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
**Algorithm 2** MAML for Few-Shot Supervised Learning

**Require:** \( p(T) \): distribution over tasks

**Require:** \( \alpha, \beta \): step size hyperparameters

1: randomly initialize \( \theta \)
2: while not done do
3: \hspace{0.5cm} Sample batch of tasks \( T_i \sim p(T) \)
4: \hspace{1cm} for all \( T_i \) do
5: \hspace{1.5cm} Sample \( K \) datapoints \( D = \{ x^{(j)}, y^{(j)} \} \) from \( T_i \)
6: \hspace{1.5cm} Evaluate \( \nabla_{\theta} L_{T_i}(f_{\theta}) \) using \( D \) and \( L_{T_i} \) in Equation (2) or (3)
7: \hspace{1.5cm} Compute adapted parameters with gradient descent: \( \theta'_i = \theta - \alpha \nabla_{\theta} L_{T_i}(f_{\theta}) \)
8: \hspace{1.5cm} Sample datapoints \( D'_i = \{ x^{(j)}, y^{(j)} \} \) from \( T_i \) for the meta-update
9: \hspace{1cm} end for
10: Update \( \theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{T_i \sim p(T)} L_{T_i}(f_{\theta'_i}) \) using each \( D'_i \) and \( L_{T_i} \) in Equation 2 or 3
11: end while

---

**Algorithm 1** Latent Embedding Optimization

**Require:** Training meta-set \( S^{tr} \in T \)

**Require:** Learning rates \( \alpha, \eta \)

1: Randomly initialize \( \phi_e, \phi_r, \phi_d \)
2: Let \( \phi = \{ \phi_e, \phi_r, \phi_d, \alpha \} \)
3: while not converged do
4: \hspace{0.5cm} for number of tasks in batch do
5: \hspace{1cm} Sample task instance \( T_i \sim S^{tr} \)
6: \hspace{1cm} Let \( (D^{tr}, D^{val}) = T_i \)
7: \hspace{1cm} Encode \( D^{tr} \) to \( z \) using \( g_{\phi_e} \) and \( g_{\phi_r} \)
8: \hspace{1cm} Decode \( z \) to initial params \( \theta_i \) using \( g_{\phi_d} \)
9: \hspace{1cm} Initialize \( z' = z, \theta'_i = \theta_i \)
10: \hspace{1cm} for number of adaptation steps do
11: \hspace{1.5cm} Compute training loss \( L_{T_i}^{tr}(f_{\theta'_i}) \)
12: \hspace{1.5cm} Perform gradient step w.r.t. \( z' \):
13: \hspace{1.75cm} \( z' \leftarrow z' - \alpha \nabla_{z'} L_{T_i}^{tr}(f_{\theta'_i}) \)
14: \hspace{1.5cm} Decode \( z' \) to obtain \( \theta'_i \) using \( g_{\phi_d} \)
15: \hspace{1cm} end for
16: \hspace{1cm} Compute validation loss \( L_{T_i}^{val}(f_{\theta'_i}) \)
17: \hspace{1cm} Perform gradient step w.r.t. \( \phi \):
18: \hspace{1.5cm} \( \phi \leftarrow \phi - \eta \nabla_{\phi} \sum_{T_i} L_{T_i}^{val}(f_{\theta'_i}) \)
19: \hspace{1cm} end for
20: \hspace{0.5cm} end for
21: end while
LEO Model

- **Encoding and Relation Network**

  \[ \mu^e_n, \sigma^e_n = \frac{1}{NK^2} \sum_{k_n=1}^K \sum_{m=1}^N \sum_{k_m=1}^K g_{\phi_r} \left( g_{\phi_e} (x_{n}^{k_n}), g_{\phi_e} (x_{m}^{k_m}) \right) \]

  \[ z_n \sim q (z_n | D_n^{tr}) = \mathcal{N} \left( \mu^e_n, \text{diag}(\sigma^e_n^2) \right) \]

  - Encoder all examples into intermediate codes, concatenated 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**

  \[ \mu^d_n, \sigma^d_n = g_{\phi_d} (z_n) \]

  \[ w_n \sim p (w | z_n) = \mathcal{N} \left( \mu^d_n, \text{diag}(\sigma^d_n^2) \right) \]

  - 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**

\[
\mathcal{L}_{T_i}^{tr}(f_{\theta_i}) = \sum_{(x, y) \in \mathcal{D}_{tr}} \left[ -w_y \cdot x + \log \left( \sum_{j=1}^{N} e^{w_j \cdot x} \right) \right]
\]

- **Outer-loop objective**

\[
\min_{\phi_c, \phi_r, \phi_d} \sum_{T_i \sim p(T)} \left[ \mathcal{L}_{T_i}^{val}(f_{\theta'}_i) + \beta D_{KL}(q(z_n | \mathcal{D}_{n}^{tr}) || p(z_n)) + \gamma \| \text{stopgrad}(z'_n) - z_n \|_2^2 \right] + R
\]
Thank you!