Meta-learning with Many Tasks

James T. Kwok

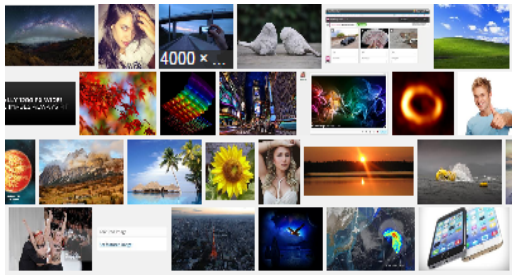
MLA 2023

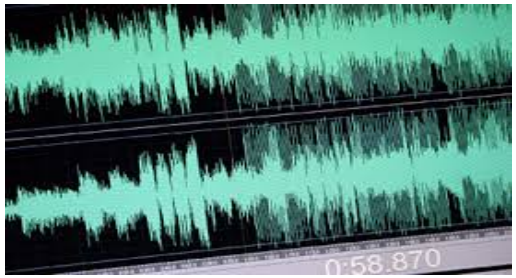
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Outline

- introduction
- weak meta-model: multiple meta-models
- large variance in meta-learning: variance reduction
- conflicting gradients in meta-learning: soft improvement function
- conclusion

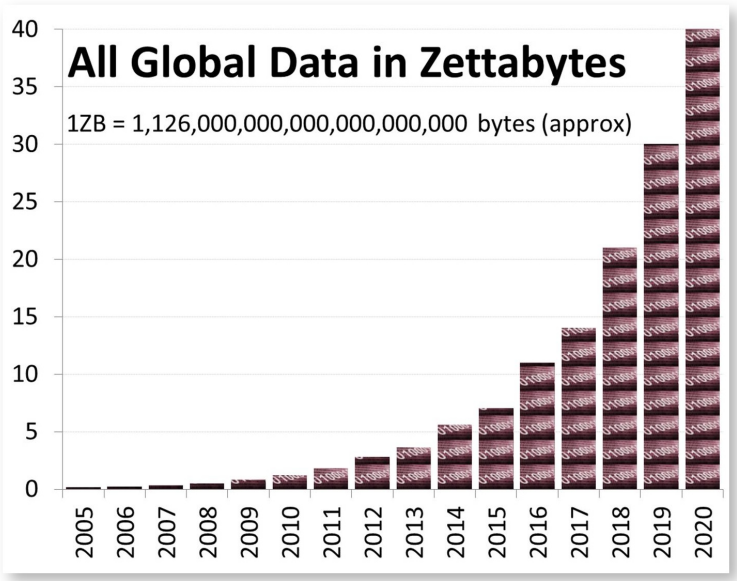
Data, Data, Data

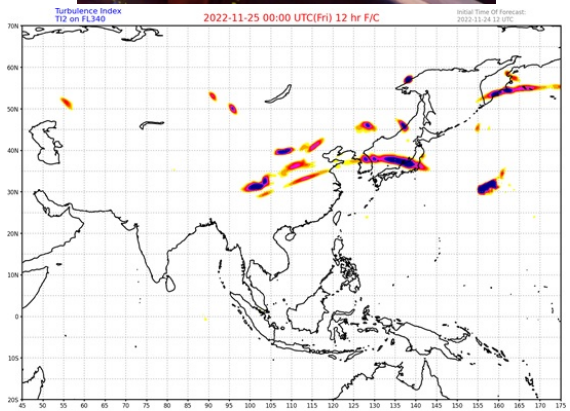




deep networks need huge amount of labeled samples

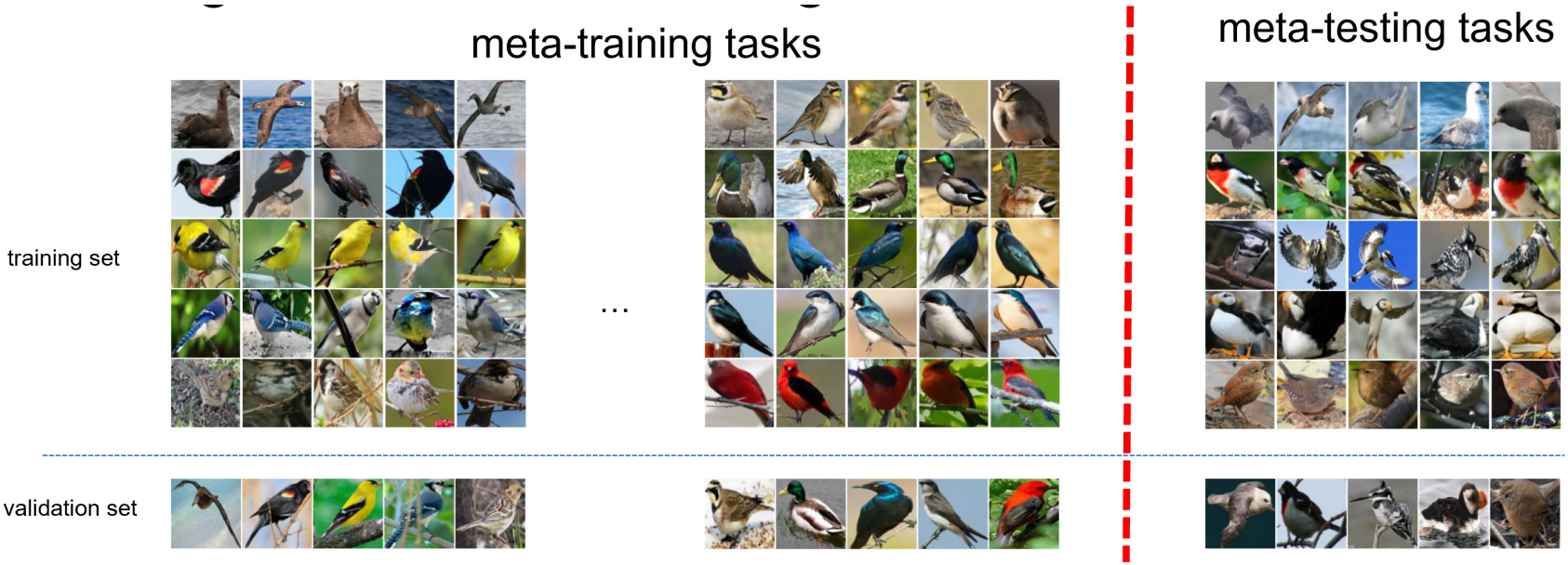
some applications have few labeled samples (e.g., aviation turbulence)





Limited Data

few-shot learning



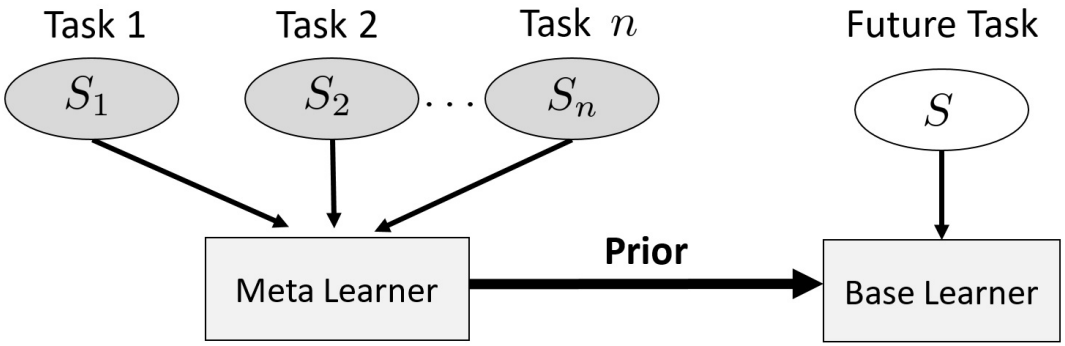
how to learn quickly with limited data?

like humans, take advantage of prior experiences

Meta-Learning (Learning to Learn)

idea

- extract meta-knowledge from learned tasks
- utilize meta-knowledge to learn new tasks more quickly



- meta-learner learns generic information (meta-knowledge) across source tasks
- base learner takes meta-knowledge as prior knowledge, then generalizes for the new task using task-specific information

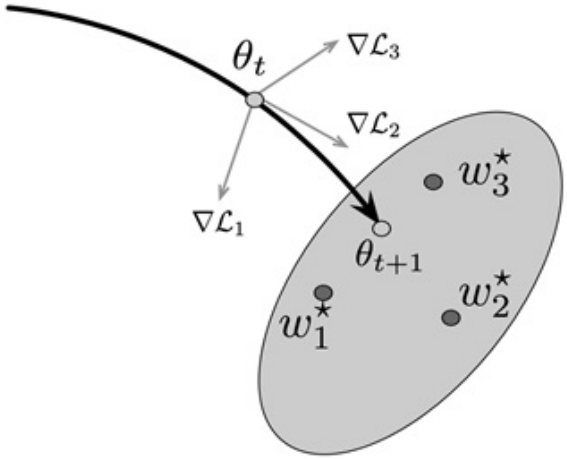
advantages

- data efficiency, compute efficiency, lifelong learning

Model-Agnostic Meta-Learning (MAML) Algorithm

MAML

- learns a meta-initialization $\boldsymbol{w}$ (shared among all tasks) from historical tasks
- fine-tune on new task (using a few gradient updates)



bilevel optimization

$$\min_{\boldsymbol{w}} \quad \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} \mathcal{L}((\boldsymbol{w}, \boldsymbol{\theta}^i(\boldsymbol{w})); \mathcal{D}^i_{\text{vld}})$$

$$\text{s.t.} \quad \boldsymbol{\theta}^i(\boldsymbol{w}) = \arg\min_{\boldsymbol{\theta}} \mathcal{L}((\boldsymbol{w}, \boldsymbol{\theta}); \mathcal{D}^i_{\text{tr}})$$

- outer level: finds a suitable meta-initialization $\boldsymbol{w}$
- inner level: adapts $\boldsymbol{w}$ to produce task-specific $\boldsymbol{\theta}^i(\boldsymbol{w})$

Lots of Tasks

Example (5-way few-shot learning)

miniImageNet dataset

- meta-training set has 64 classes
- number of meta-training tasks: $\binom{64}{5} \approx 7.6 \times 10^6$

tieredImageNet dataset

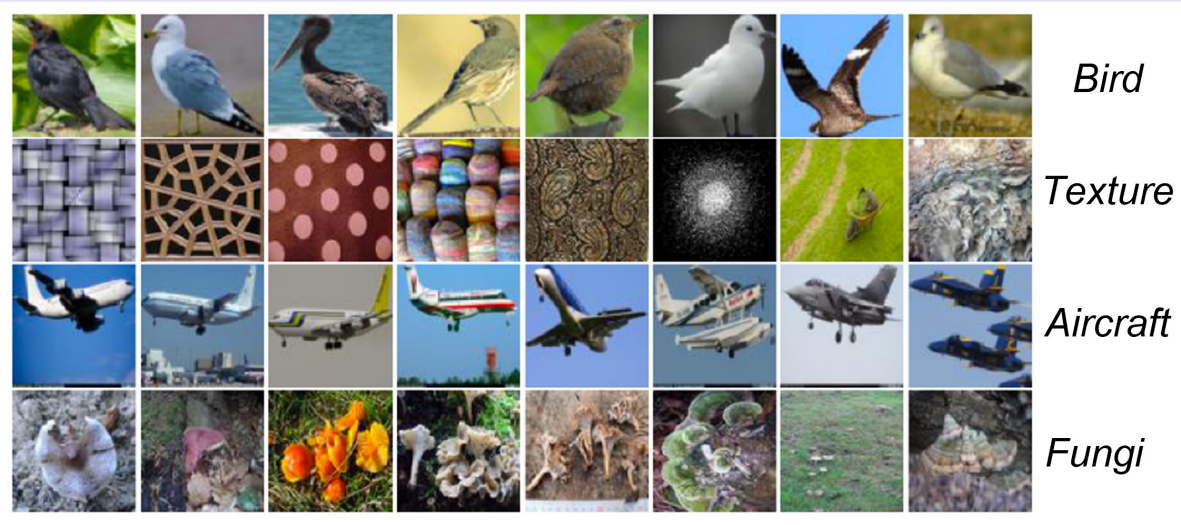
- meta-training set has 351 classes
- total number of meta-training tasks: $\binom{351}{5} \approx 4.3 \times 10^{10}$

Example (recommender system)

- each user is a task
- number of Amazon Prime subscribers: 150 million as of 2020

Problem: Weak Meta-Model

complex environment



diverse task model parameters $\rightarrow$ one meta-model is not sufficient

Problem: Large Variance

large variance

1. data variance: limited data samples for each task
2. task variance: task sampling from task distribution

large variance $\rightarrow$ slow convergence

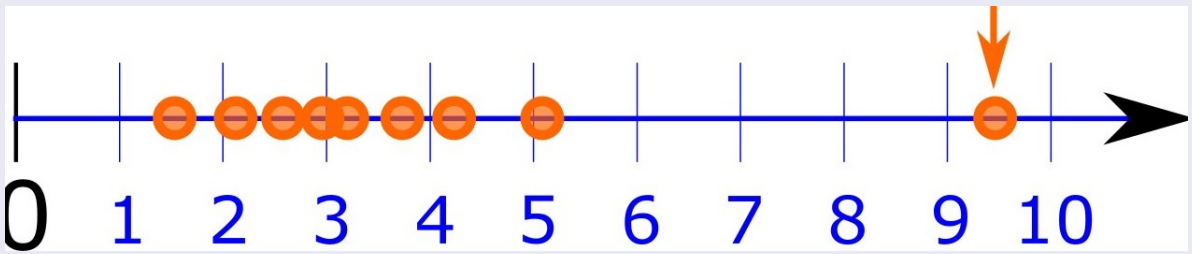
Problem: Conflicting Gradients

conflicting gradients

- meta-learning: minimize average loss in outer level

$$\min_{\boldsymbol{w}} \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \mathcal{L}((\boldsymbol{w}, \boldsymbol{\theta}^i(\boldsymbol{w})); \mathcal{D}^i_{\text{vld}})$$

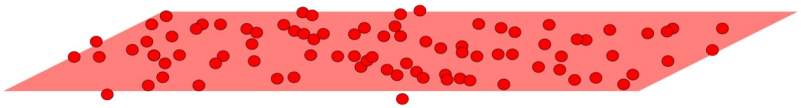
- different task gradients point in different directions
- conflicting gradients: significantly influenced by a small subset of tasks



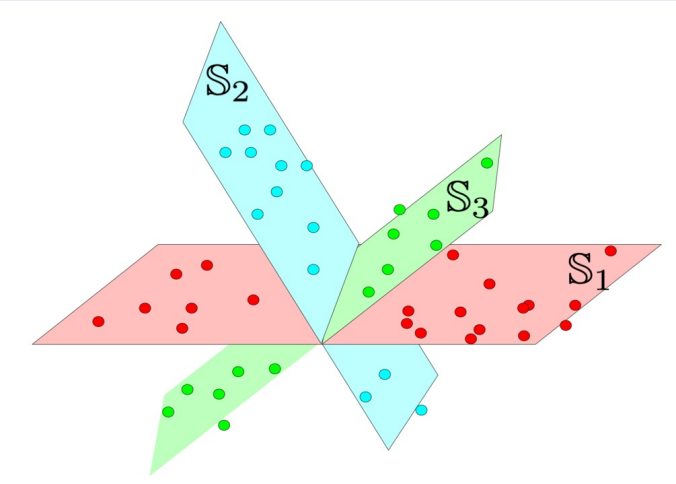
- $\rightarrow$ poor performance

One Meta-Model is not Sufficient

subspace



MUSML (MUltiple Subspaces for Meta-Learning)



multiple subspaces

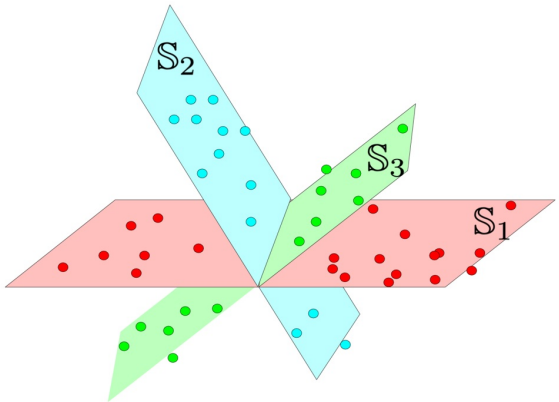
- each subspace: one type of meta-knowledge

more challenging in meta-learning

- bilevel optimization
- unseen tasks with limited samples

[Subspace learning for effective meta-learning (ICML 2022)]

MUSML



base learner

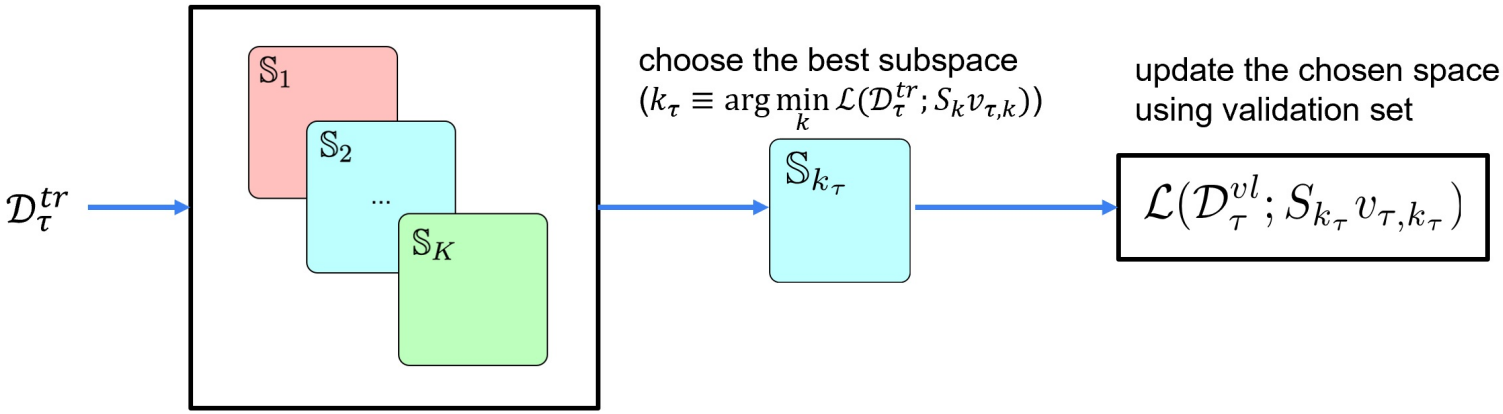
- task parameters $\boldsymbol{w}_\tau$'s for tasks τ's
- lie in K subspaces $\{\mathbb{S}_1, \ldots, \mathbb{S}_K\}$; with basis $\{S_1, \ldots, S_K\}$
- in each subspace $\mathbb{S}_k$, search for a linear combination $\boldsymbol{v}^\star_{\tau,k}$ to form $\boldsymbol{w}_\tau$

$$\boldsymbol{v}^\star_{\tau,k} = \arg\min_{\boldsymbol{v}_\tau \in \mathbb{R}^m} \mathcal{L}(\mathcal{D}^{tr}_\tau; S_k \boldsymbol{v}_\tau)$$

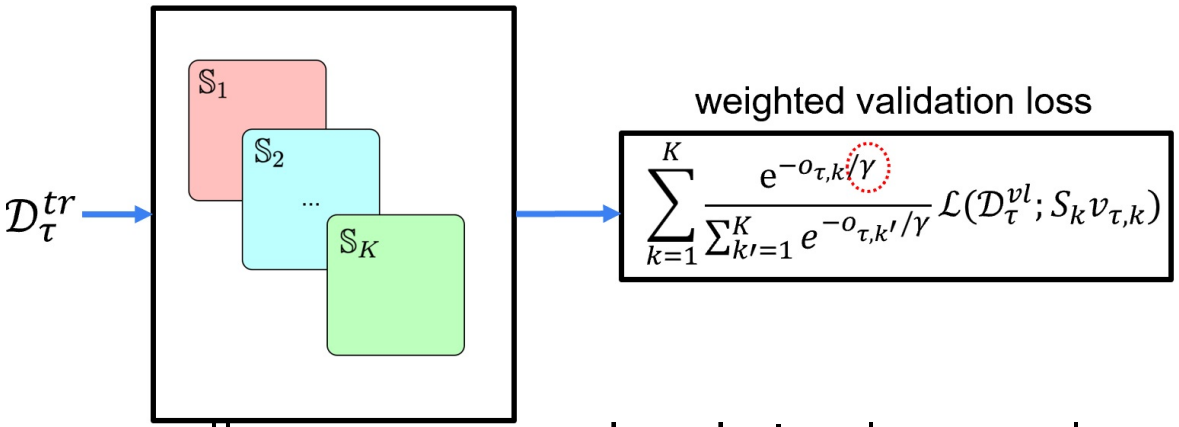
meta-learner

- learn meta-parameters $\{S_1, \ldots, S_K\}$

One-Hot Subspace Selection



- not efficient: only one subspace is updated at each step



$o_{\tau,k}$: training loss when using $\mathbb{S}_k$ to construct τ's model

- all spaces are updated simultaneously
- (one-hot) $0 \leftarrow \gamma \rightarrow \infty$ (uniform)

MUSML Algorithm

for $t = 0, 1, \ldots, T-1$ **do**
  sample a task τ with $\mathcal{D}_\tau^{tr}$ and $\mathcal{D}_\tau^{vl}$;
  base learner:
  for $k = 1, \ldots, K$ **do**
    initialize $\boldsymbol{v}_{\tau,k}^{(0)} = \boldsymbol{v}^{(0)}$;
    for $t' = 0, 1, \ldots, T_{in} - 1$ **do**
      $\boldsymbol{v}_{\tau,k}^{(t'+1)} = \boldsymbol{v}_{\tau,k}^{(t')} - \alpha \nabla_{\boldsymbol{v}_{\tau,k}^{(t')}} \mathcal{L}(\mathcal{D}_\tau^{tr}; \mathsf{S}_{k,t} \boldsymbol{v}_{\tau,k}^{(t')})$;
    end for
    $\boldsymbol{v}_{\tau,k} \equiv \boldsymbol{v}_{\tau,k}^{(T_{in})}$;
    $o_{\tau,k} = \mathcal{L}(\mathcal{D}_\tau^{tr}; \mathsf{S}_{k,t} \boldsymbol{v}_{\tau,k})$;
  end for
  meta-learner:
  $\mathcal{L}_{vl} = \sum_{k=1}^{K} \frac{\exp(-o_{\tau,k}/\gamma_t)}{\sum_{k'=1}^{K} \exp(-o_{\tau,k'}/\gamma_t)} \mathcal{L}(\mathcal{D}_\tau^{vl}; \mathsf{S}_{k,t} \boldsymbol{v}_{\tau,k})$;
  $\{\mathsf{S}_{1,t+1}, \ldots, \mathsf{S}_{k,t+1}\} = \{\mathsf{S}_{1,t}, \ldots, \mathsf{S}_{k,t}\} - \eta_t \nabla_{\{\mathsf{S}_{1,t}, \ldots, \mathsf{S}_{k,t}\}} \mathcal{L}_{vl}$;
end for
Return $\mathsf{S}_{1,T}, \ldots, \mathsf{S}_{K,T}$.

Bound on Expected Excess Risk

$$\mathcal{R}(\mathcal{S}) \leq \mathcal{R}^\star + \rho\sqrt{m}\,\mathbb{E}_{\tau'}\mathbb{E}_{\mathcal{D}^{tr}_{\tau'}}\|\boldsymbol{v}_{\tau',k_{\tau'}} - \boldsymbol{v}^\star_{\tau',k_{\tau'}}\| + \varrho\mathbb{E}_{\tau'}\mathbb{E}_{\mathcal{D}^{tr}_{\tau'}}\mathrm{dist}(\boldsymbol{w}^\star_{\tau'}, \mathbb{S}_{k_{\tau'}}) + K\sqrt{\frac{\nu^2 + 12\varrho\nu(1+m\alpha\delta)^{T_{in}}}{2N_{tr}}}$$

- $\mathrm{dist}(\boldsymbol{w}^\star_{\tau'}, \mathbb{S}_{k_{\tau'}}) \equiv \min_{\boldsymbol{w}\in\mathbb{S}_{k_{\tau'}}}\|\boldsymbol{w} - \boldsymbol{w}^\star_{\tau'}\|$: distance between $\boldsymbol{w}^\star_{\tau'}$ and $\mathbb{S}_{k_{\tau'}}$

1. minimum risk
2. distance between approximate minimizer $\boldsymbol{v}_{\tau',k_{\tau'}}$ and exact minimizer $\boldsymbol{v}^\star_{\tau',k_{\tau'}}$
3. approximation error of $\boldsymbol{w}^\star_{\tau'}$ using the learned subspaces
4. complexity of subspaces (m and K)

Few-shot Classification Experiments

data sets

1. *Meta-Dataset-BTAF*: consists of 4 image classification datasets: Bird, Texture, Aircraft, Fungi
2. *Meta-Dataset-ABF*: consists of Aircraft, Bird, and Fungi
3. *Meta-Dataset-CIO*: consists of CIFAR-FS, mini-ImageNet, Omniglot

architecture

- Conv4 backbone + Prototype classifier

meta-learning baselines

- one globally-shared meta-model: MAML, ProtoNet, ANIL, BMG
- multiple meta-models arranged in hierarchy/graph/clusters
  - Dirichlet process mixture model (DPMM)
  - hierarchically structured meta-learning (HSML)
  - automated relational meta-learning (ARML)
  - task similarity aware meta-learning (TSA-MAML, TSA-ProtoNet)

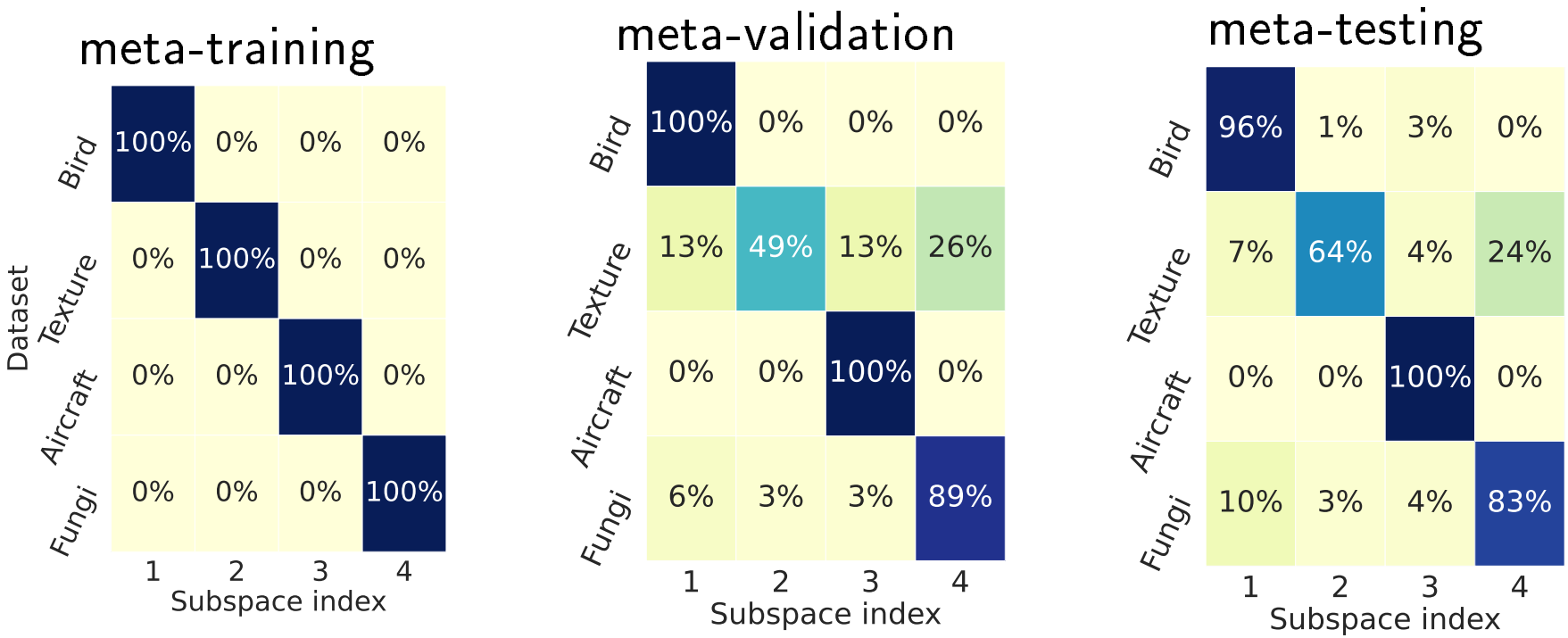
5-way 5-shot Classification Accuracy

	Meta-Dataset-BTAF	*Meta-Dataset-ABF*	*Meta-Dataset-CIO*
MAML	57.78	63.86	74.46
ProtoNet	62.29	65.62	76.51
ANIL	58.57	64.43	74.61
BMG	60.10	65.80	77.46
DPMM	63.00	66.26	76.63
TSA-MAML	63.20	68.17	76.89
HSML	62.39	64.17	75.54
ARML	63.95	64.52	76.12
TSA-ProtoNet	63.57	68.77	77.27
MUSML	66.18	71.10	77.83

- MUSML is more accurate

Task Assignment to Learned Subspaces

Meta-Dataset-BTAF (5-way 5-shot)



- tasks from the same dataset are assigned to the same subspace

Cross-Domain 5-way 5-shot Classification

Meta-Dataset-BTAF $\rightarrow$ *Meta-Dataset-CIO*

MAML	64.25
ProtoNet	66.13
ANIL	65.19
BMG	66.98
DPMM	66.73
TSA-MAML	66.85
HSML	65.18
ARML	65.37
TSA-ProtoNet	66.92
MUSML	67.41

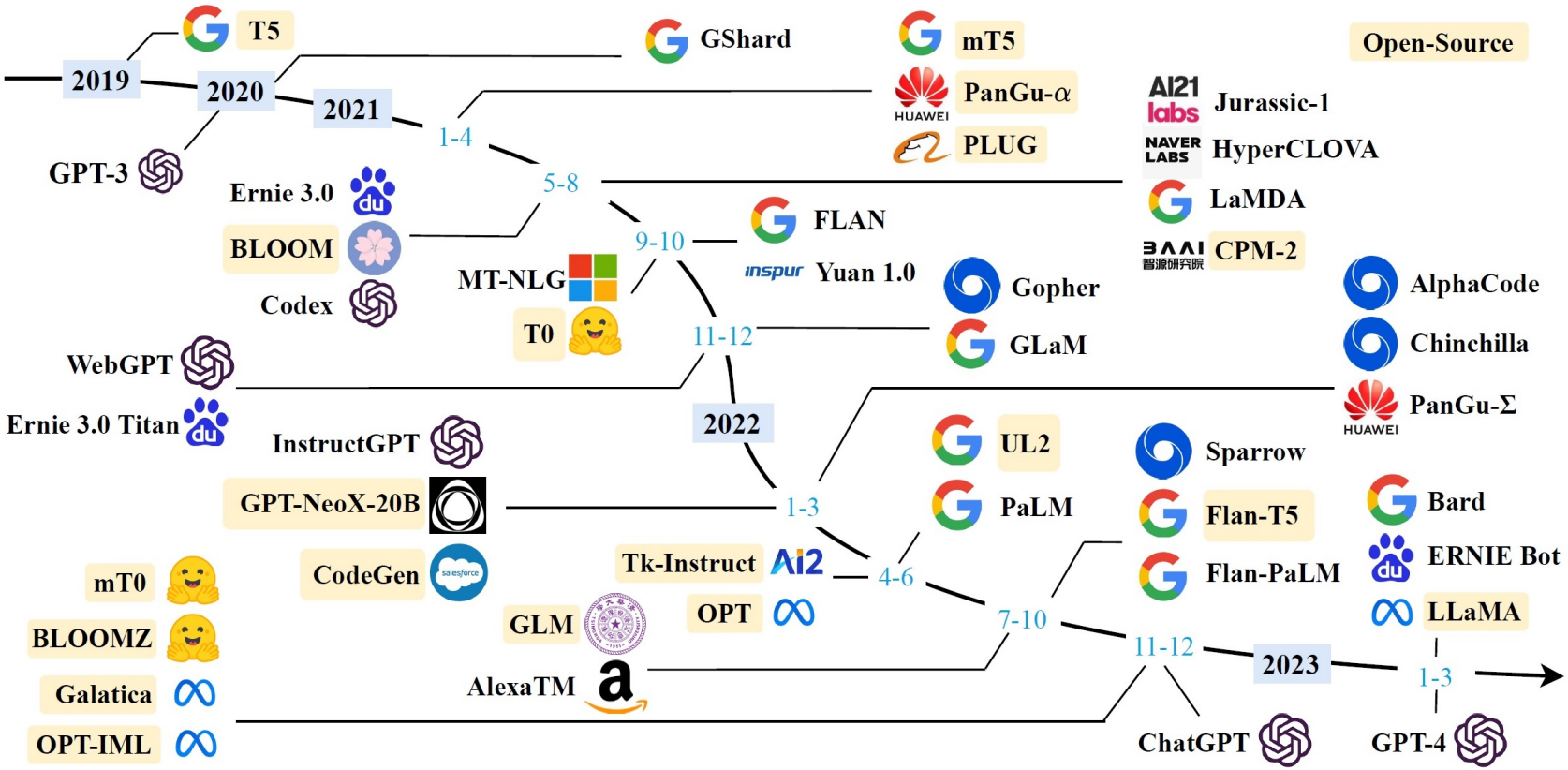
Improving Existing Meta-learning Algorithms

- MUSML can be used with any meta-learning algorithm
- 5-way 5-shot classification accuracy

	Meta-Dataset-BTAF	*Meta-Dataset-ABF*	*Meta-Dataset-CIO*
Meta-SGD	58.93	64.19	75.95
MUSML-SGD	65.72	69.15	77.48
Meta-Curvature	60.02	64.51	76.13
MUSML-Curvature	66.10	69.23	77.96

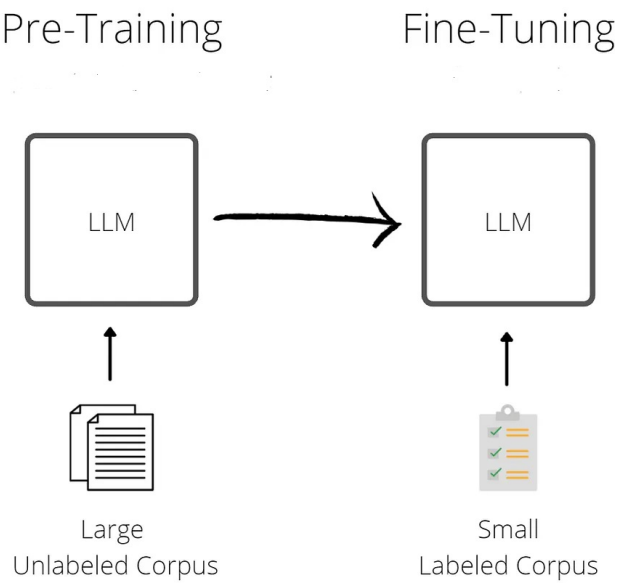
Meta-Learning for Prompt Learning in LLM

large language models



[from "A Survey of Large Language Models", Zhao et al, 20223]

Pre-Train and Fine-Tune



pre-training

- train on a large-scale corpora $\rightarrow$ a very capable LLM

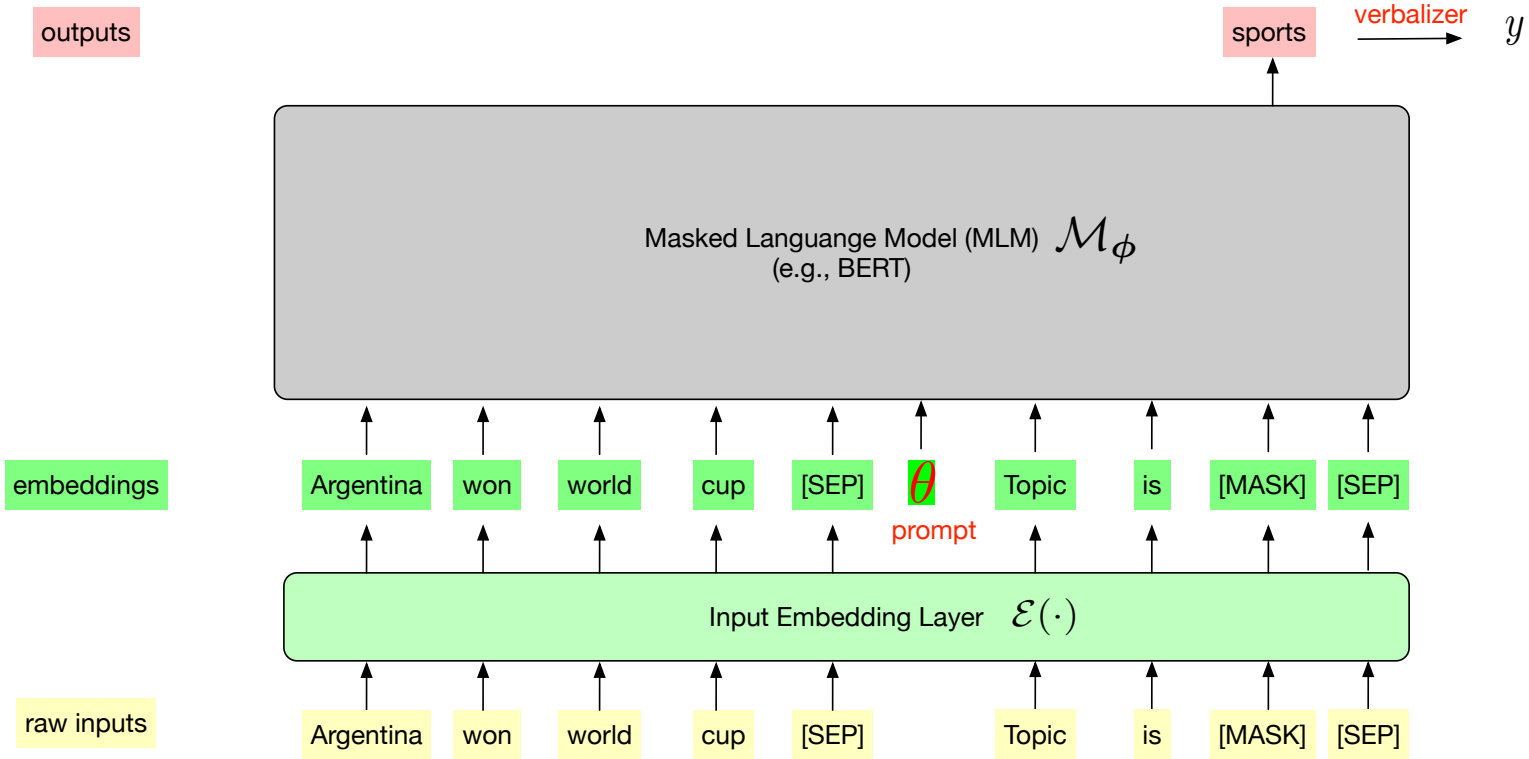
fine-tuning on downstream tasks

- adapt the pre-trained LLM according to specific goals

fine-tune the whole model? expensive for large models
(e.g., GPT-3 contains 100+ billion parameters)

Prompt Tuning

- freeze the pre-trained model
- learn a continuous-valued prompt wrapped into the input embedding

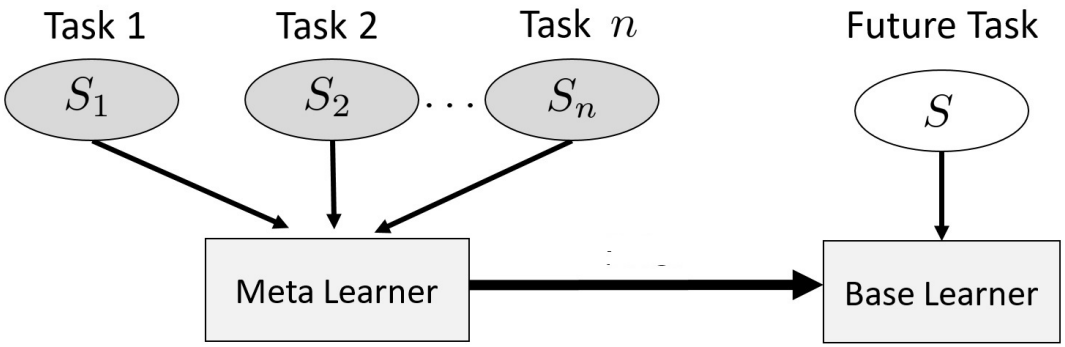


Meta-Learning the Prompt

learning the prompt can be sensitive to initialization

MetaPrompting [Hou et al., 2022]

- learn a meta-initialization for all task-specific prompts



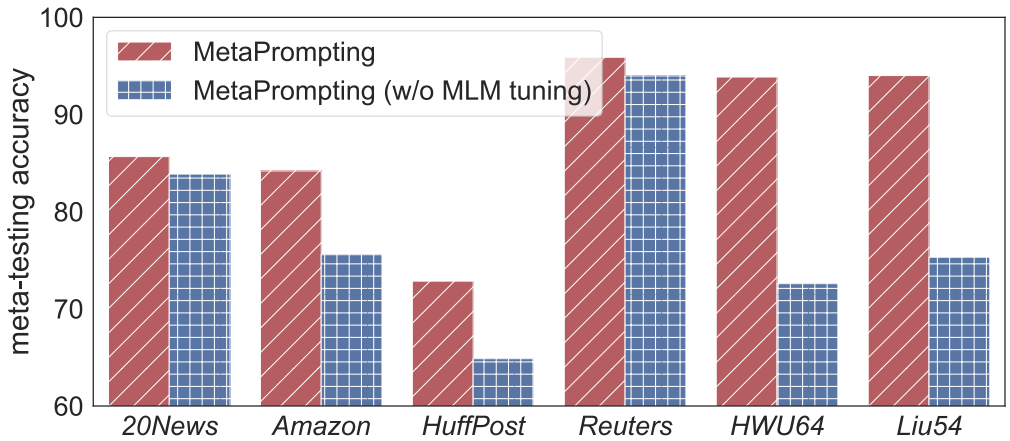
Problems

learn only one single meta-initialized prompt
$\rightarrow$ hard to find such a prompt when the tasks are complex

need to tune the whole LM $\rightarrow$ expensive

5-way 5-shot classification

- with or without MLM tuning



Meta-Learning a Prompt Pool

use a pool of prompts to extract more task knowledge

[Effective structured-prompting by meta-learning and representative verbalizer (ICML 2023)]

- more flexible
  $\rightarrow$ allows better adaptation to complex tasks
  $\rightarrow$ no need to fine-tune the whole LM
- only the prompt pool is tuned $\rightarrow$ parameter-efficient
  - e.g., 1800$\times$ fewer parameters than MetaPrompting
- also proposed a new verbalizer

MetaPrompter

- prompt pool: has K learnable prompts
  - $\{(\boldsymbol{k}_i, \boldsymbol{\theta}_i) : i = 1, \ldots, K\}$, with key $\boldsymbol{k}_i$ and value $\boldsymbol{\theta}_i$
  - shared meta-knowledge learned by meta-learning algorithms (e.g., MAML)
- compute attention between input $\boldsymbol{x}$ and the K prompts
- instance-dependent prompt
  - weighted combinations of all the prompts in the pool via attention

$$\boldsymbol{\theta}_{\boldsymbol{x}}(\mathrm{K}, \boldsymbol{\Theta}) = \sum_{i=1}^{K} a_i(\boldsymbol{x})\boldsymbol{\theta}_i$$

Experiments

- 6 topic classification data sets
- LM: pre-trained BERT

5-way 5-shot classification

	#param ($\times 10^6$)	*20News*	*Amazon*	*HuffPost*	*Reuters*	*HWU64*	*Liu54*
HATT	0.07	55.00	66.00	56.30	56.20	–	–
DS	1.73	68.30	81.10	63.50	96.00	–	–
MLADA	0.73	77.80	86.00	64.90	96.70	–	–
ConstrastNet	109.52	71.74	85.17	65.32	95.33	92.57	93.72
MetaPrompting	109.52	85.67	84.19	72.85	95.89	93.86	94.01
MetaPrompter	0.06	88.57	86.36	74.89	97.63	95.30	95.47

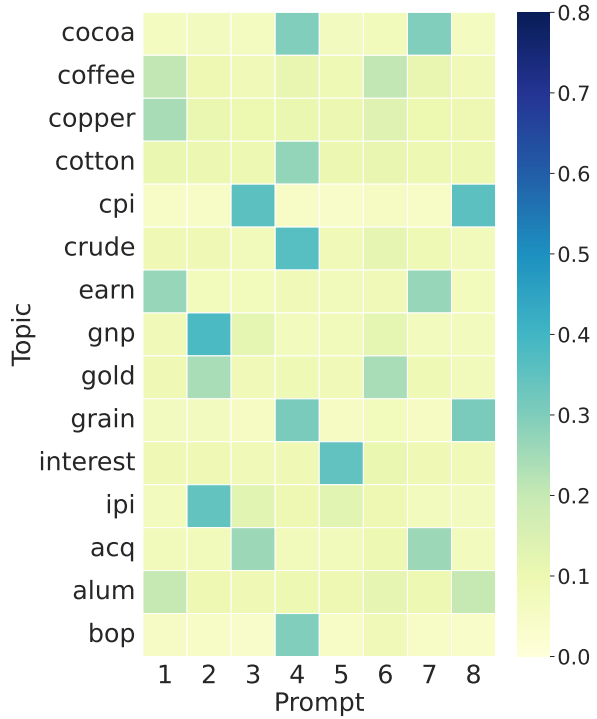
MetaPrompter

- better than both prompt-based and non-prompt-based baselines
- much more parameter-efficient than MetaPrompting

Visualization

samples from each target class prefer prompts whose tokens are related to that class

- samples from *cocoa* tend to use the 4th and 7th prompts (whose tokens are close to words like cocoa, chocolate)



prompt id	nearest tokens
1	copper, steel, trading, gas, fx, aluminum, earn, coffee
2	gross, ship, index, money, gold, tin, iron, retail
3	product, cpi, industrial, acquisitions, jobs, supplying, orange, sugar
4	cocoa, production, grain, livestock, wholesale, cotton, bop, crude
5	oil, national, rubber, nat, interest, price, reserves, regional
6	nat, wholesale, sugar, golden, reserves, drinks, production, product
7	chocolate, sugar, cheat, orange, trade, fx, cash, acquiring
8	aluminum, livestock, cpc, tin, shops, wheat, petrol, supply

Integration with Other Meta-Learning Algorithms

- similar performance gain when used with any meta-learning algorithm
- example: BMG

5-way 5-shot classification meta-testing accuracy

	20News	*Amazon*	*HuffPost*	*Reuters*	*HWU64*	*Liu54*
MetaPrompting+BMG	85.71	83.47	73.92	96.27	93.31	93.04
MetaPrompter+BMG	87.91	86.45	74.99	98.01	95.41	94.52

Variance Reduction

two sources of gradient variance

1. data sampling for each task
2. sampling of tasks

variance reduction: reduce variance in the stochastic gradients

single-level optimization

- classic methods: SVRG, SAG, SDCA
- faster convergence than SGD both theoretically and empirically

bilevel optimization

- SUSTAIN, MRBO/VRBO, RSVRB, VR-BiAdam
- faster than bilevel algorithms without variance reduction

Removing Batch Gradient

batch gradient has to be computed occasionally → expensive

STORM [Cutkosky & Orabona (2019)]

- does NOT need batch gradient
- compute stochastic gradients at two points $\boldsymbol{w}_t, \boldsymbol{w}_{t-1}$ with the same mini-batch ξ_t

1: **Input:** $\boldsymbol{w}_0$, step-size $\{\eta_t\}$, decay parameter $\{\gamma_t\}$.
2: $\boldsymbol{c}_0 = \nabla \ell(\boldsymbol{w}_0; \xi_0)$
3: $\boldsymbol{w}_1 = \boldsymbol{w}_0 - \eta_0 \boldsymbol{c}_0$
4: **for** $t = 1$ **to** $T - 1$ **do**
5: sample ξ_t
6: $\boldsymbol{c}_t = \nabla \ell(\boldsymbol{w}_t; \xi_t) + (1 - \gamma_t)(\boldsymbol{c}_{t-1} - \nabla \ell(\boldsymbol{w}_{t-1}; \xi_t))$
7: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \boldsymbol{c}_t$
8: **end for**

- cf. SGD: $\boldsymbol{c}_t = \nabla \ell(\boldsymbol{w}_t; \xi_t)$
- cf. momentum: $\boldsymbol{c}_t = \gamma_t \nabla \ell(\boldsymbol{w}_t; \xi_t) + (1 - \gamma_t)\boldsymbol{c}_{t-1}$

Variance Reduction for Meta-Learning

straightforward approach

- use existing variance reduction methods
- replace the original gradients by their variance-reduced counterparts

meta-learning as bilevel optimization

- requires storing all task-specific parameters $\rightarrow$ prohibitive storage for large number of tasks or huge task models
- requires a large number of inner steps for convergence
  - meta-learning often uses only a small number of steps to avoid overfitting the limited data

Variance Reduction in Single-Level Meta-Learning

for $t = 0$ **to** $T-1$ **do**
  sample tasks $\mathcal{I}_t \subset \mathcal{I}$
  for $i \in \mathcal{I}_t$ **do**
    $\boldsymbol{u}_0^i = \boldsymbol{w}_t$
    for $k = 0$ **to** $K-1$ **do**
      obtain task i samples $\xi_{k,t}^i$
      $\boldsymbol{u}_{k+1}^i = \boldsymbol{u}_k^i - \alpha \nabla \ell(\boldsymbol{u}_k^i, \xi_{k,t}^i)$
    end for
    $\boldsymbol{c}_t^i = \frac{1}{K\alpha}(\boldsymbol{w}_t - \boldsymbol{u}_K^i)$
  end for
  $\boldsymbol{c}_t = \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} \boldsymbol{c}_t^i$
  $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \boldsymbol{c}_t$
end for

Reptile

- uses average gradient for adaptation

VFML [Wang et al. (2021)]

- integrates STORM with Reptile

problems

1. lacks theoretical properties
2. inferior empirical performance

Proposed Family of Methods

[Efficient variance reduction for meta-learning (ICML 2022)]

- can be integrated with various meta-learning algorithms
  - Reptile $\rightarrow$ VR-Reptile (Variance-Reduced Reptile)
  - MAML $\rightarrow$ VR-MAML
  - FOMAML $\rightarrow$ VR-FOMAML
  - BMG $\rightarrow$ VR-BMG

Reptile and VR-Reptile

for $t = 0$ **to** $T-1$ **do**
  sample tasks $\mathcal{I}_t \subset \mathcal{I}$
  for $i \in \mathcal{I}_t$ **do**
    $\boldsymbol{u}_0^i = \boldsymbol{w}_t$
    for $k = 0$ **to** $K-1$ **do**
      obtain task i samples $\xi_{k,t}^i$
      $\boldsymbol{u}_{k+1}^i = \boldsymbol{u}_k^i - \alpha \nabla \ell(\boldsymbol{u}_k^i, \xi_{k,t}^i)$
    end for
    $\boldsymbol{c}_t^i = \frac{1}{K\alpha}(\boldsymbol{w}_t - \boldsymbol{u}_K^i)$
  end for
  $\boldsymbol{c}_t = \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} \boldsymbol{c}_t^i$
  $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \boldsymbol{c}_t$
end for

... initialization ...
for $t = 1$ **to** $T-1$ **do**
  sample tasks $\mathcal{I}_t \subset \mathcal{I}$
  for $i \in \mathcal{I}_t$ **do**
    $\boldsymbol{u}_0^i = \boldsymbol{w}_t$
    $\boldsymbol{v}_0^i = \boldsymbol{w}_{t-1}$
    for $k = 0$ **to** $K-1$ **do**
      obtain task i samples $\xi_{k,t}^i$
      $\boldsymbol{u}_{k+1}^i = \boldsymbol{u}_k^i - \alpha \nabla \ell(\boldsymbol{u}_k^i; \xi_{k,t}^i)$
      $\boldsymbol{v}_{k+1}^i = \boldsymbol{v}_k^i - \alpha \nabla \ell(\boldsymbol{v}_k^i; \xi_{k,t}^i)$
    end for
    $\tilde{\boldsymbol{d}}_{t-1}^i = \frac{1}{K\alpha}(\boldsymbol{w}_{t-1} - \boldsymbol{v}_K^i)$
    $\tilde{\boldsymbol{c}}_t^i = \frac{1}{K\alpha}(\boldsymbol{w}_t - \boldsymbol{u}_K^i)$
  end for
  $\tilde{\boldsymbol{d}}_{t-1} = \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} \tilde{\boldsymbol{d}}_{t-1}^i$
  $\tilde{\boldsymbol{c}}_t = \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} \tilde{\boldsymbol{c}}_t^i + (1-\gamma_t)(\tilde{\boldsymbol{c}}_{t-1} - \tilde{\boldsymbol{d}}_{t-1})$
  $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \tilde{\boldsymbol{c}}_t$
end for

Remarks

... initialization ...
for $t = 1$ **to** $T - 1$ **do**
  sample tasks $\mathcal{I}_t \subset \mathcal{I}$
  for $i \in \mathcal{I}_t$ **do**
    $\boldsymbol{u}_0^i = \boldsymbol{w}_t$
    $\boldsymbol{v}_0^i = \boldsymbol{w}_{t-1}$
    for $k = 0$ **to** $K - 1$ **do**
      obtain task i samples $\xi_{k,t}^i$
      $\boldsymbol{u}_{k+1}^i = \boldsymbol{u}_k^i - \alpha \nabla \ell(\boldsymbol{u}_k^i; \xi_{k,t}^i)$
      $\boldsymbol{v}_{k+1}^i = \boldsymbol{v}_k^i - \alpha \nabla \ell(\boldsymbol{v}_k^i; \xi_{k,t}^i)$
    end for
    $\tilde{\boldsymbol{d}}_{t-1}^i = \frac{1}{K\alpha}(\boldsymbol{w}_{t-1} - \boldsymbol{v}_K^i)$
    $\tilde{\boldsymbol{c}}_t^i = \frac{1}{K\alpha}(\boldsymbol{w}_t - \boldsymbol{u}_K^i)$
  end for
  $\tilde{\boldsymbol{d}}_{t-1} = \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} \tilde{\boldsymbol{d}}_{t-1}^i$
  $\tilde{\boldsymbol{c}}_t = \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} \tilde{\boldsymbol{c}}_t^i + (1 - \gamma_t)(\tilde{\boldsymbol{c}}_{t-1} - \tilde{\boldsymbol{d}}_{t-1})$
  $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \tilde{\boldsymbol{c}}_t$
end for

- all γ_t's $= 1 \rightarrow$ VR-Reptile reduces to Reptile
- $K = 1 \rightarrow$ VR-Reptile reduces to STORM
- space-efficient: does NOT need to store task-specific $\boldsymbol{u}_K^i / \boldsymbol{v}_K^i$ and $\tilde{\boldsymbol{c}}_t^i / \tilde{\boldsymbol{d}}_{t-1}^i$
  - cf. direct application of bilevel variance reduction methods $\rightarrow$ requires storing all task-specific parameters

Convergence

Reptile

$$\frac{1}{T}\mathbb{E}\left[\sum_{t=0}^{T-1}\|\nabla\tilde{\mathcal{L}}(\boldsymbol{w}_t)\|^2\right] \leq \frac{\sqrt{2}G_1/\eta_0}{\sqrt{T}} + \frac{\sqrt{2}G_1/\eta_0}{T}$$

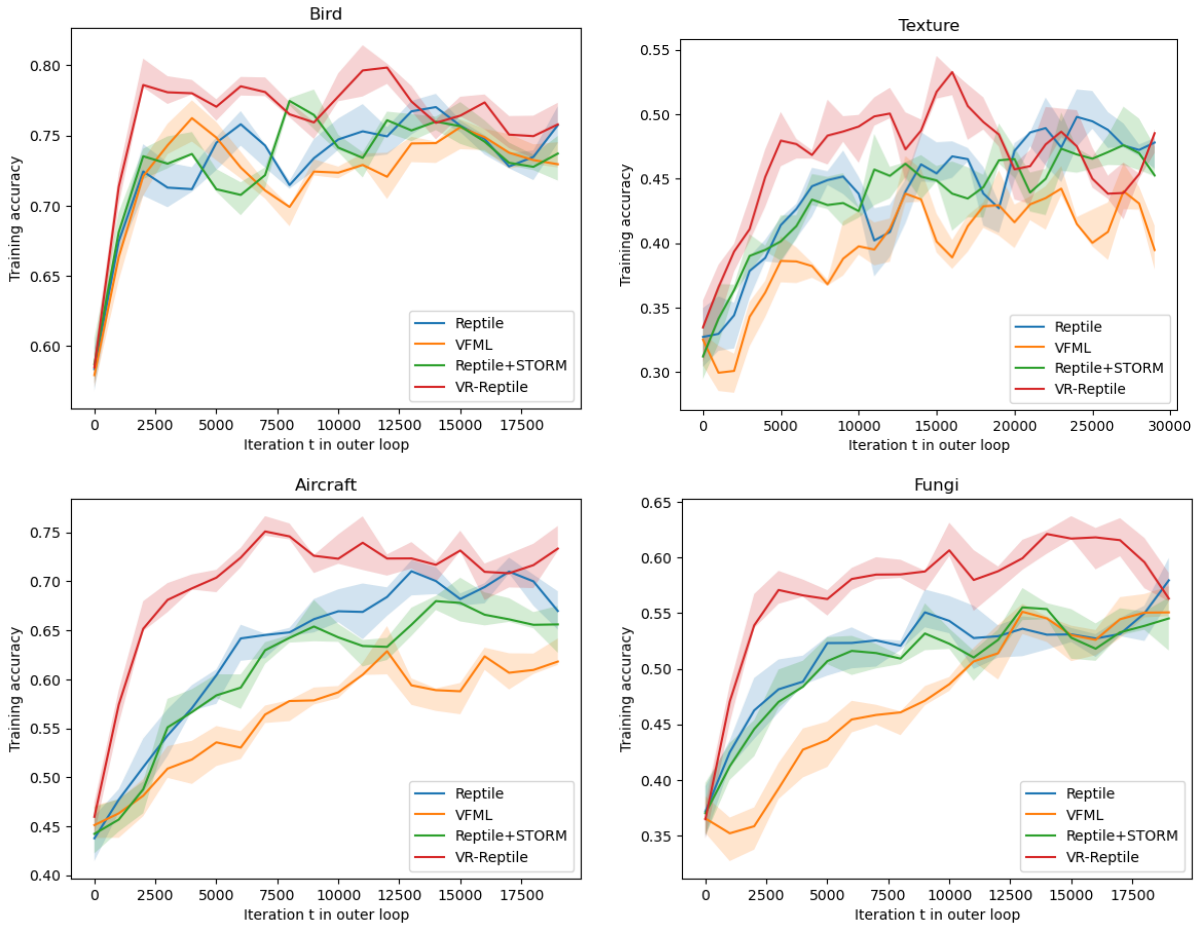
VR-Reptile

$$\frac{1}{T}\mathbb{E}\left[\sum_{t=0}^{T-1}\|\nabla\tilde{\mathcal{L}}(\boldsymbol{w}_t)\|^2\right] \leq \frac{4\tilde{M}G_2}{T^{2/3}} + \frac{65}{7}\cdot\frac{\tilde{M}G_2}{T}$$

- faster convergence rate

Experiments: Convergence

- 5-shot 5-way on Meta-Dataset (Bird, Texture, Aircraft, Fungi)
- accuracy with number of outer-loop iterations

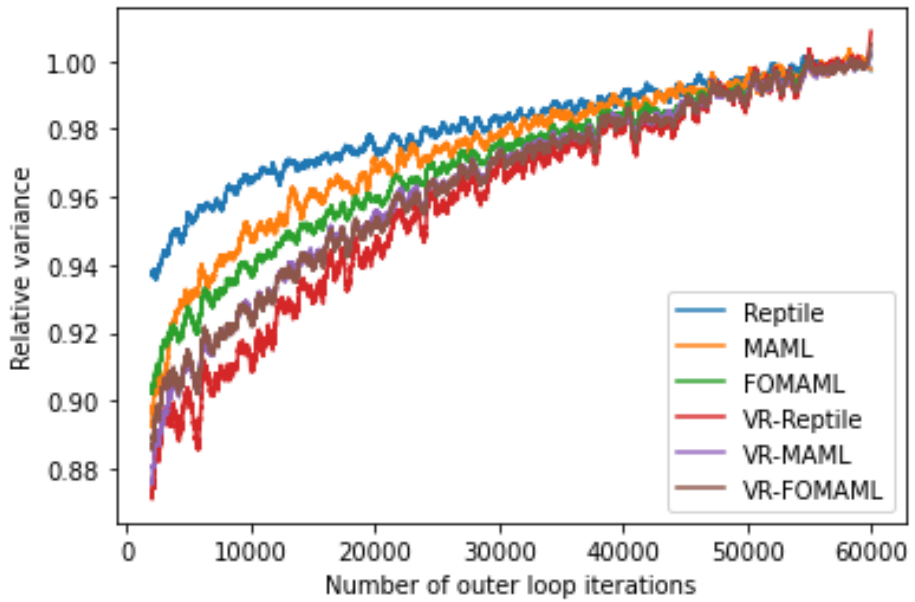


- VR-Reptile (red) has faster convergence

Reduce Stochastic Variance

variance of weight update $\tilde{\boldsymbol{c}}_t$ relative to squared norm

$$\mathbb{E}\|\tilde{\boldsymbol{c}}_t - \mathbb{E}[\tilde{\boldsymbol{c}}_t]\|^2 / \|\mathbb{E}[\tilde{\boldsymbol{c}}_t]\|^2$$



- variance ↓
- Reptile has larger variance → variance reduction also has larger improvements on Reptile

Few-Shot Classification on mini-ImageNet

architecture

- Conv4 backbone + Prototype classifier

meta-learning baselines

- standard methods: MAML/FOMAML/Reptile/BMG/DRS/ProtoNet
- straightforward STORM variants: replace stochastic gradients by variance-reduced counterparts by STORM
- VFML (Reptile+STORM)
- ANIL and variants with bilevel optimization variance reduction methods (SUSTAIN/MRBO/VRBO)

Accuracy on mini-ImageNet

	var reduction	single/bilevel	1-shot 5-way	5-shot 5-way
MAML	×	single	48.7±1.8	63.1±0.9
FOMAML	×	single	48.1±1.8	63.2±0.9
Reptile	×	single	50.0±0.3	66.0±0.6
BMG	×	single	50.7±0.5	65.6±0.6
DRS	×	single	24.5±0.8	30.4±0.6
ProtoNet	×	single	49.4±0.8	68.2±0.7
MAML+STORM	√	single	47.9±1.4	61.6±1.2
FOMAML+STORM	√	single	48.0±1.6	63.4±1.1
Reptile+STORM	√	single	49.9±0.3	66.2±0.3
BMG+STORM	√	single	46.7±0.6	60.9±0.8
DRS+STORM	√	single	24.7±1.1	30.3±0.7
VR-MAML	√	single	49.2±1.4	63.6±0.8
VR-FOMAML	√	single	48.3±1.2	63.4±0.6
VR-Reptile	√	single	50.4±0.4	67.6±0.8
VR-BMG	√	single	51.4±0.3	68.4±0.6
VFML	√	single	49.6±0.5	66.2±0.8
ANIL	×	bilevel	46.9±0.4	61.4±0.2
ANIL+SUSTAIN	√	bilevel	47.0±0.4	61.8±0.3
ANIL+MRBO	√	bilevel	47.2±0.5	62.0±0.2
ANIL+VRBO	√	bilevel	47.2±0.4	61.9±0.2

- integrating with any of the variance reduction methods leads to better performance

5-shot 5-way Classification Accuracy on Meta-Dataset

	var reduction	single/bilevel	Bird	Texture	Aircraft	Fungi
MAML	×	single	74.56	45.68	69.06	53.68
FOMAML	×	single	73.64	42.82	66.38	52.18
Reptile	×	single	74.60	43.26	66.46	52.88
BMG	×	single	74.52	43.74	66.64	53.02
DRS	×	single	53.34	33.28	41.06	37.64
ProtoNet	×	single	74.22	49.86	71.38	53.94
MAML+STORM	√	single	74.86	45.26	68.48	53.72
FOMAML+STORM	√	single	73.72	42.78	66.22	52.28
Reptile+STORM	√	single	75.24	44.60	67.48	52.54
BMG+STORM	√	single	73.16	42.32	66.46	51.74
DRS+STORM	√	single	53.48	33.42	41.12	37.72
VR-MAML	√	single	75.06	46.18	68.36	53.86
VR-FOMAML	√	single	74.28	43.28	66.98	52.16
VR-Reptile	√	single	76.48	46.94	71.62	54.24
VR-BMG	√	single	76.56	47.28	71.48	54.38
VFML	√	single	74.38	44.48	65.64	52.76
ANIL	×	bilevel	73.68	41.96	68.74	52.84
ANIL+SUSTAIN	√	bilevel	73.74	42.12	68.82	52.78
ANIL+MRBO	√	bilevel	73.78	42.18	68.78	52.86
ANIL+VRBO	√	bilevel	73.88	42.22	68.74	52.82

Conflicting Gradients

(meta-learning) $\min_w \sum_{\tau=1}^{m} \mathcal{L}_\tau(w_\tau)$ s.t. $w_\tau = w_\tau^*(w)$

conflicting gradients

significantly influenced by a small subset of tasks

Meta-Learning as Multi-Objective Optimization (MOO)

$$\text{(MOO)} \qquad \min_{w} (\mathcal{L}_1(w_1^*(w)), \ldots, \mathcal{L}_m(w_m^*(w)))$$

- each task is an objective

gradient-based MOO solvers

- e.g., MGDA, PCGard, CAGard
- in each iteration, find a descent direction common to all objectives

Example (MGDA (multiple-gradient descent algorithm))

direction: $\sum_{\tau=1}^{m} \gamma_\tau^* \nabla_x f_\tau(x)$

$$\{\gamma_\tau^*\} = \arg\min_{\{\gamma_\tau\}} \left\| \sum_{\tau=1}^{m} \gamma_\tau \nabla_x f_\tau(x) \right\|^2 \quad \text{s.t.} \quad \sum_{\tau=1}^{m} \gamma_\tau = 1, \gamma_\tau \geq 0$$

Problem

$$\{\gamma_\tau^*\} = \arg\min_{\{\gamma_\tau\}} \left\| \sum_{\tau=1}^{m} \gamma_\tau \nabla_x f_\tau(x) \right\|^2 \quad \text{s.t.} \quad \sum_{\tau=1}^{m} \gamma_\tau = 1, \gamma_\tau \geq 0$$

requires gradients from all objectives

meta-learning: computing all task gradients can be very expensive

Example (5-way few-shot classification on MiniImagenet)

total number of meta-training tasks: $\binom{64}{5} \approx 7 \times 10^6$

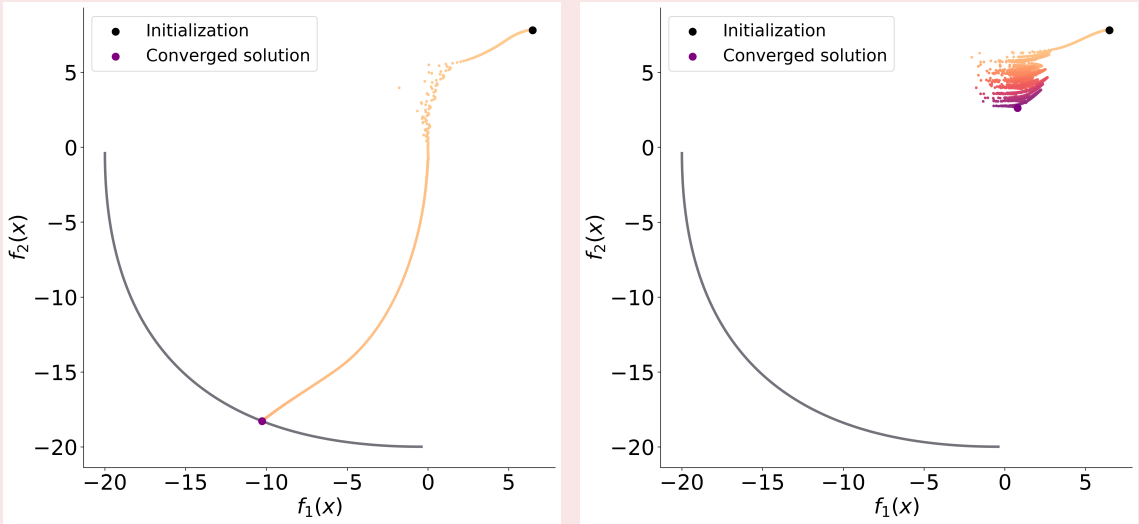
Straightforward Solution

use a mini-batch B of objectives in each iteration

$$\min_{\{\gamma_\tau\}} \left\| \sum_{\tau \in B} \gamma_\tau \nabla_\tau (\mathcal{L}_\tau(w_\tau^*(w)) \right\|^2 \text{ s.t. } \sum_{\tau \in B} \gamma_\tau = 1, \gamma_\tau \geq 0$$

problem: does not converge to Pareto front

- two tasks/objectives (mini-batch size of 1)
- (batch) MGDA vs mini-batch MGDA



Improvement Function in MOO Problem

[Enhancing meta-learning via multi-objective soft improvement functions (ICLR 2023)]

$$\min_x [f_1(x), \ldots, f_m(x)]$$

improvement function

$$H(x, x') = \max_{\tau=1\ldots,m} \{f_\tau(x) - f_\tau(x')\}$$

$x^* = \arg\min_x H(x, x^*)$ for a Pareto stationary point x^*

- to find x^*: descent on H

$$\begin{aligned} x^{s+1} &= x^s + \beta d^* \\ d^* &= \arg\min_d H(x^s + d, x^s) + \frac{\lambda'}{2}\|d\|^2 \end{aligned}$$

  - $s \to \infty$, x^s is Pareto stationary

Meta-Learning Context

$$\min_w (\mathcal{L}_1(w_1^*(w)), \ldots, \mathcal{L}_m(w_m^*(w)))$$

improvement function

$$\begin{aligned} H(w, w') &= \max_{\tau=1\ldots,m} \left\{ \mathcal{L}_\tau(w_\tau^*(w)) - \mathcal{L}_\tau(w_\tau^*(w')) \right\} \\ &= \max_\pi \mathbb{E}_{\tau\sim\pi} \left[\mathcal{L}_\tau(w_\tau^*(w)) - \mathcal{L}_\tau(w_\tau^*(w')) \right] \end{aligned}$$

- π: probability density function on τ

update

$$\begin{aligned} w^{s+1} &= w^s + \beta d^* \\ d^* &= \arg\min_d \left(\max_\pi \mathbb{E}_{\tau\sim\pi} \left[\mathcal{L}_\tau(w_\tau^*(w^s + d)) - \mathcal{L}_\tau(w_\tau^*(w^s)) \right] \right) + \frac{\lambda'}{2} \|d\|^2 \end{aligned}$$

Soft Improvement Function

take first-order approximation

$$\min_d \left(\max_\pi \mathbb{E}_{\tau\sim\pi} \left[\mathcal{L}_\tau(w_\tau^*(w^s + d)) - \mathcal{L}_\tau(w_\tau^*(w^s)) \right] + \frac{\lambda'}{2}\|d\|^2 \right)$$

$$= \max_\pi \underbrace{\left(\min_d \mathbb{E}_{\tau\sim\pi} \left[\mathcal{L}_\tau(w_\tau^*(w^s + d)) - \mathcal{L}_\tau(w_\tau^*(w^s)) \right] + \frac{\lambda'}{2}\|d\|^2 \right)}_{\text{closed-form solution: } \frac{-1}{2\lambda'}\|\mathbb{E}_{\tau\sim\pi}\nabla_w \mathcal{L}_\tau(w_\tau^*(w))|_{w=w^s}\|^2}$$

expectation $\mathbb{E}_{\tau\sim\pi}$

- sample tasks from uniform distribution U, then weight sampled task τ with $r(\tau) \equiv \pi(\tau)/U(\tau)$
- parameterize r as a neural network r_θ with parameter θ

$$\max_\theta K(\theta) \equiv \frac{-1}{2\lambda'} \left[\mathbb{E}_{\tau\sim U} r_\theta(\tau) \nabla_w \mathcal{L}_\tau(w_\tau^*(w))|_{w=w^s} \right]^2 - \frac{\lambda''}{2} (\mathbb{E}_{\tau\sim U} r_\theta(\tau) - 1)^2$$

- the last term: enforcing the constraint $\mathbb{E}_{\tau\sim U} r_\theta(\tau) = 1$

Problem

$$\max_{\theta} K(\theta) \equiv \frac{-1}{2\lambda'} \left[\mathbb{E}_{\tau\sim U} r_\theta(\tau) \nabla_w \mathcal{L}_\tau(w_\tau^*(w))|_{w=w^s}\right]^2 - \frac{\lambda''}{2}(\mathbb{E}_{\tau\sim U} r_\theta(\tau) - 1)^2$$

$\mathbb{E}_{\tau\sim U}$: still needs to access all tasks

Mini-Batch Again!

- B: mini-batch of k tasks

$$\underbrace{\frac{-1}{2\hat{\lambda}'}\left\|\frac{1}{|B|}\sum_{\tau\in B} r_\theta(\tau)\nabla_w \mathcal{L}_\tau(w_\tau^*(w))|_{w=w^s}\right\|^2 - \frac{\hat{\lambda}''}{2}\left(\frac{1}{|B|}\sum_{\tau\in B} r_\theta(\tau) - 1\right)^2}_{\tilde{K}_B(\theta)}$$

$K(\theta)$ can be approximated by using mini-batches

when $k \ll m$, $\left(K(\theta) - \frac{1}{|\mathcal{B}|}\sum_{B\in\mathcal{B}} \tilde{K}_B(\theta)\right)^2 \le \frac{G_1}{k\lambda'} + \frac{G_2}{k}\lambda''$

- in experiments: $m \ge 10^6$, $k \approx 10^2 \rightarrow$ diff ≤ 0.2

Soft Improvement Multi-Objective Meta Learning (SIMOL)

update the distribution

$$\theta^{s+1} = \theta^s + \beta' \nabla_\theta \tilde{K}_B(\theta^s)$$

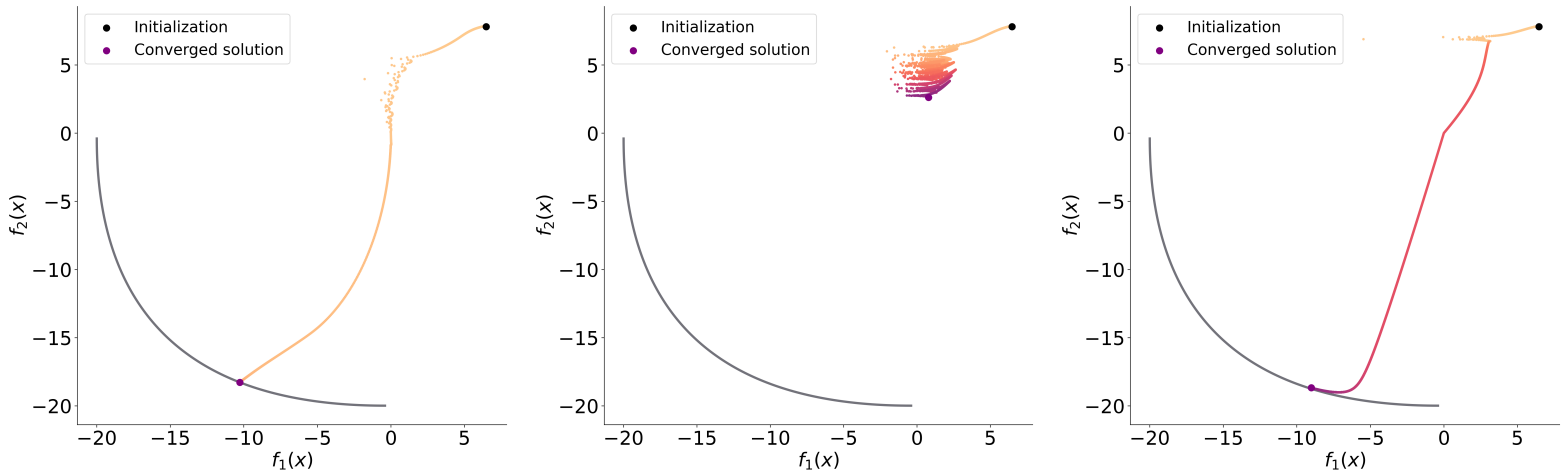
update the meta-model

$$\begin{aligned} w^{s+1} &= w^s + \beta \tilde{d}^* \\ \tilde{d}^* &= -\frac{1}{\lambda' |B|} \sum_{\tau \in B} \nabla_w \mathcal{L}_\tau (w_\tau^*(w))|_{w=w^s} \end{aligned}$$

theoretically converge to an ϵ-Pareto stationary point

Experiment

- two objectives (mini-batch size of 1)
- MGDA vs mini-match MGDA vs SIMOL



Few-shot Regression

- synthetic data: $y = a_\tau \sin(x + b_\tau)$
- $160,000$ meta-training tasks, $1,000$ meta-testing tasks
- meta-learner / re-weighting network: MLP with 2 FC layers

MAML	overall		worst-10%	
	5-shot	2-shot	5-shot	2-shot
min average loss	0.43 ± 0.11	1.70 ± 0.11	2.13 ± 0.21	7.75 ± 0.48
mini-batch MGDA	0.60 ± 0.02	1.73 ± 0.10	2.62 ± 0.10	7.04 ± 0.51
mini-batch CAGrad	1.90 ± 0.24	1.82 ± 0.44	8.18 ± 0.63	8.23 ± 2.33
SIMOL	0.34 ± 0.04	1.24 ± 0.08	1.69 ± 0.18	5.66 ± 0.33

- SIMOL consistently outperforms SGD, MGDA, and CAGrad in terms of both the overall and worst-10% MSEs
- mini-batch MGDA and CAGrad are not good

Few-Shot Image Classification (miniImageNet)

accuracy

		overall		worst-10%	
		1-shot	5-shot	1-shot	5-shot
(MAML)	min average loss	49.24 ± 0.78	62.13 ± 0.72	13.33 ± 1.07	41.71 ± 1.02
	mini-batch MGDA	46.08 ± 0.78	60.15 ± 0.41	10.60 ± 1.33	39.67 ± 0.55
	mini-batch CAGrad	44.67 ± 0.75	60.05 ± 0.67	11.33 ± 1.12	40.01 ± 0.88
	SIMOL	50.62 ± 1.39	65.83 ± 0.86	14.99 ± 1.72	44.81 ± 0.58
(MAML variants)	Reptile	47.07 ± 0.26	62.74 ± 0.37	-	-
	FOMAML	45.53 ± 1.58	61.02 ± 1.12	-	-
	IMAML	49.30 ± 1.88	59.77 ± 0.41	-	-
	Meta-MP	48.51 ± 0.92	64.15 ± 0.92	-	-
	TS-MAML	48.44 ± 0.91	65.52 ± 0.68	-	-
	MTL	49.87 ± 0.41	65.81 ± 0.33	13.64 ± 1.45	43.42 ± 0.47

- SIMOL consistently outperforms all the baselines in terms of both overall and worst-10% accuracies

Few-Shot Image Classification (tieredImageNet)

accuracy

		overall		worst-10%	
		1-shot	5-shot	1-shot	5-shot
(MAML)	min average loss	50.58 ± 1.44	69.33 ± 0.74	11.60 ± 1.96	46.95 ± 1.14
	mini-batch MGDA	22.92 ± 1.04	53.41 ± 0.74	7.12 ± 2.11	32.79 ± 0.88
	mini-batch CAGrad	49.04 ± 0.93	65.43 ± 0.73	11.40 ± 1.97	42.63 ± 0.95
	SIMOL	51.42 ± 1.50	70.13 ± 0.74	12.00 ± 1.95	47.51 ± 1.46
(MAML variants)	Reptile	49.12 ± 0.43	65.99 ± 0.42	-	-
	FOMAML	45.53 ± 1.58	61.02 ± 1.12	-	-
	IMAML	38.54 ± 1.37	60.24 ± 0.76	-	-
	Meta-MP	50.14 ± 1.37	68.30 ± 0.91	-	-
	TS-MAML	48.82 ± 0.88	67.82 ± 0.72	-	-
	MTL	51.02 ± 0.46	66.47 ± 0.39	13.60 ± 1.59	49.45 ± 0.58

Using Batch MGDA and CAGrad

per-epoch running time (miniImageNet)

standard MAML	SIMOL	batch MGDA	batch CAGrad
2.0 sec	2.3 sec	6.9 days	5.6 days

- SIMOL has comparable per-epoch running time as MAML
- batch MGDA and CAgrad are much more computationally expensive (around $432,000$ times slower)

Conclusion

how to learn with a lot of tasks in meta-learning?

use multiple meta-models into meta-learning

- subspace learning of task model parameters
- meta-learning a prompt pool

variance reduction to accelerate convergence of meta-learning

- does not need to store all task-specific parameters

soft improvement function in MOO to handle conflicting gradients

- scalable gradient-based solver with theoretical guarantees to Pareto-optimality

empirically, outperform existing meta-learning algorithms

References

- W. Jiang, J. Kwok, Y. Zhang. Subspace learning for effective meta-learning. ICML, 2022
- W. Jiang, Y. Zhang, J. Kwok. Effective structured-prompting by meta-learning and representative verbalizer. ICML, 2023.
- H. Yang, J. Kwok. Efficient variance reduction for meta-learning. ICML, 2022
- R. Yu, W. Chen, X. Wang, J. Kwok. Enhancing meta-learning via multi-objective soft improvement functions, ICLR, 2023