Normalizing Flows for Continous Distributional Reinforcement Learning

Ludvig Killingberg
Norwegian University of Science and Technology

Abstract

1 Introduction

As an agent interacts with an environment, the environment and the agent's policy interjects randomness into the return. Q-learning algorithms aim to capture the mean of the (discounted) cumulative reward from interracting with the environment. Distributional reinforcement learning (RL) in contrast considers the randomness in the reward and aims to capture the following random variable. In contrast to Q-learning algorithms, distributional reinforcement learning algorithms aim to model distribution of the discounted cumulative reward.

The difference in distributional reinforcement learning algorithms is mainly in how the distribution is modeled.

2 Background

We consider the standard RL setting, where interaction between agent and environment is modelled as a Markov Decision Process $(\mathcal{X}, \mathcal{A}, R, P, \gamma)$ (Puterman, 1994), where $\mathcal{X}$ and $\mathcal{A}$ denote the state and action space respectively, R denotes the state-action dependent reward function, $P(\cdot|x, a)$ denotes the transition probabilities, and $\gamma \in [0, 1]$ is the discount factor. A policy $\pi(\cdot|x)$ maps states to distributions over actions.

For an agent following policy π, the return Z^π, or *value distribution* (Bellemare et al., 2017), is the sum of discounted rewards along the agents trajectory

$$Z^\pi(x, a) := \sum_{t=0}^{\infty} \gamma^t R(x_t, a_t) \tag{1}$$

starting in state $x_0 = x$, with $a_0 = a$, $a_t \sim \pi(\cdot|x_t)$, and $x_t \sim P(\cdot|x_{t-1}, a_{t-1})$. The objective in RL can be summarized as finding the optimal policy π^* that maximizes the expected return $Q^\pi(x, a) := \mathbb{E}_{P,R,\pi}\left[Z^\pi(x, a)\right]$. The most common approach is to find the unique fixed point of the Bellman optimality operator (Bellman et al., 1957)

$$\mathcal{T}Q^*(x, a) := \mathbb{E}\left[R(x, a)\right] + \gamma \mathbb{E}_P\left[\max_{a' \in \mathcal{A}} Q^*(x', a')\right]. \tag{2}$$

To approximate Q^*, now usually modeled with a neural network Q_θ, Q-learning (Watkins and Dayan, 1989) iteratiely improves the estimate by minimizing the squared temporal difference (TD) error

$$\delta_t^2 = \left[r_t + \gamma \max_{a' \in \mathcal{A}} Q_\theta(x_t, a') - Q_\theta(x_t, a_t)\right]^2 \tag{3}$$

for samples (x_t, a_t, r_t, x_{t+1}) in trajectory T, observed as the agent interracts with the environment according to an ϵ-greedy policy over Q_θ.

2.1 Distributional Reinforcement Learning

Rather than reducing the optimization problem to the expected return, distributional reinforcement learning considers the full distribution of the random variable Z^π. Similar to the Bellman operator for the scalar $Q^\pi(x, a)$, we have a distributional Bellman operator

$$\mathcal{T}Z^\pi(x, a) :\stackrel{D}{=} R(x, a) + \gamma Z^\pi(X', A') \tag{4}$$

where $A \stackrel{D}{=} B$ denotes equality in distribution, i.e. random variable A and B have the same distribution functions.

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

2.1.1 Quantile Regression

2.2 Variational Inference

Lately, techniques for modelling complex distributions have have seen a surge in popularity, particularily for generating realistic images. Given a dataset $\mathcal{D}$, we would like to find the distribution of parameters $p(\boldsymbol{\theta}|\mathcal{D})$. This is known as the posterior distribution. Using full Bayesian inference we would calculate the posterior using Baye's theorem,

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}}. \tag{5}$$

Often, p is modelled by a neural network, which makes the denominator intractable to compute. Variational inference provides an alternative to computing the full posterior, by approximating $p(\boldsymbol{\theta}|\mathcal{D})$ with a distribution $q(\boldsymbol{\theta})$ from a family of distributions $\mathcal{Q}$. We do this by minimizing some measure of difference between the distributions. The KL-divergence is a common choice since it has some useful properties we will see later,

$$D_{\mathrm{KL}}\left(q(\boldsymbol{\theta})\|p(\boldsymbol{\theta}|\mathcal{D})\right) = \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\mathcal{D})}\right].$$

Although the is not a metric (it is not symmetric, and does not obey the triangle inequality), it is non-negative, and zero only when $q(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\mathcal{D})$. Finding the approximation to the posterior using the KL-divergence means solving the optimization problem,

$$q^* = \arg\min_{q\in\mathcal{Q}} D_{\mathrm{KL}}\left(q(\boldsymbol{\theta})\|p(\boldsymbol{\theta}|\mathcal{D})\right).$$

Unfortunately, this is still a hard optimization problem. We can, however, manipulate the KL-term to make the optimization problem tractable.

$$\begin{aligned}
&D_{\mathrm{KL}}\left(q(\boldsymbol{\theta})\|p(\boldsymbol{\theta}|\mathcal{D})\right) = \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\mathcal{D})}\right] \\
&= \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log q(\boldsymbol{\theta})\right] - \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log p(\boldsymbol{\theta}|\mathcal{D})\right] \\
&= \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log q(\boldsymbol{\theta})\right] - \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log \frac{p(\boldsymbol{\theta},\mathcal{D})}{p(\mathcal{D})}\right] \\
&= \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log q(\boldsymbol{\theta})\right] - \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log p(\boldsymbol{\theta},\mathcal{D})\right] + \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log p(\mathcal{D})\right] \\
&= \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log q(\boldsymbol{\theta})\right] - \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log p(\boldsymbol{\theta},\mathcal{D})\right] + \log p(\mathcal{D})
\end{aligned}$$

This shows that minimizing the KL-divergence is the same as maximizing the evidence lower bound (**ELBO**),

$$\mathrm{ELBO}(q;\boldsymbol{\theta},\mathcal{D}) = \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log p(\boldsymbol{\theta},\mathcal{D})\right] - \mathbb{E}_{q(\boldsymbol{\theta})}\left[\log q(\boldsymbol{\theta})\right].$$

How close the approximate distribution, $q(\boldsymbol{\theta})$), gets to the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ depends on the family of distributions $\mathcal{Q}$. We need to be able to quickly evaluate $\mathbb{E}_{q(\boldsymbol{\theta})}\left[p(\boldsymbol{\theta},\mathcal{D})\right]$ and $\mathbb{E}_{q(\boldsymbol{\theta})}\left[q(\boldsymbol{\theta}\right]$, but we also want flexible, complex distributions, such that $q^* \in \mathcal{Q}$ is close to $p(\boldsymbol{\theta}|\mathcal{D})$. Most early application of variational inference focused on mean-field for efficent inference. Assuming factorization in the distributional family,

$$\mathcal{Q} = \left\{ q(\boldsymbol{\theta}) \middle| \prod_{\theta_i\in\boldsymbol{\theta}} q(\theta_i) \right\}, \tag{6}$$

is a strong restriction. It is unlikely that $p(\boldsymbol{\theta}|\mathcal{D})$ will factorize, and thus not be close to any $q \in \mathcal{Q}$. The nature of the reverse KL-divergence will mean that the approximate posterior always underestimates the variance of the true posterior.

2.2.1 Normalizing Flows

Normalizing flows is a method of transforming simple distributions into more complex ones using parameterized functions. The change of variables formula is used to construct a series of invertible transformations, where the probability of the transformed variables can still be evaluated.

Let $\mathbf{Z}$ be a random variable defined over the support $S \subseteq \mathbb{R}^n$ and a continous probability density function $q(\mathbf{z})$. Further, let v be a continous, differentiable and invertible function from S to $T \in \mathbb{R}^n$, if we let $\mathbf{Z}' = v(\mathbf{Z})$, then the probability function $q(\mathbf{z}')$ of $\mathbf{Z}'$ is given by

$$\begin{aligned}
q(\mathbf{z}') &= q(\mathbf{z})\left|\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{z}'}\right)\right| \\
&= q(v^{-1}(\mathbf{z}'))\left|\det\left(\frac{\partial}{\partial \mathbf{z}'}v^{-1}(\mathbf{z}')\right)\right|
\end{aligned} \tag{7}$$

Normalizing flows takes advantage of this change of variables formula and create a family of parameterized invertible transformation V with an easy to compute jacobian. These transformations can be chained together to (usually) create an arbitrarily complex transformation.

$$\mathbf{z}_K = v_K \circ \cdots \circ v_2 \circ v_1(\mathbf{z}_0), \tag{8}$$

$$\log q(\mathbf{z}_K) = \log q(\mathbf{z}_0) - \sum_{k=1}^{K} \log\left|\det \frac{\partial v_k}{\partial \mathbf{z}_k}\right| \tag{9}$$

The *planar flow* is a normalizing flow defined by a series of transformations that expand and contract the distribution perpendicular to a hyperplane $\mathbf{w}^\top\mathbf{z} + b$. For a smooth element-wise non-linear function $h(\cdot)$, with derivative $h'(\cdot)$, the planar flow is defined as,

$$v(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}\mathbf{z} + b), \tag{10}$$

$$\det \left| \frac{\partial v}{\partial \mathbf{z}} \right| = \left| 1 + \mathbf{u}^\top h'(\mathbf{w}^\top \mathbf{z} + b)\mathbf{w} \right|, \tag{11}$$

where $\mathbf{u}$, $\mathbf{w}$, and b are free parameters. $h(z) = \tanh(z)$ is a common choice. For this choice of non-linearity, v is invertible if-and-only-if $\mathbf{w}^\top u \geq -1$. To satisfy this condition, Rezende and Mohamed (2015) suggest starting with an arbitrary vector $\mathbf{u}$ and modify its component parallel to $\mathbf{w}$, resulting in a new vector $\hat{\mathbf{u}}$ that satisfies $\mathbf{w}^\top \hat{\mathbf{u}} \leq -1$.

$$m(x) = -1 + \log\left(1 + e^x\right)$$

$$\hat{\mathbf{u}}(\mathbf{u}, \mathbf{w}) = \mathbf{u} + \left[m(\mathbf{w}^\top \mathbf{u}) - \mathbf{w}^\top \mathbf{u} \right] \frac{\mathbf{w}}{\|\mathbf{w}\|^2} \tag{12}$$

2.3 Conditional Normalizing Flows

Although normalizing flows is a good method for approximating distributions, they do not directly facilitate learning conditional likelihoods. Winkler et al. (2019) propose learning conditional distributions with *conditional normalizing flows*. Given a conditioning variable $\mathbf{x} \in U$, a random variale $\mathbf{Z} \in S \subseteq \mathbb{R}^n$, and a continous, differentiable function $v : S \times U \to T$, bijective in S and T, the probability function $q(\mathbf{z}')$, where $\mathbf{z}' = v(\mathbf{z}, \mathbf{x})$, can be expressed as

$$\begin{aligned} q(\mathbf{z}'|\mathbf{x}) &= q(\mathbf{z}|\mathbf{x}) \left| \det \left(\frac{\partial \mathbf{z}}{\partial \mathbf{z}'} \right) \right| \\ &= q(v^{-1}(\mathbf{z}', \mathbf{x})) \left| \det \left(\frac{\partial}{\partial \mathbf{z}'} v^{-1}(\mathbf{z}', \mathbf{x}) \right) \right|. \end{aligned} \tag{13}$$

Compared to Equation (7), all distributions are now conditional, and the model has a conditional argument $\mathbf{x}$. Winkler et al. (2019) suggest three conditional modules:

Conditional Prior

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu(\mathbf{x}), \sigma^2(\mathbf{x}))$$

Conditional Split Prior

$$p(\mathbf{z}_1|\mathbf{z}_0, \mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu(\mathbf{z}_0, \mathbf{x}), \sigma^2(\mathbf{z}_0, \mathbf{x}))$$

Conditional Coupling

$$\mathbf{z}'_0 = s(\mathbf{z}_1, \mathbf{x}) \cdot \mathbf{z}_0 + t(\mathbf{z}_1, \mathbf{x}); \qquad \mathbf{z}'_1 = \mathbf{z}_1$$

The functions μ, σ^2, s and t are implemented using neural networks.

3 Our Algorithm

We propose to use conditional normalizing flows to model the value distribution, Z^π, with a continous estimate. As opposed to previous work on quantile regression methods, which aim to minimize the Wasserstein distance to the target distribution, normalizing flows minimize the KL-divergence. Our target is thus the KL-divergence from the value to the target distribution.

$$\mathcal{L}_{\mathbf{x},a}(\theta) := D_{\mathrm{KL}} \left(\hat{\mathcal{T}} Z_{\tilde{\theta}}(\mathbf{x}, a) \middle\| Z_\theta(\mathbf{x}, a) \right) \tag{14}$$

Using conditional normalizing flows described in Section 2.3, we define the value distribution model as a transformation of a standard normal distribution conditioned on the state $\mathbf{x}$.

Starting with a random variable S with a standard normal distribution $\mathcal{N}(0, I_{|\mathcal{A}|})$, we want a function $v_\theta = v_{\theta_K} \circ \cdots \circ v_{\theta_2} \circ v_{\theta_1}$, such that $Z_\theta(\mathbf{x}, a) = v_\theta(S; \mathbf{x}')_a$. To learn v_θ we note that we can take the inverse transformation v_θ^{-1} of the value distribution to recover a random variable with the standard normal distribution, i.e. $S = v_\theta^{-1}(Z_\theta; \mathbf{x})$. This further means that

$$S' \overset{D}{=} v_\theta^{-1} \left(\mathcal{T} v_\theta(S; \mathbf{x}'); \mathbf{x} \right). \tag{15}$$

Because the inverse transformation of the value distribution given its state is standard normal, we can learn the parameters θ using the change-of-variables formula from Equation 7. Algorithm 1 shows how to calculate the loss for this scheme.

The most popular normalizing flows right now are buildt for large dimensional spaces such as images. This includes coupling and autoregressive flows. Planar flow is very easy to turn elementwise, but unfortunately does not have a closed form inverse for most functions of h. Although it does have one for piecewise linear funcitons, those do not have second derivative, hence the gradient of the log absolute determinant would be zero. This would mean it can not learn appropriate transformations. Instead we define our own flow.

Equation 16 shows the definitiona of the forward transformation $v(\cdot; b, c, d)$, with 3 free parameters b, c, d. Unlike the planar flow, this function has a closed form inverse and is differentiable given any values for these parameters, hence no reparameterization is necessary. Equation 18 and Equation 17 show the closed form inverse and gradient respectively.

$$v(z; b, c, d) = \operatorname{sgn}(z - c) \log \left(e^{|x-c|} + e^b - 1 \right) + d \tag{16}$$

Algorithm 1 Flow-RL Loss

Require: $N, v, v^{-1}, \theta, \tilde{\theta}$
input $\mathbf{x}, a, r, \mathbf{x}', \gamma$
  # Take N samples from the value distribution.
  for $i = 1 \dots N$
    $\epsilon_i \sim \mathcal{N}(0, 1)$
    for $a' \in \mathcal{A}$
      $Z_{\tilde{\theta}}(\mathbf{x}', a')_i \leftarrow v_{\tilde{\theta}}(\epsilon_i; \mathbf{x}')_{a'}$
    end
  end

  # Compute greedy next action.
  $a^* \leftarrow \arg\max_{a'} \frac{1}{N} \sum_{i=1}^{N} Z_{\tilde{\theta}}(\mathbf{x}', a')_i$

  # Calculate loss terms for each sample.
  for $i = 1 \dots N$
    # Apply the distributional Bellman operator.
    $\hat{Z}_{\tilde{\theta}}(\mathbf{x}, a)_i \leftarrow r + \gamma Z_{\tilde{\theta}}(\mathbf{x}', a^*)_i$
    # Calculate negative log-likelihood loss term.
    $\delta_i \leftarrow \frac{1}{2} v_\theta^{-1} \left(\hat{Z}_{\tilde{\theta}}(\mathbf{x}, a)_i; \mathbf{x} \right)^2$
    # Calculate log-derivative of v_θ^{-1}.
    $\rho_i \leftarrow \log \left| \frac{\partial v_\theta^{-1}\left(\hat{Z}_{\tilde{\theta}}(\mathbf{x}, a)_i; \mathbf{x}\right)}{\hat{Z}_{\tilde{\theta}}(\mathbf{x}, a)_i} \right|$
  end
output $\frac{1}{N} \sum_{i=1}^{N} \delta_i - \rho_i$

$$\frac{\partial v(z; b, c, d)}{\partial z} = \frac{e^{|x-c|}}{e^{|x-c|} + e^b - 1} \tag{17}$$

$$v^{-1}(z'; b, c, d) = v(z'; -b, d, c) \tag{18}$$

Figure 1 shows the effect of the free parameters. A simple description is that b changes the *linearity* of the function (v is linear when $b = 0$, an expansive mapping when $b < 0$ and a contraction when $b > 0$), c shifts v horizontally, and d shifts v vertically.

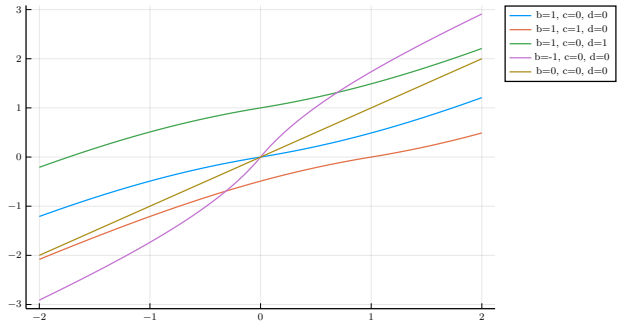

Figure 1: Illustration of the effect the free parameters have on the function.

References

Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 449–458. JMLR.org.

Bellman, R., Bellman, R., and Corporation, R. (1957). *Dynamic Programming*. Rand Corporation research study. Princeton University Press.

Puterman, M. L. (1994). Markov decision processes: Discrete stochastic dynamic programming. In *Wiley Series in Probability and Statistics*.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France. PMLR.

Watkins, C. J. C. H. and Dayan, P. (1989). Q-learning. *Machine Learning*, 8(3):279–292.

Winkler, C., Worrall, D. E., Hoogeboom, E., and Welling, M. (2019). Learning likelihoods with conditional normalizing flows.

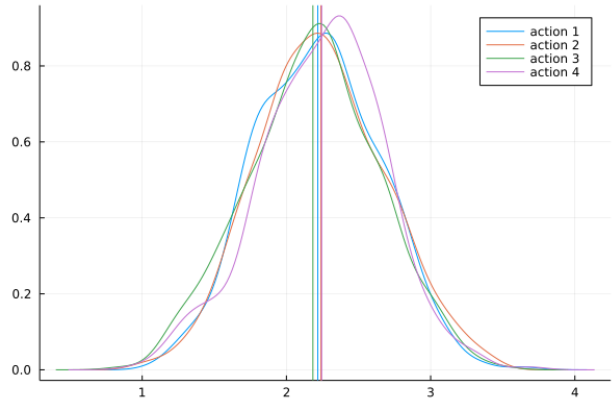

Figure 2: Learned Z^{π}-distribution during an episode in Breakout.