Distral: Robust Multitask Reinforcement Learning

Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, Razvan Pascanu

Google DeepMind, London, UK

Arxiv / Presenter: Ji Gao
Outline

1 Motivation

2 Method
   - Overview
   - Background

3 Experiment
Motivation

- Deep Reinforcement Learning is successful: DQN play ATARI games at above human performance.
- However:
  - Very costly to train: Need a lot of time and data
  - The training process is unstable. Many tricks introduce to make the process stable.
- People often think in the multitask learning, a task will require less data and achieve a higher asymptotic performance (i.e., other tasks helps). However, it’s often not true in the real case.
- It is likely that gradients from other tasks behave as noise, interfering with learning, or, in another extreme, one of the tasks might dominate the others.
Motivation: Find an efficient and stable way for multitask RL
Baseline: Multitask A3C algorithm
Learn a global policy $\pi_0$
Use a KL divergence regularizer to produce task-specific policies
Knowledge learned on one task are distilled into the shared policy, and transferred to other tasks.
### Background: Multitask RL

#### Notations of Multitask RL

A multitask RL setting:
- $n$ tasks
- an infinite horizon with discount factor $\gamma$
- $a \in A$ are actions, $s \in S$ are states.
- Transition function $p_i(s'|s, a)$ and reward $R_i(a, s)$ is different for each task $i$.
- $\pi_i$: task specific stochastic policies

We want to optimize the expectation of reward, that is

$$\sum_i \mathbb{E}_{\pi_i} \left[ \sum_{t \geq 0} \gamma^t R_i(a_t, s_t) \right]$$
Distral

- Learn a global policy $\pi_0$
- KL regularization: Ensure every task policy share information with the shared policy:
  $$E_{\pi_i} \left[ \sum_{t \geq 0} \gamma^t \log \frac{\pi_i(a_t|s_t)}{\pi_0(a_t|s_t)} \right]$$
- Additional entropy regularization to encourage exploration:
  $$E_{\pi_i} \left[ \gamma^t \log \pi_i(a_t|s_t) \right]$$
- Total loss function:
  $$J(\pi_0, \{\pi_i\}_{i=0}^n) = \sum_i E_{\pi_i} \left[ \sum_{t \geq 0} \gamma^t R_i(a_t, s_t) - cKL \gamma^t \log \frac{\pi_i(a_t|s_t)}{\pi_0(a_t|s_t)} - cEnt \gamma^t \log \pi_i(a_t|s_t) \right]$$
  $$= \sum_i E_{\pi_i} \left[ \sum_{t \geq 0} \gamma^t (R_i(a_t, s_t) + \frac{\alpha}{\beta} \log \pi_0(a_t|s_t) - \frac{1}{\beta} \log \pi_i(a_t|s_t)) \right]$$
  $$\alpha = \frac{cKL}{cKL + cEnt}, \ \beta = \frac{1}{cKL + cEnt}$$
$J(\pi_0, \{\pi_i\}_{i=0}^n) = \sum_i \mathbb{E}_{\pi_i} \left[ \sum_{t \geq 0} \gamma^t (R_i(a_t, s_t) + \frac{\alpha}{\beta} \log \pi_0(a_t|s_t) - \frac{1}{\beta} \log \pi_i(a_t|s_t)) \right]$

- Let $R'_i(a, s) = R_i(a_t, s_t) + \frac{\alpha}{\beta} \log \pi_0(a_t|s_t)$, the previous objective function becomes a regularized optimization on a new reward.
- Entropy regularization with a new constant.
Soft Q-Learning

- Bellman equation

\[ V_i(s_t) = \max_{a_t} Q_i(a_t, s_t) \]

\[ Q_i(a_t, s_t) = R_i(a_t, s_t) + \gamma \sum_{s_{t+1}} p_i(s_{t+1}|s_t, a_t) V_i(s_{t+1}) \]

- Soft Q-Learning [Derivation included in *Equivalence Between Policy Gradients and Soft Q-Learning*]:

\[ V_i(s_t) = \frac{1}{\beta} \log(\sum_{a_t} \pi_0(a_t|s_t)^\alpha \exp[\beta Q_i(a_t, s_t)]) \]

\[ Q_i(a_t, s_t) = R_i(a_t, s_t) + \gamma \sum_{s_{t+1}} p_i(s_{t+1}|s_t, a_t) V_i(s_{t+1}) \]

- The idea is to use a Softmax function at the inverse temperature \( \beta \) to approximate the Bellman equation. When \( \beta \to \infty \), Softmax turns into Max.
The optimal policy [from *Equivalence Between Policy Gradients and Soft Q-Learning*]

\[
\pi_i(a_t, s_t) = \arg \max_{\pi_i} \mathbb{E}_\pi [Q_i(s_t, a_t)] - \frac{\alpha}{\beta} \log \frac{\pi_i(a_t | s_t)}{\pi_0(a_t | s_t)}
\]

\[
= \pi_0^\alpha (a_t | s_t) e^{\beta Q_i(s_t, a_t)} - \beta V_i(s_t)
\]

\[
= \pi_0^\alpha (a_t | s_t) e^{\beta A_i(s_t, a_t)}
\]

*Ai* here is an advantage function.

- Unlike previous literature, \(\pi_0\) is learned.
- \(\pi_0\) is updated using the loss \(\frac{\alpha}{\beta} \sum_i \mathbb{E}[\frac{\alpha}{\beta} \log \pi_0(a_t | s_t)]\)
Parameterization

- It is possible to simply assign each $\pi_i$ a neural network and $\pi_0$ a neural network, and update them using policy gradient.
- However, maximize $\pi_i$ and $\pi_0$ alternatively could be slow.
- Need a better way for a joint optimization
- Method: 1. Parameterize $\pi_0$ by

$$\hat{\pi}_0 = \frac{\exp(h_{\theta_0}(a_t|s_t))}{\sum_{a'} \exp(h_{\theta_0}(a'|s_t))}$$

2. Parameterize $A_i = Q_i - V_i$ by

$$\hat{A}_i(a_t|s_t) = f_{\theta_i}(a_t|s_t) - \frac{1}{\beta} \log \sum_a \hat{\pi}_0^\alpha(a|s_t) \exp(\beta f_{\theta_i}(a|s_t))$$

In the case, the policy for task $i$ becomes

$$\hat{\pi}_i(a_t|s_t) = \frac{\exp(\alpha h_{\theta_0}(a_t|s_t) + \beta f_{\theta_i}(a_t|s_t))}{\sum_{a'} \exp(\alpha h_{\theta_0}(a'|s_t) + \beta f_{\theta_i}(a'|s_t))}$$
Figure 2: Depiction of the different algorithms and baselines. On the left are two of the Distral algorithms and on the right are the three A3C baselines. Entropy is drawn in brackets as it is optional and only used for KL+ent 2col and KL+ent 1col.
Comparing 7 algorithms

Entropy regularization vs No entropy regularization

1col(Alternative optimization) vs 2col(joint optimization)
**Small example: Two room grid world**

**Figure 3:** **Left:** Learning curves on two room grid world. The DisTraL agent (blue) learns faster, converges towards better policies, and demonstrates more stable learning overall. **Center:** Example of tasks. Green is goal position which is uniformly sampled for each task. Starting position is uniformly sampled at the beginning of each episode. **Right:** depiction of learned distilled policy $\pi_0$ only in the corridor, conditioned on previous action being left/right and no previous reward. Sizes of arrows depict probabilities of actions. Note that up/down actions have negligible probabilities. The model learns to preserve direction of travel in the corridor.

- Distral converge faster than single task case
- Distral successfully learn a distilled policy in the corridor that conditioned on the previous move
Three complex partially observable 3D case from Deepmind Lab: Mazes, Navigation and Laser-Tag

Figure 4: Panels A1, C1, D1 show task specific policy performance (averaged across all the tasks) for the maze, navigation and laser-tag tasks, respectively. The x-axes are total numbers of training environment steps per task. Panel B1 shows the mean scores obtained with the distilled policies (A3C has no distilled policy, so it is represented by the performance of an untrained network.). For each algorithm, results for the best set of hyperparameters (based on the area under curve) are reported. The bold line is the average over 4 runs, and the colored area the average standard deviation over the tasks. Panels A2, B2, C2, D2 show the corresponding final performances for the 36 runs of each.