535514: Reinforcement Learning
Lecture 24 — QR-DQN and IQN

Ping-Chun Hsieh

May 16, 2024

On-Policy vs Off-Policy Methods

<table>
<tr><th></th><th>Policy Optimization</th><th>Value-Based</th><th>Model-Based</th><th>Imitation-Based</th></tr>
<tr><td>On-Policy</td><td>Exact PG
REINFORCE (w/i baseline)
A2C
On-policy DAC
TRPO
Natural PG (NPG)
PPO-KL & PPO-Clip
RLHF by PPO-KL</td><td>Epsilon-Greedy MC
Sarsa
Expected Sarsa</td><td rowspan="2">Model-Predictive Control (MPC)
PETS</td><td rowspan="2">IRL
GAIL
IQ-Learn</td></tr>
<tr><td>Off-Policy</td><td>Off-policy DPG & DDPG
Twin Delayed DDPG (TD3)</td><td>Q-learning
Double Q-learning
DQN & DDQN
Rainbow
C51 / QR-DQN / IQN
Soft Actor-Critic (SAC)</td></tr>
</table>

Quick Review: Distributional Bellman

- Sample action-value $Z^\pi(s,a)$: sample return if we start from state s and take action a, and then follow policy π

$$Q^\pi(s,a) = \mathbb{E}[Z^\pi(s,a)] = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$$

- Distributional Bellman operator $B^\pi : \mathscr{Z} \to \mathscr{Z}$

$$B^\pi Z(s,a) :\overset{D}{=} r(s,a) + \gamma P^\pi Z(s,a)$$

where $P^\pi Z(s,a) :\overset{D}{=} Z(s',a')$

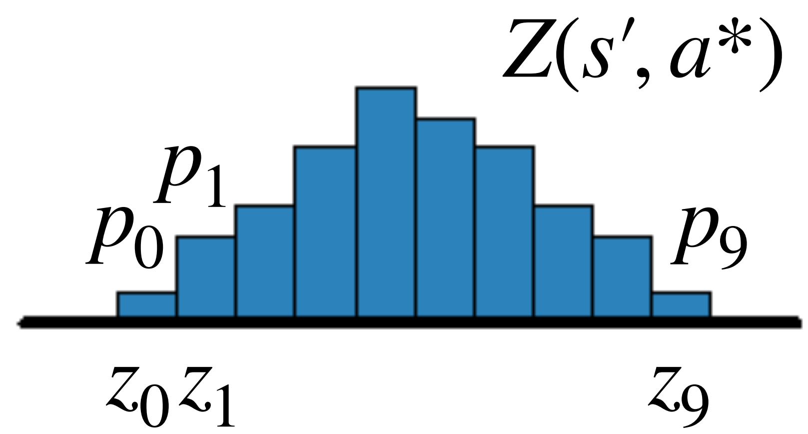
$$s' \sim P(\cdot \mid s,a),\ a' \sim \pi(\cdot \mid s')$$

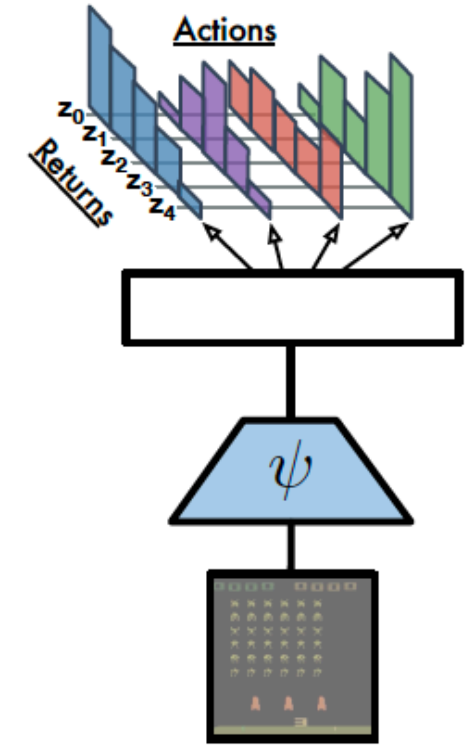
- Distributional optimality operator B^*: The B^π resulting from a greedy policy π (what does "greedy" mean here?)

Quick Review: C51

$$a^* = \arg\max_a \mathbb{E}[Z(s', a)]$$

(C1) Categorical distributions for parametrizing $Z_\theta(s, a)$





C51

(C2) Mimicking B^* for learning with sample transitions (s, a, r, s')

(C3) Cramer Projection Φ for support mismatch caused by $B^* Z_\theta(s, a)$

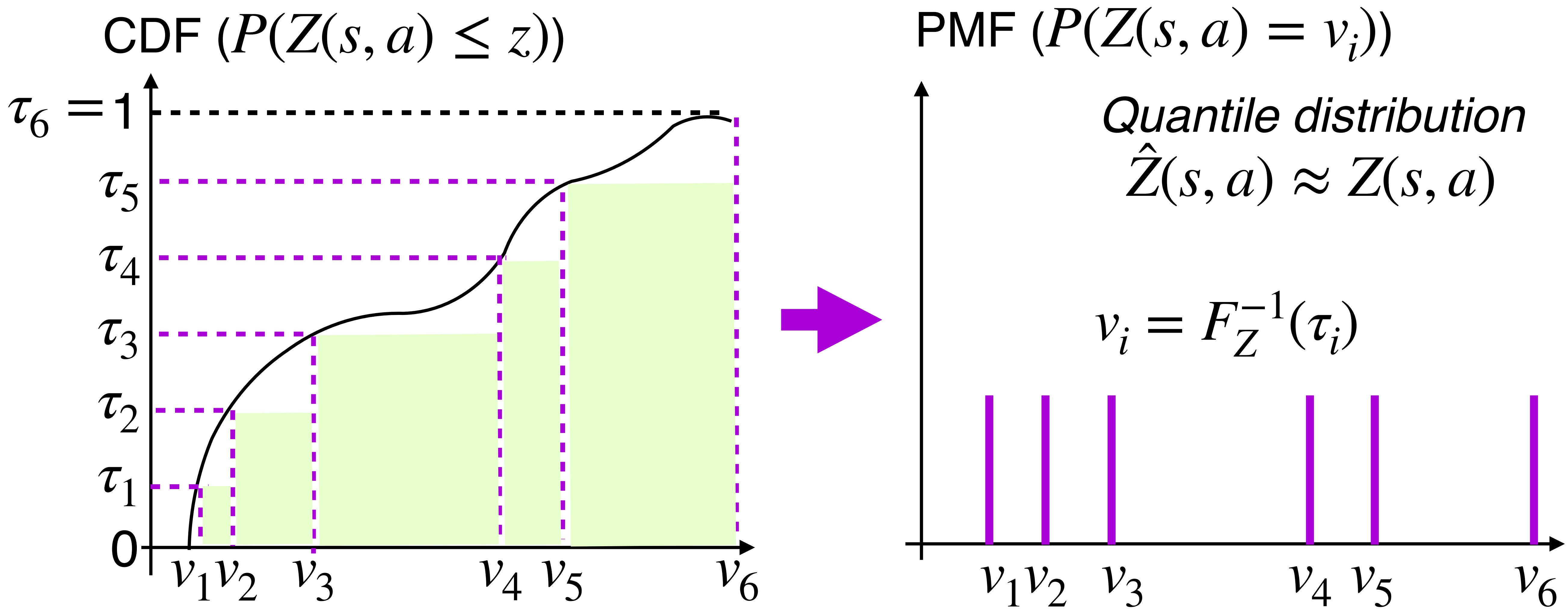
(C4) Minimize $L_{C51}(s, a, r, s'; \theta) := D_{KL}(\Phi B^* Z_{\bar{\theta}}(s, a) \| Z_\theta(s, a))$

QR-DQN

Dabney et al., Distributional Reinforcement Learning with Quantile Regression, AAAI 2018

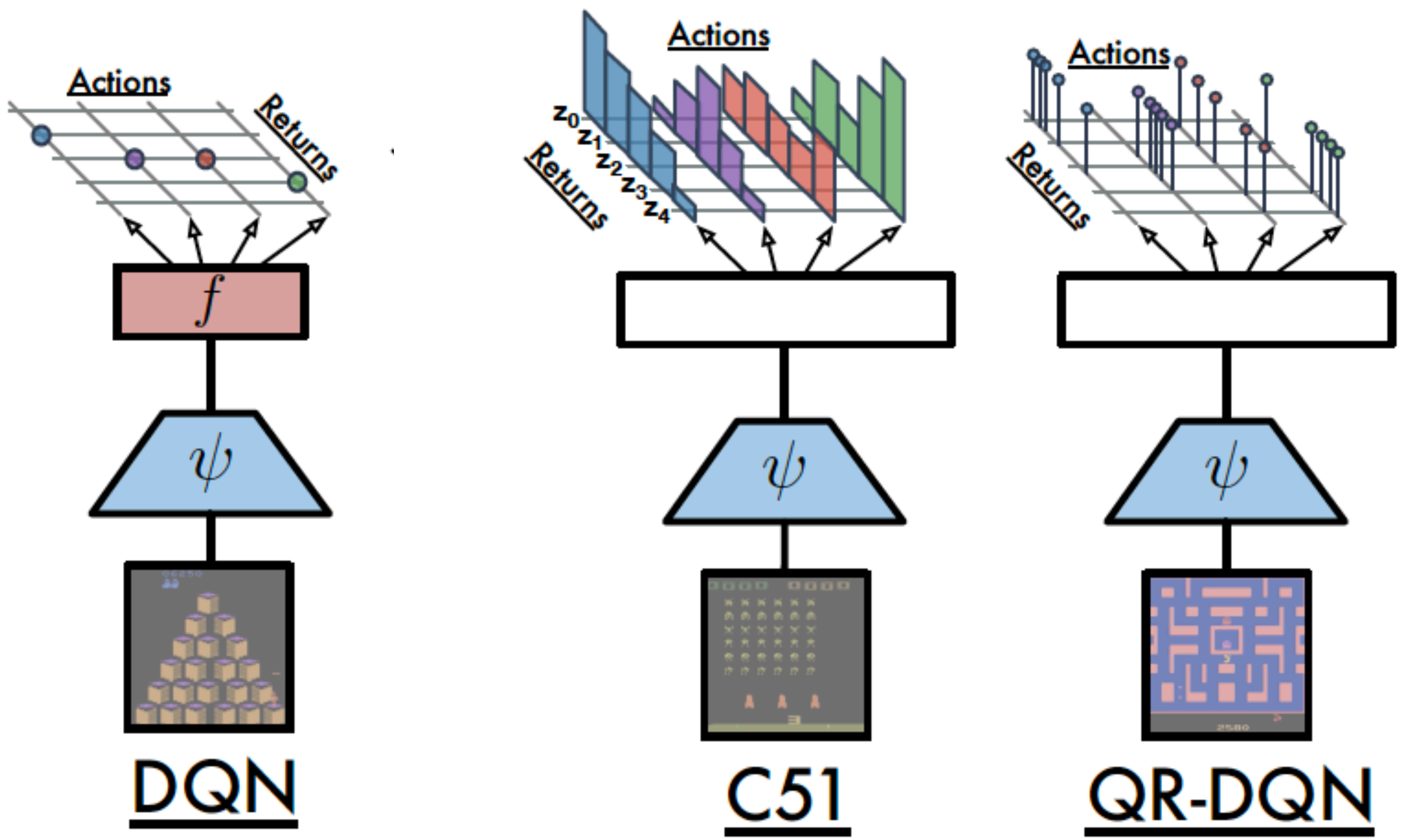
Quantile-Based Parametrization of $Z(s, a)$

- **Idea: Express $Z(s, a)$ using CDF (instead PDF)**



Quantile function: $F_Z^{-1}(\tau) := \inf\{z : P(Z \leq z) \geq \tau\}$

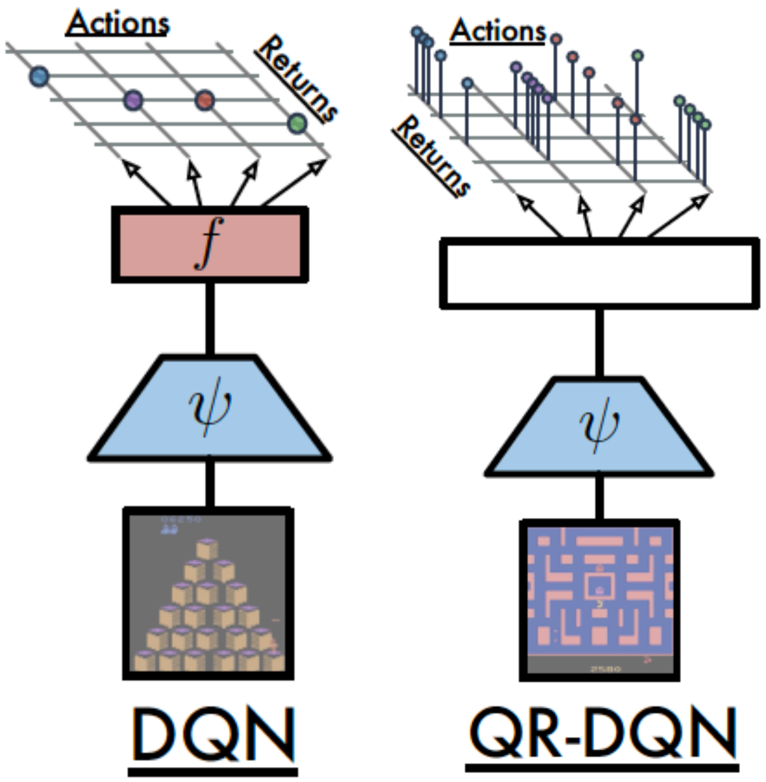
A Comparison of NN Architecture



QR-DQN: Another Popular Distributional DQN

- Q1: How to express $Z(s, a)$?

(D1) **Quantile** distributions for $Z_\theta(s, a)$



- Q2: How to update $Z(s, a)$ during training?

(D2) Mimicking B^* for learning with sample transitions (s, a, r, s')

(D3) Minimize $L_{QR}(s, a, r, s'; \theta) := D(B^* Z_{\bar{\theta}}(s, a) \| Z_\theta(s, a))$

Quantile Regression DQN (Formally)

Step 1: Initialize $Z_\theta(s, a)$ and initial state s_0

Step 2: For each step $t = 0,1,2,\cdots$

Select a_t using ε-greedy w.r.t $Q(s_t, a) \equiv \mathbb{E}[Z_\theta(s_t, a)]$

Observe (r_{t+1}, s_{t+1}) and store $(s_t, a_t, r_{t+1}, s_{t+1})$ in the buffer

Draw a mini-batch of samples B from the replay buffer

Update θ by minimizing QR loss as follows:

$$\theta \leftarrow \theta - \alpha \nabla_\theta \sum_{(s,a,r,s') \in B} L_{QRDQN}(s, a, r, s'; \theta)$$

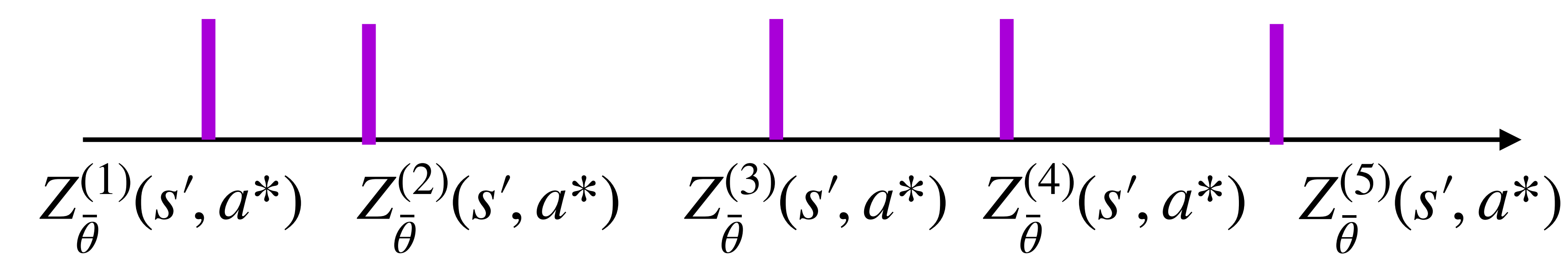
Under quantile distributions, $\mathbb{E}[Z_\theta(s, a)] = \sum_{i=1}^{N} \frac{1}{N} Z_\theta^{(i)}(s, a)$

(D2) Mimicking B^* for Learning With Sample Transitions

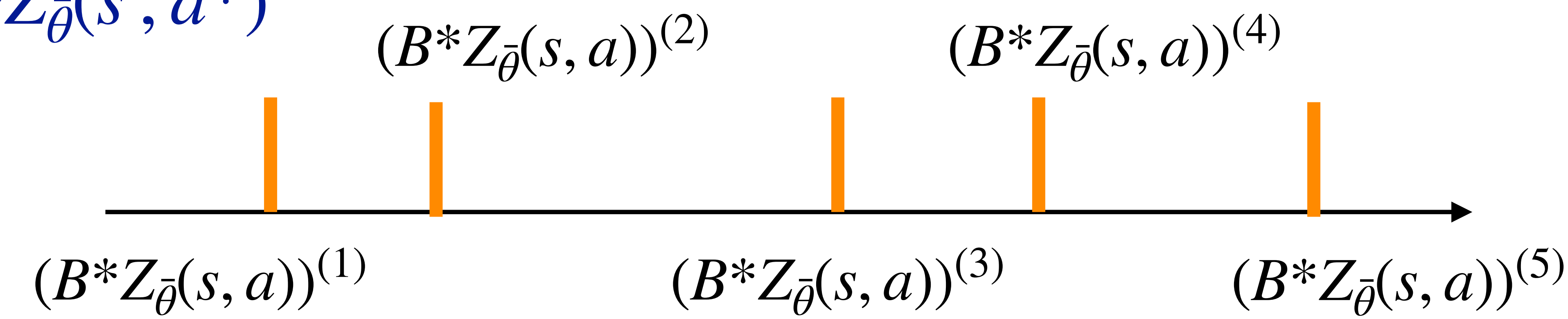
- Here we presume a <u>greedy</u> policy w.r.t Q function for B^*
- Question: Given only transitions (s, a, r, s'), how to enforce B^* to update $Z(s, a)$ on *quantile* distributions?

$$a^* = \arg\max_a Q(s', a) \equiv \arg\max_a \mathbb{E}[Z_{\bar{\theta}}(s', a)]$$

$Z_{\bar{\theta}}(s', a^*)$



$B^*Z_{\bar{\theta}}(s, a) = r + \gamma Z_{\bar{\theta}}(s', a^*)$



(D3) Loss Function

- We still need to choose a "**dissimilarity**" function $D(\,\cdot \| \cdot\,)$ in $L_{QRDQN}(s, a, r, s'; \theta) := D(B^* Z_{\bar{\theta}}(s, a) \| Z_{\theta}(s, a))$

- There are many possibilities, e.g., total variation or KL divergence

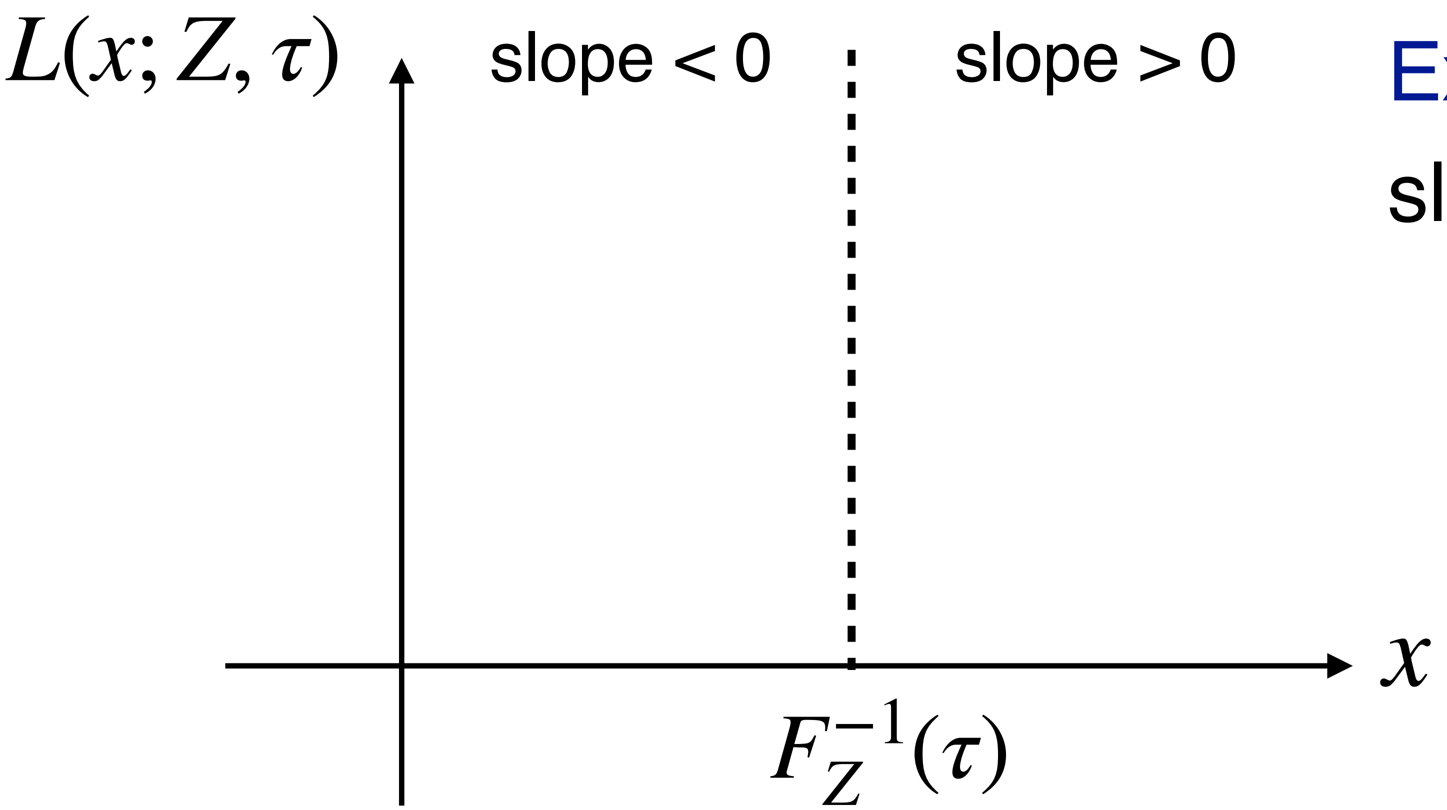
- QR-DQN uses the quantile regression loss
  - Motivation: Both $B^* Z_{\bar{\theta}}(s, a)$ and $Z_{\theta}(s, a)$ are quantile distributions

Quantile Regression Loss

- Idea: Finding a quantile $F_Z^{-1}(\tau)$ by minimizing loss $L(x; Z, \tau)$

$$F_Z^{-1}(\tau) = \arg\min_{x \in \mathbb{R}} L(x; Z, \tau)$$

- $L(x; Z, \tau)$ is *easy-to-optimize* when it is strictly convex



Example: Let $L(x; Z, \tau)$ have a slope of $F_Z(x) - \tau$, for all $x \in \mathbb{R}$

(Aka “quantile regression” loss)

Roger Koenker, Quantile Regression, 2005

The Quantile Regression Loss

- Given that the derivative of $L(x; Z, \tau)$ is $F_Z(x) - \tau$, we can recover the QR loss by integration

Quantile regression (QR) loss:

$$L_{QR}(x; Z, \tau) = (\tau - 1)\int_{-\infty}^{x} (z - x)dF_Z(z) + \tau \int_{x}^{\infty} (z - x)dF_Z(z)$$

(It is easy to verify that $\frac{d}{dx}L_{QR}(x; Z, \tau) = F_Z(x) - \tau$ by the Leibniz integral rule)

$$\frac{d}{dx}\left(\int_{a(x)}^{b(x)} f(x,t)\,dt\right)$$
$$= f\big(x, b(x)\big) \cdot \frac{d}{dx}b(x) - f\big(x, a(x)\big) \cdot \frac{d}{dx}a(x) + \int_{a(x)}^{b(x)} \frac{\partial}{\partial x} f(x,t)\,dt$$

Alternative expression of QR loss:

$$\rho_\tau(y) := y(\tau - \mathbb{I}\{y < 0\})$$

$$L_{QR}(x; Z, \tau) = E_Z[\rho_\tau(Z - x)]$$

Summary: Loss Function of QR-DQN

$$L_{QRDQN}(s,a,r,s';\theta) := \sum_{i=1}^{N} L_{QR}(B^* Z_{\bar{\theta}}(s,a); Z_\theta(s,a), \tau_i)$$

$$= \sum_{i=1}^{N} \mathbb{E}_{z \sim B^* Z_{\bar{\theta}}(s,a)}[\rho_{\tau_i}(z - Z_\theta(s,a))]$$

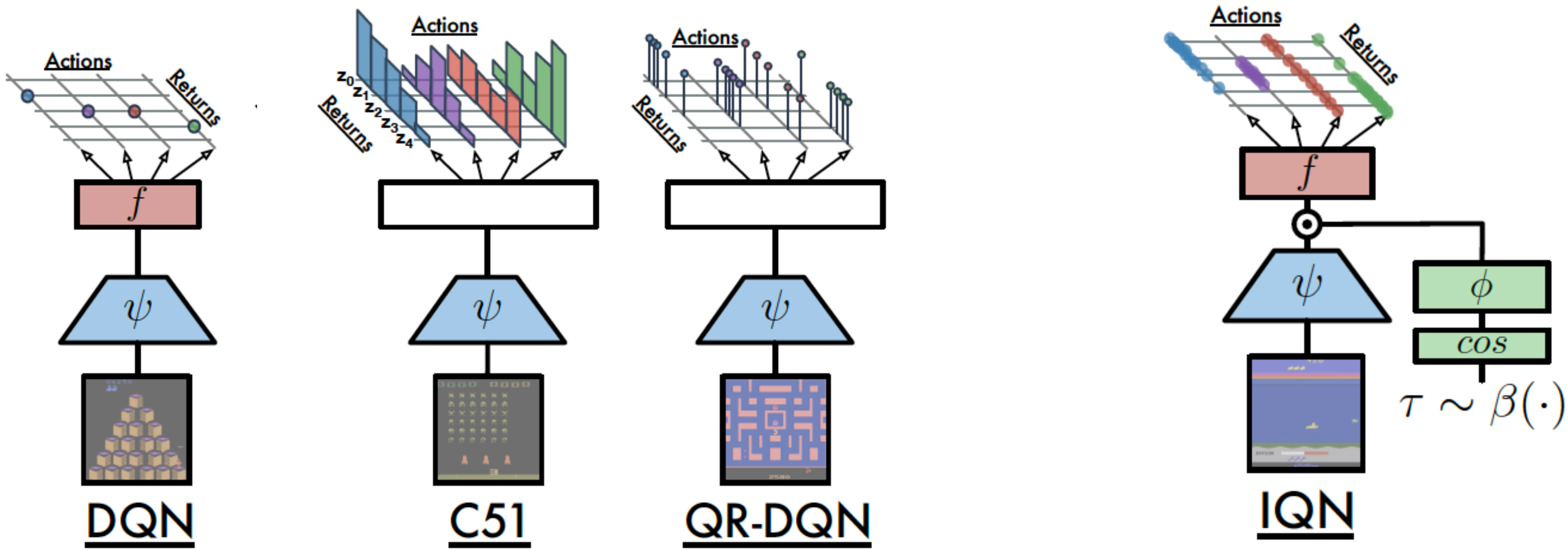
- Question: Is $L_{QRDQN}(s,a,r,s';\theta)$ easy to compute during training?

Implicit Quantile Networks (IQN)

Dabney et al., Implicit Quantile Networks for Distributional Reinforcement Learning, ICML 2018

IQN: A Generative Approach to Distributional RL

- An illustrative comparison of distributional Q-learning methods



Distributional RL via explicitly expressing the distribution $Z(s, a)$

Distributional RL via a generative model for distribution $Z(s, a)$

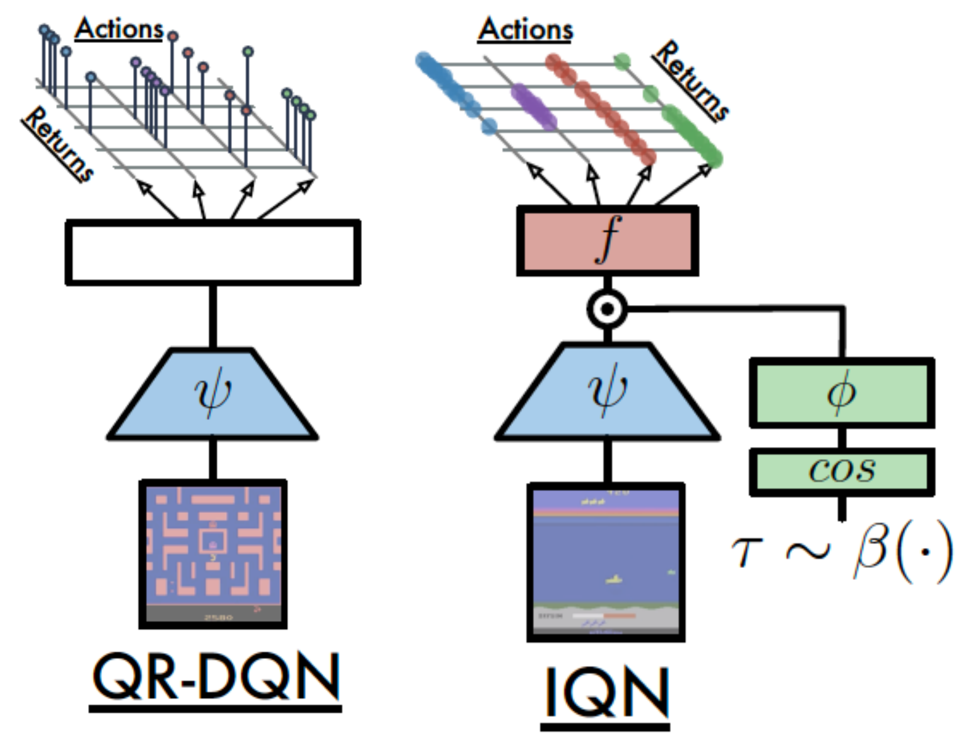
➡ Need sufficiently many atoms or quantiles for an accurate representation of $Z(s, a)$

(Figure credit: Will Dabney)

Calculate QR Loss by *Sampling*

QR loss:

$$\rho_\tau(y) := y(\tau - \mathbb{I}\{y < 0\})$$
$$L_{QR}(x; Z, \tau) = E_{z \sim Z}[\rho_\tau(z - x)]$$



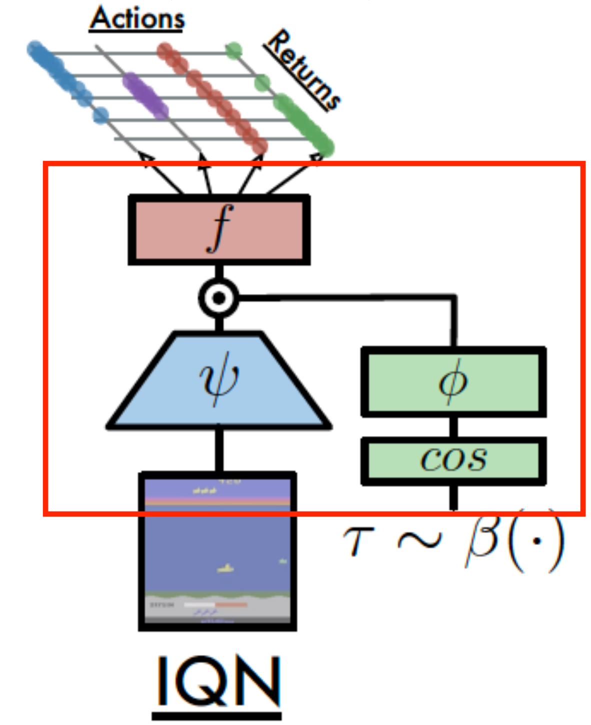
- Recall QR-DQN:
  - The QR loss is calculated explicitly
  - $Z \Rightarrow$ target distribution induced by $\{\bar{\theta}_1, \cdots, \bar{\theta}_N\}$
- Idea: A practical way to calculate the QR loss is *sampling*!

$$L_{QR}(x; Z, \tau) \approx$$

- IQN **implicitly** parameterizes Z by constructing a generator for Z

QR Loss and Inverse Transform Sampling

$\text{IQN}(s, \tau; a) \equiv F^{-1}$



QR loss:

$$\rho_\tau(y) := y(\tau - \mathbb{I}\{y < 0\})$$

$$L(x; Z, \tau) = E_{z \sim Z}[\rho_\tau(z - x)] \approx \frac{1}{K}\sum_{k=1}^{K} \rho_\tau(z_k - x)$$

$$(z_1, \cdots, z_K \sim Z)$$

Inverse Transform Sampling (ITS): Generate any random variable with CDF F from a uniform random variable

1. Generate a random variable $U \sim \text{Unif}(0,1)$
2. Let $X = F^{-1}(U)$, where $F^{-1}(u) := \inf\{z : F(z) \geq u\}$

- ITS is essentially a generative approach!

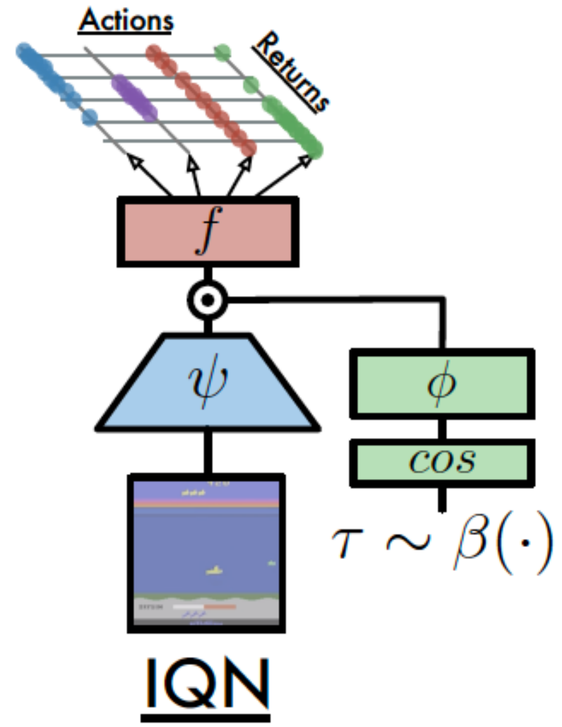
Calculating QR Loss in IQN

QR loss:

$$\rho_\tau(y) := y(\tau - \mathbb{I}\{y < 0\})$$

$$L(x; Z, \tau) = E_{z \sim Z}[\rho_\tau(z - x)] \approx \frac{1}{K}\sum_{k=1}^{K} \rho_\tau(z_k - x)$$

$$(z_1, \cdots, z_K \sim Z)$$



(Recall that Z corresponds to the target distribution in QR-DQN)

At each update, given (s, a, r, s'), for a given $\tau \in [0,1]$:

1. Draw $\tau'_1, \cdots, \tau'_K \sim$ Unif$(0,1)$ ←a generative step!
2. Get $z_1, \cdots, z_K$ by $z_i = r + \gamma \cdot \overline{\text{IQN}}(s', a'; \tau'_i)$
3. QR loss in IQN = $\frac{1}{K}\sum_{i=1}^{K} \rho_\tau(z_i - \text{IQN}(s, a; \tau))$

(can be readily extended to multiple τ)

IQN is closely related to the reparameterization trick

- Suppose we want to compute a loss $L(\theta) = E_{X\sim p_\theta}[f(X)]$
  - X is a random variable, and p_θ is the underlying distribution of X
- Question: $\nabla_\theta L(\theta)$ = ?

$$\nabla_\theta L(\theta) = \nabla_\theta E_{X\sim p_\theta}[f(X)] = \nabla_\theta\left(\int f(x)p_\theta(x)dx\right)$$
$$= \int \left(f(x)\frac{1}{p_\theta(x)}\nabla_\theta p_\theta(x)\right)p_\theta(x)dx$$
$$= \int \left(f(x)\underbrace{\nabla_\theta \log p_\theta(x)}_{\text{Easy to evaluate?}}\right)p_\theta(x)dx$$

- Reparameterization trick: $\varepsilon \sim p(\varepsilon)$, $L(\theta) = E_{\varepsilon\sim p}[g_\theta(\varepsilon)]$

$$\nabla_\theta L(\theta) = \nabla_\theta E_{\varepsilon\sim p}[g_\theta(\varepsilon)] = E_{\varepsilon\sim p}[\nabla_\theta g_\theta(\varepsilon)] \approx \frac{1}{K}\sum_{i=1}^{K}\nabla_\theta g_\theta(\varepsilon_i)$$
$$(\varepsilon_1, \cdots, \varepsilon_K \sim p)$$

Kingma and Welling, Auto-Encoding Variational Bayes, ICLR 2014