Randomized Prior Functions for Deep Reinforcement Learning

Ian Osband, John Aslanides, Albin Cassirer

Abstract

Dealing with uncertainty is essential for efficient RL. Many popular approaches for supervized learning are poorly-suited for RL. Others, such as bootstrapped ensembles, have no mechanism for 'prior' uncertainty.

We highlight this shortcoming and propose a **simple remedy: add a randomized untrainable 'prior' network to each member of ensemble.** We prove this approach is efficient with linear representations, provide simple illustrations of its efficacy with nonlinear representations and show that this approach scales to large problems.

Randomized prior function

Algorithm 1 Ensemble posterior with prior effect.

Require: Data $\mathcal{D} \subseteq \{(x,y) | x \in \mathcal{X}, y \in \mathcal{Y}\}$, loss function $\mathcal{L}$, neural model $f_\theta : \mathcal{X} \to \mathcal{Y}$, Ensemble size $K \in \mathbb{N}$, distribution over priors $\mathcal{P} \subseteq \{\mathbb{P}(p) | p : \mathcal{X} \to \mathcal{Y}\}$.

```
for k = 1,..,K do
    initialize θ_k ~ Glorot initialization.
    form D_k = data_noise(D) (e.g. bootstrap).
    sample prior function p_k ~ P.
    optimize ∇_{θ|θ=θ_k} L(f_θ + p_k; D_k) via ADAM.
return posterior ensemble {f_{θ_k} + p_k}_{k=1}^K.
```

For deep RL, we apply Algorithm 1 to DQN, with TD loss $\mathcal{L}_\gamma(\theta; \theta^-, p, \mathcal{D}) :=$

$$\sum_{t \in \mathcal{D}} \Big(r_t + \gamma \max_{a' \in \mathcal{A}} \overbrace{(f_{\theta^-} + \underbrace{p}_{\text{prior}})}^{\text{target } Q}(s'_t, a') - \overbrace{(f_\theta + \underbrace{p}_{\text{prior}})}^{\text{online } Q}(s_t, a_t) \Big)^2.$$

Algorithm 2 `learn_bootdqn_with_prior`

Agent:	$\theta_1, .., \theta_K$	trainable weights
	$p_1, .., p_K$	fixed prior functions
	$\mathcal{L}_\gamma(\theta = \cdot; \theta^- = \cdot, p = \cdot, \mathcal{D} = \cdot)$	TD error loss
	`ensemble_replay`	perturbed data
Updates:	$\theta_1, .., \theta_K$	agent weights

```
for k in (1,...,K) do
    Data D_k ← ensemble_replay[k].sample()
    optimize ∇_{θ|θ=θ_k} L(θ; θ_k, p_k, D_k) via ADAM.
```

- Ensemble $\{Q_{\theta_k}\}_{k=1}^K$ approximates posterior.
- Each episode: sample $j \sim \mathrm{Unif}(1,..,K)$ and follow Q_{θ_j} greedy policy. $\simeq$ **Thompson sampling.**
- *'Deep exploration via randomized value functions'.*

Bayesian linear regression

Let $\theta \in \mathbb{R}^d$, prior $N(\overline{\theta}, \lambda I)$ and data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ for $y_i = \theta^T x_i + \epsilon_i$ with $\epsilon_i \sim N(0, \sigma^2)$ iid. Then, conditioned on $\mathcal{D}$, the posterior for θ is Gaussian:

$$\mathbb{E}[\theta | \mathcal{D}] = \left(\frac{1}{\sigma^2} X^T X + \frac{1}{\lambda} I \right)^{-1} \left(\frac{1}{\sigma^2} X^T y + \frac{1}{\lambda} \overline{\theta} \right),$$
$$\mathrm{Cov}[\theta | \mathcal{D}] = \left(\frac{1}{\sigma^2} X^T X + \frac{1}{\lambda} I \right)^{-1}. \qquad (1)$$

Equation (1) relies on Gaussian conjugacy and linear models, which cannot easily be extended to deep neural networks. **Lemma 1 shows that our approach: 'train on noisy data with random prior functions' is Bayes posterior for linear f_θ.**

Lemma 1 (Computational posterior samples).
Let $f_\theta(x) = x^T \theta$, $\tilde{y}_i \sim N(y_i, \sigma^2)$ and $\tilde{\theta} \sim N(\overline{\theta}, \lambda I)$. Then either of the following optimization problems generate a sample $\theta \mid \mathcal{D}$ according to (1):

$$\operatorname*{argmin}_\theta \sum_{i=1}^n \| \tilde{y}_i - f_\theta(x_i) \|_2^2 + \frac{\sigma^2}{\lambda} \| \tilde{\theta} - \theta \|_2^2, \qquad (2)$$

$$\tilde{\theta} + \operatorname*{argmin}_\theta \sum_{i=1}^n \| \tilde{y}_i - (f_{\tilde{\theta}} + f_\theta)(x_i) \|_2^2 + \frac{\sigma^2}{\lambda} \| \theta \|_2^2. \qquad (3)$$

Proof. Note output is Gaussian, match moments. □

Visualizing prior effect

- Training data **black**.
- Prior $p(x)$ in blue.
- Train $f_\theta(x)$ dashed.
- Predict $(f_\theta + p)(x)$ red

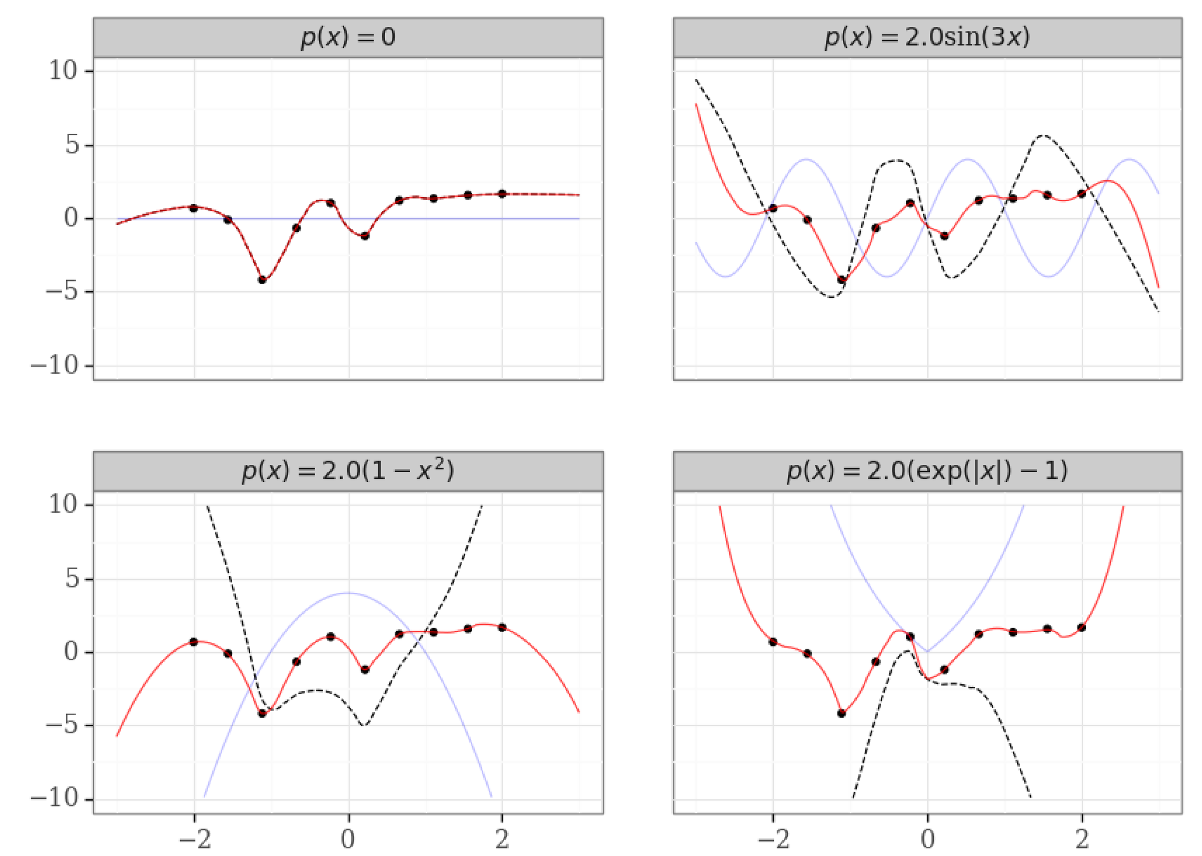

Figure 1: Visualizing output in 1D regression.

- All networks can optimize to fit observed data.
- Bootstrap (data noise) handles noisy data.
- Prior dominates outside range of data.
- Resultant **ensemble** $\{f_{\theta_k} + p_k\}_{k=1}^K \simeq$ **posterior.**

Why do we need this?

Popular approaches have serious shortcomings!

1. Dropout as posterior approximation
- Dropout does not concentrate with data.
- Even 'concrete' not necessarily correct rate.

2. Variational inference on Bellman error
- VI on Bellman error $\neq$ VI on value.
- If you train $Q_\theta(s,a) \simeq^D r + \gamma \max_\alpha Q_\theta(s', \alpha)$ must note $Q_\theta(s,a), Q_\theta(s', \alpha)$ are not indep.

3. Distributional reinforcement learning
- Distribution outcome vs. posterior of beliefs.
- 'Aleatoric' vs 'epistemic' uncertainty.

4. Count-based exploration bonus
- Density metric is not connected to task.
- With generalization 'count' $\neq$ 'uncertainty'.

For more detail see Section 2 of the paper.

Driving deep exploration

Scalable 'chain' environments test exploration.

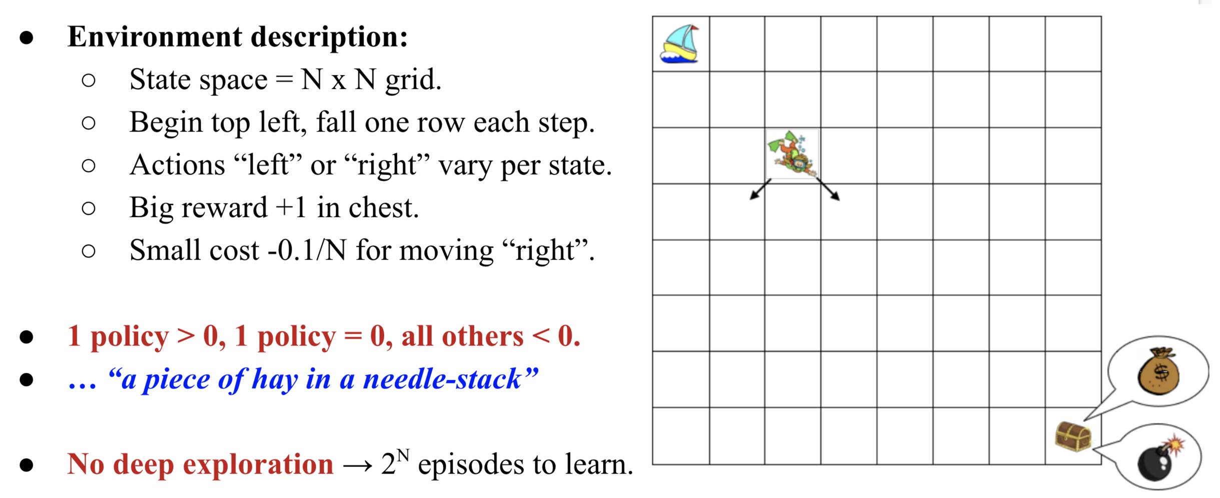

Figure 2: Describing 'deep sea' chain environments.

'Time to learn' := #episodes until AveRegret < 0.9.

- ϵ-greedy = DQN with annealing dithering.
- BS = BootDQN without explicit prior.
- BSR = BootDQN with regularize $\|\theta_k - \theta_k^{\text{init}}\|$.
- BSP = BootDQN with prior, $Q_k = f_{\theta_k} + p_k$.

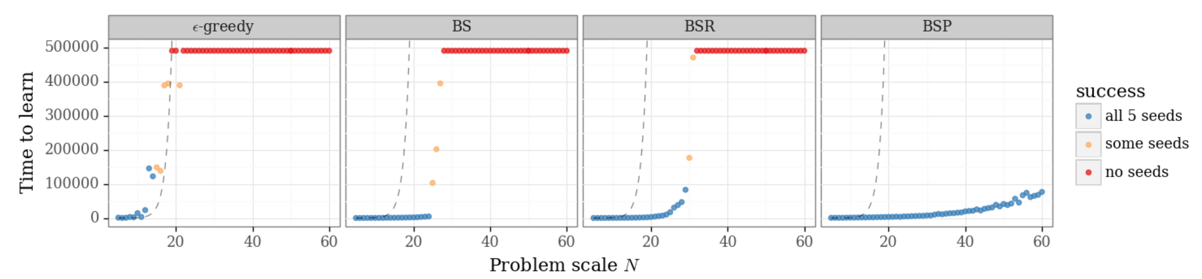

Figure 3: Only BSP scales to large problems. Plotting log-log suggests an empirical scaling $T_{\text{learn}} = \tilde{O}(N^3)$.

How does it work?

- ✓ **Posterior concentration**: prior p_k motivates uncertainty, but f_θ eventually learns to fit it away.
- ✓ **Multi-step uncertainty**: Each Q_k trains only on its *own* target value $\implies$ temporally-consistent.
- ✓ **Epistemic vs aleatoric**: Uncertainty in the *mean* TD loss and does not fit the noise in returns.
- ✓ **Task-appropriate generalization**: Explore by uncertainty in Q, rather than density on state.
- ✓ **Intrinsic motivation** (vs BootDQN no prior): Sparse rewards $\implies$ bootstrap may predict zero for *all* states. Prior p_k makes this unlikely at rarely-seen states $\tilde{s}$ where $\mathbb{E}[\max_\alpha Q_k(\tilde{s}, \alpha)] > 0$.

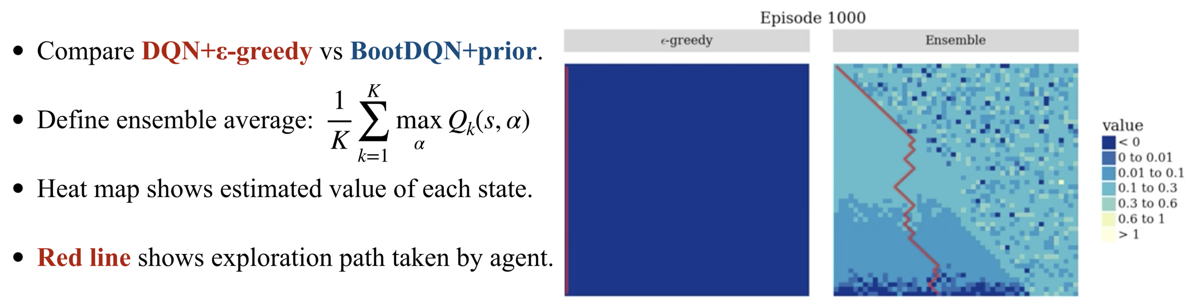

Figure 4: Visualizing how BootDQN+prior explores.

Scaling up

Insights carry over to large-scale 'deep RL'.

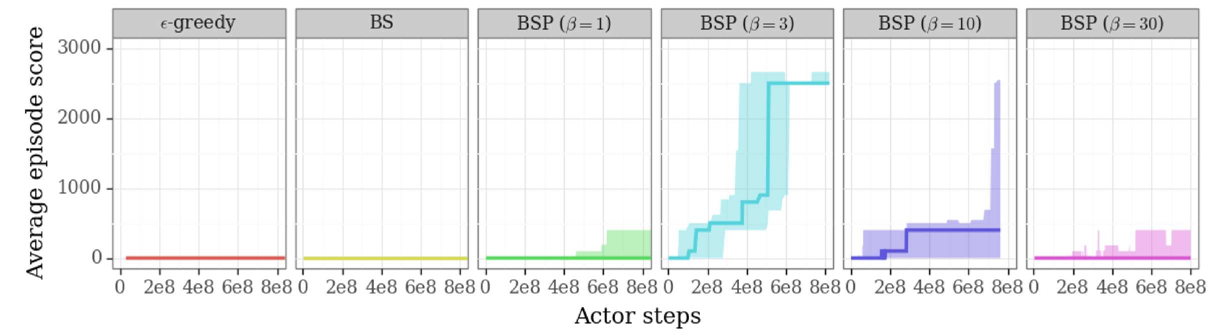

Figure 5: The prior scaling $Q_{\theta_k} = f_{\theta_k} + \beta p_k$ qualitatively changes behavior on Montezuma's revenge.

So what?

1. **Highlight need for prior effect in deep RL.**
2. **Random prior passes linear 'sanity check'.**
3. **Show scalable deep RL in toy problem.**
4. **Insights carry over to Montezuma's revenge.**

More information

Paper site (+ code): `bit.ly/rpf_nips`
Tweet: `@ianosband, @john_aslanides`
Personal site: `iosband.github.io`