Hierarchical Expert Networks for Meta-Learning

Anonymous Authors[1]

Abstract

The goal of meta-learning is to train a model on a variety of learning tasks, such that it can adapt to new problems within only a few iterations. Here we propose a principled information-theoretic model that optimally partitions the underlying problem space such that the resulting partitions are processed by specialized expert decision-makers. To drive this specialization we impose the same kind of information processing constraints both on the partitioning and the expert decision-makers. We argue that this specialization leads to efficient adaptation to new tasks. To demonstrate the generality of our approach we evaluate three meta-learning domains: image classification, regression, and reinforcement learning.

1. Introduction

Recent machine learning research has shown impressive results on incredibly diverse tasks from problem classes such as pattern recognition, reinforcement learning, and generative model learning (Devlin et al., 2018; Mnih et al., 2015; Schmidhuber, 2015). These success stories typically have two computational luxuries in common: a large database with thousands or even millions of training samples and a very long and extensive training period. However, applying these pre-trained models to new tasks naïvely usually leads to very poor performance, as with each new incoming batch of data, expensive and slow re-learning is required. In contrast to this, humans can learn from very few examples and excel at adapting quickly (Jankowski et al., 2011), for example in motor tasks (Braun et al., 2009) or at learning new visual concepts (Lake et al., 2015).

Sample-efficient adaptation to new tasks can be regarded as a form of meta-learning or "learning to learn" (Thrun & Pratt, 2012; Schmidhuber et al., 1997; Caruana, 1997) and is an ongoing and active field of research–see e.g. (Koch et al., 2015; Vinyals et al., 2016; Finn et al., 2017; Ravi & Larochelle, 2017; Ortega et al., 2019; Botvinick et al., 2019; Yao et al., 2019). Meta-learning has multiple definitions, but a common point is that the system learns on two levels, each with different time scales: slow meta-learning across different tasks, and fast learning to adapt to each task individually.

Here, we propose a novel information-theoretic learning paradigm for hierarchical meta-learning systems. Our method finds an optimal soft partitioning of the problem space by imposing information-theoretic constraints on both the process of expert selection and on the expert specialization. We argue that these constraints drive an efficient division of labor in systems that are bounded in their respective information processing power, where we make use of information-theoretic bounded rationality (Ortega & Braun, 2013). When the model is presented with previously unseen tasks it assigns them to experts specialized on similar tasks – see Figure 1. Additionally, expert networks specializing in only a subset of the problem space allow for smaller neural network architectures with only a few units per layer. To split the problem space and to assign the partitions to experts, we learn to represent tasks through a common latent embedding, that is then used by a selector network to distribute the tasks to the experts.

The outline of this paper is as follows: first, we introduce bounded rationality and meta-learning, next we introduce our novel approach and derive applications to classification, regression, and reinforcement learning. Finally, we conclude.

2. Background

An important concept in decision making is the notion of utility (Von Neumann & Morgenstern, 2007), where an agent picks an action $a_s^* \in \mathcal{A}$ such that it maximizes their utility in some context $s \in \mathcal{S}$, i.e. $a_s^* = \arg\max_a \mathbf{U}(s,a)$, where the utility is given by a function $\mathbf{U}(s,a)$ and the states distribution $p(s)$ is known and fixed. Trying to solve this optimization problem naïvely leads to an exhaustive search over all possible (a,s) pairs, which is in general a prohibitive strategy. Instead of finding an optimal strategy, a *bounded-rational decision-maker* optimally trades off expected utility and the processing costs required to

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

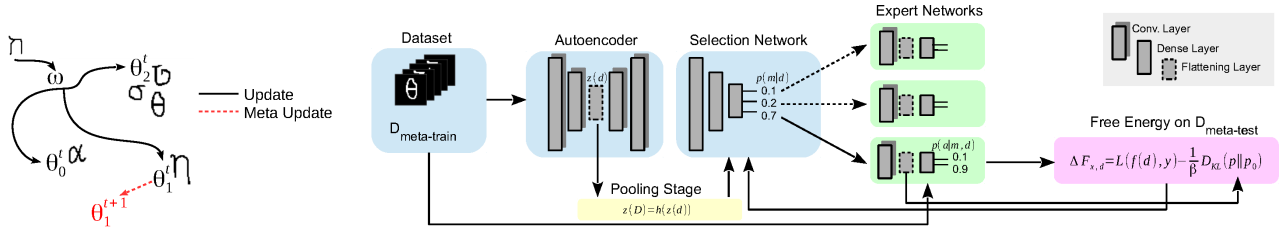

Figure 1. **Left**: The selector assigns the new input encoding to one of the three experts θ_0, θ_1, or θ_2, depending on the similarity of the input to previous inputs seen by the experts. **Right:** Our proposed method consists of three main stages. First, we feed the training dataset D_{train} through a convolutional autoencoder to find a latent representation $z(d_i)$ for each $d_i \in D_{\text{train}}$, which we get by flattening the preceding convolutional layer ("flattening layer"). We apply a pooling function to the resulting set of image embeddings which serves as input to the selection network.

adapt. In this study we consider the information-theoretic free-energy principle (Ortega & Braun, 2013) of bounded rationality, where the decision-maker's resources are modeled by an upper bound on the Kullback-Leibler divergence $\mathrm{D_{KL}}(p(a|s)||p(a)) = \sum_a p(a|s) \log \frac{p(a|s)}{p(a)}$ between the agent's prior distribution $p(a)$ and the posterior policy $p(a|s)$, resulting in the following constrained optimization problem:

$$\max_{p(a|s)} \sum_{s,a} p(s)p(a|s)\mathbf{U}(s,a) \tag{1}$$

$$\text{s.t. } \mathbb{E}_{p(s)}\left[\mathrm{D_{KL}}(p(a|s)||p(a))\right] \leq \mathrm{B}. \tag{2}$$

This constraint can also be interpreted as a regularization on $p(a|s)$. We can transform this into an unconstrained variational problem by introducing a Lagrange multiplier $\beta \in \mathbb{R}^+$:

$$\max_{p(a|s)} \mathbb{E}_{p(s|a)}\left[\mathbf{U}(s,a)\right] - \frac{1}{\beta}\mathbb{E}_{p(s)}\left[\mathrm{D_{KL}}(p(a|s)||p(a))\right]. \tag{3}$$

For $\beta \to \infty$ we recover the maximum utility solution and for $\beta \to 0$ the agent can only act according to the prior. The optimal prior in this case is given by the marginal $p(a) = \sum_{s \in S} p(s)p(a|s)$ (Ortega & Braun, 2013).

2.0.1. Hierarchical Decision Making

Aggregating several bounded-rational agents by a selection policy allows for solving optimization problems that exceed the capabilities of the individual decision-makers (Genewein et al., 2015). To achieve this, the search space is split into partitions such that each partition can be solved by a decision-maker. A two-stage mechanism is introduced: The first stage is an expert selection policy $p(x|s)$ that chooses an expert x given a state s and the second stage chooses an action according to the expert's posterior policy $p(a|s,x)$. The optimization problem given by Equation (3) can be extended to incorporate a trade-off between computational costs and utility in both stages:

$$\max_{p(a|s,x),p(x|s)} \mathbb{E}[\mathbf{U}(s,a)] - \frac{1}{\beta_1}I(S;X) - \frac{1}{\beta_2}I(S;A|X) \tag{4}$$

where β_1 is the resource parameter for the expert selection stage and β_2 for the experts. $I(\cdot;\cdot)$ is the mutual information between the two random variables. The solution can be found by iterating the following set of equations (Genewein et al., 2015):

$$\begin{cases} p(x|s) &= \frac{1}{Z(s)}p(x)\exp(\beta_1\,\Delta\mathrm{F_{par}}(s,x)) \\ p(x) &= \sum_s p(s)p(x|s) \\ p(a|s,x) &= \frac{1}{Z(s,x)}p(a|x)\exp(\beta_2\mathbf{U}(s,a)) \\ p(a|x) &= \sum_s p(s|x)p(a|s,x) \end{cases} \tag{5}$$

where $Z(s)$ and $Z(s,x)$ are normalization factors and $\Delta\mathrm{F_{par}}(s,x) = \mathbb{E}_{p(a|s,x)}[\mathbf{U}(s,a)] - \frac{1}{\beta_2}\mathrm{D_{KL}}(p(a|s,x)||p(a|x))$ is the *free energy* of the action selection stage. Thus the marginal distribution $p(a|x)$ defines a *mixture-of-experts* policy given by the posterior distributions $p(a|s,x)$ weighted by the responsibilities determined by the Bayesian posterior $p(s|x)$. Note that $p(s|x)$ is not determined by a given likelihood model, but is the result of the optimization process (4).

2.1. Meta Learning

Meta-learning algorithms can be divided roughly into Metric-Learning (Koch et al., 2015; Vinyals et al., 2016; Snell et al., 2017), Optimizer Learning (Ravi & Larochelle, 2017; Finn et al., 2017; Zintgraf et al., 2018), and Task Decomposition Models (Lan et al., 2019; Vezhnevets et al., 2019). Our approach depicted in Figure 1 can be seen as a member of the latter group.

2.1.1. Meta Supervised Learning

In a supervised learning task we are usually interested in a dataset consisting of multiple inputs and output pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=0}^{N}$ and the learner is tasked with finding a function $f(x)$ that maps from input to output, for example through a deep neural network. To do this, we split the dataset into training and test sets and fit a set of parameters θ on the training data and evaluate on test data using the learned function $f_\theta(x)$. In meta-learning, we are instead working with meta-datasets $D \in \mathcal{D}$, each containing regular datasets split into training and test sets. We thus have different meta-sets for meta-training, meta-validation, and meta-test ($D_{\text{meta-train}}$, $D_{\text{meta-val}}$, and $D_{\text{meta-test}}$, respectively). On $D_{\text{meta-train}}$, we are interested in training a learning procedure (the meta-learner) that can take as input one of its training sets D_{train} and produce a classifier (the learner) that achieves low prediction error on its corresponding test set D_{test}.

A special case of meta-learning for classification are K-Shot N-way tasks. In this setting, we are given for each dataset D a training set consisting of K labeled examples of each of the N classes ($K \cdot N$ examples per dataset) and corresponding test sets. In our study, we focus on the following variation of K-Shot 2-Way tasks: the meta-learner is presented with $2K$ samples (K positive and K negative examples) and must assign this dataset to an expert learner. Note that the negative examples may be drawn from any of the remaining classes.

2.1.2. Meta Reinforcement Learning

First, we give a short introduction to reinforcement learning in general, and then we show how this definition can be extended to meta reinforcement learning problems. We model sequential decision problems by defining a Markov Decision Process as a tuple $(\mathcal{S}, \mathcal{A}, P, r)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ the set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition probability, and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function. The aim is to find the parameter θ of a policy π_θ that maximizes the expected reward:

$$\theta^* = \arg\max_\theta \underbrace{\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} r(s_t, a_t)\right]}_{J(\pi_\theta)}. \tag{6}$$

We define $r(\tau) = \sum_{t=0}^{\infty} r(s_t, a_t)$ as the cumulative reward of trajectory $\tau = \{(s_t, a_t)\}_{i=0}^{\infty}$, which is sampled by acting according to the policy π, i.e. $(s, a) \sim \pi(\cdot|s)$, and $s_{t+1} \sim P(\cdot|s_t, a_t)$. Learning in this environment can then be modeled by reinforcement learning (Sutton & Barto, 2018), where an agent interacts with an environment over some (discrete) time steps t. At each time step t, the agent finds itself in a state s_t and selects an action a_t according to the policy $\pi(a_t|s_t)$. In return, the environment transitions to the next state s_{t+1} and generates a scalar reward r_t. This process continues until the agent reaches a terminal state after which the process restarts. The goal of the agent is to maximize the expected return from each state s_t, which is typically defined as the infinite horizon discounted sum of the rewards. A common choice to achieving this is Q-Learning (Watkins & Dayan, 1992), where we make use of an action-value function that is defined as the discounted sum of rewards $Q(\tau) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$, where $\gamma \in (0, 1]$ is a discount factor. Learning the optimal policy can be achieved in many ways. Here, we consider Policy gradient methods (Sutton et al., 2000) which are a popular choice to tackle continuous reinforcement learning problems. The main idea is to directly manipulate the parameters θ of the policy to maximize the objective $J(\pi_\theta)$ by taking steps in the direction of the gradient $\nabla_\theta J(\pi_\theta)$.

In meta reinforcement learning the problem is given by a set of tasks $t_i \in T$, where each task t_i is defined by an MDP $t_i = (\mathcal{S}, \mathcal{A}, P_i, r_i)$ as described earlier. We are now interested in finding a set of policies Θ that maximizes the average cumulative reward across all tasks in T and generalizes well to new tasks sampled from a different set of tasks T'.

Algorithm 1 Expert Networks for Supervised Meta-Learning.

Input: Data Distribution $p(\mathcal{D})$, number of samples K, batch-size M, training episodes N
Initialize parameters θ, ϑ
for all i = 0, 1, 2, ..., N **do**
  Sample batch of M datasets $D_i \sim p(\mathcal{D})$, each consisting of a training dataset $D_{\text{meta-train}}$ and a meta-validation dataset $D_{\text{meta-val}}$ with $2K$ samples each
  for all $D \in D_i$ **do**
    Find Latent Embedding $z(D_{\text{meta-train}})$
    Select expert $m \sim p_\theta(m|z(D_{\text{meta-train}}))$
    Compute $\Delta F_{D,x}$ of m on $D_{\text{meta-val}}$
    Update selection parameters θ with $\Delta F_{D,x}$
    Update Autoencoder with pos. samples in D_i
    Update expert m parameters ϑ with assigned $D_{\text{meta-train}}$
  end for
end for
return θ, ϑ

3. Expert Networks for Meta-Learning

Information-theoretic bounded rationality postulates that hierarchies and abstractions emerge when agents have only limited access to computational resources (Genewein et al., 2015), e.g. limited sampling complexity (Hihn et al., 2018)

	BASELINES		OUR METHOD					
			NUMBER OF EXPERTS					
	PRE-TRAINING	MAML	2		4		16	
K	**% Acc**	**% Acc**	**% Acc**	**I(M;X)**	**% Acc**	**I(M;X)**	**% Acc**	**I(M;X)**
1	50.6 ($\pm$ 0.03)	81.2 ($\pm$ 0.03)	66.4 ($\pm$ 0.02)	0.99 ($\pm$ 0.01)	75.8 ($\pm$ 0.02)	1.96 ($\pm$ 0.01)	**82.8** ($\pm$ 0.01)	3.2 ($\pm$ 0.03)
5	54.1 ($\pm$ 0.09)	**88.0** ($\pm$ 0.01)	67.3 ($\pm$ 0.01)	0.93 ($\pm$ 0.01)	75.5 ($\pm$ 0.01)	1.95 ($\pm$ 0.10)	85.2 ($\pm$ 0.01)	3.3 ($\pm$ 0.02
10	55.8 ($\pm$ 0.02)	89.2 ($\pm$ 0.01)	76.2 ($\pm$ 0.04)	0.95 ($\pm$ 0.30)	86.7 ($\pm$ 0.01)	1.90 ($\pm$ 0.03)	**95.9** ($\pm$ 0.01)	3.1 ($\pm$ 0.02)

Table 1. Classification accuracy after 10 gradient steps on the validation data. Adding experts consistently improves performance, obtaining the best results with an ensemble of 16 experts. Pre-training refers to a single expert system trained on the complete dataset. Our method outperforms the pre-training and MAML baseline (see Section 4.1 for experimental details).

or limited representational power (Hihn et al., 2019). We will show that forming such abstractions equips an agent with the ability of learning the underlying problem structure and thus enables learning of unseen but similar concepts. The method we propose comes out of a unified optimization principle and has the following important features:

1. A regularization mechanism to enforce the emergence of expert policies.
2. A task compression mechanism to extract relevant task information.
3. A selection mechanism to find the most efficient expert for a given task.
4. A regularization mechanism to improve generalization capabilities.

3.1. Latent Task Embeddings

Note that the selector assigns a complete dataset to an expert and that this can be seen as a meta-learning task, as described in (Ravi & Larochelle, 2017). To do so, we must find a feature vector $z(d)$ of the dataset d. This feature vector must fulfill the following desiderata: 1) invariance against permutation of data points in d, 2) high representational capacity, 3) efficient computability, and 4) constant dimensionality regardless of sample size K. In the following we propose such features for image classification, regression, and reinforcement learning problems.

For image classification we propose to pass the positive images in the dataset through a convolutional autoencoder and use the respective outputs of the bottleneck layer. Convolutional Autoencoders are generative models that learn to reconstruct their inputs by minimizing the Mean-Squared-Error between the input and the reconstructed image (see e.g. (Chen et al., 2019)). In this way we get similar embeddings $z(d)$ for similar inputs belonging to the same class. The latent representation is computed for each positive sample in d and then passed through a pooling function $h(z(d))$ to find a single embedding for the complete dataset–see figure 1 for an overview of our proposed model. While in principle functions such as mean, max, and min can be used, we found that max-pooling yields the best results. The authors of (Yao et al., 2019) propose a similar feature set.

For regression we define a similar feature vector. The K training data points are transformed into a feature vector $z(d)$ by binning the points into N bins according to their respective x value and collecting the respective y value. If more than one point falls into the same bin the y values are averaged, thus providing invariance against the order of the data points in $D_{\text{meta-train}}$. We use this feature vector to assign each data set to an expert according to $p_\theta(x|z(d))$.

In the reinforcement learning setting we use a dynamic recurrent neural network (RNN) with LSTM units (Hochreiter & Schmidhuber, 1997) to classify trajectories. We feed the RNN with (s_t, a_t, r_t, t) tuples to describe the underlying Markov Decision Process describing the task. At $t = 0$ we sample the expert x according to the learned prior distribution $p(x)$, as there is no information available so far. The authors of (Lan et al., 2019) propose a similar feature set.

3.2. Reinforcement Learning

In the following we will derive our algorithm for specialization in hierarchical reinforcement learning agents. Note that in the reinforcement learning setup the reward function $r(x, a)$ defines the utility $\mathbf{U}(x, a)$. In maximum entropy RL (see e.g., Haarnoja et al. (2017) (Haarnoja et al., 2017)) the regularization penalizes deviation from a fixed uniformly distributed prior, but in a more general setting we can discourage deviation from an arbitrary prior policy by optimizing for:

$$\max_p \mathbb{E}_p \left[\sum_{t=0}^{\infty} \gamma^t \left(r(x_t, a_t) - \frac{1}{\beta} \log \frac{p(a_t|x_t)}{p(a)} \right) \right], \quad (7)$$

where β trades off between reward and entropy, such that $\beta \to \infty$ recovers the standard RL value function and $\beta \to 0$ recovers the value function under a random policy.

To optimize the objective (7) we define two separate kinds

of value function, V_ϕ for the selector and one value function V_φ for each expert. Thus, each expert is an actor-critic with separate actor and critic networks. Similarly, the selector has an actor-critic architecture, where the actor network selects experts and the critic learns to predict the expected free energy of the experts depending on a state variable. The selector's policy is represented by p_θ, while each expert's policy is represented by a distribution p_ϑ.

3.2.1. Value Functions

In standard reinforcement learning the discounted reward is defined as

$$R_t = \sum_{l=0}^{T} \gamma^l r(x_{t+l}, a_{t+l}), \tag{8}$$

which is usually learned through a parameterized value function V_ψ by regressing

$$\psi^* = \arg\min_\psi \frac{1}{|\mathcal{D}|T} \sum_{\tau\in\mathcal{D}} \sum_{t=0}^{T} (V_\psi(x_t) - R_t)^2. \tag{9}$$

Here ψ are some arbitrary parameters of the value representation, $V_\psi(x_t)$ is the predicted value estimate for state x_t, and $\mathcal{D}$ is a set of trajectories τ up to horizon T collected by roll-outs of the policies.

Similar to the standard discounted reward R_t, we can now define a discounted free energy F_t as

$$F_t = \sum_{l=0}^{T} \gamma^l f(x_{t+l}, m_{t+l}, a_{t+l}), \tag{10}$$

where $f(x, m, a) = r(x, a) - \frac{1}{\beta_2} \log \frac{p_\vartheta(a|x,m)}{p(a|m)}$. Accordingly, we can learn a value function V_φ for each expert by parameterizing the value function with a neural network and performing regression on the mean-squared-error

$$\varphi^* = \arg\min_\varphi \frac{1}{|\mathcal{D}|T} \sum_{\tau\in\mathcal{D}} \sum_{t=0}^{T} (V_\varphi(x_t) - F_t)^2, \tag{11}$$

where φ are the parameters of the expert's value net, $V_\varphi(x_t)$ is the network's predicted value estimate for state x_t, and $\mathcal{D}$ is a set of trajectories. Similarly, we can define a discounted free energy $\bar{F}_t$ for the selector

$$\bar{F}_t = \sum_{l=0}^{T} \gamma^l \bar{f}(x_{t+l}, m_{t+l}), \tag{12}$$

with $\bar{f}(x, m) = \mathbb{E}_{p_\vartheta(a|x,m)}[r(x, a) - \frac{1}{\beta_2} \log \frac{p(a|x,m)}{p(a|m)}]$ that is learned through the selector's value function V_ϕ by regressing

$$\phi^* = \arg\min_\phi \frac{1}{|\mathcal{D}|T} \sum_{\tau\in\mathcal{D}} \sum_{t=0}^{T} (V_\phi(x_t) - \bar{F}_t)^2, \tag{13}$$

thus updating the parameters ϕ.

3.2.2. Policy Learning

In standard reinforcement learning a common technique to update a parametric policy representation $p_\omega(a|x)$ with parameters ω, is to use policy gradients that optimize the cumulative reward

$$J(\omega) = \mathbb{E}\left[p_\omega(a|x) V_\psi(x)\right] \tag{14}$$

expected under the critic's prediction $V_\psi(x)$, by following the gradient

$$\nabla_\omega J(\omega) = \mathbb{E}\left[\nabla_\omega \log p_\omega(a|x) V_\psi(x)\right]. \tag{15}$$

This policy gradient formulation (Sutton et al., 2000) is prone to producing high variance gradients. A common technique to reduce the variance is to formulate the updates using the advantage function instead of the reward (Arulkumaran et al., 2017). The advantage function $A(a_t, s_t)$ is a measure of how well a certain action a performs in a state x compared to the average performance in that state, i.e. $A(a, x) = Q(x, a) - V_\psi(x)$. Here, $V(x)$ is the value function and is a measure of how well the agent performs in state x, and $Q(x, a)$ is an estimate of the cumulative reward achieved in state x when the agent executes action a. Thus, the advantage is an estimate of how advantageous it is to pick a in state x compared to a baseline performance measured by $V_\psi(x)$. Instead of learning the value and the Q function, we can approximate the advantage function solely based on the critic's estimate $V_\psi(x)$ in the following way

$$A(x_t, a_t) = \underbrace{r(x_t, a_t) + \gamma V_\psi(x_{t+1})}_{\approx Q(x_t, a_t)} - V_\psi(x_t), \tag{16}$$

giving the following gradient estimates for the policy parameters

$$\omega \leftarrow \omega - \frac{\alpha}{|\mathcal{D}|} \sum_{\tau\in\mathcal{D}} \sum_{t=0}^{T} \nabla_\omega \log p_\omega(a_t|x_t) A(x_t, a_t), \tag{17}$$

where α is a learning rate and $\mathcal{D}$ is a set of trajectories τ produced by the policies.

Similar to the standard policy update based on the advantage function, the expert selection stage can be formulated by optimizing the expected advantage $\mathbb{E}\left[p_\vartheta(a|x, m) A_m(x, a)\right]$ for expert m with

$$A_m(x_t, a_t) = f(x_t, m, a_t) + \gamma V_\varphi(x_{t+1}) - V_\varphi(x_t),$$

resulting in the update equation

$$\vartheta \leftarrow \vartheta - \frac{\alpha}{|\mathcal{D}|} \sum_{\tau\in\mathcal{D}} \sum_{t=0}^{T} \nabla_\vartheta \log p_\vartheta(a_t|m, x_t) A_m(x_t, a_t). \tag{18}$$

Accordingly, we can define an expected advantage function $\mathbb{E}\left[p_\theta(m|x)\bar{A}(x,m)\right]$ for the selector with

$$\bar{A}(x,m) = \mathbb{E}_{p_\vartheta(a|x,m)}\left[A_m(x,a)\right],$$

resulting in the update equation

$$\theta \leftarrow \theta - \frac{\alpha}{|\mathcal{D}|}\sum_{\tau\in\mathcal{D}}\sum_{t=0}^{T}\nabla_\theta \log p_\theta(m_t|x_t)\bar{A}(x_t,m_t), \quad (19)$$

where m_t denotes the expert m selected at time t. We estimate the double expectation by Monte Carlo sampling, where in practice we use a single (x,m,a) tuple for $\hat{f}(x,m)$, which enables us to employ our algorithm in an on-line optimization fashion.

In the RL setup the reward function $r(x,a)$ defines the utility $\mathbf{U}(x,a)$. To optimize the objective we define value functions, V_ϕ for the selector and one value function V_φ for each expert. Thus, the selector and each expert is an actor-critic architecture. We define a discounted free energy as

$$F_t = \sum_{l=0}^{T}\gamma^l\left(r(x_{t+l},a_{t+l}) - \frac{1}{\beta_2}\log\frac{p_\vartheta(a_{t+l}|x_{t+l},m_{t+l})}{p(a_{t+l}|m_{t+l})}\right),$$

which we learn through a value function V_φ for each expert. We define the selector's discounted free energy

$$\bar{F}_t = \sum_{l=0}^{T}\gamma^l\bar{f}(x_{t+l},m_{t+l}),$$

with $\bar{f}(x,m) = \mathbb{E}_{p_\vartheta(a|x,m)}[r(x,a) - \frac{1}{\beta_2}\log\frac{p_\vartheta(a|x,m)}{p(a|m)}]$ that is learned through the selector's value function V_ϕ. Now let $p_\theta(m|x)$ be the selection policy and $p_\vartheta(a|x,m)$ the expert policy. The expert selection stage is optimizing the expected advantage

$$\mathbb{E}\left[p_\vartheta(a|x,m)A_m(x,a)\right]$$

for expert m with

$$A_m(x_t,a_t) = f(x_t,m,a_t) + \gamma V_\varphi(x_{t+1}) - V_\varphi(x_t).$$

Accordingly, we define an expected advantage function

$$\mathbb{E}\left[p_\theta(m|x)\bar{A}(x,m)\right]$$

for the selector with

$$\bar{A}(x,m) = \mathbb{E}_{p_\vartheta(a|x,m)}\left[A_m(x,a)\right].$$

Each policy is updated by gradient descent on the objectives.

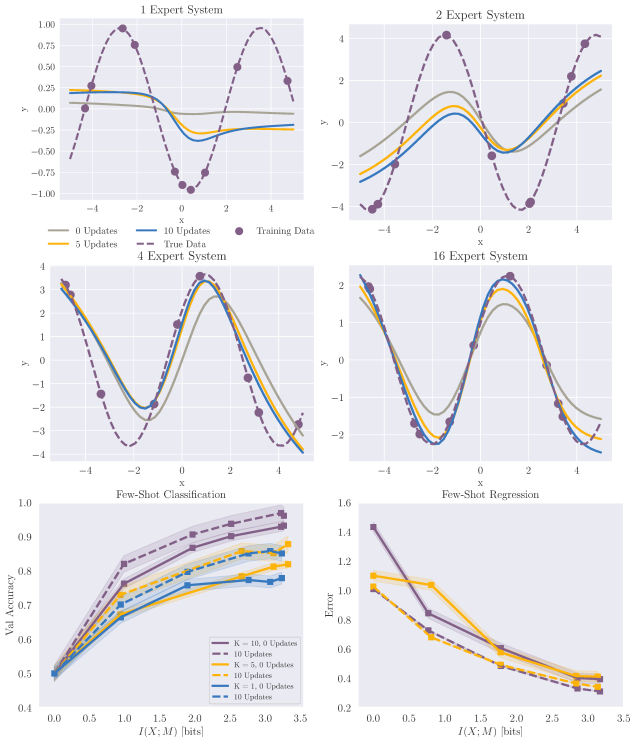

Figure 2. The single expert system is not able to learn the structure of the sine wave, where the two expert can capture the periodic structure. The lower figure shows how the error decreases as we add more experts in the classification and regression setting, indicating successful specialization and expert selection.

4. Results

4.1. Sinusoid Regression

We adopt this task from (Finn et al., 2017). In this K-shot problem, each task consists of learning to predict a function of the form $y = a\cdot\sin(x+b)$, with both $a\in[0.1,5]$ and $b\in[0,2\pi]$ chosen uniformly, and the goal of the learner is to find y given x based on only K pairs of (x,y). Given that the underlying function changes in each iteration it is impossible to solve this problem with a single learner. Our results show that by combing expert networks, we are able to reduce the generalization error iteratively as we add more experts to our system–see Figure 2 where we also show how the system is able to capture the underlying problem structure as we add more experts. In Figure 4 we visualize how the selector's partition of the problem space looks like.

Experimental details: We use a two layer selection network with 16 units each followed by tanh non-linearities. The experts are single layer networks (i.e. linear regressors) that learn log-variance and mean of a Gaussian distribution which they use for prediction.

Paramater	Task Distribution T	T'
Distance Penalty	$[10^{-3}, 10^{-1}]$	$[10^{-3}, 10^{-2}]$
Goal Position	[0.3, 0.4]	[0, 3]
Start Position	[-0.15, 0.15]	[-0.25, 0.25]
Motor Torques	$[0, 5]$	$[0, 3]$
Inverted Control	$p = 0.5$	$p = 0.5$
Gravity	[0.01, 4.9]	[4.9, 9.8]
Motor Actuation	[185, 215]	[175, 225]

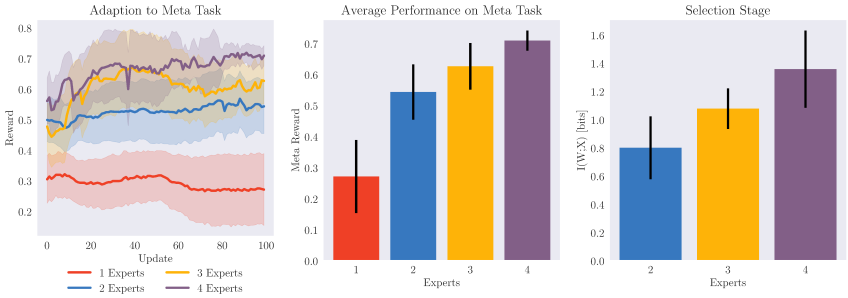

Figure 3. In each Meta-Update Step we sample N tasks from the training task set T and update the agents. After training is completed we evaluate their respective performance on a tasks from the meta test set T'. Rewards are normalized to $[0, 1]$ and the episode horizon is 500 time steps. Results are averaged over 10 random seeds and trained for 1000 episodes each with a batch of 64 environments. The agent achieves higher reward when adding more experts while the information-processing of the selection stage increases, indicating that the added experts specialize successfully.

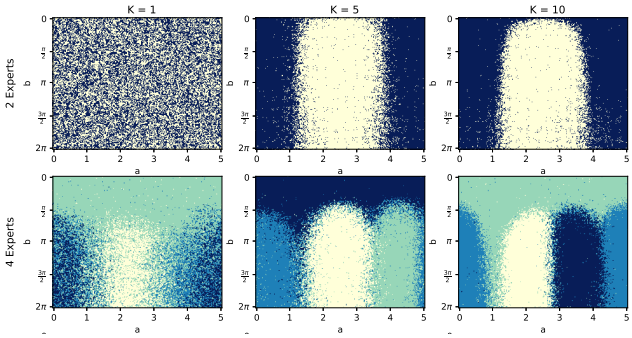

Figure 4. Here we show the soft-partition found by the selection policy for the sine prediction problem. Each color represents an expert. To generate these plots we train a system on $K = 1, 5$ or 10, sample a, b and K points and feed the dataset to the selection policy. We can see that the selection policy becomes increasingly more precise as we provide more points per dataset to the system.

4.2. Few-Shot Classification

The Omniglot dataset (Lake et al., 2011) consists of over 1600 characters from 50 alphabets. As each character has only 20 samples each drawn by a different person, this forms a challenging meta-learning benchmark, see e.g. (Finn et al., 2017; Vinyals et al., 2016; Ravi & Larochelle, 2017).

We train the learner on a subset of the dataset ($\approx 80\%$, i.e. $\approx$ 1300 classes) and evaluate the remaining ≈ 300 classes, thus investigating the generalization. In each training episode we build the datasets $D_{\text{meta-train}}$ and $D_{\text{meta-test}}$ by selecting a target class c_t and sampling K positive and K negative samples. To generate negative samples, we draw K images randomly out of the remaining $N - 1$ classes. We present the selection network with the feature presentation of the K positive training samples (see Figure 1), but evaluate the experts' performance on the $2K$ test samples in $D_{\text{meta-test}}$. In this way, the free energy of the experts becomes a generalization measure. Using this optimization scheme, we train the networks to become *experts* in recognizing a subset of classes. After a proper expert is selected we train that expert using the $2K$ samples from the training dataset—see Figure 2 and Table 1 for results. To generate this figure, we ran 10-fold cross-validation on the whole dataset and show the averaged performance metric and the respective standard-deviation across the folds. In both settings, "0 bits" corresponds to a single expert, i.e. a single neural network trained on the task.

Experimental details: We followed the design of (Vinyals et al., 2016) but reduce the number of blocks to one. We used a single convolutional block consisting of 32 3×3 filters with strided convolutions followed by a batch normalization layer and a ReLu non-linearity. During training we used a meta-batch size of 16. The convolutional autoencoder is a 3 layer network consisting of 16, 16, and 4 filters each with size 3×3 with strided convolutions followed by a leaky ReLu non-linearity. The layers are mirrored by deconvolotional layers to reconstruct the image. This results in an image embedding with dimensionality 64. The selection network is a two layer network with 32 units each, followed by a ReLu non-linearity, a dropout layer (Srivastava et al., 2014) per layer. We augment the dataset by rotating each image in 90, 180, and 270 degrees resulting in 80 images per class. We also normalize the images two be in (0,1) range. We evaluate our method by resetting the system to the state after training and allow for 10 gradient updates and report the final accuracy. To evaluate MAML on 2-way N-shot omniglot dataset we used a inner learning rate of $\alpha = 0.05$ and one inner update step per iteration for all settings. We used a single convolutional block followed by a fully connected layer with 64 units and a ReLU non-linearity. Note, that we reduce the number of layers to make the tests comparable. Using the suggested architectures by (Finn et al., 2017) we achieve classification accuracy $\geq 95\%$.

4.3. Meta Reinforcement Learning

We create a set of RL tasks by sampling the parameters for the Inverted Double Pendulum problem (Sutton, 1996) implemented in OpenAI Gym (Brockman et al., 2016), where the task is to balance a two-link pendulum in an upward position. We modify inertia, motor torques, reward function, goal position and invert the control signal – see Table 4 for details. The control signal a is continuous in the interval [-1,1] and is generated by neural network that outputs μ and $\log(\sigma)$ of a gaussian. The action is then sampled by re-parameterizing the distribution to $p(a) = \mu + \exp(\sigma)\epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$, so that the distribution is differentiable w.r.t to the network outputs.

We build the meta task set T' on the same environment, but change the parameter distribution and range, providing new but similar reinforcement learning problems. During training, we sample M environments in each episode and perform updates as described earlier. During evaluation, we measure the system's performance on tasks sampled from T'. We trained the system for 1000 Episodes with 64 tasks from T and evaluate for 100 updates on tasks from T'. We report the results in Figure 3, where we can see improving performance as more experts are added and the increasing mutual information in the selection stage indicates that the tasks can be assigned to their respective expert policy.

Experimental details: The selector's actor and critic net are build of RNNs with 200 hidden units each. The critic is trained to minimize the Huber loss between the prediction and the cumulative reward. The experts are two layer networks with 64 units each followed by ReLu non-linearities and used to learn the parameters of a Gaussian distribution. The critics have the same architecture (except for the output dimensionality).

5. Discussion

The results show that our method can identify sub-regions of the problem set and solve them efficiently with expert networks. In effect, this equips the system with several initializations covering the problem space and thus enables it to adapt quickly to new but similar tasks. To reliably identify such tasks, we have proposed feature extraction methods for classification, regression, and reinforcement learning, that could be simply be replaced and improved in future work. The strength of our model is that it follows from simple principles that can be applied to a large range of problems. Moreover, the system performance can be interpreted in terms of the information processing of the selection stage and the expert decision-makers.

Most other methods for meta-learning such as the work of (Finn et al., 2017) and (Ravi & Larochelle, 2017) find an initial parametrization of a single learner, such that the agent can adapt quickly to new problems. This initialization can be interpreted as a compression of the most common task properties over tasks and our method takes this idea one step further by finding a possibly disjunct set of such compressed task properties. Another way of thinking of such lossy compressions is to go from a task-specific posterior to a task-agnostic prior strategy. By having a set of priors, the task-specific information is thus directly available instead of a delayed availability after several iterations as in MAML (Finn et al., 2017) and the work of (Ravi & Larochelle, 2017). In principle, this can help to adapt within fewer iterations. Thus our method can be seen as the general case of such monolithic meta-learning algorithms.

Instead of learning similarities within a problem, we can also try to learn similarities between completely different problems (e.g. different classification datasets), as is described in the work of (Yao et al., 2019). In this way, the partitioning is largely governed by different tasks, where our study however focuses on discovering meta-information within the same task family, where the meta-partitioning is determined solely by the optimization process and can thus potentially discover unknown dynamics and relations within a task family. However, applying our method to this kind of problem is straightforward, but remains open for future work at this point.

6. Conclusion

We have introduced and evaluated a novel hierarchical information-theoretic approach to meta-learning. In particular, we leveraged an information-theoretic approach to bounded rationality (Leibfried et al., 2017; Grau-Moya et al., 2019; Hihn et al., 2019; Schach et al., 2018; Gottwald & Braun, 2019; Lindig-Leon et al., 2019). Although our method is widely applicable, it suffers from low sample efficiency in the RL domain. An interesting research direction would be to combine our system with model-based RL which is known to improve sample efficiency. Another research direction would be to investigate the performance of our system in continual adaption tasks, such as in (Yao et al., 2019). There the system is continuously provided with data sets (e.g. additional classes and samples). Another limitation is the restriction to binary meta classification tasks, which we leave for feature work.

ACKNOWLEDGMENTS

This work was supported.

References

Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38,

2017.

Balasundaram, S. and Meena, Y. Robust support vector regression in primal with asymmetric huber loss. *Neural Processing Letters*, 49(3):1399–1431, 2019.

Botvinick, M., Ritter, S., Wang, J. X., Kurth-Nelson, Z., Blundell, C., and Hassabis, D. Reinforcement learning, fast and slow. *Trends in cognitive sciences*, 2019.

Braun, D. A., Aertsen, A., Wolpert, D. M., and Mehring, C. Motor task variation induces structural learning. *Current Biology*, 19(4):352–357, 2009.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Caruana, R. Multitask learning. *Machine learning*, 28(1): 41–75, 1997.

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C. F., and Huang, J.-B. A closer look at few-shot classification. In *Proceedings of the International Conference on Representation Learning*, 2019.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

Galashov, A., Jayakumar, S. M., Hasenclever, L., Tirumala, D., Schwarz, J., Desjardins, G., Czarnecki, W. M., Teh, Y. W., Pascanu, R., and Heess, N. Information asymmetry in kl-regularized rl. In *Proceedings of the International Conference on Representation Learning*, 2019.

Genewein, T., Leibfried, F., Grau-Moya, J., and Braun, D. A. Bounded rationality, abstraction, and hierarchical decision-making: An information-theoretic optimality principle. *Frontiers in Robotics and AI*, 2:27, 2015.

Gottwald, S. and Braun, D. A. Systems of bounded rational agents with information-theoretic constraints. *Neural computation*, 31(2):440–476, 2019.

Grau-Moya, J., Leibfried, F., and Vrancx, P. Soft q-learning with mutual-information regularization. In *Proceedings of the International Conference on Learning Representations*, 2019.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1352–1361. JMLR. org, 2017.

Hihn, H., Gottwald, S., and Braun, D. A. Bounded rational decision-making with adaptive neural network priors. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pp. 213–225. Springer, 2018.

Hihn, H., Gottwald, S., and Braun, D. A. An information-theoretic on-line learning principle for specialization in hierarchical decision-making systems. In *Proceedings of the 2019 IEEE Conference on Decision-Making and Control (CDC)*, 2019.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Jankowski, N., Duch, W., and Grkabczewski, K. *Meta-learning in computational intelligence*, volume 358. Springer Science & Business Media, 2011.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Representation Learning*, 2014.

Koch, G., Zemel, R., and Salakhutdinov, R. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, 2015.

Konda, V. R. and Tsitsiklis, J. N. Actor-critic algorithms. In *Advances in neural information processing systems*, pp. 1008–1014, 2000.

Lake, B., Salakhutdinov, R., Gross, J., and Tenenbaum, J. One shot learning of simple visual concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33, 2011.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Lan, L., Li, Z., Guan, X., and Wang, P. Meta reinforcement learning with task embedding and shared policy. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2019.

Leibfried, F., Grau-Moya, J., and Ammar, H. B. An information-theoretic optimality principle for deep reinforcement learning. In *Deep Reinforcement Learning Workshop NIPS 2018*, 2017.

Lindig-Leon, C., Gottwald, S., and Braun, D. A. Analyzing abstraction and hierarchical decision-making in absolute identification by information-theoretic bounded rationality. *Frontiers in Neuroscience*, 13:1230, 2019.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Ortega, P. A. and Braun, D. A. Thermodynamics as a theory of decision-making with information-processing costs. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 469(2153), 2013. ISSN 1364-5021.

Ortega, P. A., Wang, J. X., Rowland, M., Genewein, T., Kurth-Nelson, Z., Pascanu, R., Heess, N., Veness, J., Pritzel, A., Sprechmann, P., et al. Meta-learning of sequential strategies. *arXiv preprint arXiv:1905.03030*, 2019.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *Proceedings of the International Conference on Learning Representations*, 2017.

Schach, S., Gottwald, S., and Braun, D. A. Quantifying motor task performance by bounded rational decision theory. *Frontiers in neuroscience*, 12, 2018.

Schmidhuber, J. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

Schmidhuber, J., Zhao, J., and Wiering, M. Shifting inductive bias with success-story algorithm, adaptive levin search, and incremental self-improvement. *Machine Learning*, 28(1):105–130, 1997.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 4077–4087, 2017.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Sutton, R. S. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in neural information processing systems*, pp. 1038–1044, 1996.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

Thrun, S. and Pratt, L. *Learning to learn*. Springer Science & Business Media, 2012.

Vezhnevets, A. S., Wu, Y., Leblond, R., and Leibo, J. Options as responses: Grounding behavioural hierarchies in multi-agent rl. *arXiv preprint arXiv:1906.01470*, 2019.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pp. 3630–3638, 2016.

Von Neumann, J. and Morgenstern, O. *Theory of games and economic behavior (commemorative edition)*. Princeton university press, 2007.

Watkins, C. J. and Dayan, P. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Yao, H., Wei, Y., Huang, J., and Li, Z. Hierarchically structured meta-learning. In *Proceedings of the International Conference on Machine Learning*, pp. 7045–7054, 2019.

Zintgraf, L. M., Shiarlis, K., Kurin, V., Hofmann, K., and Whiteson, S. Caml: Fast context adaptation via meta-learning. In *Proceedings of the Internationcal Conference on Learning Representations*, 2018.