DEEP ONLINE LEARNING VIA META-LEARNING: CONTINUAL ADAPTATION FOR MODEL-BASED RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Humans and animals can learn complex predictive models that allow them to accurately and reliably reason about real-world phenomena, and they can adapt such models extremely quickly in the face of unexpected changes. Deep neural network models allow us to represent very complex functions, but lack this capacity for rapid online adaptation. The goal in this paper is to develop a method for continual online learning from an incoming stream of data, using deep neural network models. We formulate an online learning procedure that uses stochastic gradient descent to update model parameters, and an expectation maximization algorithm with a Chinese restaurant process prior to develop and maintain a mixture of models to handle non-stationary task distributions. This allows for all models to be adapted as necessary, with new models instantiated for task changes and old models recalled when previously seen tasks are encountered again. Furthermore, we observe that meta-learning can be used to meta-train a model such that this direct online adaptation with SGD is effective, which is otherwise not the case for large function approximators. We apply our method to model-based reinforcement learning, where adapting the predictive model is critical for control; we demonstrate that our online learning via meta-learning algorithm outperforms alternative prior methods, and enables effective continuous adaptation in non-stationary task distributions such as varying terrains, motor failures, and unexpected disturbances. Videos available at:`https://sites.google.com/view/onlineanon`

1 INTRODUCTION

Human and animal learning is characterized not just by a capacity to acquire complex skills, but also the ability to adapt rapidly when those skills must be carried out under new or changing conditions. For example, animals can quickly adapt to walking and running on different surfaces (Herman, 2017) and humans can easily modulate force during reaching movements in the presence of unexpected perturbations (Flanagan & Wing, 1993). Furthermore, these experiences are remembered, and can be recalled to adapt more quickly when similar disturbances occur in the future (Doyon & Benali, 2005). Since learning entirely new models on such short time-scales is impractical, we can devise algorithms that explicitly train models to adapt quickly from small amounts of data. Such online adaptation is crucial for intelligent systems operating in the real world, where changing factors and unexpected perturbations are the norm. In this paper, we propose an algorithm for fast and continuous online learning that utilizes deep neural network models to build and maintain a task distribution, allowing for the natural development of both generalization as well as task specialization.

Our working example is continuous adaptation in the model-based reinforcement learning setting, though our approach generally addresses any online learning scenario with streaming data. We assume that each "trial" consists of multiple tasks, and that the delineation between the tasks is not provided explicitly to the learner – instead, the method must adaptively decide what "tasks" even represent, when to instantiate new tasks, and when to continue updating old ones. For example, a robot running over changing terrain might need to handle uphill and downhill slopes, and might choose to maintain separate models that become specialized to each slope, adapting to each one in turn based on the currently inferred surface.

We perform adaptation simply by using online stochastic gradient descent (SGD) on the model parameters, while maintaining a mixture model over model parameters for different tasks. The mixture is updated via the Chinese restaurant process (Stimberg et al., 2012), which enables new

tasks to be instantiated as needed over the course of a trial. Although online learning is perhaps one of the oldest applications of SGD (Bottou, 1998), modern parametric models such as deep neural networks are exceedingly difficult to train online with this method. They typically require medium-sized minibatches and multiple epochs to arrive at sensible solutions, which is not suitable when receiving data in an online streaming setting. One of our key observations is that meta-learning can be used to learn a prior initialization for the parameters that makes such direct online adaptation feasible, with only a handful of gradient steps. The meta-training procedure we use is based on model-agnostic meta-learning (MAML) (Finn et al., 2017), where a prior weight initialization is learned for a model so as to optimize improvement on any task from a meta-training task distribution after a small number of gradient steps.

Meta-learning and MAML have previously been extended to model-based RL (Clavera et al., 2018), but only for the k-shot adaptation setting: The meta-learned prior model is adapted to the k most recent time steps, but the adaptation is not carried forward in time (i.e., adaptation is always performed from the prior itself). This rigid batch-mode setting is restrictive in an online learning setup and is insufficient for tasks that are further outside of the training distribution. A more natural formulation is one where the model receives a continuous stream of data and must adapt online to a potentially non-stationary task distribution. This requires both fast adaptation and the ability to recall prior tasks, as well as an effective adaptation strategy to interpolate as needed between the two.

The primary contribution of this paper is an online learning algorithm that uses expectation maximization, in conjunction with a Chinese restaurant process prior on the task distribution, to learn mixtures of neural network models that are updated with simple online SGD. In contrast to prior multi-task and meta-learning methods, our method's online assignment of soft task probabilities allows for task specialization to emerge naturally, without requiring task delineations to be specified in advance. We evaluate our approach in the context of model-based RL on a suite of challenging simulated robotic tasks including disturbances, environmental changes, and simulated motor failures. Our simulated experiments show a half-cheetah agent and a hexapedal crawler robot performing continuous model adaptation in an online setting. Our results show online instantiation of new tasks, the ability to adapt to out-of-distribution tasks, and the ability to recognize and revert back to prior tasks. Additionally, we demonstrate that our method outperforms a state-of-the-art prior method that does k-shot model-based meta-RL, as well as natural baselines such as naïve continuous adaptation and online learning without meta-training.

2 Related Work

Online learning is one of the oldest subfields of machine learning (Bottou, 1998; Jafari et al., 2001). Prior algorithms have used online gradient updates (Duchi et al., 2011) and probabilistic filtering formulations (Murphy, 2002; Hoffman et al., 2010; Broderick et al., 2013). In principle, commonly used gradient-based learning methods, such as SGD, can easily be used as online learning algorithms (Bottou, 1998). In practice, their performance with deep neural network function approximators is limited (Sahoo et al., 2017): such high-dimensional models must be trained with batch-mode methods, minibatches, and multiple passes over the data. We aim to lift this restriction by using model-agnostic meta-learning (MAML) to explicitly pretrain a model that enables fast adaptation, which we then use for continuous online adaptation via an expectation maximization algorithm with a Chinese restaurant process (Blei et al., 2003) prior for dynamic allocation of new tasks in a dynamic task distribution.

Online learning is related to that of continual or lifelong learning (Thrun, 1998), where the agent faces a non-stationary distribution of tasks over time. However, unlike works that focus on avoiding negative transfer, i.e. catastrophic forgetting (Kirkpatrick et al., 2017; Rebuffi et al., 2017; Zenke et al., 2017; Lopez-Paz et al., 2017; Nguyen et al., 2017), online learning focuses on the ability to rapidly learn and adapt in the presence of non-stationarity. While some continual learning works consider the problem of forward transfer, e.g. Rusu et al. (2016); Aljundi et al. (2017); Wang et al. (2017), these works and others in continual learning generally focus on small sets of tasks where fast, online learning is not realistically possible, since there are simply not enough tasks to recover structure that enables fast, few-shot learning in new tasks or environments.

Our approach builds on techniques for meta-learning or learning-to-learn (Thrun & Pratt, 1998; Schmidhuber, 1987; Bengio et al., 1992; Naik & Mammone, 1992). However, most recent meta-learning work considers batch-mode updates where one task is learned at a time from a batch of

data (Santoro et al., 2016; Ravi & Larochelle, 2017; Munkhdalai & Yu, 2017; Wang et al., 2016; Duan et al., 2016). In our work, we do not assume that task boundaries are known and specifically address non-stationary task distributions. Concurrent work (Anonymous, 2019) has used a mixture model at the level of the prior parameters, considering non-stationarity at the meta-level, whereas we use a mixture model to model the task-specific parameters to perform online learning only during meta-testing. Other meta-learning works have considered non-stationarity within a task (Al-Shedivat et al., 2017) and episodes involving multiple tasks at meta-test time (Ritter et al., 2018), but do not consider continual online adaptation with unknown task separation. Prior work has also studied meta-learning for model-based RL (Clavera et al., 2018). This prior method updates the model every time step, but each update is a batch-mode k-shot update, using exactly k prior transitions and resetting the model at each step. This allows for adaptive control, but does not enable continual online adaptation, since updates from previous steps are always discarded. In our comparisons, we find that our approach substantially outperforms this prior method. To our knowledge, our work is the first to apply meta-learning to learn streaming online updates.

3 Problem Statement

We formalize our online learning problem setting as follows: at each time step, the model receives an input $\mathbf{x}_t$ and produces a prediction $\hat{\mathbf{y}}_t$. It then receives a ground truth label $\mathbf{y}_t$, which must be used to adapt the model to increase its prediction accuracy on the next input $\mathbf{x}_{t+1}$. The true labels are assumed to come from some task distribution $P(Y_t|X_t, T_t)$, where T_t is the task at time t. The tasks themselves change over time, resulting in a non-stationary task distribution, and the identity of the task T_t is unknown to the learner. In real-world settings, tasks might correspond to unknown parameters of the system (e.g., motor malfunction on a robot), user preferences, or other unexpected events. This problem statement covers a range of online learning problems that all require continual adaptation to streaming data and trading off generalization and specialization.

In our experiments, we use model-based RL as our working example, where the input $\mathbf{x}_t$ is a state-action pair, and the output $\mathbf{y}_t$ is the next state. We will discuss this in Section 6, while the derivation of our method below is for the general case of arbitrary online prediction problems.

4 Online Learning with a Mixture of Neural Networks

In this section, we explain our method which enables effective online learning using a continuous stream of incoming data from a non-stationary task distribution. We aim to retain generalization so as to not lose past knowledge, as well as gain specialization, which is particularly important for learning new tasks that are further out-of-distribution and require more learning. We discuss the process of obtaining a meta-learned prior in Sec. 5, but we first formulate in this section an online adaptation algorithm using SGD with expectation maximization to maintain and adapt a mixture model over task model parameters (i.e., a probabilistic task distribution).

4.1 Method Overview

Let $p_{\theta(T_t)}(\mathbf{y}_t|\mathbf{x}_t)$ represent the predictive distribution of our model on input $\mathbf{x}_t$, for a task T_t (which is unknown). Our goal is to estimate model parameters $\theta_t(T)$ for each task T in the non-stationary task distribution, which requires inferring the distribution over tasks at each step, using that distribution to make predictions, and also using it to update each model. In practice, the parameters of each model will corresponds to the weights in a neural network.

Each model begins from some prior parameter vector θ^*, which we will discuss in more detail in Section 5. Since the number of tasks is also unknown, we begin with one task at time step 0, where $\theta_0(T) = \{\theta^*\}$ and $|T| = 1$. From here, we continuously update the parameters $\theta_t(T_t)$ to model the true process $P(Y_t|X_t, T_t)$, and add new tasks as needed. Since task identities are unknown, we must also estimate $P(T_t)$ at each time step. Thus, the online learning problem consists of adapting $\theta_t(T)$ at each time step according to the inferred task probabilities $P(T_t)$. Since the task probabilities are unknown, we adapt the expectation maximization (EM) algorithm and optimize the expected log-likelihood, given by

$$\mathcal{L} = E_{T_t \sim P(T_t|\mathbf{x}_t, \mathbf{y}_t)}[\log p_{\theta_t(T)}(\mathbf{y}_t|\mathbf{x}_t)], \tag{1}$$

where we use $\theta_t(T)$ to denote the model parameters corresponding to task T. To handle the unknown number of tasks, we employ the Chinese restaurant process to instantiate new tasks as needed.

4.2 APPROXIMATE ONLINE INFERENCE

We use expectation maximization (EM) to update the model parameters. In our case, the E step in EM involves estimating the task distribution at the current time step, $P(T_t)$, while the M step involves updating the model parameters θ_t to obtain the new model parameters θ_{t+1}. The parameters are always updated by one gradient step per time step, according to the inferred task responsibilities.

We first estimate the expectations over all $|T|$ parameters in the task distribution. The posterior of each task probability $P(T_t = T|\mathbf{x}_t, \mathbf{y}_t)$ can be written as follows:

$$P(T_t = T|\mathbf{x}_t, \mathbf{y}_t) \propto p_{\theta(T)}(\mathbf{y}_t|\mathbf{x}_t, T_t = T)P(T_t = T). \tag{2}$$

We then formulate the task prior $P(T_t)$ using a Chinese restaurant process (CRP) to enable new tasks to be instantiated during a trial. The CRP is an instantiation of a Dirichlet process. In the CRP, at time t, the probability of each task T should be given by

$$P(T_t = T) = \frac{n_T}{t - 1 + \alpha} \tag{3}$$

where n_T is the expected number of datapoints in task T for all steps $1, \ldots, t-1$, and α is a hyperparameter that controls the instantiation of new tasks. The prior therefore becomes

$$P(T_t = T) = \frac{\sum_{t'=1}^{t-1} P(T_{t'} = T)}{t - 1 + \alpha} \quad \text{and} \quad P(T_t = \text{new}) = \frac{\alpha}{t - 1 + \alpha} \tag{4}$$

Combining the prior and likelihood, we derive the following posterior task probability distribution:

$$P(T_t = T|\mathbf{x}_t, \mathbf{y}_t) \propto p_{\theta(T)}(\mathbf{y}_t|\mathbf{x}_t, T_t = T) \left[\sum_{t'=1}^{t-1} P(T_{t'} = T) + \delta(T \text{ is new})\alpha \right] \tag{5}$$

Having estimated $P(T_t = T|\mathbf{x}_t, \mathbf{y}_t)$, we perform the M step, which improves the expected log-likelihood in Equation 1 based on the inferred task distributions. Since each task starts from the prior parameters θ^*, the value of $\theta_t(T)$ after taking one gradient update at each time step is given by

$$\theta_{t+1}(T) = \theta^* - \beta \sum_{t'=0}^{t} P_t(T_{t'} = T|\mathbf{x}_{t'}, \mathbf{y}_{t'}) \nabla_{\theta_{t'}(T)} \log p_{\theta_{t'}(T)}(\mathbf{y}_{t'}|\mathbf{x}_{t'}) \quad \forall\, T \tag{6}$$

If we assume that $\theta_t(T)$ has already been updated for all previous time steps $t-1, \ldots, 0$, we can approximate this update by simply updating the previous parameters $\theta_{t-1}(T)$ on the current sample:

$$\theta_{t+1}(T) = \theta_t(T) - \beta P_t(T_t = T|\mathbf{x}_t, \mathbf{y}_t) \nabla_{\theta_t(T)} \log p_{\theta_t(T)}(\mathbf{y}_t|\mathbf{x}_t) \quad \forall\, T \tag{7}$$

This procedure is an approximation, since updates to task parameters $\theta_t(T)$ will in reality change the task probabilities at previous time steps. However, this approximation removes the need to store previously seen data points and yields a fully online, streaming algorithm. Finally, to fully implement the EM algorithm, we must alternate the E and M steps to convergence, rolling back the previous gradient update to $\theta_t(T)$ at each iteration. In practice, we found it sufficient to perform the E and M steps only once per time step. While this is a crude simplification, successive time steps in the online learning scenario are likely to be correlated, making this procedure reasonable. However, it is also straightforward to perform multiple steps of EM while still remaining fully online.

Algorithm 1 Online Learning with a Parametric Mixture of Neural Networks

Require: θ^* from meta-training
  Initialize $|T| = 1$, $t = 0$, $\theta_0(T) = \{\theta^*\}$
  for each time step t **do**
    Calculate $p_{\theta_t(T)}(\mathbf{y}_t|\mathbf{x}_t, T_t = T)$ for each T
    Calculate $P_t = P_t(T_t = T|\mathbf{x}_t, \mathbf{y}_t)$ for each T
    Calculate θ_{t+1} by adapting from θ_t for each T
    Calculate θ_{new} by adapting from θ^* using $\mathbf{x}_{t-1}, \mathbf{y}_{t-1}$

    if $p_{\theta_{new}(t)} > p_{\theta_t(T)} \;\; \forall T$ **then**
      Add θ_{new} to θ_{t+1}
      Recalculate P_t for all T, using θ_{t+1}
      Recalculate θ_{t+1} for all T, using updated P_t
    end if
    $T^* = \text{argmax}_T(p_{\theta_t(T)}(\mathbf{y}_t|\mathbf{x}_t, T_t = T))$
    Select $\theta_{\text{best}}(t) = \theta_{t+1}(T^*)$
    Perform prediction $\hat{\mathbf{y}}_t = p_{\theta_{\text{best}}(t)}(\mathbf{y}_t|\mathbf{x}_t)$
  end for

We can now summarize the complete algorithm, which is also outlined in Alg. 1. At the first time step $t = 0$, the task distribution is initialized

to contain one entry: $\theta_0(T) = \{\theta^*\}$ and $|T| = 1$. At every time step after that, an E step is performed to estimate the task distribution and an M step is performed to update the model parameters. The CRP prior also assigns, at each time step, a probability of adding a new task to the task distribution. The parameters of this new task start from the prior θ^* and, if a new task is instantiated, are adapted on the latest datapoint. The prediction on the next datapoint is then made using the model corresponding to the most likely task.

5 Online Learning via Meta-Learning

We formulated an algorithm above for performing online adaptation using continually incoming data. For this method, we choose to meta-train the prior using the model-agnostic meta-learning (MAML) algorithm. This meta-training algorithm is an appropriate choice, because it results in a prior that is specifically intended for gradient-based fine-tuning. Before we further discuss our choice in meta-training procedure, we first give an overview of MAML and meta-learning in general.

Given a distribution of tasks, a meta-learning algorithm produces a learning procedure, which can, in some cases, quickly adapt to a new task. MAML optimizes for an initialization of a deep network that achieves good few-shot task generalization when fine-tuned using a few datapoints from that task. At train time, MAML sees small amounts of data from large numbers of tasks, where data $\mathcal{D}_T$ from each task T can be split into training and validation subsets ($\mathcal{D}_T^{\text{tr}}$ and $\mathcal{D}_T^{\text{val}}$), where $\mathcal{D}_T^{\text{tr}}$ is of size k. MAML optimizes for model parameters θ such that one or more gradients steps on $\mathcal{D}_T^{\text{tr}}$ results in a minimal loss L on $\mathcal{D}_T^{\text{val}}$. In our case, we will set $\mathcal{D}_{T_t}^{\text{tr}} = (\mathbf{x}_t, \mathbf{y}_t)$ and $\mathcal{D}_{T_t}^{\text{val}} = (\mathbf{x}_{t+1}, \mathbf{y}_{t+1})$, and the loss L will correspond to negative log likelihood. A good θ that allows such adaptation to be successful across various meta-training tasks is thus a good network initialization from which adaptation can solve various new tasks that are related to the previously seen tasks. The MAML objective is defined as follows:

$$\min_\theta \sum_T L(\theta - \eta \nabla_\theta L(\theta, \mathcal{D}_T^{\text{tr}}), \mathcal{D}_T^{\text{val}}) = \min_\theta \sum_T L(\phi_T, \mathcal{D}_T^{\text{val}}). \tag{8}$$

Here, η is the inner learning rate. Once this meta-objective is optimized, the resulting θ^* acts as a prior from which fine-tuning can occur at test-time, using the k new points in $\mathcal{D}_{T_{\text{test}}}^{\text{tr}}$ as follows:

$$\phi_{T_{\text{test}}} = \theta^* - \eta \nabla_{\theta^*} L(\theta^*, \mathcal{D}_{T_{\text{test}}}^{\text{tr}}). \tag{9}$$

Here, $\phi_{T_{\text{test}}}$ is adapted from the meta-learned prior θ^* to be more representative for the current time.

Although Finn et al. (2017) demonstrated this fast adaptation of deep neural networks and Clavera et al. (2018) extended this framework to model-based meta RL, these methods address adaptation in the k-shot setting, always adapting directly from the meta-learned prior and not allowing further adaptation or specialization. In this work, we have extended these capabilities by enabling more evolution of knowledge through a temporally-extended online adaptation procedure.

While our procedure for continual online learning is still initialized with this meta-training for k-shot adaptation (i.e., MAML), we found that this prior was sufficient to enable effective continual online adaptation at test time. The intuitive rationale for this is that MAML trains the model to be able to change significantly using only a small number of datapoints and gradient steps. Note that this meta-trained prior can be used at test time in a k-shot setting, similar to how it was trained, or it can be used at test time by taking substantially more gradient steps away from this prior. We show that our method outperforms both of these methods in Sec. 7, but the mere ability to use this meta-learned prior in these ways makes the use of MAML enticing.

We note that it is quite possible to modify the MAML algorithm to optimize the model directly with respect to the weighted updates discussed in Section 4.2. This simply requires computing the task weights (the E step) on each batch during meta-training, and then constructing a computation graph where all gradient updates are multiplied by their respective weights. Standard automatic differentiation software can then compute the corresponding meta-gradient. For short trial lengths, this is not substantially more complex than standard MAML, and for longer trial lengths, truncated backpropagation is an option. Although such a meta-training procedure better matches the way that the model is used during online adaptation, we found that it did not substantially improve our results. While it's possible that the difference might be more significant if meta-training for longer-term adaptation, this observation does suggest that simply meta-training with MAML is sufficient

for enabling effective continuous online adaptation in non-stationary multi-task settings. To clarify, although this modified training procedure of incorporating the EM weight updates (during meta-training) did not explicitly improve our results, we see that test-time performance did indeed improve with more data used for the standard MAML meta-training procedure (see Appendix).

6 APPLICATION TO MODEL-BASED RL

In our experiments, we apply our online learning via meta-learning method to model-based RL. RL in general aims to act in a way that maximizes the sum of future rewards. At each time step t, the agent executes action $\mathbf{a}_t \in A$ from state $\mathbf{s}_t \in S$, transitions to the next state $\mathbf{s}_{t+1}$ according the transition probabilities $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ and receives rewards $r_t = r(\mathbf{s}_t, \mathbf{a}_t)$. The goal at each step is to execute the action $\mathbf{a}_t$ that maximizes the discounted sum of future rewards $\sum_{t'=t}^{\inf} \gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}_{t'})$, where $\gamma \in [0, 1]$ is a discount factor that prioritizes near-term rewards. In model-based RL, in particular, the predictions from a (known or learned) dynamics model are used to either learn a policy, or are used directly inside a planning algorithm to plan the actions that produce the largest reward. We use the latter approach in our implementation, planning over a sequence of future actions $\mathbf{a}_0, \ldots, \mathbf{a}_H$ to maximize the future reward. The planning procedure is based on stochastic optimization, following prior work (Clavera et al., 2018).

To instantiate our method in a model-based RL context, we set the input $\mathbf{x}_t$ to be the concatenation of K previous states and actions, given by $\mathbf{x}_t = [\mathbf{s}_{t-1}, \mathbf{a}_{t-1}, \mathbf{s}_{t-2}, \mathbf{a}_{t-2}, \ldots, \mathbf{s}_{t-K}, \mathbf{a}_{t-K}]$, and the output to be the corresponding next states $\mathbf{y}_t = [\mathbf{s}_t, \ldots, \mathbf{s}_{t-K+1}]$. This provides us with a slightly larger batch of data for each online update, since individual time steps at high frequency can be very noisy, and using the past K transitions helps to damp out the updates. The underlying predictive model represents each transition as an independent Gaussian, such that $p_\theta(\mathbf{y}_t|\mathbf{x}_t) = \prod_{i=1}^{K} \mathcal{N}(\mathbf{s}_{t-i+1}; f_\theta(\mathbf{s}_{t-i}, \mathbf{a}_{t-i}), \sigma^2)$, where σ^2 is a constant. To perform control, the model with the highest task probability at the current time step is used for planning.

7 EXPERIMENTS

The questions that we aimed to study from our experiments include: Can our method 1) autonomously discover some task structure amid a stream of non-stationary data?, 2) adapt to tasks that are further outside of the task distribution than can be handled by a k-shot learning approach?, 3) recognize and revert to tasks it has seen before?, 4) avoid overfitting to a recent task to prevent deterioration of performance upon the next task switch?, and 5) outperform other methods?

To study these questions, we conduct experiments on agents in the MuJoCo physics engine (Todorov et al., 2012). The agents we used are a half-cheetah (S R21, A R6) and a hexapedal crawler (S R50, A R12). Using these agents, we design a number of challenging online learning problems that involve multiple sudden and gradual changes in the underlying task distribution, including tasks that are extrapolated from those seen previously, where online learning is criticial. Through these experiments, we aim to build problem settings that are representative of the types of disturbances and shifts that a real RL agent might encounter. We implement our dynamics model $f_\theta(s_t, a_t)$ as a neural network model with three hidden layers each of dimension 500, and ReLU nonlinearities. Our MPC controller selects the action that leads to highest predicted reward over some horizon H.

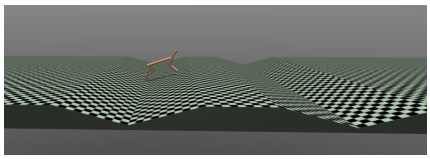

Figure 1: Half-cheetah agent, shown traversing a landscape with 'basins'

We compare to several alternative methods. We consider two approaches that still leverage meta-training but either (a) always adapt from the meta-trained prior θ^* (**TS**) as typically done with meta-learning methods (Clavera et al., 2018), or (b) always take gradient steps from the previous parameter (**GS**). The latter version oftens overfits to recently observed tasks, so it should indicate the importance of our method effectively identifying task structure to avoid overfitting and enable recall. We also compare to two more standard approaches that do not use meta-learning. The first is training a model on all of the meta-training data that is fixed throughout the trials (**MB**), and the second is using that same model, but adapting it online using gradient-descent at each timestep (**DE**), which is representative

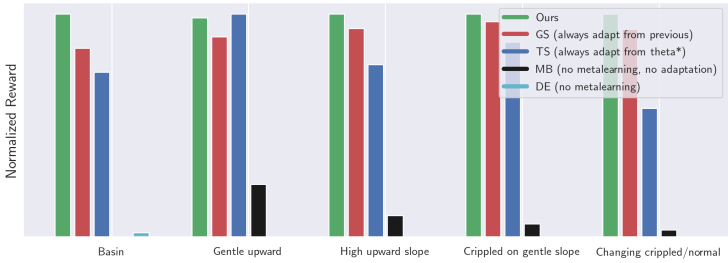

Figure 2: Results on half-cheetah landscape traversal. The MB and DE methods indicate that a single model is not sufficient for effectively handling this task and online learning is critical. The meta-learning approaches perform similarly; however, since these trials involve terrain and other physics changes that is extrapolated from the data seen previously, it is important to take multiple gradient steps during online learning.

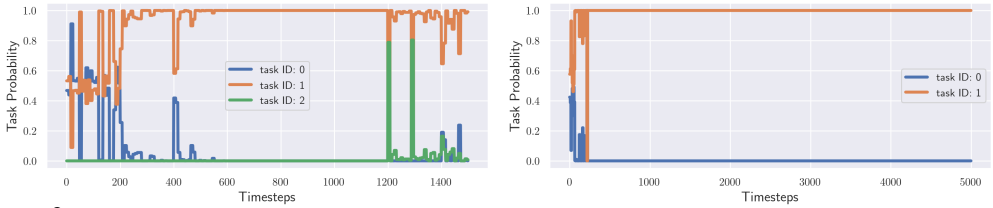

Figure 3: Latent task distribution over time for two different half-cheetah landscape traversal tasks, where the encountered terrain slopes vary within each run. Interestingly, we find that our method chooses to only use one latent task variable to describe the varying terrain.

of commonly used dynamic evaluation methods (Rei, 2015; Krause et al., 2017; 2016; Fortunato et al., 2017). In following three sections, we present results and analysis of our findings.

7.1 Terrain Slopes on Half-Cheetah

We start with the task of a half-cheetah (Fig. 1) agent, traversing terrains of differing slopes. The prior model is meta-trained on data from terrains with random slopes of low magnitudes, and the test trials are executed on difficult out-of-distribution tasks such as basins, steep hills, etc. As shown in Fig. 2, MB and its corresponding adaptation version DE do not perform well on these out-of-distribution tasks, even though those models were trained on the same data that the meta-trained model received. The bad performance of MB indicates the need for adaptation (as opposed to assuming a single model can do everything), while the bad performance of DE indicates the need for specialized meta-learning to enable online learning with neural networks.

For the three meta-learning and adaptation options (ours, GS, TS), we expect GS to perform poorly due to continuous gradient steps causing it to overfit to recent data; that is, we expect that experience on the upward slopes to lead to deterioration of performance on downward slopes, or something similar. However, based on both our qualitative and quantitative results, we see that the meta-learning procedure seems to have initialized the agent with a parameter space in which these various "tasks" are not seen as substantially different, where online learning by SGD performs well. This suggests that the meta-learning process finds a task space where there is an easy skill transfer of slopes; thus, even when our method is faced with the option of switching tasks or adding new tasks to its dynamic latent task distribution, it chooses not to do so (Fig. 3). Unlike findings that we will see later, it is interesting that the discovered task space here does not correspond to human-distinguishable categorical labels. Finally, we clarify that these tasks of changing slopes are not particularly similar to each other (and that the discovered task space is perhaps useful), because the two non-meta-learning baselines do indeed fail at these test tasks despite having similar training performance on the shallow training slopes.

7.2 Half-Cheetah Motor Malfunctions

While the findings from the half-cheetah on sloped terrains illustrated that separate task parameters aren't always necessary for what might externally seem like separate tasks, we also want to study agents that experience more drastically changing non-stationary task distributions during their experience in the world. For this set of experiments, we train all models on data where an actuator is selected at random to experience a malfunction during the rollout. In this case, malfunction means that the polarity or magnitude of actions applied to that actuator are altered. Fig. 4 shows the results of various methods on drastically out-of-distribution test tasks, such as altering all actuators at once.

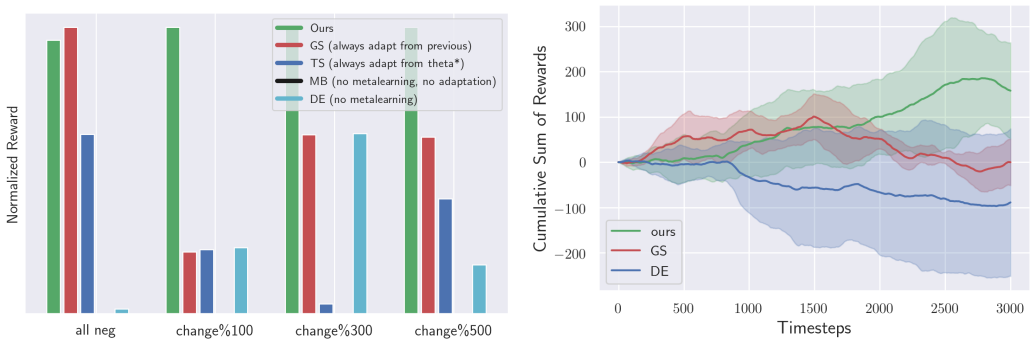

Figure 4: Results on the online motor malfunction trials, where different trials are shown task distributions that modulate at different frequencies (or stay constant, for the first category). Here, online learning is critical for good performance, and simply taking gradient steps (GS) leads to overfitting to the current task.

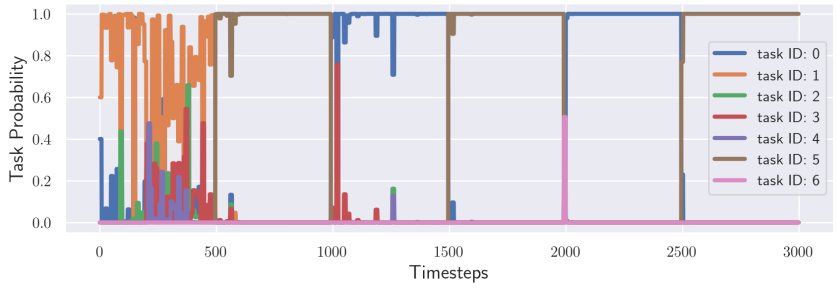

Figure 5: Latent task variable distribution over the course of the online learning trial, where the underlying motor malfunction changes every 500 timesteps. We find that our method is able to successful recover the task structure, recognize when the underlying task has changed, and recall tasks seen previously.

The left of Fig. 4 shows that when the task distribution during the test trials contains only a single task, such as 'sign negative' where all actuators are prescribed to be the opposite polarity, then GS performs well by continuously performing gradient updates on incoming data. However, as shown in the other tasks of Fig. 4, the performance of GS substantially deteriorates when the agent experiences a non-stationary task distribution. Due to overspecialization on recent incoming data, methods that continuously adapt like GS tend to forget and lose previously existing skills. This overfitting and forgetting of past skills is also illustrated in Fig. 4 as a consistent deterioration in the performance of GS. Our method, on the other hand, dynamically builds a probabilistic task distribution and allows adaptation to these difficult tasks, without forgetting past skills. We show a sample task setup in Fig. 5, where the agent experiences alternating periods of normal and crippled-leg operation. This plot shows the successful recognition of new tasks as well as old tasks; note that both the recognition and adaptation are all done online, without using a bank of past data to perform the adaptation, and without a human-specified set of task categories.

7.3 Crippling of End Effectors on Six-Legged Crawler

To further examine the effects of our continual online adaptation algorithm, we study another, more complex agent: a 6-legged crawler, as shown in Fig. 6. In these experiments, all models are trained on random joints being crippled (i.e., unable to apply actuator commands). In Fig. 7, we present two illustrative test tasks: (1) the agent sees a set configuration of crippled joints for the entire duration of its test-time experience, and (2) the agent receives alternating periods of experience, between regions of normal operation and regions of having crippled legs.

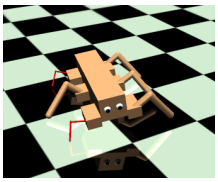

Figure 6: Six-legged crawler robot, shown with crippled legs.

The first setting is similar to data seen during training, and thus, we see that even the MB and DE baselines do not fail. The methods that include both meta-learning and adaptation, however, do have higher performance. Furthermore, we see again that continuously taking gradient steps (GS) in this case of a single-task setting is not detrimental. The second setting's non-stationary task distribution (when the leg crippling is dynamic) illustrates the need for online adaptation (MB fails), the need for a good prior to adapt from (DE fails), the harm of overfitting to

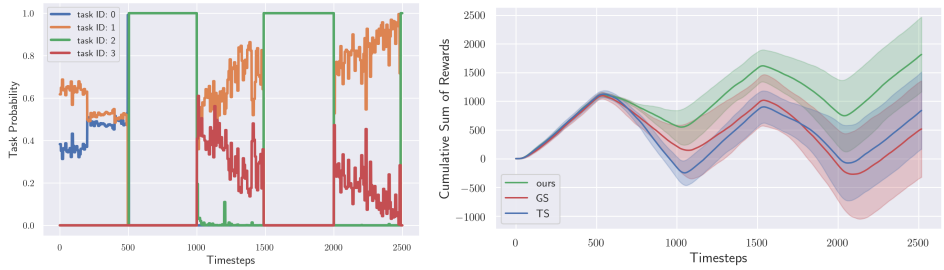

Figure 8: Results on the crawler experiments. Left: online recognition of latent task probability distribution for alternating periods of normal/crippled experience for crawler. Right: TS/GS not improving after multiple times of seeing the same tasks.

recent experience and thus forgetting older skills (GS low performance), and the need for further adaptation away from the prior (limited TS performance). With our method, this agent is able to build its own representation of "task" switches, and we see that this switch does indeed correspond to recognizing regions of leg crippling (left of Fig. 7.3). The plot of the cumulative sum of rewards (right of Fig. 7.3) of each of our three meta-learning plus adaptation methods includes this same task switch pattern every 500 steps: Here, we can clearly see that steps 500-1000 and 1500-2000 were the crippled regions, where the main difference between GS and our method is that GS actually gets worse on the second and third times it sees normal operation, whereas our method is noticeable better as it sees the task more often. Note that this improvement of both skills is possible with our method, where the development of one skill actually does not hinder the other.

To further test this observation, we ran another experiment by letting the crawler experience (during each trial) walking straight, making turns, and sometimes having a crippled leg. We compared the performance during the first 500 time steps of "walking forward in a normal configuration" to its last 500 time steps of "walking forward in a normal configuration." While the beginning performance of GS was comparable to our method (average performance difference of +/-10%), its ending performance was **200%** lower. Note the detrimental effect of updating knowledge without allowing for separate task specialization/adaptation.

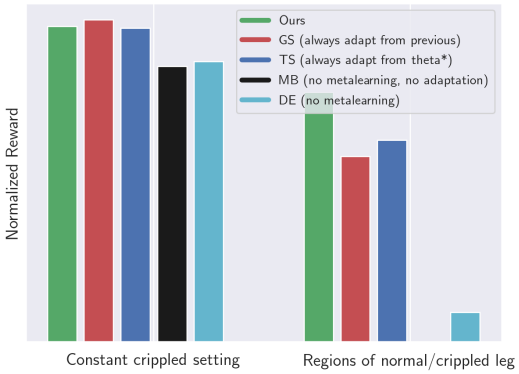

Figure 7: Quantitative online learning results on the crawler. When the task is fixed during training, adaptation is not necessary and all methods perform well. In contrast, when the task switches occur dynamically within the trial, our method is the only approach that can effectively learn online throughout the trial.

8 Discussion and Future Work

We presented an online learning method for neural network models that can handle non-stationary, multi-task settings within each trial. Our method adapts the model directly with SGD, where an EM algorithm uses a Chinese restaurant process prior to maintain a distribution over tasks and handle non-stationarity. Although SGD generally makes for a poor online learning algorithm in the streaming setting for large parametric models such as deep neural networks, we observe that, by (1) meta-training the model for fast adaptation with MAML and (2) employing our algorithm for probabilistic updates at test time, we can enable effective online learning with neural networks. In our experiments, we applied this approach to model-based RL, and demonstrated that it could be used to adapt the behavior of simulated robots faced with various new and unexpected tasks. Our results showed that our method can develop its own notion of task, continuously adapt away from the prior as necessary (to learn even tasks that require more adaptation), and recall tasks it has seen before. While we use model-based RL as our evaluation domain, our method is general and could be applied to other streaming and online learning settings. An exciting direction for future work would be to apply our method to domains such as time series modeling and active online learning.

REFERENCES

Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. *arXiv preprint arXiv:1710.03641*, 2017.

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Anonymous. Modulating transfer between tasks in gradient-based meta-learning. 2019. URL `https://openreview.net/forum?id=rkxLDyBuFX`. under review.

Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the optimization of a synaptic learning rule. In *Optimality in Artificial and Biological Neural Networks*, 1992.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

Léon Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia C Wilson, and Michael I Jordan. Streaming variational bayes. In *Advances in Neural Information Processing Systems*, pp. 1727–1735, 2013.

Ignasi Clavera, Anusha Nagabandi, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt: Meta-learning for model-based control. *arXiv preprint arXiv:1803.11347*, 2018.

Julien Doyon and Habib Benali. Reorganization and plasticity in the adult brain during learning of motor skills. *Current opinion in neurobiology*, 15(2):161–167, 2005.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv:1611.02779*, 2016.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 2011.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *International Conference on Machine Learning (ICML)*, 2017.

J Randall Flanagan and Alan M Wing. Modulation of grip force with load force during point-to-point arm movements. *Experimental Brain Research*, 95(1):131–143, 1993.

Meire Fortunato, Charles Blundell, and Oriol Vinyals. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*, 2017.

Robert Herman. *Neural control of locomotion*, volume 18. Springer, 2017.

Matthew Hoffman, Francis R Bach, and David M Blei. Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pp. 856–864, 2010.

Amir Jafari, Amy Greenwald, David Gondek, and Gunes Ercal. On no-regret learning, fictitious play, and nash equilibrium. In *ICML*, volume 1, pp. 226–233, 2001.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 2017.

Ben Krause, Liang Lu, Iain Murray, and Steve Renals. Multiplicative lstm for sequence modelling. *arXiv preprint arXiv:1609.07959*, 2016.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. Dynamic evaluation of neural sequence models. *CoRR*, abs/1709.07432, 2017.

David Lopez-Paz et al. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, 2017.

Tsendsuren Munkhdalai and Hong Yu. Meta networks. *International Conference on Machine Learning (ICML)*, 2017.

K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, Dept. Computer Science, UC Berkeley, 2002. URL `https://www.cs.ubc.ca/~murphyk/Thesis/thesis.html`.

Devang K Naik and RJ Mammone. Meta-neural networks that learn by learning. In *International Joint Conference on Neural Netowrks (IJCNN)*, 1992.

Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv:1710.10628*, 2017.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations (ICLR)*, 2017.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proc. CVPR*, 2017.

Marek Rei. Online representation learning in recurrent neural language models. *CoRR*, abs/1508.03854, 2015.

Samuel Ritter, Jane X Wang, Zeb Kurth-Nelson, Siddhant M Jayakumar, Charles Blundell, Razvan Pascanu, and Matthew Botvinick. Been there, done that: Meta-learning with episodic recall. *arXiv preprint arXiv:1805.09692*, 2018.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv:1606.04671*, 2016.

Doyen Sahoo, Quang Pham, Jing Lu, and Steven CH Hoi. Online deep learning: Learning deep neural networks on the fly. *arXiv preprint arXiv:1711.03705*, 2017.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning (ICML)*, 2016.

Jurgen Schmidhuber. Evolutionary principles in self-referential learning. *Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*, 1987.

Florian Stimberg, Andreas Ruttor, and Manfred Opper. Bayesian inference for change points in dynamical systems with reusable states-a chinese restaurant process approach. In *Artificial Intelligence and Statistics*, pp. 1117–1124, 2012.

Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*. Springer, 1998.

Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 1998.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv:1611.05763*, 2016.

Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Growing a brain: Fine-tuning by increasing model capacity. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, 2017.

A TEST-TIME PERFORMANCE VS TRAINING DATA

We verify below that as the meta-trained models are trained with more data, their performance on test tasks does improve.

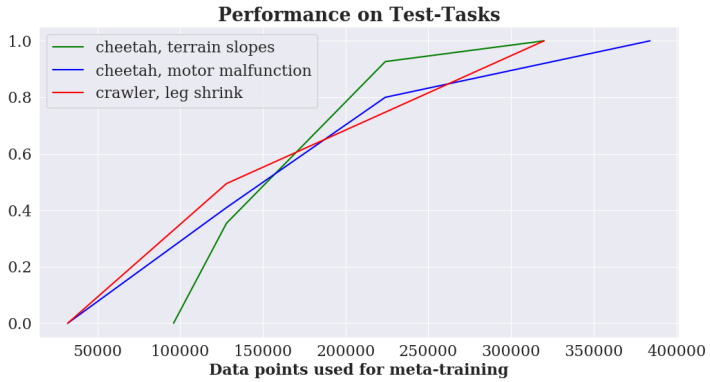

Figure 9: Performance on test tasks (i.e., unseen during training) of models that are meta-trained with differing amounts of data. Performance numbers here are normalized per agent, between 0 and 1.

B HYPERPARAMETERS

In all experiments, we use a dynamics model consisting of three hidden layers, each of dimension 500, with ReLU nonlinearities. The control method that we use is random-shooting model predictive control (MPC) where 1000 candidate action sequences each of horizon length H=10 are sampled at each time step, fed through the predictive model, and ranked by their expected reward. The first action step from the highest-scoring candidate action sequence is then executed before the entire planning process repeats again at the next time step.

Below, we list relevant training and testing parameters for the various methods used in our experiments. # Task/itr corresponds to the number of tasks sampled during each iteration of collecting data to train the model, and # TS/itr is the total number of times steps collected during that iteration (sum over all tasks).

Table 1: Hyperparameters for train-time

	Iters	Epochs	# Tasks/itr	# TS/itr	K	outer LR	inner LR (η)
Ours/TS/GS	12	50	16	2000-3000	16	0.001	0.01
MB/DE	12	50	16	2000-3000	16	0.001	N/A

Table 2: Hyperparameters for run-time

	$\alpha(CRP)$	LR (model update)	K (previous data)
Ours	1	0.01	16
GS	N/A	0.01	16
TS	N/A	0.01	16
MB	N/A	N/A	N/A
DE	N/A	0.01	16