

Meta-Learning

Muhammad Ahmad Kaleem

University of Toronto, Vector Institute

November 23, 2022

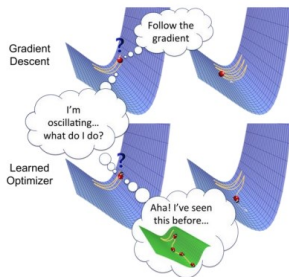
Outline

- 1 Introduction/Motivation
- 2 Meta-Learning Approaches
- 3 Meta-Learning and Trustworthy ML

Background

Introduction

- "Learning to learn"
- Distribution of tasks instead of over samples
- Tasks can be any type of learning problem e.g. supervised learning, reinforcement learning
- Use many prior tasks to learn how to learn \Rightarrow learn new tasks more efficiently with few examples



(Li, 2017)

Motivation

- Humans can learn fast with few examples
- Training with large datasets is expensive, training examples are hard to find
- Tackling conventional deep learning challenges



(Lake et al., 2015)

Applications

- Few-shot Learning
- Neural Architecture Search (NAS)
- Hyperparameter optimization
- Transfer Learning
- Reinforcement Learning, Robotics

Overview

Framework

- Classical learning: Learn from training examples, evaluate on test examples
- Meta-learning: Learn from a set of training tasks (meta-training), evaluate on a set of test tasks (meta-testing)
- Each task associated with dataset $\mathcal{D}_i = \{S_i, Q_i\}$
 - S is support set (for learning), Q is query set (for evaluation)

Training task 1

Support set

{Cat, Lamb, Pig}

K=2



N=3

Query set



Training task 2 . . .

Support set

{Dog, Shark, Lion}



Query set



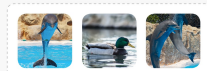
Test task 1 . . .

Support set

{Duck, Dolphin, Hen}



Query set



Few shot classification as meta-learning (Zi et al., 2019)

Framework (cont'd)

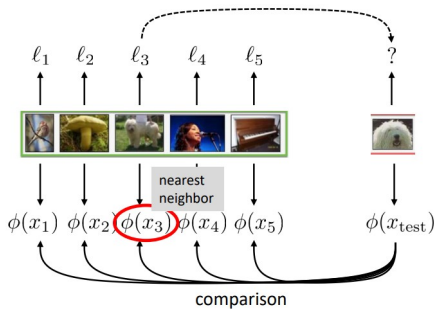
- At each training step, update model parameters based on randomly sampled training task
- Loss function is based on performance on query set of training task
- Different task at each step \Rightarrow learn to learn tasks in general

Formulation

- Standard training: $\theta^* = \arg \min_{\theta} \mathcal{L}(\theta, \mathcal{D})$
- Meta-learner θ , learner ϕ_i for specific task i
- $\phi_i = f_{\theta}(S_i)$, $f_{\theta}(S_i, x) \rightarrow y$,
- Meta-learning: $\theta^* = \arg \min_{\theta} \sum_{i=1}^n \mathcal{L}_i(\phi_i, Q_i)$,
- Adaptation / test time: $\phi_{ts}^* = f_{\theta^*}(S_{ts})$
- Three main approaches: **metric-based**, **model-based**, **optimization-based**

Metric-based

- Learn an embedding ϕ into feature space



Overview for a single task (Levine, 2021)

- $\theta^* = \arg \min_{\theta} \sum_{i=1}^n \mathcal{L}_i(f_{\theta}(S_i), Q_i)$, $\phi_i = f_{\theta}(S_i)$ is learned nearest neighbor classifier

Metric-based (cont'd)

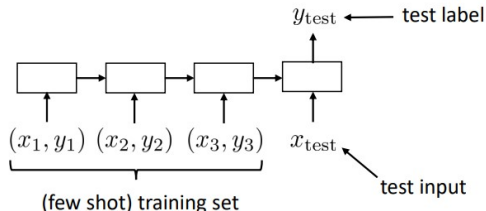
- Measure similarity between data samples:

$$p_{\text{nearest}}(x_k^{tr} | x_j^{ts}) \propto \exp(\phi(x_k^{tr})^T \phi(x_j^{ts}))$$

- $p_{\theta}(y|x, S_i)$ modeled as $\sum_{(x_j, y_j) \in S_i, y_j = k} p_{\text{nearest}}(x_j | x)$
- Meta-learn feature space to get meaningful comparisons
- Examples: Prototypical Networks (Snell et al., 2017), Matching Networks (Vinyals et al., 2016)

Model-based

- Specifically designed architectures for fast learning
- Use of external memory e.g. with sequence models
- Use neural network for $p_{\theta}(y|x, S_i) = f_{\theta}(x, S_i)$ e.g. RNN



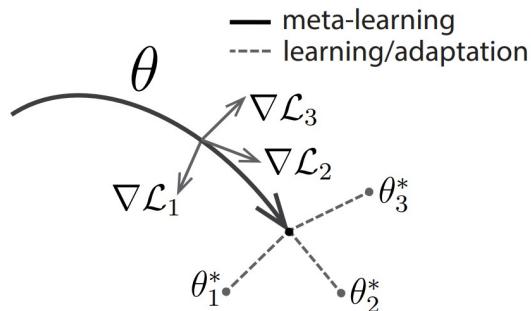
Example of network (Levine, 2021)

- Meta-learner uses gradient descent, task learner then uses network
- Examples: Memory-augmented neural networks (MANN) (Santoro et al., 2016), Meta Networks (Munkhdalai and Yu, 2017)

Optimization-based

- Frame meta-learning as an optimization problem
- f with parameters θ : optimize for θ which can quickly adapt to new tasks
- For a task i let $\theta'_i \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}_i(f_{\theta}, S_i)$ (finetuning)
- Update θ across tasks $\theta \leftarrow \theta - \beta \sum_i \nabla_{\theta} \mathcal{L}_i(f_{\theta'_i}, Q_i)$ (meta-update)
- Double gradient descent. Can be implemented with a computational graph
- Examples: Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017), Reptile (Nichol et al., 2018)

Optimization-based (cont'd)



MAML (Finn et al., 2017)

- Find parameters θ sensitive to changes in task
- Fast adaptation by changing θ in direction of gradient of loss for a task
- Model $p_{\theta}(y|x, S_i) = f_{\theta(S_i)}(x)$

Comparison

- Metric-based
 - Performs well, easier optimization
 - Limited to specific settings such as classification
- Model-based
 - Conceptually simple
 - Requires more learning, hard to scale to large tasks
- Optimization-based
 - Converges to local optimum, good generalization to OOD tasks
 - Harder to train

Trustworthy Meta-Learning

Adversarial Robustness

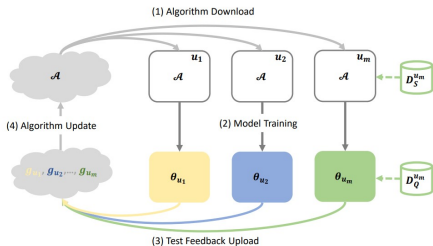
- Meta-learning vulnerable to adversarial examples
- Adversarially robust learning expensive
- Adversarial Querying: introduce adversarial examples during query step of meta-learning (Goldblum et al., 2020)
- Fast adversarial robustness adaptation (Wang et al., 2021)
 - Robustifying meta update stage sufficient
 - Robustness regularized meta-learning framework

Differential Privacy

- Sensitive information in a task's dataset
- Query within task (using ϕ_i) or meta-level (using θ) \Rightarrow Record-level and task-level privacy
- Privacy for ϕ_i and \mathcal{D}_i
- $\mathcal{D}_i \rightarrow \phi_i$ (task learner), $\phi_i \rightarrow \theta$ (meta-learner)
- DP-SGD to obtain $\phi_i \Rightarrow$ global DP guarantees for task
- DP Meta-Learning (Li et al., 2019)

Federated Meta-Learning

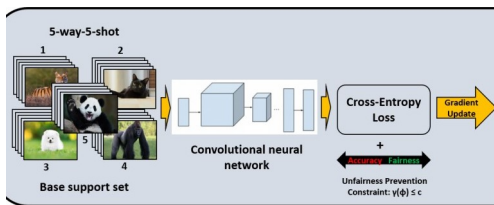
- Meta-learning to address challenges in federated learning with large number of devices (Chen et al., 2018)
 - MAML used on the level of individual clients, server then performs update using gradients of meta test losses



- Personalized federated learning with meta-learning (Fallah et al., 2020)
 - Find shared model which individual users can easily adapt to their local dataset

Fairness






- Biases in meta-training process may lead to unfair algorithms
- Additional optimization constraints for fairness (Zhao et al., 2020)
- Fair-MAML: Balance fairness and accuracy using fairness regularizers in MAML optimization (Slack et al., 2020)



(Zhao et al., 2020)

Bibliography

Bibliography I

-  Bosc, Tom (2016). “Learning to learn neural networks”. In: *arXiv preprint arXiv:1610.06072*.
-  Chen, Fei, Mi Luo, Zhenhua Dong, Zhenguo Li, and Xiuqiang He (2018). “Federated meta-learning with fast convergence and efficient communication”. In: *arXiv preprint arXiv:1802.07876*.
-  Fallah, Alireza, Aryan Mokhtari, and Asuman Ozdaglar (2020). “Personalized federated learning: A meta-learning approach”. In: *arXiv preprint arXiv:2002.07948*.
-  Finn, Chelsea, Pieter Abbeel, and Sergey Levine (2017). “Model-agnostic meta-learning for fast adaptation of deep networks”. In: *International conference on machine learning*. PMLR, pp. 1126–1135.
-  Goldblum, Micah, Liam Fowl, and Tom Goldstein (2020). “Adversarially robust few-shot learning: A meta-learning approach”. In: *Advances in Neural Information Processing Systems 33*, pp. 17886–17895.

Bibliography II



Lake, Brenden M., Ruslan Salakhutdinov, and Joshua B. Tenenbaum (2015). “Human-level concept learning through probabilistic program induction”. In: *Science* 350.6266, pp. 1332–1338. DOI: [10.1126/science.aab3050](https://doi.org/10.1126/science.aab3050). eprint: <https://www.science.org/doi/pdf/10.1126/science.aab3050>. URL:

<https://www.science.org/doi/abs/10.1126/science.aab3050>.



Levine, Sergey (2021). “Meta-Learning”. In: URL: <https://cs182sp21.github.io/static/slides/lec-21.pdf>.



Li, Jeffrey, Mikhail Khodak, Sebastian Caldas, and Ameet Talwalkar (2019). “Differentially private meta-learning”. In: *arXiv preprint arXiv:1909.05830*.



Li, Ke (2017). “Learning to Optimize with Reinforcement Learning”. In: URL: <https://bair.berkeley.edu/blog/2017/09/12/learning-to-optimize-with-rl/>.



Munkhdalai, Tsendsuren and Hong Yu (2017). “Meta networks”. In: *International Conference on Machine Learning*. PMLR, pp. 2554–2563.

Bibliography III



Nichol, Alex, Joshua Achiam, and John Schulman (2018). “On first-order meta-learning algorithms”. In: *arXiv preprint arXiv:1803.02999*.



Ravi, Sachin and Hugo Larochelle (2017). “Optimization as a Model for Few-Shot Learning”. In: *International Conference on Learning Representations*. URL: <https://openreview.net/forum?id=rJY0-Kc11>.



Santoro, Adam, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap (2016). “Meta-Learning with Memory-Augmented Neural Networks”. In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, pp. 1842–1850.



Slack, Dylan, Sorelle A Friedler, and Emile Givental (2020). “Fairness warnings and Fair-MAML: learning fairly with minimal data”. In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pp. 200–209.



Snell, Jake, Kevin Swersky, and Richard Zemel (2017). “Prototypical networks for few-shot learning”. In: *Advances in neural information processing systems* 30.

Bibliography IV



Vinyals, Oriol, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. (2016). “Matching networks for one shot learning”. In: *Advances in neural information processing systems* 29.



Wang, Ren, Kaidi Xu, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Chuang Gan, and Meng Wang (2021). “On fast adversarial robustness adaptation in model-agnostic meta-learning”. In: *arXiv preprint arXiv:2102.10454*.



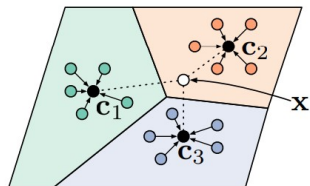
Zhao, Chen, Changbin Li, Jincheng Li, and Feng Chen (2020). “Fair meta-learning for few-shot classification”. In: *2020 IEEE International Conference on Knowledge Graph (ICKG)*. IEEE, pp. 275–282.



Zi, W., L.S. Ghoaraie, and S. Prince (2019). “Tutorial 2: few-shot learning and meta-learning I”. In: URL: <https://www.borealisai.com/research-blogs/tutorial-2-few-shot-learning-and-meta-learning-i/>.

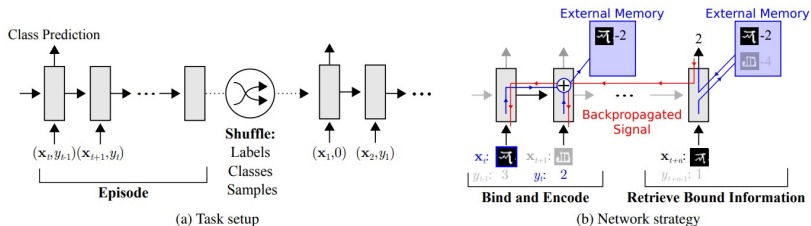
Extra Slides

Prototypical Networks



Few-shot classification with ProtoNets

- Compute prototype $c_k \in \mathbb{R}^m$ of each class through embedding function $f_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^m$
- Prototype corresponds to mean vector of embedded points from the same class k : $c_k = \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} f_\theta(x_i)$
- Given a query point x , produce distribution over classes based on embedding space distances and softmax



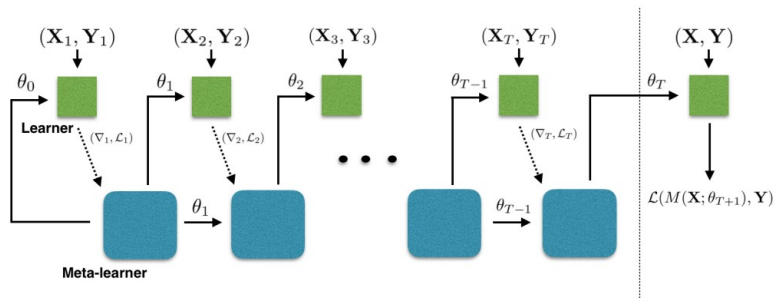
MANN (Bosc, 2016)

- Can be implemented using neural turing machine

Reptile

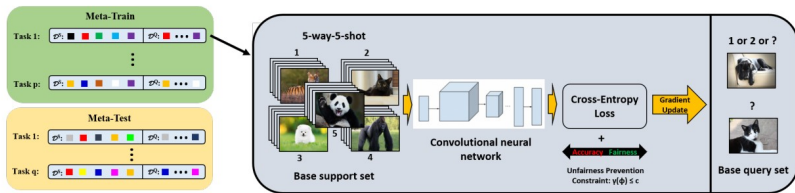
- Very simple method closely related to standard training
- Start with params Φ
- At each iteration, randomly sample a task i
- Perform k steps of SGD on task i starting with Φ and leading to params W
- Update $\Phi \leftarrow \Phi + \epsilon(W - \Phi)$

Optimization based (2)



(Ravi and Larochelle, 2017)

Fairness



(Zhao et al., 2020)

- Additional optimization constraint based on decision boundary covariance: the covariance between protected variables and signed distance from vectors to decision boundary

Implementation and Benchmarks

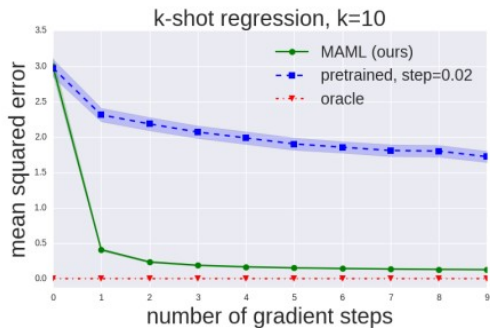
- Commonly used datasets for classification: Omniglot, mini-imagenet
- Omniglot has 1623 characters from 50 different alphabets
- Training and test tasks usually assumed to be from a similar distribution i.e. similar number of classes and support points per class

Table 1: Few-shot classification accuracies on Omniglot.

Model	Dist.	Fine Tune	5-way Acc.		20-way Acc.	
			1-shot	5-shot	1-shot	5-shot
MATCHING NETWORKS [29]	Cosine	N	98.1%	98.9%	93.8%	98.5%
MATCHING NETWORKS [29]	Cosine	Y	97.9%	98.7%	93.5%	98.7%
NEURAL STATISTICIAN [6]	-	N	98.1%	99.5%	93.2%	98.1%
PROTOTYPICAL NETWORKS (OURS)	Euclid.	N	98.8%	99.7%	96.0%	98.9%

Few shot classification performance for ProtoNets (Snell et al., 2017)

Example of model performance



(Finn et al., 2017)