bit.ly/rpf_nips
@ianosband

# Randomized Prior Functions for Deep Reinforcement Learning

Ian Osband, John Aslanides, Albin Cassirer

bit.ly/rpf_nips
@ianosband

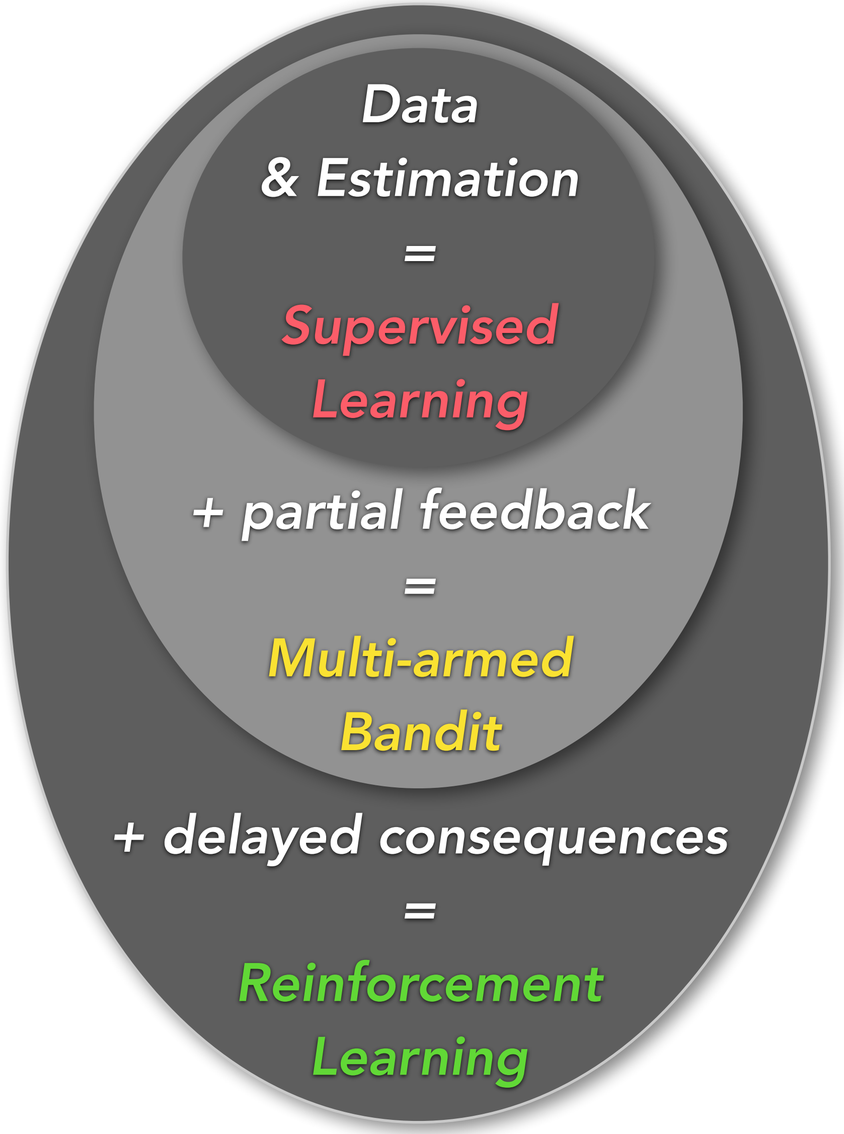
# Reinforcement Learning

- *"Sequential decision making under uncertainty."*

- Three necessary building blocks:
  1. **Generalization**
  2. **Exploration vs Exploitation**
  3. **Credit assignment**

- As a field, we are pretty good at combining any 2 of these 3.
  … but we need practical solutions that combine them all.

**We need effective uncertainty estimates for Deep RL**



bit.ly/rpf_nips
@ianosband

# Estimating uncertainty in deep RL

| Dropout sampling | Variational inference | Distributional RL | Count-based density | Bootstrap ensemble |
|---|---|---|---|---|
| *"Dropout sample ≈ posterior sample" (Gal+Gharamani 2015)* | *Apply VI to Bellman error as if it was an i.i.d. supervised loss.* | *Models Q-value as a distribution, rather than point estimate.* | *Estimate number of "visit counts" to state, add bonus.* | *Train ensemble on noisy data - classic statistical procedure!* |
| Dropout rate does not concentrate with the data. | Bellman error: $Q(s,a) = r + \gamma \max_{\alpha} Q(s',\alpha)$<br>Uncertainty in Q ➡ correlated TD loss.<br>VI on i.i.d. model does not propagate uncertainty. | This distribution != posterior uncertainty. | The "density model" has nothing to do with the actual task. | No explicit "prior" mechanism for "intrinsic motivation" |
| Even "concrete" dropout not necessarily right rate. | | *Aleatoric* vs *Epistemic* … it's *not* the right thing for exploration. | With generalization, state "visit count" != uncertainty. | If you've never seen a reward, why would the agent explore? |

# Randomized prior functions

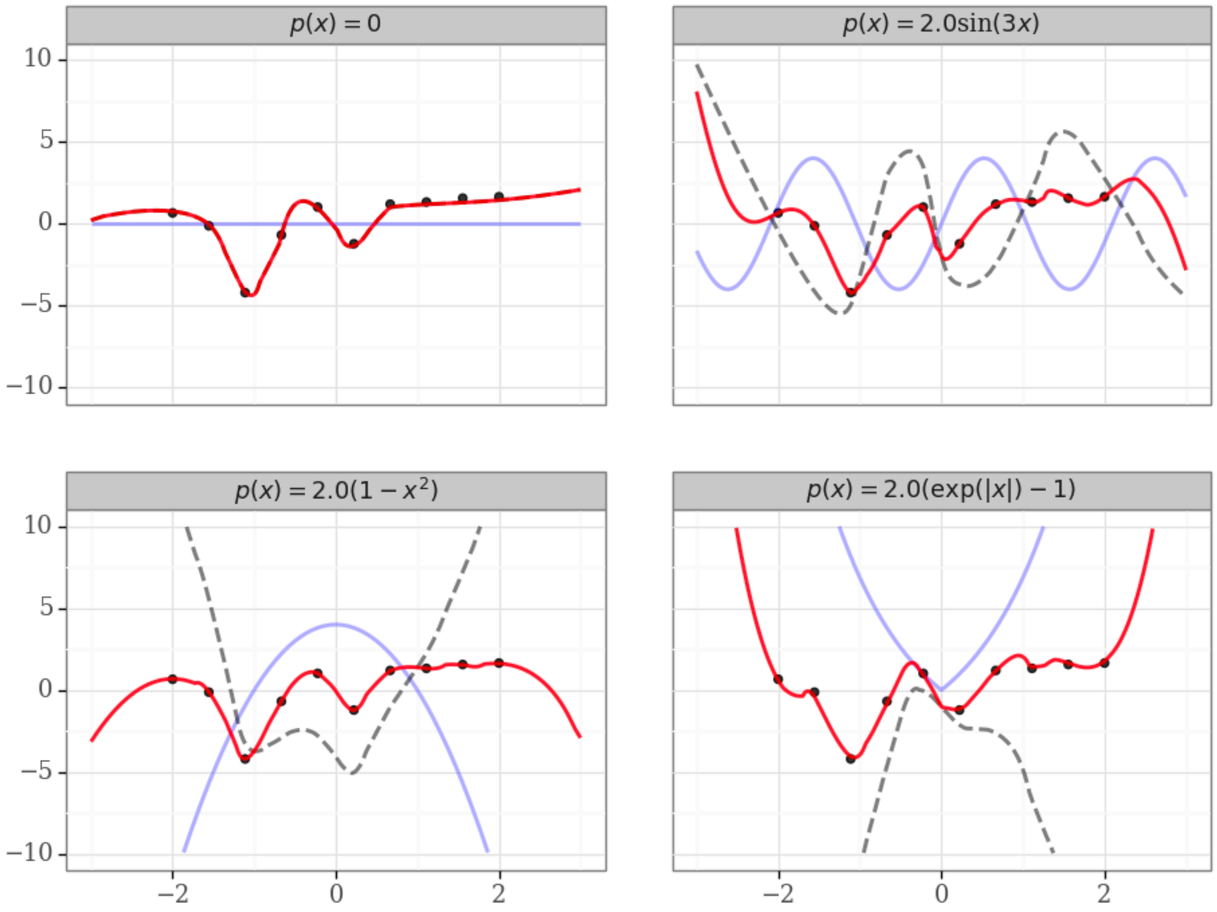
bit.ly/rpf_nips
@ianosband

- **Key idea:** add a random untrainable "prior" function to each member of the ensemble.

$$Q_\theta(x) = \underbrace{f_\theta(x)}_{\text{trainable}} + \underbrace{p(x)}_{\text{prior}}.$$

- Visualize effects in 1D regression:
  - Training data **(x,y) black points.**
  - Prior function **p(x) blue line.**
  - Trainable function f(x) dotted line.
  - Output prediction **Q(x) red line.**

**Exact Bayes posterior for linear functions!**



# "Deep Sea" Exploration

bit.ly/rpf_nips
@ianosband

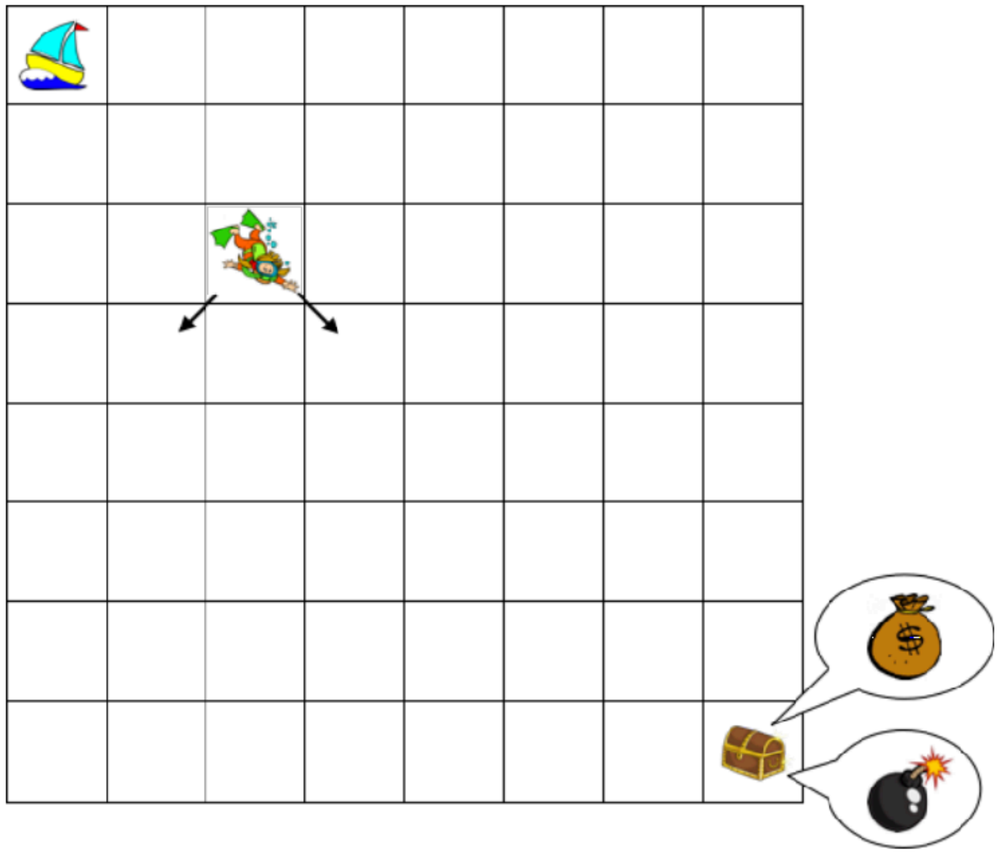
- **Stylized "chain" domain testing "deep exploration":**
  - *State = N x N grid, observations 1-hot.*
  - *Start in top left cell, fall one row each step.*
  - *Actions {0,1} map to left/right in each cell.*
  - ***"left" has reward = 0, "right" has reward = -0.1/N***
  - ***… but if you make it to bottom right you get +1.***

- Only one policy (out of more than $2^N$) has positive return.

- **ε-greedy / Boltzmann / policy gradient / are useless.**

- **Algorithms with deep exploration can learn fast!**

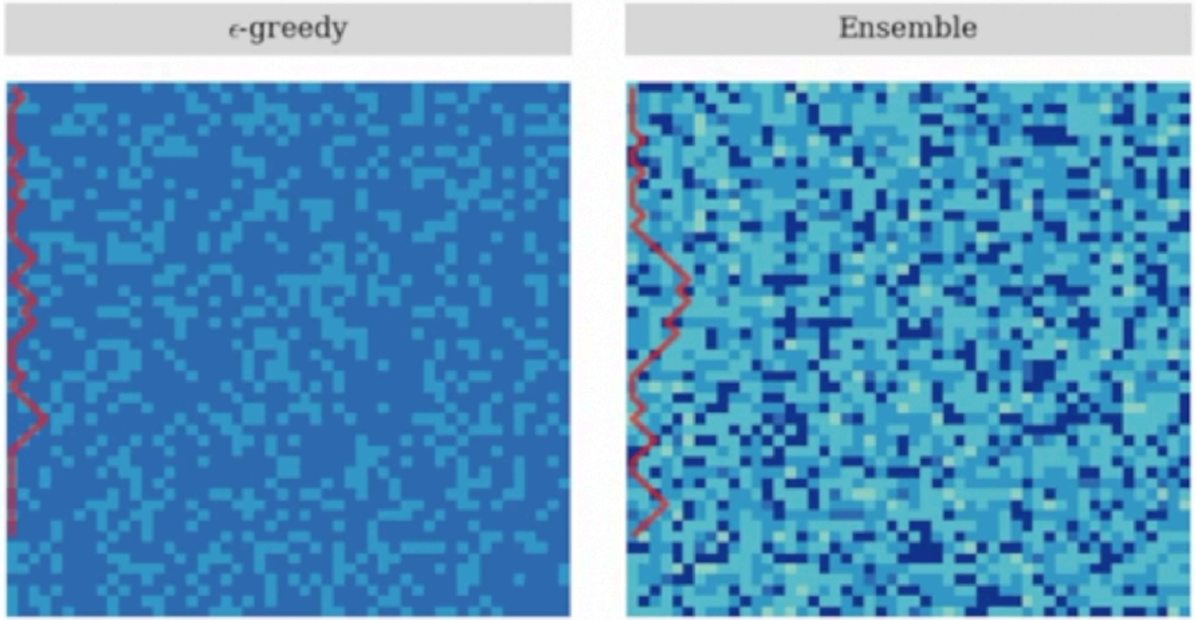
*[1] "Deep Exploration via Randomized Value Functions"*



# Visualize BootDQN+prior exploration.

bit.ly/rpf_nips
@ianosband

- Compare **DQN+ε-greedy** vs **BootDQN+prior**.
- Define ensemble average: $\frac{1}{K}\sum_{k=1}^{K}\max_{\alpha} Q_k(s,\alpha)$
- Heat map shows estimated value of each state.
- **Red line** shows exploration path taken by agent.
- **DQN+ε-greedy gets stuck on the left, gives up.**
- **BootDQN+prior hopes something is out there, keeps exploring potentially-rewarding states… learns fast!**



bit.ly/rpf_nips
@ianosband

# VIDEO TO COVER THIS WHOLE SLIDE

DeepMind

bit.ly/rpf_nips
@ianosband

# Come visit our poster!

Ian Osband, John Aslanides, Albin Cassirer

**Blog post**
**bit.ly/rpf_nips**

**Montezuma's Revenge!**

**Demo code**
**bit.ly/rpf_nips**