Mix & Match – Agent Curricula for Reinforcement Learning [Appendix]

Wojciech Marian Czarnecki [*1] **Siddhant M. Jayakumar** [*1] **Max Jaderberg** [1] **Leonard Hasenclever** [1]
Yee Whye Teh [1] **Simon Osindero** [1] **Nicolas Heess** [1] **Razvan Pascanu** [1]

1 Network architectures

Default network architecture consists of:

- Convolutional layer with 16 8x8 kernels of stride 4
- ReLU
- Convolutional layer with 32 4x4 kernels of stride 2
- ReLU
- Linear layer with 256 neurons
- ReLU
- Concatenation with one hot encoded last action and last reward
- LSTM core with 256 hidden units
  - Linear layer projecting onto policy logits, followed by softmax
  - Linear layer projecting onto baseline

Depending on the experiment, some elements are shared and/or replaced as described in the text.

2 PBT (Jaderberg et al., 2017) details

In all experiments PBT controls adaptation of three hyperparameters: α, learning rate and entropy cost regularisation. We use populations of size 10.

The `explore` operator for learning rate and entropy regularisation is the permutation operator, which randomly multiplies the corresponding value by 1.2 or 0.8. For α it is an adder operator, which randomly adds or substracts 0.05 and truncates result to $[0, 1]$ interval. Exploration is executed with probability 25% independently each time worker is ready.

[*]Equal contribution [1]DeepMind, London, UK. Correspondence to: Wojciech M. Czarnecki <lejlot@google.com>, Siddhant M. Jayakumar <sidmj@google.com>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

The `exploit` operator copies all the weights and hyperparameters from the randomly selected agent if it's performance is significantly better.

Worker is deemed ready to undergo adaptation each 300 episodes.

We use T-Test with p-value threshold of 5% to answer the question whether given performance is significantly better than the other, applied to averaged last 30 episodes returns.

Initial distributions of hyperparameters are as follows:

- learning rate: loguniform(1e-5, 1e-3)
- entropy cost: loguniform(1e-4, 1e-2)
- alpha: loguniform(1e-3, 1e-2)

2.1 Single task experiments

The `eval` function uses π_{mm} rewards.

2.2 Multi task experiments

The `eval` function uses π_{mt} rewards, which requires a separate evaluation worker per learner.

3 M&M details

λ for action space experiments is set to 1.0, and for agent core and multitask to 100.0. In all experiments we allow backpropagation through both policies, so that teacher is also regularised towards student (and thus does not diverge too quickly), which is similar to Distral work.

While in principle we could also transfer knowledge between value functions, we did not find it especially helpful empirically, and since it introduces additional weight to be adjusted, we have not used it in the reported experiments.

4 IMPALA (Espeholt et al., 2018) details

We use 100 CPU actors per one learner. Each learner is trained with a single K80 GPU card. We use vtrace correction with truncation as described in the original paper.

Agents are trained with a fixed unroll of 100 steps. Optimisation is performned using RMSProp with decay of 0.99,

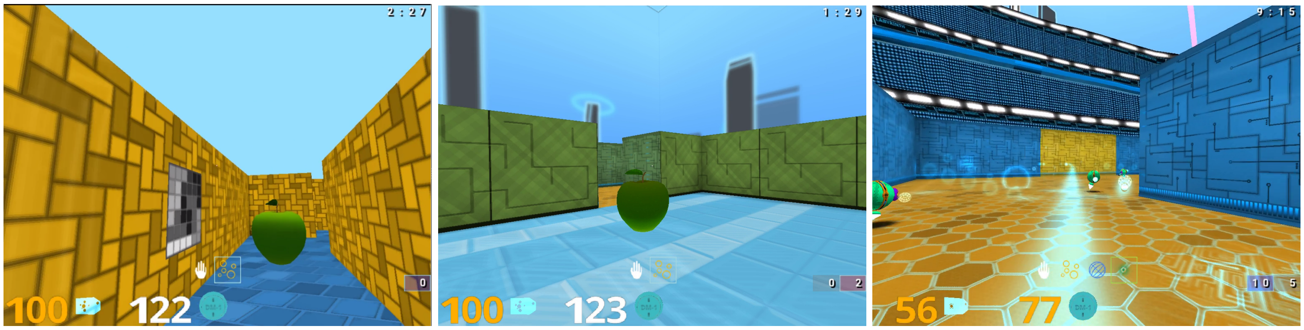

Figure 1. Exemplary tasks of interest from DM Lab environment (Beattie et al., 2016). From left: *Nav maze static 02* – The task involves finding apples (+1 reward) and a final goal (+10 reward) in a fixed maze. Every time an agent finds the goal it respawns in a random location, and all objects respawn too. *Explore Object Locations Small* – The task is to navigate through a 3D maze and eat all the apples (each gives +1 reward). Once it is completed, the task restarts. Each new episode differs in terms of apples locations, map layout as well as visual theme. *Lt Horseshoe Color* – the task is a game of lasertag, where player tries to tag as many of high skilled built-in bots as possible, while using pick-up gadgets (which enhance tagging capabilities).

epsilon of 0.1. Discounting factor is set to 0.99, baseline fitting cost is 0.5, rewards are clipped at 1. Action repeat is set to 4.

5 Environments

We ran DM Lab using $96 \times 72 \times 3$ RGB observations, at 60 fps.

5.1 Explore Object Locations Small

The task is to find all apples (each giving 1 point) in the procedurally generated maze, where each episode has different maze, apples locations as well as visual theme. Collecting all apples resets environment.

5.2 Nav Maze Static 01/02

Nav Maze Static 01 is a fixed geometry maze with apples (worth 1 point) and one calabash (worth 10 points, getting which resets environment). Agent spawns in random location, but walls, theme and objects positions are held constant.

The only difference for Nav Maze Static 02 is that it is significantly bigger.

5.3 LaserTag Horseshoe Color

Laser tag level against 6 built-in bots in a wide horseshoe shaped room. There are 5 Orb Gadgets and 2 Disc Gadgets located in the middle of the room, which can be picked up and used for more efficient tagging of opponents.

5.4 LaserTag Chasm

Laser tag level in a square room with Beam Gadgets, Shield Pickups (50 health) and Overshield Pickups (50 armor) hanging above a tagging floor (chasm) splitting room in half. Jumping is required to reach the items. Falling into the chasm causes the agent to lose 1 point. There are 4 built-in bots.

6 Proofs

First let us recall the loss of interest

$$\mathcal{L}_{\mathrm{mm}}(\theta) = \frac{1-\alpha}{|S|} \sum_{s \in S} \sum_{t=1}^{|s|} \mathrm{D}_{\mathrm{KL}}(\pi_1(\cdot|s_t) \| \pi_2(\cdot|s_t)), \quad (1)$$

where each $s \in S$ come from $\pi_{\mathrm{mm}} = (1-\alpha)\pi_1 + \alpha\pi_2$.

Proposition 1. *Lets assume we are given a set of N trajectories from some predefined mix $\pi_{\mathrm{mm}} = (1-\alpha)\pi_1 + \alpha\pi_2$ for any fixed $\alpha \in (0,1)$ and a big enough neural network with softmax output layer as π_2. Then in the limit as $N \to \infty$, the minimisation of Eq. 1 converges to π_1 if the optimiser used is globally convergent when minimising cross entropy over a finite dataset.*

Proof. For D_N denoting set of N sampled trajectories over state space $\mathcal{S}$ let as denote by $\hat{\mathcal{S}}_N$ the set of all states in D_N, meaning that $\hat{\mathcal{S}}_N = \cup D_N$. Since π_2 is a softmax based policy, it assigns non-zero probability to all actions in every state. Consequently also π_{mm} does that as $\alpha \in (0,1)$. Thus we have

$$\lim_{N\to\infty} \hat{\mathcal{S}}_N = \mathcal{S}.$$

Due to following the mixture policy, actual dataset $\hat{D}_N$ gathered can consist of multiple replicas of each element in $\hat{\mathcal{S}}_N$, in different proportions that one would achieve when following π_1. Note, note however that if we use optimiser which is capable of minimising the cross entropy over finite dataset, it can also minimise loss (1) over $\hat{D}_N$ thus in particular over $\hat{\mathcal{S}}_N$ which is its strict subset. Since the network is big enough, it means that it will converge to 0 training error:

$$\forall_{s \in \hat{\mathcal{S}}_N} \lim_{t\to\infty} \mathrm{D}_{\mathrm{KL}}(\pi_1(a|s)\|\pi_2(a|s,\theta_t)) = 0$$

where θ_t is the solution of tth iteration of the optimiser used. Connecting the two above we get that in the limit of N and t

$$\forall_{s \in \mathcal{S}} \mathrm{D}_{\mathrm{KL}}(\pi_1(a|s_t)\|\pi_2(a|s_t,\theta_t)) = 0 \iff \pi_1 = \pi_2.$$

$\square$

While the global convergence might sound like a very strong property, it holds for example when both teacher and student policies are linear. In general for deep networks it is hypothesised that if they are big enough, and well initialised, they do converge to arbitrarily small training error even if trained with a simple gradient descent, thus the above proposition is not too restrictive for Deep RL.

7 On α based scaling of knowledge transfer loss

Let as take a closer look at the proposed loss

$$\ell_{\mathrm{mm}}(\theta) = (1-\alpha)\mathrm{D}_{\mathrm{KL}}(\pi_1(\cdot|s)\|\pi_2(\cdot|s)) =$$

$$= (1-\alpha)\mathrm{H}(\pi_1(\cdot|s)\|\pi_2(\cdot|s)) - (1-\alpha)\mathrm{H}(\pi_1(\cdot|s))$$

and more specifically at $1-\alpha$ factor. The intuitive justification for this quantity is that it leads to D_{KL} gradually disappearing as M&M agent is switching to the final agent. However, one can provide another explanation. Let us instead consider divergence between mixed policy and the target policy (which also has the property of being 0 once agent switches):

$$\hat{\ell}_{\mathrm{mm}}(\theta) = \mathrm{D}_{\mathrm{KL}}(\pi_{\mathrm{mm}}(\cdot|s)\|\pi_2(\cdot|s)) =$$

$$= \mathrm{H}(\pi_{\mathrm{mm}}(\cdot|s)\|\pi_2(\cdot|s)) - \mathrm{H}(\pi_{\mathrm{mm}}(\cdot|s)) =$$

$$= \mathrm{H}((1-\alpha)\pi_1(\cdot|s) + \alpha\pi_2(\cdot|s)\|\pi_2(\cdot|s)) - \mathrm{H}(\pi_{\mathrm{mm}}(\cdot|s))$$

$$= \mathrm{H}((1-\alpha)\pi_1(\cdot|s)\|\pi_2(\cdot|s)) + \alpha\mathrm{H}(\pi_2(\cdot|s) - \mathrm{H}(\pi_{\mathrm{mm}}(\cdot|s))$$

$$= (1-\alpha)\mathrm{H}(\pi_1(\cdot|s)\|\pi_2(\cdot|s)) - (\mathrm{H}(\pi_{\mathrm{mm}}(\cdot|s)) - \alpha\mathrm{H}(\pi_2(\cdot|s))$$

One can notice, that there are two factors of both losses, one being a cross entropy between π_1 and π_2 and the other being a form of entropy regularisers. Furthermore, these two losses differ only wrt. regularisations:

$$\ell_{\mathrm{mm}}(\theta) - \hat{\ell}_{\mathrm{mm}}(\theta) =$$

$$= -(1-\alpha)\mathrm{H}(\pi_1(\cdot|s)) + (\mathrm{H}(\pi_{\mathrm{mm}}(\cdot|s)) - \alpha\mathrm{H}(\pi_2(\cdot|s)) =$$

$$= \mathrm{H}(\pi_{\mathrm{mm}}(\cdot|s)) - (\alpha\mathrm{H}(\pi_2(\cdot|s) + (1-\alpha)\mathrm{H}(\pi_1(\cdot|s)))$$

but since entropy is concave, this quantitiy is non-negative, meaning that

$$\ell_{\mathrm{mm}}(\theta) \geq \hat{\ell}_{\mathrm{mm}}(\theta)$$

therefore

$$-(1-\alpha)\mathrm{H}(\pi_{\mathrm{mm}}(\cdot|s)) \geq -(\mathrm{H}(\pi_{\mathrm{mm}}(\cdot|s)) - \alpha\mathrm{H}(\pi_2(\cdot|s))$$

Thus the proposed scheme is almost equivalent to minimising KL between mixed policy and π_2 but simply with more severe regularisation factor (and thus it is the upper bound of the $\hat{\ell}_{\mathrm{mm}}$.

Further research and experiments need to be performed to asses quantitative differences between these costs though. In preliminary experiments we ran, the difference was hard to quantify – both methods behaved similarly well.

8 On knowledge transfer loss

Through this paper we focused on using Kulback-Leibler Divergence for knowledge transfer $\mathrm{D}_{\mathrm{KL}}(p\|q) = \mathrm{H}(p,q) - \mathrm{H}(p)$. For many distillation related methods, it is actually equivalent to minimising cross entropy (as p is constant), in M&M case the situation is more complex. When both p and q are learning D_{KL} provides a two-way effect – from one perspective q is pulled towards p and on the other p is mode seeking towards q while at the same time being pushed towards uniform distribution (entropy maximisation). This has two effects, first, it makes it harder for the teacher to get too ahead of the student (similarly to (Teh et al., 2017; Zhang et al., 2017)); second, additional entropy term makes it expensive to keep using teacher, and so switching is preffered.

Another element which has not been covered in depth in this paper is possibility of deep distillation. Apart from matching policies one could include inner activation matching (Parisotto et al., 2016), which could be beneficial for deeper models which do not share modules. Furthermore, for speeding up convergence of distillation one could use Sobolev Training (Czarnecki et al., 2017) and match both policy and its Jacobian matrix. Since policy matching was enough for current experiments, none of these methods has been used in this paper, however for much bigger models and more complex domains it might be the necesity as M&M depends on ability to rapidly transfer knowledge between agents.

References

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Legg, S., and Petersen, S. Deepmind lab. *CoRR*, 2016.

Czarnecki, W. M., Osindero, S., Jaderberg, M., Swirszcz, G., and Pascanu, R. Sobolev training for neural networks. In *Advances in Neural Information Processing Systems*, pp. 4281–4290, 2017.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., Fernando, C., and Kavukcuoglu, K. Population based training of neural networks. *CoRR*, 2017.

Parisotto, E., Ba, L. J., and Salakhutdinov, R. Actor-mimic: Deep multitask and transfer reinforcement learning. *ICLR*, 2016.

Teh, Y., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. Distral: Robust multitask reinforcement learning. In *NIPS*. 2017.

Zhang, Y., Xiang, T., Hospedales, T. M., and Lu, H. Deep mutual learning. *arXiv preprint arXiv:1706.00384*, 2017.