Learning to Propagate for Graph Meta-Learning

Lu Liu¹, Tianyi Zhou², Guodong Long¹, Jing Jiang¹, Chengqi Zhang¹

INTRODUCTION

We show that a meta-learner that explicitly relates tasks on a graph describing the relations of their output dimensions (e.g., classes) can significantly improve the performance of few-shot learning. This type of graph is usually free or cheap to obtain but has rarely been explored in previous works. We study the prototype-based few-shot classification, in which a prototype is generated for each class, such that the nearest neighbor search between the prototypes produces an accurate classification. We introduce "Gated Propagation Network (GPN)", which learns to propagate messages between prototypes of different classes on the graph, so that learning the prototype of each class benefits from the data of other related classes. In GPN, an attention mechanism is used for the aggregation of messages from neighboring classes, and a gate is deployed to choose between the aggregated messages and the message from the class itself. GPN is trained on a sequence of tasks from many-shot to few-shot generated by subgraph sampling. During training, it is able to reuse and update previously achieved prototypes from the memory in a lifelong learning cycle. In experiments, we change the training-test discrepancy and test task generation settings for thorough evaluations. GPN outperforms recent meta-learning methods on two benchmark datasets in all studied cases.

MODEL: GATED PROPAGATION NETWORK

An initial prototype for each class $y$ by averaging over all the K-shot samples belonging to class $y$ as in prototypical networks:

$$P^y = \frac{1}{|\mathcal{S}_y^y|} \sum_{(x_i, y_i) \in \mathcal{S}_y^y} f(x_i),$$

At step $t$, for each class $y$, we firstly compute the aggregated messages from its neighbors $\mathcal{N}_y$ by a dot-product attention module $a(p, q)$, i.e.,

$$P^{y+1}_{\mathcal{N}_y} = \sum_{\mathcal{N}_y} a(P^y_g, P^y_q) \times P^y_q : a[p, q] = \left| h_1(p), h_2(q) \right| \times \left| h_1(p) \times h_2(q) \right|.$$

Then we apply a gate $g$ to make decisions of whether accepting messages from its neighbors $P^y_{\mathcal{N}_y}$ or message from itself $P^y_g$, i.e.,

$$P^y_g = g \times P^y_g + (1 - g) \times P^y_{\mathcal{N}_y}.$$

To capture different types of relation and jointly use them for propagation, we aggregate k attention and gated propagation modules with untitled parameters

$$P^{y+1}_g = \frac{1}{k} \sum_{i=1}^k P^{y+1}_g[i].$$

The final prototype is given as the weighted sum of the initial prototype and the refined prototype:

$$P^y = \lambda \times P^0 + (1 - \lambda) \times P^7_g.$$

TRAINING STRATEGIES

Generating training tasks by subgraph sampling: random sampling and snowball sampling.

Random sampling captures strongly-related classes.

Snowball sampling captures weakly-related classes.

Building propagation pathways by training on maximum spanning trees.

Only propagate through the most related / close classes according to cosine similarity.

CURRICULUM LEARNING

Early stage: traditional supervised learning tasks

Later stage: train on few-shot learning tasks

EXPERIMENTAL RESULTS

Validation accuracy (mean±95% CI) on 600 test tasks achieved by GPN and baselines on tieredImageNet-Close with few-shot tasks generated by random sampling.

<table>
<thead>
<tr>
<th>Model</th>
<th>5Way1Shot</th>
<th>5Way5Shot</th>
<th>1Way1Shot</th>
<th>1Way5Shot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protonic Net [23]</td>
<td>42.87±1.60%</td>
<td>40.12±0.20%</td>
<td>34.64±3.76%</td>
<td>29.35±1.90%</td>
</tr>
<tr>
<td>GNN [6]</td>
<td>42.33±0.80%</td>
<td>59.17±0.69%</td>
<td>30.50±0.57%</td>
<td>44.33±0.72%</td>
</tr>
<tr>
<td>Closer Look [3]</td>
<td>35.60±1.53%</td>
<td>47.60±0.87%</td>
<td>21.58±0.96%</td>
<td>28.01±0.40%</td>
</tr>
<tr>
<td>PPN [12]</td>
<td>41.80±1.35%</td>
<td>60.04±0.97%</td>
<td>28.48±1.09%</td>
<td>48.66±0.70%</td>
</tr>
<tr>
<td>GPN</td>
<td>49.37±1.80%</td>
<td>64.14±1.00%</td>
<td>33.23±1.05%</td>
<td>50.50±0.70%</td>
</tr>
<tr>
<td>GPN+</td>
<td>56.54±1.67%</td>
<td>66.74±0.98%</td>
<td>34.74±1.06%</td>
<td>51.50±0.70%</td>
</tr>
</tbody>
</table>

See our paper for more results on tieredImageNet-Close and tieredImageNet-Far by different subgraph sampling strategies: random sampling and snowball sampling. We found 1) propagation is more effective between close classes and 2) propagation improves the performance both when discriminating between close classes (snowball sampling) and far classes (random sampling).