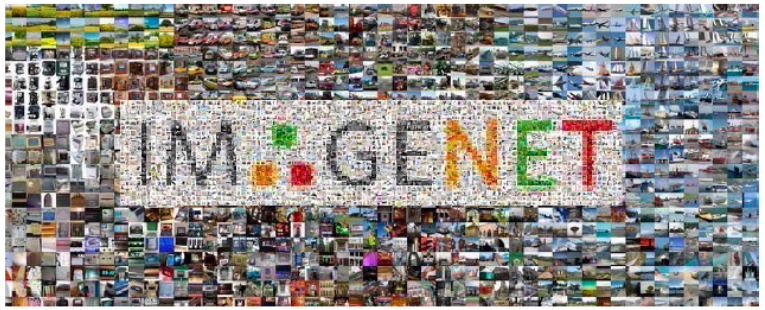
Lecture 05: Meta Learning

Readings:

- Survey Paper (posted)

Large, diverse data
(+ large models)



Russakovsky et al. ‘14

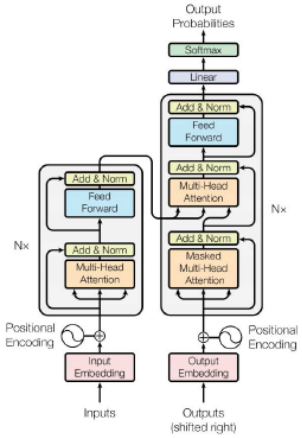


Figure 1: The Transformer - model architecture.

Vaswani et al. ‘18

GPT-2

Radford et al. ‘19

Under the paradigm of supervised learning.

What if you don't have a large dataset?

medical imaging

robotics

personalized education, recommendations

translation for rare languages

What if you want a general-purpose AI system in the real world?

Need to continuously adapt and learn on the job.

Learning each thing from scratch won't cut it.

What if your data has a long tail?

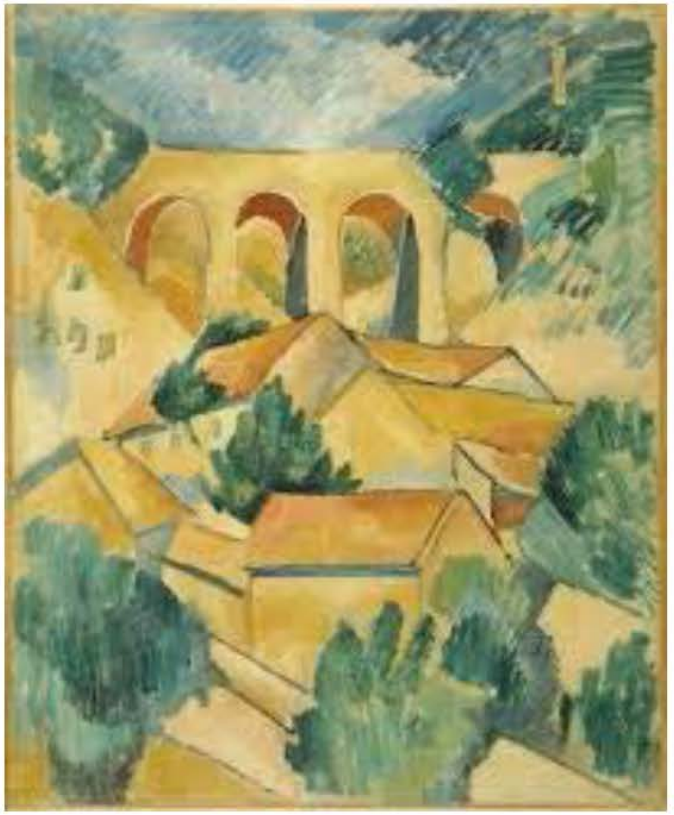
These settings break the supervised learning paradigm.

training data

Braque

Cezanne

test datapoint



By Braque or Cezanne?

How did you accomplish this?

Through previous experience.

How might you get a machine to accomplish this task?

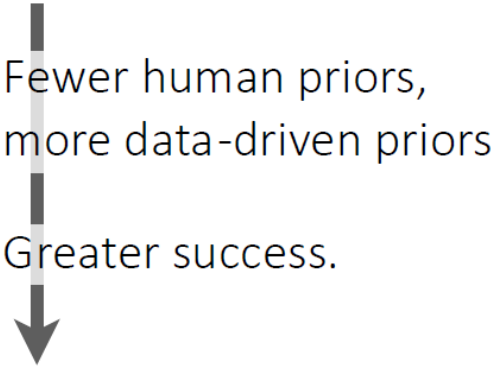
Modeling image formation

Geometry

SIFT features, HOG features + SVM

Fine-tuning from ImageNet features
Domain adaptation from other painters

???



Can we explicitly learn priors from previous experience that lead to efficient downstream learning?

Can we learn to learn?

Problem Statement

Two ways to view meta-learning

Mechanistic view

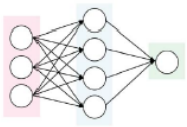
- Deep neural network model that can read in an entire dataset and make predictions for new datapoints
- Training this network uses a meta-dataset, which itself consists of many datasets, each for a different task
- This view makes it easier to implement meta-learning algorithms

Probabilistic view

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

Problem definitions

supervised learning:



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

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)

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

What is wrong with this?

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

Problem definitions

supervised learning:

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

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

can we incorporate *additional* data?

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

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

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

$\mathcal{D}_{\text{meta-train}}$

$\mathcal{D}$

$\mathcal{D}_1$

$\mathcal{D}_2$

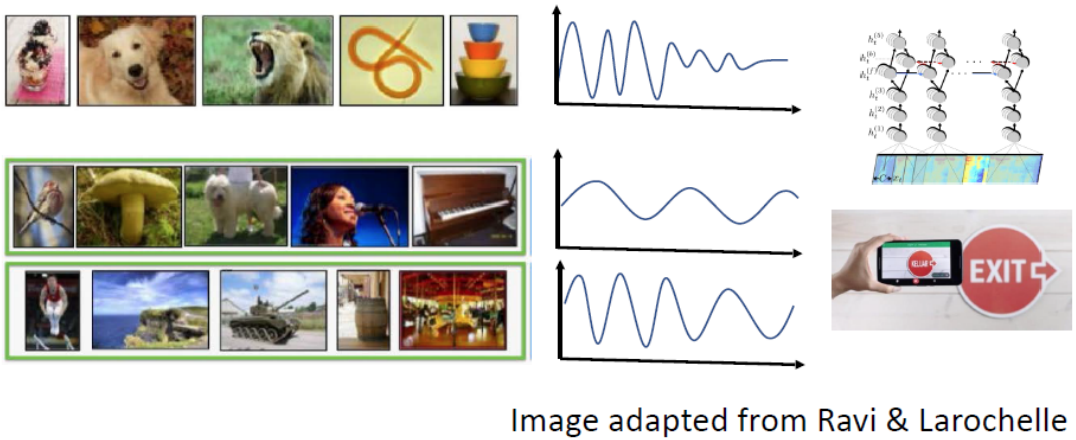


Image adapted from Ravi & Larochelle

The 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 $\mathcal{D}_{\text{meta-train}}$ around forever?

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

whatever we need to know about $\mathcal{D}_{\text{meta-train}}$ to solve new tasks

this is the meta-learning problem

assume $\phi \perp\!\!\!\perp \mathcal{D}_{\text{meta-train}}|\theta$

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

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

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

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

A Quick Example

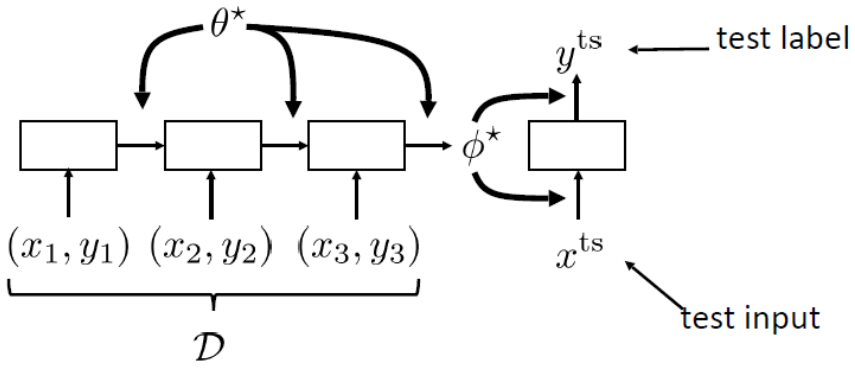
$$\text{meta-learning: } \theta^\star = \arg\max_\theta \log p(\theta|\mathcal{D}_{\text{meta-train}})$$

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

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

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

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



$\mathcal{D}$

$\mathcal{D}_{\text{meta-train}}$ $\mathcal{D}_1$

$\mathcal{D}_2$

How do we train this thing?

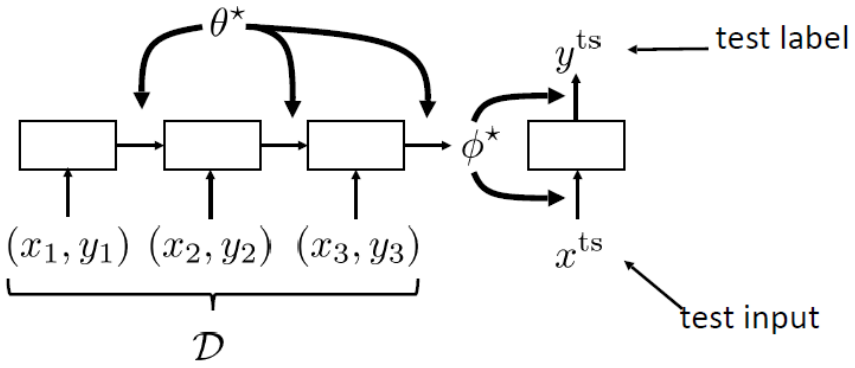
meta-learning: $\theta^\star = \arg\max_\theta \log p(\theta|\mathcal{D}_{\text{meta-train}})$

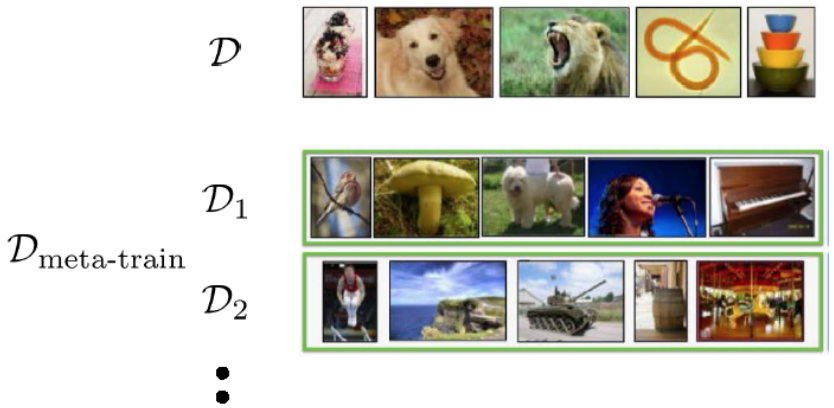
adaptation: $\phi^\star = \arg\max_\phi \log p(\phi|\mathcal{D}, \theta^\star)$

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

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

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





Key idea:

"our training procedure is based on a simple machine learning principle: test and train conditions must match"

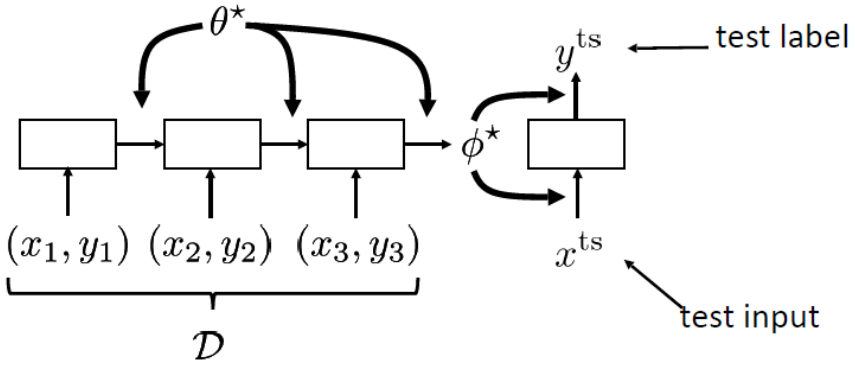
Vinyals et al., Matching Networks for One-Shot Learning

How do we train this thing?

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

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

(meta) test-time

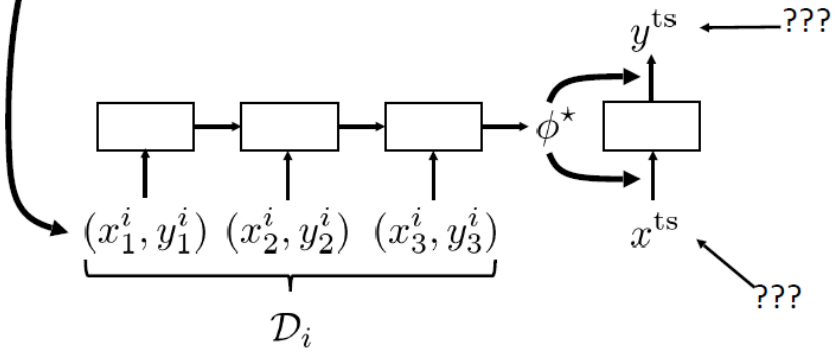


$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$$

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

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

(meta) training-time



Key idea:

"our training procedure is based on a simple machine learning principle: test and train conditions must match"

Vinyals et al., Matching Networks for One-Shot Learning

Reserve a test set for each task!

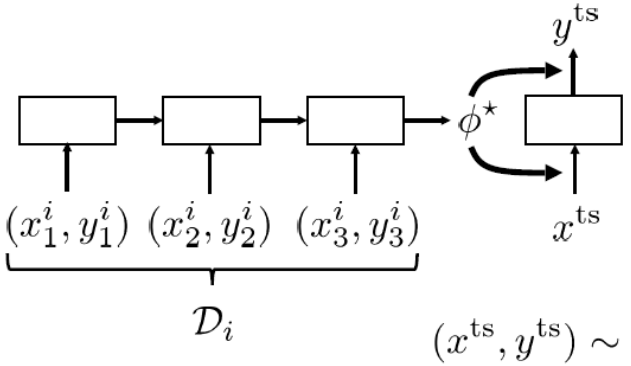
$\mathcal{D}_{\text{meta-train}}$ $\mathcal{D}_1$ $\mathcal{D}_2$ $\vdots$

(meta) training-time

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

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

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



Key idea:

"our training procedure is based on a simple machine learning principle: test and train conditions must match"

Vinyals et al., Matching Networks for One-Shot Learning

The complete meta-learning optimization

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

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

$$\Updownarrow$$

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

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

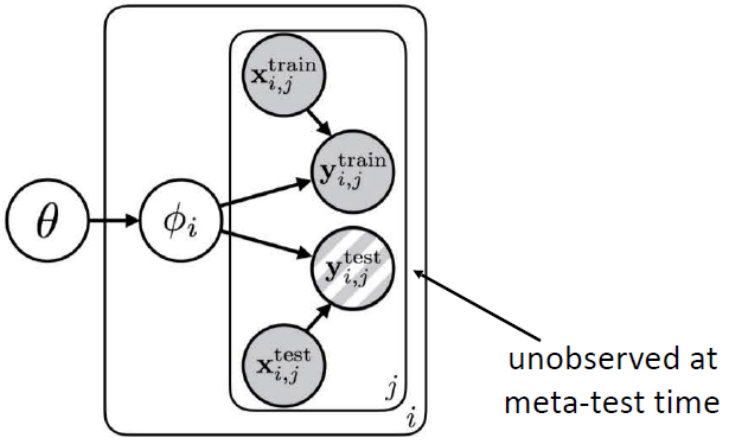
$$\theta^\star = \max_\theta \sum_{i=1}^{n} \log p(\phi_i|\mathcal{D}_i^{\text{ts}})$$

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

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

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

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



Some meta-learning terminology

learn θ such that $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$ 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 $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$

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

$$\mathcal{T}_i \begin{cases} \mathcal{D}_i^{\text{tr}} = \{(x_1^i, y_1^i), \ldots, (x_k^i, y_k^i)\} \\ \mathcal{D}_i^{\text{ts}} = \{(x_1^i, y_1^i), \ldots, (x_l^i, y_l^i)\} \end{cases}$$

shot
(i.e., k-shot, 5-shot)

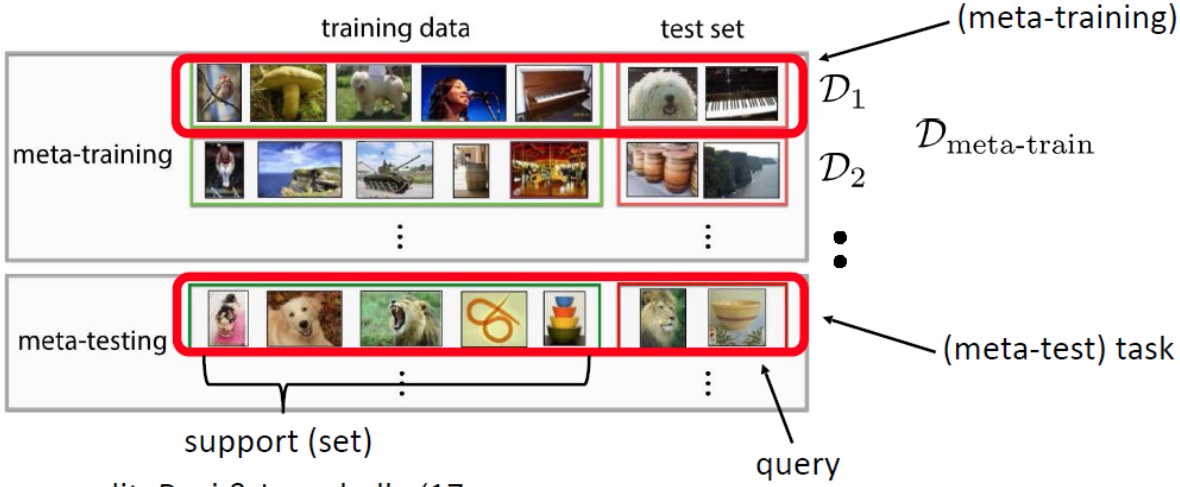


image credit: Ravi & Larochelle '17

Closely related problem settings

meta-learning:

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

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

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

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

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

multi-task learning: learn model with parameters $\theta^\star$ that solves multiple tasks $\theta^\star = \arg\max_\theta \sum_{i=1}^{n} \log p(\theta | \mathcal{D}_i)$

can be seen as special case where $\phi_i = \theta$ (i.e., $f_\theta(\mathcal{D}_i) = \theta$)

hyperparameter optimization & auto-ML: can be cast as meta-learning

hyperparameter optimization: θ = hyperparameters, ϕ = network weights

architecture search: θ = architecture, ϕ = network weights

very active area of research! but outside the scope of this tutorial

Outline

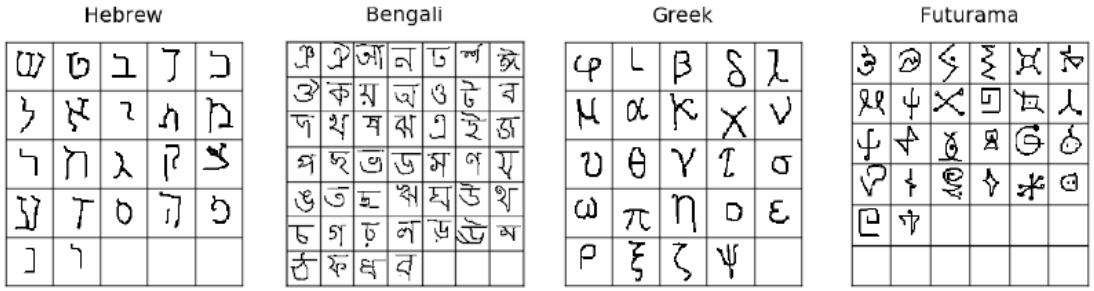
- Motivation
- Problem Statement
- Meta-Learning Approaches
  - Black-Box Adaptation
  - Non-Parametric Methods
- Applications

General recipe

How to *evaluate* a meta-learning algorithm

the Omniglot dataset Lake et al. Science 2015

1623 characters from 50 different alphabets



20 instances of each character

many classes, few examples

the "transpose" of MNIST

statistics more reflective of the real world

Proposes both **few-shot discriminative** & **few-shot generative** problems

Initial few-shot learning approaches w/ **Bayesian models, non-parametrics**

Fei-Fei et al. '03 Lake et al. '11 Salakhutdinov et al. '12 Lake et al. '13

Other datasets used for **few-shot image recognition**: MiniImagenet, CIFAR, CUB, CelebA, others

General recipe

How to *evaluate* a meta-learning algorithm

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

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

General recipe

How to *design* a meta-learning algorithm

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 $\mathcal{D}_{\text{meta-train}}$

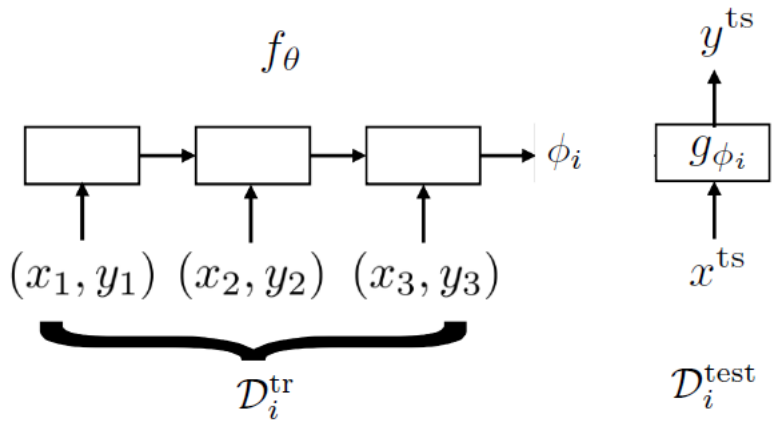
Can we treat $p(\phi_i|\mathcal{D}_i^{\text{tr}}, \theta)$ as an **inference** problem?

Neural networks are good at inference.

Black-Box Adaptation

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

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



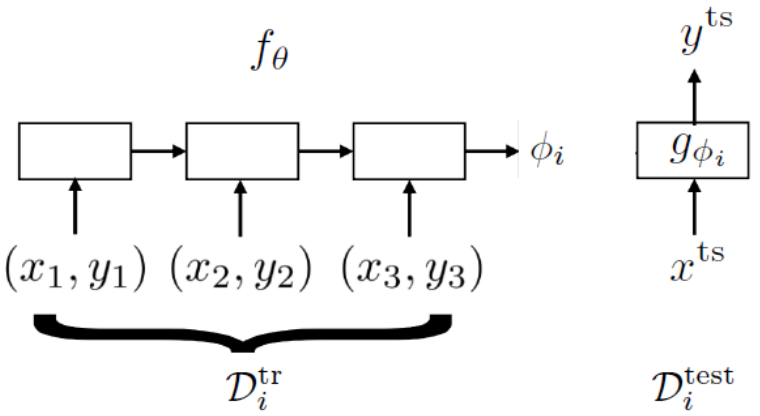
Train with standard supervised learning!

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

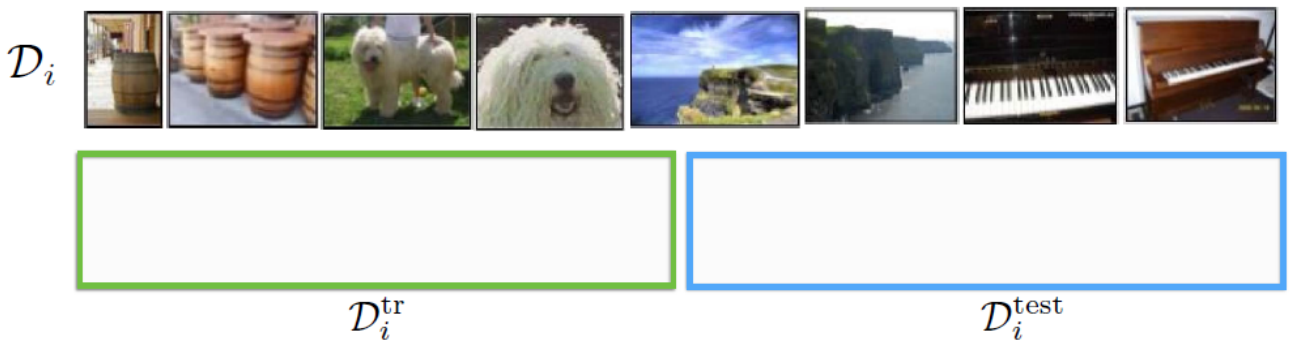
$$\max_\theta \sum_{\mathcal{T}_i} \mathcal{L}(f_\theta(\mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{test}})$$

Black-Box Adaptation

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

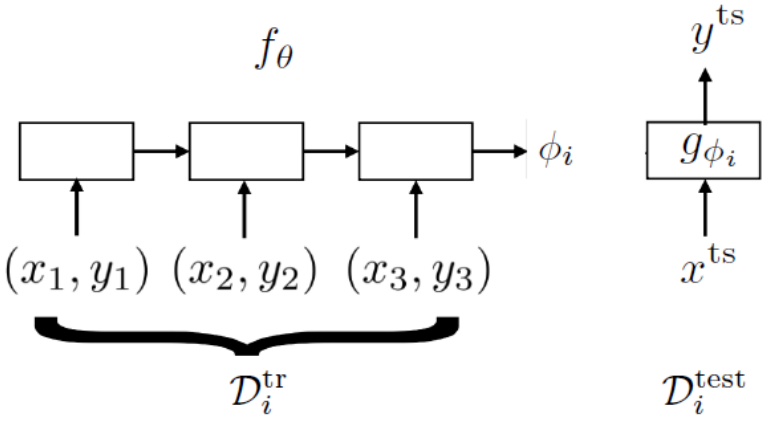


1. Sample task $\mathcal{T}_i$ *(or mini batch of tasks)*
2. Sample disjoint datasets $\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{test}}$ from $\mathcal{D}_i$

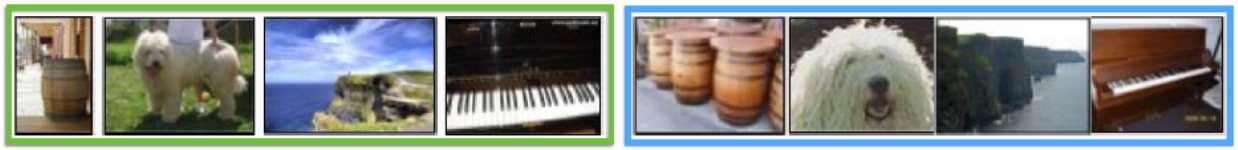


Black-Box Adaptation

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



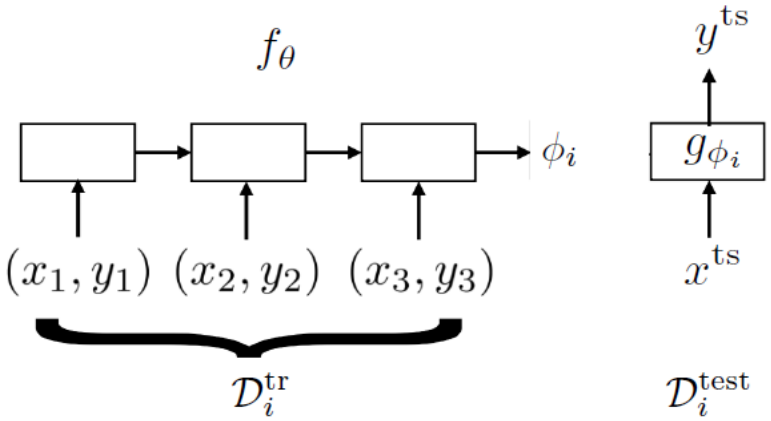
1. Sample task $\mathcal{T}_i$ *(or mini batch of tasks)*
2. Sample disjoint datasets $\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{test}}$ from $\mathcal{D}_i$
3. Compute ϕ_i $f_\theta(\mathcal{D}_i^{\text{tr}})$
4. Update θ using $\nabla_\theta \mathcal{L}(\phi_i, \mathcal{D}_i^{\text{test}})$



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

Black-Box Adaptation

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



Form of f_θ ?

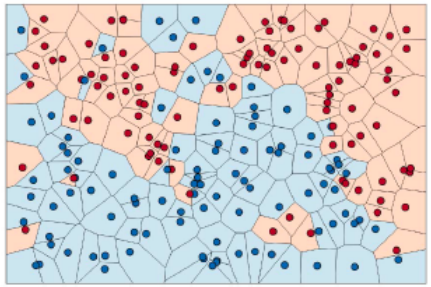
- LSTM
- Neural turing machine (NTM)
- Self-attention
- 1D convolutions
- feedforward + average

Outline

- Motivation
- Problem Statement
- Meta-Learning Approaches
  - Black-Box Adaptation
  - Non-Parametric Methods
- Applications

So far: Learning parametric models.

In low data regimes, **non-parametric** methods are simple, work well.



During **meta-test time**: few-shot learning <-> low data regime

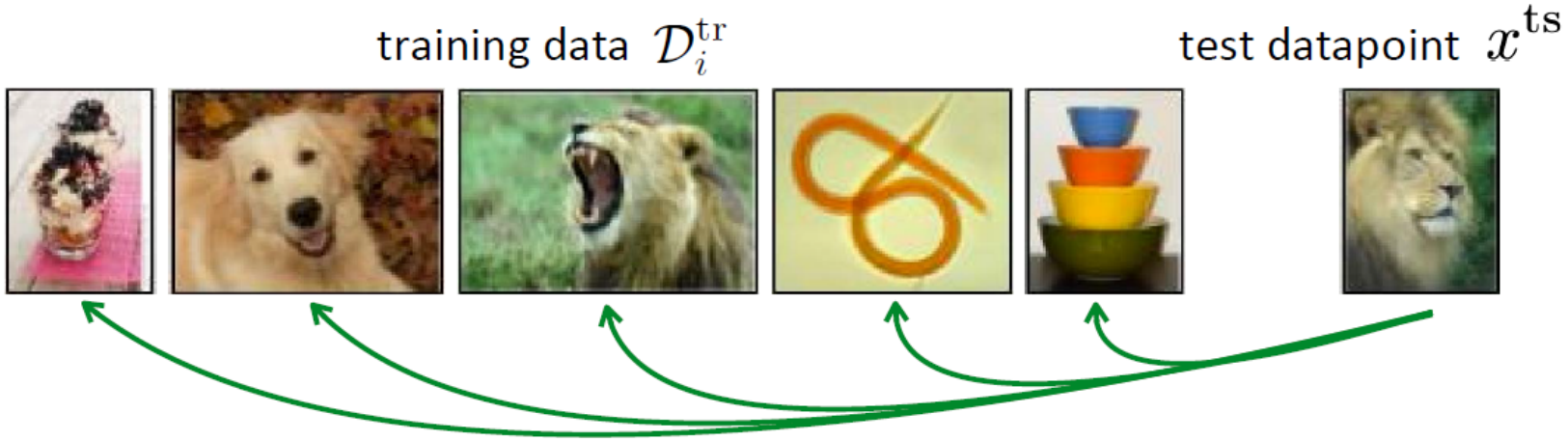
During **meta-training**: still want to be parametric

Can we use **parametric meta-learners** that produce effective **non-parametric learners**?

Note: some of these methods precede parametric approaches

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?

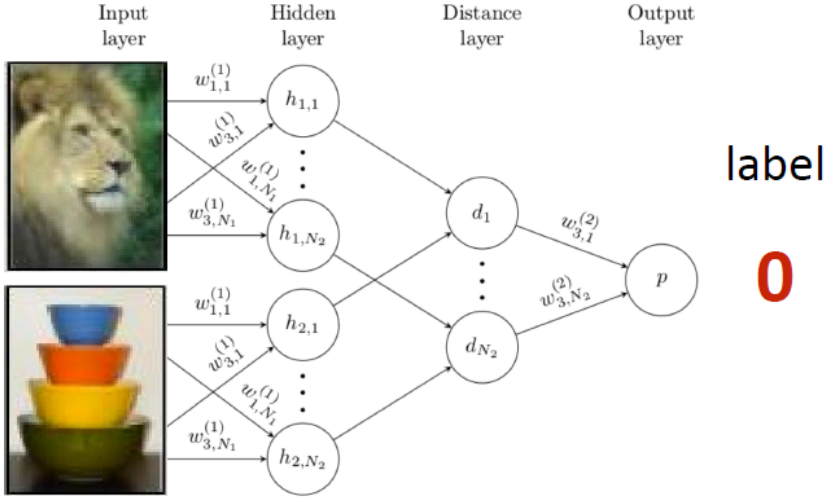
~~pixel space, l_2 distance?~~

Learn to compare using data!

Non-parametric methods

Key Idea: Use non-parametric learner.

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

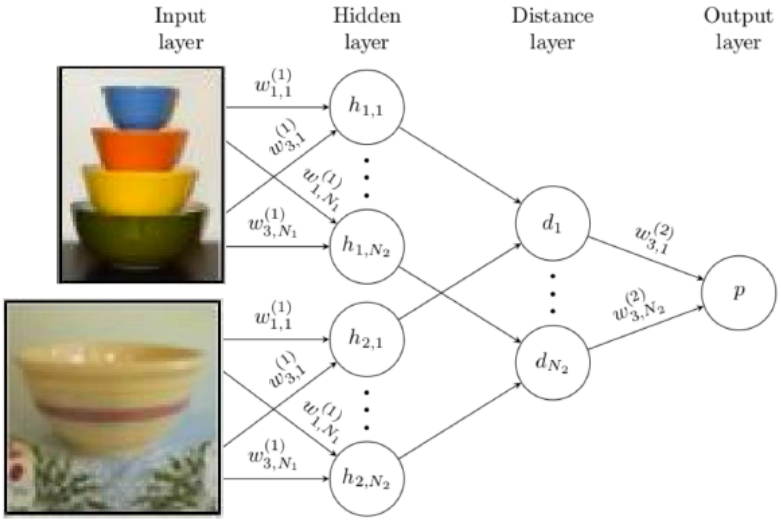


Koch et al., ICML '15

Non-parametric methods

Key Idea: Use non-parametric learner.

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



label

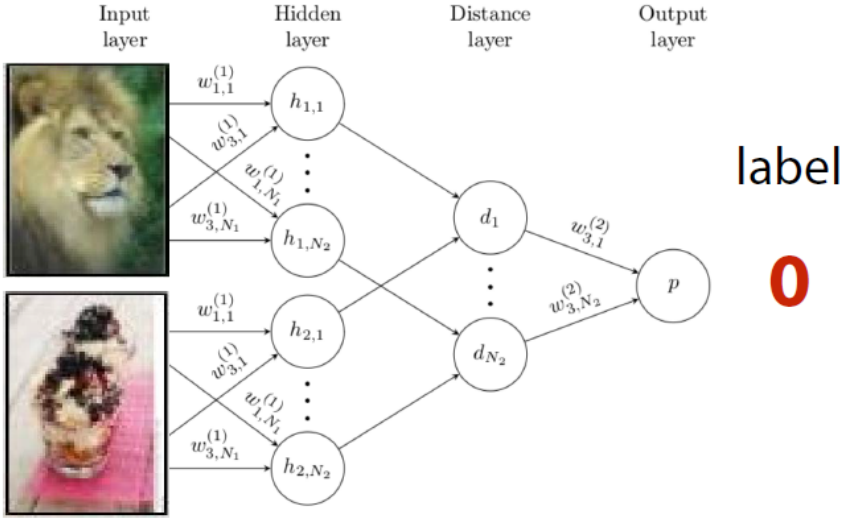
1

Koch et al., ICML '15

Non-parametric methods

Key Idea: Use non-parametric learner.

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

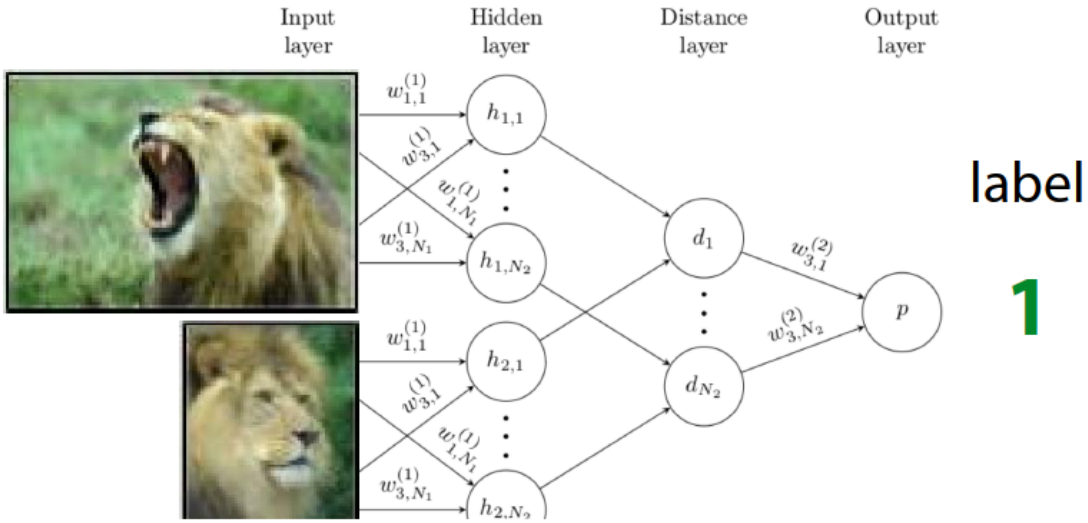


Koch et al., ICML '15

Non-parametric methods

Key Idea: Use non-parametric learner.

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



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

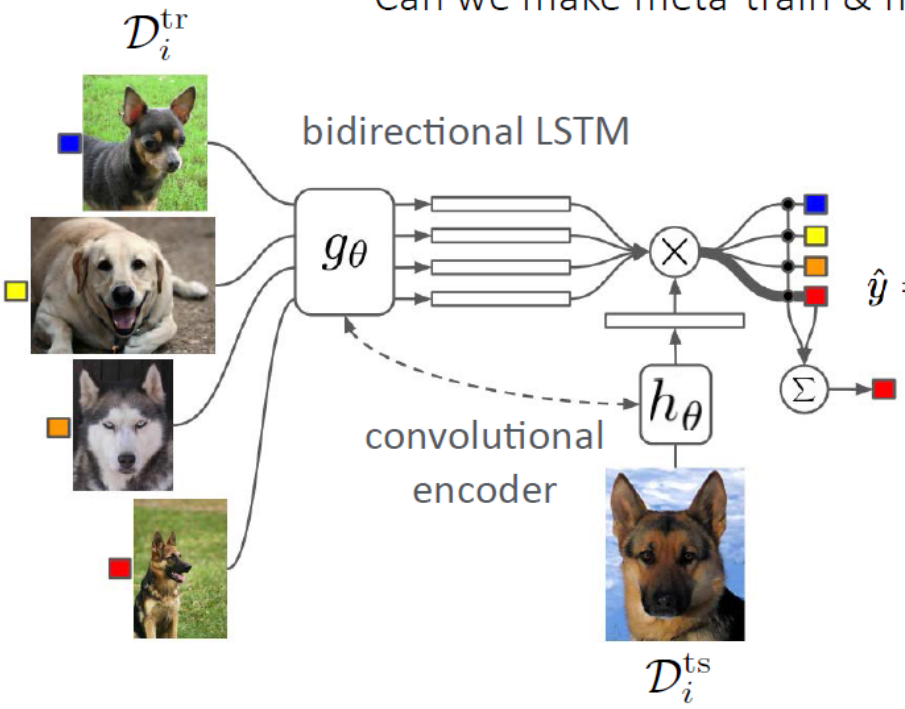
Meta-training: 2-way classification
Meta-test: N-way classification

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

Non-parametric methods

Key Idea: Use non-parametric learner.

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



Weighed nearest neighbors in learned embedding space

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

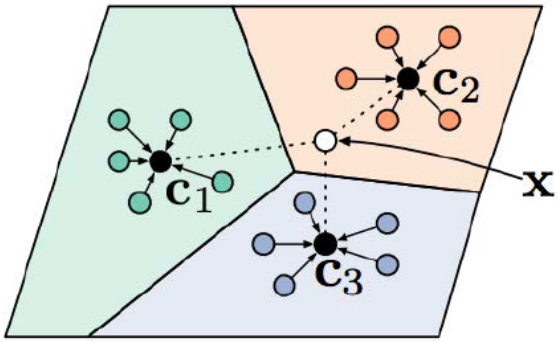
What if >1 shot?

Can we aggregate class information to create a prototypical embedding?

Vinyals et al. Matching Networks, NeurIPS '16

Non-parametric methods

Key Idea: Use non-parametric learner.



(a) Few-shot

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

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

d: Euclidean, or cosine distance

Snell et al. Prototypical Networks, NeurIPS '17

Non-parametric methods

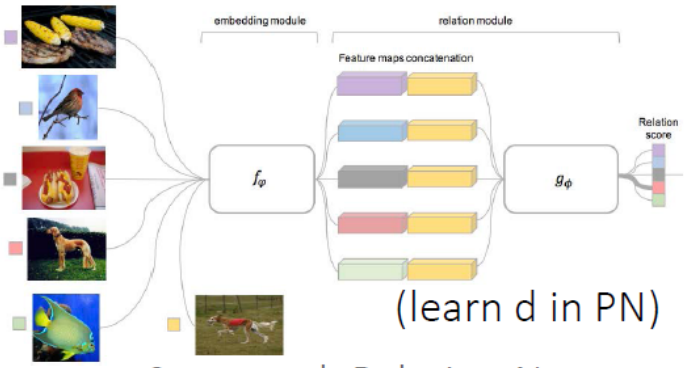
So far: Siamese networks, matching networks, prototypical networks

Embed, then nearest neighbors.

Challenge

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

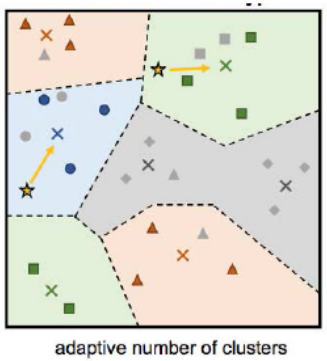
Idea: Learn non-linear relation module on embeddings



(learn d in PN)

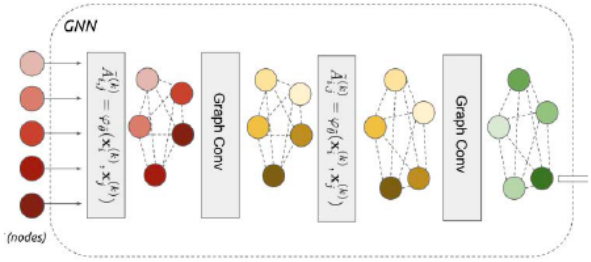
Sung et al. Relation Net

Idea: Learn infinite mixture of prototypes.



Allen et al. IMP, ICML '19

Idea: Perform message passing on embeddings



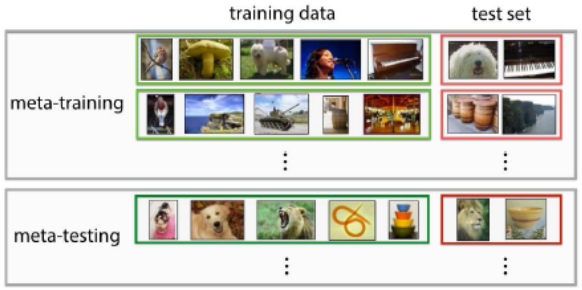
Garcia & Bruna, GNN

Outline

- Motivation
- Problem Statement
- Meta-Learning Approaches
  - Black-Box Adaptation
  - Non-Parametric Methods
- Applications

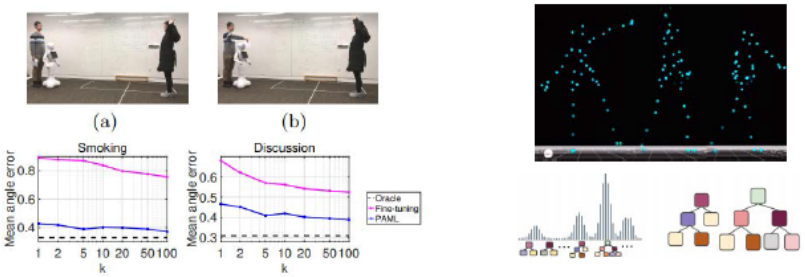
Applications in computer vision

few-shot image recognition



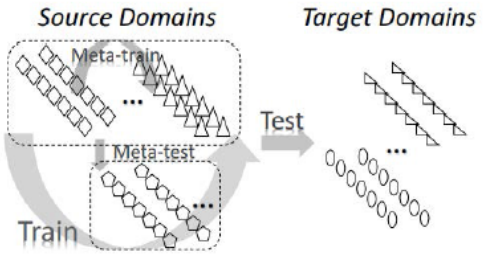
see, e.g.: Vinyals et al. **Matching Networks for One Shot Learning**, and many many others

human motion and pose prediction



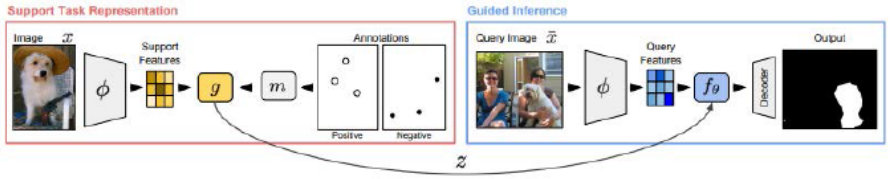
see, e.g.: Gui et al. **Few-Shot Human Motion Prediction via Meta-Learning.**
Alet et al. **Modular Meta-Learning.**

domain adaptation



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

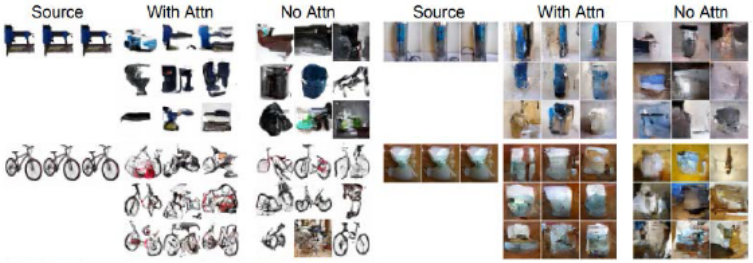
few-shot segmentation



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.**

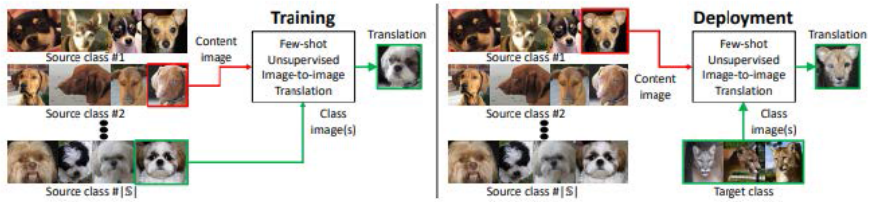
Applications in image & video generation

few-shot image generation



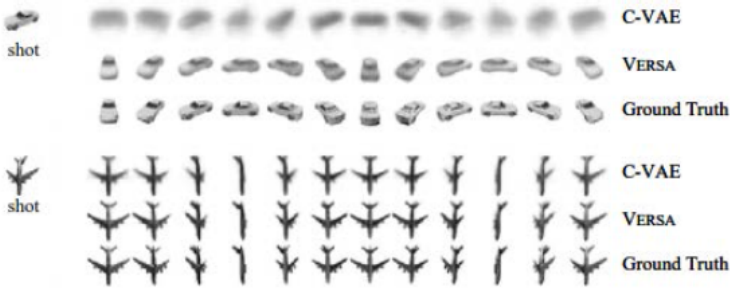
see, e.g.: Reed, Chen, Paine, van den Oord, Eslami, Rezende, Vinyals, de Freitas. **Few-Shot Autoregressive Density Estimation**. and many many others.

few-shot image-to-image translation



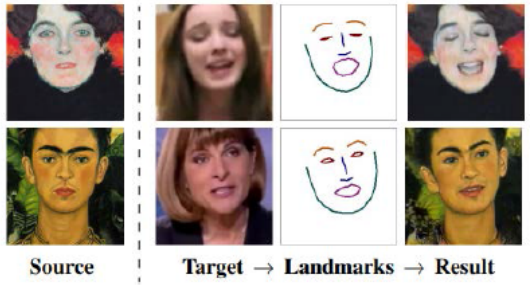
see, e.g.: Liu, Huang, Mallya, Karras, Aila, Lehtinen, Kautz. **Few-Shot Unsupervised Image-to-Image Translation.**

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**

Meta-Learning for ***Language***

Adapting to *new programs*

Meta Program Induction

Learn new program from a few I/O examples.

Devlin, Bunel* et al. NeurIPS '17*

Program Synthesis

Question:

How many CFL teams are from York College?

SQL:

```
SELECT COUNT CFL Team FROM
CFLDraft WHERE College = "York"
```

Result:

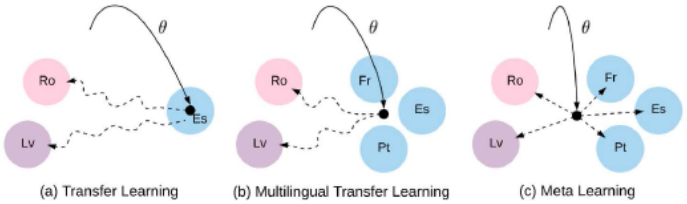
2

Construct pseudo-tasks with relevance function

Huang et al. NAACL '18

Adapting to *new languages*

Low-Resource Neural Machine Translation



(a) Transfer Learning (b) Multilingual Transfer Learning (c) Meta Learning

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

Gu et al. EMNLP '18

Learning *new words*

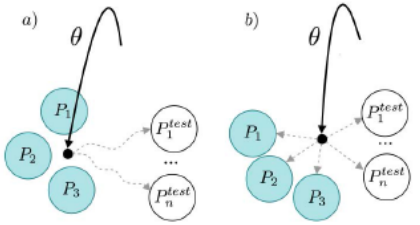
One-Shot Language Modeling

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*