Parameterizing uncertainty by deep invertible networks

Gabrio Rizzuti\textsuperscript{1,2}, Ali Siahkoohi\textsuperscript{1}, Philipp A. Witte\textsuperscript{2}, and Felix J. Herrmann\textsuperscript{1}

\textsuperscript{1} Georgia Institute of Technology, School of Computational Science and Engineering
\textsuperscript{2} Microsoft Research
Uncertainty quantification for imaging

- Siahkoohi, A., Rizzuti, G., and Herrmann, F. J., 2020, Uncertainty quantification in imaging and automatic horizon tracking: a Bayesian deep-prior based approach
Uncertainty quantification for imaging

Siahkoohi, A., Rizzuti, G., and Herrmann, F. J., 2020, *Uncertainty quantification in imaging and automatic horizon tracking: a Bayesian deep-prior based approach*
Bayesian framework

\[
d \sim p_{\text{like}}(d|m)
\]

\[
m \sim p_{\text{prior}}(m)
\]

\[
m \sim p_{\text{post}}(m|d) \propto p_{\text{like}}(d|m) p_{\text{prior}}(m)
\]
Bayesian framework

\[- \log p_{\text{like}}(d|m) = \frac{1}{2\sigma_d^2} \|d - F(m)\|^2\]

\[- \log p_{\text{prior}}(m) = R(m)\]

\[- \log p_{\text{post}}(m|d) = \frac{1}{2\sigma_d^2} \|d - F(m)\|^2 + R(m)\]
MCMC

Langevin Dynamics

\[ \mathbf{m}_{t+\Delta t} = \mathbf{m}_t + \Delta t \nabla_{\mathbf{m}} \log p_{\text{post}}(\cdot | \mathbf{d})|_{\mathbf{m}=\mathbf{m}_t} + \mathcal{N}(\mathbf{0}, 2\Delta t) \]

Computationally inefficient → stochastic gradient LD?

\[- \log p_{\text{post}}(\mathbf{m}|\mathbf{d}) \approx \frac{1}{2\sigma_d^2} \sum_{i=1}^{n_s} \frac{N_s}{n_s} ||\mathbf{d}_i - \mathcal{F}(\mathbf{m})_i||^2 + R(\mathbf{m})\]
Variational Inference

\[ p_{\text{post}}(m|d) \approx p_{\theta}(m), \quad \theta \in \Theta \]

- Density encoded by transport maps (e.g. neural networks)
- Stochastic VI vs MCMC
  - small-batch optimization
  - parallelization
  - preconditioning/transfer learning

Transport Maps

\[ z \sim p_Z(z), \quad T : Z \rightarrow M, \quad T_#p_Z \approx p_{\text{post}}(\cdot | d) \]
Variational inference w/ TMs

$$\min_T \text{KL}(T \# p_Z \| p_{\text{post}}) =$$

$$= \mathbb{E}_{z \sim p_Z(z)} \frac{1}{2\sigma_d^2} \| d - \mathcal{F}(T(z)) \|^2 + R(T(z)) - H(T \# p_Z)$$
Normalizing Flows

Advantages: memory efficient, change of variable formula

\[ p_T(m) = p_Z(T^{-1}(m)) \left| \det J_T(T^{-1}(m)) \right|^{-1} \]

Training and testing UQ w/ NFs

\[
\min_\theta \mathbb{E}_{z \sim p_Z(z)} \frac{1}{2\sigma^2 d} \left\| d - \mathcal{F}(T_\theta(z)) \right\|^2 + R(T_\theta(z)) - \log |\det J_{T_\theta}(z)|
\]
Example (2D)
Example (2D)
Example (AVP)

➢ Sleipner data, co2datashare.org/, thanks to Equinor!
Example (AVP)

True and background models (on a 3m grid)

Sleipner data, co2datashare.org/, thanks to Equinor!
Example (AVP)
Example (AVP)
# Glow network

```julia
G = NetworkGlow(nx, ny, n_in, batchsize, n_hidden, L, K) #/gpu

# Objective function
function loss(X)
    Y, logdet = G.forward(X)
    f = 0.5f0/batchsize*norm(Y)^2 - logdet
    ΔX, X₀ = G.backward(1f0./batchsize*Y, Y)
    return f
end

# Evaluate loss
f = loss(X)

# Update weights
opt = Flux.ADAM()
Params = get_params(G)
for p in Params
    update!(opt, p.data, p.grad)
end
clear_grad!(G)
```
Discussion and conclusions

- Variational inference vs MCMC for seismic inversion
- Non-smooth priors? Hard constraints?
- Extension to 2D/3D w/ invertible networks
- Problem-specific inductive bias (loop unrolling)?
- Supervised+unsupervised via transfer learning?
