Adaptive Gradient-Based Meta-Learning Methods

Mikhail Khodak, Maria-Florina Balcan, Ameet Talwalkar
Carnegie Mellon University
khodak@cmu.edu

Gradient-Based Meta-Learning (GBML)

GBML is a popular way to use multi-task data to learn a personalized model for each task [2, 3, 4]. The goal is to learn a *meta-initialization* ϕ and *learning rate* η for gradient descent.

Contribution: ARUBA, a new theoretical framework for analyzing GBML:
- new theoretical guarantees for online and statistical meta-learning.
- better methods for few-shot and federated learning.
- principled and general approach for designing meta-learning algorithms.

Average Regret-Upper-Bound Analysis

Suppose each task t is an online learning problem:
- learner must sequentially decide $\theta_{t,i} \in \mathbb{R}^d$ to pass to adversarial losses $\ell_{t,i} : \mathbb{R}^d \mapsto \mathbb{R}$
- performance of online gradient descent (OGD) measured by its **regret** compared to playing the best fixed action θ_t^*:

$$\mathbf{R}_t \;=\; \sum_{i=1}^m \ell_{t,i}(\theta_{t,i}) \;-\; \min_{\theta_t^*} \sum_{i=1}^m \ell_{t,i}(\theta_t^*)$$

For convex Lipschitz losses $\ell_{t,i}$ OGD has the following **regret-upper-bound** $\mathbf{U}_t$ [6]:

$$\mathbf{R}_t \;\le\; \mathbf{U}_t(\phi, \eta) \;=\; \frac{1}{2\eta}\|\theta_t^* - \phi\|_2^2 \;+\; \eta m$$

ARUBA exploits the following properties of $\mathbf{U}_t$:
- **data-dependence:** strong dependence on the task-data via θ_t^*
- **niceness:** joint convexity in algorithm parameters ϕ and η

The ARUBA Framework:

Run online learning on regret-upper-bounds $\mathbf{U}_t$ to set within-task algorithm parameters (e.g. initialization ϕ, learning rate η of OGD). Use online learning to show data-dependent bounds on the average regret per-task and excess transfer risk.

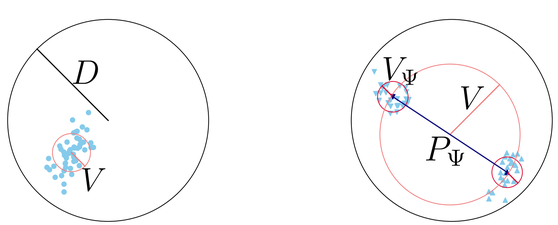


Illustration of task-similarity notions, in the static case (left) and dynamic case (right). The optimal task-parameters θ_t^* are in blue. ARUBA can exploit dynamic regret guarantees to adapt to a dynamic sequence of comparators Φ with small path-length P_Φ.

Generic GBML Training:

For task $t = 1, \ldots, T$:
- initialize gradient descent with initialization $\phi_t \in \mathbb{R}^d$, learning rate $\eta_t > 0$
- run gradient descent on task-samples to obtain task model weights $\hat{\theta}_t$
- update ϕ_{t+1}, η_{t+1} using $\hat{\theta}_t$

Output $\phi = \phi_{T+1}$ and $\eta = \eta_{T+1}$.

Average Regret Guarantees

ARUBA yields average regret bounds of form

$$\frac{1}{T}\sum_{t=1}^T \mathbf{R}_t \;\le\; \frac{1}{T}\sum_{t=1}^T \mathbf{U}_t(\phi_t, \eta_t) \;=\; o_T(1) + \min_{\phi,\eta} \frac{1}{T}\sum_{t=1}^T \mathbf{U}_t(\phi, \eta)$$

As $T \to \infty$ average regret per-task converges to that of always using the initialization and learning rate that minimizes the regret-upper-bound.

Theorem 1: For convex Lipschitz losses on the unit ball, if OGD is run using $\phi_{t+1} = \phi_t + \frac{1}{t\eta_t}(\theta_s^* - \phi_t)$, one can (efficiently) set η_t so that average regret is

$$\tilde{\mathcal{O}}\left(V\sqrt{m} \;+\; D/\sqrt{T}\right)$$

for $V^2 = \min_\phi \frac{1}{T}\sum_{t=1}^T \|\theta_t^* - \phi\|_2^2$.

Main Point: single-task regret is $\Omega(D\sqrt{m})$, where D is the diameter of Θ, so this shows that GBML does much better if $V \ll D$.

Statistical Guarantees

In the i.i.d. $\ell_{t,i} \sim \mathcal{P}_t \sim \mathcal{Q}$ setting we also get strong excess transfer risk guarantees:

Theorem 2: The algorithm from Theorem 1 but fixed $\eta_t = \frac{V+1/\sqrt{T}}{\sqrt{m}}$ learns a meta-initialization so that w.p. $1-\delta$ when OGD is run on samples from a new task distribution $\mathcal{P} \sim \mathcal{Q}$ the learned task-parameter $\bar{\theta}$ satisfies (in expectation):

$$\mathop{\mathbb{E}}_{\mathcal{Q}} \ell_{\mathcal{P}}(\bar{\theta}) \;\le\; \mathop{\mathbb{E}}_{\mathcal{Q}} \ell_{\mathcal{P}}(\theta^*) \;+\; \tilde{\mathcal{O}}\left(\frac{V}{\sqrt{m}} \;+\; \left(\frac{1}{\sqrt{mT}} + \frac{1}{T}\right)\log\frac{1}{\delta}\right)$$

Main Point: faster task-dependent rates for excess transfer risk bounds. Can also get good rates when the step-size is learned in addition to the initialization.

Other things we can prove using ARUBA:
- per-coordinate and full-matrix learning rates
- dynamic regret for changing task-environments
- high probability bounds in i.i.d. setting

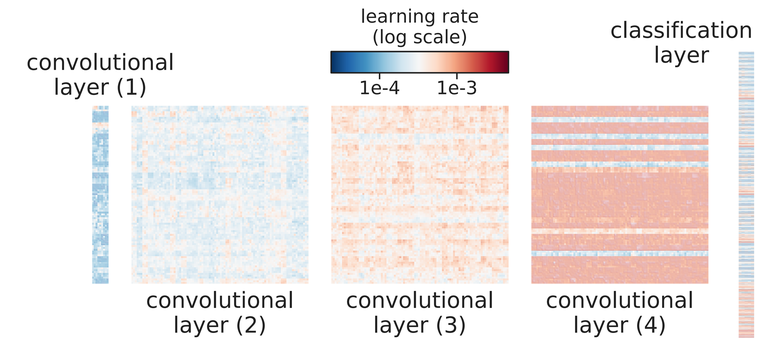
[1] Duchi, Hazan, Singer. *Adaptive Sub-Gradient Methods.* JMLR 2011.
[2] Finn, Abbeel, Levine. *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks.* ICML 2018.
[3] McMahan, Moore, Ramage, Hampson, Aguera y Arcas. *Communication-Efficient Learning of Deep Networks from Decentralized Data.* AISTATS 2017.
[4] Nichol, Achiam, Schulman. *On First-Order Meta-Learning Algorithms.* arXiv 2018.
[5] Ravi, Larochelle. *Optimization as a Model for Few-Shot Learning.* ICLR 2017.
[6] Shalev-Shwartz. *Online Learning and Online Convex Optimization.* FTML 2011.

ARUBA yields a **multi-task per-coordinate learning rate** with provable guarantees:

$$\eta_t = \sqrt{\frac{\sum_{s<t} \varepsilon_s + (\theta_s^* - \phi_s)^2}{\sum_{s<t} \zeta_s + \sum_{i=1}^{m_s} \nabla_{s,i}^2}} \quad \text{for} \;\; \varepsilon_t, \zeta_t = o_t(1)$$

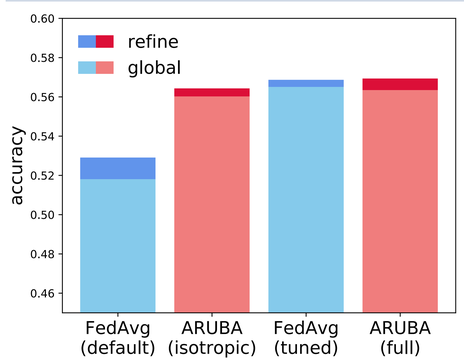
- Comes from regret-upper-bound of preconditioned OGD: $\theta_{t,i+1} = \theta_{t,i} - \eta \odot \nabla_{t,i}$.
- Denominator like AdaGrad [1] but numerator corrected using the distance of learned parameter from initialization.
- Can be directly applied to existing meta-learning algorithms.

Application to Few-Shot Meta-Learning



- Our meta-learning theory holds for Reptile [4], a popular GBML method.
- Applying our multi-task per-coordinate learning rate yields the above step-sizes after meta-training a CNN on Mini-ImageNet [5].
- Following common intuition, learning rates are low for lower-level layers (feature extractors) and high for upper layers (classification).

Application to Federated Learning



- FedAvg [3], a popular algorithm for federated learning (learning on a distributed network of heterogeneous devices), is a special case of Reptile [4].
- Applying our multi-task learning rate yields a way to personalize federated models without on-device fine-tuning.