Learning to Few-Shot Learn Across Diverse Natural Language Classification Tasks

Trapit Bansal, Rishikesh Jha, Andrew McCallum
Motivation

- Fine-tuning on a new task still requires large amounts of task-specific labelled data
- Generalize to new tasks with few examples as a meta-learning problem
- Enables optimization-based meta-learning across tasks with different number of classes
Notation

Task: Episodic task (consist of support set and query set)

\[\{T_1, \ldots, T_M\}\]

Train: Support set \[D_{i}^{tr} \sim T_{i}\]

Validation: Query set \[D_{i}^{val} \sim T_{i}\]
Model-Agnostic Meta Learning (MAML)

Inner loop

$$\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_i(\theta, \mathcal{D}_i^{tr})$$

Outer loop

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{T_i \sim P(T)} \mathcal{L}_i(\theta'_i, \mathcal{D}_i^{val})$$
Generating Softmax Parameters

Task-dependent softmax parameters

\[ w_i^n, b_i^n = \frac{1}{|C_i^n|} \sum_{x_j \in C_i^n} g_{\psi}(f_{\theta}(x_j)) \quad \text{where} \quad C_i^n = \{x_j | y_j = n\}, \text{where} \ n \in [N_i] \]

Prediction for a new instance \( x^* \)

\[ p(y|x^*) = \text{softmax} \{W_i h_{\phi}(f_{\theta}(x^*)) + b_i\} \]
Learn to Adapt

- Task-agnostic: BERT lower layers
- Task-specific: BERT higher layers, softmax-params
- Inner loop: Update task-specific params

\[
\Phi^{(s+1)}_i = \Phi^{(s)}_i - \alpha_s \mathbb{E}_{D_i^{tr} \sim T_i} \left[ \nabla_{\Phi_i} \mathcal{L}_i(\{\Theta, \Phi_i\}, D_i^{tr}) \right]
\]

- Outer loop: update the task-agnostic params
  - Use first-order approximation for efficient computation

\[
\Theta = \theta_{\leq \nu} \cup \{\psi\}, \quad \Phi_i = \theta_{> \nu} \cup \{\phi, W_i, b_i\}
\]
Algorithm 1 LEOPARD

**Require:** set of $M$ training tasks and losses $\{(T_1, L_1), \ldots, (T_M, L_M)\}$, model parameters $\Theta = \{\theta, \psi, \alpha\}$, hyper-parameters $\nu, G, \beta$

Initialize $\theta$ with pre-trained BERT-base;

1: while not converged do
2:   # sample batch of tasks
3:   for all $T_i \in T$ do
4:       $D_{tr}^i \sim T_i$  # sample a batch of train data
5:       $C_{tr}^n_i \leftarrow \{x_j | y_j = n\}$  # partition data according to class labels
6:       $w_{tr}^i, b_{tr}^i \leftarrow \frac{1}{|C_{tr}^n_i|} \sum_{x_j \in C_{tr}^n_i} g_{\psi}(f_{\theta}(D_{tr}^i))$  # generate softmax parameters
7:       $W_i \leftarrow [w_1^i; \ldots; w_{N_i}^i]; \quad b_i \leftarrow [b_1^i; \ldots; b_{N_i}^i]$  # task-specific parameters
8:       $\Phi_i^{(0)} \leftarrow \theta_{\nu} \cup \{\phi, W_i, b_i\}$
9:   for $s := 0 \ldots G - 1$ do
10:      $D_{tr}^i \sim T_i$  # sample a batch of train data
11:      $\Phi_i^{(s+1)} \leftarrow \Phi_i^{(s)} - \alpha_s \nabla_{\Phi} L_i(\{\Theta, \Phi_i\}, D_{tr}^i)$  # adapt task-specific parameters
12:   end for
13:   $D_{val}^i \sim T_i$  # sample a batch of validation data
14:   $g_i \leftarrow \nabla_{\Theta} L_i(\{\Theta, \Phi_i^{(G)}\}, D_{val}^i)$  # gradient of task-agnostic parameters on validation
15:   end for
16:   $\Theta \leftarrow \Theta - \beta \cdot \sum_i g_i$  # optimize task-agnostic parameters
17: end while
Evaluation

● Training and Validation
  ○ GLUE benchmark

● Testing:
  ○ Entity typing: CoNLL-2003
  ○ Rating classification: Amazon Review
  ○ Text classification:
## Results

<table>
<thead>
<tr>
<th></th>
<th>(N)</th>
<th>(k)</th>
<th>(\text{BERT}_{\text{base}})</th>
<th>(\text{MT-BERT}_{\text{softmax}})</th>
<th>(\text{MT-BERT})</th>
<th>(\text{Proto-BERT})</th>
<th>(\text{LEOPARD})</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CoNLL</strong></td>
<td>4</td>
<td>8</td>
<td>50.44 ± 08.57</td>
<td>52.28 ± 4.06</td>
<td>55.63 ± 4.99</td>
<td>32.23 ± 5.10</td>
<td>54.16 ± 6.32</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>8</td>
<td>74.47 ± 03.10</td>
<td>71.67 ± 3.03</td>
<td>71.29 ± 3.30</td>
<td>33.75 ± 6.05</td>
<td>76.37 ± 3.08</td>
</tr>
<tr>
<td><strong>MITR</strong></td>
<td>4</td>
<td>8</td>
<td>49.38 ± 4.28</td>
<td>45.52 ± 5.90</td>
<td>50.49 ± 4.40</td>
<td>17.36 ± 2.75</td>
<td>49.84 ± 3.31</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>8</td>
<td>69.24 ± 3.68</td>
<td>66.09 ± 2.24</td>
<td>66.16 ± 3.46</td>
<td>16.41 ± 1.87</td>
<td>70.44 ± 2.89</td>
</tr>
</tbody>
</table>

## Text Classification

<table>
<thead>
<tr>
<th></th>
<th>(N)</th>
<th>(k)</th>
<th>(\text{BERT}_{\text{base}})</th>
<th>(\text{MT-BERT}_{\text{softmax}})</th>
<th>(\text{MT-BERT})</th>
<th>(\text{Proto-BERT})</th>
<th>(\text{LEOPARD})</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Airline</strong></td>
<td>4</td>
<td>8</td>
<td>42.76 ± 13.50</td>
<td>43.73 ± 7.86</td>
<td>46.29 ± 12.26</td>
<td>40.27 ± 8.19</td>
<td>54.95 ± 11.81</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>8</td>
<td>58.01 ± 08.23</td>
<td>58.79 ± 2.97</td>
<td>57.25 ± 09.90</td>
<td>48.73 ± 6.79</td>
<td>62.15 ± 05.56</td>
</tr>
</tbody>
</table>
Table 3: Ablations: \textsc{LEOPARD}_\nu does not adapt layers $0 - \nu$ (inclusive) in the inner loop, while \textsc{LEOPARD} adapts all parameters. Note that the outer loop still optimizes all parameters for all models. For new tasks (like entity typing) adapting all parameters is beneficial while for tasks seen at training time (like NLI) adapting fewer parameters is better. \textsc{LEOPARD-ZERO} is the model without the softmax-generator and a zero initialized softmax classifier which shows the importance of the softmax generator in \textsc{LEOPARD}. 

<table>
<thead>
<tr>
<th>$\nu$</th>
<th>Model</th>
<th>Entity Typing</th>
<th>Sentiment Classification</th>
<th>NLI</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>\textsc{LEOPARD} $\nu$</td>
<td>37.62 ± 7.37</td>
<td>58.10 ± 5.40</td>
<td>78.53 ± 1.55</td>
</tr>
<tr>
<td>16</td>
<td>\textsc{LEOPARD} 5 $\nu$</td>
<td>62.49 ± 4.23</td>
<td>71.50 ± 5.93</td>
<td>73.27 ± 2.63</td>
</tr>
<tr>
<td>16</td>
<td>\textsc{LEOPARD} $\nu$</td>
<td>69.00 ± 4.76</td>
<td>76.65 ± 2.47</td>
<td>76.10 ± 2.21</td>
</tr>
<tr>
<td>16</td>
<td>\textsc{LEOPARD-ZERO} $\nu$</td>
<td>44.79 ± 9.34</td>
<td>74.45 ± 3.34</td>
<td>74.36 ± 6.67</td>
</tr>
</tbody>
</table>