Meta-Learning

Seminar in Deep Neural Networks

Haocheng Yin

01.03.2022

Outline

- Motivation
- Problem Definition
- Meta-learning Algorithms
    - Optimization-based Inference
    - Black-box Adaptation
    - Non-parametric Method
- Applications

Outline

- Motivation
- Problem Definition
- Meta-learning Algorithms
  - Optimization-based Inference
  - Black-box Adaptation
  - Non-parametric Method
- Applications

Why meta-learning?

Large, diverse data → Broad generalization

+ Large models

Deng et al. 2009

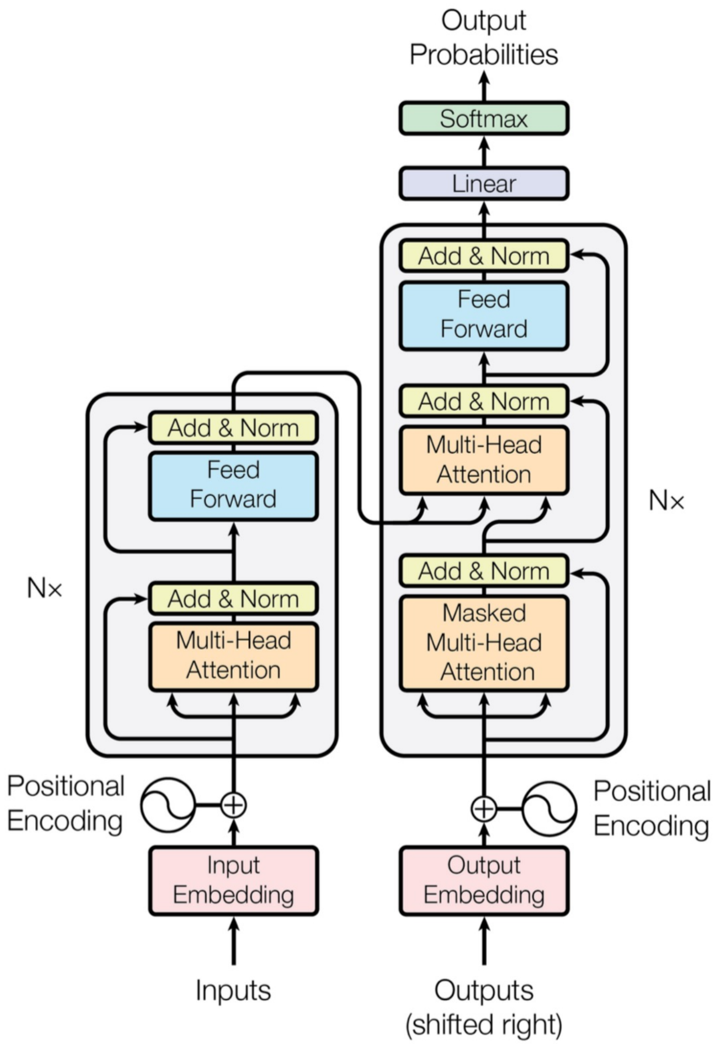


Figure 1: The Transformer - model architecture.

Vaswani et al. 2017

Why meta-learning?

- Sometimes we don't have large datasets/huge compute resources

medical imaging translation for rare languages

robotics personalized education recommendation system

- We want to develop a general-purpose AI system in the real world

continuously gain experiences over multiple related tasks and improve its future learning performance

learning strategies improve on an evolutionary timescale

Outline

- Motivation
- Problem Definition
- Meta-learning Algorithms
  - Optimization-based Inference
  - Black-box Adaptation
  - Non-parametric Method
- Applications

Two ways to view meta-learning

Mechanistic view

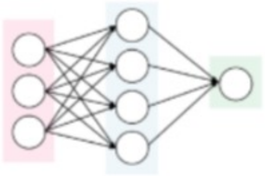
- DNN model that can read in an entire dataset and make predictions for new datapoints
- Training this network uses a meta-dataset, which itself consists of many datasets, each for a different task
- This view makes it easier to implement meta-learning algorithms

Probablisitic view

- Extract prior information from a set of tasks that allows efficient learning of new tasks
- Learning a new task using this prior and a (small) training dataset to infer most likely posterior parameters
- This view makes it easier to understand meta-learning algorithms

Meta-learning: Probablistic View

- supervised learning:



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

input (e.g., image) label

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

model parameters training data

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

data likelihood regularizer (e.g., weight decay)

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

- The most powerful models typically require large amounts of labeled data
- Labeled data for some tasks may be very limited

Meta-learning: Probablistic View

- supervised learning:

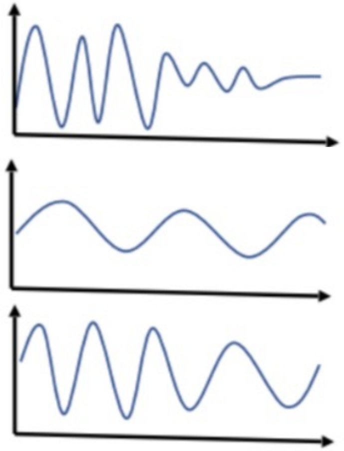
$$\arg\max_{\phi} \log p(\phi|\mathcal{D}) \qquad \mathcal{D} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$$

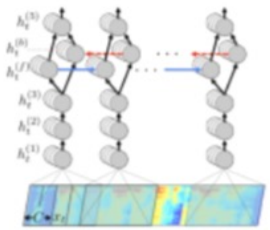
- Can we incorporate additional data?

$$\arg\max_{\phi} \log p(\phi|\mathcal{D}, \mathcal{D}_{\text{meta-train}}) \qquad \mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$$

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

5-way classification





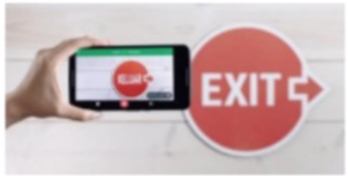


Image adapted from Ravi & Larochelle

Meta-learning Problem

- meta-learning:

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

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

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

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

- What if we don't want to keep $\mathcal{D}_{\text{meta-train}}$?

learn **meta-parameters** θ: $p(\theta|\mathcal{D}_{\text{meta-train}})$ (information we need to know about $\mathcal{D}_{\text{meta-train}}$)

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

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

meta-learning problem

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

adaptation

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

Reserve a test set for each task

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

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

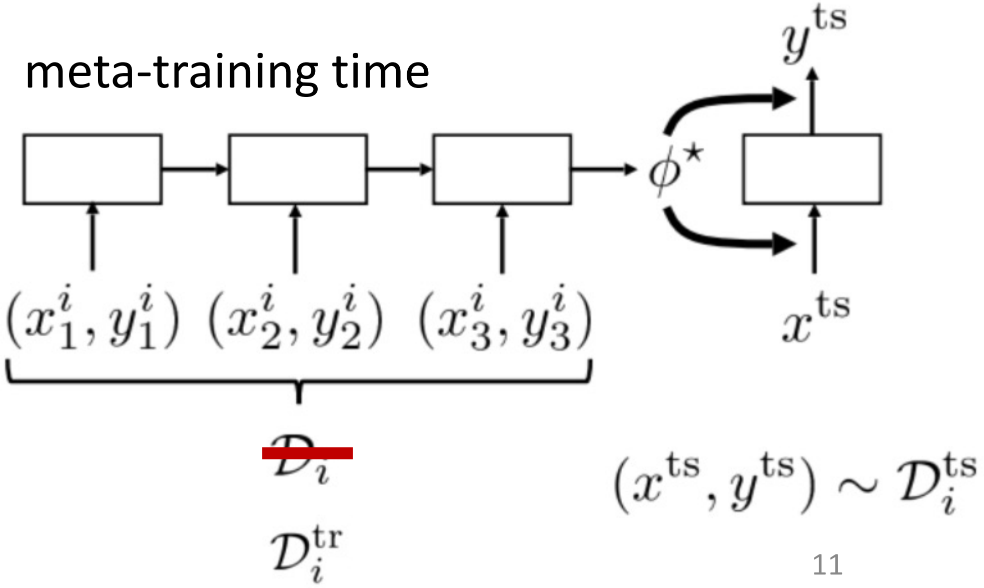
$$\cancel{\mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_1, ..., \mathcal{D}_n\}}$$

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

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

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

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



Meta-learning: Bilevel Optimization View

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

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

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

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

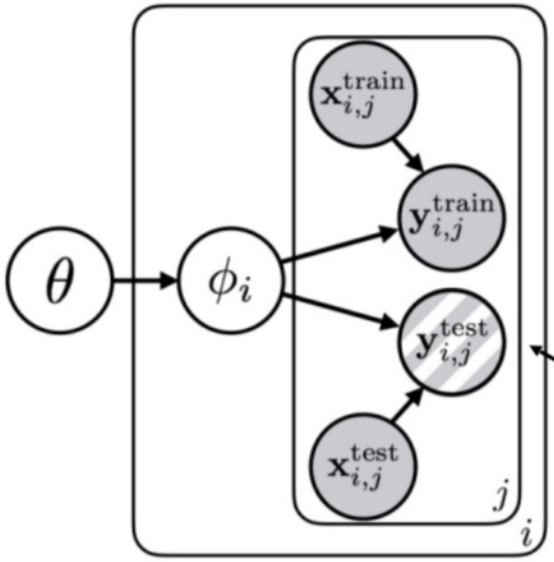
$\mathcal{D}_i^{\text{tr}} = \{(x_1^i, y_1^i), ..., (x_k^i, y_k^i)\}$

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

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

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

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



unobserved at meta-test time

Meta-learning: Terminology

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

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

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

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

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

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

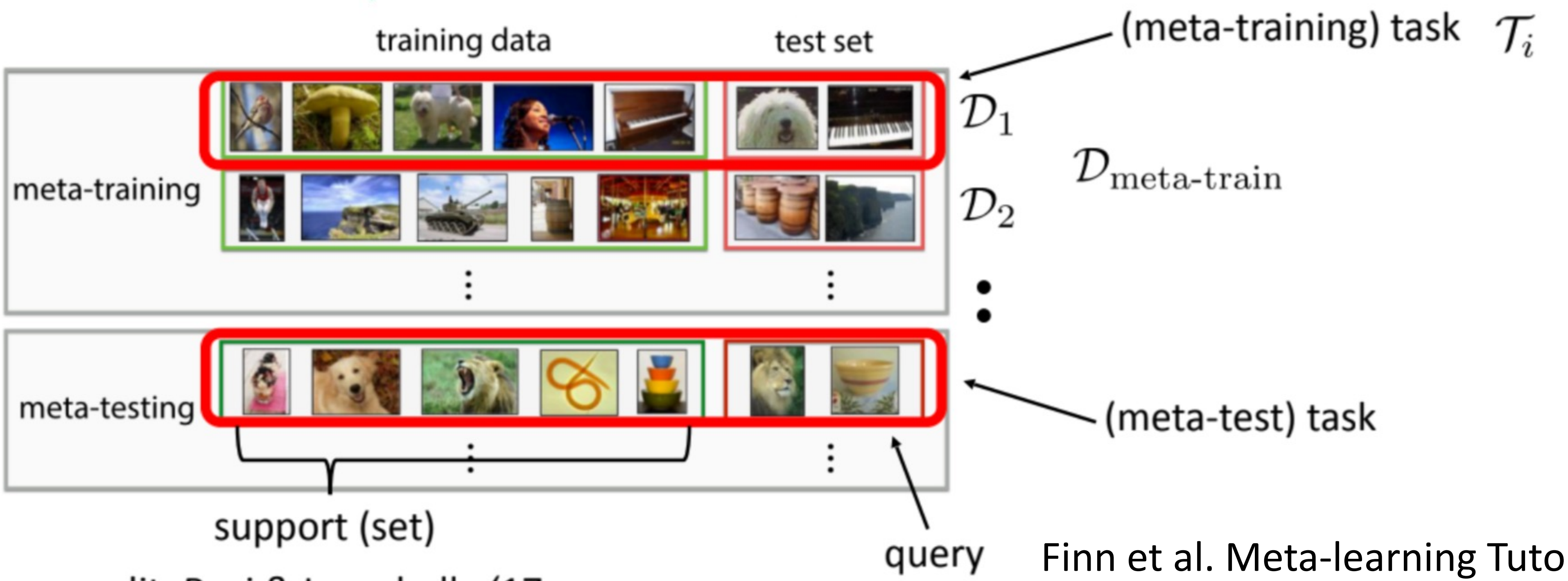


image credit: Ravi & Larochelle '17

Finn et al. Meta-learning Tutorial, ICML 2019

Related Fields

Meta-Learning

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

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

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

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

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

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

v.s.

Multi-task Learning jointly learns several tasks with parameter sharing.

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

special case where $\phi_i = \theta$

Related Fields

Meta-Learning

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

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

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

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

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

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

v.s.

Transfer Learning uses past experience from source task to improve learning on a target task.

Common approach: parameter transfer + optional fine tuning
The prior is extrated by vanilla learning on the source task <u>without the use of meta-objective</u>

Outline

- Motivation
- Problem Definition
- Meta-learning Algorithms
  - Optimization-based Inference
  - Black-box Adaptation
  - Non-parametric Method
- Applications

Black-box Adaptation

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

adaptation

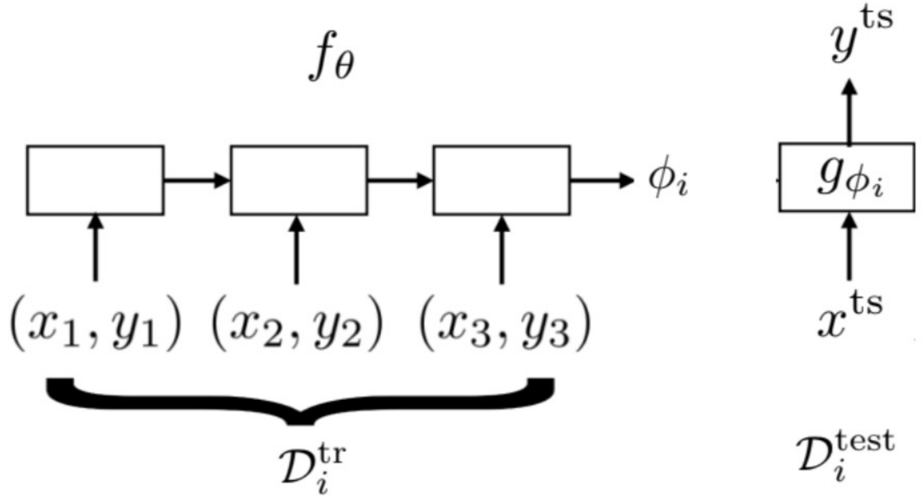
$$\phi^\star = \arg\max_{\phi} \log p(\phi|\mathcal{D}^{\mathrm{tr}}, \theta^\star)$$

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

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

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

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



Train with standard supervised learning!

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

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

Black-box Adaptation

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

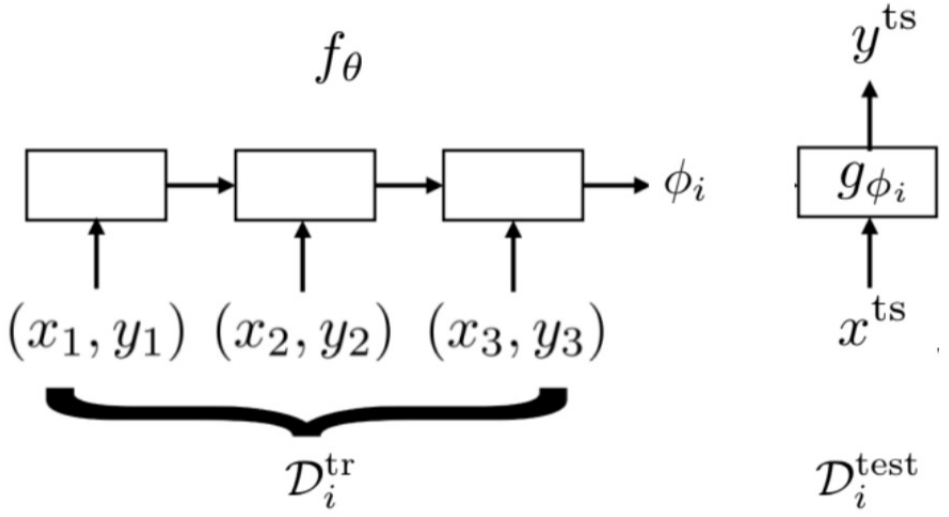
adaptation $\phi^\star = \arg\max_\phi \log p(\phi|\mathcal{D}^{\text{tr}}, \theta^\star)$

$\Updownarrow$

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

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

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



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

Finn et al. Meta-learning Tutorial, ICML 2019

Black-box Adaptation

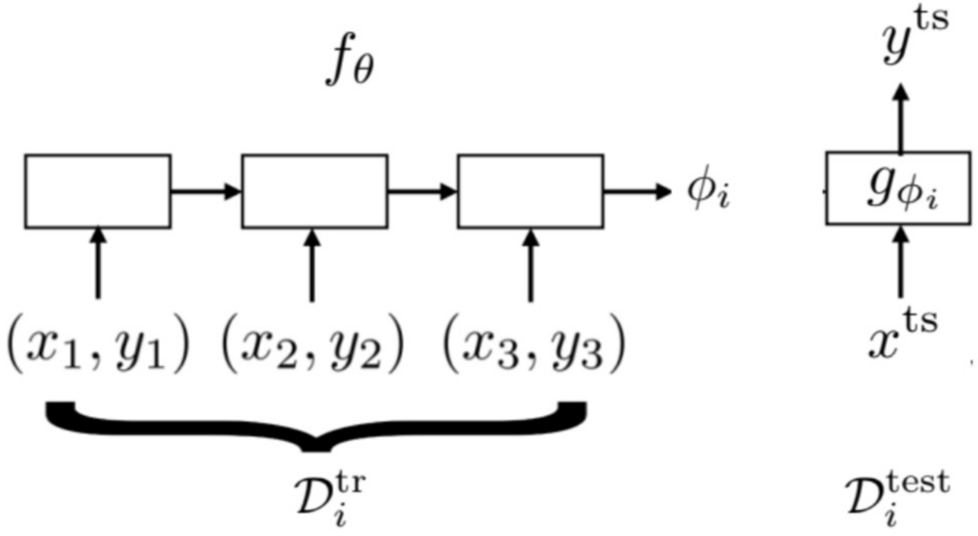
meta-learning $\theta^\star = \arg\max_\theta \log p(\theta | \mathcal{D}_{\text{meta-train}})$

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

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

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

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

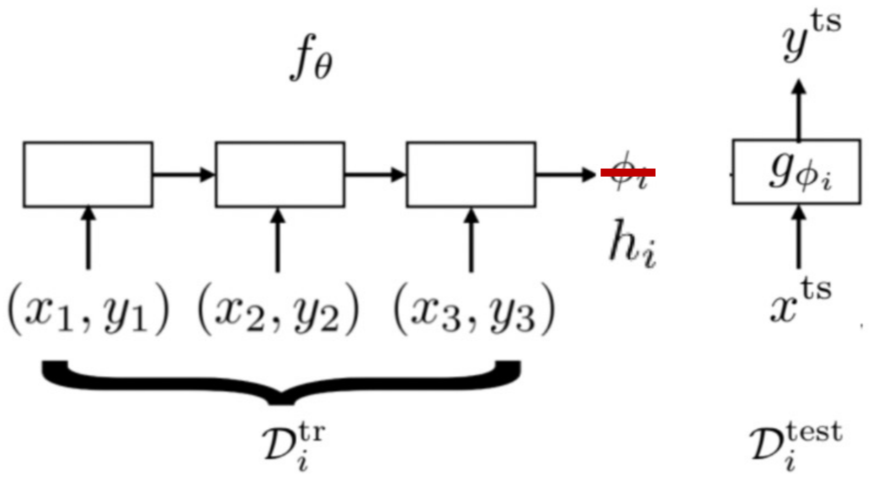


Form of f_θ ?

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

Black-box Adaptation

- Key idea: Train a neural network to represent $p(\phi_i | \mathcal{D}_i^{tr}, \theta)$
- Challenge: outputting all neural net parameters does not seem scalable
- Idea: Do not need to output **all** parameters, only sufficient statistics



(Santoro et al. MANN, Mishra et al. SNAIL)

Low dimentional vector h_i
represents contextual task information

$$\phi_i = \{h_i, \theta_g\}$$

Is there a way to infer **all parameters** in a scalable way?
Can we treat it as an **optimization** procedure?

Optimization-based Inference

- Key idea: acquire ϕ_i through optimization

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

- Meta-parameters θ serve as a prior.
  - One form of prior knowledge: **initialization for fine-tuning**

~~Amortized approach~~ Optimization-based approach

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

Finn et al. Meta-learning Tutorial, ICML 2019

Optimization-based Inference

- Key idea: acquire ϕ_i through optimization

pre-trained parameters

Fine-tuning [test-time]

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

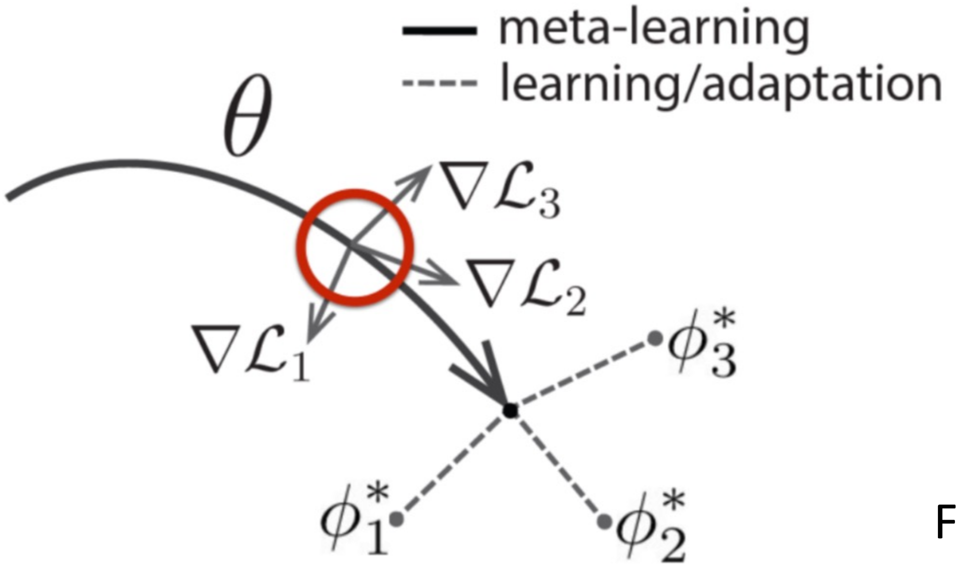
training data for new task

Meta-learning

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

θ parameter vector being meta-learned

ϕ_i^* optimal parameter vector for task i



Finn et al. MAML

Optimization-based Inference

- Key idea: acquire ϕ_i through optimization
- Challenge: second-order derivative :(

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

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)

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**

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

Other forms of priors?

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

Rajeswaran et al. implicit MAML '19

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

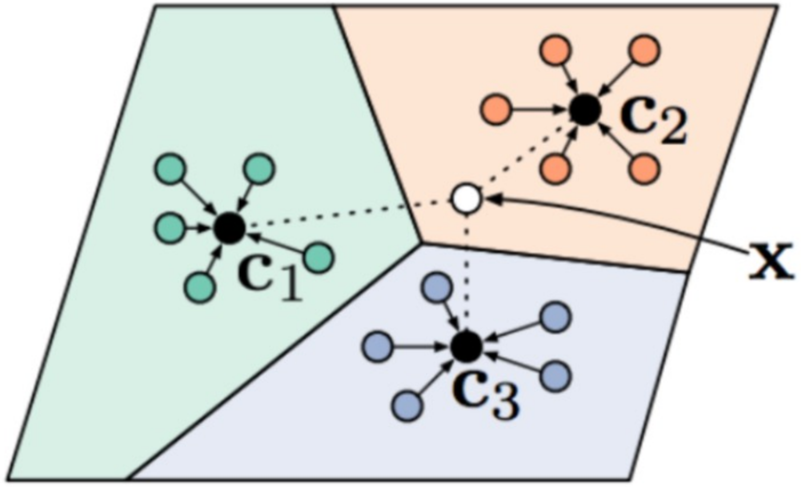
Closed-form *or* **convex optimization** *on learned features*

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

support vector machine — Lee et al. MetaOptNet '19

Non-parametric Methods

- Key idea: use non-parametric learner



(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

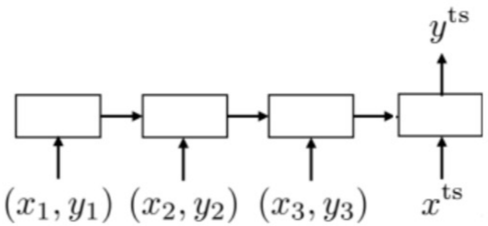
Snell et al. Prototypical Networks, NeurIPS 2017

Black-box vs. Optimization vs. Non-parametric

Computation graph perspective

Black-box amortized

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



Optimization-based

$$\begin{aligned} y^{\text{ts}} &= f_{\text{MAML}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}}) \\ &= f_{\phi_i}(x^{\text{ts}}) \end{aligned}$$
$$\text{where } \phi_i = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}})$$

Non-parametric

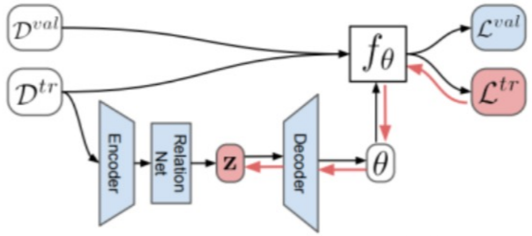
$$\begin{aligned} y^{\text{ts}} &= f_{\text{PN}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}}) \\ &= \text{softmax}\left(-d(f_\theta(x), c_k)\right) \end{aligned}$$
$$\text{where } c_k = \frac{1}{|\mathcal{D}_i^{\text{tr}}|} \sum_{(x,y) \in \mathcal{D}_i^{\text{tr}}} f_\theta(x)$$

Note: (again) Can mix & match components of computation graph

Both condition on data & run gradient descent.

Jiang et al. CAML '19

Gradient descent on relation net embedding.



Rusu et al. LEO '19

MAML, but initialize last layer as ProtoNet during meta-training

Triantafillou et al. Proto-MAML '19

Black-box vs. Optimization vs. Non-parametric

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**

Generally, well-tuned versions of each perform **comparably** on existing few-shot benchmarks!

Outline

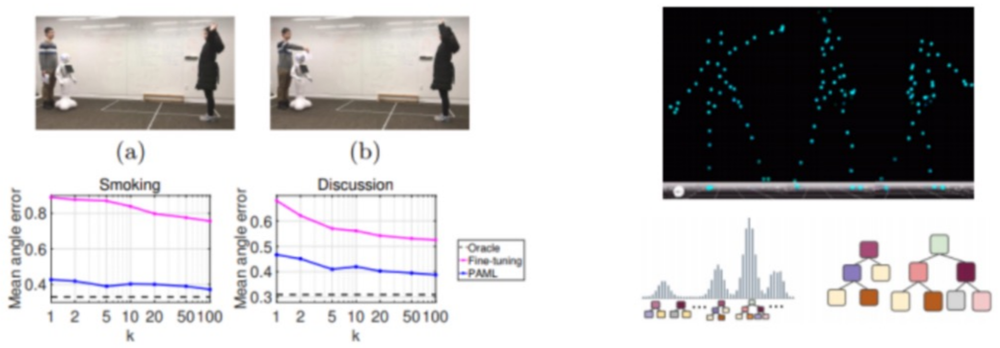
- Motivation
- Problem Definition
- Meta-learning Algorithms
  - Optimization-based Inference
  - Black-box Adaptation
  - Non-parametric Method
- Applications

Applications in computer vision

few-shot image recognition

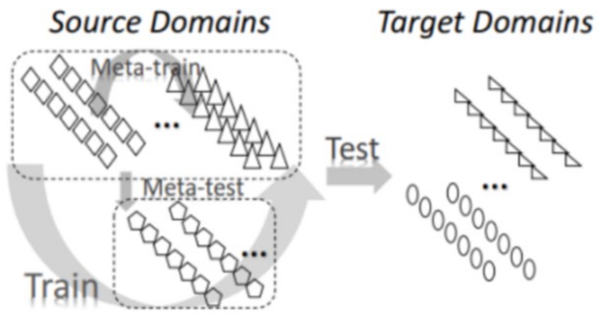
see, e.g.: Vinyals et al. **Matching Networks for One Shot Learning**, and many many others

human motion and pose prediction



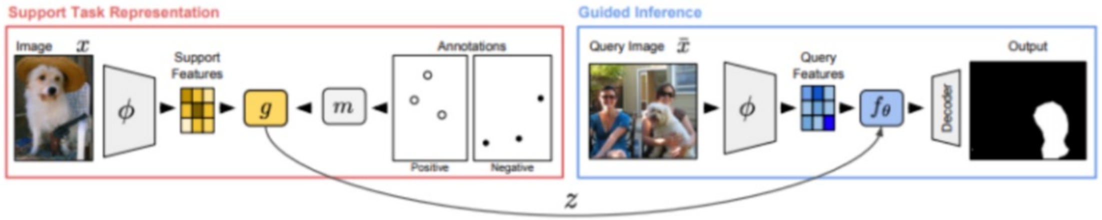
see, e.g.: Gui et al. **Few-Shot Human Motion Prediction via Meta-Learning.**
Alet et al. **Modular Meta-Learning.**

domain adaptation



see, e.g.: Li, Yang, Song, Hospedales. **Learning to Generalize: Meta-Learning for Domain Adaptation.**

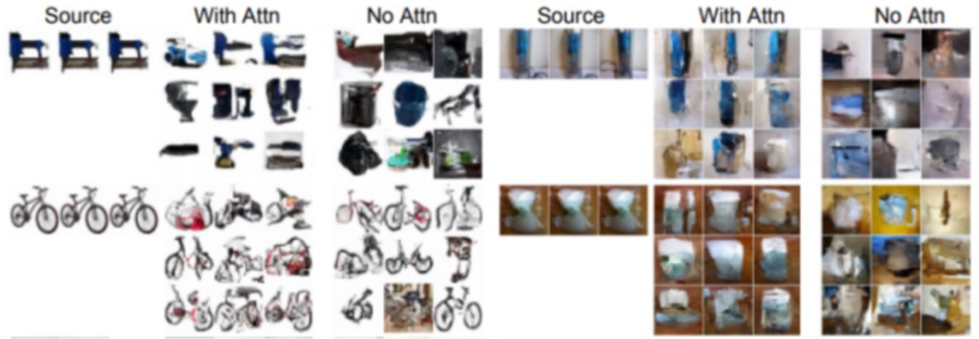
few-shot segmentation



see, e.g.: Shaban, Bansal, Liu, Essa, Boots. **One-Shot Learning for Semantic Segmentation.**
Rakelly, Shelhamer, Darrell, Efros, Levine. **Few-Shot Segmentation Propagation with Guided Networks.**
Dong, Xing. **Few-Shot Semantic Segmentation with Prototype Learning.**

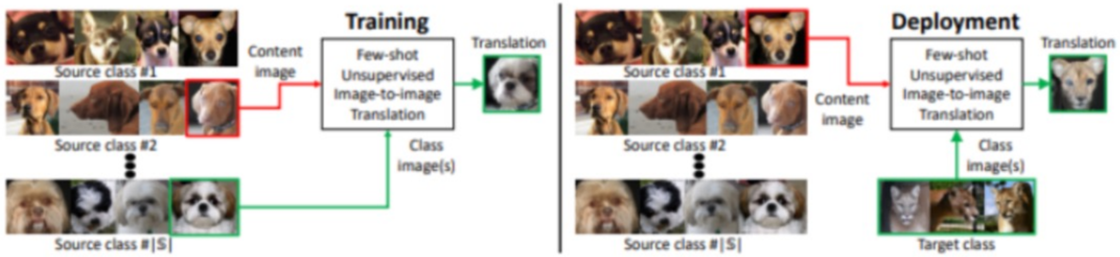
Applications in image & video generation

few-shot image generation



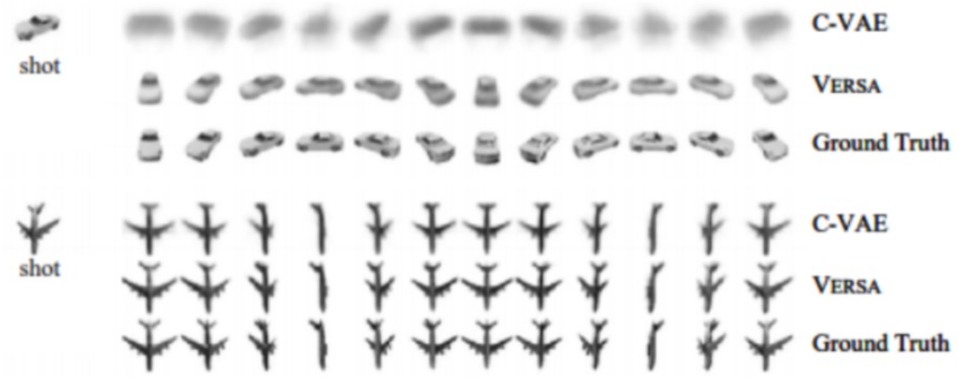
see, e.g.: Reed, Chen, Paine, van den Oord, Eslami, Rezende, Vinyals, de Freitas. **Few-Shot Autoregressive Density Estimation**. and many many others.

few-shot image-to-image translation



see, e.g.: Liu, Huang, Mallya, Karras, Aila, Lehtinen, Kautz. **Few-Shot Unsupervised Image-to-Image Translation.**

generation of novel viewpoints



see, e.g.: Gordon, Bronskill, Bauer, Nowozin, Turner. **VERSA: Versatile and Efficient Few-Shot Learning.**

generating talking heads from images

see, e.g.: Zakharov, Shysheya, Burkov, Lempitsky. **Few-Shot Adversarial Learning of Realistic Neural Talking Head Models**

Applications in NLP

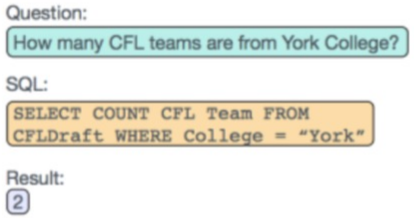
Adapting to *new programs*

Meta Program Induction

Learn new program from a few I/O examples.

Devlin, Bunel* et al. NeurIPS '17*

Program Synthesis

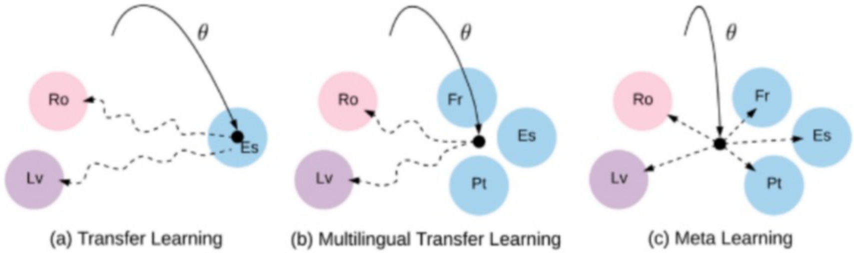


Construct pseudo-tasks with relevance function

Huang et al. NAACL '18

Adapting to *new languages*

Low-Resource Neural Machine Translation



Learn to translate new language pair w/o a lot of paired data?

Gu et al. EMNLP '18

Learning *new words*

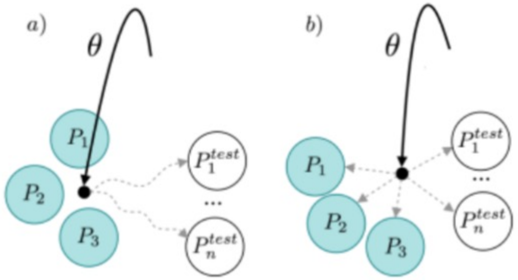
One-Shot Language Modeling

Learn how to use a new word from one example usage.

Vinyals et al. Matching Networks, '16

Adapting to *new personas*

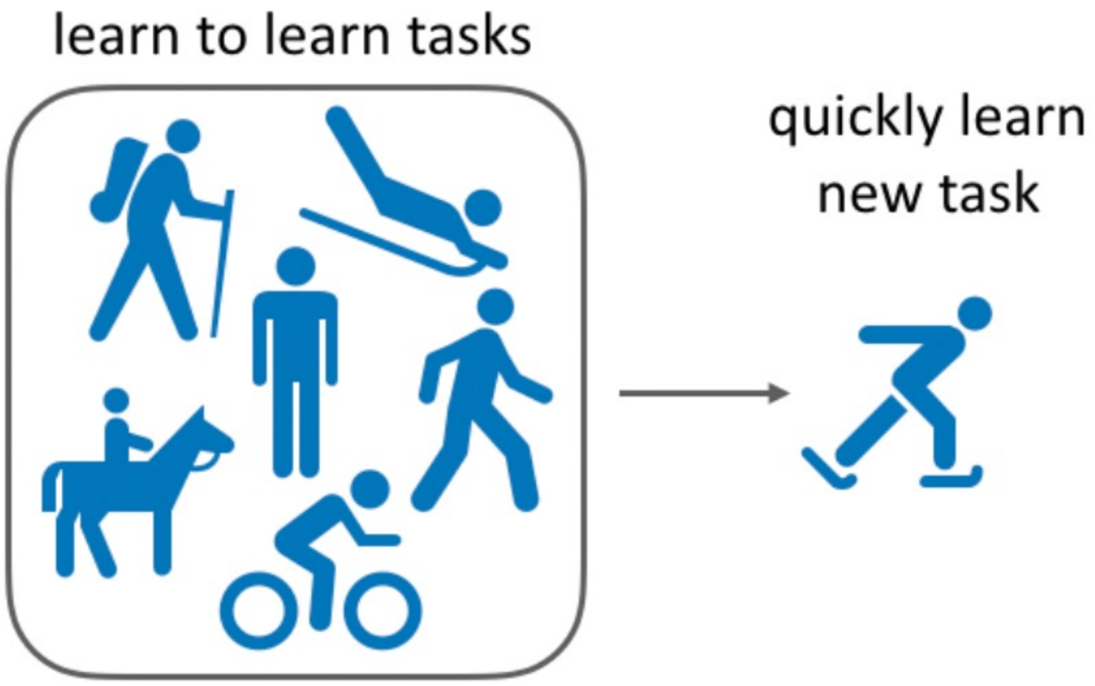
Personalizing Dialogue Agents



Adapt dialogue to a persona with a few examples

Lin, Madotto* et al. ACL '19*

Thank you for your attention!



References

- Koch et al. "Siamese Neural Networks for One-shot Image Recognition." ICML, 2015.
- Vinyals et al. "Matching networks for one shot learning." NeuraIPS, 2016.
- Snell et al. "Prototypical networks for few-shot learning." NeuraIPS, 2017.
- Finn et al. "Model-agnostic meta-learning for fast adaptation of deep networks.” PMLR, 2017.
- Finn et al. "Meta-learning Tutorial." ICML, 2019.
- Hospedales et al. "Meta-learning in neural networks: A survey." *arXiv*, 2020.