IF ONLY MY POSTERIOR WERE NORMAL: INTRODUCING FISHER HMC

Adrian Seyboldt

In Bayesian statistics, we want to sample from the posterior distribution, which is defined by a non-normalized density function $\pi(\theta)$.

Variants of Hamiltonian Markov Chain Monte Carlo (HMC) are the most popular methods to sample from these distribution

DIFFICULTIES WITH HMC

- Not all distributions can be sampled
- Choice of parametrization matters

Huge issue in practice!

GOAL OF FISHER HMC

We want to automate the process of finding a good parametrization.

A more robust HMC, even if it is slower, would be a big win.

PRIOR ART

- Mass matrix adaptation: Adapt the mass matrix of the HMC sampler during tuning. This is a form of reparametrization.
- Variational inference: Find a good parametrization by minimizing the KL divergence between the posterior and a simpler distribution. Then, in a second step, sample from the transformed distribution.
- Riemannian HMC: Somehow come up with a local metric and take that into account during sampling.

WHAT IS A REPARAMETRIZATION?

Posterior defined on space $X \subset \mathbb{R}^n$.

Bijective function $F : Y \to X$:

$$F(\eta) = \theta$$
$$F^{-1}(\theta) = \eta$$

We get a new density

$$\nu(\eta) = \pi(F(\eta)) \cdot |\det dF(\eta)|$$

EXAMPLES

$$\sigma \sim N^+(0,1)$$

So $X = \mathbb{R}^+$ and for instance $F(\eta) = \exp(\eta) = \sigma$.

Rescaling variables:

$$x \sim N(0, 1000)$$

Reparametrization $F(\eta) = 1000\eta = x$

$$\sigma \sim N^{+}(0,1)$$
$$x \sim N(0,\sigma^2)$$

We might want to use

$$F(\eta_0,\eta_1) = (\exp(\eta_0), \exp(\eta_0)\eta_1) = (\sigma, x).$$

HMC

Choose θ_0 and $v_0 \sim N(0, 1)$

Repeat leapfrog steps:

$$
\begin{aligned}
v_{n+1} &= v_n + \frac{\epsilon}{2} \nabla \log \pi(\theta_n) \\
\theta_{n+1} &= \theta_n + \epsilon v_{n+1} \\
v_{n+1} &= v_n + \frac{\epsilon}{2} \nabla \log \pi(\theta_{n+1})
\end{aligned}
$$

Accept or reject θ_n.

TRANSFORMED HMC

Choose $\eta_0 = F^{-1}(\theta_0)$ and $v_0 \sim N(0,1)$

Repeat leapfrog steps:

$$
\begin{aligned}
v_{n+1} &= v_n + \frac{\epsilon}{2} \nabla \log \nu(\eta_n) \\
\eta_{n+1} &= \eta_n + \epsilon v_{n+1} \\
v_{n+1} &= v_n + \frac{\epsilon}{2} \nabla \log \nu(\theta_{n+1})
\end{aligned}
$$

Accept or reject $\theta = F(\eta_n)$.

The proposal only depends on the gradient of the log density $\nabla \log \nu$, not the density itself!

Standard normal posteriors work well, so we want

$$\nabla \log \nu(\eta) \approx \nabla \log N(\eta \mid 0, I)$$

Or we want this to be small:

$$\begin{aligned} &\mathbb{E}_{\nu(\eta)}[\|\nabla \log \nu(\eta) - \nabla \log N(\eta \mid 0, I)\|^2] \\ &= \mathbb{E}_{\nu(\eta)}[\|\nabla \log \nu(\eta) + \eta\|^2] \end{aligned}$$

Fisher divergence with a natural choice of norm

Family of bijections F_λ.

Given posterior draws θ_i and scores s_i in the original space X

we minimize the Monte Carlo estimate of the fisher divergence to find μ:

$$\hat{\mu} = \operatorname{argmin}_\mu \sum_i \|F^*(s_i) + F^{-1}(\theta_i)\|^2$$

NOTE: COMPUTE SCORES ON Y BASED ON SCORES ON X

Given $\nabla \log \pi(\theta)$, we can compute $\nabla \log \nu(\eta)$:

```
def grad_of_nu(theta, grad_theta):
    eta = F_inv(theta)
    _, pull_grad_fn = jax.vjp(F_and_logdet, eta)
    grad_eta = pull_grad_fn((grad_theta, 1.))
    return grad_eta
```

FISHER HMC

Choose some bijection F_0.

Repeat:

1. Generate θ and s using HMC with bijection F_i
2. Fit F_{i+1} by minimizing Fisher Divergence on θ and s.

FISHER HMC

Choose some bijection F_0.

Repeat:

1. Generate θ and s using HMC with bijection F_i
2. Fit F_{i+1} by minimizing Fisher Divergence on θ and s.
   - Which family of bijections?
   - How do we minimize?

COMPARISON TO VI

Family of parametrizations: F_λ

$$p_\lambda(\eta) = (F_\lambda^* \pi)(\eta), \quad q(\eta) = N(\eta \mid 0, I)$$

Loss in Fisher HMC:

$$\min_\lambda \int \|\nabla \log q(\eta) - \nabla \log p_\lambda(\eta)\|^2 p_\lambda(\eta) d\eta$$

Loss in Variational Inference:

$$\min_\lambda \int (\log q(\eta) - \log p_\lambda(\eta)) q(\eta) d\eta$$

POSSIBLE EXTENSION

We are still not using the density!

Possible extensions:

- Minimzie sum of KL divergence and Fisher divergence
- Minimize Sobolev norm $\int \|\nabla f(x)\|^2 + \int f(x)^2$

Using only the Fisher divergence seems to be better?

CHOICES FOR BIJECTIONS

- Coordinate-wise affine functions: Diagonal mass matrix adaptation
- Affine functions: Full mass matrix adaptation
- Affine functions with few non-unit eigenvalues: Low rank modified mass matrix adaptation
- Normalizing flows
- Model-informed families

COORDINATE-WISE AFFINE

$$F(x) = \sigma \odot x + \mu$$

Closed form solution for the optimization problem!

$$\sigma = \sqrt{\frac{\mathrm{Std}(\theta_i)}{\mathrm{Std}(s_i)}} \qquad \mu = \mathrm{Mean}(\theta_i) + \sigma^2 \odot \mathrm{Mean}(s_i)$$

Minimizes $\sum(\lambda_i + \lambda_i^{-1})$.

Mass matrix $M = \mathrm{diag}(s)^{-2}$.

AFFINE BIJECTIONS

$$F(x) = Ax + \mu$$

Closed form solution for optimization problem:

$$AA^T \mathrm{Cov}(s) AA^T = \mathrm{Cov}(\theta)$$
$$\mu = \mathrm{Mean}(\theta) + AA^T \mathrm{Mean}(s_i)$$

Mass matrix $M^{-1} = AA^T$.

LOW-RANK MODIFICATIONS

$$F(x) = D(U(\Lambda - I)U^T + I)x$$

Greedy closed form minimum. First find D, then U and Λ.

Allows fitting some eigenvalues in high dimensional spaces without quadratic cost.

Somehow doesn't work all that well?

NORMALIZING FLOWS:

Other families of functions for F. We need:

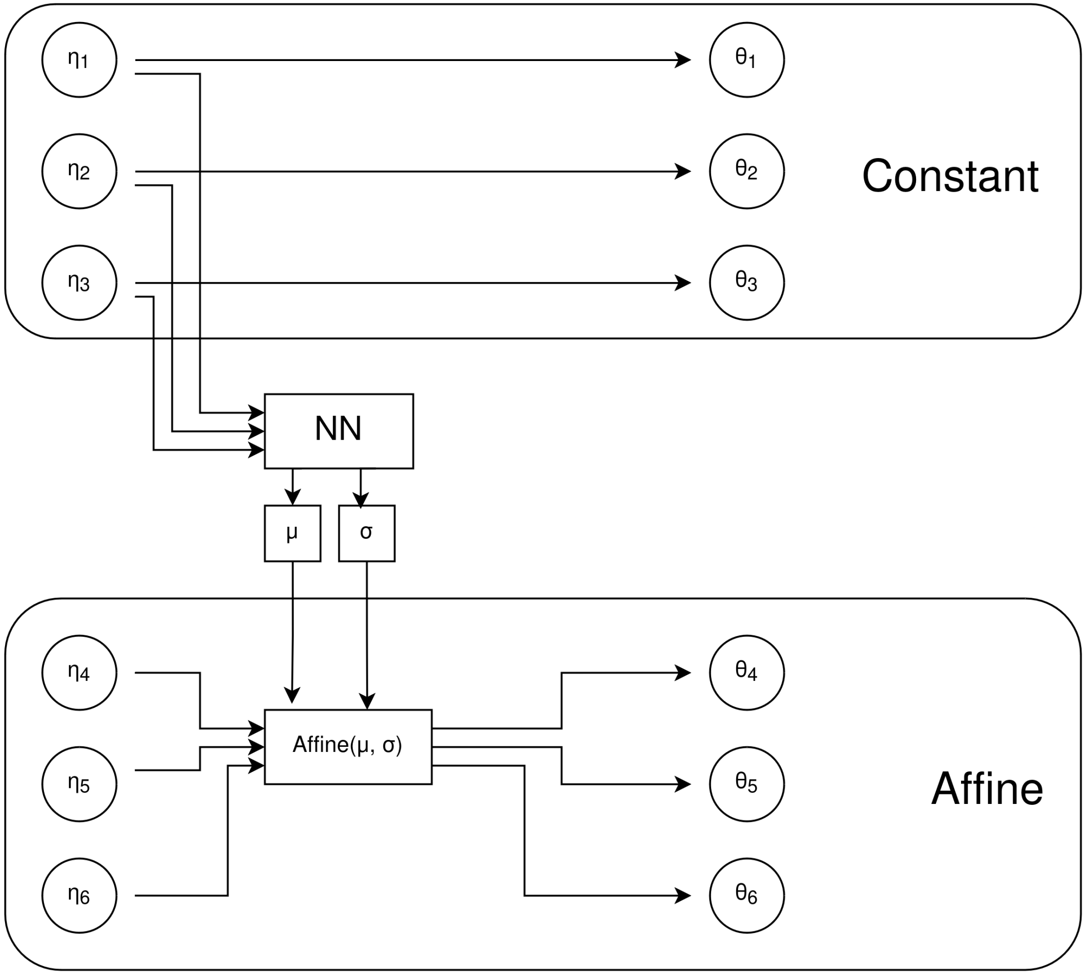
- Evaluations of F
- Evaluations of F^{-1}
- Logdet of F
- Autodiff of those

Can we generalize the affine function family?

Problem: The parameters of Affine are always the same!

We want to scale parameters depending on other values, but the function needs to stay invertable, with easy to compute logp.

REALNVP


η1
θ1
η2
θ2
Constant
η3
θ3
NN
μ
σ
η4
θ4
Affine(μ, σ)
η5
θ5
Affine
η6
θ6

REALNVP

Fix some parameters $\eta_{0:k}$

$$F(\eta) = \mathrm{Concat}(\eta_{0:k}, \mathrm{Affine}(\mathrm{NN}_\mu(\eta_{0:k}), \mathrm{NN}_\sigma(\eta_{0:k}))$$

Several layers of those with different subsets of parameters.

No closed form minimization, we need to minimize with adam or similar

SOME MODIFICATIONS TO REALNVP:

- Parameterize scaling variables as $\exp(\sinh(x))$.
- Replace affine transformations with non-linear function.
- We choose systematically, which parameters to keep constant.
- Additional MvScale layers that scales the space in a particular direction
- Low rank approximations for the hidden layers in the NNs for higher dimensional problems

- Can deal with a much wider range of posteriors!
- Speedups for many badly conditioned models
- Extra cost for well behaved models

FUTURE?

Specialized reparametrizations based on the model.

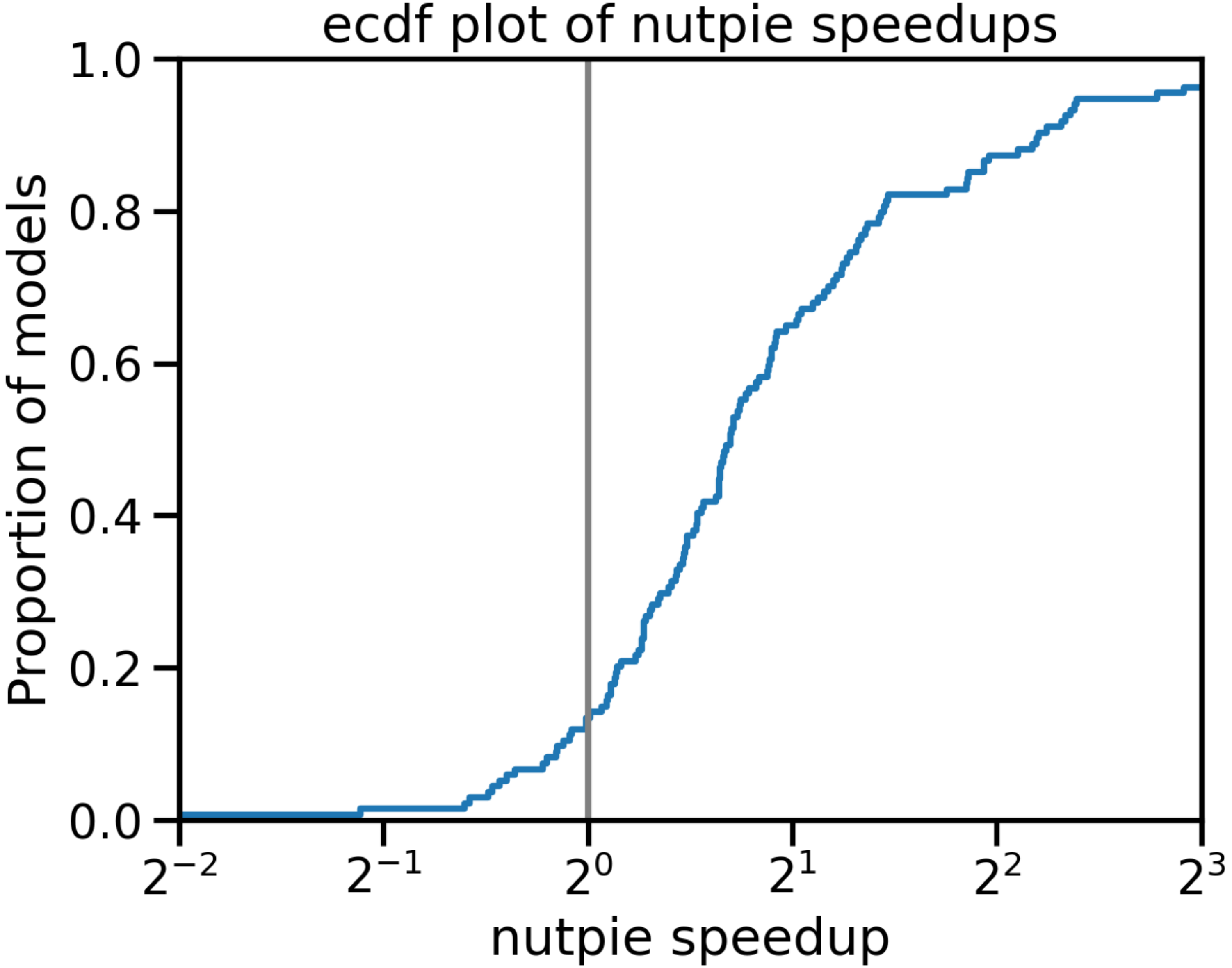
Could be much cheaper than black box methods like neural networks?

IMPLEMENTATION STATUS

- Diagonal: Used in nutpie, might become default in PyMC and Stan?
- Low rank: Implemented in nutpie
- Normalizing flows: nutpie PR, getting merged soon

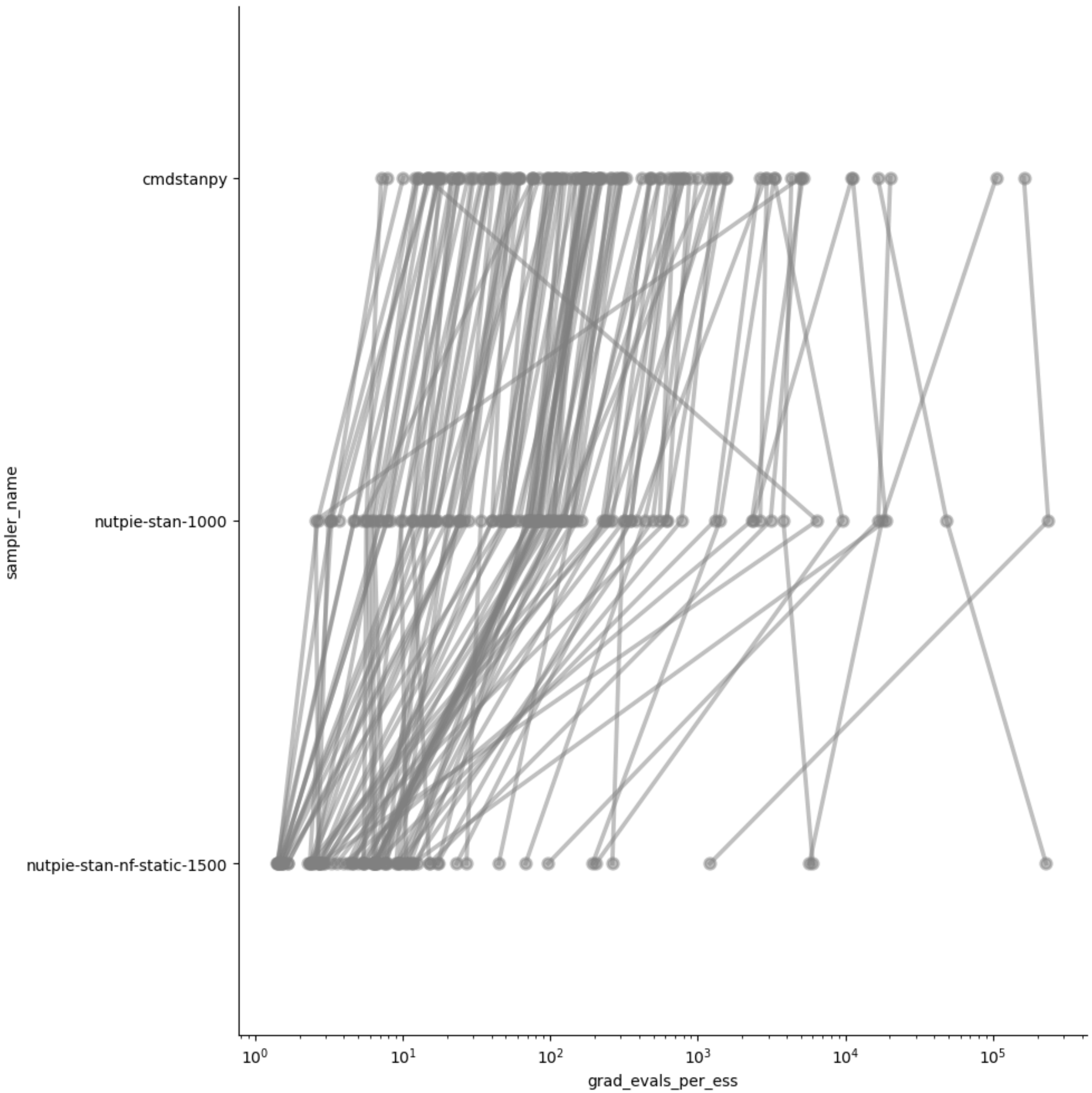
RESULTS FROM POSTERIORDB

DIAGONAL



ecdf plot of nutpie speedups
Proportion of models
1.0
0.8
0.6
0.4
0.2
0.0
2^{-2}
2^{-1}
2^{0}
2^{1}
2^{2}
2^{3}
nutpie speedup

NORMALIZING FLOW


cmdstanpy
nutpie-stan-1000
nutpie-stan-nf-static-1500
sampler_name
10^0
10^1
10^2
10^3
10^4
10^5
grad_evals_per_ess

Feedback and benchmarks welcome!

@aseyboldt on discourse or github.

Questions?