Extended Comparisons of Pyro and PyTorch

(Exploring some gaps in Pyro experiments)

Sarah Zhao
Motivation - Pyro
Bingham et al, 2019

- Expressive, scalable, and flexible PPL with minimal cognitive overhead
- Expressive generative model (models, guides, pyro.sample, pyro.param)
- SVI + PyTorch for scalability
- Modularity via Poutine library
- built on Python and PyTorch

```python
def model():
    loc, scale = torch.zeros(20), torch.ones(20)
    z = pyro.sample("z", Normal(loc, scale))
    u, b = pyro.param("weight"), pyro.param("bias")
    ps = torch.sigmoid(torch.mm(z, w) + b)
    return pyro.sample("x", Bernoulli(ps))

def conditioned_model(x):
    return pyro.condition(model, data={"x": x}())

optimizer = pyro.optim.Adam({"lr": 0.001})
loss = pyro.infer.Trace_ELBO()
svi = pyro.infer.SVI(model=conditioned_model,
                     guide=guide,
                     optimizer=optimizer,
                     loss=loss)
losses = []
for batch in batches:
    losses.append(svi.step(batch))
```
Pyro Experiments
Bingham et al, 2019

- Demonstrate performance in paper with VAEs and DMMs

<table>
<thead>
<tr>
<th># z</th>
<th># h</th>
<th>PyTorch (ms)</th>
<th>Pyro (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>400</td>
<td>3.82 ± 0.02</td>
<td>6.79 ± 0.04</td>
</tr>
<tr>
<td>30</td>
<td>400</td>
<td>3.73 ± 0.07</td>
<td>6.67 ± 0.03</td>
</tr>
<tr>
<td>10</td>
<td>2000</td>
<td>7.65 ± 0.02</td>
<td>10.14 ± 0.06</td>
</tr>
<tr>
<td>30</td>
<td>2000</td>
<td>7.66 ± 0.02</td>
<td>10.19 ± 0.03</td>
</tr>
</tbody>
</table>

- Since then, many more functionalities added but no direct comparisons with PyTorch implementations or PPLs
  - e.g. GPs, GP-LVM (and its variants), Bayesian Optimization
- Focus on **GPs** for now (and expand if time permits)
GPs in PyTorch
Gardner et al, 2018

- Standard PyTorch implementations
- GPyTorch (2018)
  - Sped-up, more modular implementation for GPs on PyTorch via BlackBox Matrix-Matrix (BBMM) inference
    - Modified conjugate gradient vs Cholesky implementations -> $\mathcal{O}(n^2)$ runtime for GP regression
  - There exists Pyro integration! But minimal discussion on when it should be used

https://www.youtube.com/watch?v=PZICg8r3Zhl&ab_channel=AIPursuitbyTAIR
Goal

A more comprehensive understanding of when to use Pyro vs. PyTorch vs. joint Pyro+PyTorch library
Goal

A more comprehensive understanding of when to use Pyro vs. PyTorch vs. joint Pyro+PyTorch library

(At least for some applications of GPs)
Work Plan

- Get acquainted with Pyro + PyTorch (and GPyTorch)
  - Re-implement experiments demonstrated in Pyro paper
  - Figure out data sets to test on for GP tasks
- Set-up GP regression experiments (exact, scalable/sparse, multi-task, etc.) for Pyro, PyTorch, GPyTorch, and GPyTorch + Pyro
- Run experiments + summarize run-time and test error results
- Write-up
Thanks!
Questions/Comments?