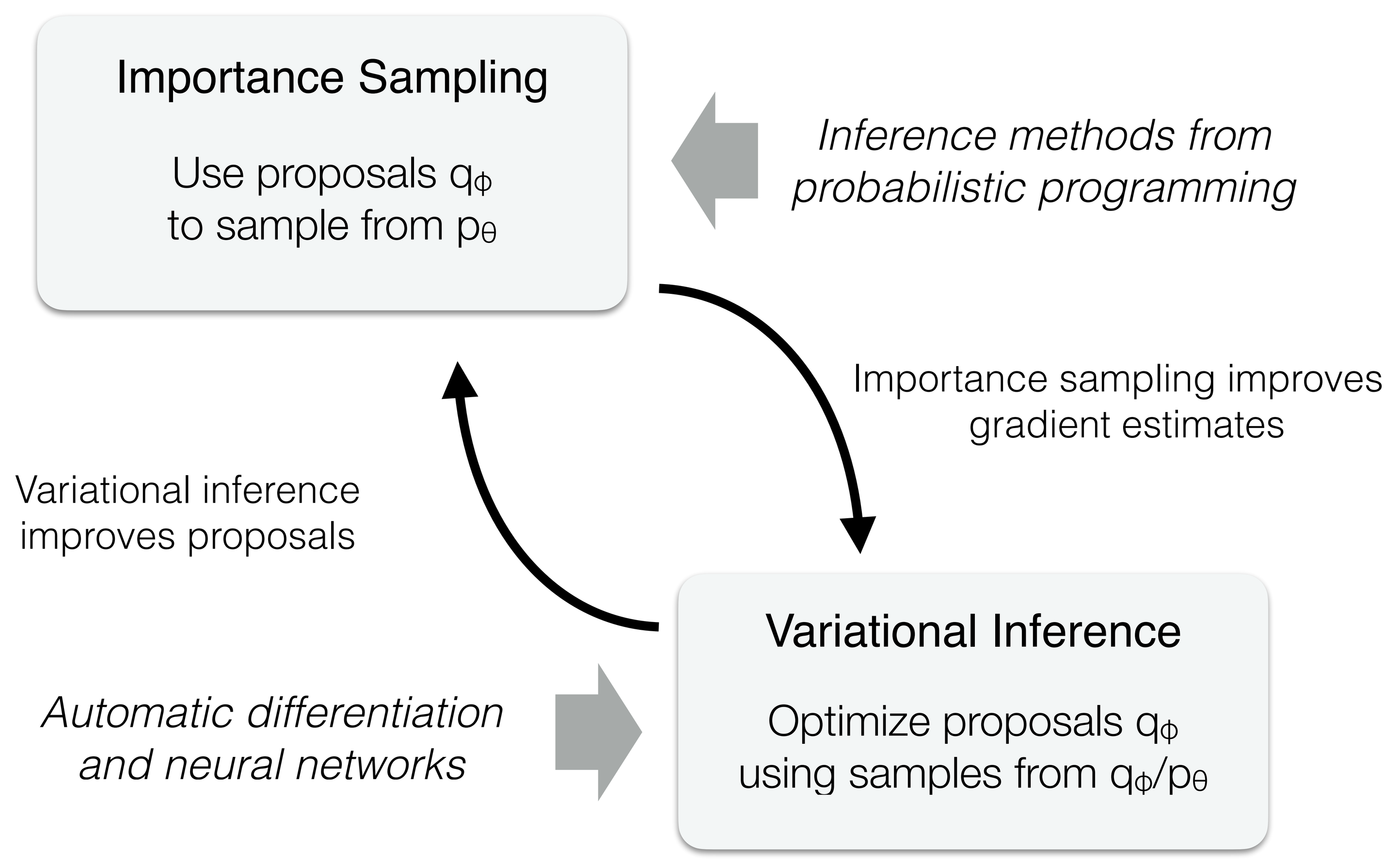
Learning Proposals for Probabilistic Programs with Inference Combinators

Sam Stites*[1] **Heiko Zimmermann*[2]** **Hao Wu[1]** **Eli Sennesh[1]** **Jan-Willem van de Meent[1,2]**

[1]Khoury College of Computer Sciences, Northeastern University, Massachusetts, USA [2]AMLab, University of Amsterdam, Amsterdam, The Netherlands

1 Importance Sampling and Variational Inference



2 Properly Weighted Program Evaluations

Denotational Semantics. *We assume the existence of denotational semantics, which for a program* f, *define a prior and unnormalized density*

$$[\![\mathtt{f}(c')]\!]_\gamma(\tau) = \gamma_f(\tau; c'), \qquad [\![\mathtt{f}(c')]\!]_p(\tau) = p_f(\tau; c').$$

Inference. *Given the unnormalized density, we want to approximate the corresponding normalized density*

$$\pi_f(\tau; c') = \frac{\gamma_f(\tau; c')}{Z_f(c')}, \qquad Z_f(c') = \int d\tau \, \gamma_f(\tau; c').$$

Definition. *A evaluation* $c, \tau, \rho, w \leftarrow\!\!\sim \mathtt{q}(c')$ *is strictly properly weighted for an unnormalized density* $\gamma_q \equiv Z_q \pi_q$, *if for all measurable functions* h

$$\mathbb{E}_{\mathtt{q}(c')}[w \, h(\tau)] = Z_q(c') \, \mathbb{E}_{\pi_q(\cdot; c')}[h(\tau)].$$

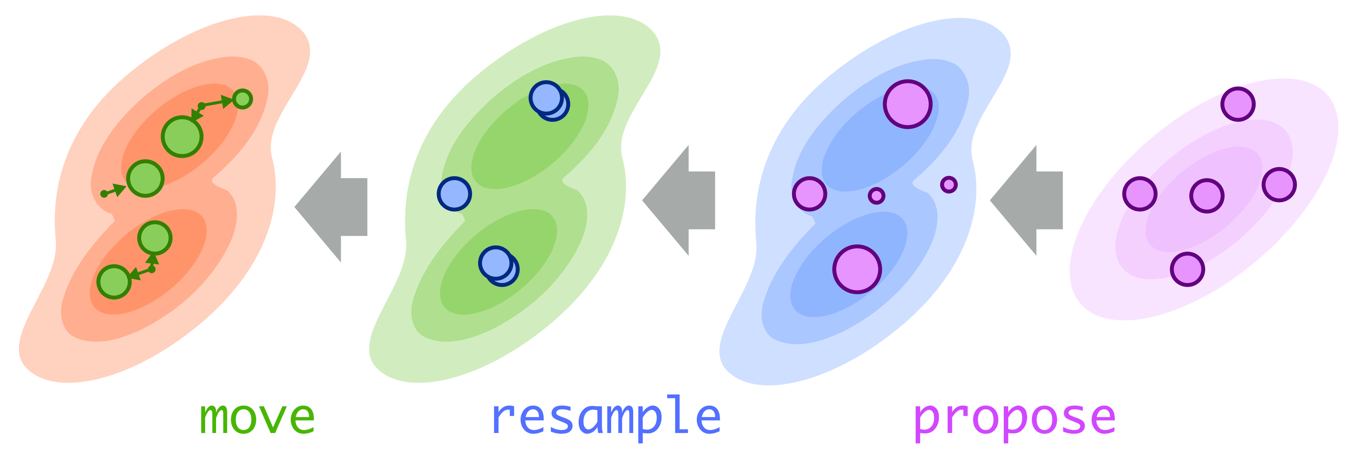
Sampler (can be a black box)

Constant of proportionality (marginal likelihood)

Density of interest (program posterior)

Quantity of interest (return value of program)

3 A DSL for importance sampling



Operational Semantics. *We propose a grammar for composing importance samplers and develop inference rules which guarantee that these samplers are valid by construction.*

f ::= A primitive program

p ::= f | extend(p, f)

q ::= p | resample(q) | compose(q′, q) | propose(p, q)

Theorem. *Evaluation of an inference program* $\mathtt{q}(c)$ *is strictly properly weighted for its unnormalized density* $[\![\mathtt{q}(c)]\!]_\gamma$.

$$\frac{\begin{array}{c} c_1, \tau_1, \rho_1, w_1 \leftarrow\!\!\sim \mathtt{q}(c_0) \qquad c_2, \tau_2, \rho_2, w_2 \leftarrow\!\!\sim \mathtt{p}(c_0)[\tau_1] \\ c_3, \tau_3, \rho_3, w_3 \leftarrow\!\!\sim \mathtt{marginal(p)}(c_0)[\tau_2] \\ u_1 = \prod_{\alpha \in \mathrm{dom}(\rho_1) \setminus (\mathrm{dom}(\tau_1) \setminus \mathrm{dom}(\tau_2))} \rho_1(\alpha) \end{array}}{c_3, \tau_3, \rho_3, \; w_2 \cdot w_1 / u_1 \leftarrow\!\!\sim \mathtt{propose(p,\ q)}(c_0)}$$

$$\frac{\begin{array}{c} c_1, \tau_1, \rho_1, w_1 \leftarrow\!\!\sim \mathtt{q_1}(c_0) \quad c_2, \tau_2, \rho_2, w_2 \leftarrow\!\!\sim \mathtt{q_2}(c_1) \\ \mathrm{dom}(\rho_1) \cap \mathrm{dom}(\rho_2) = \emptyset \end{array}}{c_2, \tau_2 \oplus \tau_1, \rho_2 \oplus \rho_1, \; w_2 \cdot w_1 \leftarrow\!\!\sim \mathtt{compose(q_2,\ q_1)}(c_0)}
\qquad
\frac{\begin{array}{c} c_1, \tau_1, \rho_1, w_1 \leftarrow\!\!\sim \mathtt{p}(c_0) \quad c_2, \tau_2, \rho_2, w_2 \leftarrow\!\!\sim \mathtt{f}(c_1) \\ \mathrm{dom}(\rho_1) \cap \mathrm{dom}(\rho_2) = \emptyset \quad \mathrm{dom}(\rho_2) = \mathrm{dom}(\tau_2) \end{array}}{c_2, \tau_1 \oplus \tau_2, \rho_1 \oplus \rho_2, w_1 \cdot w_2 \leftarrow\!\!\sim \mathtt{extend(p,\ f)}(c_0)}
\qquad
\frac{\vec{c}_2, \vec{\tau}_2, \vec{\rho}_2 = \mathrm{REINDEX}(\vec{a}_1, \vec{c}_1, \vec{\tau}_1, \vec{\rho}_1) \quad \vec{w}_2 = \mathrm{MEAN}(\vec{w}_1)}{\vec{c}_2, \vec{\tau}_2, \vec{\rho}_2, \vec{w}_2 \leftarrow\!\!\sim \mathtt{resample(q)}(\vec{c}_0)}$$

4 Learning Neural Proposals

Optimization. *We optimize the parameters of the neural proposals by optimizing a divergence or divergence-based stochastic bound* $\mathscr{D}$ *at each level of nesting (propose statement).*

$$\mathscr{D}(\theta, \phi) = D_1(p_1 \,||\, \pi_1) + \sum_{k=2}^{K} D_k(\pi_{k-1} \,||\, \pi_k)$$

Initial proposal program

Intermediate IS targets

Final IS targets

5 Example: Amortized Population Gibbs (APG) Samplers

Writing inference programs with combinators.
Combinators allow us to implement otherwise complicated inference algorithms with comparative ease.

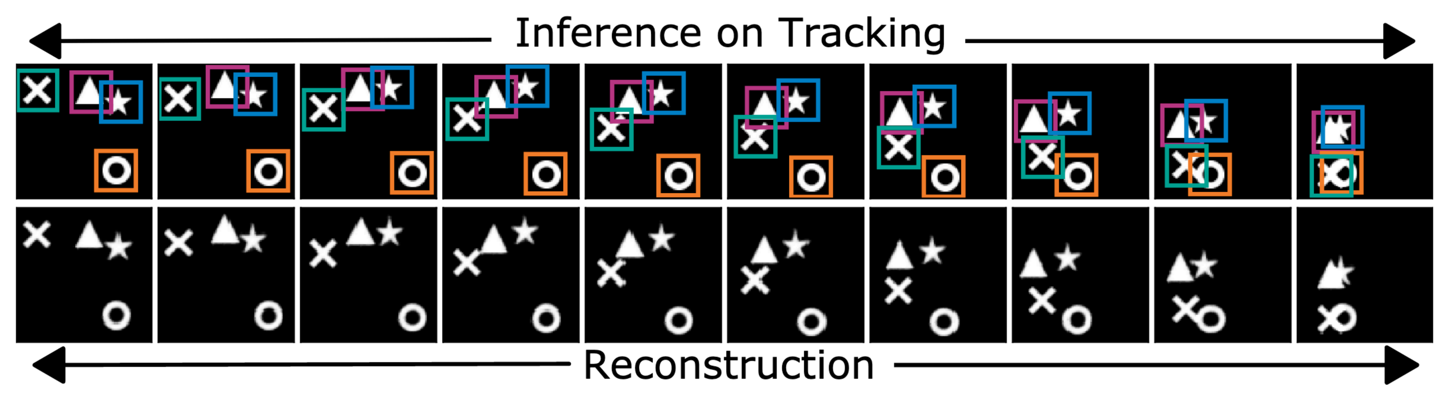
APG algorithm block from paper:

for n **in** $1, \ldots, N$ **do**
  $G_\phi = 0$
  $x^n \sim p^{\mathrm{DATA}}(x)$
  for l **in** $1, \ldots, L$ **do**
    $z^{n,1,l} \sim q_\phi(z \mid x^n)$
    $w^{n,1,l} \leftarrow p_\theta(x^n, z^{n,1,l}) \,/\, q_\phi(z^{n,1,l})$
  for k **in** $2, \ldots, K$ **do**
    $\tilde{z}, \tilde{w} = z^{n,k-1}, w^{n,k-1}$
    for b **in** $1, \ldots, B$ **do**
      $\tilde{z}, \tilde{w} = \mathrm{RESAMPLE}(\tilde{z}, \tilde{w})$
      for l **in** $1, \ldots, L$ **do**
        $\tilde{z}_b'^{\,l} \sim q_\phi(\cdot \mid x^n, \tilde{z}_{-b}^l)$
        $\tilde{w}^l = \frac{p_\theta(x^n, \tilde{z}_b'^{\,l}, \tilde{z}_{-b}^l)\, q_\phi(\tilde{z}_b^l \mid x^n, \tilde{z}_{-b}^l)}{p_\theta(x^n, \tilde{z}_b^l, \tilde{z}_{-b}^l)\, q_\phi(\tilde{z}_b'^{\,l} \mid x^n, \tilde{z}_{-b}^l)}\, \tilde{w}^l$
        $\tilde{z}_b^l = \tilde{z}_b'^{\,l}$
      $G_\phi = G_\phi + \sum_{l=1}^{L} \frac{\tilde{w}^l}{\sum_{l'} \tilde{w}^{l'}} \frac{d}{d\phi} \log q_\phi(\tilde{z}_b^l \mid x^n, \tilde{z}_{-b}^l)$
    $z^{n,k}, w^{n,k} = \tilde{z}, \tilde{w}$
return G_ϕ, z, w

APG code in combinators:

```
def pop_gibbs(target, proposal, kernels, sweeps):
  q = propose(partial(target, suffix=0), partial(proposal, suffix=0))
  for s in range(sweeps):
    for k in kernels:
      q = propose(
          extend(partial(target, suffix=s+1), partial(k, suffix=s)),
          compose(partial(k, suffix=s+1), resample(q, dim=0)))
    return q
```

Inference results:



Implementation.
Combinators can implemented on top of most current PPLs. We provide an implementation based on probtorch and an open design document for pyro at the following:

https://github.com/probtorch/combinators

https://bit.ly/pyro-design