Optimization-Based Meta-Learning

CS 330

Course Reminders

Project group form due **tonight**.
(for assigning project mentors)

Homework 1 due **Wednesday**

Following up on some high-res feedback:

- optional reading materials posted on website
- coding environment set-up: detailed guidance on Azure for HW2 & HW3
- managing questions in lecture

Plan for Today

Recap
- Meta-learning problem & black-box meta-learning

Optimization Meta-Learning } **Part of Homework 2!**
- Overall approach
- Compare: optimization-based vs. black-box
- Challenges & solutions
- Case study of land cover classification (time-permitting)

Goals for by the end of lecture:
- Basics of optimization-based meta-learning techniques (& how to implement)
- Trade-offs between black-box and optimization-based meta-learning

Problem Settings Recap

Multi-Task Learning

Solve multiple tasks $\mathcal{T}_1, \cdots, \mathcal{T}_T$ at once.

$$\min_{\theta} \sum_{i=1}^{T} \mathscr{L}_i(\theta, \mathcal{D}_i)$$

Transfer Learning

Solve target task $\mathcal{T}_b$ after solving source task $\mathcal{T}_a$ by *transferring* knowledge learned from $\mathcal{T}_a$

Meta-Learning Problem

Transfer Learning with Many Source Tasks

Given data from $\mathcal{T}_1, \ldots, \mathcal{T}_n$, solve new task $\mathcal{T}_{\text{test}}$ more quickly / proficiently / stably

Example Meta-Learning Problem

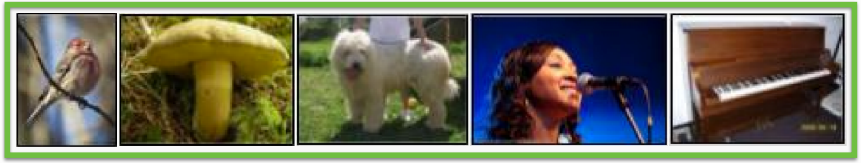
5-way, **1**-shot image classification (MiniImagenet)

Given 1 example of 5 classes: Classify new examples

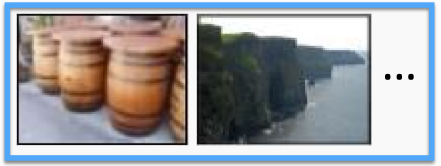
meta-test

meta-training

$\mathcal{T}_1$



$\mathcal{T}_2$



Can replace image classification with: regression, language generation, skill learning, **any ML problem**

Black-Box Adaptation

those params are updated by the meta learning procedure.

so it is a hypernetwork!

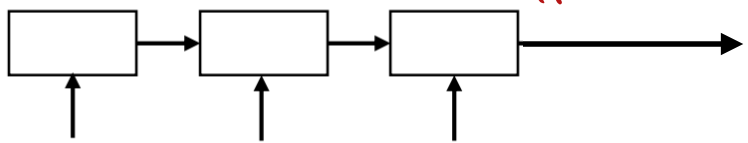
Something like an RNN or LSTM

f_θ

have that NN output a set of params for that task

ϕ_i

alternatively, it could just give us a hidden state of the RNN (instead of giving us an entire NN params) ⇒ just use the RNN from before.



4

y^{ts}

$\mathcal{D}_i^{\text{tr}}$

x^{ts}

make predictions for new inputs for that task

general form:

$$y^{\text{ts}} = f_{\text{black-box}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}})$$

+ **expressive**

↳ i.e. can represent lots of diff. learning algorithms

- **challenging optimization** problem

How else can we represent $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$?

going from training datasets to a set of task parameters

↳ NN parameters optimized to execute a certain task.

What if we treat it as an **optimization** procedure?

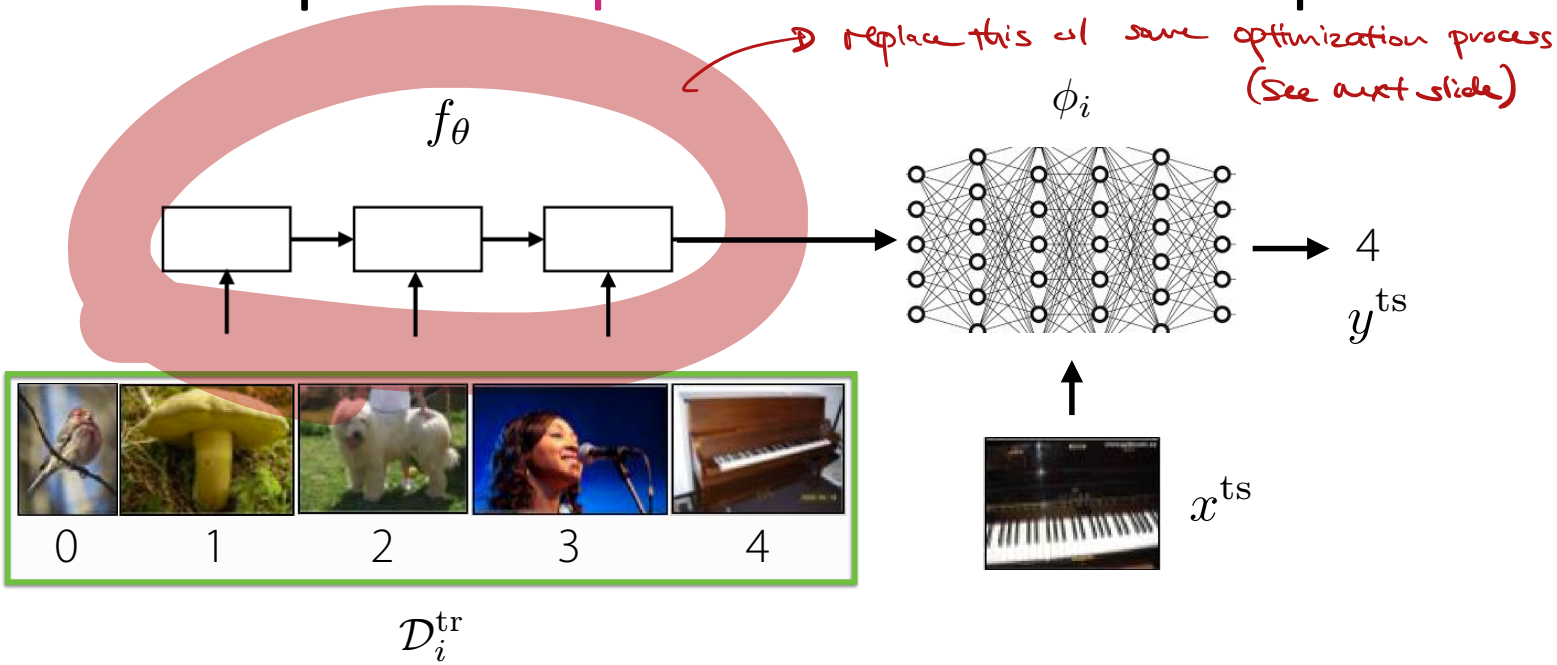
Plan for Today

Recap
- Meta-learning problem & black-box meta-learning

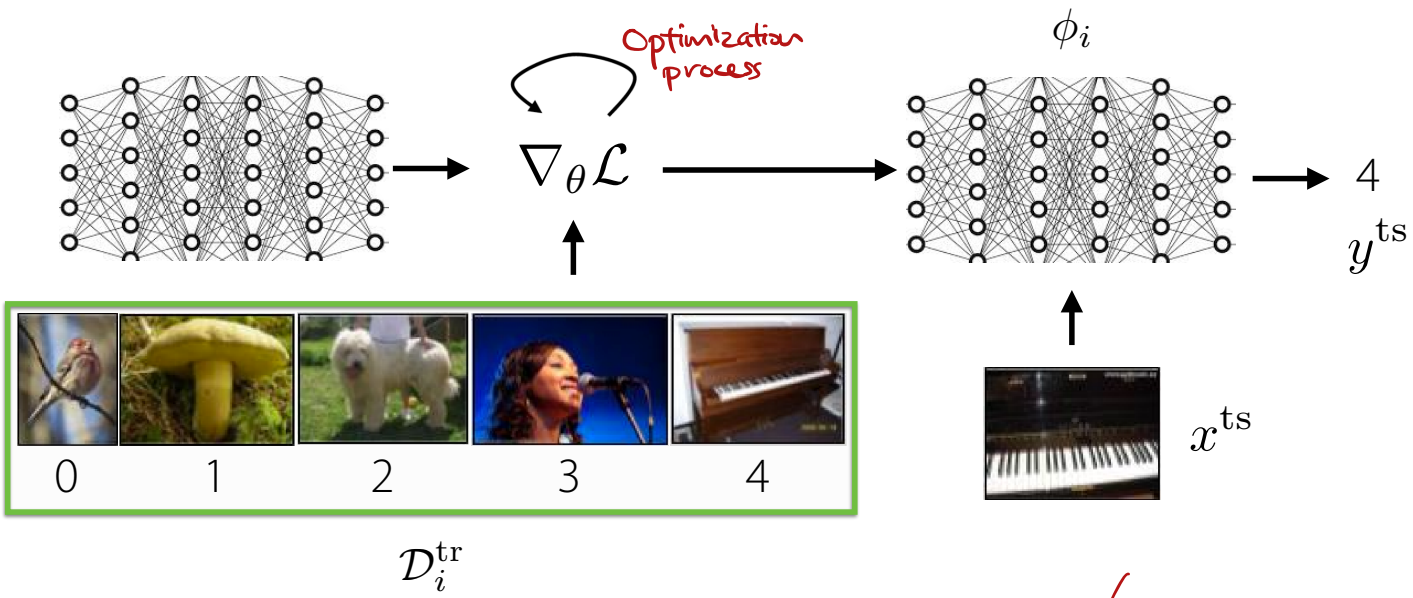
Optimization Meta-Learning
- Overall approach
- Compare: optimization-based vs. black-box
- Challenges & solutions
- Case study of land cover classification (time-permitting)

} Part of Homework 2!

~~Black-Box~~ ~~Adaptation~~ Optimization-Based Adaptation



~~Black-Box~~ ~~Adaptation~~ Optimization-Based Adaptation



outer / inner

Key idea: embed optimization inside the learning process → inner optim. process

→ personally, this actually makes more sense than using RNN/LSTM (prev. method)

Why might this make sense?

Recall: Fine-tuning

Fine-tuning

$$\phi \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\text{tr}})$$

pre-trained parameters

training data for new task

(typically for many gradient steps)

Universal Language Model Fine-Tuning for Text Classification. Howard, Ruder. '18

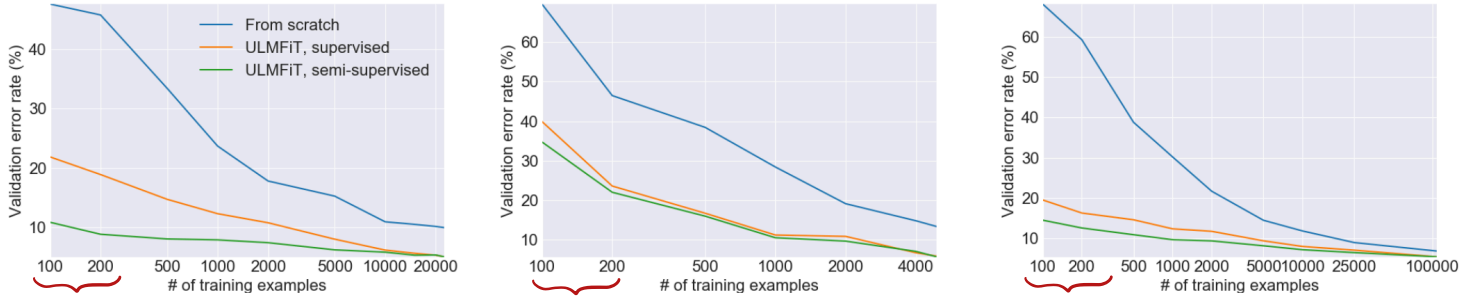


Figure 3: Validation error rates for supervised and semi-supervised ULMFiT vs. training from scratch with different numbers of training examples on IMDb, TREC-6, and AG (from left to right).

Fine-tuning less effective with **very small datasets**.

why people came up with meta-learning

Could look @ it this essay : [Fine Tuning] → [Meta Learning] (plugging in FT in ML)

Optimization-Based Adaptation

pre-trained parameters

Fine-tuning [test-time]

$$\phi \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\text{tr}})$$

training data for new task

Meta-learning

$$\min_\theta \sum_{\text{task } i} \mathcal{L}(\underbrace{\theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}})}_{\text{one step of gradient descent to get parameters for task } i}, \mathcal{D}_i^{\text{ts}})$$

evaluate how good those parameters (for task i) are for new examples in test dataset for the task.

Key idea: Over many tasks, learn parameter vector θ that transfers via fine-tuning

initial parameter vector

has two layers of loss functions!

Then optimize for this pre-trained parameters s.t. one step of gradient descent is generalizing well to new examples. → do this for all tasks

3-Way 1-Shot

0. Randomly initialize θ → "meta parameters"
1. Sample task (3 chars)
2. Sample 2 images/char →

→ output : meta params θ

X = [a] [b] [c]   [a] [b] [c]

y = 0 1 2   0 1 2

D^{tr} D^{ts}

3. Take one gradient descent step on the training dataset

$\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, D_i^{tr})$ ⊛ Key difference from the Black Box.

cf) For Black-Box approach, this was

3. $\phi_i \leftarrow f_\theta(D_i^{tr})$ → get the training data for a task, & output the model parameters (or the hidden state-equivalent)

⊛ f_θ

→ $\mathcal{L}_i(\theta_i, D_i^{tr}) = \sum_{x_i, y_i \in D_i^{tr}} \| \hat{y}_i^{tr} - y_i^{tr} \|^2$ → eg) for regression

4. Update θ

Take our test examples, & run it through our NN w/ params ϕ_i to get a corresponding prediction, & compare the prediction w/ the corresponding label. See how well this new network is generalizing to new examples, & back-propagate to our initial parameter vectors θ to update our meta-parameters.

→ $\hat{y}^{ts} = f_{\phi_i}(x^{ts})$

Q: How are θ & ϕ_i related to each other?

⇒ θ is a meta starting point, & ϕ_i is a set of params optimized from θ for a specific task.

⇒ Through the above process, we're trying to determine what would have been a better initialization that would allow us to better generalize w/ the given model.

Q: Can we use different $\mathcal{L}$ for different tasks?

⇒ Yes. Change $\mathcal{L}$ to $\mathcal{L}_i$ in the above equations.

Q: Last time we were taking gradients on the query set, but this time we're doing it for the support set. Correct?

⇒ In both Black-Box approach & the Optimization-based approach, we're updating the gradients on both the support set and the query set of the test dataset.

⊛ 요거 대시 짚고 넘어가야될듯

Q: Can we add a term for regularization?

⇒ Yes, but oftentimes for Step 3 ($\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}_i(\theta, D_i^{tr})$), we don't need a regularization (despite the tiny dataset used in the inner loop), because we're optimizing explicitly for generalization, it's optimizing for part of the optimization landscape that's well-behaved.

⊛ In step #3, we're taking a gradient w.r.t. θ, not ϕ_i

$\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, D_i^{tr})$

⇒ Treating ϕ_i more as activations than weights

Meta-Test

→ 여기 D_j^{ts} 아닝??

1. Given task j, D_j^{tr}
2. $\phi_j \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, D_j^{tr})$

$x \rightarrow f_{\phi_j}(x) \rightarrow \hat{y}$

→ "Literally just running fine-tuning"

⇒ What does she mean by this?

Optimization-Based Adaptation

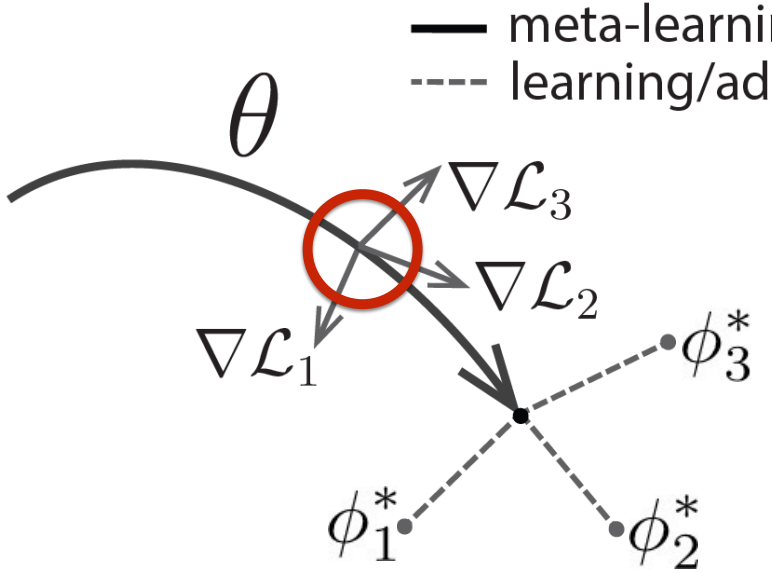
In practice, this LR could be much larger than your typical LR ∵ you want to traverse v. far in the param space

Q: Can you use hyperparam optimization to tune α?
A: Yes, but a lot of them are designed to optimize a v. low dimensional set of hyperparams. → won't scale well for millions of model params.
⇒ That's why we use end-to-end differentiation/gradient-based updating.

$$\min_{\theta} \sum_{\text{task } i} \mathcal{L}(\theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{ts}})$$

θ parameter vector being meta-learned

ϕ_i^* optimal parameter vector for task i



Caveat: there isn't just a single optimum for any given task ⇒ More of a region.

★ Key visualization for MAML!

Model-Agnostic Meta-Learning

Finn, Abbeel, Levine. Model-Agnostic Meta-Learning. ICML 2017

Optimization-Based Adaptation

Key idea: Acquire ϕ_i through optimization.

General Algorithm:

~~Black box approach~~ Optimization-based approach

1. Sample task $\mathcal{T}_i$ *(or mini batch of tasks)*
2. Sample disjoint datasets $\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{test}}$ from $\mathcal{D}_i$
3. ~~Compute $\phi_i \leftarrow f_\theta(\mathcal{D}_i^{\text{tr}})$~~ Optimize $\phi_i \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}})$
4. Update θ using $\nabla_\theta \mathcal{L}(\phi_i, \mathcal{D}_i^{\text{test}})$

—> brings up **second-order** derivatives

Both w.r.t. θ, not ϕ_i.

Do we need to compute the full Hessian? 😱

Bad bec. N-dim params → Hessian N^2 curse of dimensionality.

-> whiteboard

Do we get higher-order derivatives with more inner gradient steps?

Hessian-Vector Products

→ Can compute much more cheaply than computing Hessian itself.
↳ PyTorch uses this

→ Goal: Compute $H(x)v$

Function: f

f's gradient: g

$g(x+\underbrace{\Delta x}_{\text{replace}}) \cong g(x) + H(x)\Delta x$ → 1st order Taylor Expansion approximation

$g(x + rv) \cong g(x) + rH(x)v$

$H(x)v \cong \frac{g(x+rv) - g(x)}{r}$ → Can approximately compute Hessian-vector product w/ just two gradient evals
↳ Just a few more backward passes on the NN

Notation

$\frac{d}{dx}$: full deriv. $\nabla_{x'}\big|_{x'=x}$: partial deriv.

$$\phi_i = \theta - \alpha \frac{d}{d\theta}\mathcal{L}(\theta, D_i^{tr})$$

MAML objective (from here)

$$\min_{\theta} \mathcal{L}(\phi_i, D_i^{ts})$$

chain rule:

$$\frac{d}{d\theta}\mathcal{L}(\phi_i, D_i^{ts}) = \underbrace{\nabla_{\bar{\theta}}\mathcal{L}(\bar{\theta}, D_i^{ts})\big|_{\bar{\theta}=\phi_i}}_{\text{row vector } v} \underbrace{\frac{d\phi_i}{d\theta}}_{\text{matrix}}$$

$$\frac{d\phi_i}{d\theta} = I - \alpha \underbrace{\frac{d^2}{d\theta^2}\mathcal{L}(\theta, D_i^{tr})}_{\text{Hessian}}$$

$$= vI - \alpha vH$$

For multiple steps. (2-step case example)

$$\phi_i = \underbrace{\theta - \alpha \frac{d}{d\theta}\mathcal{L}(\theta, D_i^{tr})}_{\theta'} - \alpha \underbrace{\frac{d}{d\theta'}\mathcal{L}(\theta', D_i^{ts})}_{\text{this is w.r.t. } \theta', \text{ not } \theta}$$

$$\frac{d\phi_i}{d\theta} = I - \alpha \frac{d^2}{d\theta^2}\mathcal{L}(\theta, D_i^{tr}) - \underbrace{\alpha \nabla^2_{\bar{\theta}'}\mathcal{L}(\bar{\theta}', D_i^{ts})\big|_{\bar{\theta}'=\theta'} \frac{d\theta'}{d\theta}}_{\text{added term for 2nd chain}}$$

⇒ Products, no 3rd order deriv terms.

$$\frac{d}{d\theta}\mathcal{L}(\phi_i, \mathcal{D}_i^{\text{ts}})$$
$$= \nabla_{\bar{\phi}}\mathcal{L}(\bar{\phi}, \mathcal{D}_i^{\text{ts}})|_{\bar{\phi}=\phi_i} \frac{d\phi_i}{d\theta}$$
$$= \nabla_{\bar{\phi}}\mathcal{L}(\bar{\phi}, \mathcal{D}_i^{\text{ts}})|_{\bar{\phi}=\phi_i} \left(I - \alpha \frac{d^2}{d\theta^2}\mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}})\right)$$

PyTorch

Deep learning libraries handle the math for you.

See notes (2 pages back)

Optimization-Based Adaptation

Key idea: Acquire ϕ_i through optimization.

Meta-Test Time:

Optimization-based approach

1. Given task $\mathcal{T}_j$
2. Given training data $\mathcal{D}_j^{\text{tr}}$
3. Fine-tune $\phi_j \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_j^{\text{tr}})$
4. Make predictions on new datapoints $f_{\phi_j}(x)$

Plan for Today

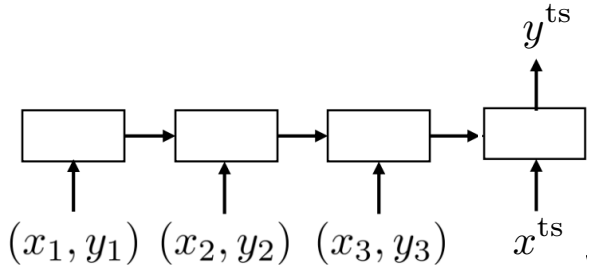
Recap
- Meta-learning problem & black-box meta-learning

Optimization Meta-Learning } Part of Homework 2!
- Overall approach
- **Compare: optimization-based vs. black-box**
- Challenges & solutions
- Case study of land cover classification (time-permitting)

Optimization vs. Black-Box Adaptation

Black-box adaptation

general form: $y^{\text{ts}} = f_{\text{black-box}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}})$



Model-agnostic meta-learning

$$y^{\text{ts}} = f_{\text{MAML}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}}) = f_{\phi_i}(x^{\text{ts}})$$

where $\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}})$

MAML can be viewed as **computation graph**, ⊛ interesting...

with embedded gradient operator

Note: Can mix & match components of computation graph

Learn initialization but replace gradient update with learned network

where $\phi_i = \theta - \alpha \cancel{\nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}})}$ $f(\theta, \mathcal{D}_i^{\text{tr}}, \nabla_\theta \mathcal{L})$ → Interesting, but in practice, MAML is preferred.

Ravi & Larochelle ICLR '17

(actually precedes MAML)

This **computation graph view** of meta-learning will come back again!

All trained on the original Omniglot dataset, but tested on "modified" datasets → warped.

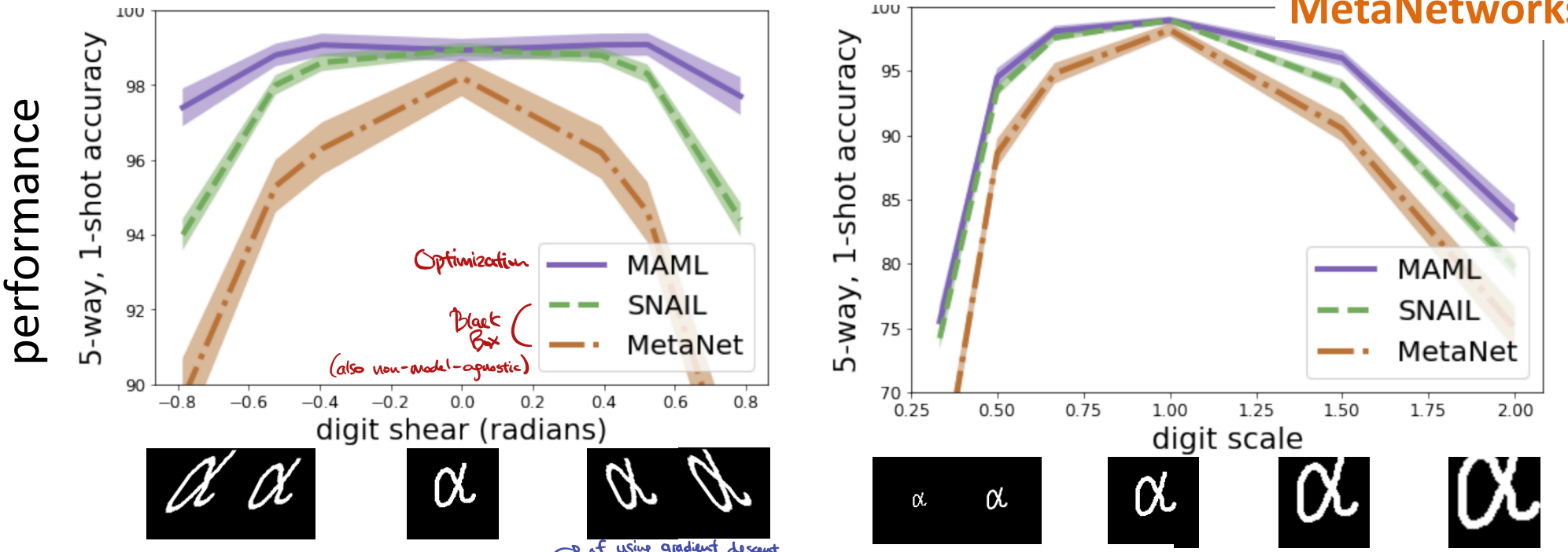
⇒ MAML generalizes better than other Black-Box meta learners

Optimization vs. Black-Box Adaptation

How well can learning procedures generalize to similar, but extrapolated tasks?

Omniglot image classification

MAML SNAIL, MetaNetworks



→ of using gradient descent

Does this structure come at a cost?

Finn & Levine ICLR '18

Black-box adaptation

$$y^{\text{ts}} = f_{\text{black-box}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}})$$

Optimization-based (MAML)

$$y^{\text{ts}} = f_{\text{MAML}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}})$$

Does this structure come at a cost?

(For a sufficiently deep network,) → Why is this the condition?

MAML function can approximate any function of $\mathcal{D}_i^{\text{tr}}, x^{\text{ts}}$ ✱

Finn & Levine, ICLR 2018

Assumptions:

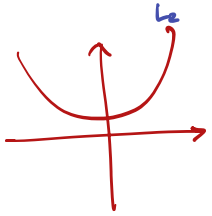
- nonzero α
- loss function gradient does not lose information about the label
- datapoints in $\mathcal{D}_i^{\text{tr}}$ are unique

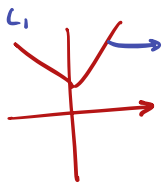
L_2, CrossEntropy ☺ L_1 ☹

✱ 아직 정확히?

If you have duplicates, or you want it to give you different ϕ_i's for diff. numbers of duplicates in the dataset, it's hard to understand how many duplicates there are in the dataset.

? two examples of loss functions





If you just look @ the gradient of this function, you can't tell where you are ⇒ No info

Why is this interesting?

MAML has benefit of inductive bias without losing expressive power.

set of assumptions a learning algorithm uses to predict outputs of given inputs that it has not encountered.

Plan for Today

Recap
- Meta-learning problem & black-box meta-learning

Optimization Meta-Learning } Part of Homework 2!
- Overall approach
- Compare: optimization-based vs. black-box
- **Challenges & solutions**
- Case study of land cover classification (time-permitting)

Optimization-Based Adaptation

Good set of references!

Challenges. Bi-level optimization can exhibit instabilities.

Ways to stabilize the process

Idea: Automatically learn inner vector learning rate, tune outer learning rate
(Li et al. Meta-SGD, Behl et al. AlphaMAML)

Idea: Optimize only a subset of the parameters in the inner loop
(Zhou et al. DEML, Zintgraf et al. CAVIA)

Idea: Decouple inner learning rate, BN statistics per-step (Antoniou et al. MAML++)

Idea: Introduce context variables for increased expressive power.
(Finn et al. bias transformation, Zintgraf et al. CAVIA)

Takeaway: a range of simple tricks that can help optimization significantly

Optimization-Based Adaptation

$$\frac{d\phi_i}{d\theta} = I - 2\frac{d^2}{d\theta^2}\kappa(\theta, D_i^{tr})$$

this is set to zero.

Challenges. Backpropagating through many inner gradient steps is compute- & memory-intensive.

Assuming the loss landscape of ϕ_i is v. similar to that of θ → unclear why this works. Maybe because they're both well-conditioned?

Idea: [Crudely] approximate $\frac{d\phi_i}{d\theta}$ as identity

Started off as a Tensorflow bug :)

(Finn et al. first-order MAML '17, Nichol et al. Reptile '18)

Surprisingly works for simple few-shot problems, but (anecdotally) not for more complex meta-learning problems.

Idea: Only optimize the *last layer* of weights.

→ much cheaper computationally.

→ There's another paper that says optimizing everything but the last layer yields good results. By Boyle (?) et al.

ridge regression, *logistic regression*

(Bertinetto et al. R2-D2 '19)

support vector machine

(Lee et al. MetaOptNet '19)

—> leads to a closed form or convex optimization on top of meta-learned features

Idea: Derive meta-gradient using the implicit function theorem

(Rajeswaran, Finn, Kakade, Levine. Implicit MAML '19)

—> compute full meta-gradient ***without differentiating through optimization path***

Optimization-Based Adaptation

Challenges. How to choose architecture that is effective for inner gradient step?

Idea: Progressive neural architecture search + MAML

(Kim et al. Auto-Meta)

- finds highly non-standard architecture (deep & narrow)
- different from architectures that work well for standard supervised learning

MiniImagenet, 5-way 5-shot MAML, basic architecture: **63.11%**

MAML + AutoMeta: **74.65%**

Optimization-Based Adaptation

Key idea: Acquire ϕ_i through optimization.

Takeaways: Construct *bi-level optimization* problem.

- \+ positive inductive bias at the start of meta-learning
- \+ tends to extrapolate better via structure of optimization
- \+ maximally expressive with sufficiently deep network
- \+ model-agnostic (easy to combine with your favorite architecture)
- \- typically requires second-order optimization
- \- usually compute and/or memory intensive

Before you do any meta-learning, you're already running gradient descent on your training dataset on the inner loop → already have a pretty good starting point.
cf) Black-Box starts off w/ a randomly initialized network.

→ expensive & challenging

Plan for Today

Recap
- Meta-learning problem & black-box meta-learning

Optimization Meta-Learning } Part of Homework 2!
- Overall approach
- Compare: optimization-based vs. black-box
- Challenges & solutions
- **Case study of land cover classification** (time-permitting)

Case Study

이거 구현해보면 재밌을듯.

Q: What was the core meta learner? MAML?

Meta-Learning for Few-Shot Land Cover Classification

Marc Rußwurm[1,*,†], Sherrie Wang[2,3,*], Marco Körner[1], and David Lobell[2]

[1]Technical University of Munich, Chair of Remote Sensing Technology
[2]Stanford University, Center on Food Security and the Environment
[3]Stanford University, Institute for Computational and Mathematical Engineering

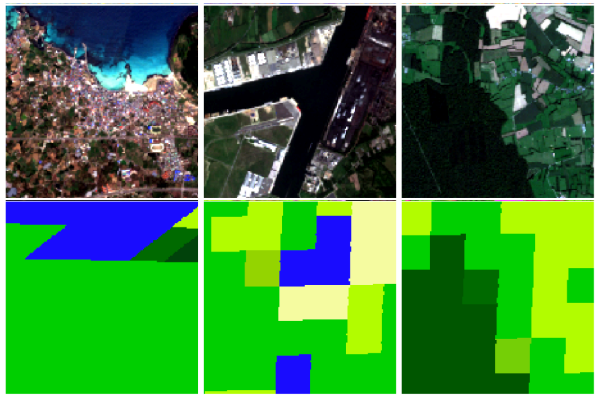
CVPR 2020 EarthVision Workshop

Link: https://arxiv.org/abs/2004.13390

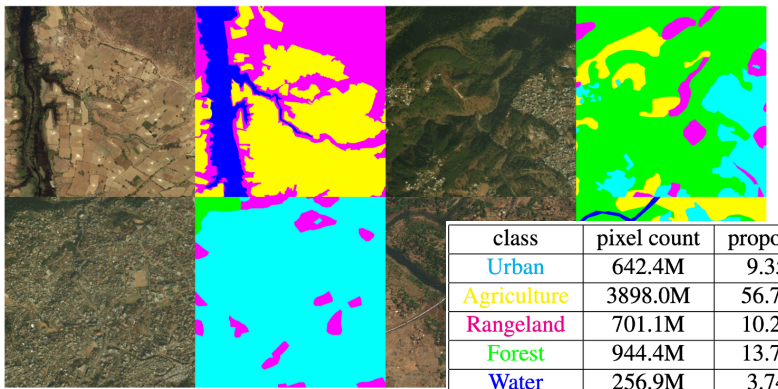
→ Predict how the land is being used.

Problem: Map land covering from satellite images

SEN12MS dataset
(Schmitt et al. 2019)



DeepGlobe dataset
(Demir et al. 2018)



class	pixel count	proportion
Urban	642.4M	9.35%
Agriculture	3898.0M	56.76%
Rangeland	701.1M	10.21%
Forest	944.4M	13.75%
Water	256.9M	3.74%
Barren	421.8M	6.14%
Unknown	3.0M	0.04%

Applications in global urban planning, climate change research

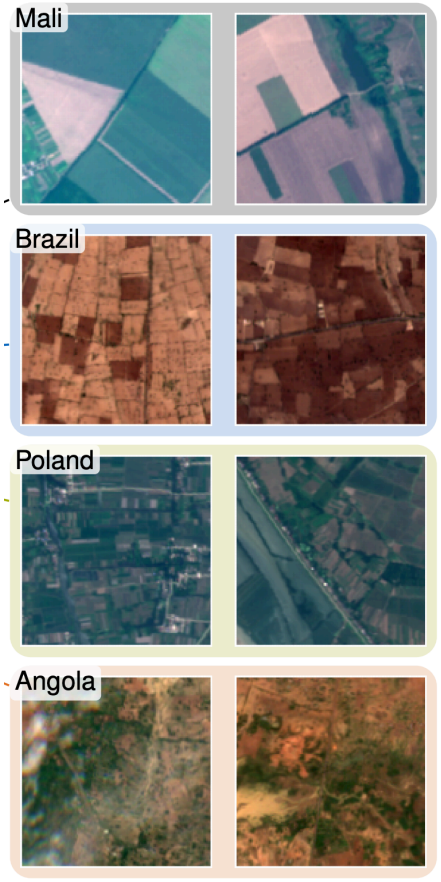
Challenges:

Labeling data is expensive. ☹

Different regions look different & have different land use proportions

↳ Transfer learning may not generalize well.

Framing land cover mapping as a meta-learning problem

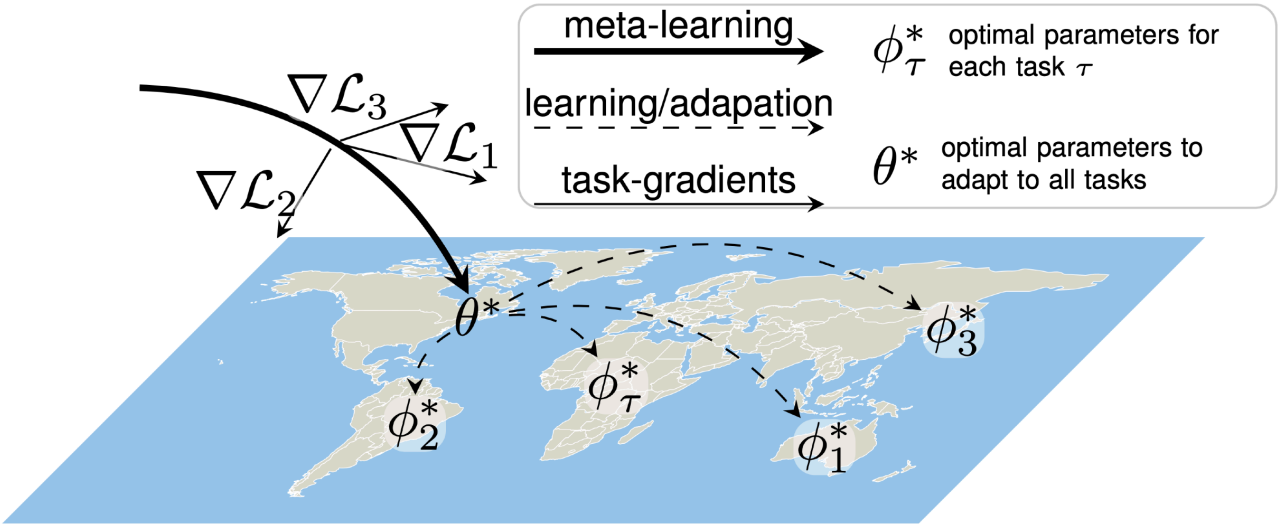


Croplands from four countries.

Different tasks: different regions of the world

The beauty of meta-learning

Goal: Segment/classify images from a new region with a small amount of data



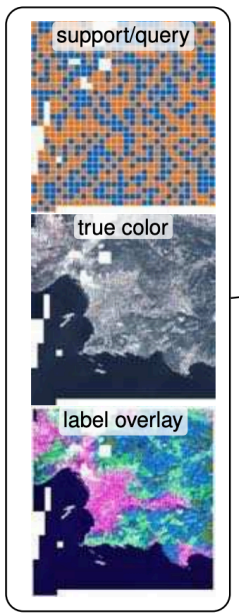
Framing land cover mapping as a meta-learning problem

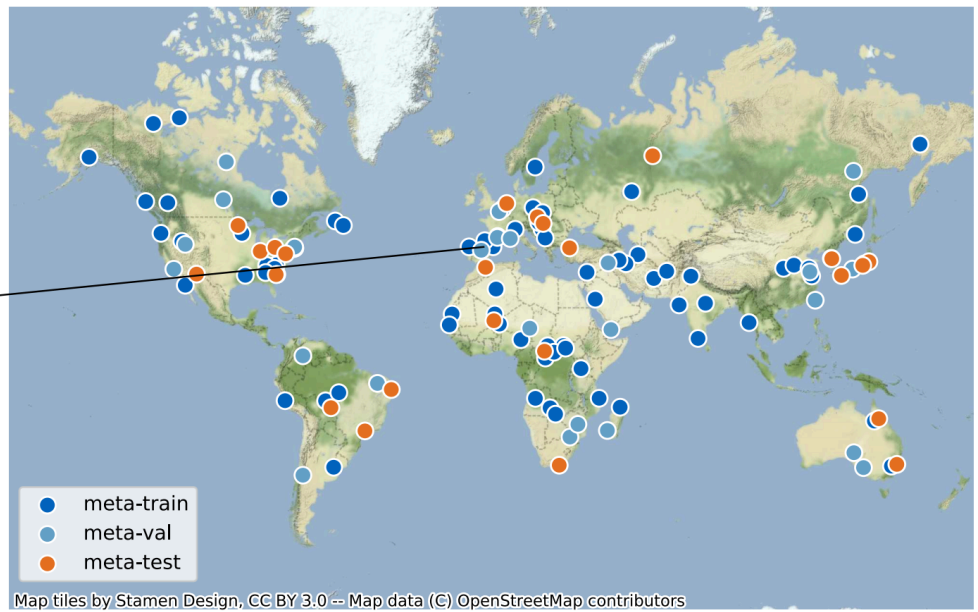
Goal: Segment/classify images from a new region with a small amount of data

SEN12MS dataset (Schmitt et al. 2019)

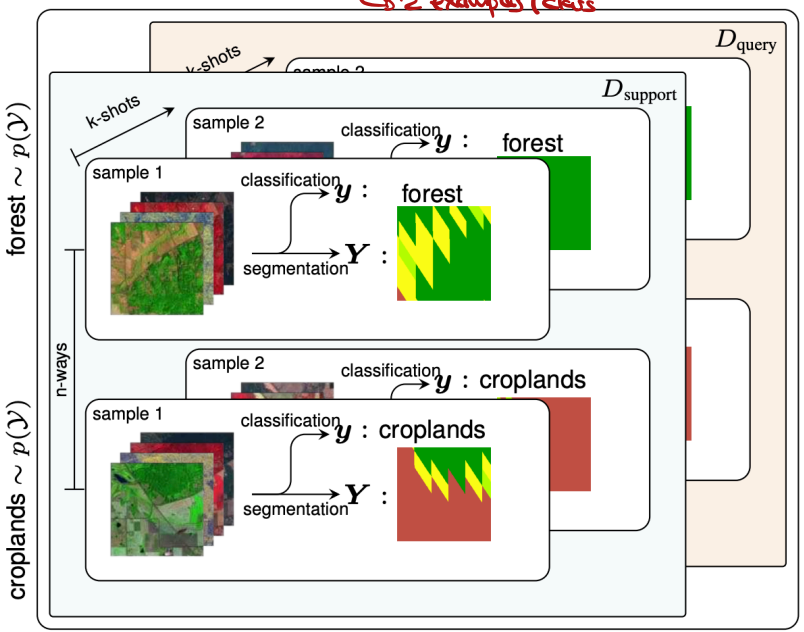
Geographic meta-data provided

Marseille (summer)





Example 2-way 2-shot classification task

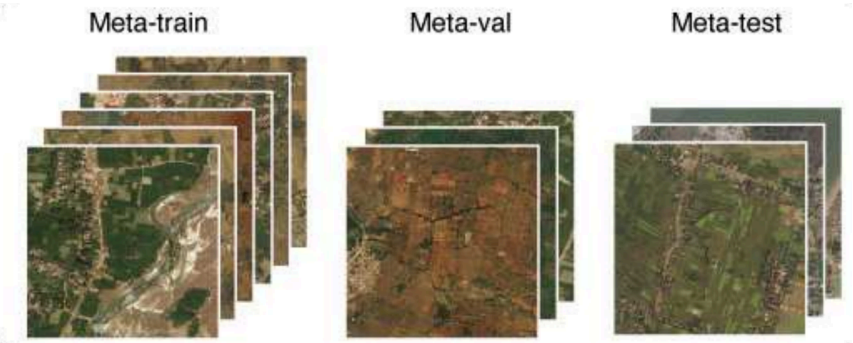


Framing land cover mapping as a meta-learning problem

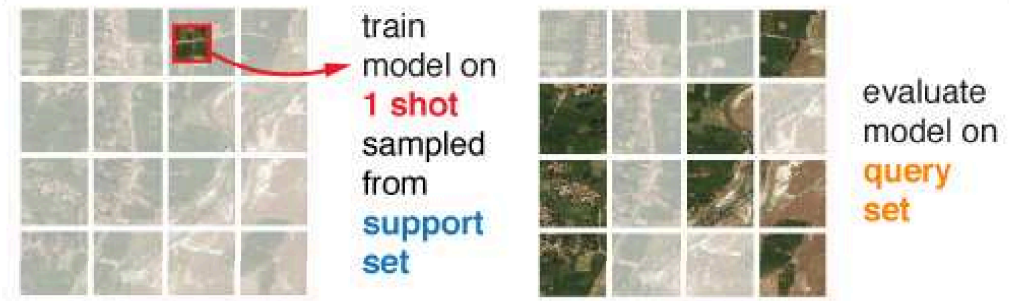
Goal: Segment/classify images from a new region with a small amount of data

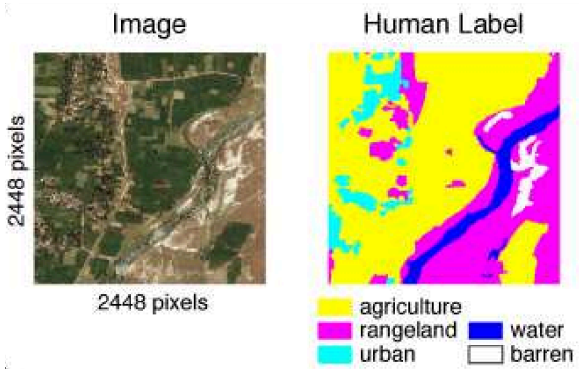
DeepGlobe dataset (Demir et al. 2018)

No geographic metadata, used clustering to guess region



Example 1-shot learning segmentation task.





Evaluation

Meta-training data: $\{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$

Meta-test time: small amount of data from new region: $\mathcal{D}_j^{\text{tr}}$
(meta-test training set / meta-test support set)

Compare:

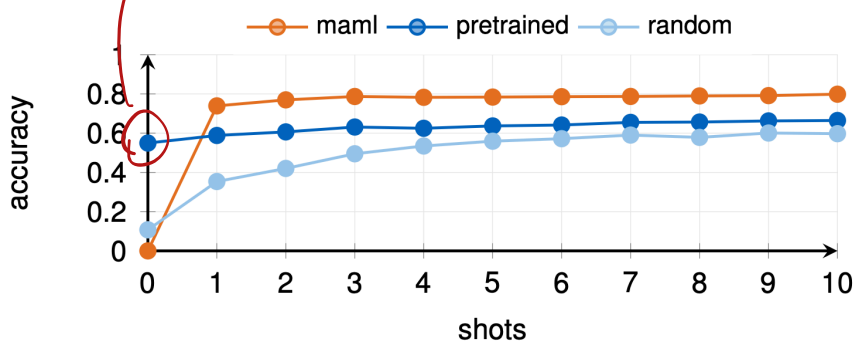
Random init: Train from scratch on $\mathcal{D}_j^{\text{tr}}$

Pre-train on meta-training data $\mathcal{D}_1 \cup \ldots \cup \mathcal{D}_T$, **fine-tune** on $\mathcal{D}_j^{\text{tr}}$

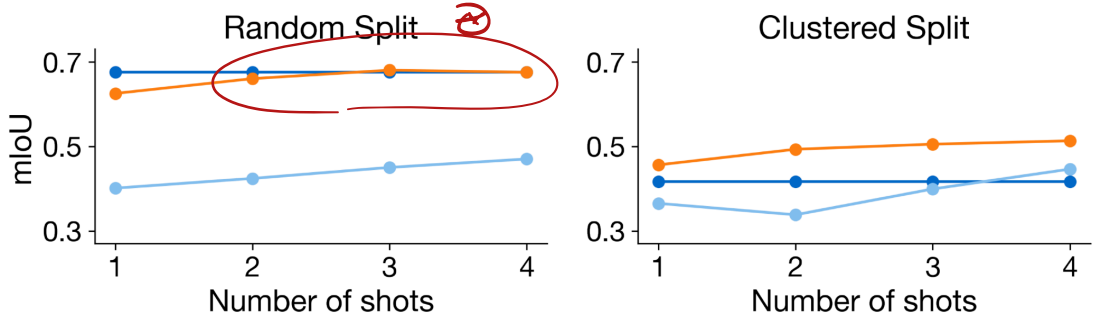
MAML on meta-training data $\{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$, adapt with $\mathcal{D}_j^{\text{tr}}$

Pre-train can do pretty decent elf even w/o fine-tuning

SEN12MS dataset



DeepGlobe dataset



(*) Makes sense: pre-training ~ target data are more similar

More visualizations and analysis in the paper!

Plan for Today

Recap
- Meta-learning problem & black-box meta-learning

Optimization Meta-Learning } **Part of Homework 2!**
- Overall approach
- Compare: optimization-based vs. black-box
- Challenges & solutions
- Case study of land cover classification (time-permitting)

Goals for by the end of lecture:
- Basics of optimization-based meta-learning techniques (& how to implement)
- Trade-offs between black-box and optimization-based meta-learning

Roadmap for upcoming lectures

Wednesday: Non-parametric few-shot learners, comparison of approaches

Next week: Unsupervised pre-training for few-shot learning

Following week: Advanced meta-learning topics (e.g. memorization, large-scale meta-optimization)

Course Reminders

Project group form due **tonight**.
(for assigning project mentors)

Homework 1 due **Wednesday**