Introduction

Statistical heterogeneity can be detrimental for clients in FL

- need for personalization

Existing approaches are per-client

Need for a per-instance approach:

- instances that fall under the global data distribution can benefit more by using the global model

Methodology

Depending on the input it receives, Flow creates a personalized model that can dynamically make decisions on

(1) when to use a client’s local parameters, and

(2) when to use the global parameters.

Stages of Flow:

(a) Receive the global model $w^g$. Create a local model $w^l$ from $w^g$ by finetuning for single epoch.

(b) Create the personalized model which has $w^g$, $w^l$, and a dynamic policy $\theta$ for per-instance routing.

Shallow sequential models are computationally less intensive, and still allow dynamic routing temporally, hence we pick language tasks

- $r_t = \text{softmax}(\tau(f_{\theta}[x_t, h_{t-1}]))$ [prob. of picking routes $w^g$ and $w^l$]

- $h_t = [g(w^g, h_{t-1}; x_t); \ell(w^l, h_{t-1}; x_t)] \cdot r_t$ [updating hidden state with the mixture of global and local updates, based on the route probability]

- Beneficial for the newly joined clients with heterogeneous data, which has insufficient sample count for a near-optimal $w^l$.

- This improves inference performance of those instances which are not well-learned by the less generalizable local model.

Inference & Experiments

A client doesn’t persist any personalized/local states, since it’s impractical in cross-device FL

(1) Generates $w^l$ from $w^g$

(2) Trains routing policy $\theta$ on latest feature repr learned by $w^g$

We validate our approach against:

- No personalization: to understand $w^g$ behavior

- Per-client personalization: to see how much $w^l$ is better over $w^g$

- Stateful personalization: to observe the disadvantages of clients without $w^l$ or with stale $w^l$

Results

<table>
<thead>
<tr>
<th>Baseline</th>
<th>Stackoverflow</th>
<th>Reddit</th>
</tr>
</thead>
<tbody>
<tr>
<td>FedAvg</td>
<td>24.99%</td>
<td>17.39%</td>
</tr>
<tr>
<td>FedYogi †</td>
<td>27.90%</td>
<td>18.85%</td>
</tr>
<tr>
<td>APFL ‡</td>
<td>25.48%</td>
<td>18.48%</td>
</tr>
<tr>
<td>Ditto §</td>
<td>25.55%</td>
<td>18.74%</td>
</tr>
<tr>
<td>FedProx †</td>
<td>25.38%</td>
<td>17.58%</td>
</tr>
<tr>
<td>LG-FedAvg §</td>
<td>26.78%</td>
<td>18.59%</td>
</tr>
<tr>
<td>FedRecon (500 OOV) †</td>
<td>28.26%</td>
<td>18.89%</td>
</tr>
<tr>
<td>Dynamic Routing ‡</td>
<td>26.54%</td>
<td>18.70%</td>
</tr>
<tr>
<td>Flow ‡</td>
<td>29.16%</td>
<td>19.36%</td>
</tr>
</tbody>
</table>

- † = Server-side Optimization
- ‡ = Stateful Pers.
- § = Stateless Pers.
- † = Per-Client Pers.
- ‡ = Per-Instance Personalization
- Validation (below) and Test (left) accuracies for Reddit and Stackoverflow datasets respectively

Discussion

- FedYogi performs better than client-side personalization FedProx and LGFedAvg (both uses FedAvg as server-side aggregation) because of low statistical variance across clients

- Per-instance Dynamic Routing outperforms per-client FedProx and LGFedAvg, proving that we can seamlessly integrate local model with the global model, to outperform individual models

- FedRecon (using FedYogi as aggregation), outperforms stateful approaches since the latter require training of personalized models over multiple rounds, which is impractical when only ~5-6% of total clients participate

- Flow outperforms FedRecon since it is finer-grained in its choice of a client-specific local model and a generalized global model

References

1. Federated Reconstruction: Partially Local Federated Learning by Karan Singhal et al NeurIPS 2021
2. Neural Speed Reading via Skim-RNN by Minjoon Seo et al ICLR 2018
3. Federated Optimization in Heterogeneous Networks by Tian Li et al MLSys 2020