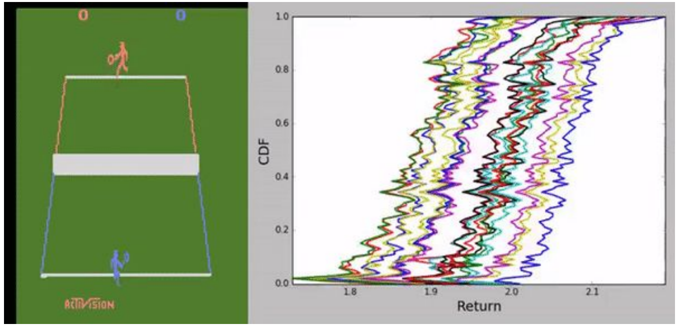


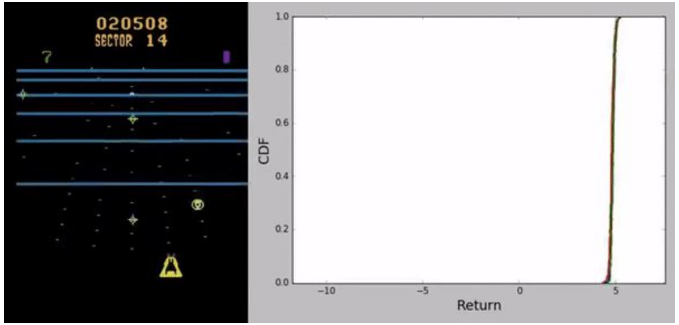
Implicit Quantile Networks for TD

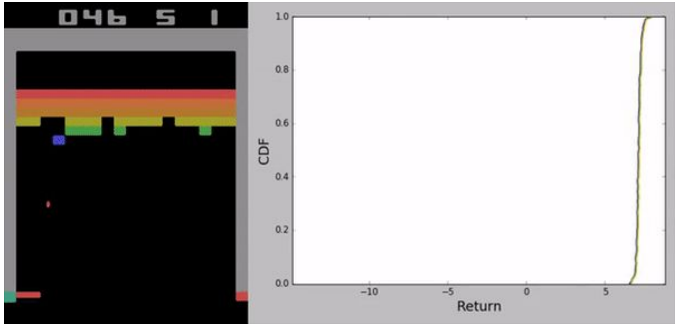
$$\tau \sim \mathcal{U}[0,1], \qquad z = Z_\tau(x_t, a_t)$$
$$\tau' \sim \mathcal{U}[0,1], \qquad z' = Z_\tau(x_{t+1}, a^*)$$

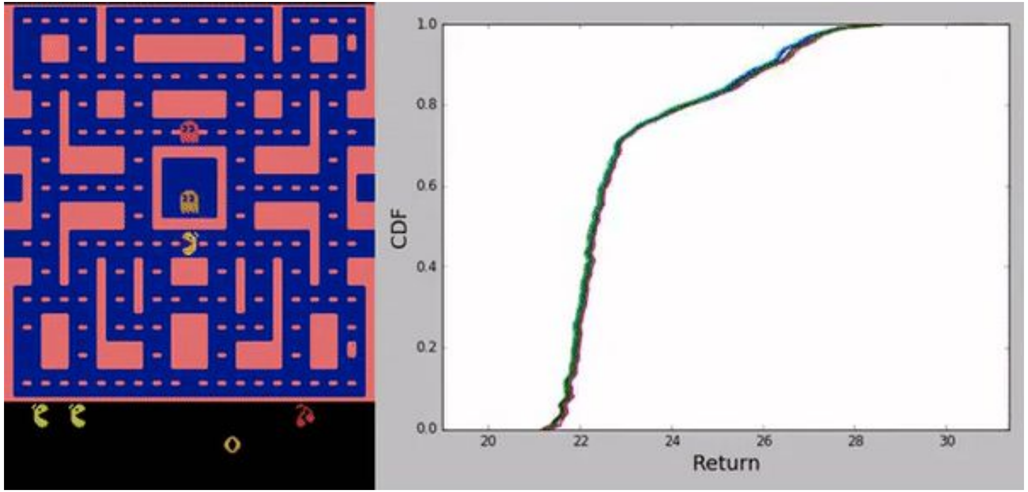
$$\delta_t = r_t + \gamma z' - z$$

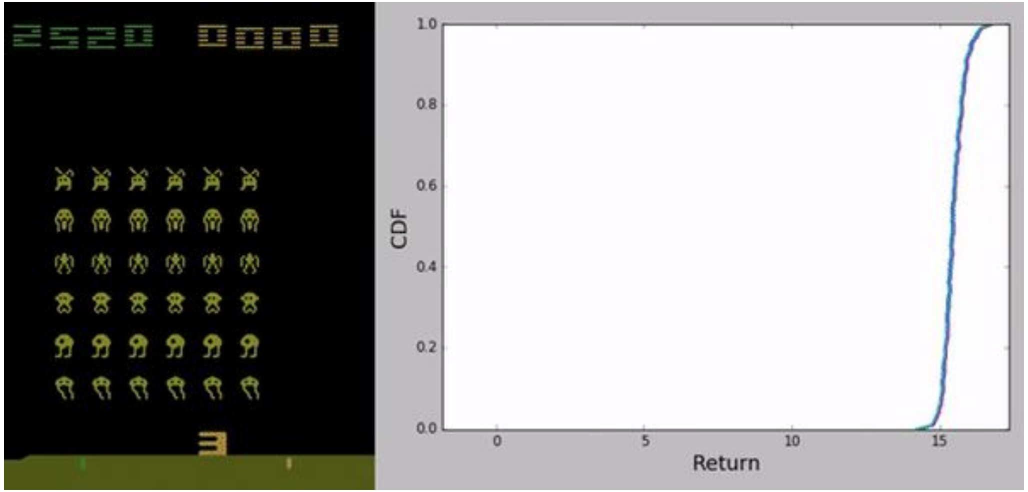
$$\text{QR loss: } \rho_\tau(\delta) = \delta(\tau - \mathbb{I}_{\delta<0})$$


ACTIVISION
CDF
Return
1.0
0.8
0.6
0.4
0.2
0.0
1.8
1.9
2.0
2.1


020508
SECTOR 14
CDF
Return
1.0
0.8
0.6
0.4
0.2
0.0
−10
−5
0
5


CDF
Return
1.0
0.8
0.6
0.4
0.2
0.0
−10
−5
0
5


CDF
Return
1.0
0.8
0.6
0.4
0.2
0.0
20
22
24
26
28
30


CDF
Return
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15

Implicit Quantile Networks

	Mean	Median	Human starts
DQN	228%	79%	68%
Prio. Duel.	592%	172%	128%
C51	701%	178%	116%
QR-DQN	864%	193%	153%
IQN	**1019%**	**218%**	**162%**

Almost as good as SOTA (Rainbow/Reactor) which combine prio/dueling/categorical/...

What is going on?

- We learn these distributions, but in the end **we only use their mean**

What is going on?

- We learn these distributions, but in the end **we only use their mean**

Non-trivial **interactions between deep learning and RL**:

- Learn richer representations
  - Same signal to learn from but more predictions
  - More predictions → richer signal → better representations
  - Can better disambiguate between different states (state aliasing)
- Density estimation instead of l2-regressions
  - Express RL in terms of usual tools in deep learning
  - Variance reduction

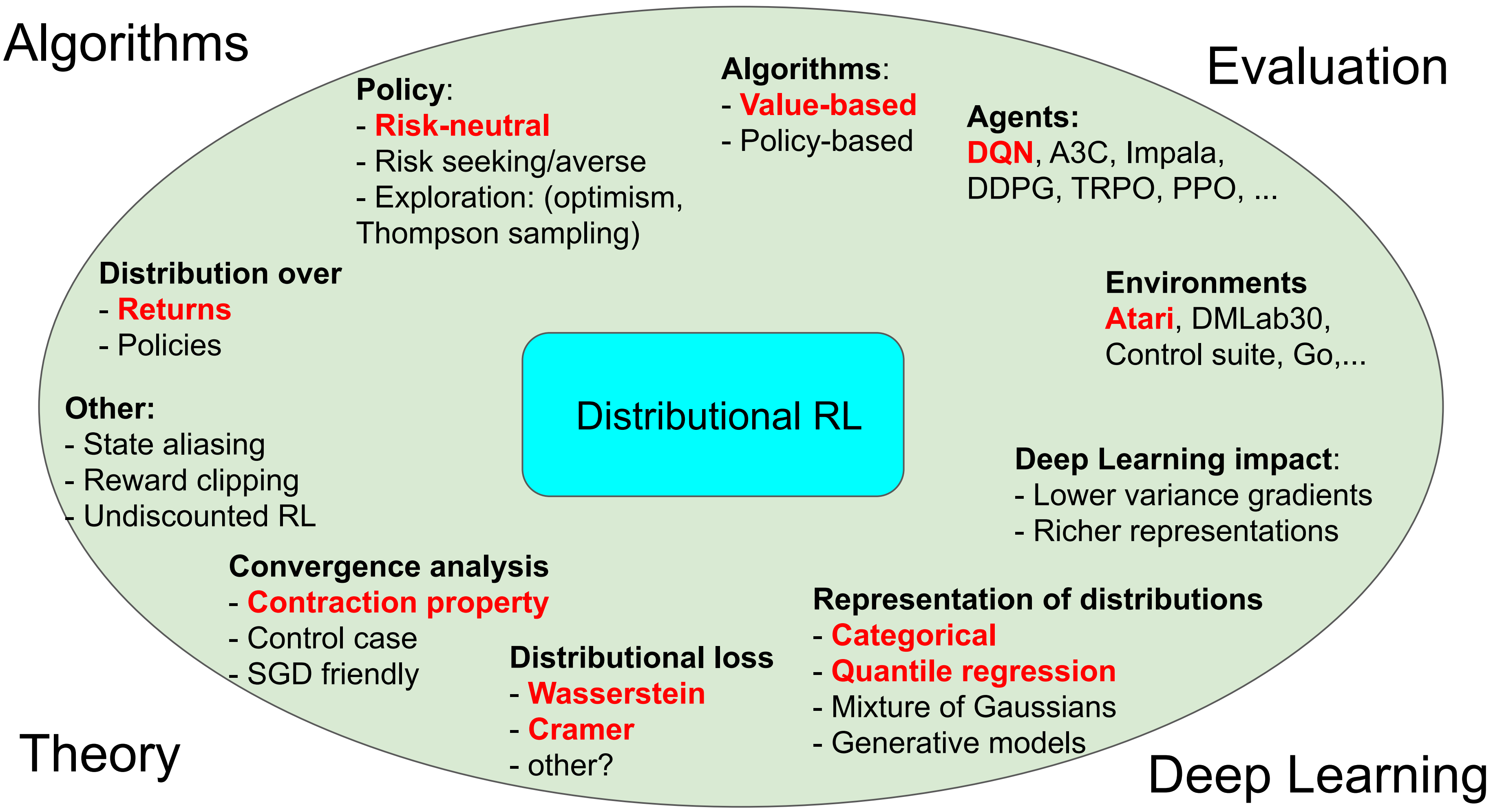
What is going on?

- We learn these distributions, but in the end **we only use their mean**

Non-trivial **interactions between deep learning and RL**:

- Learn richer representations
  - Same signal to learn from but more predictions
  - More predictions → richer signal → better representations
  - Can better disambiguate between different states (state aliasing)
- Density estimation instead of l2-regressions
  - Express RL in terms of usual tools in deep learning
  - Variance reduction

Now maybe we could start using those distributions? (e.g, risk-sensitive control, exploration, …)


Algorithms
Evaluation
Theory
Deep Learning
Distributional RL
Policy:
- Risk-neutral
- Risk seeking/averse
- Exploration: (optimism, Thompson sampling)
Algorithms:
- Value-based
- Policy-based
Agents:
DQN, A3C, Impala, DDPG, TRPO, PPO, ...
Distribution over
- Returns
- Policies
Environments
Atari, DMLab30, Control suite, Go,...
Other:
- State aliasing
- Reward clipping
- Undiscounted RL
Deep Learning impact:
- Lower variance gradients
- Richer representations
Convergence analysis
- Contraction property
- Control case
- SGD friendly
Distributional loss
- Wasserstein
- Cramer
- other?
Representation of distributions
- Categorical
- Quantile regression
- Mixture of Gaussians
- Generative models

References:

- *A distributional perspective on reinforcement learning*, (Bellemare, Dabney, Munos, ICML 2017)
- *An Analysis of Categorical Distributional Reinforcement Learning*, (Rowland, Bellemare, Dabney, Munos, Teh, AISTATS 2018)
- *Distributional reinforcement learning with quantile regression*, (Dabney, Rowland, Bellemare, Munos, AAAI 2018)
- *Implicit Quantile Networks for Distributional Reinforcement Learning*, (Dabney, Ostrovski, Silver, Munos, ICML 2018)

Thanks!