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

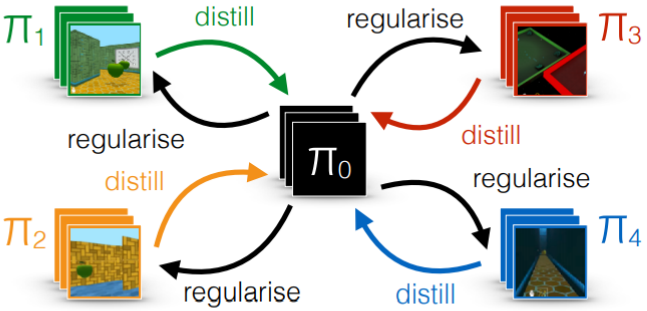
Motivation

- 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

- Motivation: Find an efficient and stable way for multitask RL
- Baseline: Multitask A3C algorithm

Overview of Distral: DIStill and TRAnsfer Learning



- Learn a global policy π_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 γ
- $a \in A$ 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.
- π_i: task specific stochastic policies

We want to optimize the expectation of reward, that is

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

Distral

- Learn a global policy π_0
- KL regularization: Ensure every task policy share information with the shared policy: $\mathbb{E}_{\pi_i}[\sum_{t\geq 0}\gamma^t \log \frac{\pi_i(a_t|s_t)}{\pi_0(a_t|s_t)}]$
- Additional entropy regularization to encourage exploration: $\mathbb{E}_{\pi_i}[\gamma^t \log \pi_i(a_t|s_t)]$
- Total loss function:

$$
\begin{aligned}
&J(\pi_0, \{\pi_i\}_{i=0}^n) \\
&= \sum_i \mathbb{E}_{\pi_i}[\sum_{t\geq 0} \gamma^t R_i(a_t, s_t) - c_{KL}\gamma^t \log \frac{\pi_i(a_t|s_t)}{\pi_0(a_t|s_t)} - c_{Ent}\gamma^t \log \pi_i(a_t|s_t)] \\
&= \sum_i \mathbb{E}_{\pi_i}[\sum_{t\geq 0} \gamma^t (R_i(a_t, s_t) + \frac{\alpha}{\beta} \log \pi_0(a_t|s_t) - \frac{1}{\beta} \log \pi_i(a_t|s_t))]
\end{aligned}
$$

$\alpha = \frac{c_{KL}}{c_{KL}+c_{Ent}}$, $\beta = \frac{1}{c_{KL}+c_{Ent}}$

Distral

$$J(\pi_0, \{\pi_i\}_{i=0}^{n}) = \sum_i \mathbb{E}_{\pi_i}[\sum_{t\geq 0} \gamma^t (R_i(a_t, s_t) + \frac{\alpha}{\beta} \log \pi_0(a_t|s_t) - \frac{1}{\beta} \log \pi_i(a_t|s_t))$$

- 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

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

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

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

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

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

- The idea is to use a Softmax function at the inverse temperature β to approximate the Bellman equation. When $\beta \to \infty$, Softmax turns into Max.

Soft Q-Learning in DisTral

- The optimal policy [from *Equivalence Between Policy Gradients and Soft Q-Learning*]

$$\begin{aligned}
\pi_i(a_t, s_t) &= \arg\max_{\pi_i}[\mathbb{E}_\pi[Q_i(s_t, a_t)] - \frac{\alpha}{\beta}\log\frac{\pi_i(a_t|s_t)}{\pi_0(a_t|s_t)}] \\
&= \pi_0^\alpha(a_t|s_t)e^{\beta Q_i(s_t,a_t) - \beta V_i(s_t)} \\
&= \pi_0^\alpha(a_t|s_t)e^{\beta A_i(s_t,a_t)}
\end{aligned}$$

  A_i here is an advantage function.
- Unlike previous literature, π_0 is learned.
- π_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 π_i a neural network and π_0 a neural network, and update them using policy gradient.
- However, maximize π_i and π_0 alternatively could be slow.
- Need a better way for a joint optimization
- Method: 1. Parameterize π_0 by

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

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

$$\hat{\pi}_i(a_t|s_t) = \frac{\exp(\alpha h_{\theta_0}(a_t|s_t) + \beta f_{\theta_i}(a_t|s_t))}{\sum_{a'} \exp(\alpha h_{\theta_0}(a'_t|s_t) + \beta f_{\theta_i}(a'_t|s_t))}$$

Algorithms

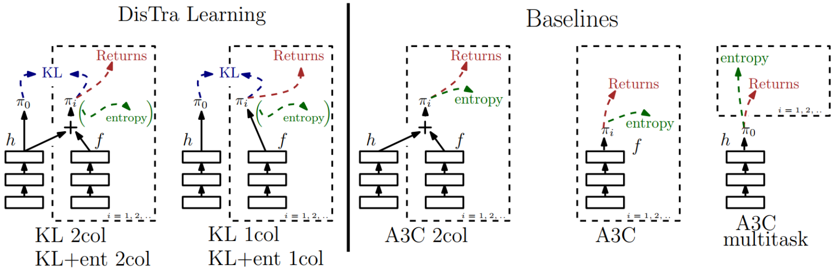
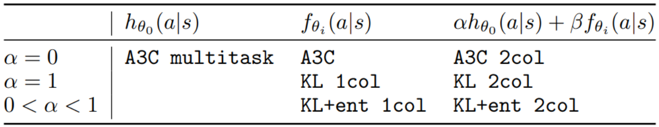


Figure 2: Depiction of the different algorithms and baselines. On the left are two of the Distral algorithms and on the right are the three A3C baselines. Entropy is drawn in brackets as it is optional and only used for KL+ent 2col and KL+ent 1col.

Experiment: Algorithms



	$h_{\theta_0}(a\|s)$	$f_{\theta_i}(a\|s)$	$\alpha h_{\theta_0}(a\|s) + \beta f_{\theta_i}(a\|s)$
$\alpha = 0$	`A3C multitask`	`A3C`	`A3C 2col`
$\alpha = 1$		`KL 1col`	`KL 2col`
$0 < \alpha < 1$		`KL+ent 1col`	`KL+ent 2col`

Table 1: The seven different algorithms evaluated in our experiments. Each column describes a different architecture, with the column headings indicating the logits for the task policies. The rows define the relative amount of KL vs entropy regularization loss, with the first row comprising the A3C baselines (no KL loss).

- Comparing 7 algorithms
- Entropy regularization vs No entropy regularization
- 1col(Alternative optimization) vs 2col(joint optimization)

Small example: Two room grid world

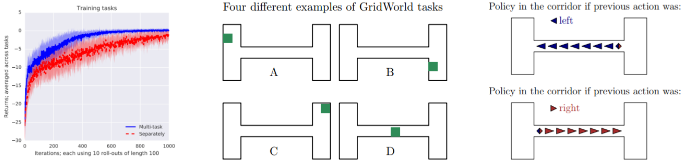


Figure 3: **Left:** Learning curves on two room grid world. The DisTraL agent (blue) learns faster, converges towards better policies, and demonstrates more stable learning overall. **Center:** Example of tasks. Green is goal position which is uniformly sampled for each task. Starting position is uniformly sampled at the beginning of each episode. **Right:** depiction of learned distilled policy π_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

Complex case

- Three complex partially observable 3D case from Deepmind Lab: Mazes, Navigation and Laser-Tag

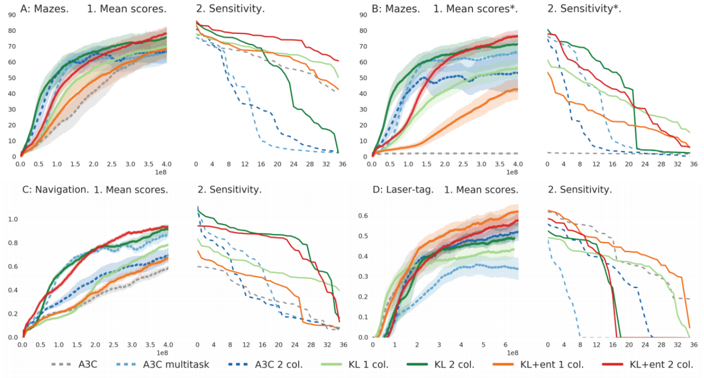


Figure 4: Panels A1, C1, D1 show task specific policy performance (averaged across all the tasks) for the maze, navigation and laser-tag tasks, respectively. The x-axes are total numbers of training environment steps per task. Panel B1 shows the mean scores obtained with the distilled policies (A3C has 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 shows the corresponding final performances for the 36 runs of each