Distral

Антон Урусов

6 ноября 2017 г.

RL recap: general statement

Постановка задачи: есть некоторое множество состояний S и множество действий A. В каждой пары (s, a) есть распределение на S, из которого выбирается следующее состояние, в которое мы перейдем из заданного, сделав действие a, а также распределение на $\mathbb{R}$, из которого выбирается значение награды за это действие.
Цель - найти стратегию π (которая представляет собой распределение на действиях для каждого состояния), максимизирующую среднюю прибыль.
При этом обычно награды дисконтируются, то прибыль имеет вид $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$.

RL recap: ценности

Определим ценность состояния:

$$V_\pi(s) = E[R_t|s_t = s] = E\left[\sum_{i=1}^{+\infty} r_{t+i}\gamma^{i-1}|s_t = s\right] \tag{1}$$

Введем также ценность действия в состоянии:

$$Q_\pi(s,a) = E[R_t|s_t = s, a_t = a] = E\left[\sum_{i=1}^{+\infty} r_{t+i}\gamma^{i-1}|s_t = s, a_t = a\right] \tag{2}$$

RL recap: Q-learning

Для решения задачи будем оценивать $Q(s, a)$, накапливая информацию о среде. Изначально проинициализируем как-то $Q^*(s, a)$, а далее будем взаимодействовать со средой по следующему алгоритму:

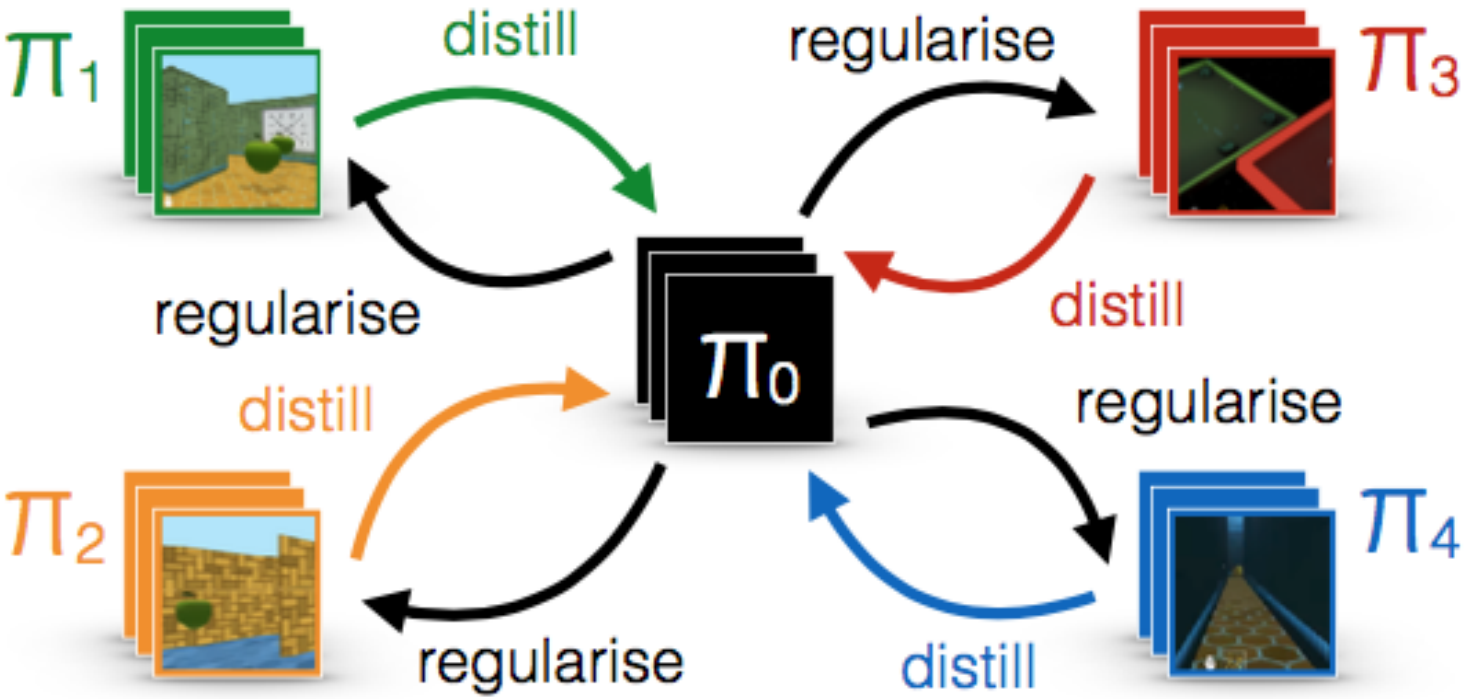
- В текущем состоянии s_t выбираем действие a_t, на основе $Q^*(s_t, a_t)$ (например, жадно)
- Получаем из среды s_{t+1}, r_t
- Обновляем значение $Q^*(s_t, a_t)$, используя $r_t + \gamma \max\limits_a Q(s_{t+1}, a)$.

При этом алгоритм, оценивающий Q^* может быть разным: в самом простом варианте мы просто собираем скользящее среднее для каждой пары (s, t), в более продвинутом DQN - используем для этого нейросеть.

Multitask problem

В жизни часто бывает так, что в одной и той же среде мы хотим решать сразу несколько задач, которые, тем не менее, могут быть очень похожими друг на друга. Соответственно, имеет смысл попробовать решать эти задачи совместно. Однако, на практике просто обучать одну сеть решать все задачи сразу - плохо работающая идея. Поэтому предлагается попробовать общую часть в отдельную сеть, и стараться сделать индивидуальные стратегии близкими к общей, что должно учиться быстрее.

Distral framework



Оптимизируемый функционал

$$J(\pi_0, \{\pi_i\}_{i=1}^n) =$$

$$\sum_i \mathbb{E}_{\pi_i} \left[\sum_{t \geq 0} \left(\gamma^t R_i(a_t, s_t) - c_{KL} \gamma^t \log \frac{\pi_i(a_t, s_t)}{\pi_0(a_t, s_t)} - c_{Ent} \gamma^t \log \pi_i(a_t, s_t) \right) \right] =$$

$$\sum_i \mathbb{E}_{\pi_i} \left[\sum_{t \geq 0} \left(\gamma^t R_i(a_t, s_t) + \frac{\gamma^t \alpha}{\beta} \log \pi_0(a_t, s_t) - \frac{\gamma^t}{\beta} \log \pi_i(a_t, s_t) \right) \right] \tag{3}$$

Объяснение функционала

- Первая часть - дисконтированный reward для каждой задачи
- Вторая обеспечивает то, что действия, имеющие высокую вероятность в общей стратегии, имеют больший приоритет
- Третья обеспечивает исследование среды
- c_{KL}, c_{Ent}, а также получаемые из них $\alpha = \frac{c_{KL}}{c_{KL}+c_{Ent}}, \beta = \frac{1}{c_{KL}+c_{Ent}}$ - гиперпараметры, которые, в данном случае, общие на все задачи, но формула легко обобщается на отдельные гиперпараметры для каждой задачи, что может быть полезно, например, в случае разного масштаба reward.

Alternating optimizing

Заметим, что при фиксированном π_0 функционал распадается на отдельную задачу максимизации для каждой задачи, которая представляет из себя стандартную задачу RL с переопределенной функцией награды $R'_i(s,a) = R_i(s,a) + \frac{\alpha}{\beta}\log\pi_0(a|s)$ и с энтропийной регуляризацией.

Soft Q-learning

Это можно оптимизировать с помощью алгоритма soft Q-learning, обновляя значения следующим образом:

$$V_i(s_t) = \frac{1}{\beta} \log \sum_{a_t} \pi_0^{\alpha}(a_t|s_t) \exp\left[\beta Q_i(a_t, s_t)\right] \tag{4}$$

$$Q_i(a_t, s_t) = R_i(a_t, s_t) + \gamma \sum_{s_{t+1}} p_i(s_{t+1}|s_t, a_t) V_i(s_{t+1}) \tag{5}$$

По сути это тот же Q-learning, но вместо строгого максимума при выборе следующего действия используется мягкий с обратной температурой β. Оптимальная стратегия же принимает вид:

$$\pi_i(a_t|s_t) = \pi_0^{\alpha}(a_t|s_t) e^{\beta Q_i(a_t, s_t) - \beta V_i(s_t)} \tag{6}$$

Обновление π_0

Теперь посмотрим на то, как обновляется π_0. В исходном функционале от нее зависит только

$$\frac{\alpha}{\beta} \sum_i \mathbb{E}_{\pi_i} \left[\sum_{t \geq 0} \gamma^t \log \pi_0(a_t | s_t) \right] \tag{7}$$

Легко видеть, что это логарифм правдоподобия для вероятностей появления a_t, s_t, дисконтированных и просуммированных для каждой задачи. В простом случае можно просто накапливать эти вероятности, в DQN - можно оптимизировать SGD.

Gradient optimizing

Приведенный алгоритм - некоторое подобие EM-алгоритма. Это может быть слишком медленным в случае с DQN. В качестве альтернативы будем просто оптимизировать градиентным спуском по параметрам.
Простой способ параметризовать - просто взять по сети для каждой стратегии, включая общую.
Рассмотрим альтернативный способ, который, по мнению авторов, может обеспечивать более быстрый перенос знаний. Для этого посмотрим на то, как представляется оптимальная частная стратегия при фиксированной общей.

Alternative model representation

Параметризуем π_0 сетью с параметрами θ_0:

$$\hat{\pi}_0 = \frac{\exp(h_{\theta_0}(a_t|s_t))}{\sum_{a'} \exp(h_{\theta_0}(a'|s_t))} \tag{8}$$

Также будем оценивать A_i отдельной сетью для каждой задачи:

$$\hat{A}_i = f_{\theta_i}(a_t|s_t) - \frac{1}{\beta} \log \sum_a \hat{\pi}_0^{\alpha}(a|s_t) \exp(\beta f_{\theta_i}(a, s_t)) \tag{9}$$

Тогда стратегия для отдельной задачи примет вид:

$$\hat{\pi}_i(a_t, s_t) = \hat{\pi}_0(a_t|s_t) \exp(\beta \hat{A}_i(a_t|s_t)) = \frac{\exp(h_{\theta_0}(a_t|s_t) + \beta f_{\theta_i}(a_t|s_t))}{\sum_{a'} \exp(h_{\theta_0}(a'|s_t) + \beta f_{\theta_i}(a'|s_t))} \tag{10}$$

Gradient of alternative model representation

Градиент у этой функции получается следующий:

$$\nabla_{\theta_i} J = \mathbb{E}_{\hat{\pi}_i} \left[\sum_{t \geq 1} \nabla_{\theta_i} \log \hat{\pi}_i(a_t|s_t) \left(\sum_{u \geq t} \gamma^u \left(R_i^{reg}(a_u, s_u) \right) \right) \right] \quad (11)$$

Здесь $R_i^{reg}(a, s) = R_i(a, s) + \frac{\alpha}{\beta} \log \hat{\pi}_0(a|s) - \frac{1}{\beta} \log \hat{\pi}_i(a|s)$ - регуляризованная награда. Градиент для общей стратегии получается следующим:

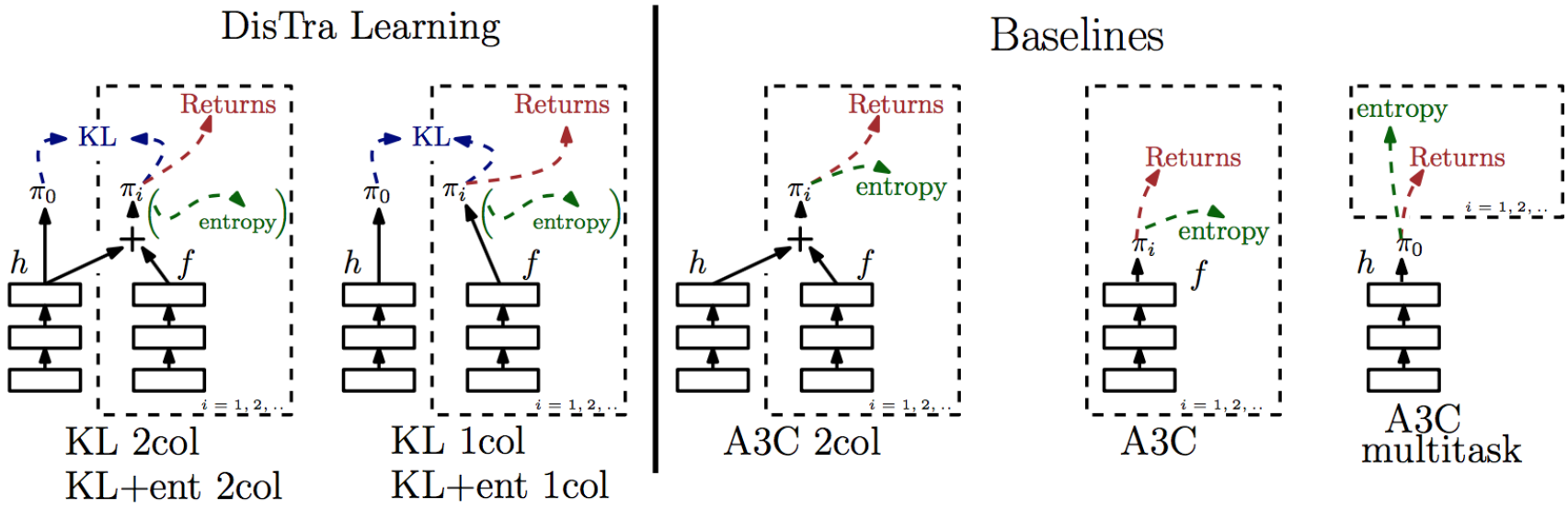
$$\nabla_{\theta_0} J = \sum_i \mathbb{E}_{\hat{\pi}_i} \left[\sum_{t \geq 1} \nabla_{\theta_i} \log \hat{\pi}_i(a_t|s_t) \left(\sum_{u \geq t} \gamma^u \left(R_i^{reg}(a_u, s_u) \right) \right) \right] + \frac{\alpha}{\beta} \sum_i \mathbb{E}_{\hat{\pi}_i} \left[\sum_{t \geq 1} \gamma^t \sum_{a'_t} (\hat{\pi}_i(a'_t|s_t) - \hat{\pi}_0(a'_t|s_t)) \nabla_{\theta_0} h_{\theta_0}(a'_t|s_t) \right] \quad (12)$$

Algorithm parameters

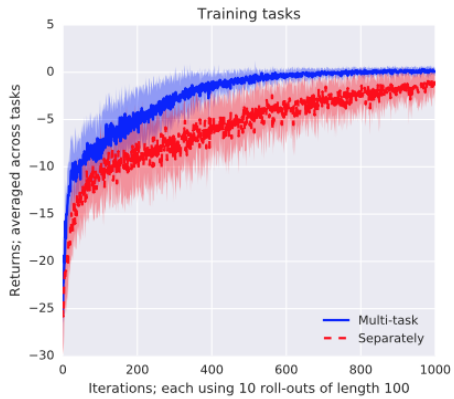
Из вышесказанного следуют следующие параметры алгоритмов:

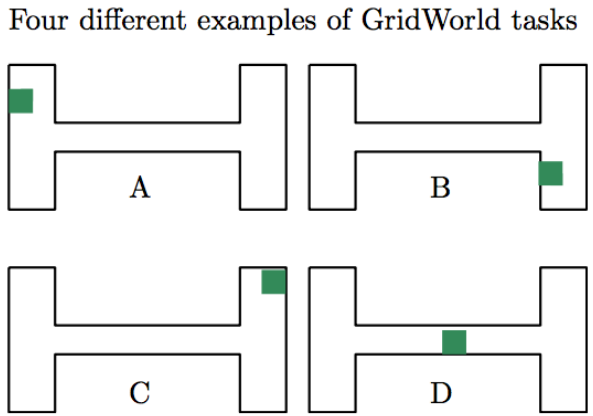
- KL vs entropy регуляризация
  - $\alpha = 0$ - без KL-регуляризации
  - $\alpha = 1$ - без энтропийной регуляризации
  - $0 < \alpha < 1$ - комбинация двух
- Alternating vs joint optimization. Оптимизация попеременно обычно медленнее, но стабильнее.
- Separate vs 2col optimization. В первом случае все стратегии, включая общую, представляются независимыми сетями, во втором - альтернативным способом, описанным выше.

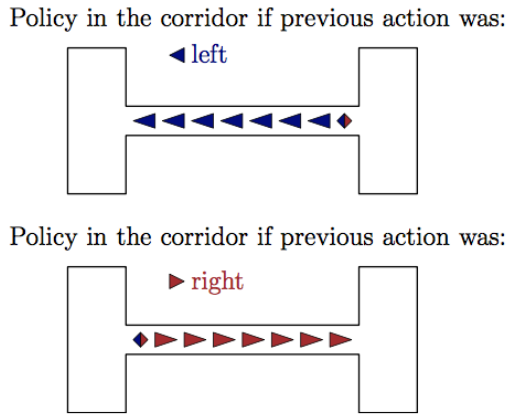
Algorithms


DisTra Learning
Baselines
Returns
KL
entropy
π_0
π_i
h
f
$i = 1, 2, ..$
KL 2col
KL+ent 2col
KL 1col
KL+ent 1col
A3C 2col
A3C
A3C multitask

Experiments: simple tasks

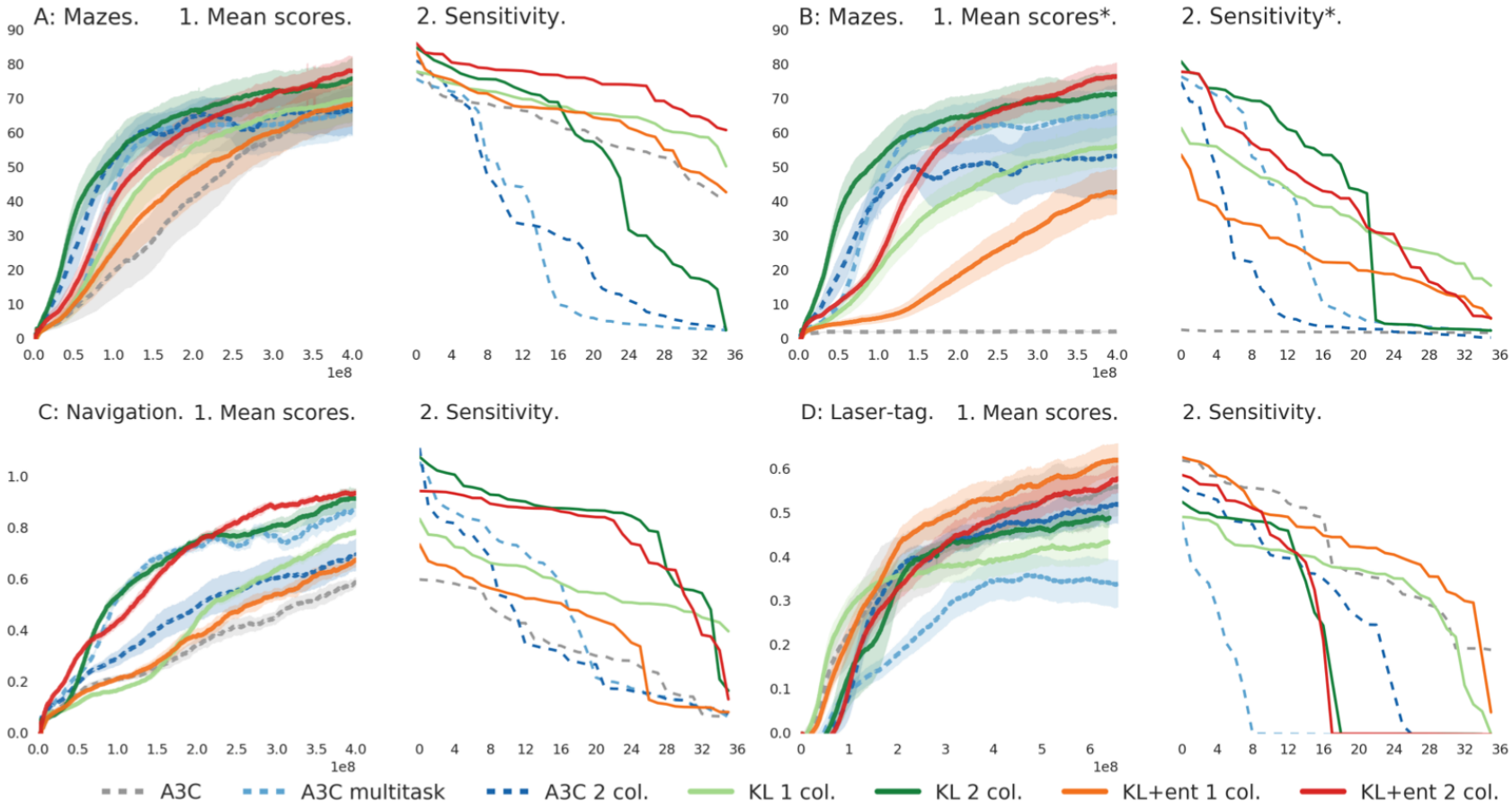






В качестве состояния для MDP используется тройка (location, prev. action, prev. reward).

Complex tasks



Complex tasks descriptions

- Maze. Каждая задача - случайный лабиринт со случайно раскиданными наградами и случайно сгенерированной целевой позицией.
- Navigation. Случайные карты, генерируемые на каждом проходе заново. Первая задача задает случайные объекты, которые надо собирать, вторая дополнительно требует вернуться в начало, третья имеет зафиксированную точку выхода, а четвертая добавляет случайные двери.
- Lazertag. Надо играть против ботов, которые могут быть очень разными.

Results: maze

- Baselines учатся нестабильно: иногда работают хорошо, иногда совсем плохо.
- Лучший алгоритм KL + Ent, 2col.
- Общая стратегия получается лучшей у 2col алгоритмов.
- Общая стратегия получается хуже с добавлением энтропийной регуляризации

Results: navigation

- Distral лучше и стабильнее.
- 2col работает лучше.

Results: lazer-tag

- Лучший из baselines - A3C.
- Лучшие из Distral - 1col.
- Distral работает лучше, чем A3C multitask и на уровне с A3C.

Литература

- Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, Razvan Pascanu. Distral: Robust Multitask Reinforcement Learning.
- Roy Fox, Michal Moshkovitz, and Naftali Tishby. Principled option learning in markov decision processes. In European Workshop on Reinforcement Learning (EWRL), 2016.