ARUBA: Efficient and Adaptive Meta-Learning with Provable Guarantees

Misha Khodak

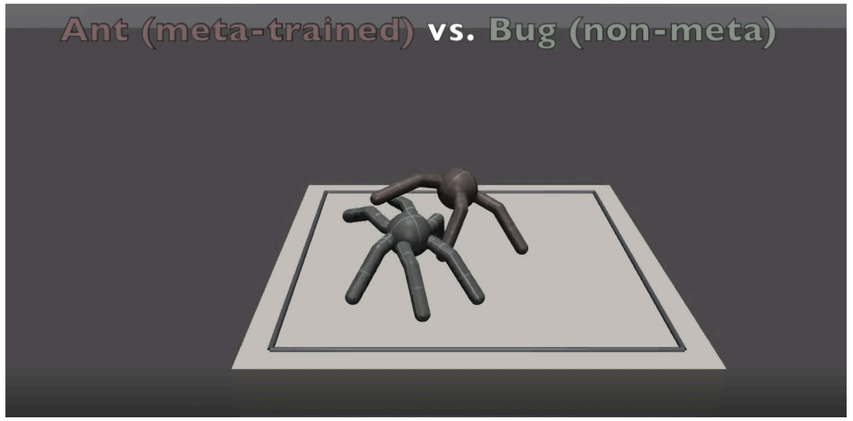
June 15, 2019

Based on joint work with:

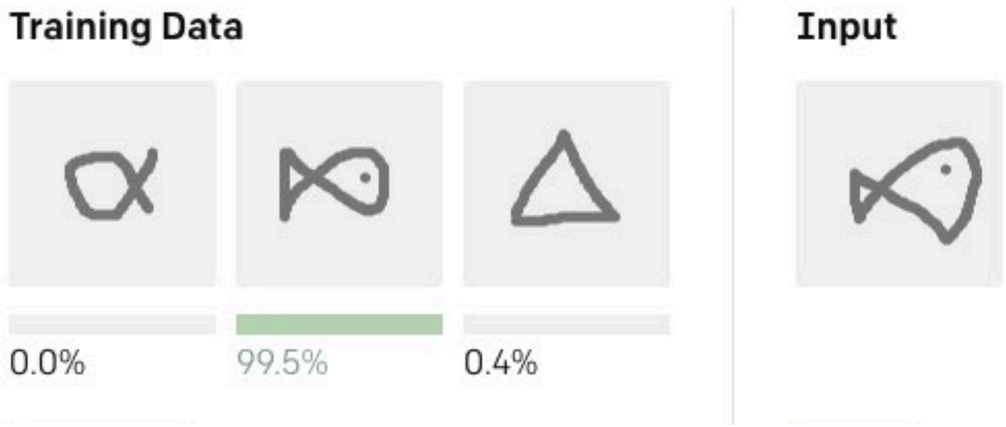
- **Nina Balcan, Ameet Talwalkar**
- **Jeff Li, Sebastian Caldas, Ameet Talwalkar**

Carnegie Mellon University

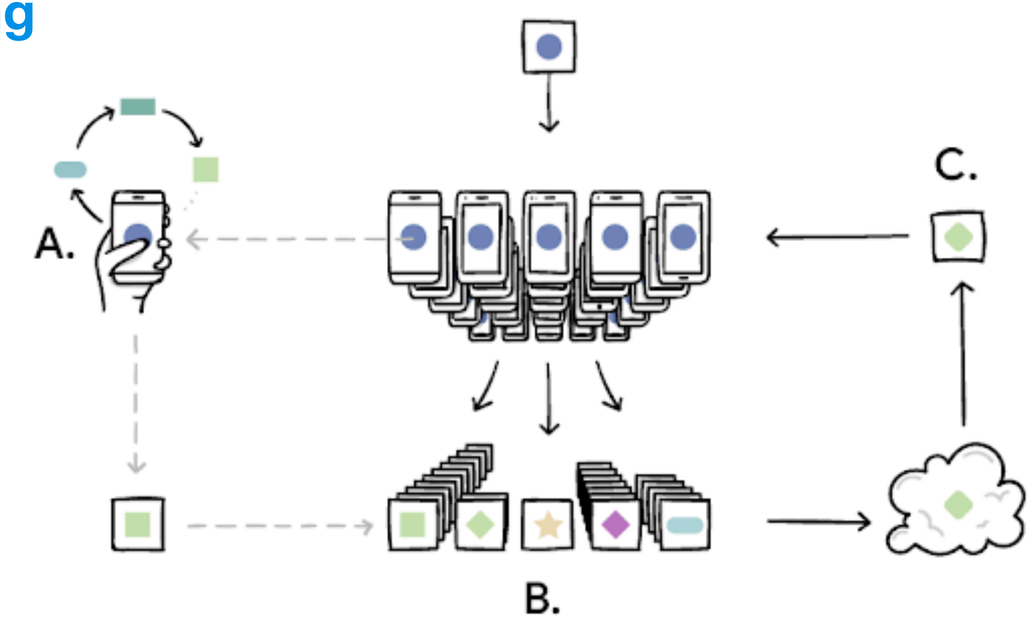
Success of Gradient-Based Meta-Learning (GBML)



Meta Reinforcement Learning



Few-Shot Learning



Federated Learning with Personalization

GBML is simple & flexible

Input: T few-shot training tasks $\{\mathscr{D}\}_1^T$

Algorithm: General; only assumes gradient updates

Output: Initialization ϕ_{GBML} for few-shot test task

GBML is simple & flexible…What is it doing?

Input: T few-shot training tasks $\{\mathscr{D}\}_1^T$

Algorithm: General; only assumes gradient updates

Output: Initialization ϕ_{GBML} for few-shot test task

Why/when do GBML methods work?

GBML is simple & flexible…What is it doing?

Input: T few-shot training tasks $\{\mathscr{D}\}_1^T$

Algorithm: General; only assumes gradient updates

Output: Initialization ϕ_{GBML} for few-shot test task

Why/when do GBML methods work?

- Can we develop **improved training algorithms**?
- Can we **personalize models** while preserving **privacy**?

Many GBML methods are **Online Gradient Descent**, twice

Input: T few-shot training tasks $\{\mathscr{D}\}_1^T$

Algorithm: General; only assumes gradient updates

Output: Initialization ϕ_{GBML} for few-shot test task

Many GBML methods are **Online Gradient Descent**, twice

Input: T few-shot training tasks $\{\mathscr{D}\}_1^T$

Algorithm: General; only assumes gradient updates

Output: Initialization $\phi_{Reptile}$ for few-shot test task

e.g. **Reptile** [Nichol-Achiam-Schulman]

for task $t = 1, \ldots, T$:

$\hat{\theta}_t =$ **Within-task-OGD**$(\mathscr{D}_t, \phi_t)$

update ϕ_{t+1} **using** $\hat{\theta}_t$

Many GBML methods are **Online Gradient Descent**, twice

Input: T few-shot training tasks $\{\mathscr{D}\}_1^T$

Algorithm: General; only assumes gradient updates

Output: Initialization $\phi_{Reptile}$ for few-shot test task

e.g. **Reptile** [Nichol-Achiam-Schulman]

$\phi_{Reptile} =$ **Across-task-OGD**$(\{\mathscr{D}\}_1^T)$

for task $t = 1, \ldots, T$:

$\hat{\theta}_t =$ **Within-task-OGD**$(\mathscr{D}_t, \phi_t)$

update ϕ_{t+1} **using** $\hat{\theta}_t$

Many GBML methods are **Online Gradient Descent**, twice

Input: T few-shot training tasks $\{\mathscr{D}\}_1^T$

Algorithm: General; only assumes gradient updates

Output: Initialization ϕ_{MAML} for few-shot test task

MAML [Finn-Abbeel-Levine]

- Replace by OGD by GD
- Update involves holdout data

for task $t = 1, \ldots, T$:

$\hat{\theta}_t =$ **Within-task-OGD**$(\mathscr{D}_t, \phi_t)$

update ϕ_{t+1} **using** $\hat{\theta}_t$

Many GBML methods are **Online Gradient Descent**, twice

Input: T few-shot training tasks $\{\mathscr{D}\}_1^T$

Algorithm: General; only assumes gradient updates

Output: Initialization ϕ_{FedAvg} for few-shot test task

FedAvg [McMahan et al.]

- Process k tasks in parallel
- Update aggregates over k tasks

for task $t = 1, \ldots, T$:

$\hat{\theta}_t =$ **Within-task-OGD**$(\mathscr{D}_t, \phi_t)$

update ϕ_{t+1} **using** $\hat{\theta}_t$

GBML through the Lens of Online Learning

Online Learning

for $i = 1, \ldots, m$

pick action $\theta_i \in \Theta$

suffer loss $\ell_i(\theta_i)$

GBML through the Lens of Online Learning

Measure per-task performance via **Regret**

$$R = \sum_{i=1}^{m} \ell_i(\theta_i) - \ell_i(\theta^*)$$

best fixed action
in hindsight

Online Learning

for $i = 1, \ldots, m$

pick action $\theta_i \in \Theta$

suffer loss $\ell_i(\theta_i)$

GBML through the Lens of Online Learning

Measure per-task performance via **Regret**

$$R = \sum_{i=1}^{m} \ell_i(\theta_i) - \ell_i(\theta^*)$$

best fixed action
in hindsight

Online Learning

for $i = 1, \ldots, m$

pick action $\theta_i \in \Theta$

suffer loss $\ell_i(\theta_i)$

Non-IID Data / Tasks: Models realistic settings (e.g. mobile, RL data; lifelong learning)

GBML through the Lens of Online Learning

Measure per-task performance via **Regret**

$$R = \sum_{i=1}^{m} \ell_i(\theta_i) - \ell_i(\theta^*)$$

best fixed action
in hindsight

Online Learning

for $i = 1, \ldots, m$

pick action $\theta_i \in \Theta$

suffer loss $\ell_i(\theta_i)$

Non-IID Data / Tasks: Models realistic settings (e.g. mobile, RL data; lifelong learning)

IID Implications: Online-to-batch conversion results

GBML through the Lens of Online Learning

Measure per-task performance via **Regret**

$$R = \sum_{i=1}^{m} \ell_i(\theta_i) - \ell_i(\theta^*)$$

best fixed action
in hindsight

Online Learning

for $i = 1, \ldots, m$

pick action $\theta_i \in \Theta$

suffer loss $\ell_i(\theta_i)$

Non-IID Data / Tasks: Models realistic settings (e.g. mobile, RL data; lifelong learning)

IID Implications: Online-to-batch conversion results

Generality: Can adapt / generalize numerous online learning results to GBML setup

ARUBA: Novel theoretical Framework for GBML

Nina Balcan **Ameet Talwalkar**

ARUBA: Novel theoretical Framework for GBML

Provides **regret bounds** for a sequence of online learning problems

ARUBA: Novel theoretical Framework for GBML

Provides **regret bounds** for a sequence of online learning problems

Theoretical focus on **online convex optimization**

ARUBA: Novel theoretical Framework for GBML

Provides **regret bounds** for a sequence of online learning problems

Theoretical focus on **online convex optimization**

Practical implications in **nonconvex settings**

ARUBA Framework

- Few Shot Learning and GBML
- An Illustrative Result

Applications

Single-Task Few-Shot Learning

Training Data	Hypothesis Class	Loss Function
$(x_1, y_1), \ldots, (x_m, y_m)$	$\{f_\theta : \mathcal{X} \mapsto \mathcal{Y} : \theta \in \Theta \subset \mathbb{R}^d\}$	$\ell_i(\theta) = L(f_\theta(x_i), y_i)$

$L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$

Single-Task Few-Shot Learning

Training Data	Hypothesis Class	Loss Function
$(x_1, y_1), \ldots, (x_m, y_m)$	$\{f_\theta : \mathcal{X} \mapsto \mathcal{Y} : \theta \in \Theta \subset \mathbb{R}^d\}$	$\ell_i(\theta) = L(f_\theta(x_i), y_i)$

$L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$

Online Gradient Descent (OGD)

randomly initialize $\theta_1 \in \Theta, \quad \eta > 0$

for $i = 1, \ldots, m$

$$\theta_{i+1} \leftarrow \theta_i - \eta \nabla \ell_i(\theta_i)$$

Single-Task Few-Shot Learning

Training Data	Hypothesis Class	Loss Function
$(x_1, y_1), \ldots, (x_m, y_m)$	$\{f_\theta : \mathcal{X} \mapsto \mathcal{Y} : \theta \in \Theta \subset \mathbb{R}^d\}$	$\ell_i(\theta) = L(f_\theta(x_i), y_i)$

$L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$

Online Gradient Descent (OGD)

randomly initialize $\theta_1 \in \Theta, \quad \eta > 0$

for $i = 1, \ldots, m$

$$\theta_{i+1} \leftarrow \theta_i - \eta \nabla \ell_i(\theta_i)$$

Cannot hope to do well when $\boldsymbol{m}$ **is small**

Single-Task Regret

Training Data	Hypothesis Class	Loss Function
$(x_1, y_1), \ldots, (x_m, y_m)$	$\{f_\theta : \mathcal{X} \mapsto \mathcal{Y} : \theta \in \Theta \subset \mathbb{R}^d\}$	$\ell_i(\theta) = L(f_\theta(x_i), y_i)$

Measure performance via Regret

$$R = \sum_{i=1}^{m} \ell_i(\theta_i) - \ell_i(\theta^*)$$

best fixed action in hindsight

Online Gradient Descent (OGD)

randomly initialize $\theta_1 \in \Theta, \quad \eta > 0$

for $i = 1, \ldots, m$

$$\theta_{i+1} \leftarrow \theta_i - \eta \nabla \ell_i(\theta_i)$$

suffer $\ell_i(\theta_i)$

Single-Task Regret

Training Data	Hypothesis Class	Loss Function
$(x_1, y_1), \ldots, (x_m, y_m)$	$\{f_\theta : \mathcal{X} \mapsto \mathcal{Y} : \theta \in \Theta \subset \mathbb{R}^d\}$	$\ell_i(\theta) = L(f_\theta(x_i), y_i)$

Measure performance via Regret

$$R = \sum_{i=1}^{m} \ell_i(\theta_i) - \ell_i(\theta^*)$$

best fixed action in hindsight

Online Gradient Descent (OGD)

Size of Action Space: $D = \mathrm{diam}(\Theta)$

OGD upper-bound: $R = \mathcal{O}(D\sqrt{m})$

[Abernethy-Bartlett-Rakhlin-Tewari]

Single-Task Regret

Training Data	Hypothesis Class	Loss Function
$(x_1, y_1), \ldots, (x_m, y_m)$	$\{f_\theta : \mathcal{X} \mapsto \mathcal{Y} : \theta \in \Theta \subset \mathbb{R}^d\}$	$\ell_i(\theta) = L(f_\theta(x_i), y_i)$

Measure performance via Regret

$$R = \sum_{i=1}^{m} \ell_i(\theta_i) - \ell_i(\theta^*)$$

best fixed action in hindsight

Online Gradient Descent (OGD)

Size of Action Space: $D = \mathrm{diam}(\Theta)$

OGD upper-bound: $R = \mathcal{O}(D\sqrt{m})$

Matching lower-bound: $R = \Omega\left(D\sqrt{m}\right)$
(for any algorithm)

[Abernethy-Bartlett-Rakhlin-Tewari]

Single-Task Regret

Training Data	Hypothesis Class	Loss Function
$(x_1, y_1), \ldots, (x_m, y_m)$	$\{f_\theta : \mathcal{X} \mapsto \mathcal{Y} : \theta \in \Theta \subset \mathbb{R}^d\}$	$\ell_i(\theta) = L(f_\theta(x_i), y_i)$

Measure performance via Regret

$$R = \sum_{i=1}^{m} \ell_i(\theta_i) - \ell_i(\theta^*)$$

best fixed action in hindsight

Key Question: can GBML do better?

Online Gradient Descent (OGD)

Size of Action Space: $D = \mathrm{diam}(\Theta)$

OGD upper-bound: $R = \mathcal{O}(D\sqrt{m})$

Matching lower-bound: $R = \Omega\left(D\sqrt{m}\right)$
(for any algorithm)

[Abernethy-Bartlett-Rakhlin-Tewari]

GBML Meta-Testing, i.e., using ϕ_{GBML}

Training Data	Hypothesis Class	Loss Function
$(x_1, y_1), \ldots, (x_m, y_m)$	$\{f_\theta : \mathcal{X} \mapsto \mathcal{Y} : \theta \in \Theta \subset \mathbb{R}^d\}$	$\ell_i(\theta) = L(f_\theta(x_i), y_i)$

Learn an initialization
sequentially from previous t tasks

Key Question: can GBML do better **on-average across tasks**?

Online Gradient Descent (OGD)

meta initialize $\phi_t \in \Theta, \quad \eta > 0$

for $i = 1, \ldots, m$

$$\theta_{i+1} \leftarrow \theta_i - \eta \nabla \ell_i(\theta_i)$$

Average Regret and Task Similarity

Training Data	Hypothesis Class	Loss Function
$(x_{t,1}, y_{t,1}), \ldots, (x_{T,m}, y_{T,m})$	$\{f_\theta : \mathcal{X} \mapsto \mathcal{Y} : \theta \in \Theta \subset \mathbb{R}^d\}$	$\ell_{t,i}(\theta) = L(f_\theta(x_{t,i}), y_{t,i})$

Average Regret and Task Similarity

Training Data	Hypothesis Class	Loss Function
$(x_{t,1}, y_{t,1}), \ldots, (x_{T,m}, y_{T,m})$	$\{f_\theta : \mathcal{X} \mapsto \mathcal{Y} : \theta \in \Theta \subset \mathbb{R}^d\}$	$\ell_{t,i}(\theta) = L(f_\theta(x_{t,i}), y_{t,i})$

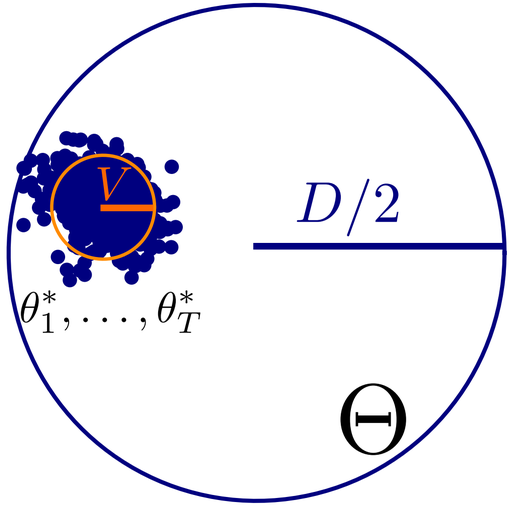
Average Regret: $$\bar{R} = \frac{1}{T}\sum_{t=1}^{T} R_t = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{m} \ell_{t,i}(\theta_{t,i}) - \ell_{t,i}(\theta_t^*)$$

Average Regret and Task Similarity

Training Data	Hypothesis Class	Loss Function
$(x_{t,1}, y_{t,1}), \ldots, (x_{T,m}, y_{T,m})$	$\{f_\theta : \mathcal{X} \mapsto \mathcal{Y} : \theta \in \Theta \subset \mathbb{R}^d\}$	$\ell_{t,i}(\theta) = L(f_\theta(x_{t,i}), y_{t,i})$

Average Regret: $$\bar{R} = \frac{1}{T}\sum_{t=1}^{T} R_t = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{m} \ell_{t,i}(\theta_{t,i}) - \ell_{t,i}(\theta_t^*)$$

Task Similarity: $$V^2 = \min_{\phi \in \Theta} \frac{1}{T}\sum_{t=1}^{T} \|\theta_t^* - \phi\|_2^2$$



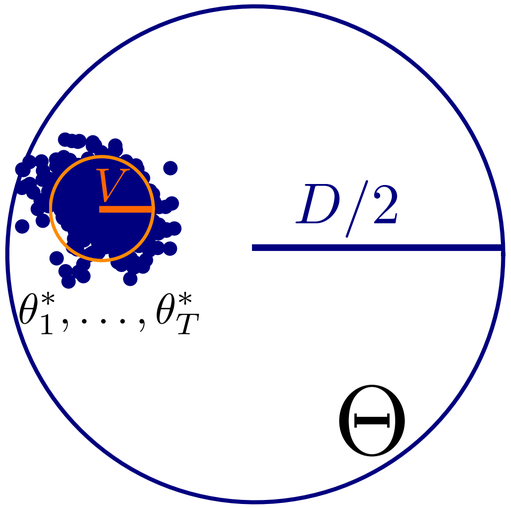
V is small when optimal parameters are close together

ARUBA: An Illustrative Result

Our Guarantee: $\bar{R} = \mathcal{O}\left(V + \frac{\log T}{VT}\right)\sqrt{m}$

Single-Task Lower Bound: $R_t = \Omega\left(D\sqrt{m}\right)$

Multi-Task Lower Bound: $\bar{R} = \Omega\left(V\sqrt{m}\right)$



Task Similarity V

When **optimal task parameters are close together**, GBML leads to much **better average performance**

Recall: Reptile Algorithm

$\phi_{Reptile} =$ **Across-task-OGD**$(\{\mathscr{D}\}_1^T)$

for task $t = 1, \ldots, T$:

$\hat{\theta}_t =$ **Within-task-OGD**$(\mathscr{D}_t, \phi_t)$

update ϕ_{t+1} **using** $\hat{\theta}_t$

Recall: Reptile Algorithm

$\phi_{Reptile} =$ **Across-task-OGD**$(\{\mathscr{D}\}_1^T)$

for task $t = 1, \ldots, T$:

$\hat{\theta}_t =$ **Within-task-OGD**$(\mathscr{D}_t, \phi_t)$

update ϕ_{t+1} **using** θ_t^*

assume oracle access to optimum in hindsight

can be relaxed under nondegeneracy assumption

Main Observation

Single-task regret guarantees are often nice, data-dependent functions of the algorithm parameters.

Main Observation

Single-task regret guarantees are often nice, data-dependent functions of the algorithm parameters.

e.g. **OGD from** ϕ

regret-upper-bound (RUB)

$$R_t = \sum_{i=1}^{m} \ell_{t,i}(\theta) - \ell_{t,i}(\theta_t^*) \leq \hat{R}_t(\phi) = \frac{\|\theta_t^* - \phi\|_2^2}{\eta} + \eta m$$

Main Observation

Single-task regret guarantees are often nice, data-dependent functions of the algorithm parameters.

e.g. **OGD from** ϕ

regret-upper-bound (RUB)

$$R_t = \sum_{i=1}^{m} \ell_{t,i}(\theta) - \ell_{t,i}(\theta_t^*) \leq \hat{R}_t(\phi) = \frac{\|\theta_t^* - \phi\|_2^2}{\eta} + \eta m$$

Reduces GBML to OCO over a sequence of **regret-upper-bounds**

3 Key Steps of ARUBA Framework

1. **Within-task OGD**: **RUB** controls performance as data-dependent (θ^*) function of ϕ

Step 1: Substitute RUB

$$\frac{1}{T}\sum_{t=1}^{T} R_t \leq \frac{1}{T}\sum_{t=1}^{T} \hat{R}_t(\phi_t) =$$

3 Key Steps of ARUBA Framework

1. **Within-task OGD**: **RUB** controls performance as data-dependent (θ^*) function of ϕ

Step 1: Substitute RUB **Addition/Subtraction**

$$\frac{1}{T}\sum_{t=1}^{T} R_t \leq \frac{1}{T}\sum_{t=1}^{T} \hat{R}_t(\phi_t) = \frac{1}{T}\left(\sum_{t=1}^{T} \hat{R}_t(\phi_t) - \min_{\phi\in\Theta}\sum_{t=1}^{T} \hat{R}_t(\phi)\right) + \min_{\phi\in\Theta}\frac{1}{T}\sum_{t=1}^{T} \hat{R}_t(\phi)$$

3 Key Steps of ARUBA Framework

1. **Within-task OGD**: **RUB** controls performance as data-dependent (θ^*) function of ϕ

Step 1: Substitute RUB **Addition/Subtraction**

$$\frac{1}{T}\sum_{t=1}^{T} R_t \leq \frac{1}{T}\sum_{t=1}^{T} \hat{R}_t(\phi_t) = \frac{1}{T}\left(\sum_{t=1}^{T} \hat{R}_t(\phi_t) - \min_{\phi\in\Theta}\sum_{t=1}^{T} \hat{R}_t(\phi)\right) + \min_{\phi\in\Theta}\frac{1}{T}\sum_{t=1}^{T} \hat{R}_t(\phi)$$

OGD **R**egret-**U**pper-**B**ound (RUB)

$$\hat{R}(\phi) = \frac{\|\theta^* - \phi\|_2^2}{2\eta} + \eta m$$

3 Key Steps of ARUBA Framework

1. **Within-task OGD**: **RUB** controls performance as data-dependent (θ^*) function of ϕ
2. **Across-task OGD**: Choose initializations ϕ_t such that RUB is small on **A**verage

Step 1: Substitute RUB **Addition/Subtraction**

$$\frac{1}{T}\sum_{t=1}^{T} R_t \leq \frac{1}{T}\sum_{t=1}^{T} \hat{R}_t(\phi_t) = \frac{1}{T}\left(\sum_{t=1}^{T} \hat{R}_t(\phi_t) - \min_{\phi\in\Theta}\sum_{t=1}^{T} \hat{R}_t(\phi)\right) + \min_{\phi\in\Theta}\frac{1}{T}\sum_{t=1}^{T} \hat{R}_t(\phi)$$

Step 2: Across-task OGD

$$= \frac{1}{T}\left(\sum_{t=1}^{T} \frac{\|\theta_t^* - \phi_t\|_2^2}{2\eta} - \min_{\phi\in\Theta}\frac{\|\theta_t^* - \phi\|_2^2}{2\eta}\right)$$

3 Key Steps of ARUBA Framework

1. **Within-task OGD**: **RUB** controls performance as data-dependent (θ^*) function of ϕ
2. **Across-task OGD**: Choose initializations ϕ_t such that RUB is small on **A**verage

Step 1: Substitute RUB **Addition/Subtraction**

$$\frac{1}{T}\sum_{t=1}^{T} R_t \leq \frac{1}{T}\sum_{t=1}^{T} \hat{R}_t(\phi_t) = \frac{1}{T}\left(\sum_{t=1}^{T} \hat{R}_t(\phi_t) - \min_{\phi\in\Theta}\sum_{t=1}^{T} \hat{R}_t(\phi)\right) + \min_{\phi\in\Theta}\frac{1}{T}\sum_{t=1}^{T} \hat{R}_t(\phi)$$

Step 2: Across-task OGD

$$= \frac{1}{T}\left(\sum_{t=1}^{T}\frac{\|\theta_t^* - \phi_t\|_2^2}{2\eta} - \min_{\phi\in\Theta}\frac{\|\theta_t^* - \phi\|_2^2}{2\eta}\right)$$

$$= \mathcal{O}\left(\frac{\log T}{VT}\right)\sqrt{m}$$

3 Key Steps of ARUBA Framework

1. **Within-task OGD**: **RUB** controls performance as data-dependent (θ^*) function of ϕ
2. **Across-task OGD**: Choose initializations ϕ_t such that RUB is small on **A**verage
3. **Task-Relatedness**: **A**nalyze its impact on resulting bound

Step 1: Substitute RUB **Addition/Subtraction**

$$\frac{1}{T}\sum_{t=1}^{T} R_t \leq \frac{1}{T}\sum_{t=1}^{T} \hat{R}_t(\phi_t) = \frac{1}{T}\left(\sum_{t=1}^{T} \hat{R}_t(\phi_t) - \min_{\phi\in\Theta}\sum_{t=1}^{T} \hat{R}_t(\phi)\right) + \min_{\phi\in\Theta}\frac{1}{T}\sum_{t=1}^{T} \hat{R}_t(\phi)$$

Step 2: Across-task OGD **Step 3: Impact of Task Relatedness** V

$$= \frac{1}{T}\left(\sum_{t=1}^{T}\frac{\|\theta_t^* - \phi_t\|_2^2}{2\eta} - \min_{\phi\in\Theta}\frac{\|\theta_t^* - \phi\|_2^2}{2\eta}\right) + \boxed{\min_{\phi\in\Theta}\frac{1}{T}\sum_{t=1}^{T}\frac{\|\theta_t^* - \phi\|_2^2}{2\eta}} + \eta m$$

$$= \mathcal{O}\left(\frac{\log T}{VT}\right)\sqrt{m}$$

3 Key Steps of ARUBA Framework

1. **Within-task OGD**: **RUB** controls performance as data-dependent (θ^*) function of ϕ
2. **Across-task OGD**: Choose initializations ϕ_t such that RUB is small on **A**verage
3. **Task-Relatedness**: **A**nalyze its impact on resulting bound

Step 1: Substitute RUB **Addition/Subtraction**

$$\frac{1}{T}\sum_{t=1}^{T} R_t \leq \frac{1}{T}\sum_{t=1}^{T} \hat{R}_t(\phi_t) = \frac{1}{T}\left(\sum_{t=1}^{T} \hat{R}_t(\phi_t) - \min_{\phi\in\Theta}\sum_{t=1}^{T} \hat{R}_t(\phi)\right) + \min_{\phi\in\Theta}\frac{1}{T}\sum_{t=1}^{T} \hat{R}_t(\phi)$$

Step 2: Across-task OGD **Step 3: Impact of Task Relatedness** V

$$= \frac{1}{T}\left(\sum_{t=1}^{T}\frac{\|\theta_t^* - \phi_t\|_2^2}{2\eta} - \min_{\phi\in\Theta}\frac{\|\theta_t^* - \phi\|_2^2}{2\eta}\right) + \boxed{\min_{\phi\in\Theta}\frac{1}{T}\sum_{t=1}^{T}\frac{\|\theta_t^* - \phi\|_2^2}{2\eta}} + \eta m$$

$$= \mathcal{O}\left(\frac{\log T}{VT}\right)\sqrt{m} \quad + \quad \frac{3}{2}V\sqrt{m}$$

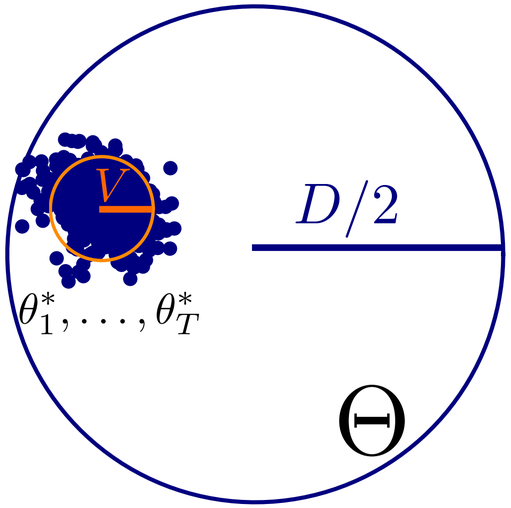
substitute $\eta = V/\sqrt{m}$

Recap: What have we achieved?

Our Guarantee: $\bar{R} = \mathcal{O}\left(V + \frac{\log T}{VT}\right)\sqrt{m}$

Single-Task Lower Bound: $R_t = \Omega\left(D\sqrt{m}\right)$

Multi-Task Lower Bound: $\bar{R} = \Omega\left(V\sqrt{m}\right)$



Task Similarity V

When **optimal task parameters are close together**, GBML leads to much **better average performance**

What can we get by applying ARUBA?

Adaptivity

- Learn any base-learner parameter from data
- e.g., **improved training algorithms** by learning ϕ and η simultaneously

What can we get by applying ARUBA?

Adaptivity

- Learn any base-learner parameter from data
- e.g., **improved training algorithms** by learning ϕ and η simultaneously

Generality

- Low-dynamic-regret algorithms for **changing task-environments**
- **Stronger online-to-batch conversions** for faster statistical rates
- Specialized within-task algorithms, e.g., satisfying **privacy guarantees**

ARUBA in Context of Three ICML'19 Papers

Excess transfer risk bounds on regularized SGD via online-to-batch conversion [Denevi-Ciliberto-Grazzi-Pontil]

Online learnability of MAML via Follow-the-Leader [Finn-Rajeswaran-Kakade-Levine]

Average regret bounds for Online Mirror Descent meta-algo under max-deviation assumption on task-parameters [K-Balcan-Talwalkar]

ARUBA Framework

Applications

- Adaptivity for Improved Training
- Federated Learning & Privacy-Preserving GBML

Is learning an initialization good enough?

per-coordinate learning rate

gradient on sample i from task t

$$\theta_{t,i+1} = \theta_{t,i} - \eta_{t,i} \odot \nabla_{t,i}$$

Is learning an initialization good enough?

per-coordinate learning rate

gradient on sample i from task t

$$\theta_{t,i+1} = \theta_{t,i} - \eta_{t,i} \odot \nabla_{t,i}$$

Adaptively preconditioned gradient descent is popular in GBML

- Reptile uses Adam with accumulated gradient information
- Meta-SGD learns a per-coordinate learning rate for MAML [Li et al.]

Is learning an initialization good enough?

per-coordinate learning rate

gradient on sample i from task t

$$\theta_{t,i+1} = \theta_{t,i} - \eta_{t,i} \odot \nabla_{t,i}$$

Adaptively preconditioned gradient descent is popular in GBML

- Reptile uses Adam with accumulated gradient information
- Meta-SGD learns a per-coordinate learning rate for MAML [Li et al.]

Can we do this rigorously?

Applying ARUBA: the regret-upper-bound

Single-task regret guarantees are often nice, data-dependent functions of the algorithm parameters.

η-preconditioned OGD from ϕ

$$\hat{R}_t(\phi, \eta) = \|\theta_t^* - \phi\|^2_{\frac{1}{2\eta}} + \sum_{i=1}^{m} \|\nabla_{t,i}\|^2_\eta$$

Mahalanobis norm

Adaptive ARUBA

Setting the learning rate

$$B_{t,j} = \frac{1}{2} \sum_{s<t} (\phi_{s,j} - \theta^*_{s,j})^2$$

sum of sq. distances from initialization

$$G_{t,j} = \sum_{s<t} \sum_{i=1}^{m} \nabla^2_{t,i,j}$$

sum of sq. gradients

learning rate coordinate j at task t

$$\eta_{t,j} = \sqrt{\frac{B_{t,j} + \varepsilon_t}{G_{t,j} + \zeta_t}}$$

$$\varepsilon_t = o(t), \quad \zeta_t = o(t)\sqrt{m}$$

smoothing terms

Adaptive ARUBA

Setting the learning rate

$$B_{t,j} = \frac{1}{2} \sum_{s<t} (\phi_{s,j} - \theta^*_{s,j})^2$$

sum of sq. distances from initialization

$$G_{t,j} = \sum_{s<t} \sum_{i=1}^{m} \nabla^2_{t,i,j}$$

sum of sq. gradients

learning rate coordinate j at task t

$$\eta_{t,j} = \sqrt{\frac{B_{t,j} + \varepsilon_t}{G_{t,j} + \zeta_t}}$$

$$\varepsilon_t = o(t), \quad \zeta_t = o(t)\sqrt{m}$$

smoothing terms

Guarantee: $\tilde{\mathcal{O}}\left(1/T^{2/5}\right)$ - convergence of average regret to that of always using the optimal init and diagonal preconditioner

Connection to AdaGrad

Applying result to practical GBML with OGD base learners:

sum of squared distances traveled on each task

$$\eta_{t,j} = \sqrt{\frac{B_{t,j} + \varepsilon_t}{G_{t,j} + \zeta_t}}$$

sum of all squared gradients across tasks

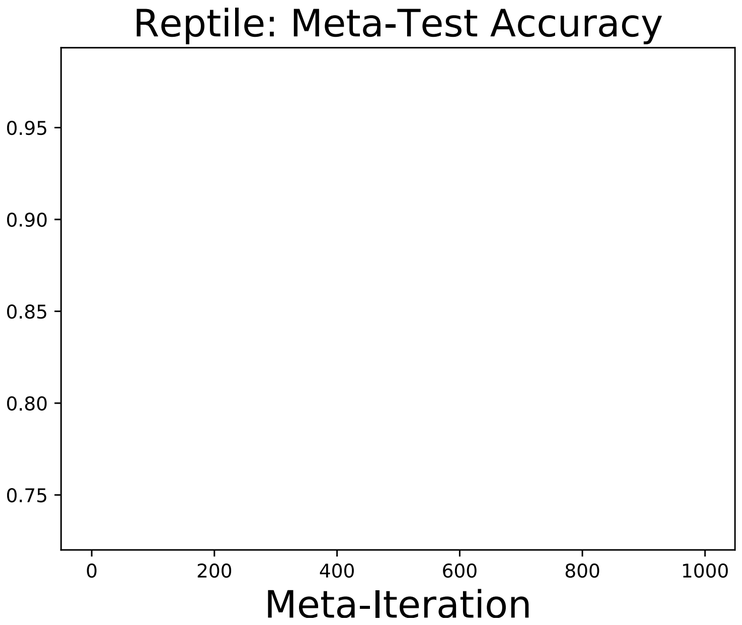
Compare to AdaGrad:

$$\eta_{t,j} = \sqrt{\frac{1}{G_{t,j} + \delta}}, \qquad \delta = O(1)$$

smoothing term

Improved Training via Adaptive ARUBA

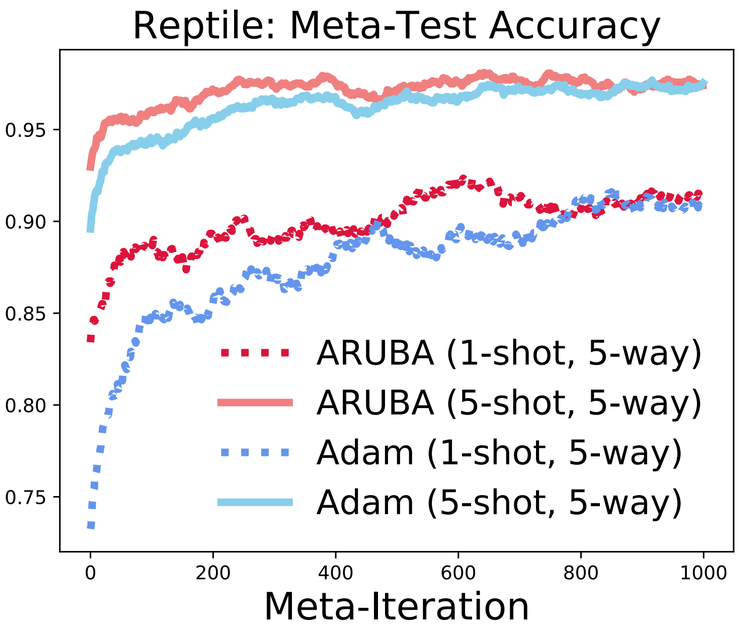
Reptile: preconditioning with ARUBA instead of Adam

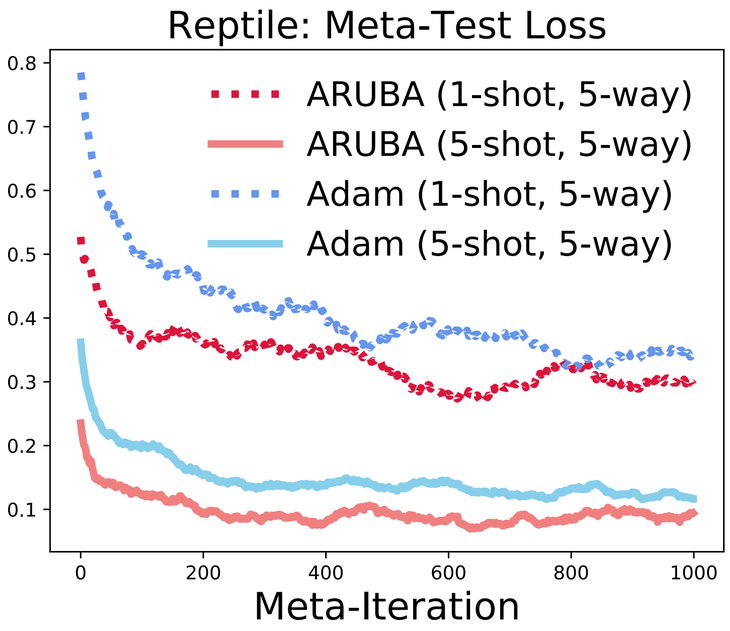


Results on Omniglot character classification task, averaged over three runs

Improved Training via Adaptive ARUBA

Reptile: preconditioning with ARUBA instead of Adam





Results on Omniglot character classification task, averaged over three runs

ARUBA Framework

Applications

- Adaptivity for Improved Training
- Federated Learning & Privacy-Preserving GBML

Personalized Federated Learning

- Massively distributed
- Small sample sizes
- Privacy concerns
- Non-IID data and tasks
- Underlying task similarity

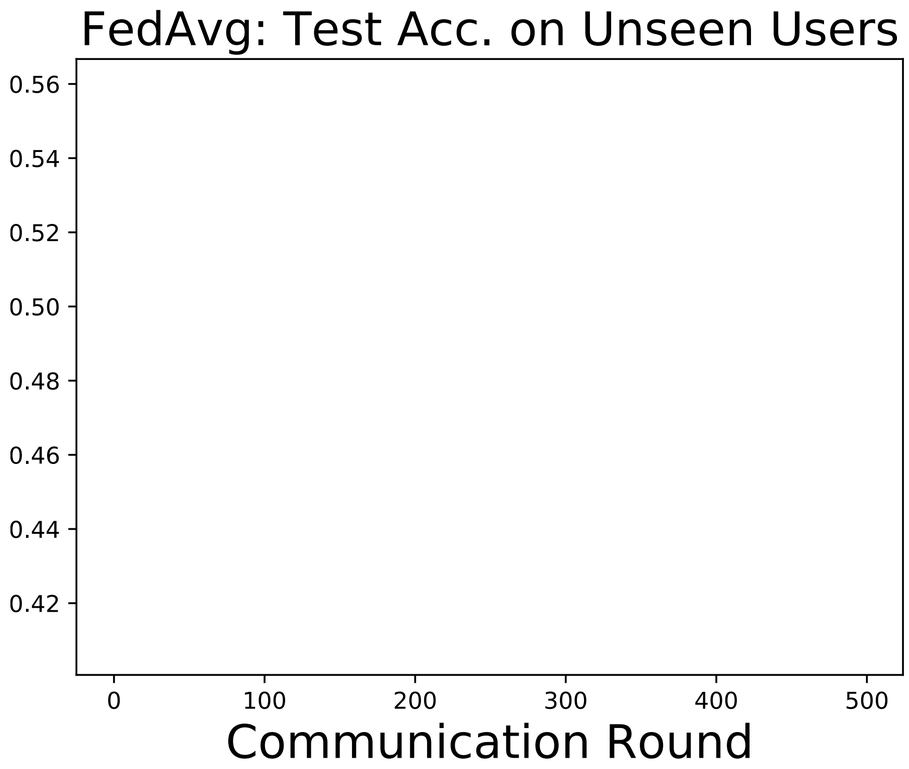
FedAvg ≈ Reptile [with a batch-averaged meta-update]

FedAvg ≈ Reptile [with a batch-averaged meta-update]

- Most popular algorithm in federated learning
- Usually run without personalization - just use the meta-initialization within-task

Personalization in Federated Learning via Adaptive ARUBA

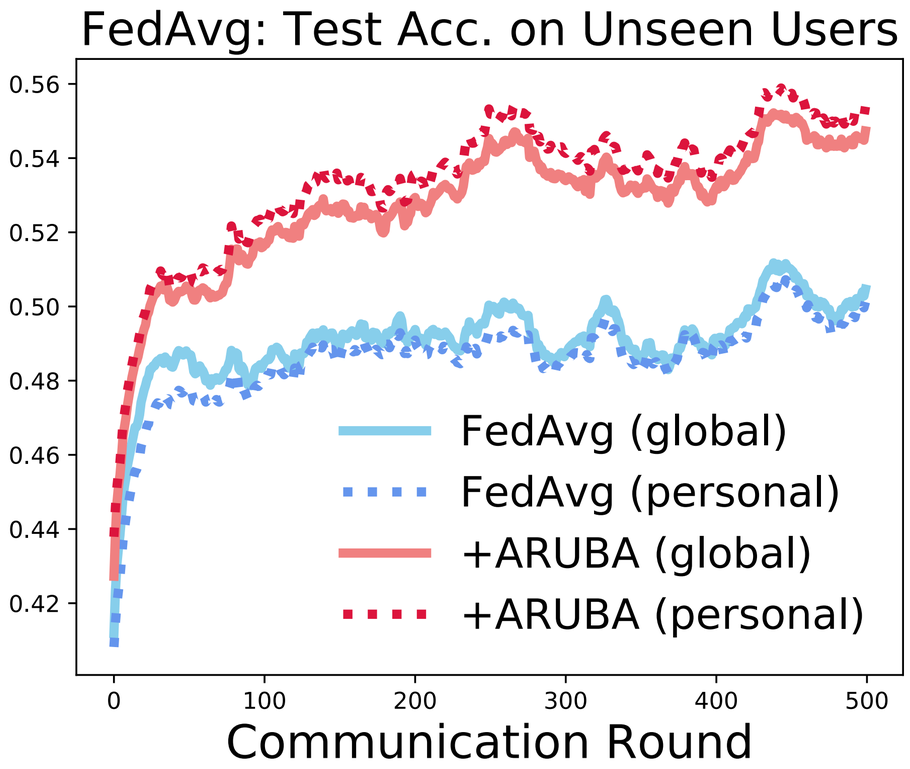
- Meta-training: run FedAvg with ARUBA optimizer within-task



Results on Shakespeare next-character prediction task

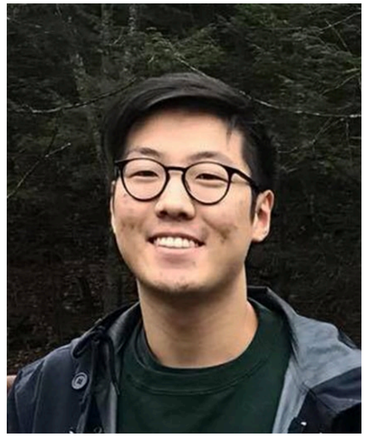
Personalization in Federated Learning via Adaptive ARUBA

- Meta-training: run FedAvg with ARUBA optimizer within-task
- Meta-testing - use (preconditioned) OGD to learn a personalized model for each user



Results on Shakespeare next-character prediction task

Private and Low-Risk Federated Learning via ARUBA



Jeff Li **Sebastian Caldas** **Ameet Talwalkar**

Private and Low-Risk Federated Learning via ARUBA

- Approach:
  - ARUBA handles approximate meta-updates, e.g. due to noise from differential privacy
  - use private OCO algorithms within-task to get per-sample privacy

Private and Low-Risk Federated Learning via ARUBA

- Approach:
  - ARUBA handles approximate meta-updates, e.g. due to noise from differential privacy
  - use private OCO algorithms within-task to get per-sample privacy

- analysis of their regret-upper-bounds yields a GBML method that's private **even to the central aggregator** and has **risk bounds improving with task-similarity**.

Summary

ARUBA: a framework to obtain GBML algorithms with provable and mathematically interpretable guarantees via reduction to OCO.

New methods for meta-training, federated learning, private GBML.

Future directions:

- Beyond adversarial analysis within-task — can the base-learners be statistical or RL algos?
- Better multi-task optimizers, e.g. for regimes beyond few-shot learning

Thank You!

ARUBA: https://arxiv.org/abs/1906.02717

More Info: http://www.cs.cmu.edu/~mkhodak/