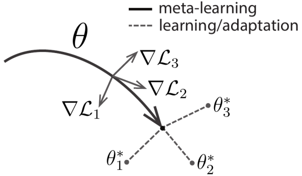
Meta-Learning with Latent Embedding Optimization

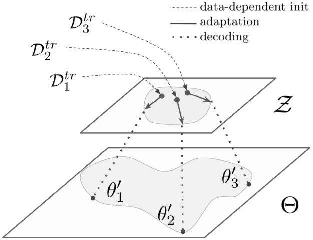
Rusu et al., ICLR 2019

Outline

- This paper model Latent Embedding Optimization (LEO), an extension over the MAML model
  - Learn a low-dimensional latent embedding of model parameters and performs optimization-based meta learning in this space
  - Provide 2 advantages over the MAML model
    - initial parameters for new tasks are conditioned on the training data, which enables a task-specific starting point for adaptation
    - optimizing in low-dimensional latent space is more efficient

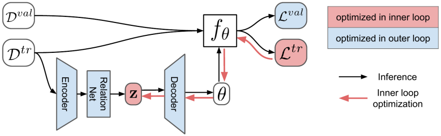
Comparision between LEO vs MAML





- The MAML model aims to find an set of parameters that can use for many different tasks
- The LEO model initializes task-dependent set of parameters, update them in a general framework, it is more desirable

LEO Algorithm and Arch



Algorithm 2 MAML for Few-Shot Supervised Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters

randomly initialize θ
while not done **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**
    Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
    Compute adapted parameters with gradient descent: $\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
    Sample datapoints $\mathcal{D}'_i = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$ for the meta-update
  end for
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$ using each $\mathcal{D}'_i$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
end while

Algorithm 1 Latent Embedding Optimization

Require: Training meta-set $\mathcal{S}^{tr} \in \mathcal{T}$
Require: Learning rates α, η

Randomly initialize ϕ_e, ϕ_r, ϕ_d
Let $\phi = \{\phi_e, \phi_r, \phi_d, \alpha\}$
while not converged **do**
  for number of tasks in batch **do**
    Sample task instance $\mathcal{T}_i \sim \mathcal{S}^{tr}$
    Let $\left(\mathcal{D}^{tr}, \mathcal{D}^{val}\right) = \mathcal{T}_i$
    Encode $\mathcal{D}^{tr}$ to $\mathbf{z}$ using g_{ϕ_e} and g_{ϕ_r}
    Decode $\mathbf{z}$ to initial params θ_i using g_{ϕ_d}
    Initialize $\mathbf{z}' = \mathbf{z}$, $\theta'_i = \theta_i$
    for number of adaptation steps **do**
      Compute training loss $\mathcal{L}^{tr}_{\mathcal{T}_i}\left(f_{\theta'_i}\right)$
      Perform gradient step w.r.t. $\mathbf{z}'$: $\mathbf{z}' \leftarrow \mathbf{z}' - \alpha \nabla_{\mathbf{z}'} \mathcal{L}^{tr}_{\mathcal{T}_i}\left(f_{\theta'_i}\right)$
      Decode $\mathbf{z}'$ to obtain θ'_i using g_{ϕ_d}
    end for
    Compute validation loss $\mathcal{L}^{val}_{\mathcal{T}_i}\left(f_{\theta'_i}\right)$
  end for
  Perform gradient step w.r.t ϕ: $\phi \leftarrow \phi - \eta \nabla_\phi \sum_{\mathcal{T}_i} \mathcal{L}^{val}_{\mathcal{T}_i}\left(f_{\theta'_i}\right)$
end while

LEO Model

- Encoding and Relation Network

$$\begin{aligned}\boldsymbol{\mu}_n^e, \boldsymbol{\sigma}_n^e &= \frac{1}{NK^2}\sum_{k_n=1}^{K}\sum_{m=1}^{N}\sum_{k_m=1}^{K} g_{\phi_r}\left(g_{\phi_e}\left(\mathbf{x}_n^{k_n}\right), g_{\phi_e}\left(\mathbf{x}_m^{k_m}\right)\right) \\ \mathbf{z}_n \sim q\left(\mathbf{z}_n|\mathcal{D}_n^{tr}\right) &= \mathcal{N}\left(\boldsymbol{\mu}_n^e, diag({\boldsymbol{\sigma}_n^e}^2)\right)\end{aligned}$$

  - Encoder all examples into intermediate codes, concatentated pair-wise
  - Use the relation network to learn specific code for each class, forming a Gaussian distribution
  - The hidden code z is drawn from the distribution

- Decoding Network

$$\begin{aligned}\boldsymbol{\mu}_n^d, \boldsymbol{\sigma}_n^d &= g_{\phi_d}\left(\mathbf{z}_n\right) \\ \mathbf{w}_n \sim p\left(\mathbf{w}|\mathbf{z}_n\right) &= \mathcal{N}\left(\boldsymbol{\mu}_n^d, diag({\boldsymbol{\sigma}_n^d}^2)\right)\end{aligned}$$

  - Decode the hidden codes into distribution's parameters, forming a Gaussian distribution
  - Task-specific parameters are drawn from the resulted distribution

LEO Model

- Inner-loop objective

$$\mathcal{L}^{tr}_{\mathcal{T}_i}\big(f_{\theta_i}\big) = \sum_{(\mathbf{x},y)\in\mathcal{D}^{tr}} \Big[-\mathbf{w}_y \cdot \mathbf{x} + \log\Big(\sum_{j=1}^{N} e^{\mathbf{w}_j \cdot \mathbf{x}}\Big)\Big]$$

- Outer-loop objective

$$\min_{\phi_e,\phi_r,\phi_d} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \Big[\mathcal{L}^{val}_{\mathcal{T}_i}\big(f_{\theta'_i}\big) + \beta D_{KL}\big(q(\mathbf{z}_n|\mathcal{D}^{tr}_n)||p(\mathbf{z}_n)\big) + \gamma||\text{stopgrad}(\mathbf{z}'_n) - \mathbf{z}_n||_2^2\Big] + R$$

Thank you !