7. OPTIMIZATION AS A MODEL FOR FEW-SHOT LEARNING

논문 : https://openreview.net/pdf?id=rJY0-Kcll

요약

- Optimization-based meta learning
- Learner가 모든 Task에서 빠르게 generalization할 수 있는 Initial parameter/optimize 기능을 Meta-Learner가 가지고 있는 형태
    - Meta-Learner : Long-term knowledge common among all the task (Learner의 Inital parameter 설정에 사용)
    - Learner : Shot-term knowledge within a task (Meta-Learner가 주는 parameter를 가지고 N-shot learning)
- Meta-Learner는 LSTM 기반으로 만들어짐
- 논문의 실험결과랑 달리 Matching Network와 성능이 비슷

문제제기

- few-shot learning에서 gradient-based optimization이 실패하는 이유
    - ADAM, Momentum, Adagrad 같은 Optimizer는 지정한 파라미터 업데이트 횟수만에 최적화되도록 설계돼있지 않음
    - Learner는 초기 parameter를 random initialization
        - global optima로 가는데 걸리는 횟수가 랜덤
        - pre-train/fine-tuning은 성능이 좋지만 pre-train에 사용한 task와 fine-tuning에 사용한 task가 다른경우 성능이 크게 감소
- 논문의 제안
    - LSTM-based meta-learner를 Optimizer로 사용하자
        - epoch마다 하나의 episode(그림 1)로 구성
        - 같은 episode를 T번 동안 meta-learner로 learner의 parameter update를 진행
        - episode 학습 후 learner의 Evaluation set에 대한 Loss로 Meta-Learner의 parameter update 진행

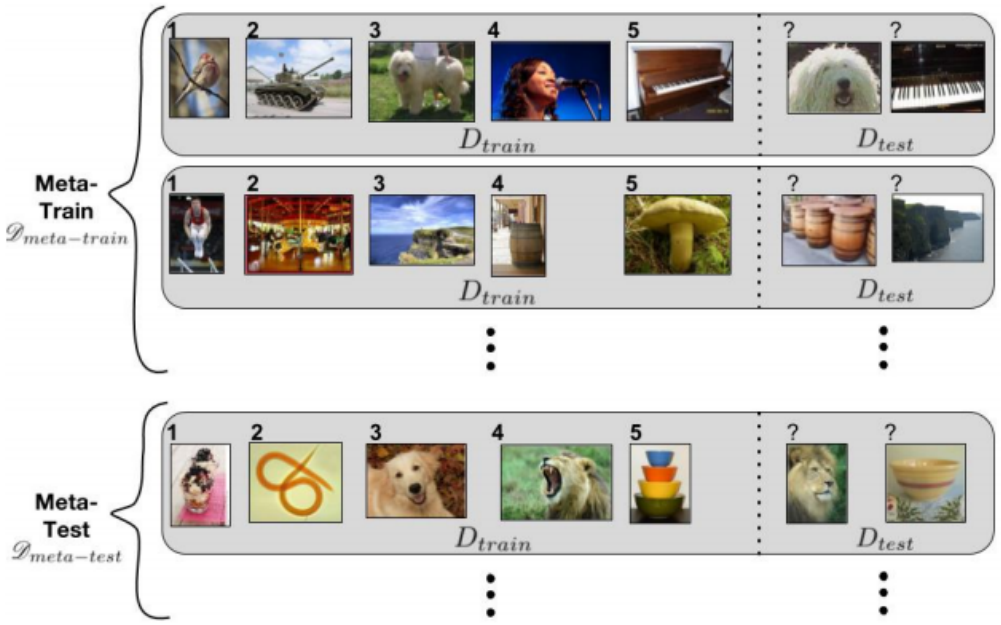

Figure 1: Example of meta-learning setup. The top represents the meta-training set $\mathscr{D}_{meta-train}$, where inside each gray box is a separate dataset that consists of the training set D_{train} (left side of dashed line) and the test set D_{test} (right side of dashed line). In this illustration, we are considering the 1-shot, 5-class classification task where for each dataset, we have one example from each of 5 classes (each given a label 1-5) in the training set and 2 examples for evaluation in the test set. The meta-test set $\mathscr{D}_{meta-test}$ is defined in the same way, but with a different set of datasets that cover classes not present in any of the datasets in $\mathscr{D}_{meta-train}$ (similarly, we additionally have a meta-validation set that is used to determine hyper-parameters).

모델 구조 (그림 2, 알고리즘 1)

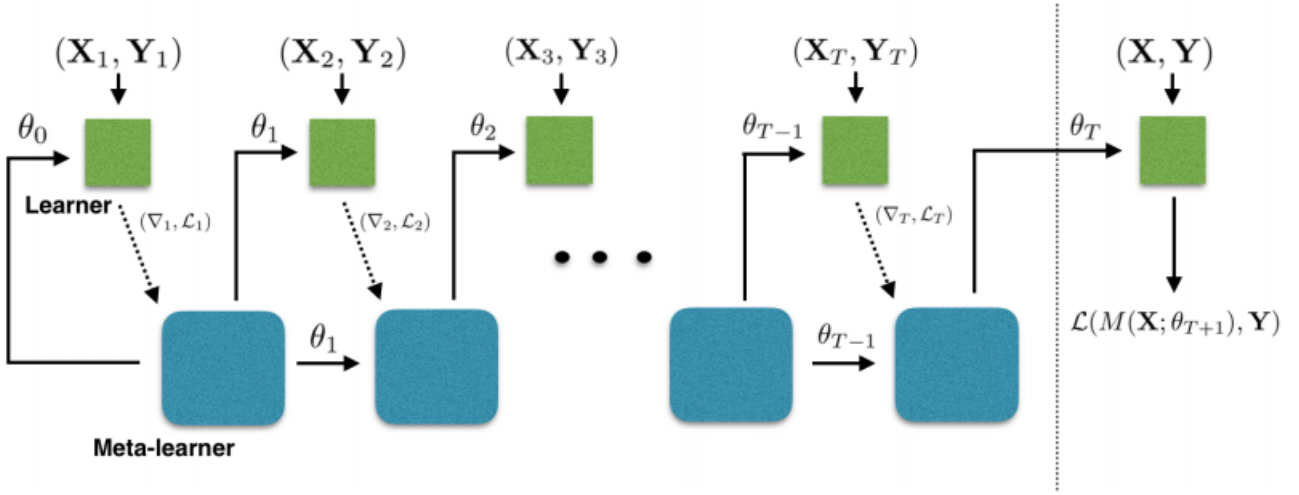

Figure 2: Computational graph for the forward pass of the meta-learner. The dashed line divides examples from the training set D_{train} and test set D_{test}. Each $(\mathbf{X}_i, \mathbf{Y}_i)$ is the i^{th} batch from the training set whereas $(\mathbf{X}, \mathbf{Y})$ is all the elements from the test set. The dashed arrows indicate that we do not back-propagate through that step when training the meta-learner. We refer to the learner as M, where $M(\mathbf{X}; \theta)$ is the output of learner M using parameters θ for inputs $\mathbf{X}$. We also use ∇_t as a shorthand for $\nabla_{\theta_{t-1}} \mathcal{L}_t$.

- 그림 2
  - 한 Episode에서 같은 데이터셋을 T번 학습하는 과정
  - Learner는 meta-learner를 통해 parameter update가 이뤄짐
- 일반적인 optimization algorithm은 식 1과 같음
  - $$\theta_t = \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} \mathcal{L}_t, \quad (1)$$
  - 시간 t에서의 학습은 이전 시간 t-1의 모델 파라미터 θ_{t-1}에 gradient of loss $\nabla_{\theta_{t-1}} \mathcal{L}_t$ 만큼 업데이트 하는것
- LSTM의 cell state update는 식 2와 같음
  - $$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \quad (2)$$
  - **기존의 LSTM이 forget gate f_t, input gate i_t, cell input activation vector $\tilde{c}_t$ 로 cell state를 업데이트할 때, $f_t = 1, c_{t-1} = \theta_{t-1}, i_t = \alpha_t,$ and $\tilde{c}_t = -\nabla_{\theta_{t-1}} \mathcal{L}_t.$ 로 치환해서 Learner의 파라미터를 업데이트하고자함**
  - Learner와 관련된것
    - cell state → learner의 parameter θ_t, cell input activation vector → gradient of loss $\nabla_{\theta_{t-1}} \mathcal{L}_t$
  - Meta-Learner가 기억하는것
    - input gate, fotget gate 위치의 값을 계산하기 위한 shallow model을 가짐
    - **input gate**
      - 현재 파라미터 θ_{t-1}에 대한 **learning rate conrol**에 사용됨
      - $$i_t = \sigma\left(\mathbf{W}_I \cdot \left[\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, i_{t-1}\right] + \mathbf{b}_I\right)$$
      - 현재 gradient $\nabla_{\theta_{t-1}} \mathcal{L}_t$, loss $\mathcal{L}_t$, 시간 t-1에서의 learner parameter θ_{t-1}와 input gate i_{t-1}를 concat
    - **forget gate**
      - 현재 파라미터 θ_{t-1}에 대한 **decay contol**에 사용됨
      - $$f_t = \sigma\left(\mathbf{W}_F \cdot \left[\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, f_{t-1}\right] + \mathbf{b}_F\right)$$
    - input gate와 forget gate의 파라미터는 small random weight로 초기화할 것
      - small learning rate를 갖고 normal gradient descent에 가깝게 시작하기 위함
      - fotget gate의 bias는 큰 값을 지정해서 forget gate값이 1에 가깝게 설정해서 decay를 하지 않는 형태로 학습

Algorithm 1 Train Meta-Learner

Input: Meta-training set $\mathscr{D}_{meta-train}$, Learner M with parameters θ, Meta-Learner R with parameters Θ.

$\Theta_0 \leftarrow$ random initialization

for $d = 1, n$ **do**
  $D_{train}, D_{test} \leftarrow$ random dataset from $\mathscr{D}_{meta-train}$
  $\theta_0 \leftarrow c_0$ ▷ Intialize learner parameters
  for $t = 1, T$ **do**
    $\mathbf{X}_t, \mathbf{Y}_t \leftarrow$ random batch from D_{train}
    $\mathcal{L}_t \leftarrow \mathcal{L}(M(\mathbf{X}_t; \theta_{t-1}), \mathbf{Y}_t)$ ▷ Get loss of learner on train batch
    $c_t \leftarrow R((\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t); \Theta_{d-1})$ ▷ Get output of meta-learner using Equation 2
    $\theta_t \leftarrow c_t$ ▷ Update learner parameters
  end for
  $\mathbf{X}, \mathbf{Y} \leftarrow D_{test}$
  $\mathcal{L}_{test} \leftarrow \mathcal{L}(M(\mathbf{X}; \theta_T), \mathbf{Y})$ ▷ Get loss of learner on test batch
  Update Θ_d using $\nabla_{\Theta_{d-1}} \mathcal{L}_{test}$ ▷ Update meta-learner parameters
end for

실험

- miniImageNet 데이터셋으로 기존 기법과 성능 비교 (표 1)
  - 성능이 더 좋다고 하나 Matching Network 논문에서의 성능과 본 논문의 실험결과와 차이가 있음

Model	5-class	
	1-shot	5-shot
Baseline-finetune	$28.86 \pm 0.54\%$	$49.79 \pm 0.79\%$
Baseline-nearest-neighbor	$41.08 \pm 0.70\%$	$51.04 \pm 0.65\%$
Matching Network	$\mathbf{43.40 \pm 0.78\%}$	$51.09 \pm 0.71\%$
Matching Network FCE	$\mathbf{43.56 \pm 0.84\%}$	$55.31 \pm 0.73\%$
Meta-Learner LSTM (OURS)	$\mathbf{43.44 \pm 0.77\%}$	$\mathbf{60.60 \pm 0.71\%}$

Table 1: Average classification accuracies on Mini-ImageNet with 95% confidence intervals. Marked in bold are the best results for each scenario, as well as other results with an overlapping confidence interval.

Table 2: Results on *mini*ImageNet.

Model	Matching Fn	Fine Tune	5-way Acc	
			1-shot	5-shot
PIXELS	Cosine	N	23.0%	26.6%
BASELINE CLASSIFIER	Cosine	N	36.6%	46.0%
BASELINE CLASSIFIER	Cosine	Y	36.2%	52.2%
BASELINE CLASSIFIER	Softmax	Y	38.4%	51.2%
MATCHING NETS (OURS)	Cosine	N	41.2%	56.2%
MATCHING NETS (OURS)	Cosine	Y	42.4%	58.0%
MATCHING NETS (OURS)	Cosine (FCE)	N	44.2%	57.0%
MATCHING NETS (OURS)	Cosine (FCE)	Y	**46.6%**	**60.0%**