Massachusetts Institute of Technology

Meta Learning

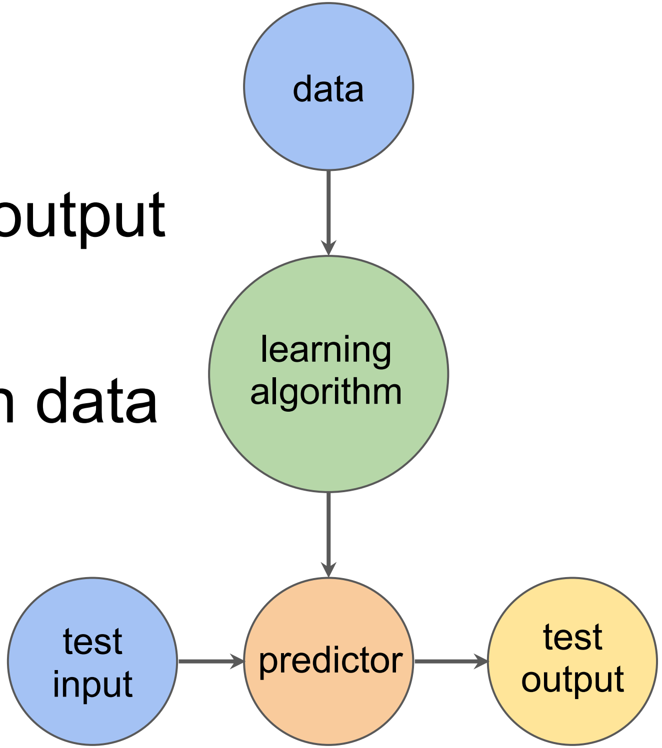
MIT
Iddo Drori, Fall 2020

Supervised Learning

- Data
- Learning algorithm for finding predictor
- For test input, predictor estimates test output

- Inference of predictor parameters given data

p(parameters | data)



Supervised Learning

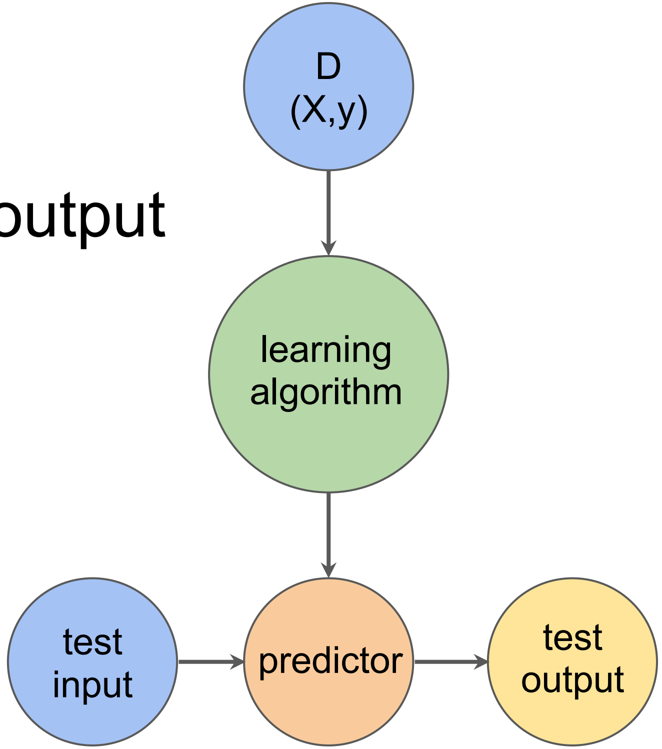
- Data
- Learning algorithm for finding predictor
- For test input, predictor estimates test output

- Inference of ϕ given D: $p(\phi \mid D)$

$\phi^* = \text{argmax}_\phi \log p(\phi \mid D)$

$= \text{argmax}_\phi \log p(D \mid \phi) + \log p(\phi)$

$= \text{argmax}_\phi \sum_i \log p(y_i \mid X_i, \phi) + \log p(\phi)$



Meta Learning

- Learn meta parameter $\boldsymbol{\theta}$ given meta training data D

$$p(\boldsymbol{\theta} \mid D)$$

$$\boldsymbol{\theta}^* = \text{argmax}_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta} \mid D)$$

Meta Learning for Few Shot Classification

- Adaptation

$$p(\phi \mid Dt, \boldsymbol{\theta}^*)$$

$$\phi^* = \text{argmax}_{\phi} \log p(\phi \mid Dt, \boldsymbol{\theta}^*)$$

Massachusetts Institute of Technology

Meta Learning for Few Shot Classification

Massachusetts Institute of Technology

Meta Learning for Few Shot Classification

- Training task 1: cats vs. dogs

cat

dog

1 2 3

?

Meta Learning for Few Shot Classification

- Training task 1: 2 way (classes), 3 shot (samples)

class 1

class 2

sample 1 sample 2 sample 3

?

Meta Learning for Few Shot Classification

- Training task 1: c = 2 classes, k = 3 samples

class 1

class 2

sample 1 sample 2 sample 3

?

Meta Learning for Few Shot Classification

- Training task 2: flower vs. bird

flower

bird

1 2 3

?

Meta Learning for Few Shot Classification

- Testing task: lion vs. monkey

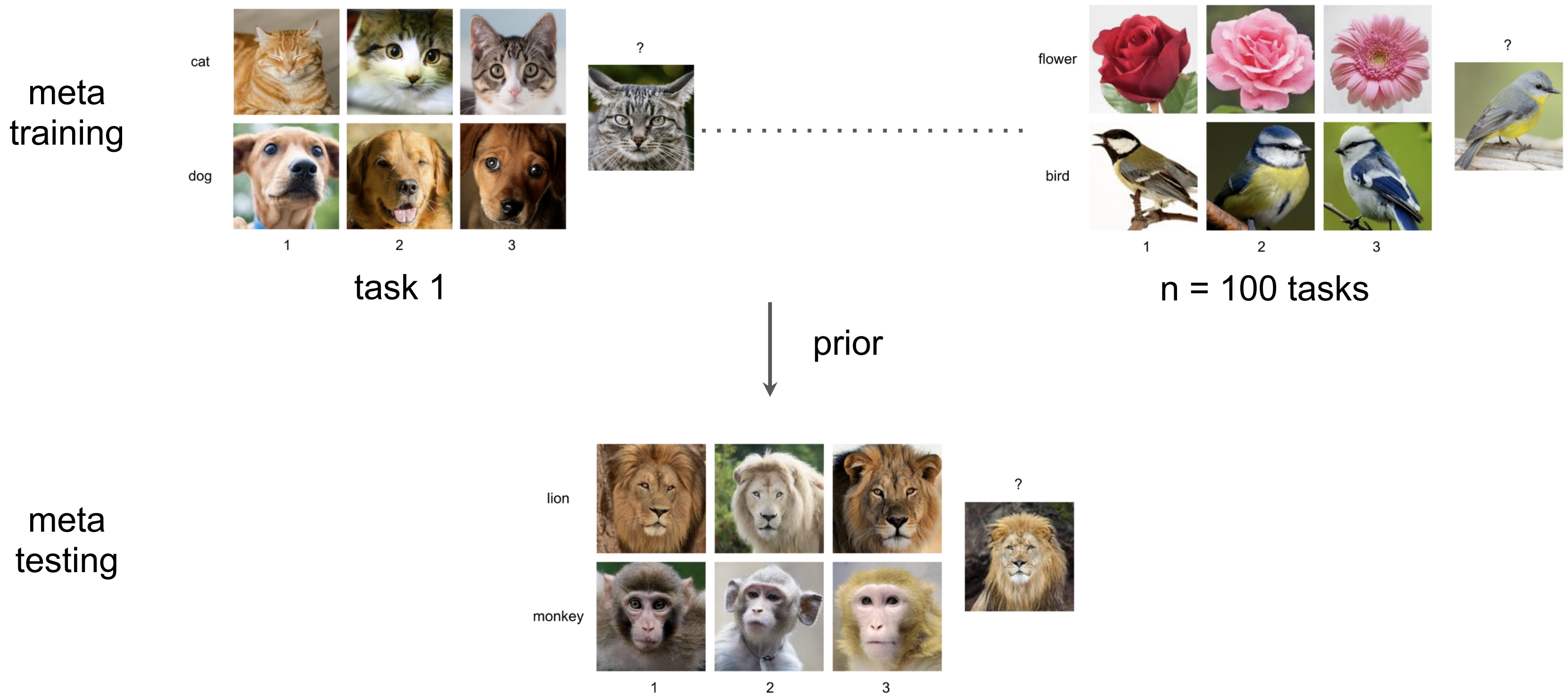
lion

monkey

1 2 3

?

Meta Learning for Few Shot Classification



Meta Learning for Few Shot Classification

- c classes
- k samples per class for training
- n tasks for meta training

Massachusetts Institute of Technology

Task Support and Query Sets

- For each task i with meta training dataset $D_i = Ds_i \cup Dq_i$
  - Training set Ds_i (support set)
  - Testing set Dq_i (query set)

task i

Dsi support set

Dqi query set

Meta Data

Dsi support set

Dqi query set

D_1

$D_{s1} \cup D_{q1}$

D_2

$D_{s2} \cup D_{q2}$

D_n

$D_{sn} \cup D_{qn}$

$$D = (D_1,...,D_n)$$

Meta Learning

D_1

$D_{s1} \cup D_{q1}$

D_2

$D_{s2} \cup D_{q2}$

D_n

$D_{sn} \cup D_{qn}$

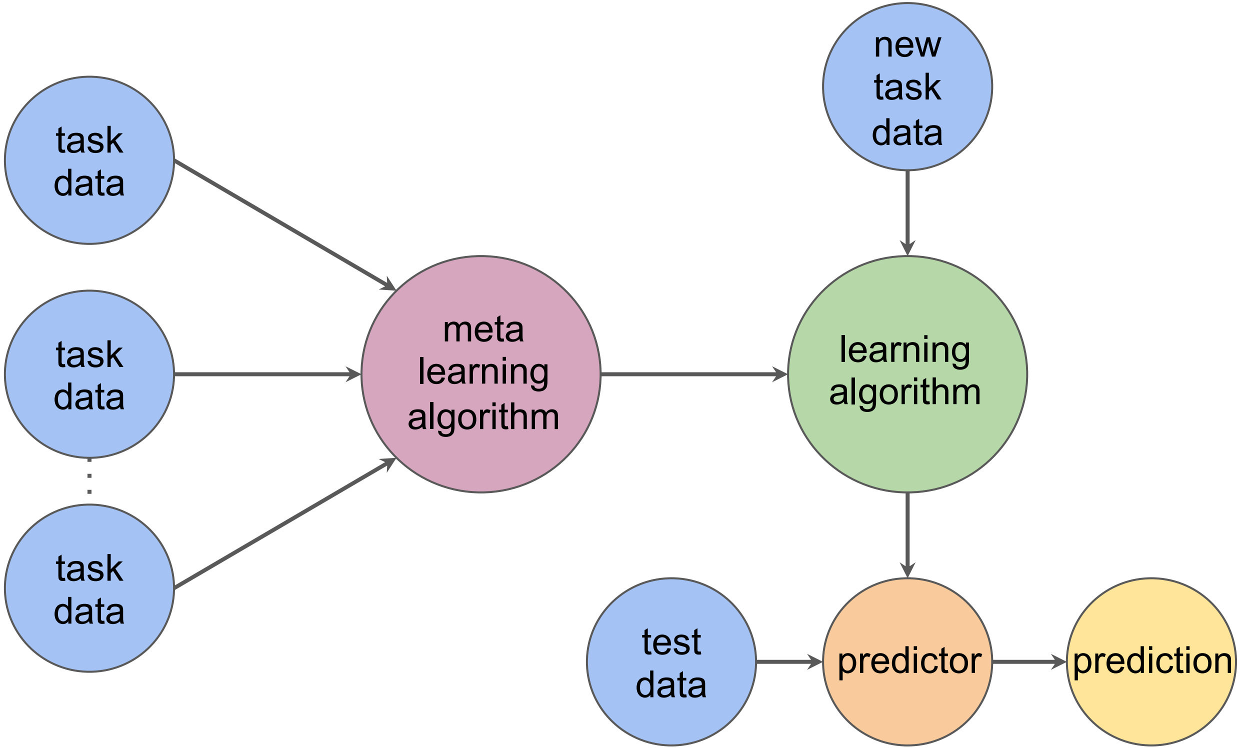
D_t

X_{test}

y_{test}

Meta Learning

task = data splits, priors



Black-Box Methods

Massachusetts Institute of Technology

Black-Box Meta Learning for Few Shot Classification

- Meta training data: $D = (D_1, \ldots, D_n)$

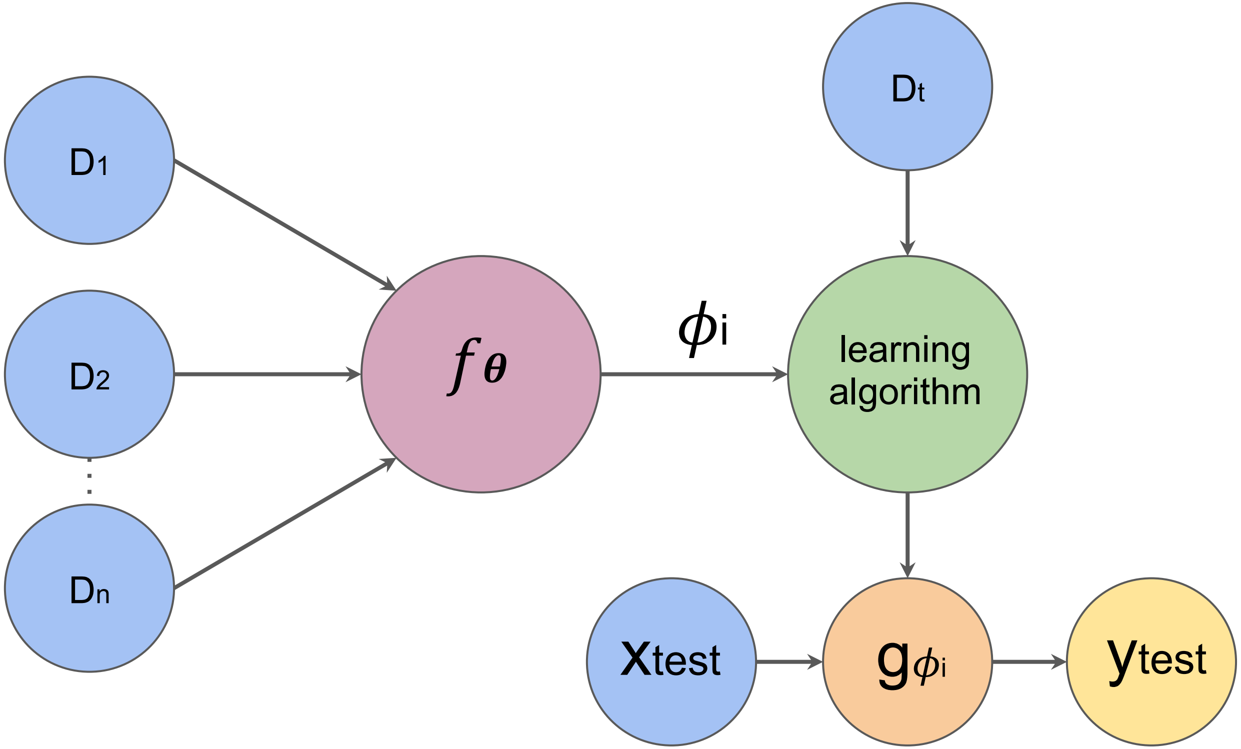
- Inference over task specific parameters ϕ_i given meta training dataset and meta parameters

$$p(\phi_i \mid Ds_i, \boldsymbol{\theta})$$

$$\max_{\boldsymbol{\theta}} \sum_i \log(\phi_i \mid Dq_i)$$

Black-Box Meta Learning for Few Shot Classification

task = data splits, priors



Meta Learning for Few Shot Classification

- $p(\phi_i \mid D_{s_i}, \boldsymbol{\theta})$
- Optimize $\boldsymbol{\theta}$ MLE using meta training dataset D
- Model as $p(\phi_i \mid D_{s_i}, \boldsymbol{\theta})$ as NN $f_{\boldsymbol{\theta}}$
- Meta NN $f_{\boldsymbol{\theta}}$ with input Di and output ϕ_i

$$\phi_i = f_{\boldsymbol{\theta}}(D_{s_i})$$

- Second task specific NN g with parameters ϕ_i computing

$$y_{test} = g_{\phi_i}(x_{test})$$

- $\max_{\boldsymbol{\theta}} \sum_i {}_{(X,y)\sim D_{q_i}} \log g_{\phi_i}(y|x)$
- $\max_{\boldsymbol{\theta}} \sum_i \mathcal{L}\,(f_{\boldsymbol{\theta}}(D_{s_i}), D_{q_i})$

Massachusetts Institute of Technology

Meta Learning Algorithm for Few Shot Classification

- Sample task i
- Sample task i dataset $D_i = D_{s_i} \cup D_{q_i}$:
  - Training set D_{s_i} (support set)
  - Testing set D_{q_i} (query set)
- Compute $\phi_i = f_{\boldsymbol{\theta}}(D_{s_i})$
- Update $\boldsymbol{\theta}$ by $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\phi_i, D_{q_i})$

Gradient-based Methods

Gradient-based Inference

- Meta model parameters $\boldsymbol{\theta}$ is a prior, model initialization
- For each task i: task adapted parameter ϕ_i

$$\max_{\boldsymbol{\theta}} \log p(Ds_i \mid \phi_i) + \log p(\phi_i \mid \boldsymbol{\theta})$$

Gradient-based Inference

- Meta model parameters $\boldsymbol{\theta}$ is a prior, model initialization
- For each task i: task adapted parameter ϕ_i
- Fine tuning
- Initialization with pre-trained parameters $\boldsymbol{\theta}$
  - CNN parameters trained on image dataset
  - Transformer parameters trained on text corpus
- Training data for new task Dt

$$\phi_i = \boldsymbol{\theta} - \boldsymbol{\alpha} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \mathrm{Dt})$$

Gradient-based Bi-Level Optimization

- Meta model parameters $\boldsymbol{\theta}$ is a prior, model initialization
- Optimize $\boldsymbol{\theta}$ across many tasks so fine tuning does well
- For each task i: task adapted parameter ϕ_i

$$\min_{\boldsymbol{\theta}} 1/n \sum_i \mathcal{L}_i(\phi_i, Dq_i)$$

$$\phi_i = \text{algorithm}(\boldsymbol{\theta}, Ds_i)$$

$$\min_{\boldsymbol{\theta}} 1/n \sum_i \mathcal{L}_i(\text{algorithm}(\boldsymbol{\theta}, Ds_i), Dq_i)$$

Model Agnostic Meta Learning (MAML)

- Meta training

$$\min_{\boldsymbol{\theta}} 1/n \sum_i \mathcal{L}_i(\phi_i, Dq_i)$$

$$\phi_i = \boldsymbol{\theta} - \boldsymbol{\alpha}\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, Ds_i)$$

$$\min_{\boldsymbol{\theta}} 1/n \sum_i \mathcal{L}_i(\boldsymbol{\theta} - \boldsymbol{\alpha}\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, Ds_i), Dq_i)$$

- Meta testing
- Ds: training data of new task
- $\boldsymbol{\theta}^*$: pre-trained parameters

$$\phi = \boldsymbol{\theta}^* - \boldsymbol{\alpha}\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, Ds)$$

Meta Algorithm

- Sample task i
- Sample task i dataset $D_i = Ds_i \cup Dq_i$:
  - Training set Ds_i (support set)
  - Testing set Dq_i (query set)
- Optimize $\phi_i = \boldsymbol{\theta} - \boldsymbol{\alpha}\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, Ds_i)$
- Update $\boldsymbol{\theta}$ by $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\phi_i, Dq_i) = \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta} - \boldsymbol{\alpha}\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, Ds_i), Dq_i)$

Gradient-based Meta Learning

- Meta training

$$\min_{\boldsymbol{\theta}} 1/n \sum_i \mathcal{L}_i(\phi_i, Dq_i)$$

- Update algorithm

$$\phi_i = \text{algorithm}(\boldsymbol{\theta}, Ds_i)$$

- Meta testing
- Ds: training data of new task
- $\boldsymbol{\theta}^*$: pre-trained parameters

$$\phi = \boldsymbol{\theta}^* - \boldsymbol{\alpha} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, Ds)$$

Gradient-based Meta Learning

- Meta training

$$\min_{\boldsymbol{\theta}} 1/n \sum_i \mathcal{L}_i(\phi_i, D_{q_i})$$

$$\phi_i = \text{algorithm}(\boldsymbol{\theta}, D_{s_i})$$

- MAML $= \boldsymbol{\theta} - \boldsymbol{\alpha}\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, D_s)$
- MetaSGD $= \boldsymbol{\theta} - \boldsymbol{\alpha}\text{diag}(w)\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, D_s)$
- Tnet $= \boldsymbol{\theta} - \boldsymbol{\alpha}\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, w, D_s)$
- Meta curvature $= \boldsymbol{\theta} - \boldsymbol{\alpha}B(\boldsymbol{\theta},w)\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, D_s)$
- Wrap-grad $= \boldsymbol{\theta} - \boldsymbol{\alpha}\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, w, D_s)$

Gradient-based Meta Learning

- Second order derivatives

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\phi, \mathrm{Dq_i})$$

$$\phi = \mathrm{algorithm}(\boldsymbol{\theta}, \mathrm{Ds})$$

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\mathrm{algorithm}(\boldsymbol{\theta}, \mathrm{Ds}), \mathrm{Dq_i})$$

$$d_{\boldsymbol{\theta}}\mathcal{L}(\phi, \mathrm{Dq_i}) = \nabla_{\boldsymbol{\theta}}\mathcal{L}(a, \mathrm{Dq_i})\big|_{a=\mathrm{algorithm}(\boldsymbol{\theta}, \mathrm{Ds})} d_{\boldsymbol{\theta}}\mathrm{algorithm}(\boldsymbol{\theta}, \mathrm{Ds})$$

Massachusetts Institute of Technology

Gradient-based Meta Learning

- Second order derivatives

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\phi, Dq_i)$$

$$\phi = \text{algorithm}(\boldsymbol{\theta}, Ds) = \boldsymbol{\theta} - \boldsymbol{\alpha}\, d_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, Ds)$$

$$d_{\boldsymbol{\theta}}\text{algorithm}(\boldsymbol{\theta}, Ds) = I - \boldsymbol{\alpha}\, d^2_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, Ds)$$

$$d_{\boldsymbol{\theta}}\mathcal{L}(\phi, Dq_i) = \nabla_{\boldsymbol{\theta}}\mathcal{L}(a, Dq_i)|_{a=u(\boldsymbol{\theta}, Ds)} d_{\boldsymbol{\theta}}\text{algorithm}(\boldsymbol{\theta}, Ds)$$

Gradient-based Meta Learning

- Second order derivatives

$$\min_{\boldsymbol{\theta}} 1/n \sum_i \mathcal{L}_i(\phi_i, Dq_i)$$

$$\phi_i = u(\boldsymbol{\theta}, Ds_i)$$

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}(f_\phi, Dq) = (I - \boldsymbol{\alpha}\, Hs(\boldsymbol{\theta}))gq(\phi)$$

- Reptile update for $\boldsymbol{\theta}$: $\boldsymbol{\theta} - \boldsymbol{\beta}\ 1/n\ (\boldsymbol{\theta} - \phi_i)$

Meta Learning for Few Shot Classification

- Why not take all meta training data together with meta testing data to learn a representation from all of them together?

- This may work better than other meta learning methods.

  Rethinking Few-Shot Image Classification: A Good Embedding Is All You Need?, Tian et al, 2020.

Metric-based Methods (non-parametric)

Metric-based Meta Learning

- Matching network
- Prototypical network
- Relation network
- GNN
- MetaOptNet

Naive Approach

- Compare Dq_i with each sample in Ds_i
- Label by nearest neighbor.
- Other methods?

Siamese Networks

- Are two samples from the same class?
- Training: pairwise comparisons of x_{test} with all Ds_i
- Binary classification
- Testing: one vs. many
- $\varphi(x_i,x_j) = ||\phi(x_i) - \phi(x_j)||$

Massachusetts Institute of Technology

Matching Network

- Training on multi-class classification
- Nearest neighbors at test time
- Learn an embedding at train time such that nearest neighbors at test time provides accurate predictions
- Meta training: learn g_θ and f_θ

Similarity score $f_\theta(x_{test}, x_k)$

$y_{test} = \sum_{(x_k, y_k) \text{ in } D_s} f_\theta(x_{test}, x_k) y_k$

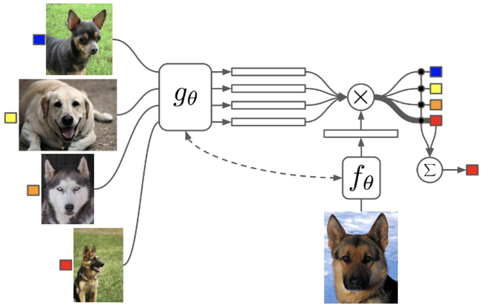


Figure source: Matching networks for one shot learning, Vinyals et al, 2016

Non-Parametric Meta Learning Algorithm

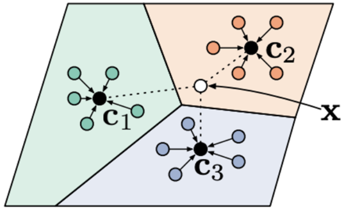
- Sample task i
- Sample task i dataset $D_i = Ds_i \cup Dq_i$:
  - Training set Ds_i (support set)
  - Testing set Dq_i (query set)
- Compute $y_{test} = \sum_{(x_k, y_k) \text{ in } Ds} f_{\boldsymbol{\theta}}(x_{test}, x_k) y_k$
- Update $\boldsymbol{\theta}$ by $\nabla_{\boldsymbol{\theta}} \mathcal{L}(y'_{test}, y_{test})$

Non-parametric, independent of ϕ

Massachusetts Institute of Technology

Prototypical Network

- Aggregate class information, prototypical for each class
- Embed each training image in each class and take mean
- Embed test image
- Embedding of data and nearest neighbors
- $c_k = 1/|D_{si}| \sum_{(x,y) \text{ in } D_{si}} f_\theta(x)$
- $p_\theta(y = k|x) = \exp(-d(f_\theta(x),c_k)) / \sum_k \exp(-d(f_\theta(x),c_k))$
- Euclidean or cosine distance



(a) Few-shot

Figure source: Prototypical networks for few-shot learning, Snell et al, 2017

Relation Network

- Instead of defining d (Euclidean or cosine), learn d
- Relation module

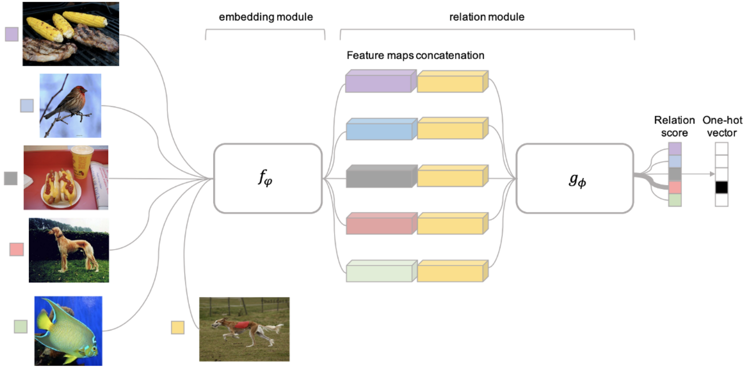


Figure source: Learning to compare: Relation network for few-shot learning, Sung et al, 2018

Massachusetts Institute of Technology

Graph neural network (GNN)

- Embedding using GNN

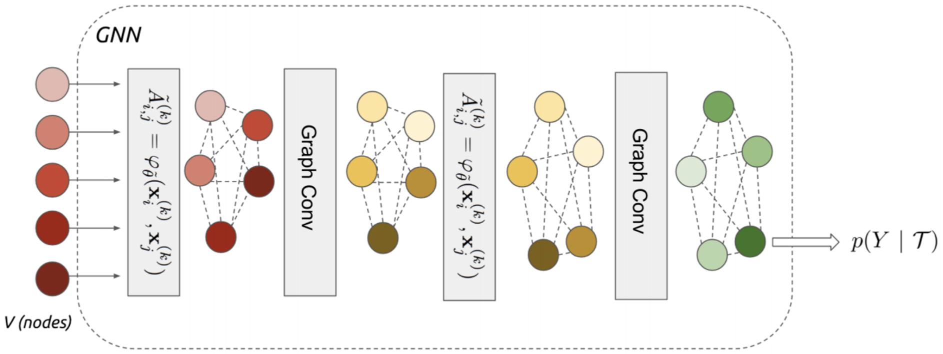


Figure source: Few-shot learning with graph neural networks, Garcia and Bruna, 2018

Massachusetts Institute of Technology

MetaOptNet

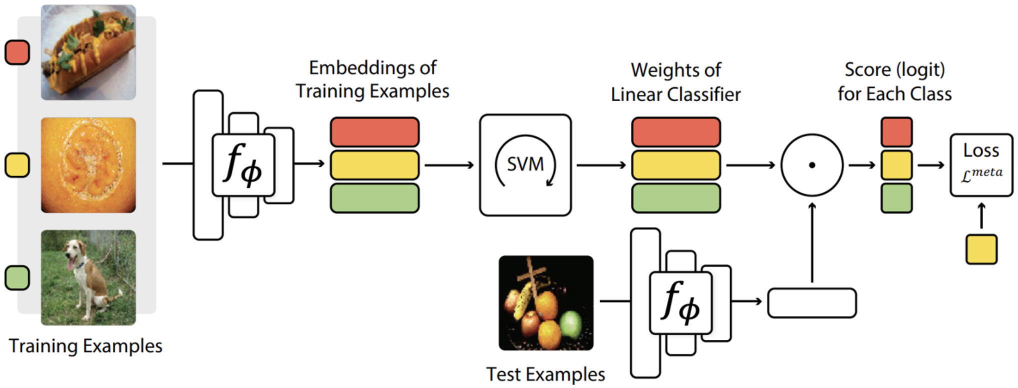


Figure source: Meta-learning with differentiable convex optimization, Lee et al, 2019

Comparison of Approaches

- Black-box: $y_{test} = f_{\theta}(D_{s_i}, x_{test})$

- Gradient-based (optimization): $y_{test} = f(D_{s_i}, x_{test})$
  $= f_{\phi_i}(x_{test})$ where $\phi_i = \boldsymbol{\theta} - \boldsymbol{\alpha} \nabla_{\theta} \mathcal{L}(\boldsymbol{\theta}, D_{s_i})$

- Metric-based (non-parametric): $y_{test} = f(D_{s_i}, x_{test}) = \mathrm{softmax}(-d\,(f_{\theta}(x), c_k))$, $c_k = 1/|D_{s_i}| \sum_{(x,y) \text{ in } D_{s_i}} f_{\theta}(x)$

Comparison of Approaches

- Black-box: data intensive

- Gradient-based (optimization): classification, regression, reinforcement learning; second order, computation intensive

- Metric-based (non-parametric): classification; simple feed forward; fast; dependent on distance metric

Meta Learning

MIT
Iddo Drori, Fall 2020

Massachusetts Institute of Technology