Distributional Reinforcement Learning

Rémi Munos

Marc Bellemare, Will Dabney, Georg Ostrovski, Mark Rowland

DeepMindParis
Deep RL is already a successful empirical research domain.
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

I learned to ride with RL...
The RL agent in its environment

\[ x_{t+1} \sim p(\cdot | x_t, a_t) \]

**The agent**

**The environment**
2 core ingredients of RL

Credit assignment problem:
  which actions are responsible for a reward?
  → Value-based methods ([Bellman 1957]'s Dynamic programming)
  → Policy-based methods ([Pontryagin 1957]'s Maximum principle)

Representation problem:
  how to represent functions, models and policies?
  → use deep learning!
  → *DeepRL*
Bellman’s dynamic programming

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

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

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:

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

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

- Optimal policy $\pi^*(x) = \arg \max_a Q^*(x, a)$
Use a neural net for approximating the value function

\[ \text{State } x \rightarrow \text{Convolution} \rightarrow \text{Convolution} \rightarrow \text{Fully connected} \rightarrow \text{Fully connected} \rightarrow \text{Q-value } Q_w(x,a) \]

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 $x_t, a_t \rightarrow x_{t+1}$,

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

- Minimize loss

  $$(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))^2$$

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

Remi Munos DeepMind
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

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

**Properties:** DQN is off-policy, and uses 1-step bootstrapping.
DQN Results in Atari

At this point the agent finds and exploits the best strategy of tunnelling and then hitting the ball behind the wall

The same algorithm learns to play 57 games
Improvements since Nature DQN

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

$$Q(s, a) \gets r(s, a) + \gamma \max_{a'} Q(s', a', w_{\text{target}})$$

[van Hasselt et al., 2015]

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

$$|r + \gamma \max_{a'} Q(s', a', w_{\text{target}}) - Q(s, a, w)|$$

[Schaul et al., 2015]

- **Dueling network**: Split Q-network into two channels
  - Action-independent value function $V(s, v)$
  - Action-dependent advantage function $A(s, a, w)$

$$Q(s, a) = V(s, v) + A(s, a, w)$$

[Wang et al., 2015]
Other improvements

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

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

  [Bellemare et al., 2015]

- **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:

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

  [Munos et al., 2016]

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

- Introduction
- Elements of theory
- Neural net representations
- Experiments on Atari
- Conclusion
Intro to distributional RL

Expected immediate reward

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

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

\[ R(x_0) + \gamma R(x_1) + \gamma^2 R(x_2) \ldots \]

- Returns are often complex, multimodal
- Modelling the expected return hides this intrinsic randomness
- Model all possible returns!
The r.v. Return $Z^\pi(x, a) = \sum_{t \geq 0} \gamma^t r(x_t, a_t) \bigg|_{x_0 = x, a_0 = a, \pi}$

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:

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

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

Does this equation make sense?
Example

Reward = Bernoulli ($\frac{1}{2}$), discount factor $\gamma = \frac{1}{2}$

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

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

$R = \left\{ \begin{array}{cc} 1 & \text{w.p. } \frac{1}{2} \\ 0 & \text{w.p. } \frac{1}{2} \end{array} \right.$
Example

Reward = Bernoulli (½), discount factor $\gamma = \frac{1}{2}$

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

Reward = Bernoulli (½), discount factor $\gamma = \frac{1}{2}$

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: $Z = B\left(\frac{1}{2}\right) + \frac{1}{2} Z$

In terms of distribution: $\eta(z) = \frac{1}{2} \left(\delta(0) + \delta(1)\right) \ast 2\eta(2z)$

$= \eta(2z) + \eta(2(z - 1))$
Distributional Bellman operator

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

Does there exist a fixed point?
Properties

**Theorem** [Rowland et al., 2018]

\[ T' \pi \text{ is a contraction in Cramer metric} \]

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

**Theorem** [Bellemare et al., 2017]

\[ T' \pi \text{ is a contraction in Wasserstein metric,} \]

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

(but not in KL neither in total variation)

Intuition: the size of the support shrinks.
Distributional dynamic programming

For a given policy \( \pi \), 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 \( Z^\pi \)

And the iterate \( Z \leftarrow T^\pi Z \) converges to \( Z^\pi \)
The control case

Define the distributional Bellman optimality operator

\[ T_Z(x, a) \overset{D}{=} r(x, a) + \gamma Z(x', \pi_Z(x')) \]

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

Is this operator a contraction mapping?
The control case

Define the distributional Bellman optimality operator

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

where $x' \sim p(\cdot|x, a)$ 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 $Z_\varepsilon$

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

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

Thus the map $Z_\varepsilon \mapsto T Z_\varepsilon$ is not continuous
Theorem [Bellemare et al., 2017]

if the optimal policy is unique, then the iterates
\[ Z_{k+1} \leftarrow T Z_k \]
converge to \[ Z^{\pi^*} \]

**Intuition:** The distributional Bellman operator preserves the mean, thus the mean will converge to the optimal policy \( \pi^* \) eventually. If the policy is unique, we revert to iterating \( T^{\pi^*} \), 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 \( \{z_1, \ldots, z_n\} \)

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

Transition

$P^\pi Z$

$\gamma P^\pi Z$

$R + \gamma P^\pi Z$

$\prod_n T^\pi Z$
Projected Distributional Bellman Update

\[ P^\pi Z \]

\[ R + \gamma P^\pi Z \]

\[ \gamma P^\pi Z \]

\[ \Pi_n T^\pi Z \]
Projected Distributional Bellman Update

\[ P^\pi Z \]

\[ \gamma P^\pi Z \]

\[ R + \gamma P^\pi Z \]

\[ \Pi_n T^\pi Z \]

Reward / Shift
Projected Distributional Bellman Update

\[ P^\pi Z \]

\[ \gamma P^\pi Z \]

\[ R + \gamma P^\pi Z \]

\[ \Pi_\eta T^\pi Z \]

Fit / Project
Projected distributional Bellman operator

Let $\Pi_n$ be the projection onto the support (piecewise linear interpolation)

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

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

**Policy iteration:** iterate
- **Policy evaluation:**
  \[ Z_k = \prod_n T^{\pi_k} Z_k \]
- **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 \[ Z_{\pi^*} \], 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 $\pi^*$, then $Z \rightarrow Z_{\pi^*}$ and the resulting policy is optimal. [Rowland et al., 2018]
DeepRL implementation
DQN

[<NAME> et al., 2013]
Categorical DQN [Bellemare et al., 2017]
Categorical DQN
Randomness from future choices
## Results on 57 games Atari 2600

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
<th>Median</th>
<th>&gt;human</th>
</tr>
</thead>
<tbody>
<tr>
<td>DQN</td>
<td>228%</td>
<td>79%</td>
<td>24</td>
</tr>
<tr>
<td>Double DQN</td>
<td>307%</td>
<td>118%</td>
<td>33</td>
</tr>
<tr>
<td>Dueling</td>
<td>373%</td>
<td>151%</td>
<td>37</td>
</tr>
<tr>
<td>Prio. Duel.</td>
<td>592%</td>
<td>172%</td>
<td>39</td>
</tr>
<tr>
<td>C51</td>
<td>701%</td>
<td>178%</td>
<td>40</td>
</tr>
</tbody>
</table>
Categorical representation

$p_0, p_1, p_2, \ldots, p_{n-1}$

Fixed support, learned probabilities

$z_0, i\Delta z, \ldots$
Quantile Regression Networks

$z_0, z_1, z_2, \ldots, z_{n-1}$

Fixed probabilities, learned support
Inverse CDF learnt by Quantile Regression
l2-regression

\[ \text{loss} = x^2 \]
l1-regression

\[ \text{loss} = |x| \]
$$\text{loss} = \begin{cases} \\[0.5em] \frac{1}{4}x, & \text{for } x \geq 0 \\[0.2em] \frac{3}{4}x, & \text{for } x < 0 \end{cases}$$
\( \frac{3}{4} \)-quantile-regression

\[
\text{loss} = \begin{cases} 
\frac{3}{4} x, & \text{for } x \geq 0 \\
-\frac{1}{4} x, & \text{for } x \geq 0 
\end{cases}
\]
many-quantiles-regression

\[ \text{loss} = \begin{cases} 
  \tau x, & \text{for } x \geq 0 \\
  (\tau - 1)x, & \text{for } x \geq 0
\end{cases} \]
Quantile Regression = projection in Wasserstein! (on a uniform grid)
QR distributional Bellman operator

**Theorem:** \( \Pi_Q R T^\pi \) is a contraction (in Wasserstein) [Dabney et al., 2018]

Intuition: quantile regression = projection in Wasserstein

**Reminder:**
- \( T^\pi \) is a contraction (both in Cramer and Wasserstein)
- \( \Pi_n T^\pi \) is a contraction (in Cramer)
DQN
## Quantile-Regression DQN

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>DQN</td>
<td>228%</td>
<td>79%</td>
</tr>
<tr>
<td>Double DQN</td>
<td>307%</td>
<td>118%</td>
</tr>
<tr>
<td>Dueling</td>
<td>373%</td>
<td>151%</td>
</tr>
<tr>
<td>Prio. Duel.</td>
<td>592%</td>
<td>172%</td>
</tr>
<tr>
<td>C51</td>
<td>701%</td>
<td>178%</td>
</tr>
<tr>
<td>QR-DQN</td>
<td>864%</td>
<td>193%</td>
</tr>
</tbody>
</table>
Implicit Quantile Networks (IQN)

Learn a parametric inverse CDF

$$\tau \mapsto F_Z^{-1}(\tau)$$
Implicit Quantile Networks for TD

\[ \tau \sim U[0,1], \quad z = Z_\tau(x_t, a_t) \]
\[ \tau' \sim U[0,1], \quad z' = Z_\tau(x_{t+1}, a^*) \]
\[ \delta_t = r_t + \gamma z' - z \]

QR loss: \[ \rho_\tau(\delta) = \delta(\tau - \mathbb{1}_{\delta < 0}) \]
## Implicit Quantile Networks

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Median</th>
<th>Human starts</th>
</tr>
</thead>
<tbody>
<tr>
<td>DQN</td>
<td>228%</td>
<td>79%</td>
<td>68%</td>
</tr>
<tr>
<td>Prio. Duel.</td>
<td>592%</td>
<td>172%</td>
<td>128%</td>
</tr>
<tr>
<td>C51</td>
<td>701%</td>
<td>178%</td>
<td>116%</td>
</tr>
<tr>
<td>QR-DQN</td>
<td>864%</td>
<td>193%</td>
<td>153%</td>
</tr>
<tr>
<td>IQN</td>
<td>1019%</td>
<td>218%</td>
<td>162%</td>
</tr>
</tbody>
</table>

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, …)
Distributional RL

**Algorithms**

- **Policy:**
  - Risk-neutral
  - Risk seeking/averse
  - Exploration: (optimism, Thompson sampling)

- **Agents:**
  - Value-based
  - Policy-based
  - DQN, A3C, Impala, DDPG, TRPO, PPO, ...

- **Distributional loss:**
  - Wasserstein
  - Cramer
  - Other?

- **Convergence analysis:**
  - Contraction property
  - Control case
  - SGD friendly

- **Environments:**
  - Atari, DMLab30, Control suite, Go, ...

- **Deep Learning impact:**
  - Lower variance gradients
  - Richer representations

- **Other:**
  - State aliasing
  - Reward clipping
  - Undiscounted RL

**Theory**

- **Representation of distributions:**
  - Categorical
  - Quantile regression
  - Mixture of Gaussians
  - Generative models

**Deep Learning**

- **Evaluation**
  - Distribution over
    - Returns
    - Policies

- **Algorithms:**
  - Value-based
  - Policy-based

- **Distribution over:**
  - Returns
  - Policies

- **Deep Learning:**
  - Representation of distributions
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!