Propagating Gradients through Weights in Particle Filters

Adam Ścibior, Vaden Masrani, Frank Wood

{ascibior,vadmas,fwood}@cs.ubc.ca

Full version: Differentiable Particle Filtering without Modifying the Forward Pass

https://arxiv.org/abs/2106.10314

Problem

- Differentiating particle filters enables model learning
- Resampling steps are not reparameterizable
- Likelihood ratio method produces high variance
- Good score function estimators exist using Fisher's identity
- We show they can be obtained with automatic differentiation after a suitable correction to weights
- We also get good estimators for expectations under the posterior
- Correction is simple, cheap, and does not affect the forward pass

Usual autodiff-based score function estimator

$$\log \hat{Z}_{\mathrm{PF}} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \overline{w}_t^i + \nabla_\theta \log \hat{Z}_{\mathrm{PF}} =$$

$$\sum_{i=1}^{N} \sum_{t=1}^{T} \log \hat{Z} \; \nabla_\theta \log \overline{w}_t^i + \overline{w}_t^i \nabla_\theta \log p_\theta(x_t^i, y_t | x_{t-1}^{a_t^i})$$

high variance terms typically dropped

ignores dependencies across time and is not consistent by itself

Implementation

function PF(p, q, N, T)
- $x_0^i \sim p(x_0)$
- $w_0^i = \overline{w}_0^i = \frac{1}{N}$
- **for** $t \in 1:T$ **do**
  - **if** resampling condition **then**
    - $a_t^{1:N} \sim R(\overline{w}_{t-1}^{1:N})$
    - $\tilde{w}_t^i = \frac{1}{N}$
  - **else**
    - $a_t^i = i$
    - $\tilde{w}_t^i = \overline{w}_{t-1}^i$
  - $x_t^i \sim q_\phi(x_t | x_{t-1}^{a_t^i})$
  - $w_t^i = \tilde{w}_{t-1}^i \frac{p_\theta(x_t^i, y_t | x_{t-1}^{a_t^i})}{q_\phi(x_t^i | x_{t-1}^{a_t^i})}$
  - $W_t = \sum_{i=1}^N w_t^i$
  - $\overline{w}_t^i = w_t^i / W_t$
- $\hat{Z}_{\mathrm{PF}} = \prod_{t=1}^T W_t$

problematic sampling

Usual

function DPF(p, q, N, T)
- $x_0^i \sim p(x_0)$
- $v_0^i = \overline{v}_0^i = \frac{1}{N}$
- **for** $t \in 1:T$ **do**
  - **if** resampling condition **then**
    - $a_t^{1:N} \sim R(\perp(\overline{v}_{t-1}^{1:N}))$
    - $\tilde{v}_t^i = \frac{1}{N} \boxed{\overline{v}_{t-1}^{a_t^i} / \perp\left(\overline{v}_{t-1}^{a_t^i}\right)}$
  - **else**
    - $a_t^i = i$
    - $\tilde{v}_t^i = \overline{v}_{t-1}^i$
  - $x_t^i \sim q_\phi(x_t | x_{t-1}^{a_t^i})$
  - $v_t^i = \tilde{v}_{t-1}^i \frac{p_\theta(x_t^i, y_t | x_{t-1}^{a_t^i})}{q_\phi(x_t^i | x_{t-1}^{a_t^i})}$
  - $V_t = \sum_{i=1}^N v_t^i$
  - $\overline{v}_t^i = v_t^i / V_t$
- $\hat{Z}_{\mathrm{DPF}} = \prod_{t=1}^T V_t$

no gradients needed here

Ours

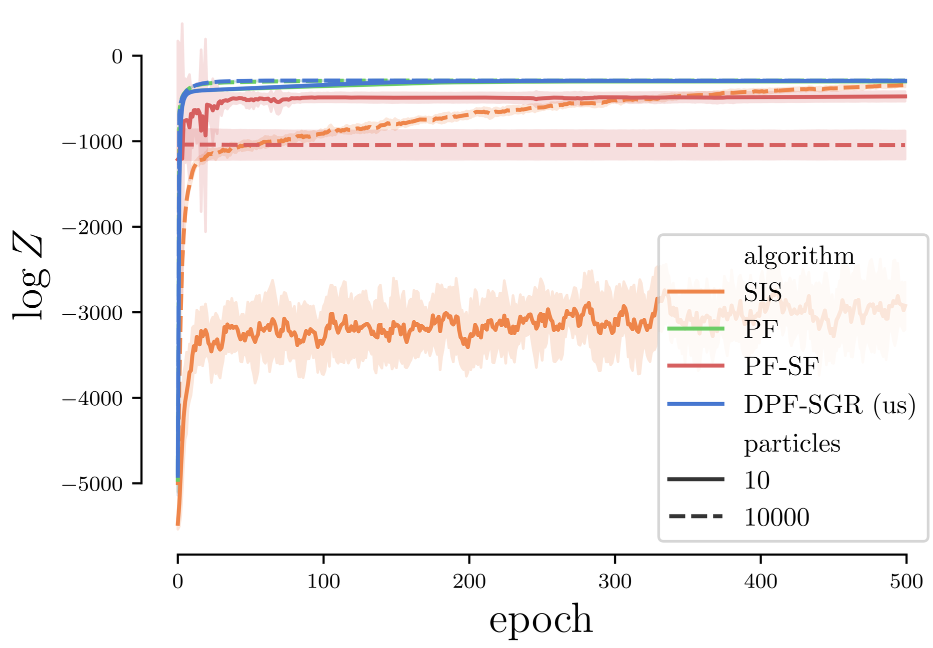
in PyTorch

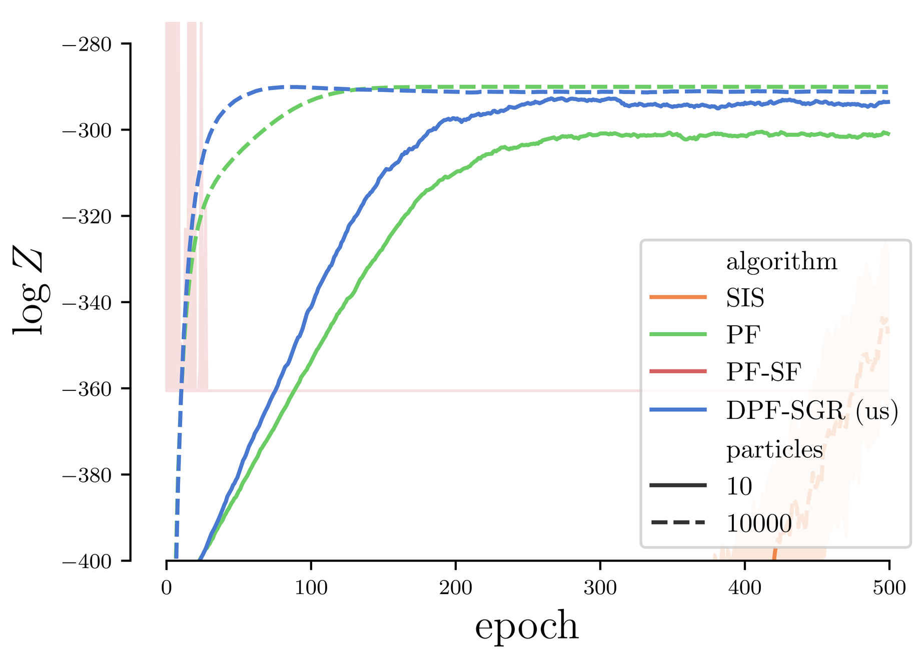
```
ancestral_logw = torch.gather(
    log_weights[-1], index=
    ancestral_indices[-1], dim=1)
logw += ancestral_logw -
    ancestral_logw.detach()
```

that's all it takes!

Results

LGSSM





Estimators

$$\overrightarrow{f(E_1, \ldots, E_n)} = f(\overrightarrow{E_1}, \ldots, \overrightarrow{E_2})$$
$$\overrightarrow{\perp(E)} = \overrightarrow{E}$$
$$\nabla \perp(E) = 0$$
$$\overrightarrow{\nabla E} = \nabla E \quad \text{if } \perp \notin E$$

$\perp$ stop-gradient

$\overrightarrow{\cdot}$ evaluation

stop-gradient calculus

$$\tilde{x}_{1:t}^i = (\tilde{x}_{1:t-1}^{a_t^i}, x_t^i)$$

ancestral lineage

$$\overrightarrow{\nabla_\theta \log \hat{Z}_{\mathrm{DPF}}} = \sum_{i=1}^N \overline{w}_T^i \nabla_\theta \log p_\theta(\tilde{x}_{1:T}^i, y_{1:T})$$

score function

$$\overrightarrow{\nabla_\theta \sum_{i=1}^N \overline{v}_T^i f_\theta(\tilde{x}_{1:T}^i)} = \sum_{i=1}^N \overline{w}_T^i \left(\nabla_\theta f_\theta(\tilde{x}_{1:T}^i) + \left(f_\theta(\tilde{x}_{1:T}^i) - \overline{f}_\theta\right) \nabla_\theta \log p_\theta(\tilde{x}_{1:T}^i, y_{1:T})\right)$$

expectation under posterior

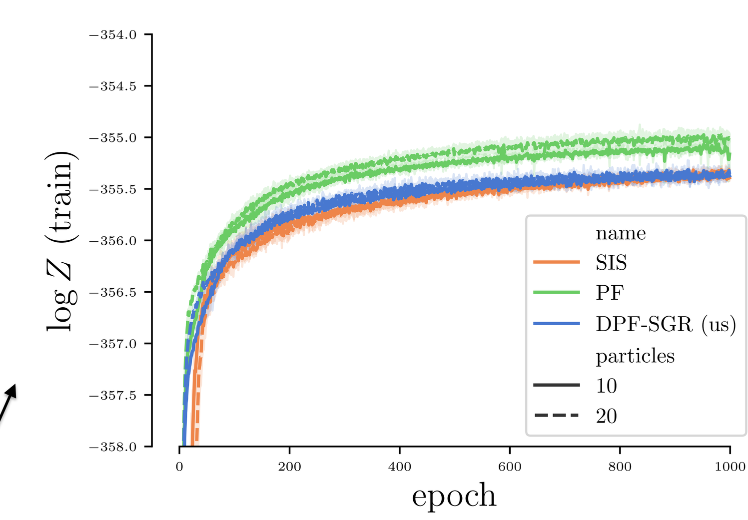
$$\overrightarrow{\nabla_\theta \hat{Z}_{\mathrm{DPF}} \sum_{i=1}^N \overline{v}_T^i f_\theta(\tilde{x}_{1:T}^i)} = \hat{Z} \sum_{i=1}^N \overline{w}_T^i \left(f_\theta(\tilde{x}_{1:T}^i) \nabla_\theta \log p_\theta(\tilde{x}_{1:T}^i, y_{1:T}) + \nabla_\theta f_\theta(\tilde{x}_{1:T}^i)\right)$$

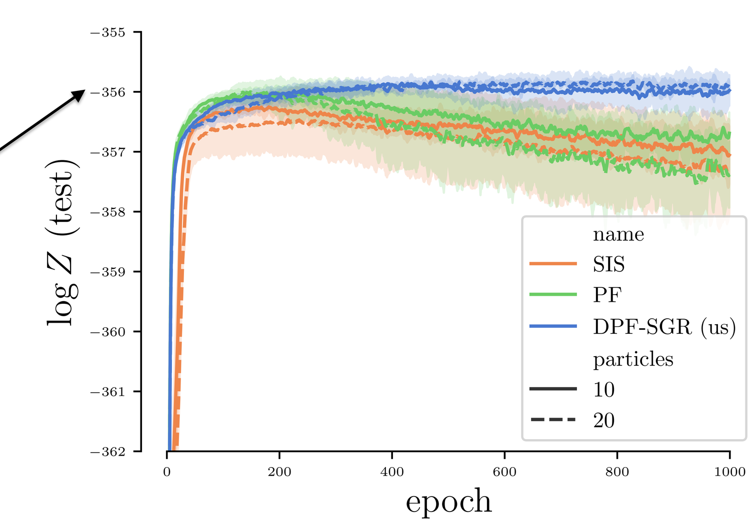
unbiased expectation under unnormalized posterior

Future Work

Proposal learning

VRNN





no overfitting with DPF why?

$$\overrightarrow{\nabla_\phi \log \hat{Z}_{\mathrm{DPF}}} = \sum_{i=1}^N \overline{w}_T^i \nabla_\phi \log \frac{p_\theta(\tilde{h}_\phi(\tilde{\epsilon}_{1:T}^i), y_{1:T})}{q_\phi(\tilde{h}_\phi(\tilde{\epsilon}_{1:T}^i))}$$

Integration theory

For all n, we have

$$\nabla_\theta^n \mathop{\mathbb{E}}_{x \sim p_\theta(x)} [f_\theta(x)] = \mathop{\mathbb{E}}_{x \sim p_\theta(x)} \left[\overrightarrow{\nabla_\theta^n \left(\frac{p_\theta(x)}{\perp(p_\theta(x))} f_\theta(x)\right)}\right]$$

Can we write

$$\mathop{\mathbb{E}}_{x \sim p_\theta(x)} [f_\theta(x)] = \mathop{\mathbb{E}}_{x \sim \perp(p_\theta(x))} \left[\frac{p_\theta(x)}{\perp(p_\theta(x))} f_\theta(x)\right]$$

and do stop-gradient importance sampling?

Maybe extend the calculus to include expectations

$$\overrightarrow{\mathop{\mathbb{E}}_{x \sim E_2} [E_1]} = \mathop{\mathbb{E}}_{x \sim \overrightarrow{E_2}} \left[\overrightarrow{E_1}\right]$$
$$\nabla \mathop{\mathbb{E}}_{x \sim E_2} [E_1] = \mathop{\mathbb{E}}_{x \sim E_2} [\nabla E_1] \quad \text{if } \nabla E_2 = 0$$

What compositional correctness properties can we guarantee?

THE UNIVERSITY OF BRITISH COLUMBIA

Partners

INVERTED AI

PLAI

Mila