LGM-Net: Learning to Generate Matching Networks for Few-Shot Learning

Huaiyu Li, Weiming Dong, Xing Mei, Chongyang Ma, Feiyue Huang, Bao-Gang Hu

ICML, 2019
Presenter: Huaiyu Li
Email: lihuaiyu2014@ia.ac.cn
Motivation

• Training a DNN with SGD algorithm from random initialization
  • Overfitting when training data is scarce
  • Fitting well when training data is sufficient
  • Weights determine DNN functionality
  • Functional weights as a conditional distribution $P(\theta|S^{train})$

• Can we directly obtain functional weights of a DNN for a few-shot learning task?
  • Let’s learn a neural network $M$ to directly generate the weights $\theta$ for a neural network $T$ from just a few training samples.
  • e. g. $\theta = M(S^{train})$
The architecture of our LGM-Net for few-shot learning on 5-way 1-shot classification problems.

**Approach**

- **TargetNet Module (base-learner)**
  - A neural network with fixed architecture for classification

- **MetaNet Module (meta-learner)**
  - Encoding training samples and generating functional weights for TargetNet

- **Embedding Module**
  - Learnable neural network to extract low dimensional features
MetaNet Module (meta-learner)

- Task context encoder
  - To produce fixed-sized task context features

- Weight generator
  - To produce the weights of TargetNet based on task context features
  - With weight normalization on the generated weights

The architecture of our LGM-Net for few-shot learning on 5-way 1-shot classification problems.
TargetNet Module (base-learner)

- Use matching networks as the computing structure of TargetNet
- The weights of each layer are generated by MetaNet

The architecture of our LGM-Net for few-shot learning on 5-way 1-shot classification problems.
Learning Algorithm

**Algorithm 1** The training algorithm of LGM-Net for $N$-way $K$-shot problems

**Required:** Meta training dataset $D_{meta-train}$

**Required:** MetaNet $M$ with parameters $\phi$, TargetNet computational structure $T$ with parameter placeholder $\theta$.

Randomly initialize $\phi$

while not converged do

Sample a $N$-way $K$-shot task batch $T_{batch}$ from $D_{meta-train}$

for all the task instances in a batch do

Divide a task instance as $(S_{i}^{train}, S_{i}^{test}) = T_{i}$

Sample a functional weights point $\hat{\theta}$ for TargetNet from $M(S_{i}^{train})$

Assign generated weights $\hat{\theta}$ to TargetNet placeholder weights $\theta$

Compute TargetNet test loss for this task on $S_{i}^{test}$ as $L_{T_{i}}$

end for

Compute batch loss $L_{T_{batch}} = \sum T_{i} L_{T_{i}}$

Update $\phi$ using $\nabla_{\phi} L_{T_{batch}}$

end while

- Few-shot classification task episodic training
- Intertask normalization
  - To incorporate information across tasks in a task batch
Comparison

• Current meta-learning approaches:
  • Learning an initialization (Finn et al. 2017, ICML)
  • Learning an optimizer (Ravi & Larochelle. 2017, ICLR)
  • Learning a metric mapping function (Vinyals et al. 2016, NIPS)
  • others

• Our approach
  • Learning a conditional weight generator

• Advantages:
  • Neural weights are dynamically adapted to unseen tasks
  • Further fine-tuning is unnecessary
Results on Synthetic Datasets

Comparing the decision boundary of TargetNet with different weights

The weights generated by MetaNet contain prior knowledge for solving unseen tasks.
Evaluation

- Competitive performance on Omniglot
- STOA 1-shot learning performance on mini-ImageNet
- Ablation Study
  - Task context encoder and intertask normalization are important.
At the poster:
additional details, experiments and discussions

[Tue Jun 11\textsuperscript{th} 06:30—09:00 PM @Pacific Ballroom\#10]

Thanks!