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
  \[ p(\text{parameters} \mid \text{data}) \]
Supervised Learning

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

Inference of \( \phi \) given \( D \): \( p(\phi | D) \)
\[
\phi^* = \arg\max_{\phi} \log p(\phi | D)
\]
\[
= \arg\max_{\phi} \log p(D | \phi) + \log p(\phi)
\]
\[
= \arg\max_{\phi} \sum_i \log p(y_i | X_i, \phi) + \log p(\phi)
\]
Meta Learning

• Learn meta parameter $\theta$ given meta training data $D$

\[ p(\theta \mid D) \]
\[ \theta^* = \text{argmax}_\theta \log p(\theta \mid D) \]
Meta Learning for Few Shot Classification

- Adaptation

\[
p(\phi \mid D_t, \theta^*) \\
\phi^* = \arg\max_{\phi} \log p(\phi \mid D_t, \theta^*)
\]
Meta Learning for Few Shot Classification
Meta Learning for Few Shot Classification

- Training task 1: cats vs. dogs
Meta Learning for Few Shot Classification

- Training task 1: 2 way (classes), 3 shot (samples)
Meta Learning for Few Shot Classification

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

![Images of cats and dogs](Image)

- 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 training

meta testing

n = 100 tasks

prior

task 1
Meta Learning for Few Shot Classification

• c classes
• k samples per class for training
• n tasks for meta training
Task Support and Query Sets

• For each task $i$ with meta training dataset $D_i = D_{si} U D_{qi}$
  – Training set $D_{si}$ (support set)
  – Testing set $D_{qi}$ (query set)
Meta Data

\[ D = (D_1, \ldots, D_n) \]
Meta Learning

- $D_1$, $D_{s1} \cup D_{q1}$
- $D_2$, $D_{s2} \cup D_{q2}$
- $D_n$, $D_{sn} \cup D_{qn}$
- $Dt$
- $X_{test}$
- $Y_{test}$
Meta Learning

task = data splits, priors

[Diagram showing the process of meta learning with data splits, priors, task data, meta learning algorithm, new task data, learning algorithm, test data, predictor, and prediction.]
Black-Box Methods
Black-Box Meta Learning for Few Shot Classification

• Meta training data: $D = (D_1,...,D_n)$

• Inference over task specific parameters $\phi_i$ given meta training dataset and meta parameters

$$p(\phi_i \mid D_{si}, \theta)$$

$$\max_{\theta} \sum_i \log(\phi_i \mid D_{qi})$$
Black-Box Meta Learning for Few Shot Classification

$\text{task} = \text{data splits, priors}$

$D_1 \rightarrow f_\theta \rightarrow \phi_i \rightarrow \text{learning algorithm} \rightarrow g_{\phi_i} \rightarrow y_{\text{test}}$

$D_2 \rightarrow f_\theta \rightarrow \phi_i \rightarrow \text{learning algorithm} \rightarrow g_{\phi_i} \rightarrow y_{\text{test}}$

$\vdots$

$D_n \rightarrow f_\theta \rightarrow \phi_i \rightarrow \text{learning algorithm} \rightarrow g_{\phi_i} \rightarrow y_{\text{test}}$
Meta Learning for Few Shot Classification

- \( p(\phi_i \mid Ds_i, \theta) \)
- Optimize \( \theta \) MLE using meta training dataset \( D \)
- Model as \( p(\phi_i \mid Ds_i, \theta) \) as NN \( f_\theta \)
- Meta NN \( f_\theta \) with input \( D_i \) and output \( \phi_i \)
- \( \phi_i = f_\theta(Ds_i) \)
- Second task specific NN \( g \) with parameters \( \phi_i \) computing
  \( y_{\text{test}} = g_{\phi_i}(x_{\text{test}}) \)
- \( \max_\theta \sum_i (X,y) \sim D_{qi} \log g_{\phi_i}(y \mid x) \)
- \( \max_\theta \sum_i \mathcal{L} (f_\theta(Ds_i), D_{qi}) \)
Meta Learning Algorithm for Few Shot Classification

- Sample task $i$
- Sample task $i$ dataset $D_i = D_{si} U D_{qi}$:
  - Training set $D_{si}$ (support set)
  - Testing set $D_{qi}$ (query set)
- Compute $\phi_i = f_\theta(D_{si})$
- Update $\theta$ by $\nabla_\theta L(\phi_i, D_{qi})$
Gradient-based Methods
Gradient-based Inference

- Meta model parameters $\theta$ is a prior, model initialization
- For each task $i$: task adapted parameter $\phi_i$

$$\max_\theta \log p(D_{Si} | \phi_i) + \log p(\phi_i | \theta)$$
Gradient-based Inference

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

$$\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, D_t)$$
Gradient-based Bi-Level Optimization

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

\[
\min_{\theta} \frac{1}{n} \sum_i L_i(\phi_i, D_{qi})
\]

\[
\phi_i = \text{algorithm}(\theta, D_{si})
\]

\[
\min_{\theta} \frac{1}{n} \sum_i L_i(\text{algorithm}(\theta, D_{si}), D_{qi})
\]
Model Agnostic Meta Learning (MAML)

- Meta training

\[
\min_\theta \frac{1}{n} \sum_i \mathcal{L}_i(\phi_i, D_{qi}) \\
\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, D_{si}) \\
\min_\theta \frac{1}{n} \sum_i \mathcal{L}_i(\theta - \alpha \nabla_\theta \mathcal{L}(\theta, D_{si}), D_{qi})
\]

- Meta testing
- Ds: training data of new task
- \(\theta^*\): pre-trained parameters

\[
\phi = \theta^* - \alpha \nabla_\theta \mathcal{L}(\theta, D_s)
\]
Meta Algorithm

- Sample task i
- Sample task i dataset $D_i = D_{si} U D_{qi}$:
  - Training set $D_{si}$ (support set)
  - Testing set $D_{qi}$ (query set)
- Optimize $\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, D_{si})$
- Update $\theta$ by $\nabla_\theta \mathcal{L}(\phi_i, D_{qi}) = \nabla_\theta \mathcal{L}(\theta - \alpha \nabla_\theta \mathcal{L}(\theta, D_{si}), D_{qi})$
Gradient-based Meta Learning

- Meta training
  \[
  \min_\theta \frac{1}{n} \sum_i \mathcal{L}_i(\phi_i, Dq_i)
  \]
- Update algorithm
  \[
  \phi_i = \text{algorithm}(\theta, Ds_i)
  \]
- Meta testing
- \( Ds \): training data of new task
- \( \theta^* \): pre-trained parameters
  \[
  \phi = \theta^* - \alpha \nabla_\theta \mathcal{L}(\theta, Ds)
  \]
Gradient-based Meta Learning

- Meta training
  \[
  \min_{\theta} \frac{1}{n} \sum_i \mathcal{L}_i(\phi_i, D_{qi})
  \]
  \[
  \phi_i = \text{algorithm}(\theta, D_{si})
  \]

- MAML
  \[
  = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, D_s)
  \]

- MetaSGD
  \[
  = \theta - \alpha \text{diag}(w) \nabla_\theta \mathcal{L}(\theta, D_s)
  \]

- Tnet
  \[
  = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, w, D_s)
  \]

- Meta curvature
  \[
  = \theta - \alpha B(\theta, w) \nabla_\theta \mathcal{L}(\theta, D_s)
  \]

- Wrap-grad
  \[
  = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, w, D_s)
  \]
Gradient-based Meta Learning

- Second order derivatives

\[
\min_\theta \mathcal{L}(\phi, Dq_i)
\]

\[
\phi = \text{algorithm}(\theta, Ds)
\]

\[
\min_\theta \mathcal{L}(\text{algorithm}(\theta, Ds), Dq_i)
\]

\[
d_\theta \mathcal{L}(\phi, Dq_i) = \nabla_\theta \mathcal{L}(a, Dq_i)|_{a=\text{algorithm}(\theta, Ds)}d_\theta \text{algorithm}(\theta, Ds)
\]
Gradient-based Meta Learning

- Second order derivatives

\[
\min_{\theta} \mathcal{L}(\phi, Dq_i) \\
\phi = \text{algorithm}(\theta, Ds) = \theta - \alpha \, d_{\theta}\mathcal{L}(\theta, Ds)
\]

\[
d_{\theta}\text{algorithm}(\theta, Ds) = I - \alpha \, d_{\theta}\mathcal{L}(\theta, Ds)
\]

\[
d_{\theta}\mathcal{L}(\phi, Dq_i) = \nabla_{\theta}\mathcal{L}(a, Dq_i)|_{a=u(\theta, Ds)}d_{\theta}\text{algorithm}(\theta, Ds)
\]
Gradient-based Meta Learning

- Second order derivatives
  \[
  \min_\theta \frac{1}{n} \sum_i \mathcal{L}_i(\phi_i, Dq_i)
  \]
  \[
  \phi_i = u(\theta, Ds_i)
  \]
  \[
  \nabla_\theta \mathcal{L}(f_{\phi}, Dq) = (I - \alpha Hs(\theta))gq(\phi)
  \]

- Reptile update for \(\theta\):
  \[
  \theta - \beta \frac{1}{n} (\theta - \phi_i)
  \]
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 $D_{q_i}$ with each sample in $D_{s_i}$
• Label by nearest neighbor.
• Other methods?
Siamese Networks

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

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

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

Figure source: Matching networks for one shot learning, Vinyals et al, 2016
Non-Parametric Meta Learning Algorithm

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

Non-parametric, independent of $\phi$
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
- $c_k = 1/|D_i| \sum_{(x,y) \in D_i} f_\theta(x)$
- $p_\theta(y = k|x) = \exp(-d(f_\theta(x),c_k)) / \sum_k \exp(-d(f_\theta(x),c_k))$
- Euclidean or cosine distance

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
Graph neural network (GNN)

- Embedding using GNN

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

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

- **Black-box:** $y_{test} = f\theta(Ds_i, x_{test})$

- **Gradient-based (optimization):** $y_{test} = f(Ds_i, x_{test})$
  
  $= f\phi_i(x_{test})$ where $\phi_i = \theta - \alpha \nabla \theta L(\theta, Ds_i)$

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