Meta-learning from sparse recovery

Anonymous Author(s)
Affiliation
Address
email

Abstract

Meta-learning aims to train a model on various tasks so that given sample data from a task, even if unforeseen, it can adapt fast and perform well. We apply techniques from compressed sensing to shed light on the effect of inner-loop regularization in meta-learning, with an algorithm that minimizes cross-task interference without compromising weight-sharing. In our algorithm, which is representative of numerous similar variations, the model is explicitly trained such that upon adding a pertinent sparse output layer, it can perform well on a new task with very few number of updates, where cross-task interference is minimized by the sparse recovery of output layer. We demonstrate that this approach produces good results on few-shot regression, classification and reinforcement learning, with several benefits in terms of training efficiency, stability and generalization.

1 Introduction

It's been a long-lasting dream to have artificial intelligence leverage prior knowledge to learn new things quickly. The approaches to this problem of 'meta-learning' can be largely divided into metric-based, model-based and optimization-based [1]. There are naturally two extremes: complete multiplexing between task-specific models, which ignores connection between tasks and defeats the purpose of meta-learning, and complete weight-sharing among all tasks, which hampers expressivity. Methods between the two extremes inevitably suffer from certain degree of cross-task interference and limited representational capacity. A popular branch centering model-agnostic meta-learning (MAML) [2] utilizes a bi-level optimization to explicitly optimize (as outer-loop) the performance of the meta-network after a few-shot training (as inner-loop) on the task. When the inner loop involves multiple steps of update, the gradient propagation to the outer loop becomes inefficient as high-order terms are no longer negligible, which also causes training instability. While there are ways to alleviate the problem [3, 4, 5], it cannot be eradicated.

In this work, with an algorithm complementing MAML with sparse recovery, we show both analytically and numerically that in certain scenarios, the high-order derivatives in the gradient propagation from inner-loop to outer-loop can be safely ignored, and in more general cases, the simple use of sparse recovery significantly improves the training dynamics. It is particularly advantageous on deep and wide neural networks, where high dimensionality turns out to be a blessing. Most importantly, it provides a framework where we can apply techniques from the compressed sensing literature and sheds light on the effect of inner-loop regularization for meta-learning.

Intuitively, all layers up to the final one can be viewed as providing pertinent features that enable linear decision boundaries. Our meta-learner aims to maintain the most efficient collection of common features shared among various tasks, as well as their subfeatures if skip-connections are enabled. As in compressed sensing, the $L1$ regularization allows us to efficiently recover the linear mapping from these features to network output (even with correctness guarantee if some other criteria are met), which well serves the purpose of inner-loop update as it screens neurons for the irrelevant features away from the meta-gradient backpropagation. The effects of high-dimensionality scales in a similar

Submitted to the 5th Workshop on Meta-Learning at NeurIPS 2021, held virtually. Do not distribute.

way as in compressed sensing, where the very high number of parameters almost make sure the features are orthogonal to each other on the relatively few samples for inner-loop optimization. The recovery scheme used as inner-loop optimization is thus robust to noise on the meta-network weights, which subsequently improves gradient propagation efficiency and training dynamics.

2 Preliminaries

2.1 Meta-learning

Given a distribution of tasks $\mathcal{T}_i \sim p(\mathcal{T})$, meta-learning aims to train a model that can adapt quickly on a new task $\mathcal{T}$, leveraging what is learnt during the meta-training on other tasks as well as a small amount of data $\mathcal{D}$ on the new task. Compared to various algorithms for meta-learning [6, 7, 8, 9, 10, 11], e.g. using matching networks [12] or task inference [13], MAML [2] uses a bi-level optimization to explicitly optimize for a network parametrized by θ that is a few updates away from the optimal network for all the tasks. In MAML, the inner-loop optimization samples a task $\mathcal{T}$, trains on the relevant data $\mathcal{D}$ with initialization at θ and obtains a network parametrized by ϕ, whereas the outer-loop optimization propagates the loss gradient at ϕ to θ and thereby improves the parameters of the meta-network. If the inner-loop takes one-step updates with step size α, the meta-objective is simply:

$$\min_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_\phi) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)}) \tag{1}$$

where $\mathcal{L}_{\mathcal{T}_i}(f_\phi)$ is the loss function given the input-output mapping parametrized by ϕ.

When multi-step inner-loop optimization is wanted, we will need to compute higher-order derivatives along the optimization path, which may incur significantly more computational cost. There have been variations to MAML that aim to mitigate this problem, such as Reptile [4] where the outer-loop gradient is simply an average of inner-loop gradients and iMAML [5] where the outer-loop gradient can be semi-analytically computed.

2.2 Compressed sensing

Compressed sensing (CS) [14, 15] aims to recover a signal $x \in \mathbf{R}^n$ with some low-dimensional structure from corrupted measurements $y = \Phi x + e$, where $\Phi \in \mathbf{R}^{m \times n}$ is the measurement matrix (typically $m < n$) and e is an arbitrary vector of error (e.g. Gaussian noise). The low dimensional structure of x can be formulated as $x = f_\psi(s)$ where $s \in l$ and $||s||_0 \ll m$. $f_\psi(s)$ can be a linear mapping, as in the classical problem of reconstructing a frequency-sparse signal from few measurements in time domain, where f_ψ is Fourier transformation and Φ is the sampling in time domain (e.g. convolution with δ or sinusoidal function).

In the most classical case where the measurement matrix Φ and the sparsity transformation f_ψ are both linear, the problem reduces to recovering signal s from $y = Fs + e$ where $F \in \mathbf{R}^{m \times l}$. CS is particularly powerful in the regime where $m \ll l$ and $m < \dim(y) < l$, namely when we want to reconstruct a high-dimensional signal with low-dimensional structure from a moderate number of measurements. This underdetermined problem can be solved by leveraging our prior knowledge that s is sparse. In particular, when F satisfies the Restricted Isometry Property (RIP)[16]:

$$(1-\delta)||s_1 - s_2||_2^2 \leq ||F(s_1 - s_2)||_2^2 \leq (1+\delta)||s_1 - s_2||_2^2$$

with $\delta < \sqrt{2} - 1$, the sparse solution can be perfectly recovered by the convex optimization problem

$$\min_{s} \quad ||s||_1 \ : \ ||y - Fs||_2 \leq ||e||_2 \tag{2}$$

There has been enormous literature on CS, e.g. extending the method for nonlinear measurements [17] and structural prior [18] instead of $L1$, and implementing the idea on neural network [19, 20]. With the method we propose, many of the rich results from CS can be applied to meta-learning for bounding errors and improving stability.

3 Meta-learning from sparse recovery

All networks mapping input to output can be viewed as first mapping input to a layer of features (complex) and then mapping the features to the output (simple). If we view the meta agent as carrying a pool of good features with shared structure for the collection of tasks, then the few-shot learning problem reduces to finding the correct features from the pool to use or fine-tune for a particular task. Compared to MAML where the meta-learnt network is supposed to perform well upon very few updates, here we expect meta-learning from sparse recovery (MLSR) to perform well upon adding a pertinent output layer, which can be obtained through sparse recovery.

The hope for meta-learning is that optimal networks for different tasks have a significant amount of shared structure. For practical tasks, it is conceivable that different tasks can have drastically different output layers and therefore, the structure sharing is mainly in the layers for the features and less for the output layer, which provides one of the motivations for MLSR. This is increasingly true when the dimension gets high and in particular, when hidden layer right before output is very wide and the output layer naturally becomes sparse for one particular task. Intuitively, if the inner loop update can select the optimal sparse output layer, then the meta-update, namely the outer-loop update that improves our meta-learnt network, will propagate the loss only to those features relevant for this particular task as the final layer weights for the irrelevant features are zeroed, which thus minimizes cross-task interference. Here weight-sharing is not compromised since each feature can potentially be densely connected to all the neurons in the preceding layer without penalty.

Algorithm 1: Meta-learning from sparse recovery

α: step size hypeparameter;
$p(\mathcal{T})$:distribution of tasks;
Randomly initialize θ,ϕ;
while *not done* **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T}_i)$;
  forall $\mathcal{T}_i$ **do**
    Compute features $f_\theta(x)$;
    Compute ϕ_i from sparse recovery given features and target;
    Update $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\theta, \phi)$;

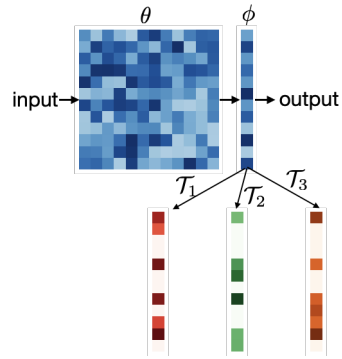

Figure 1: Diagram of MLSR

Technically, this procedure corresponds to the diagram in Figure 3, where θ is the meta-network and ϕ is the result from inner-loop optimization and can take completely different values for different tasks. Given a meta-network, we do a sparse recovery to find the optimal output layer. The exact method of sparse recovery can be selected for particular problem, where the most rudimentary one is simply adding a $L1$ regularization term to the inner-loop update objective. Denote the output layer weight parameters as ϕ, then the inner-loop update using $L1$ norm would be:

$$\min_{\phi} \quad \mathcal{L}_I(\theta, \phi) \equiv \mathcal{L}_{\mathcal{T}_i}(\theta, \phi) + \lambda ||\phi||_1 \tag{3}$$

where $\mathcal{L}_{\mathcal{T}_i}(\theta, \phi)$ denotes the loss of the network parametrized by θ, ϕ on the dataset for task $\mathcal{T}_i$.

Denote the optimal ϕ for task $\mathcal{T}_i$ as ϕ_i. The outer loop is then

$$\min_{\theta} \quad \mathcal{L}_O(\theta) \equiv \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(\theta, \phi_i) \tag{4}$$

To get the full gradient for outer loop, the tricky part would be the implicit θ dependence in ϕ. Namely, the $\frac{\partial \phi}{\partial \theta}$ term in $\frac{d\mathcal{L}}{d\theta} = \frac{\partial \mathcal{L}}{\partial \theta} + \frac{\partial \mathcal{L}}{\partial \phi}\frac{\partial \phi}{\partial \theta}$. However, here we can apply the result from compressive sensing that when the variation in θ satisfies certain criteria (Theorem 1), the sparse recovery of ϕ is robust to noise and effect of θ dependence in ϕ is negligible.

Theorem 1 *Denote the mapping from input x to features as $F(\theta, x)$ and the final-layer mapping with weights Φ such that $y = F(\theta, x)\Phi$. If $F(\theta, x)$ satisfies RIP with isometry constant $\delta_{2s} < \sqrt{2} - 1$*

and for given x, y, θ_1 we have solution Φ_1 such that $y = F(\theta_1, x)\Phi_1$, then upon updating θ_1 to $\theta_2 = \theta_1 + \delta\theta$, the new solution Φ_2 will satisfy:

$$||\Phi_2 - \Phi_1||_2 \leq C_0 s^{-1/2} ||\Phi_1 - \Phi_{1s}||_1 + C_1 \epsilon \tag{5}$$

where Φ_{1s} is the s-sparse approximation to Φ_1, $\epsilon = ||(F(\theta_2, x) - F(\theta_1, x))\Phi_2||_2$ and C_0, C_1 are δ_{2s}-dependent constants that can be made very small (e.g. when the dimension of Φ is high). If Φ_1 is indeed s-sparse, then the error incurred by omitting $\frac{\partial \mathcal{L}}{\partial \Phi}\frac{\partial \Phi}{\partial \theta}$ in $\frac{d\mathcal{L}}{d\theta} = \frac{\partial \mathcal{L}}{\partial F(\theta,x)}\frac{\partial F(\theta,x)}{\partial \theta} + \frac{\partial \mathcal{L}}{\partial \Phi}\frac{\partial \Phi}{\partial \theta}$ can be bounded as:

$$||\sum_i \frac{\partial \mathcal{L}}{\partial \Phi_i}\frac{\partial \Phi_i}{\partial \theta_k}|| \leq C_1 \sum_{i,j,n} ||\frac{\partial \mathcal{L}}{\partial \Phi_i}\Phi_j \frac{\partial F(\theta, x)_{n,j}}{\partial \theta_k}|| \tag{6}$$

Details on C_0, C_1, implication and derivation can be found in supplementary materials. The important thing to note is that C_1 can be made very small for small isometry constants e.g. when the dimension of Φ is high, thus alleviating the pain to track the gradient on θ when updating Φ in the inner-loop optimization.

In the view of implicit gradient, if ϕ is the optimal solution to Eq. 3 for a given θ, then the gradient is

$$\frac{d\phi}{d\theta} = (\mathbf{I} + \frac{1}{\lambda}\nabla^2_\phi \mathcal{L}_I(\theta, \phi))^{-1} = (\mathbf{I} + \frac{1}{\lambda}\nabla^2_\phi \mathcal{L}(\theta, \phi))^{-1}, \tag{7}$$

where the $L1$ norm does not enter the gradient expression at a given θ, ϕ.

One obvious benefit is that since θ is not changed during inner-loop update, we don't need to track the gradient along a particular optimization path during inner loop when the θ dependence in ϕ can be neglected. If the accurate gradient information is wanted, this procedure is compatible with the various techniques for obtaining high-order gradients e.g. the implicit gradient method.

Note that here the sparse recovery method may take other forms to better exploit what we know about the problem. For example, for supervised-learning of mapping $\hat{f}(x)$, when the noise is known to be sparse, the recovery routine can be chosen to be [21]:

$$\min_\phi \quad \mathcal{L}_I(\theta, \phi) = ||f_{\theta,\phi}(x) - \hat{f}(x)||_1 \tag{8}$$

Furthermore, MLSR can be extended into a hierarchical fashion or implemented with skip-connects, which would make the dichotomy of feature layer and output layer softer, more general and potentially more conducive for weight-sharing. It can also be combined with the various versions of MAML as a tri-level optimization, where the sparse recovery of output layer helps to propagate the loss gradient to more relevant weights in the previous layers.

To show the advantage of MLSR and verify our understanding, we run numerical experiments in the following domains.

3.1 Supervised regression and classification

In supervised regression, we have a family of functions and seek a meta-network that can well approximate a sampled function with few input-output pairs. For example, in the classical problem of signal reconstruction, the family of functions may be signals that are relatively sparse in frequency domain (e.g. sound) or wavelet domain (e.g. image). The practical problem is to reconstruct such a signal with very little information. For classification, the problem is very similar except that the output is typically discrete.

For regression, in general, one may use the mean squared error (MSE) with $L1$ regularization:

$$\mathcal{L}_{\mathcal{T}_i}(\theta, \phi) = \sum_{x^{(j)}, y^{(j)} \sim \mathcal{T}_i} ||f_{\theta,\phi}(x^{(j)}) - y^{(j)}||_2^2 + \lambda ||\phi||_1 \tag{9}$$

or iterative hard/soft thresholding when convenient. When the noise is sparse, one may directly use the mean absolute error (MAE):

$$\mathcal{L}_{\mathcal{T}_i}(\theta, \phi) = \sum_{x^{(j)}, y^{(j)} \sim \mathcal{T}_i} ||f_{\theta,\phi}(x^{(j)}) - y^{(j)}||_1 \tag{10}$$

For discrete classification, one may use cross-entropy loss with $L1$ regularization:

$$\mathcal{L}_{\mathcal{T}_i}(\theta, \phi) = \sum_{x^{(j)}, y^{(j)} \sim \mathcal{T}_i} \sum_{k \sim D_i} y_k^{(j)} \log f_{\theta,\phi}(x^{(j)})_k + \lambda ||\phi||_1 \tag{11}$$

where $k \sim D_i$ index the classes for task $\mathcal{T}_i$. Since the classification involves applying a softmax after the final layer, strictly speaking, it is no longer a sparse recovery in the context of compressed sensing, but the insights obtained from the compressed sensing analysis still apply.

For both regression and classification, the $L1$ regularization part may be substituted by constraining ϕ to be the output of a generative model with low-dimensional input [20, 19].

While the training on the support set involves the use of regularization encouraging sparsity, the few-shot training on the target set will proceed without such regularization, so as to eliminate the confounding factor. Note that MLSR is at a disadvantage in this way since it is explicitly optimized to perform well after an inner-loop optimization with the regularization, whereas when comparing to MAML, it uses inner-loop optimization without the regularization.

3.2 Reinforcement learning

Reinforcement learning (RL) typically involves an agent actively interacting with the environment and meta RL seeks an agent that performs well on a new task after few interactions with the environment, where one of the most important problems is data efficiency. While certain techniques from adaptive sensing do apply in the context of MLSR, algorithms explicitly optimized for off-policy experience and efficient exploration are more relevant [22, 23]. Meta RL can be implemented in various models ranging from black-box adaptation with recurrent neural network to optimization-based inference with MAML [9, 24]. Applying sparse recovery before propagating the gradient to the meta-network is in general compatible with these methods, where one simply optimizes the output layer weight in the beginning of each inner-loop optimization.

In the context of meta imitation learning, the goal is similar except that the agent is learning from demonstrations of an 'expert', which typically allows supervised algorithms and MLSR is more relevant.

4 Experiments

4.1 Few-shot regression

If we know the bases of the domain where the signal is sparse, then many arguments from compressed sensing are directly transferable and MLSR is naturally advantageous. For example, if the features are sinusoidal functions and the output is a frequency-sparse signal, there is a theoretical guarantee that the correct signal can be recovered by some form of $L1$ regularization. The usefulness of meta-learning arises particularly when the optimal features are unknown. For that purpose, we test MLSR on sinusoidal regression.

During our training and testing process, we generate our tasks by sampling the frequency of a sinusoidal function. The frequencies are uniformly sampled from $[0.5, 2]$. For each training set in a specific task, we sample the target function at arguments x uniformly across $[-5.0, 5.0]$. There are 20 samples in each training set. The network we use is the same as the one used in Ref. [2], which has 2 hidden layers of size 40 with ReLU nonlinearities. The input and output of the network both have dimension 1. To compare with MAML, we train the network with a fixed step size $\alpha = 0.01$ and use Adam as the meta-optimizer for all of our algorithms. We calculate the mean-squared error of the output with respect to the ground truth as a metric for evaluation.

Our baseline is the network with random initialized weights. In addition to the MLSR algorithm we described in Section 3, we provide a comparison to MAML freezing all layers except for the last layer (MAML-FZ) for the inner loop training, which would have the same number of inner-loop trainable parameters as MLSR. We also implement a multi-gradient-step MAML (Multi-MAML) and compare it with the multi-gradient-step MLSR (Multi-MLSR). We keep both the random network and the MAML as baseline in both cases.

In Fig. 2, we plot one example of the sinusoidal regression task. The tuning of meta-networks obtained from MAML and MLSR is significantly more efficient than random network, where the qualities are

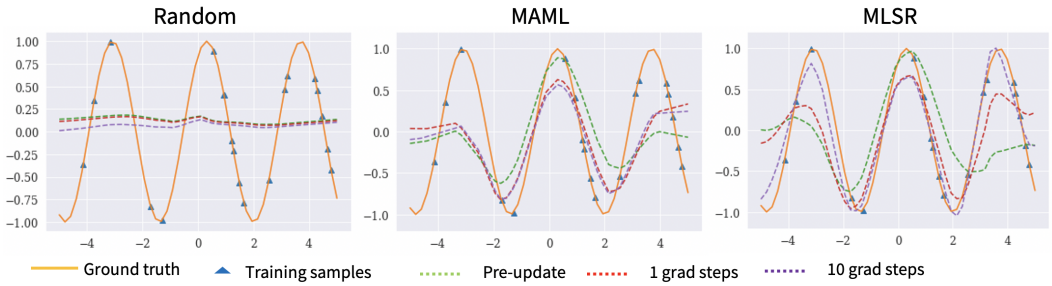

Figure 2: Qualitative comparison for tasks sampled from sinusoidal regression tasks (with different frequencies). The orange lines are the ground truth. The green lines are the pre-update result after we train the model. We can understand it as a pre-conditioner. The red dash lines are the result after one step iteration. The purple dash lines represent the results after 10 gradient steps.

comparable between MAML and MLSR despite the latter has significantly fewer trainable parameters during inner-loop optimization.

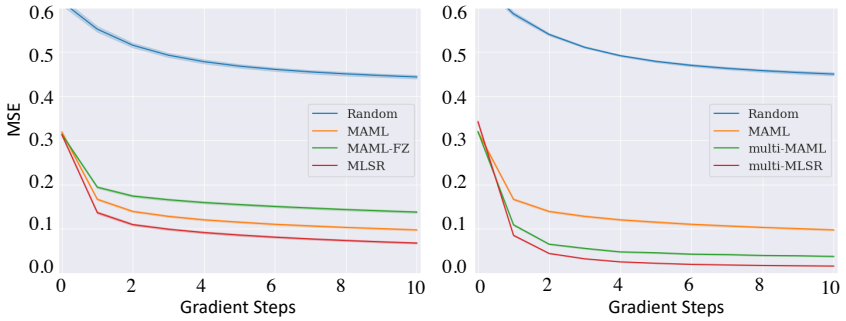

Figure 3: Mean-square error with respect to the gradient steps after the meta-training. Left: one-step gradient update during meta-training. Right: 5-step gradient update during meta-training.

In Fig. 3, we show the comparison of the mean-squared errors between different algorithms with respect to the gradient steps after meta-training. For both the single-step and the multi-step cases, MLSR outperform MAML and random initialization.

4.2 Classification

For a simple illustration, we use the MNIST dataset. We define a pool of tasks where each task is the binary classification among pairs of digits. A sample of these tasks is shown in Fig. 5. Since labels in different tasks do not correspond to the digit of the image, all networks are expected to score 0.5 accuracy without on-sample updates.

The network for classification has two hidden layers with hidden size (100,20) and sigmoid activation. The output is run through a sigmoid activation to yield the probability for class 0 or class 1. The meta-networks are obtained using MAML and MLSR as discussed before, which converges after 100 epochs. We evaluate their adaptability by sampling 1000 new tasks and tune the meta-network in the same way as the inner-loop routine. The meta-training and evaluation are repeated 100 times and the average performance is reported.

In Fig. 5, the three curves show the accuracy improvement of meta-networks obtained from MAML, MLSR and randomization when trained on the target set. The training curves for MAML and MLSR show faster improvement in accuracy than the training curve for random initialization, reflecting the success of meta-learning in terms of fast adaptability. However, the accuracy for the meta-network from MAML does not reach the same height as the accuracy for random network, indicating that meta-network obtained from MAML may be more prone to local minima. This is plausible as MAML explicitly optimizes for meta-networks that perform well for all tasks at one step away in

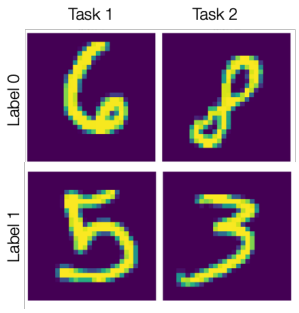

Figure 4: Example of 2 sampled tasks with the labels used

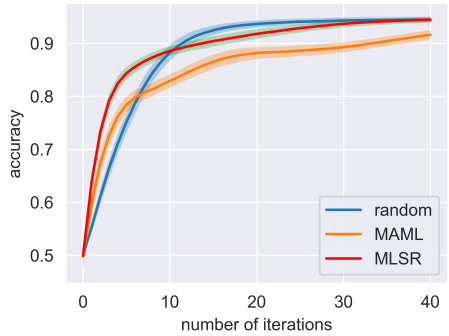

Figure 5: Short-term and long-term training performance on meta-network obtained through MAML and MLSR, compared to random.

the parameter space, without requiring the meta-network to reside in the deepest valley of the loss landscape. It can be understood as a manifestation of inevitable cross-task interference as well, which limits expressivity. In contrast, MLSR meta-network has fast adaptibility without compromising the maximum achievable accuracy, validating our intuition that MLSR minimizes cross-task interference with minimal compromise to weight-sharing.

Note that the training on the target set does not involve regularization, so the advantage of MLSR over MAML completely comes from the initialization. Since MLSR is optimized to perform well for on-sample training with regularization, the advantage of MLSR in this case is unambiguous.

4.3 Reinforcement learning

To demonstrate the general applicability of MLSR in the context of RL, we use the half-cheetah problem in the garage benchmark suite [25], where MLSR is not naturally advantageous due to the unsupervised nature.

The model consists of a two-hidden-layer network of hidden size (32,64) for inner-loop update using vanilla policy gradient (REINFORCE) [26] and a two-hidden-layer network of hidden size (32,32) for meta-optimization. The pool of tasks consists of reinforcement learning problems with various wanted directions of motion for the half-cheetah. Each task has 40 rollout steps. The meta-training has 300 epochs with batch size being 20.

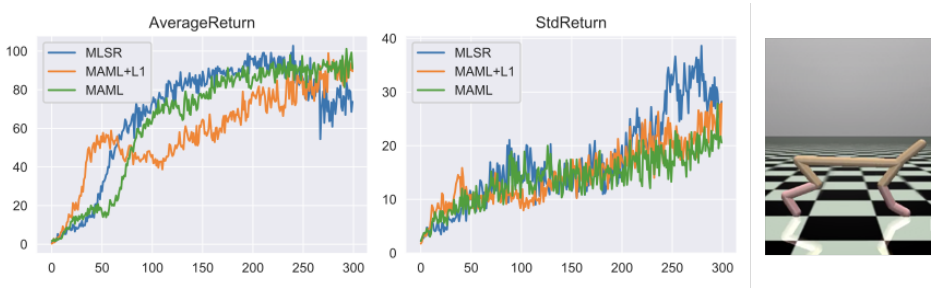

Figure 6: Comparison among MLSR, MAML, $L1$-regularized MAML on the half-cheetah problem with different directions

Figure 6 shows the average return, evaluated at the end of each inner loop update across meta-training epochs. While MLSR where all the layers other than the final layer are frozen has significantly fewer trainable parameters during the inner loop compared to MAML, it is still able to achieve comparable results as MAML and $L1$-regularized MAML.

5 Related work

For most 'descendents' of MAML, our architecture has significant differences while being compatible. Compared to MAML [2] which explicitly seeks a model that lies a few steps away from the optimal networks for all tasks, MLSR seeks a model that performs well upon adding a pertinent output layer. As a result, while the various optimal networks for different tasks in MAML are bound to lie within reachable distance to each other in parameter space for the sake of efficient gradient propagation, the various optimal networks for MLSR can be completely different in the output layer. As for proximity-based meta-parameter search [4, 27], while it also allows multi-step inner-loop updates and proves efficient on some examples, it essentially abandons MAML's core idea of taking the gradient at the end of inner-loop optimization and aims to minimize the expected distance to all solution manifolds, which inevitably suffers in terms of expressivity. MLSR allows multi-step inner-loop updates with a different mechanism, which still takes gradient at the end of inner-loop optimization and propagates it to the meta-network. On top of all, MLSR can be combined with algorithms like MAML and Reptile by extending to bilevel optimization to a trilevel one.

There are indeed works with similar architecture [28, 29, 30, 31] or operational procedure [32]. While they share the benefits of propagating gradients to most relevant 'expert' weights, MLSR exploits the noise robustness of sparse recovery to alleviate the difficulty with large-step inner-loop updates in the context of meta-learning. Nevertheless, many techniques e.g. for improving mixture of experts are transferable. We used the sparse recovery scheme in its simplest form for a clear demonstration of the connection between compressed sensing and meta-learning.

The connection between meta-learning and compressed sensing was previously noted by [20], but in a completely different way where they use the bilevel framework of meta-learning to implement their generative adversarial network for deep compressed sensing.

In general, one can always carefully design clever and sophisticated regularization for certain purposes, such as in [33] where memorization of task information is penalized. MLSR provides a simple remedy to balance between expressivity (less cross-task interference) and inductive bias (more weight sharing between tasks). The selection of specific routine for sparse recovery can be made more sophisticated by applying layer-specific regularization in a hierarchical fashion or using generative-model-based constraint in place of L_1.

6 Discussion and future work

We introduced a meta-learning variant that utilizes sparse recovery and analytically showed that in the high-dimensional regime where the output layer is sparse and the feature mapping satisfies RIP, sparse recovery is robust to noise and allows more efficient gradient propagation. In particular, we applied techniques from compressed sensing to bound the error in the meta-gradient calculation and explained how meta-gradient computation with large inner-loop updates is computationally more affordable. The meta-network obtained as weight initialization can be further adapted with desired data and gradient steps. Practically, our method is compatible with most MAML-like methods, as it simply adds a sparse recovery process at the beginning of inner loop. It can be made more general by allowing skip-connects, hierarchical recovery and multi-level optimization. The sparse recovery can also be extended for structural prior.

Overall, the connection between meta-learning and compressed sensing is a useful resource especially in the high-dimensional setting. The idea to exploit prior knowledge to simplify higher-order gradients may be extended to other forms of regularization. By allowing larger inner-loop updates, meta-learning can be significantly more powerful, with potential application to various fields. Further research on this topic can explore how other forms of prior knowledge may be exploited in meta-learning.

References

[1] Lilian Weng. Meta-learning: Learning to learn fast. *lilianweng.github.io/lil-log*, 2018.

[2] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *34th International Conference on Machine Learning, ICML 2017*, 3:1856–1868, 2017.

[3] Antreas Antoniou, Amos Storkey, and Harrison Edwards. How to train your MAML. *7th International Conference on Learning Representations, ICLR 2019*, pages 1–11, 2019.

[4] Alex Nichol, Joshua Achiam, and John Schulman. On First-Order Meta-Learning Algorithms. pages 1–15, 2018.

[5] Aravind Rajeswaran, Chelsea Finn, Sham Kakade, and Sergey Levine. Meta-Learning with Implicit Gradients. 2019.

[6] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in Neural Information Processing Systems*, 2017-Decem:4078–4088, 2017.

[7] Mark Woodward and Chelsea Finn. Active One-shot Learning. 2016.

[8] Yan Duan, Marcin Andrychowicz, Bradly Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *Advances in Neural Information Processing Systems*, 2017-Decem(Nips):1088–1099, 2017.

[9] Lantao Yu, Tianhe Yu, Chelsea Finn, and Stefano Ermon. Meta-Inverse Reinforcement Learning with Probabilistic Context Variables. (NeurIPS):1–15, 2019.

[10] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pages 1–11, 2019.

[11] Marcin Andrychowicz, Misha Denil, Sergio Gómez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in Neural Information Processing Systems*, (Nips):3988–3996, 2016.

[12] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *Advances in Neural Information Processing Systems*, pages 3637–3645, 2016.

[13] Jan Humplik, Alexandre Galashov, Leonard Hasenclever, Pedro A. Ortega, Yee Whye Teh, and Nicolas Heess. Meta reinforcement learning as task inference. 2019.

[14] Massimo Fornasier and Holger Rauhut. Compressive sensing. *Handbook of Mathematical Methods in Imaging: Volume 1, Second Edition*, (July):205–256, 2015.

[15] Emmanuel J. Candes and Michael B. Wakin. An introduction to compressive sampling: A sensing/sampling paradigm that goes against the common knowledge in data acquisition. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008.

[16] Emmanuel J. Candès. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, 346(9-10):589–592, 2008.

[17] Thomas Blumensath. Compressed sensing with nonlinear observations and related nonlinear optimization problems. *IEEE Transactions on Information Theory*, 59(6):3466–3474, 2013.

[18] Sivan Gleichman and Yonina C. Eldar. Blind compressed sensing. *IEEE Transactions on Information Theory*, 57(10):6958–6975, 2011.

[19] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G. Dimakis. Compressed sensing using generative models. *34th International Conference on Machine Learning, ICML 2017*, 2:822–841, 2017.

[20] Yan Wu, Mihaela Rosea, Timothy Lillicrap, Mihaela Rosca, and Timothy Lillicrap. Deep compressed sensing. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:11872–11883, 2019.

[21] Emmanuel J. Candes and Terence Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

[22] Kate Rakelly, Aurick Zhou, Deirdre Quiilen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:9291–9301, 2019.

[23] Abhishek Gupta, Russell Mendonca, Yu Xuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. *Advances in Neural Information Processing Systems*, 2018-Decem:5302–5311, 2018.

[24] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *7th International Conference on Learning Representations, ICLR 2019*, pages 1–17, 2019.

[25] The garage contributors. Garage: A toolkit for reproducible reinforcement learning research. `https://github.com/rlworkgroup/garage`, 2019.

[26] Ronald J. Willia. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3):229–256, 1992.

[27] Jonas Rothfuss, Tamim Asfour, Dennis Lee, Ignasi Clavera, and Pieter Abbeel. PrOMP: Proximal meta-policy search. *7th International Conference on Learning Representations, ICLR 2019*, pages 1–25, 2019.

[28] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. Modeling Task Relationships in Multi-task Learning with. *Sigkdd*, pages 1930–1939, 2018.

[29] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. 2017.

[30] David Eigen, Marc Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. *2nd International Conference on Learning Representations, ICLR 2014 - Workshop Track Proceedings*, pages 1–8, 2014.

[31] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pages 1–19, 2019.

[32] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples. 2019.

[33] Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-learning without memorization. *arXiv preprint arXiv:1912.03820*, 2019.