**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 supervised 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 this approach scales to large problems.

**Bayesian Linear Regression**

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

\[
E[\theta | D] = \left( \frac{1}{\sigma^2} \mathbf{X}^T \mathbf{X} + \frac{1}{\lambda} \mathbf{I} \right)^{-1} \left( \frac{1}{\sigma^2} \mathbf{X}^T y + \frac{\lambda}{\lambda} \mathbf{I} \right).
\]

Con[\( \theta | D \)] = \left( \frac{1}{\sigma^2} \mathbf{X}^T \mathbf{X} + \frac{1}{\lambda} \mathbf{I} \right)^{-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 \( f_s \).

**Why do we need this?**

Popular approaches have serious shortcomings!

1. **Dropout as prior 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 \( Q(s,a) \rightleftharpoons r + \gamma \max \alpha Q(s',\alpha) \) must note \( Q(s,a) | Q(s',\alpha) \) 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.

**Randomized Prior Function**

Algorithm 1: Ensemble posterior with prior effect.

**Require:** Data \( D = \{(x,y) \in \mathbb{R}^d \times \mathbb{R} \} \), loss function \( \mathcal{L} \), neural model \( f_s(X) \rightarrow \mathbb{R} \). Ensemble size \( K \in \mathbb{N} \), distribution over priors \( P(\theta | p : X \rightarrow \mathbb{R}) \).

1. for \( k = 1 \ldots K \) do
2. \( \theta_k \sim \) Glorot initialization.
3. \( D_k = \) data_noise(\( D \)) (e.g. bootstrap).
4. \( \theta_k \sim \) prior function \( p_k \). ~ \( P(p : X \rightarrow \mathbb{R}) \).
5. \( \text{optimize } \mathcal{L}(\theta_k ; p_k ; D_k) \) via ADAM.
6. \( \text{return posterior ensemble } \{f_{s,\theta_k} + p_k \}_{k=1}^K \).

For deep RL, we apply Algorithm 1 to DQN, with TD loss \( \mathcal{L}(\theta, \theta - \varphi, p, \gamma) \).:

\[
\sum_{(s,a) \in D} \left( r + \gamma \max_{a'} f_{s,a'}(s',a') - f_{s,a}(s,a) \right)^2.
\]

**Driving Deep Exploration**

Scalable ‘chain’ environments test exploration.

- **Environment description:**
  - State-space: \( s \times s \) grid.
  - Begin top left, all-state row each step.
  - Actions ‘left’ or ‘right’ vary per step.
  - Big reward +1 in chest.
  - Small cost -0.1 for moving ‘right’.

- **1 policy + 6 policies + 6 all others + 6**
- **‘a priori of key in a needlestick’**
- **No deep exploration \( \rightarrow 2 \) episodes to learn**

**Visualizing Prior Effect**

- **Training data black.**
- **Prior \( p(\theta) \) in blue.**
- **Train \( f_s(x) \) dashed.**
- **Predict \( (f_s + p(\theta)) \) 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 \( \{f_{s,\theta_k} + p_k \}_{k=1}^K \approx \text{posterior} \).**

**Figure 2:** Describing ‘deep sea’ chain environments.

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

**How does it work?**

- **Posterior concentration:** prior \( p_k \) motivates uncertainty, but \( f_s \) eventually learns to fit it away.
- **Multi-step uncertainty:** Each \( Q_k \) trains only on its own target value \( \rightarrow \) 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 \( \Rightarrow \) bootstrap may predict zero for all states. Prior \( p_k \) makes this unlikely at rarely-seen states \( s \) where \( E[\max \alpha Q(s,\alpha)] > 0 \).

**Scaling Up**

Insights carry over to large-scale ‘deep RL’.

- **Figure 5:** The prior scaling \( Q_{p_k} = f_{s,\theta_k} + p_k \) 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: @iosband, @john_aslanides
Personal site: iosband.github.io