DIVERSITY IS ALL YOU NEED: LEARNING SKILLS WITHOUT A REWARD FUNCTION

Benjamin Eysenbach*
Carnegie Mellon University
beysenba@cs.cmu.edu

Abhishek Gupta
UC Berkeley

Julian Ibarz
Google Brain

Sergey Levine
UC Berkeley
Google Brain

【强化学习 77】DIAYN

张楚珩
清华大学 交叉信息院博士在读

DIAYN 算法全称是 Diversity Is All You Need，这里讲它以及同样这一拨人接着它做的一个 meta learning 的工作。

原文传送门

Eysenbach, Benjamin, et al. "Diversity is all you need: Learning skills without a reward function." arXiv preprint arXiv:1802.06070 (2018).

Gupta, Abhishek, et al. "Unsupervised Meta-Learning for Reinforcement Learning." arXiv preprint arXiv:1806.04640 (2018).

Finn, Chelsea, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks." Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017.

特色

DIAYN 在给定一个没有奖励的环境中能够自适应地产生奖励函数、并且自动探索出来一些有用的技能（skills），学习到的技能可以用于 1）作为策略学习的初始化，加速学习；2）作为分层强化学习的下层策略；3）用于模仿学习。

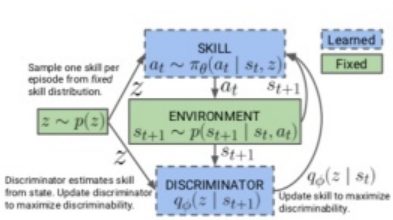
后面紧接着的一篇是这一组人把 DIAYN 用于 meta-learning 的工作。

过程

1. DIAYN

DIAYN 主要包括两个网络：一个是策略网络（下图中的蓝色框 SKILL），它和通常策略网络的区别在于，它还接受一个控制该策略的参数 z，用于生成不同的策略；另外一个是一个判别器网络（下图蓝色框中的 DISCRIMINATOR），它的作用是给定一个由参数 z 控制的策略方位的状态，输出该状态可能对应参数 z 的概率分布。

其目标如下，给定一个状态，学习一个判别器来预测这个状态最可能被哪个参数 z 控制下的策略访问到。同时，给定一个参数 z，希望它对应的策略能够访问到和其他参数不一样的状态空间，因此训练该策略的奖励函数可以由前面学习到的判别器来给出，即如果访问到了一个关于参数 z "特殊"的状态，就给予较大的奖励。



Algorithm 1: DIAYN

while *not converged* **do**
  Sample skill $z \sim p(z)$ and initial state $s_0 \sim p_0(s)$
  for $t \leftarrow 1$ **to** $steps\_per\_episode$ **do**
    Sample action $a_t \sim \pi_\theta(a_t \mid s_t, z)$ from skill.
    Step environment: $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$.
    Compute $q_\phi(z \mid s_{t+1})$ with discriminator.
    Set skill reward $r_t = \log q_\phi(z \mid s_{t+1}) - \log p(z)$
    Update policy (θ) to maximize r_t with SAC.
    Update discriminator (ϕ) with SGD.

Figure 1: **DIAYN Algorithm**: We update the discriminator to better predict the skill, and update the skill to visit diverse states that make it more discriminable.

策略训练的过程可以最大化以下目标

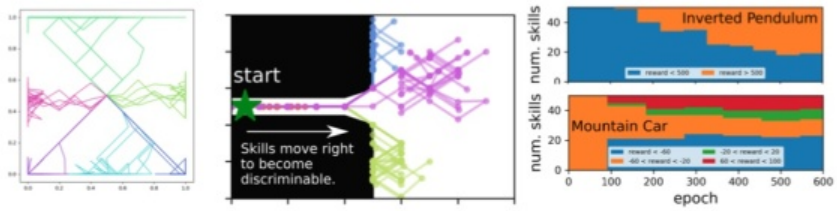
$$\begin{aligned}\mathcal{F}(\theta) &\triangleq I(S;Z) + \mathcal{H}[A \mid S] - I(A;Z \mid S) \quad (1)\\ &= (\mathcal{H}[Z] - \mathcal{H}[Z \mid S]) + \mathcal{H}[A \mid S] - (\mathcal{H}[A \mid S] - \mathcal{H}[A \mid S,Z])\\ &= \mathcal{H}[Z] - \mathcal{H}[Z \mid S] + \mathcal{H}[A \mid S,Z] \quad (2)\end{aligned}$$

$$\begin{aligned}\mathcal{F}(\theta) &= \mathcal{H}[A \mid S,Z] - \mathcal{H}[Z \mid S] + \mathcal{H}[Z]\\ &= \mathcal{H}[A \mid S,Z] + \mathbb{E}_{z\sim p(z), s\sim\pi(z)}[\log p(z \mid s)] - \mathbb{E}_{z\sim p(z)}[\log p(z)]\\ &\geq \mathcal{H}[A \mid S,Z] + \mathbb{E}_{z\sim p(z), s\sim\pi(z)}[\log q_\phi(z \mid s) - \log p(z)] \triangleq \mathcal{G}(\theta,\phi)\end{aligned}$$

由此，策略训练可以设定奖励函数为

$$r_z(s,a) \triangleq \log q_\phi(z \mid s) - \log p(z) \quad (3)$$

贴一张图让大家对于 DIAYN 学习到的策略有一定的直观感受。



(a) 2D Navigation (b) Overlapping Skills (c) Training Dynamics

Figure 2: *(Left)* DIAYN skills in a simple navigation environment; *(Center)* skills can overlap if they eventually become distinguishable; *(Right)* diversity of the rewards increases throughout training.

2. 用于策略初始化

这一点比较直观，就直接把学习到的策略网络权重作为权重初始化。唯一不一样的是，DIAYN 学到的策略网络还接受一个 z 的输入，可能用于权值初始化的时候把与之相关的权重去掉吧。效果如下。

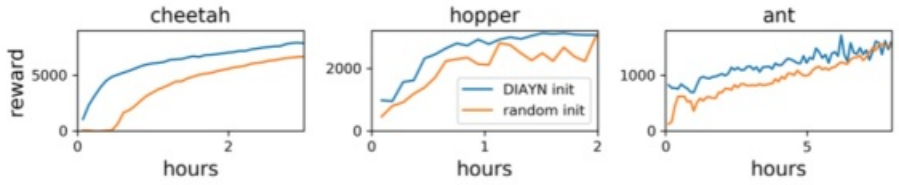


Figure 5: **Policy Initialization**: Using a DIAYN skill to initialize weights in a policy accelerates learning, suggesting that pretraining with DIAYN may be especially useful in resource constrained settings. Results are averages across 5 random seeds.

3. 用于分层强化学习

把学习到的 skills 用于作为 option（action），另外训练一个 meta-controller 来在不同的 skills 上做选择，meta-controller 的学习使用普通的强化学习算法就好。这里设计了两种比较困难的任务（具体见原文），可以看到把 DIAYN 学习到的 skills 作为 option 效果比直接学习更好。这里还提到了一种使用一些先验知识使得学习到的 skills更为有用的办法（即，下图中的 DIAYN+prior）。比如，对于蜘蛛来说，我们希望它探索出来的不同策略主要反映在它能够到达的位置的不同，就可以在判别器前加上一个先验，即 $q_\phi(z|s) \rightarrow q_\phi(z|f(s))$ ，而 $f(s)$ 只反映蜘蛛的质心位置。

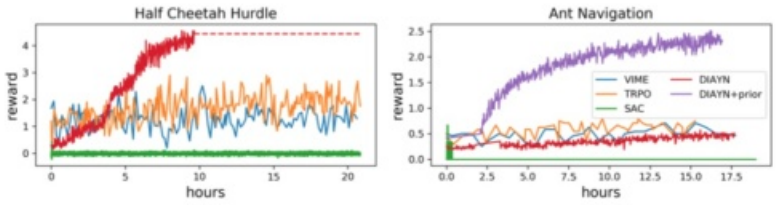


Figure 7: **DIAYN for Hierarchical RL**: By learning a meta-controller to compose skills learned by DIAYN, cheetah quickly learns to jump over hurdles and ant solves a sparse-reward navigation task.

4. 用于模仿学习

给定专家的轨迹之后，挑选出来一个 z 使得它能够最大可能地产生专家轨迹上的状态，然后把对应的策略作为模仿学习的策略。

$$\hat{z} = \arg\max_z \Pi_{s_t \in \tau^*} q_\phi(z \mid s_t)$$

5. 用于 meta-learning

这部分的内容是第二篇文章讲的。其主要利用了 DIAYN 能够自动产生奖励函数（学习目标）的特点，来避免 meta-learning 中需要人为设计任务的环节。这里 meta-learning 的设定如下。给定一个没有 reward 的 MDP，即 controlled Markov process （CMP），$C = (\mathcal{S}, \mathcal{A}, T, \gamma, \rho)$，其中最后一个代表初始状态分布。在该环境下进行探索和学习，之后在给定特定奖励函数之后，希望能够快速地学习到一个较好的策略。大致流程如下图所示。

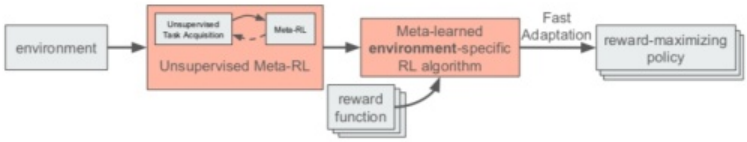


Figure 1: **Unsupervised meta-reinforcement learning**: Given an environment, unsupervised meta-reinforcement learning produces an environment-specific learning algorithm that quickly acquires new policies that maximizes any task reward function.

算法过程如下，先使用 DIAYN 得到一个判别器（其实就是得到一族有意义的奖励函数，或者一族自动生成的 MDP），然后在这一族任务中使用 model-agnostic meta-learning （MAML） 来学习。

Algorithm 1: Unsupervised Meta-Reinforcement Learning Pseudocode

Data: $\mathcal{M} \setminus R$, an MDP without a reward function
Result: a learning algorithm $f : \mathcal{D} \to \pi$
Initialize $\mathcal{D} = \emptyset$
$D_\phi \leftarrow$ DIAYN() or $D_\phi \leftarrow random$
while *not converged* **do**
  Sample latent task variables $z \sim p(z)$
  Extract corresponding task reward functions $r_z(s)$ using $\mathcal{D}_\phi(z|s)$
  update f using MAML with reward $r_z(s)$

6. 补充：MAML

补充一下 MAML 的学习的主要思路。其目标是给定一组任务，希望能够找到一个参数控制的”中间“策略，使得给定一个具体任务的时候，该策略能够快速地调整到该任务下较好的策略。其训练目标如下

$$\max_\theta \sum_{\tau_i \sim p(\tau)} \mathbb{E}_{\pi_{\theta_i'}} \left[\sum_t R_i(s_t) \right] \quad \theta' = \theta + \alpha \mathbb{E}_{\pi_\theta} \left[\sum_t R_i(s_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \right] \tag{2}$$

其中， θ 是中间策略的策略参数。训练的时候，每给定一个任务 τ_i 之后，就使用“中间”策略做一次 rollout，然后使用相应的数据做一次策略梯度更新（右边的式子），得到相应的策略参数 θ_i' 。目标就是希望调整 θ 希望做过一次策略梯度更新之后的策略 θ_i' 在任务 τ_i 下表现最好。

编辑于 2019-10-16

强化学习 (Reinforcement Learning)