Taming the Herd: Multi-Modal Meta-Learning with a Population of Agents

Robert Müller [1] **Jack Parker-Holder** [2] **Aldo Pacchiano** [3]

Abstract

Meta-learning is a paradigm whereby an agent is trained with the specific goal of fast adaptation. With promising recent results, it is an effective way to use prior knowledge for future learning. However, most of the prominent successes focus entirely on adapting to similar tasks, from a unimodal distribution. This drastically differs from the real world, where problems may be more diverse. In this paper we address this issue, and provide a simple approach to solve it, which we call *Taming the Herd*. The Herd refers to a population of agents, each specializing on a subset (or mode) of the task distribution. At test time, we automatically allocate the appropriate member of the Herd and thus perform comparably with an oracle trained solely on those tasks. We apply our approach to both MAML and PEARL, and demonstrate its efficacy on a simple yet challenging multi-modal task distribution.

1. Introduction

In recent times there has been great excitement in the field of meta-learning, where agents are trained with the goal of adapting to new tasks. This approach resonates with many since it offers the promise of agents which can learn something more fundamental than just an individual task. Our focus is on meta-*reinforcement*-learning, where agents can quickly learn new RL tasks.

Model-agnostic meta-learning (MAML, (Finn et al., 2017)) is a popular approach, which seeks to learn a set of parameters which can successfully adapt to a new task in just a few steps. Its simplicity and effectiveness make it a canonical algorithm in the space, inspiring a series of follow-up work. One crucial limitation of MAML is its requirement to train on **unimodal** task distributions. This means that in reality, it can only adapt to a few very similar tasks. Thus far, existing approaches to improve MAML have focused on improving efficiency (Rakelly et al., 2019; Behl et al., 2019), while continuing to focus on the unimodal distributions.

In this paper we take the first step towards moving to more challenging task distributions. We use task embeddings to learn clusters of similar tasks, in an unsupervised manner, and train a population of meta-learners, each specialized to one of the learned task clusters. At test time we use a probe policy to assign the new task to the closest member of the learned cluster population, which can then rapidly adapt in the remaining test steps. We call our approach: *Taming the Herd* and depict it in Figure 1.

Our contributions are twofold: first we propose a variety of task embeddings to meaningfully measure similarity between tasks for the purpose of meta-learning. Second, we introduce a way to leverage the task geometry induced by these embeddings to improve existing meta-learning algorithms such as MAML (Finn et al., 2017) and PEARL (Rakelly et al., 2019). We show that by incorporating these insights, both MAML and PEARL can deal with multi-modal task distributions, where they would otherwise fail.

The paper is organized as follows. We begin with related work, and required notation in Section 3. Next we introduce our algorithm, before demonstrating the efficacy of our approach on a challenging multi-modal task.

2. Related Work

Meta-learning as a concept has been around for decades (Schmidhuber, 1987; Naik and Mammone, 1992). However, with the introduction of model-agnostic meta-learning (MAML (Finn et al., 2017)), it has experienced a surge of interest. MAML seeks to learn a set of weights which are amenable to fast adaptation, by backpropagating through the update step. This has led to a series of follow up works (Antoniou et al., 2019; Behl et al., 2019; Rajeswaran et al., 2019), all of which assume access to a distribution of related tasks. We will introduce MAML more formally in Sec: 3.

More recently, a series of works have considering alternative ways to explicitly decouple the task inference module, from the learned shared representation that allows the agent to generalize across tasks. One existing approach is to explicitly allocate a subset, or a low dimensional representation

[1]Technical University of Munich [2]University of Oxford [3]UC Berkeley. Correspondence to: Robert Müller <2robert.mueller@gmail.com>.

4th Lifelong Learning Workshop at ICML 2020,

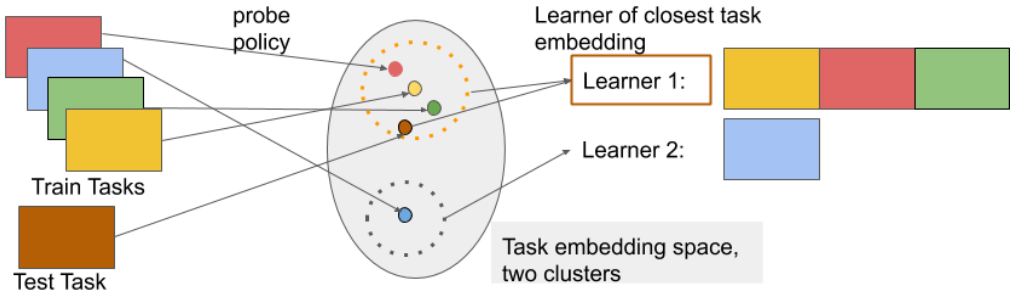


Figure 1. Pictogram of our proposed divide and conquer of the task space approach

of the parameters to be updated during the adaptation phase such as (Zintgraf et al., 2019; Rusu et al., 2019). A competing approach, and one we explore further in this work is to learn a task embedding aimed at inferring the nature of the task at hand and produce a policy conditioned on this information (Rakelly et al., 2019; Zintgraf et al., 2020). Despite these advances, these approaches fail to deal with multi-modal task distributions. We show that with our approach, PEARL can perform well in this setting.

Our work is connected to Task2Vec (Achille et al., 2019) in that we also embed tasks. While their work selects the best classifier on a given task we do not stop at this point but use the identified closest ensemble member for meta-testing.

In addition, a recent paper suggested MAML's success is partially due to effective feature learning (Raghu et al., 2020). The performance gap for a single MAML suggests these features may not generalize across multiple modes of a task distribution. Beyond meta-learning, multi-modal task distributions arise in model-based RL (Kaushik et al., 2020) or natural language processing (Vu et al., 2020).

3. Background

3.1. Reinforcement Learning

A Markov Decision Process (MDP, (Bellman, 1957)) is a tuple $(\mathcal{S}, \mathcal{A}, \mathrm{P}, \mathrm{R})$. Here $\mathcal{S}$ and $\mathcal{A}$ stand for the sets of states and actions respectively, such that for $s_t, s_{t+1} \in \mathcal{S}$ and $a_t \in \mathcal{A}$: $\mathrm{P}(s_{t+1}|s_t, a_t)$ is the probability that the system/agent transitions from s_t to s_{t+1} given action a_t and $\mathrm{R}(a_t, s_t, s_{t+1})$ is a reward obtained by an agent transitioning from s_t to s_{t+1} via a_t.

A policy $\pi_\theta : \mathcal{S} \rightarrow \mathcal{A}$ is a (possibly randomized) mapping (parameterized by $\theta \in \mathbb{R}^d$, e.g. weight of the neural network) from $\mathcal{S}$ to $\mathcal{A}$. Policy learning is the task of optimizing parameters θ of π_θ such that an agent applying it in the environment given by a fixed MDP maximizes total (expected/discounted) reward over given horizon H. In this paper we consider MDPs with finite horizons. We denote the trajectory of a policy as τ, defined as the sequence of states and actions taken during an episode, $\tau = \{s_0, a_0, r_0, \ldots s_H, a_H, r_H\}$.

3.2. Meta-Learning

When moving to meta-RL, we first define a task distribution $p(\mathcal{T})$. Here each task is an MDP as described. We denote $\mathcal{T}_i$ as a task sampled from $p(\mathcal{T})$, such that $\mathcal{T}_i = \{p(s_0), \mathrm{P}_i(s_{t+1}|s_t, a_t), \mathrm{R}_i(a_t, s_t, s_{t+1})\}$. As such, the loss function for a given task is generally of the following form:

$$\mathcal{L}_{\mathcal{T}_i} = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)]$$

Meta-learning considers the problem of *fast adaptation*. In MAML this means that we no longer consider a loss function with respect to the parameters θ, but those θ_i', where:

$$\theta_i' \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\theta)$$

for some learning rate α. Thus, the meta-RL loss function is as follows:

$$\mathcal{L}(\theta) = \sum_{\mathcal{T}_i \sim \mathcal{T}} \mathcal{L}_{\mathcal{T}_i}(\theta_i')$$

in other words, we consider the loss on each task after taking one gradient step on the task. In practice, we also consider increasing beyond a single gradient step. Note the crucial component here, that the tasks $\mathcal{T}_i$ are sampled iid from $p(\mathcal{T})$. This means that all existing approaches can only work when trained and tested on similar tasks, reducing the ability to learn multiple different tasks, as present in the real world.

4. Taming the Herd

We now present our method: Taming the Herd. We begin by training a single meta-learning policy across all tasks. This policy is referred to as the *probe* policy, $\pi_{\theta_{\text{probe}}}$. We then use this policy to gain experience from each task, used to form a task embedding, which we denote $\phi(\mathcal{T}_i)$. Once we have the embeddings (we use $\phi(\mathcal{T})$ to refer to the embeddings for all tasks), we cluster the tasks into K clusters C_K, which we refer to as $\mathrm{Cluster}$. We typically used hierarchical clustering for our experiments, but other methods could be considered. Once we have the K clusters, we train a population of K meta-learning policies $\{\theta_k\}_{k \in K}$ on each cluster of tasks in C_k. We refer to this procedure as

$\mathrm{MetaLearn}$, and this could be any meta-learning algorithm, but in practice we apply our approach to MAML (Finn et al., 2017) and PEARL (Rakelly et al., 2019). The full meta-train procedure is shown in Algorithm 1.

Algorithm 1 Meta-Train

Initialize: Train task distribution $p(\mathcal{T}_{train})$, parameters θ.
1. **Train Probing Policy:**θ_{probe}
2. **Embed Training Tasks:** $\phi(\mathcal{T}) = \{\phi(\mathcal{T}_i)\}_{\mathcal{T}_i \sim \mathcal{T}}$
3. **Cluster Tasks:** $C_K = \mathrm{Cluster}(\phi(\mathcal{T}))$.
4. **Train Herd:** $\theta_i = \mathrm{MetaLearn}(C_i)$

At test time, we first use the probe policy to gather sufficient experience (typically one episode). We then use this data to match the task with the corresponding cluster. At this point, we can use the remaining meta-test gradient steps to update the corresponding member of the population on the task. The full meta-test procedure is shown in Algorithm 2.

Algorithm 2 Meta-Test

Input: Test task distribution $p(\mathcal{T}_{test})$, clisters C_K, corresponding policies $\{\theta_i\}_{i=1}^K$
1. **Embed Test Task:** Generate one rollout with $\phi_{\theta_{\text{probe}}}$, and let $\phi(\mathrm{T_i})$ be its embedding.
2. **Assign to Task Cluster:** Select $C_i = \arg\min_i d(c_i, \phi(\mathrm{T_i}))$ where c_i is the centroid of the ith cluster. We now use the corresponding policy θ_i
3. **Meta-Test:** $\mathrm{MetaTest}(\theta_i, \mathcal{T}_{test})$.

When applied to MAML we refer to our method as a Herd of MAMLs (HoM), while for PEARL, we introduce a String of PEARLs (SoP). Next we go into more detail on each component of our method.

4.1. Probe policies

A crucial part of the algorithm is the choice of the probe policy $\pi_{\theta_{\text{probe}}}$. We wish to obtain task embeddings by rolling out $\pi_{\theta_{\text{probe}}}$, such that modes of tasks are clustered together. That means a) very similar tasks should be clustered together and b) if there is a task B such that success on A is only possible if jointly trained with B, ϕ_A and ϕ_B are also close together. It is important that the probe policy can quickly learn behaviors which can then be used to distinguish between tasks. We focus on a simple multi-task probe policy principle any multi-task method, and use the baseline single version of the MAML or PEARL algorithm, but other multi-task methods could be considered (Teh et al., 2017). We empirically found randomly initialized policy-networks to be insufficient. The design of specific acquisition functions for task inference is left to exciting future work.

4.2. Task Embeddings

Depending on the choice of $\pi_{\theta_{\text{probe}}}$ there are a multitude of options to find an embedding ϕ of task $\mathcal{T}_i$. We propose several here, with discovery of others left to future work. For Taming the Herd, we embed each transition individually and average the transition embeddings along a trajectory to get a trajectory embedding. We propose the following approaches to embed $\{s_t, a_t, r_t, s_{t+1}\}$:

1. The Fisher information matrix (FIM, defined in the Appendix Sec. A) can be seen as the curvature of the parameter-space. Thus we consider a **reward weighted FIM** embedding:

$$\phi(\mathcal{T}) = r_t \left[\nabla_\theta \log \pi_\theta(\tilde{a}_t|s_t) \nabla_\theta \log \pi_\theta(\tilde{a}_t|s_t)^T\right],$$

   where $\tilde{a}_t \sim \pi_\theta(\cdot|s_t)$, as we want the Fisher and not the empirical Fisher.

2. a **reward weighted state-next-state** embedding as product of reward and the outer product of concatenated state - next state tuples:

$$\phi(\mathcal{T}) = r_t \left[s_t, s_{t+1}\right] \left[s_t, s_{t+1}\right]^T$$

3. When using a PEARL as probe policy a particular natural choice of task-embedding is the **inferred latent context** of the exploration trajectories of task $\mathcal{T}_i$

4. A **reward weighted gradient** embedding is:

$$\phi(\mathcal{T}) = r_t \nabla_\theta \pi_\theta(a_t|s_t)$$

We found empirically that normalizing the rewards across the observed trajectories of each task works best, thus we use $r_t = r_t / \max_t r_t$. We focus on the reward weighted FIM embedding for our experiments, and plot obtained task embeddings in the appendix.

4.3. Number of clusters

One of the key choices of our algorithm is the number of clusters K we assign the tasks to. Given N tasks any number in the range $[1, N]$ is a candidate. Choosing $K = 1$ recovers the basic case to train one meta-learner on all test tasks, allowing for positive transfer between all tasks but at the expensive of possibly one task hindering the others. Choosing $K = N$ is the other extreme, here one meta-learner is trained for each task, allowing for no knowledge transfer between the tasks. While this might be optimal for very different tasks, most real world tasks share some common struture where the seperate training prevents the exchange of information. The goal of the task embeddings presented in 4.2 is to embed tasks in a way such that similar tasks are close to each other.

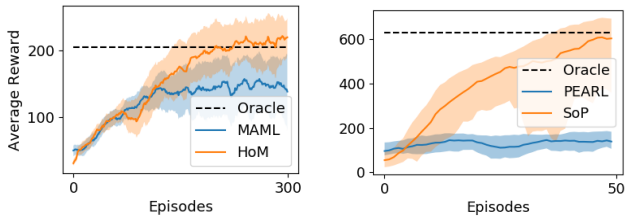
5. Experiments

The primary goal of our experiments it to evaluate whether our approach to meta-learning can improve performance on multi-modal task distributions. This differs from existing works, which only consider variations of the same task.

We create a multi-modal distribution by taking the union of two common meta-RL problems: $\{\text{Cheetah:Forward}, \text{Cheetah:Backward}\}$ and $\{\text{Walker:Forward}, \text{Walker:Backward}\}$. These two sets of tasks correspond to different agents (a Cheetah and a Walker), which get a positive reward walking in different directions (forward or backward). We call this set of tasks ML2. We focus our investigation on two questions:

1. Is it beneficial to train a population of meta-learners instead of a single one?
2. Are the task embeddings from Section 4.2 suitable proxies to the task-assignment-oracle?

Our implementation is based on (garage contributors, 2019). We use the default the default hyperparamaters for MAML, and the default for PEARL from the original paper (Rakelly et al., 2019). We include these in the Appendix.



(a) MAML (b) PEARL

Figure 2. Comparison of our approach (orange) vs. an individual meta-learner (blue) and the oracle which is aware of the task distribution (black, dotted). Curves show the mean $\pm$ 1 std.

To ensure a fair comparison of the single meta-learner and the herd we make sure that the single meta-learner has at least as many environment steps at test and train time as all the ensemble members together. We now describe the training protocol in detail for MAML on ML2. At train time the probe-policy is trained for x episodes for a total of y steps per task. Subsequently each member of the herd is trained on z samples form task i, meaning the single meta-learner is trained on $i \times N$ total environment interactions. This means the evaluation protocol differs. If fewer environment steps than the total number of probe-policy training steps have been taken, we are still in the single task case. At test time the single meta-learner uses l exploration trajectories to do one-shot learning on top of it. The herd approach can use only $l - 2$ adaptation trajectories as we dedicate a sample-budget of two trajectories to embed the test-task with the probe-policy. For PEARL we embed the task at test-time with only one trajectory and use 1 exploration trajectory.

We show the training curved in Fig. 2, where we see that in both cases, our approach achieves dramatically better performance than the baseline of a single agent. In Table 1 & 2 we show the median best performance achieved for the MAML and PEARL experiments respectively. For MAML, we see that a single MAML outperforms on the HalfCheetah task, but it is significantly worse on Walker. Interestingly, we see larger gains for PEARL than MAML.

Table 1. ML2 performance breakdown for the MAML setting. We show the median of the best reward achieved for each of 10 seeds.

	Oracle	Single	HoM
HalfCheetah	221.6	236.6	221.2
Walker	298.5	229.7	298.5

Table 2. ML2 performance breakdown for the PEARL setting. We show the median of the best reward achieved for each of 10 seeds.

	Oracle	Single	SoP
HalfCheetah	1004.5	447.7	1004.5
Walker	414.9	507.6	414.9

6. Discussion

We presented *Taming the Herd*, a new approach for multi-modal meta-learning using a population of agents. Our results show significant promise, dramatically outperforming the baseline of a single agent, for two different popular meta-learning algorithms, MAML and PEARL. We believe our approach achieves positive forward and backward transfer. For forward transfer, the results in this work demonstrate the performance in few shot learning. However, potentially more interesting is the concept of backward transfer. We believe adding more tasks may improve the meta learned representations for the corresponding clusters.

We believe our method could be made more efficient by utilizing the trajectories obtained by the probe policy not only for task-inference but also for meta training. Advances in this could be applied also to PEARL which does also only task-inference at test-time without adapting the learner's parameters. We can also consider measuring distances between probabilistic embeddings, for example using Wasserstein Distances (Pacchiano et al., 2020).

We could consider training the probe policy in a way to maximize information about the task, such that it explicitly seeks to discover its' mode. Measures employing active learning have recently achieved success in RL (Ball et al., 2020; Pathak et al., 2019), and may be useful here.

7. Acknowledgement

RM was supported by appliedAI.

References

Achille, A., Lam, M., Tewari, R., Ravichandran, A., Maji, S., Fowlkes, C. C., Soatto, S., and Perona, P. (2019). Task2vec: Task embedding for meta-learning. *CoRR*, abs/1902.03545.

Antoniou, A., Edwards, H., and Storkey, A. (2019). How to train your MAML. In *International Conference on Learning Representations*.

Ball, P., Parker-Holder, J., Pacchiano, A., Choromanski, K., and Roberts, S. (2020). Ready policy one: World building through active learning. *accepted to ICML 2020*.

Behl, H. S., Baydin, A. G., and Torr, P. H. S. (2019). Alpha MAML: adaptive model-agnostic meta-learning. *CoRR*, abs/1905.07435.

Bellman, R. (1957). A Markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 1126–1135, International Convention Centre, Sydney, Australia.

garage contributors, T. (2019). Garage: A toolkit for reproducible reinforcement learning research. https://github.com/rlworkgroup/garage.

Kaushik, R., Anne, T., and Mouret, J. (2020). Fast online adaptation in robotics through meta-learning embeddings of simulated priors. *CoRR*, abs/2003.04663.

Naik, D. K. and Mammone, R. J. (1992). Meta-neural networks that learn by learning. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 1, pages 437–442 vol.1.

Pacchiano, A., Parker-Holder, J., Tang, Y., Choromanska, A., Choromanski, K., and Jordan, M. I. (2020). Learning to score behaviors for guided policy optimization. In *accepted to ICML 2020*.

Pathak, D., Gandhi, D., and Gupta, A. (2019). Self-supervised exploration via disagreement. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 5062–5071.

Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. (2020). Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *International Conference on Learning Representations*.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. (2019). Meta-learning with implicit gradients. In Wallach, H., Larochelle, H., Beygelzimer, A., dAlché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 113–124. Curran Associates, Inc.

Rakelly, K., Zhou, A., Finn, C., Levine, S., and Quillen, D. (2019). Efficient off-policy meta-reinforcement learning via probabilistic context variables. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5331–5340, Long Beach, California, USA. PMLR.

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. (2019). Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*.

Schmidhuber, J. (1987). Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook.

Teh, Y., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. (2017). Distral: Robust multitask reinforcement learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4496–4506. Curran Associates, Inc.

Vu, T., Wang, T., Munkhdalai, T., Sordoni, A., Trischler, A., Mattarella-Micke, A., Maji, S., and Iyyer, M. (2020). Exploring and predicting transferability across nlp tasks.

Zintgraf, L., Shiarli, K., Kurin, V., Hofmann, K., and Whiteson, S. (2019). Fast context adaptation via meta-learning. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 7693–7702.

Zintgraf, L., Shiarlis, K., Igl, M., Schulze, S., Gal, Y., Hofmann, K., and Whiteson, S. (2020). Varibad: A very good method for bayes-adaptive deep rl via meta-learning. In *International Conference on Learning Representations*.

A. Additional Definitions

A.1. Fisher Information Matrix

The *Fisher information matrix* (FIM) is defined as:

$$F = \mathop{\mathbb{E}}_{x\sim p(x), y\sim p_w(y|x)} \left[\nabla_w \log p_w(y|x) \nabla_w \log p_w(y|x)^T\right]$$

which is the expected covariance of the gradients of the log-likelihoods w.r.t the model parameters. The information content of each weight of a neural network can be computed by the divergence of the original and a perturbed output distribution for $w^{'} = w + \delta w$ in second order approximation as: $\mathbb{E}_{x\sim p}\, KL(p_{w'}(y|x); p_w(y|x)) = \delta w \dot{F} \delta w + o(\delta w^2)$. The FIM can be seen as the curvature of the model in the parameter-space.

B. Implementation Details

We used hyperparameters from open source repos. For MAML we used garage (garage contributors, 2019) and for PEARL we used the author's implementation (Rakelly et al., 2019). For completeness we list these below.

B.1. Hyperparameters MAML + TRPO

Table 3. Hyperparameters used for MAML + TRPO.

parameter	value
rollouts per task	10
max path length	100
discount	0.99
gae lambda	1
inner lr	0.1
num grad$_u pdates$	1
epochs	40
n exploration traj (ensemble)	8
n exploration traj (single maml)	10
policy hidden sizes	(64, 64)
policy hidden nonlinearity	torch.tanh
policy output nonlinearity	None
value hidden sizes	[32, 32]
value hidden nonlinearity	torch.tanh
value output nonlinearity	None)
inner lr	Default(1e-2)
outer lr	1e-3
ax kl step	0.01
center adv	True
positive adv	False
policy ent coeff	0.0
use softplus entropy	False
stop entropy gradient	False
entropy method	no$_e ntropy$

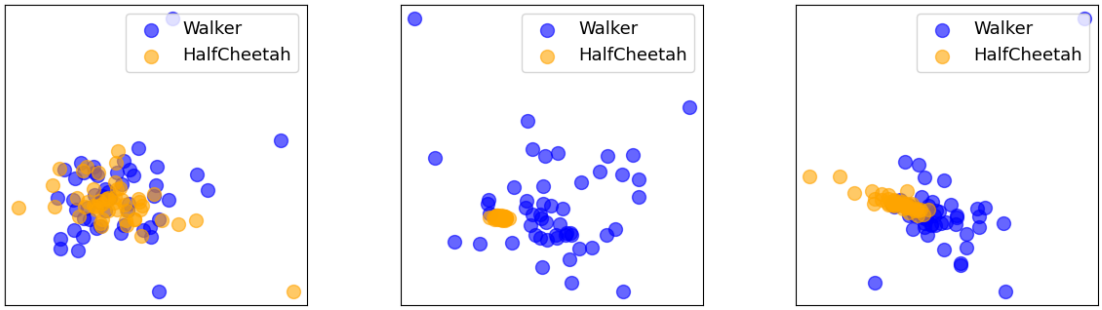
Table 4. Hyperparameters used for PEARL.

parameter	value
latent size	5
encoder hidden size	200
net size	300
meta batch size	4
num steps per epoch	2000
num initial steps	2000
num tasks sample	10
num steps prior	1000
num extra rl steps posterior	1000
batch size	256
embedding batch size	256
embedding mini batch size	256
max path length	200
reward scale	5.
n exploration traj	2
n exploration traj (ensemble)	1
n exploration traj (single maml)	2
num$_t asks_s ample$	5 len(task$_u niverse$)

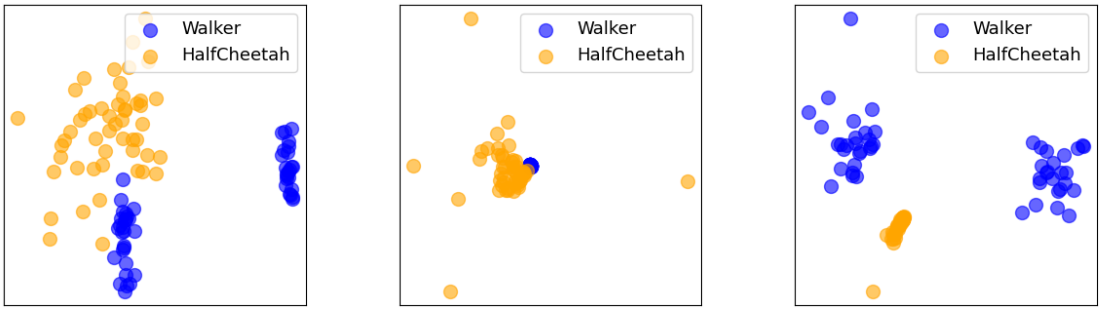
B.2. Hyperparameters PEARL

C. Additional Experimental Results

C.1. Plot of all embeddings

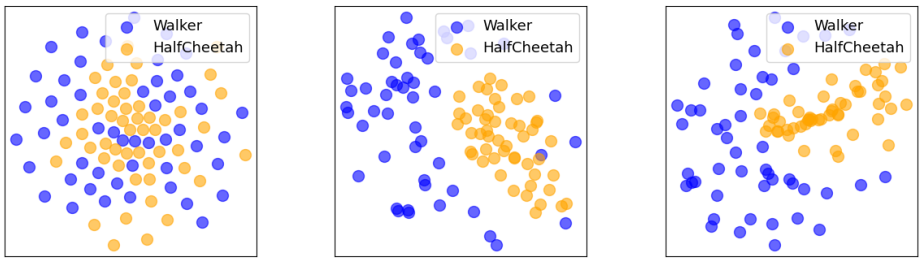


(a) MAML-reward-grad (b) MAML-reward-state (c) MAML-Fisher



(d) PEARL-latent (e) PEARL-reward-state (f) PEARL-Fisher

Figure 3. 2-dimensional PCA of different task embeddings or 50 tasks sampled in each mode after 1 training episode of the probe policy on PEARL and 10 training episodes on MAML.



(a) MAML-reward-grad (b) MAML-reward-state (c) MAML-Fisher

(d) PEARL-latent (e) PEARL-reward-state (f) PEARL-Fisher

Figure 4. 2-dimensional TSNE of different task embeddings or 50 tasks sampled in each mode after 1 training episode of the probe policy on PEARL and 10 training episodes on MAML.