Meta Learning

Vineeth N Balasubramanian
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Acknowledgements

- Slides adapted from ICML 2019 tutorial on Meta-Learning by Chelsea Finn and Sergey Levine
- PhD students, Arghya Pal and Joseph, for slides!

Motivation

- Today's success of deep learning -> supervised learning -> large amounts of labeled data
- What about applications/domains that do not have large datasets?
  - Medical imaging, Translation for rare languages, Personalized education, etc
- What if your data has a long tail?
- Continual learning?

Motivation

training data

Braque

Cezanne

test datapoint

By Braque or Cezanne?

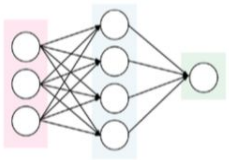
Contents

Introduction

- Previous experiences aid human learning.
- Can a machine emulate this:
  - Learn from previous experiences
  - Formulate a prior from those experiences
  - Or to put plainly, can we **learn to learn** from earlier experiences

Problem Formulation

We will formulate Meta-Learning from a Supervised Learning setting.



$$\arg\max_{\phi} \log p(\phi|\mathcal{D})$$

model parameters

training data

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$$

input (e.g., image)

label

$$= \arg\max_{\phi} \log p(\mathcal{D}|\phi) + \log p(\phi)$$

data likelihood

regularizer (e.g., weight decay)

$$= \arg\max_{\phi} \sum_i \log p(y_i|x_i, \phi) + \log p(\phi)$$

This would require large amount of labelled data to learn ϕ

Slides adapted from Meta-Learning tutorial by Chelsea Finn & Sergey Levine, presented at ICML 2019.

Problem Formulation

Can we incorporate additional data from related tasks?

If so, we can re-formulate the supervised learning problem to:

$$\arg\max_{\phi} \log p(\phi | \mathcal{D}, \mathcal{D}_{\text{meta-train}})$$

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$$

$$\mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$$

$$\mathcal{D}_i = \{(x_1^i, y_1^i), \ldots, (x_k^i, y_k^i)\}$$

Slides adapted from Meta-Learning tutorial by Chelsea Finn & Sergey Levine, presented at ICML 2019.

Problem Formulation

Lets learn a set of meta-parameters (θ), to model $\mathcal{D}_{\text{meta\_train}}$

$$meta\text{-}parameters\ \theta\text{:}\ p(\theta|\mathcal{D}_{\text{meta-train}})$$

Hence,

$$\log p(\phi|\mathcal{D}, \mathcal{D}_{\text{meta-train}}) = \log \int_{\Theta} p(\phi|\mathcal{D}, \theta) p(\theta|\mathcal{D}_{\text{meta-train}}) d\theta$$

$$\approx \log p(\phi|\mathcal{D}, \theta^\star) + \log p(\theta^\star|\mathcal{D}_{\text{meta-train}})$$

Where θ^*, is the MAP estimate:

$$\theta^\star = \arg\max_{\theta} \log p(\theta|\mathcal{D}_{\text{meta-train}})$$

This is the actual meta-learning problem.

Slides adapted from Meta-Learning tutorial by Chelsea Finn & Sergey Levine, presented at ICML 2019.

Problem Formulation

Assuming we have found θ^*, then the solution to the problem at hand ($\mathcal{D}$), would be:

$$\phi^\star = \arg\max_\phi \log p(\phi|\mathcal{D}, \mathcal{D}_{\text{meta-train}}) \approx \arg\max_\phi \log p(\phi|\mathcal{D}, \theta^\star)$$

This step of finding ϕ^* is referred to as ***Adaptation***.

Slides adapted from Meta-Learning tutorial by Chelsea Finn & Sergey Levine, presented at ICML 2019.

Problem Formulation

Hence, given a dataset and meta-dataset, this is how inference works.

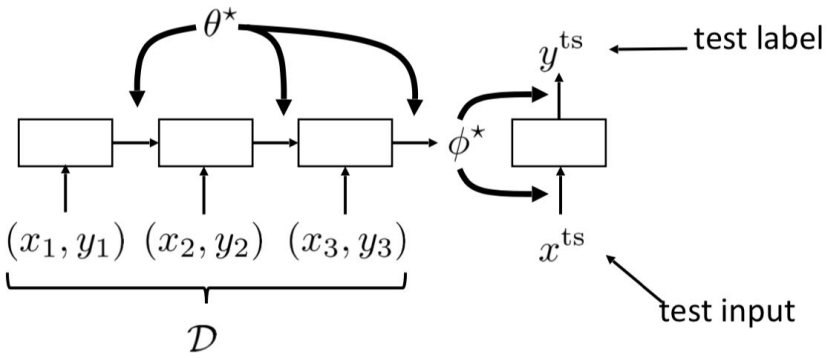
$$\text{meta-learning: } \theta^\star = \arg\max_\theta \log p(\theta|\mathcal{D}_{\text{meta-train}})$$

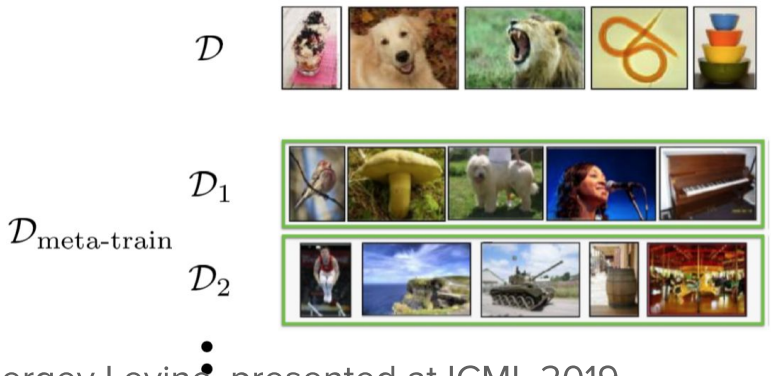
$$\text{adaptation: } \phi^\star = \arg\max_\phi \log p(\phi|\mathcal{D}, \theta^\star)$$

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$$

$$\mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$$

$$\mathcal{D}_i = \{(x_1^i, y_1^i), \ldots, (x_k^i, y_k^i)\}$$

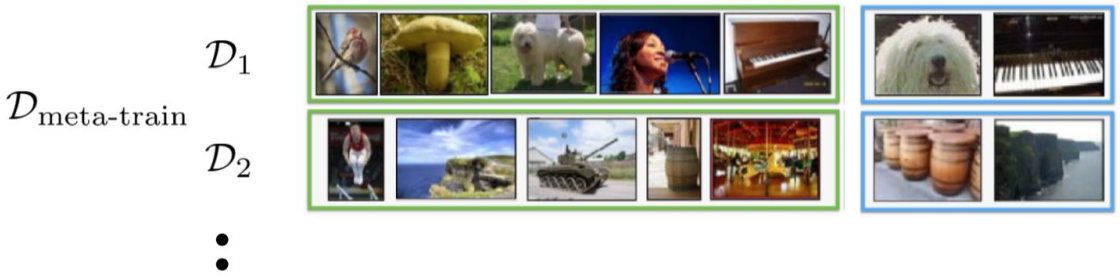




Slides adapted from Meta-Learning tutorial by Chelsea Finn & Sergey Levine, presented at ICML 2019.

How do we meta-learn? (finding θ^*)

Reserve a test-split for each $\mathcal{D}_{\text{meta\_train}}$

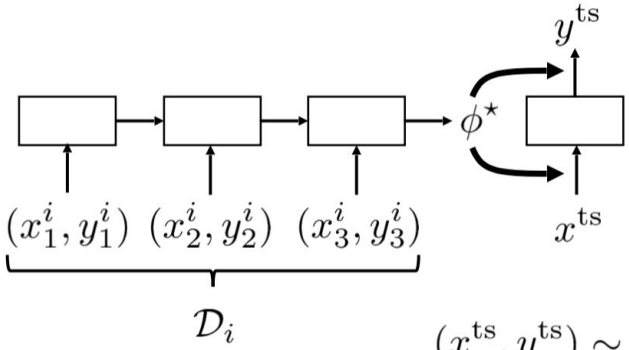


$$\mathcal{D}_{\text{meta-train}} = \{(\mathcal{D}_1^{\text{tr}}, \mathcal{D}_1^{\text{ts}}), \ldots, (\mathcal{D}_n^{\text{tr}}, \mathcal{D}_n^{\text{ts}})\}$$

$$\mathcal{D}_i^{\text{tr}} = \{(x_1^i, y_1^i), \ldots, (x_k^i, y_k^i)\}$$

$$\mathcal{D}_i^{\text{ts}} = \{(x_1^i, y_1^i), \ldots, (x_l^i, y_l^i)\}$$

Meta-training time



Slides adapted from Meta-Learning tutorial by Chelsea Finn & Sergey Levine, presented at ICML 2019.

How do we meta-learn? (finding θ^*)

$$\text{meta-learning: } \theta^\star = \arg\max_\theta \log p(\theta|\mathcal{D}_{\text{meta-train}})$$

$$\text{adaptation: } \phi^\star = \arg\max_\phi \log p(\phi|\mathcal{D}^{\text{tr}}, \theta^\star)$$

$$\updownarrow$$

$$\phi^\star = f_{\theta^\star}(\mathcal{D}^{\text{tr}})$$

$$\mathcal{D}_{\text{meta-train}} = \{(\mathcal{D}_1^{\text{tr}}, \mathcal{D}_1^{\text{ts}}), \ldots, (\mathcal{D}_n^{\text{tr}}, \mathcal{D}_n^{\text{ts}})\}$$

$$\mathcal{D}_i^{\text{tr}} = \{(x_1^i, y_1^i), \ldots, (x_k^i, y_k^i)\}$$

$$\mathcal{D}_i^{\text{ts}} = \{(x_1^i, y_1^i), \ldots, (x_l^i, y_l^i)\}$$

learn θ such that $\phi = f_\theta(\mathcal{D}_i^{\text{tr}})$ is good for $\mathcal{D}_i^{\text{ts}}$

$$\theta^\star = \max_\theta \sum_{i=1}^{n} \log p(\phi_i|\mathcal{D}_i^{\text{ts}})$$

$$\text{where } \phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$$

Slides adapted from Meta-Learning tutorial by Chelsea Finn & Sergey Levine, presented at ICML 2019.

Meta Learning Algorithms

General Principle:

1. Choose a form of $p(\phi_i|\mathcal{D}_i^{\text{tr}}, \theta)$
2. Choose how to optimise θ, using $\mathcal{D}_{\text{meta\_train}}$

Meta Learning Algorithms

- Usually **evaluated** as m-way, k-shot classification tasks.
- Popular datasets:
  - Omniglot (1623 classes, with 20 instances per class)
  - MiniImagenet (100 classes, with 600 examples pre class)
  - Modified splits of CIFAR, CUB and CelebA

Meta Learning Algorithms

- Types
  - Black-box adaptation
    - Santoro et al. MANN, Mishra et al. SNAIL
  - Optimization based methods
    - Finn et al., MAML, Nichol et al. Reptile, Zintgraf et al. CAVIA
  - Non-parametric methods
    - Vinyals et al. Matching Networks, Snell et al. Prototypical Networks, Sung et al. Relation Net
  - Bayesian meta-learning

Credit: Slides (14 - 52) are fully or partially adapted from Meta-Learning tutorial by Chelsea Finn & Sergey Levine, presented at ICML 2019.

How to Design a Meta Learning Algorithms

1. Choose a form of $p(\phi_i|\mathcal{D}_i^{\text{tr}}, \theta)$

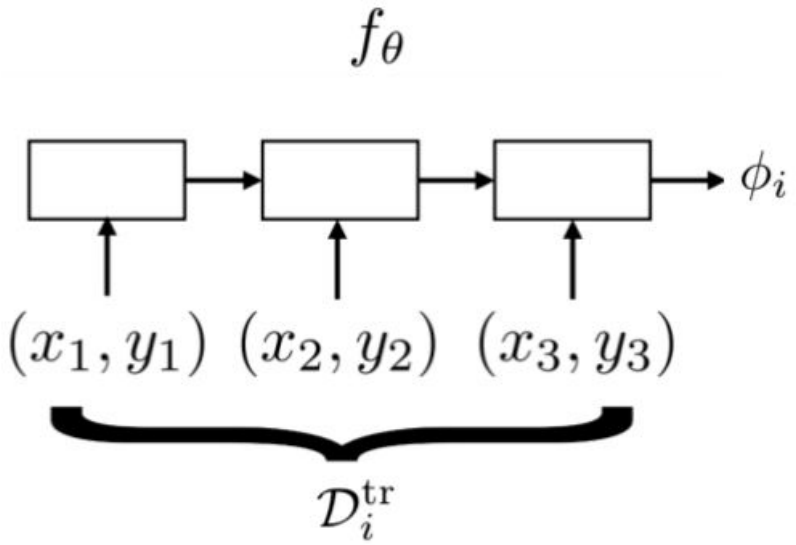
2. Choose how to optimize θ w.r.t. max-likelihood objective using $\mathcal{D}_{\text{meta-train}}$

Can we treat $p(\phi_i|\mathcal{D}_i^{\text{tr}}, \theta)$ as an **inference** problem?

Neural networks are good at inference.

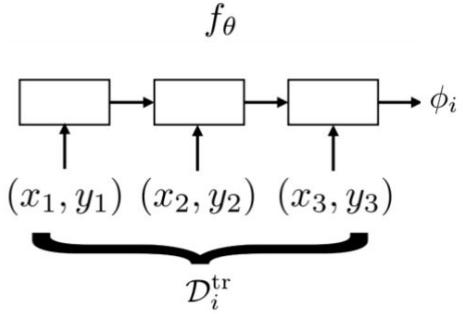
Black box adaptation

Key idea: Train a neural network to represent $p(\phi_i | \mathcal{D}_i^{\text{tr}}, \theta)$

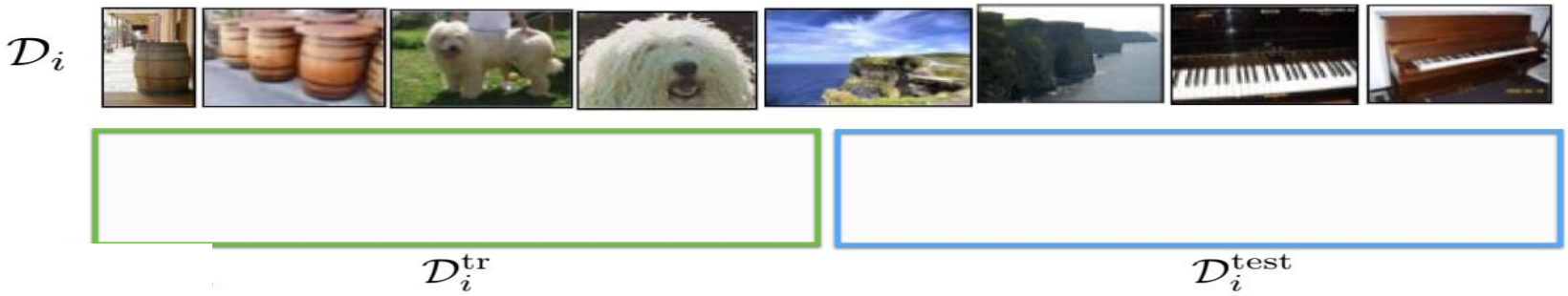


Black box adaptation

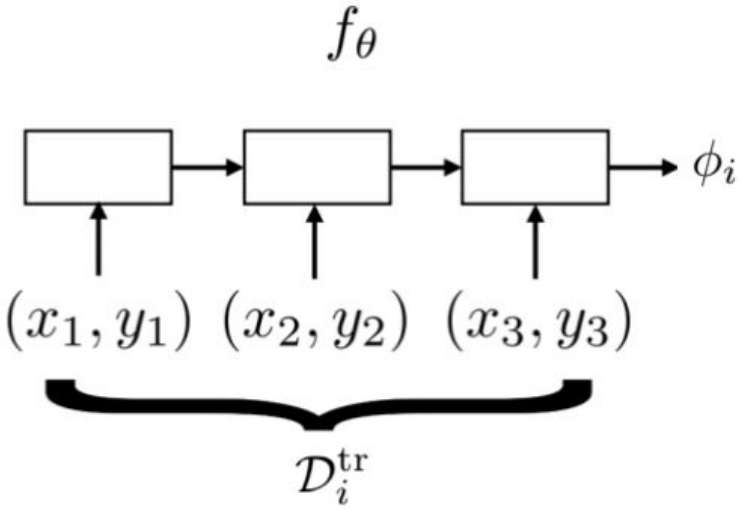
Key idea: Train a neural network to represent $p(\phi_i | \mathcal{D}_i^{\text{tr}}, \theta)$



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 $\mathcal{D}_i$



Black box adaptation



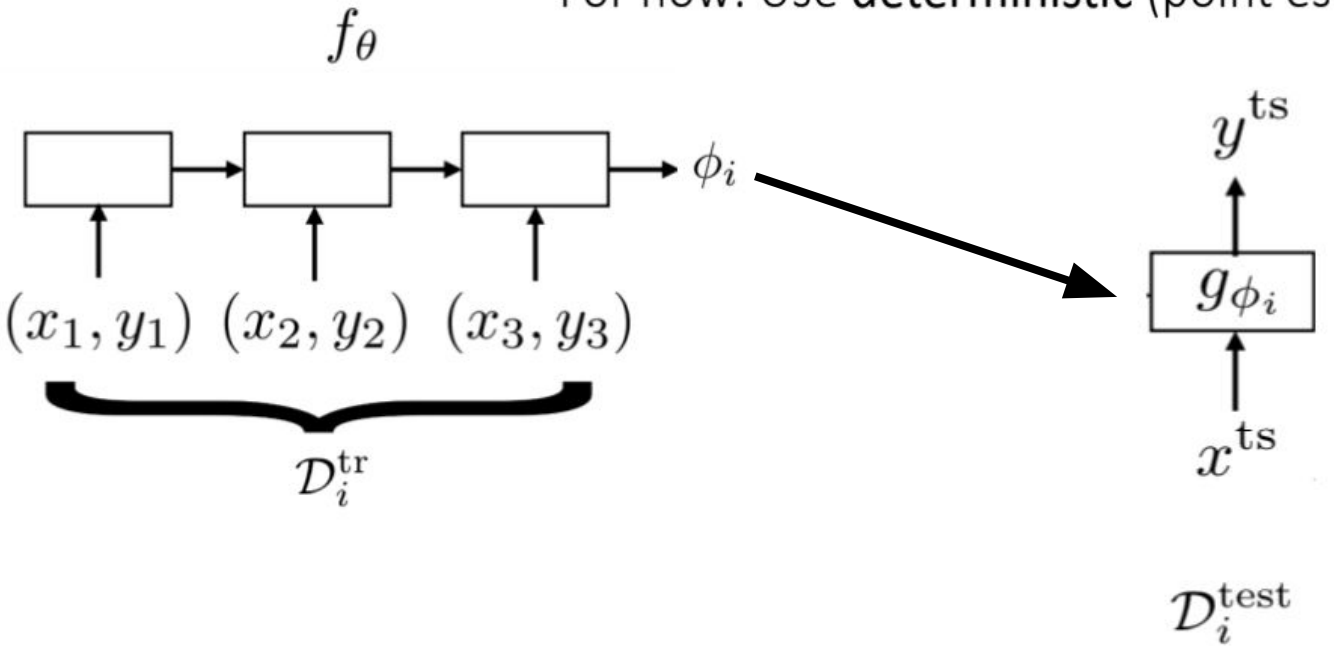
Train with standard supervised learning!

$$\max_\theta \sum_{\mathcal{T}_i} \underbrace{\sum_{(x,y)\sim\mathcal{D}_i^{\text{test}}} \log g_{\phi_i}(y|x)}_{\mathcal{L}(\phi_i, \mathcal{D}_i^{\text{test}})}$$

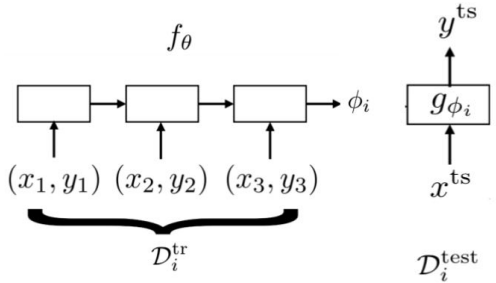
$$\max_\theta \sum_{\mathcal{T}_i} \mathcal{L}(f_\theta(\mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{test}})$$

Black box adaptation

For now: Use **deterministic** (point estimate) $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$

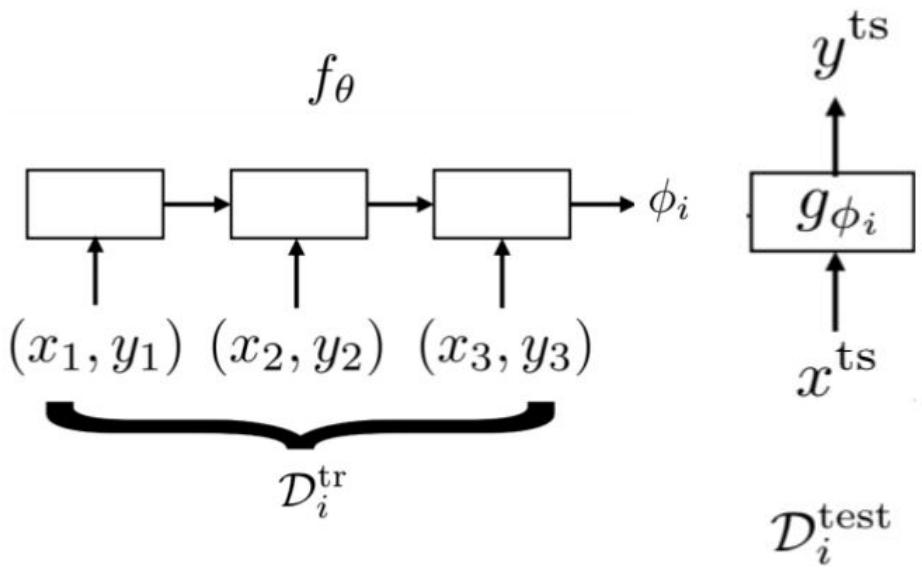


Black box adaptation



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 $\phi_i \leftarrow f_\theta(\mathcal{D}_i^{\text{tr}})$
4. Update θ using $\nabla_\theta \mathcal{L}(\phi_i, \mathcal{D}_i^{\text{test}})$

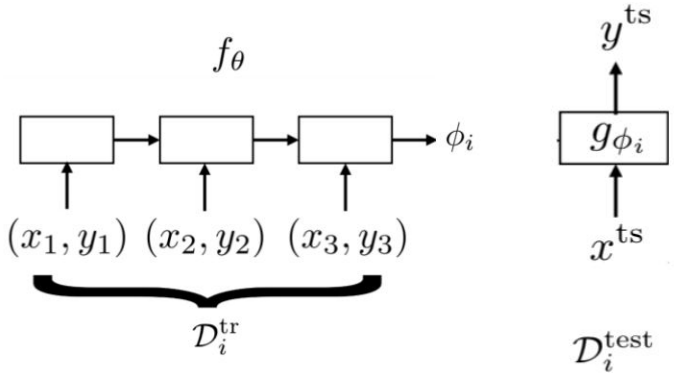
Black box adaptation



Form of f_θ ?

- LSTM
- Neural turing machine (NTM)
- Self-attention
- 1D convolutions
- feedforward + average

Black box adaptation



Is this scalable?

Outputting all neural net parameters:

- MANN (ICLR 2016)
- HyperNetwork (ICLR 2017)
- Introspection (ICLR 2017)
- SNAIL (ICLR 2018)
- Zero Shot Task Transfer (CVPR 2019)

Meta Learning Algorithms

- Types
  - Black-box adaptation
    - Santoro et al. MANN, Mishra et al. SNAIL
  - Optimization based methods
    - Finn et al., MAML, Nichol et al. Reptile, Zintgraf et al. CAVIA
  - Non-parametric methods
    - Vinyals et al. Matching Networks, Snell et al. Prototypical Networks, Sung et al. Relation Net
  - Bayesian meta-learning

Credit: Slides (14 - 52) are fully or partially adapted from Meta-Learning tutorial by Chelsea Finn & Sergey Levine, presented at ICML 2019.

Optimization Based Methods

Key idea: Acquire ϕ_i through optimization.

$$\max_{\phi_i} \log p(\mathcal{D}_i^{\text{tr}}|\phi_i) + \log p(\phi_i|\theta)$$

Meta-parameters θ serve as a prior. What form of prior?

One successful form of prior knowledge: **initialization** for **fine-tuning**

Optimization Based Methods

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

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

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

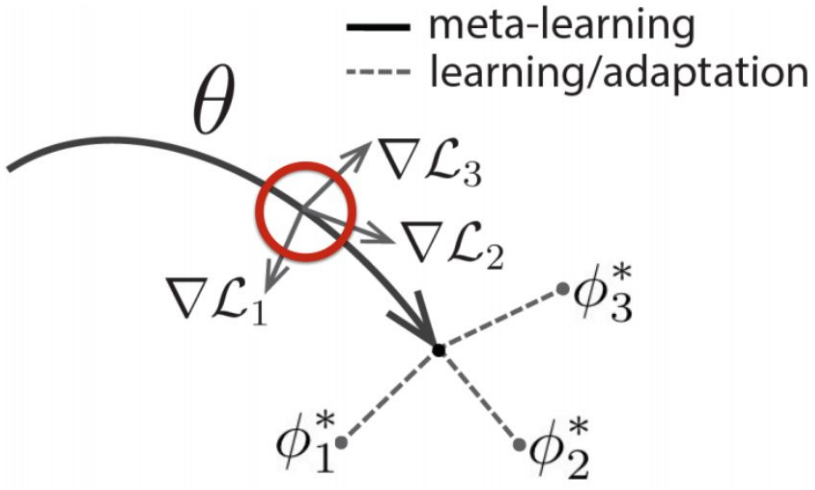
Reference: Model Agnostic Meta Learning, Finn *et al*

Optimization Based Methods

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

θ parameter vector being meta-learned

ϕ_i^* optimal parameter vector for task i



Reference: Model Agnostic Meta Learning, Finn *et al*

Optimization Based Methods

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7: **end for**
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9: **end while**

Reference: Model Agnostic Meta Learning, Finn *et al*

Optimization Based Methods

Key idea: Acquire ϕ_i through optimization.

General Algorithm:

~~Amortized 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 $\nabla_\theta \mathcal{L}(\phi_i, \mathcal{D}_i^{\text{test}})$

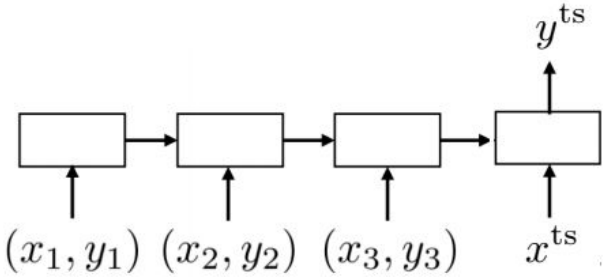
—> brings up **second-order** derivatives (more on this later)

Reference: Model Agnostic Meta Learning, Finn *et al*

Optimization vs. Black Box Adaptation

Black-box adaptation

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



Model-agnostic meta-learning

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

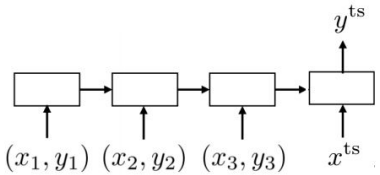
$$\text{where } \phi_i = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}})$$

MAML can be viewed as **computation graph**, with embedded gradient operator

Optimization vs. Black Box Adaptation

Black-box adaptation

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



Model-agnostic meta-learning

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

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

MAML can be viewed as **computation graph**, with embedded gradient operator

Note: Can mix & match components of computation graph

Learn initialization but replace gradient update with learned network

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

Ravi & Larochelle ICLR '17

(actually precedes MAML)

Optimization vs. Black Box Adaptation

Black-box adaptation

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

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 f,

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

Finn & Levine, ICLR 2018

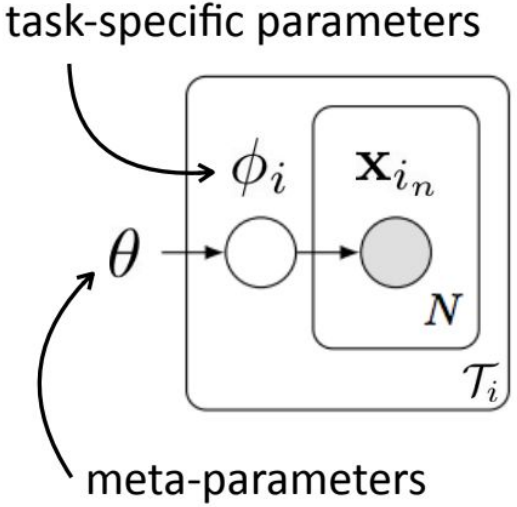
Assumptions:

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

Why is this interesting?

MAML has benefit of inductive bias without losing expressive power.

Probabilistic Interpretation of Optimization Based Inference



Key idea: Acquire ϕ_i through optimization.

Meta-parameters θ serve as a prior.

One form of prior knowledge: **initialization** for **fine-tuning**

Optimization Based Methods

Gradient-descent + early stopping (MAML): implicit Gaussian prior $\phi \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\text{tr}})$

Other forms of priors?

Gradient-descent with **explicit Gaussian prior** $\phi \leftarrow \min_{\phi'} \mathcal{L}(\phi', \mathcal{D}^{\text{tr}}) + \frac{\lambda}{2}||\theta - \phi'||^2$

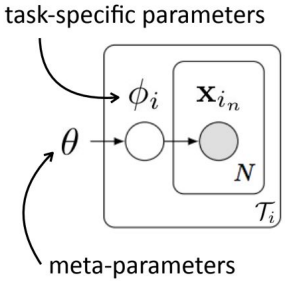
Rajeswaran et al. implicit MAML '19

Bayesian linear regression *on learned features* Harrison et al. ALPaCA '18

Closed-form *or* **convex optimization** *on learned features*

ridge regression, *logistic regression*
Bertinetto et al. R2-D2 '19

support vector machine
Lee et al. MetaOptNet '19



Optimization Based Methods

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 Methods

Challenges

Second-order meta-optimization can exhibit instabilities.

Idea: [Crudely] approximate $\frac{d\phi_i}{d\theta}$ as identity
(Finn et al. first-order MAML, Nichol et al. Reptile)

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)

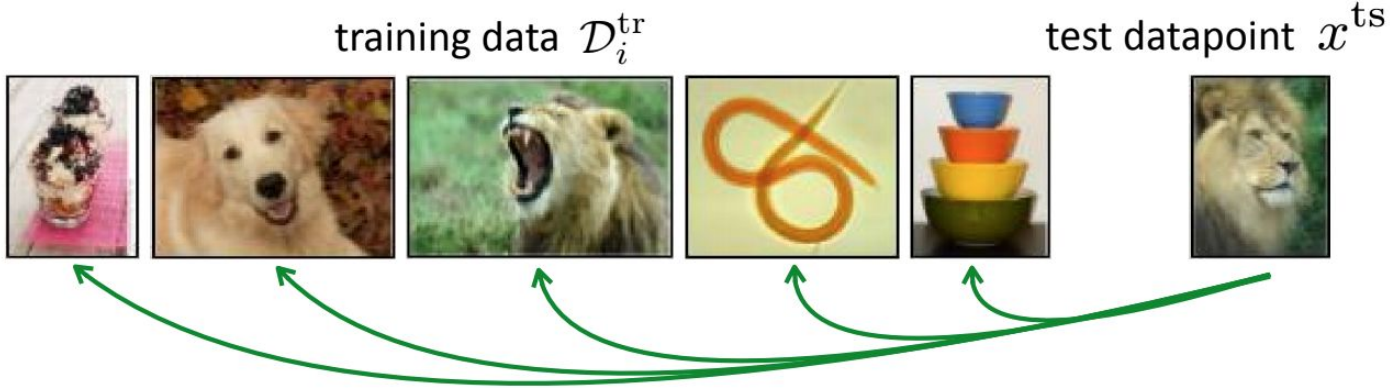
Meta Learning Algorithms

- Types
  - Black-box adaptation
    - Santoro et al. MANN, Mishra et al. SNAIL
  - Optimization based methods
    - Finn et al., MAML, Nichol et al. Reptile, Zintgraf et al. CAVIA
  - Non-parametric methods
    - Vinyals et al. Matching Networks, Snell et al. Prototypical Networks, Sung et al. Relation Net
  - Bayesian meta-learning

Credit: Slides (14 - 52) are fully or partially adapted from Meta-Learning tutorial by Chelsea Finn & Sergey Levine, presented at ICML 2019.

Non Parametric Methods

Key Idea: Use non-parametric learner.



Compare test image with training images

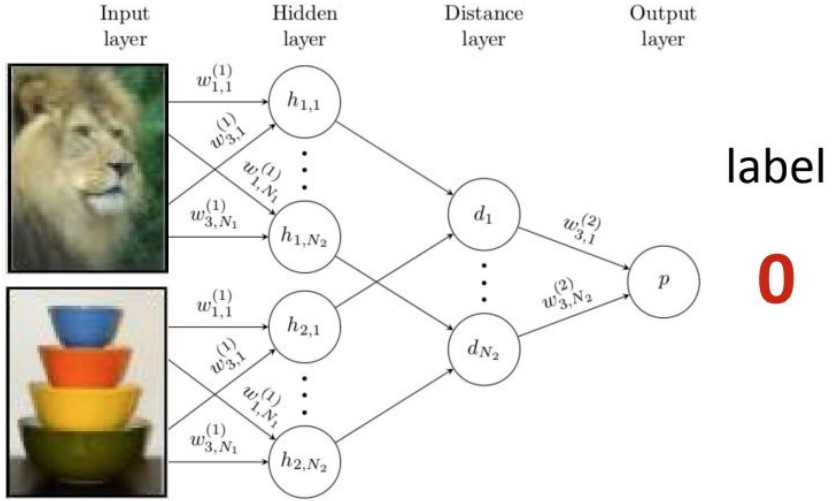
In what space do you compare? With what distance metric?

~~pixel space, l_2 distance?~~

Non Parametric Methods

Learn to compare using data!

train Siamese network to predict whether or not two images are the same class



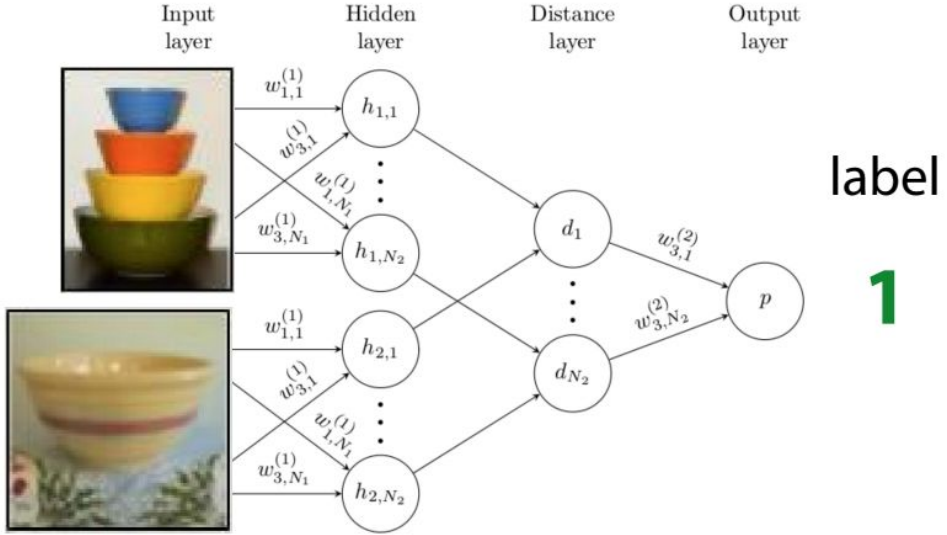
label

0

Non Parametric Methods

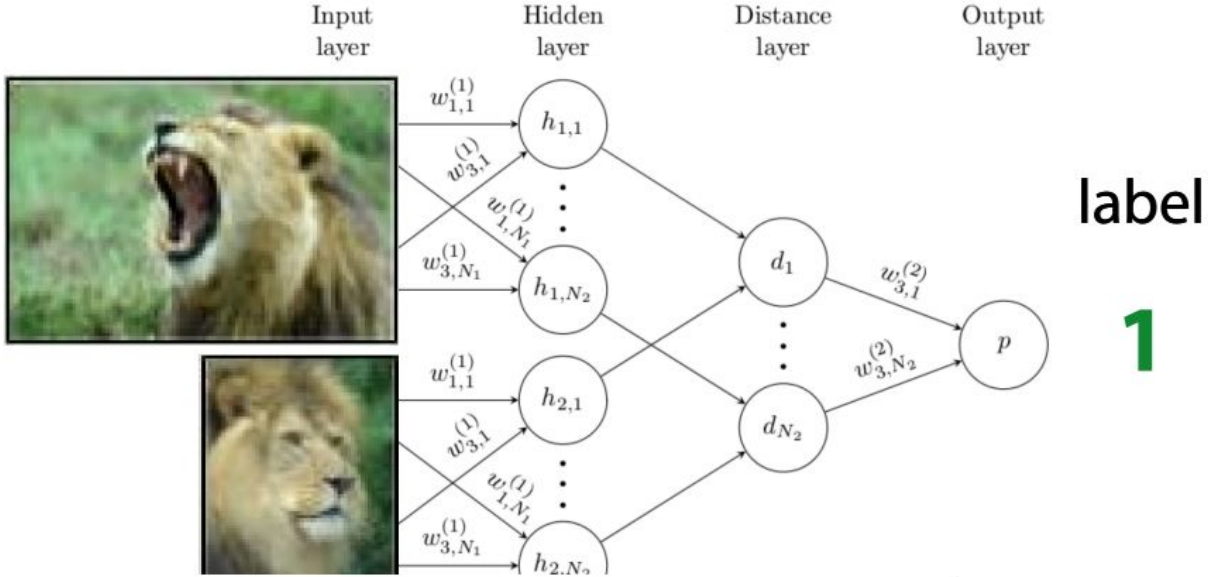
Learn to compare using data!

train Siamese network to predict whether or not two images are the same class



Non Parametric Methods

Learn to compare using data!



Meta-test time: compare image $\mathbf{x}_{\text{test}}$ to each image in $\mathcal{D}_j^{\text{tr}}$

Non Parametric Methods

Learn to compare using data!

Meta-test time: compare image $\mathbf{x}_{\text{test}}$ to each image in $\mathcal{D}_j^{\text{tr}}$

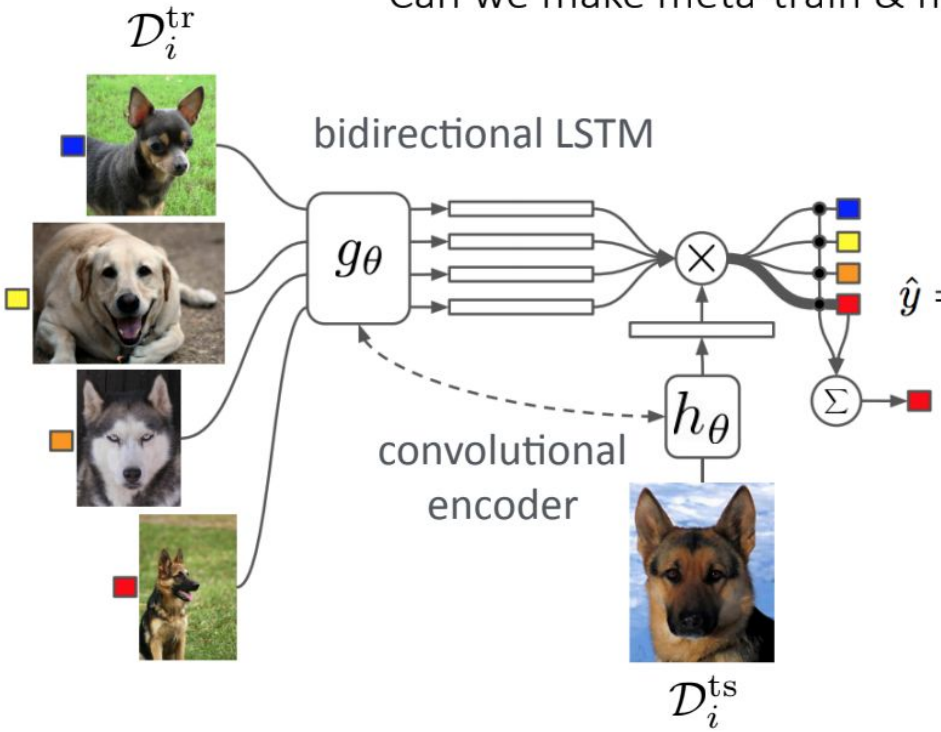
Meta-training: 2-way classification

Meta-test: N-way classification

Can we **match** meta-train & meta-test?

Non Parametric Methods

Can we make meta-train & meta-test match?



Weighed nearest neighbors in learned embedding space

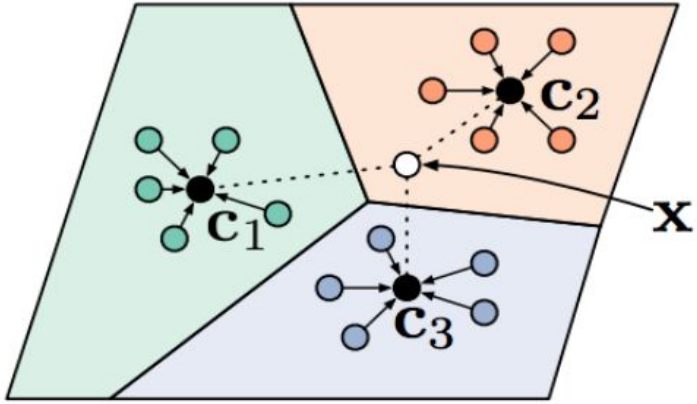
$$\hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i$$

What if >1 shot?

Can we **aggregate class information** to create a **prototypical embedding**?

Vinyals et al. Matching Networks, NeurIPS '16

Non Parametric Methods



(a) Few-shot

$$\mathbf{c}_k = \frac{1}{|\mathcal{D}_i^{\mathrm{tr}}|} \sum_{(x,y)\in\mathcal{D}_i^{\mathrm{tr}}} f_\theta(x)$$

$$p_\theta(y=k|x) = \frac{\exp(-d(f_\theta(x), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_\theta(x), \mathbf{c}_{k'}))}$$

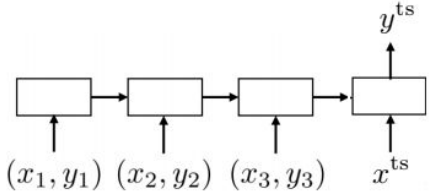
d: Euclidean, or cosine distance

Prototypical Networks, Snell et al, NeurIPS 2017

Comparison

Black-box amortized

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



Optimization-based

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

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

Non-parametric

$$y^{\text{ts}} = f_{\text{PN}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}}) = \text{softmax}\left(-d(f_\theta(x), c_k)\right)$$

$$\text{where } c_k = \frac{1}{|\mathcal{D}_i^{\text{tr}}|} \sum_{(x,y) \in \mathcal{D}_i^{\text{tr}}} f_\theta(x)$$

Non Parametric Methods

So far: Siamese networks, matching networks, prototypical networks

Embed, then nearest neighbors.

Challenge

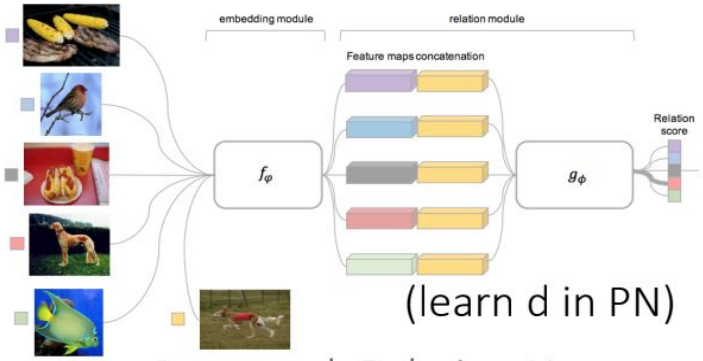
What if you need to reason about more complex relationships between datapoints?

Non Parametric Methods

Challenge

What if you need to reason about more complex relationships between datapoints?

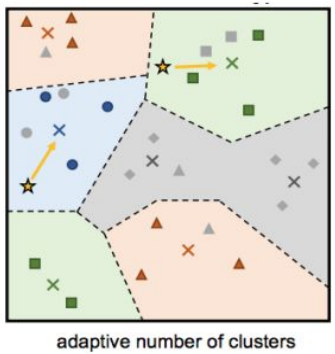
Idea: Learn non-linear relation module on embeddings



(learn d in PN)

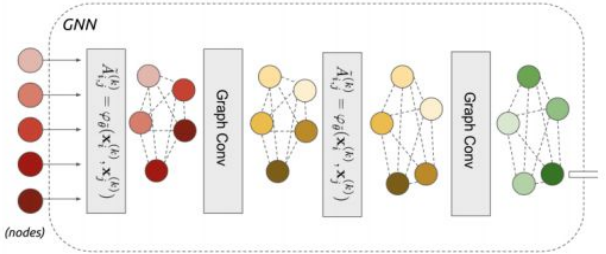
Sung et al. Relation Net

Idea: Learn infinite mixture of prototypes.



Allen et al. IMP, ICML '19

Idea: Perform message passing on embeddings



Garcia & Bruna, GNN

Takeaway

Black-box amortized

+ easy to combine with **variety of learning problems** (e.g. SL, RL)

- **challenging optimization** (no inductive bias at the initialization)
- often **data-inefficient**
- **model** & **architecture** intertwined

Optimization-based

+ handles **varying** & **large K** well
+ **structure lends well** to **out-of-distribution tasks**

- **second-order optimization**

Non-parametric

+ **simple**
+ entirely **feedforward**
+ **computationally fast** & **easy to optimize**

- **harder to generalize** to **varying K**
- hard to scale to **very large K**
- so far, **limited to classification**

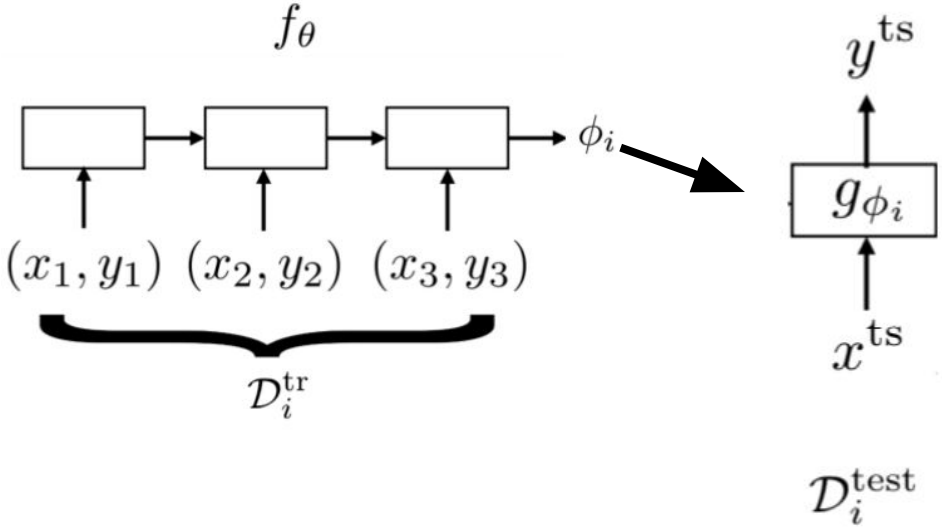
Meta Learning Algorithms

- Types
  - Black-box adaptation
    - Santoro et al. MANN, Mishra et al. SNAIL
  - Optimization based methods
    - Finn et al., MAML, Nichol et al. Reptile, Zintgraf et al. CAVIA
  - Non-parametric methods
    - Vinyals et al. Matching Networks, Snell et al. Prototypical Networks, Sung et al. Relation Net
  - Bayesian meta-learning

Credit: Slides (14 - 52) are fully or partially adapted from Meta-Learning tutorial by Chelsea Finn & Sergey Levine, presented at ICML 2019.

Recall: Black box adaptation

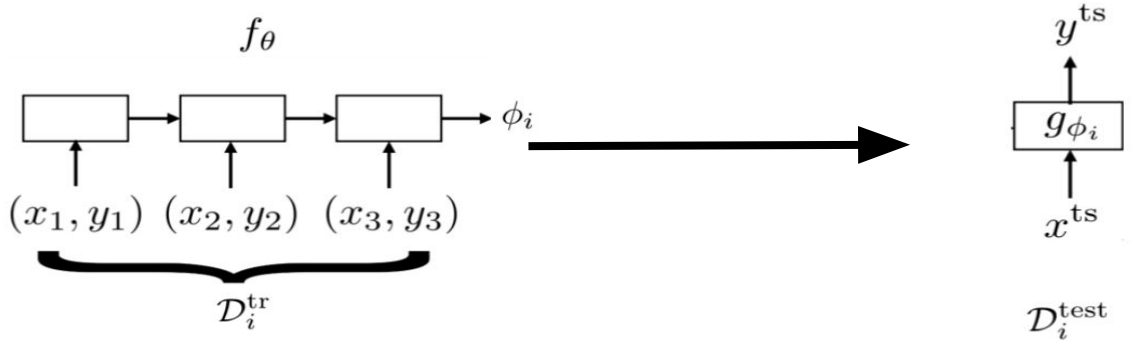
For now: Use **deterministic** (point estimate) $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$



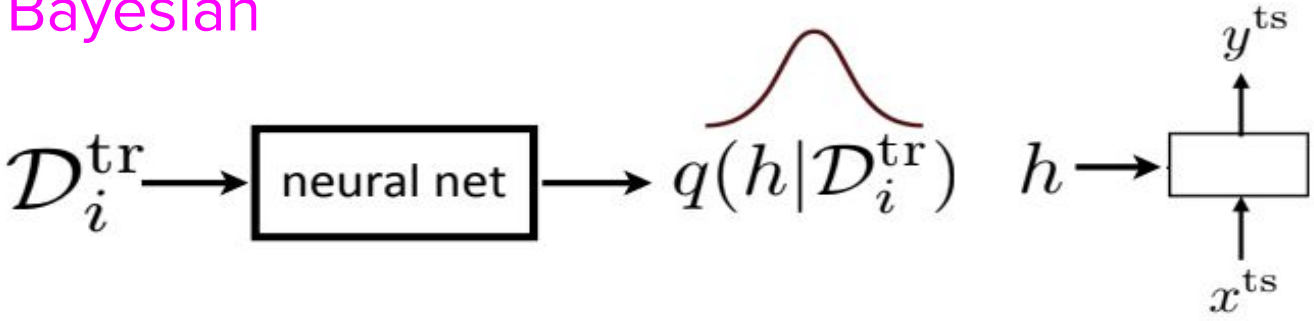
Why/when is this a problem?

Bayesian Meta Learning

Initially

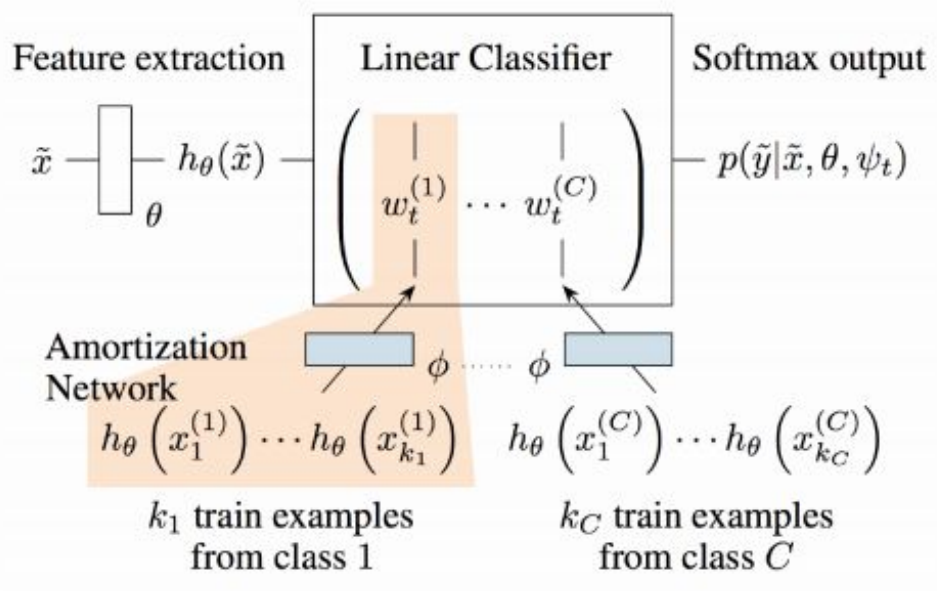


In Bayesian



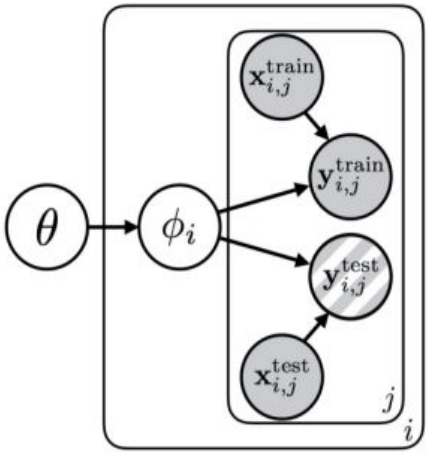
Bayesian Meta Learning

Output distribution over weights of last layer



Gordon et al. VERSA '19

Bayesian Meta Learning



$$\theta \sim p(\theta)$$

$$\phi_i \sim p(\phi_i|\theta)$$

$$\log p(y_i^{\text{train}}|x_i^{\text{train}}, \phi_i)$$

$$\log p(y_i^{\text{test}}|x_i^{\text{test}}, \phi_i)$$

Goal: sample $\phi_i \sim p(\phi_i|x_i^{\text{train}}, y_i^{\text{train}}, x_i^{\text{test}})$

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Sampling Parameter Vectors

$$\theta \sim p(\theta) = \mathcal{N}(\mu_\theta, \Sigma_\theta) \qquad \log p(y_i^{\text{train}} | x_i^{\text{train}}, \phi_i)$$

$$\phi_i \sim p(\phi_i | \theta) \qquad \log p(y_i^{\text{test}} | x_i^{\text{test}}, \phi_i)$$

Goal: sample $\phi_i \sim p(\phi_i | x_i^{\text{train}}, y_i^{\text{train}})$

$$p(\phi_i | x_i^{\text{train}}, y_i^{\text{train}}) \propto \int p(\theta) p(\phi_i | \theta) p(y_i^{\text{train}} | x_i^{\text{train}}, \phi_i) d\theta$$

$\Rightarrow$ this is completely intractable!

Sampling Parameter Vectors

what if we knew $p(\phi_i|\theta, x_i^{\text{train}}, y_i^{\text{train}})$?

$\Rightarrow$ now sampling is easy! just use ancestral sampling!

key idea: $p(\phi_i|\theta, x_i^{\text{train}}, y_i^{\text{train}}) \approx \delta(\hat{\phi}_i)$

approximate with MAP

this is **extremely** crude

but **extremely** convenient!

$$\hat{\phi}_i \approx \theta + \alpha \nabla_\theta \log p(y_i^{\text{train}}|x_i^{\text{train}}, \theta)$$

(Santos '92, Grant et al. ICLR '18)

Applications of meta-learning

- Computer Vision
  - **Few shot image recognition** : Vinyals et al. Matching Networks for One Shot Learning.
  - **Domain Adaptation**: Li et al. Learning to Generalize: Meta-learning for domain adaptation.
  - **Human motion and pose prediction**: Gui et al. Few-Shot Human Motion Prediction via Meta-Learning. Alet et al. Modular Meta-Learning.
  - **Few shot segmentation**: Shaban et al. One-Shot Learning for Semantic Segmentation.
  - **Task transfer**: Pal and Balasubramanian, Zero-Shot Task Transfer.

Applications of meta-learning

- Generative modelling
  - **Few shot image-to-image translation:** Liu, Huang, Mallya, Karras, Aila, Lehtinen, Kautz. Few-Shot Unsupervised Image-to-Image Translation.
  - **Few shot image generation**: Reed et al.. Few-Shot Autoregressive Density Estimation.
  - **Generation of novel viewpoints**: Gordon et al. VERSA: Versatile and Efficient Few-Shot Learning
  - **Generating talking heads from images**: Zakharov et al. Few-Shot Adversarial Learning of Realistic Neural Talking Head Models

Applications of meta-learning

- Language modeling
  - **Adapting to new languages**: Learn to translate new language pair w/o a lot of paired data? Gu et al.
  - **Learning new words**: Vinyals et al. Matching Networks for One Shot Learning
  - **Adapting to new personas**: Lin et al. Personalizing Dialogue Agents via Meta-Learning

Applications of meta-learning

- Reinforcement learning
  - Yu et al. One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning. 2018.
  - Wang et al. Learning to Reinforcement Learning. 2016.
  - Duan et al. RL2: Fast Reinforcement Learning via Slow Reinforcement Learning. 2016.
  - Humplik, et al. Meta reinforcement learning as task inference. 2019.

Questions?

Contact

vineethnb@iith.ac.in

https://www.iith.ac.in/~vineethnb