INFOBOT: TRANSFER AND EXPLORATION VIA THE INFORMATION BOTTLENECK

Anirudh Goyal, Riashat Islam, Daniel Strouse et al
Presented by Martin Liivak
Glossary

**Agent** - autonomous or semi-autonomous system that uses deep learning to perform and improve at its tasks

**Policy** - state to action mapping

**Regularization** - a technique used in an attempt to solve overfitting

**Reinforcement learning (RL)** - agents learn how to take actions in an environment so as to maximize some cumulative reward

**Deep learning** - a machine learning technique that teaches agents to learn by example
Background

- Deep RL is successful in domains where large amounts of training time and a dense reward function are provided.
- Environments with sparse rewards remain a major challenge.
- Providing agents with useful signals to pursue instead of environmental reward becomes crucial in these scenarios.
Main goal

Incentivizing agents to learn about and exploit multi-goal task structure in order to efficiently explore in new environments.
Goal-conditioned policy

- Focus is on multi-goal environments and goal-conditioned policies.
- Goal G is sampled and gives agent information about environment’s reward structure.
- Agent’s policy is denoted as

\[ \pi_\theta(A | S, G) \]

Where S is the agent’s state, A the agent’s action, and \( \theta \) the policy parameters.
Information bottleneck

- Agents are made to learn a task structure by training policies that perform well under a variety of goals, while not overfitting to any individual goal.
- Achieved by minimizing policy dependence on the individual goal, quantified by conditional mutual information.
- Agents are encouraged to follow some goal agnostic default policy.
Regularizer

- Conditional mutual information is used as the regularizer for RL agents.

\[
I(A; G \mid S) = \mathbb{E}_{\pi_\theta} \left[ D_{\text{KL}}[\pi_\theta(A \mid S, G) \mid \pi_0(A \mid S)] \right]
\]
Regularizer

\[ \mathbb{E}_{\pi_\theta} \left[ D_{KL} \left[ \pi_\theta (A \mid S, G) \mid \pi_0 (A \mid S) \right] \right] \]

Default policy with the goal marginalized out
Regularizer

$\mathbb{E}_{\pi_\theta} \left[ D_{KL}(\pi_\theta(A | S, G) || \pi_0(A | S)) \right]$
Regularizer

Kullback-Leibler divergence between policy distributions
Regularizer

$$\mathbb{E}_{\pi^\theta} \left[ D_{KL} [\pi^\theta(A \mid S, G) \mid \pi_0(A \mid S)] \right]$$
Decision state

- States where diversions from default behaviour occur are called decision states.
- They are natural subgoals for efficient exploration.
Training approach

- Visiting decision states is encouraged by first training an agent with the information regularizer to recognize decision states.
- Agent’s policy is then frozen and KL divergence is used as an exploration bonus for training a new policy.
- Bonus for continued visits is decayed.
- This is tuned to the family of tasks the agent is trained on.
for \( \text{episodes} = 1 \) to \( N_{\text{train}} \) do
  Sample a task \( T \sim p_{\text{train}}(T) \) and goal \( G \sim p(G \mid T) \)
  Produce trajectory \( \tau \) on task \( T \) with goal \( G \) using policy \( \pi_\theta(A \mid S, G) \)
  Update policy parameters \( \theta \) over \( \tau \) using Eqn 5
end for

**Optional:** use \( \pi_\theta \) directly on tasks sampled from \( p_{\text{test}}(T) \)
for \( \text{episodes} = 1 \) to \( N_{\text{test}} \) do
  Sample a task \( T \sim p_{\text{test}}(T) \) and goal \( G \sim p(G \mid T) \)
  Produce trajectory \( \tau \) on task \( T \) with goal \( G \) using policy \( \pi_\phi(A \mid S, G) \)
  Update policy parameters \( \phi \) using algorithm \( \mathcal{A} \) to maximize the reward given by Eqn 6
end for

- Eqn 5 - using conditional mutual information regularizer when updating objective function values
- Eqn 6 - decaying the exploration bonus with visit count
Experimental results

- The goal-conditioned policy with information bottleneck leads to much better policy transfer than standard RL training procedures.
- Using decision states as an exploration bonus leads to better performance than a variety of standard task-agnostic exploration methods.
Minigrid environments

(a) MultiRoomN4S4  (b) MultiRoomN12S10  (c) FindObjS5  (d) FindObjS6
Policy generalization on MultiRoomNXSY
Policy generalization on MultiRoomNXSY

![Graph showing success rate over timesteps for InfoBot and Baseline, trained on N2S6, evaluated on N10S10.](image-url)
Policy generalization on MultiRoomNXSY

Trained on N2S6, Evaluation on N12S10

Success Rate (%) vs. Timesteps

InfoBot vs. Baseline
MultiRoomNXSY strategy

(a) MultiRoomN4S4
(b) MultiRoomN12S10
Policy generalization on FindObjSY

<table>
<thead>
<tr>
<th>Method</th>
<th>FindObjS7</th>
<th>FindObjS10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal-conditioned A2C</td>
<td>56%</td>
<td>36%</td>
</tr>
<tr>
<td>InfoBot with $\beta = 0$</td>
<td>44%</td>
<td>24%</td>
</tr>
<tr>
<td>InfoBot</td>
<td>81%</td>
<td>61%</td>
</tr>
</tbody>
</table>
FindObjSY strategy

(c) FindObjS5

(d) FindObjS6
Transferable exploration strategies on MultiRoomNXSY

<table>
<thead>
<tr>
<th>Method</th>
<th>MultiRoomN3S4</th>
<th>MultiRoomN5S4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal-conditioned A2C</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>TRPO + VIME</td>
<td>54%</td>
<td>0%</td>
</tr>
<tr>
<td>Count based exploration</td>
<td>95%</td>
<td>0%</td>
</tr>
<tr>
<td>Curiosity-based exploration</td>
<td>95%</td>
<td>54%</td>
</tr>
<tr>
<td>InfoBot (decision state exploration bonus)</td>
<td>90%</td>
<td>85%</td>
</tr>
</tbody>
</table>
Transferable exploration strategies for continuous control

To show that the InfoBot architecture can also be applied to continuous control, the performance of InfoBot was evaluated on three continuous control tasks from OpenAI Gym.
Humanoid results
Walker2D results

![Graph showing Walker2D results with multiple lines representing different algorithms: Baseline, InfoBot, Infobot-low-value, and Infobot-zero-kl. The x-axis represents iterations, and the y-axis represents average reward.](image-url)
Transferable exploration strategies for Atari

InfoBot framework is further evaluated on a few Atari games against A2C. The goal is to show that InfoBot can generalize to even more complex domains compared to the maze tasks above.
Pong and Qbert results

![Pong Results](image1)

![Qbert Results](image2)
Seaquest and Breakout results

Seaquest

BreakOut

Average Task Return

Timesteps

Timesteps

InfoBot
Baseline

InfoBot
Baseline
Goal-based navigation tasks
Goal-based navigation task results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Evaluate on 11 × 11 maze</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor-Critic</td>
<td>5%</td>
</tr>
<tr>
<td>PPO (Proximal Policy Optimization)</td>
<td>8%</td>
</tr>
<tr>
<td>Actor-Critic + Count-Based</td>
<td>7%</td>
</tr>
<tr>
<td>Curiosity Driven Learning (ICM)</td>
<td>47%</td>
</tr>
<tr>
<td>Goal Based (UVFA) Goal - TopDownImage of the goal</td>
<td>7%</td>
</tr>
<tr>
<td>Goal Based (UVFA) Goal - Relative Dist</td>
<td>15%</td>
</tr>
<tr>
<td>Feudal RL</td>
<td>37%</td>
</tr>
<tr>
<td>InfoBot (proposed)</td>
<td>64%</td>
</tr>
</tbody>
</table>
Conclusion

- Proposed to train agents to develop default behaviours and the knowledge of when to break them using information bottleneck.
- Demonstrated that this training procedure leads to better direct policy transfer across tasks.
- Demonstrated that the decision states can be used as the basis for a learned exploration bonus that leads to more effective training than other task-agnostic exploration methods.
Thanks for listening