introduction-to-inference

June 4, 2025

1 Bayesian inference with compact binaries

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import uniform, norm
from tqdm.auto import trange

%matplotlib inline
```

1.1 Data model

The data is measured strain obtained by calibrating photodetector measurements.

$$d(t) = \int d\tau\, C(t;\tau) * d_{\rm err}(\tau)$$

Assumptions

- Detector calibration is approximately stationary and described by some model $D(f|\theta_C)$
- Stationary gaussian noise n with power spectrum $S(f|\theta_S)$
- Deterministic astrophysical signal $h(f|\theta_A)$
- The response of the detector to the signal depends on the calibration and geometric factors $R(f;\theta_C,\theta_A)$
- Deterministic terrestrial noise “glitches” $g(f|\theta_G)$

$$\tilde{d}(f;\theta_C,\theta_S,\theta_A,\theta_G) = \tilde{n}(f;\theta_S) + R(f;\theta_A,\theta_C)h(f;\theta_A) + g(f;\theta_G)$$

NOTE: the noise (and PSD) does not include a calibration correction.

1.2 Data model

The data is measured strain obtained by calibrating photodetector measurements.

$$d(t) = \int d\tau\, C(t;\tau) * d_{\rm err}(\tau)$$

Common simplifications

- The power spectrum and glitch content are known: $\theta_S, \theta_G \sim \delta(\theta_S, \theta_G)$ (with signifcant exceptions)

$$\tilde{d}(f; \theta_C, \theta_A) = \tilde{n}(f; S(f)) + h_{\rm cal}(f; \theta_A, \theta_C)$$

$$\tilde{d}(f; \theta_C, \theta_A) \sim N(n; \mu = h_{\rm cal}(f; \theta_A, \theta_C), \sigma^2 = S(f))$$

1.3 Likelihood

$$\ln L(\tilde{d}|h, C, \theta_A, \theta_C, S, G) = \sum_{f_i} \left[-\frac{2}{T} \frac{|\tilde{d}(f_i) - h_{\rm cal}(f_i; \theta_A, \theta_C)|^2}{S(f_i)} \right] - \ln \left(\frac{\pi T S(f_i)}{2} \right)$$

Since S and $\tilde{d}$ are fixed we can reduce this to

$$\ln L(\tilde{d}|h, C, \theta_A, \theta_C, S, G) \sim -\frac{2}{T} \sum_{f_i} \left[\frac{|h_{\rm cal}(f_i; \theta_A, \theta_C)|^2}{S(f_i)} - 2\mathbb{R} \left(\frac{|\tilde{d}(f_i)^* h_{\rm cal}(f_i; \theta_A, \theta_C)|^2}{S(f_i)} \right) \right]$$

$$\ln L(\tilde{d}|h, C, \theta_A, \theta_C, S, G) \sim -\frac{1}{2} \langle h_{\rm cal}, h_{\rm cal} \rangle_S + \mathbb{R} \left(\langle d, h_{\rm cal} \rangle_S \right)$$

1.3.1 Modifications

- Reduce the computational cost of the inner products
  - Reduce the amount of time-frequency space
  - Pre-compress the data to isolate the known signal morphology
- Marginalise over extrinsic parameters
- Multi-stage/importance sampling for improved model fidelity

Multi-banding

- Use time-dependent frequency bins
- Explicitly excises much of the time-frequency space

Figure: Morisaki arXiv:2104.07813

1.3.2 Wavelet Transform

- Perform the analysis in a pixel basis.
- For each template, only pixels where the template is non-zero is needed.

Figure: Cornish arXiv:2009.00043

1.3.3 Heterodyning/Relative Binning

- Assume all relevant signals look like a fiducial point (typically a high-likelihood point)
- Express other waveforms as a sum of smooth corrections to the fiducial waveform

Figure: Zackay+ arXiv:1806.08792

1.3.4 Reduced Order Quadrature

- Compute an optimal compression matrix for some subset of possible signals
- Estimate the likelihood from the compressed data

1.3.5 Extrinsic marginalization

- Some extrinsic parameters have a simple effect on the observed signal
- This can lead to an analytic (or fast) method to marginalize over the parameter
- Removing strongly degenerate degrees of freedom makes exploring the space simpler

Examples

- distance
  - the distance scales the amplitude, the likelihood can be integrated trivially, typically uses a cached lookup table
- phase
  - for simple waveform models, the likelihood can be analytically integrated using Bessel functions
- time
  - the time of arrival can be numerically integrated using a FFT
- other extrinsic
  - marginalize over all extrinsic parameters using Monte Carlo integration (or similar)
  - e.g., RIFT (Lange+ arXiv:1805.10457), Roulet+ arXiv:2404.02435

1.4 Priors

The choice of prior distribution can have a significant impact on the inferred posterior for individual events, especially for poorly measured quantities like spin.

1.4.1 How to choose them

There are two broad classes of methods for choosing the prior distribution: - physically agnostic, typically uniform or geometrically motivated - physically informed, typically using past observations to motivate the prior for future events, a.k.a., the posterior predictive as prior.

Parameter	Agnostic	Motivated*
Sky location	Isotropic	Known location of EM multimessenger events
Distance	Uniform in comoving frame	Follow star formation + time delay
Relative orientation	Isotropic	???
Masses	Uniform in components	Inferred population, galctic NS popualtion
Spin magnitudes	Uniform	Inferred population, theoretical predictions, X-ray binaries
Spin orientations	Isotropic	Inferred population, theoretical predictions, X-ray binaries

Parameter	Agnostic	Motivated*
Calibration	Broad Gaussian	Informed by detector calibration measurements
Power spectrum	Spline + Lorentzian	Theoretical noise budgets
Glitches	Wavelets	Theoretical models, e.g., scattering arches

* The final column are not unique (or even actively used) and just intended to be illustrative

1.5 Parameterization

The most obvious choice of parameters often leads to strongly correlated spaces that are difficult to explore. Changing variables can significantly simplify the problem and improve robustness of sampling at reduced computational cost. In many cases, the original prior distribution can be maintained by computing the Jacobian for the tranformation, either analytically of via autodiff.

1.5.1 Example: masses

There is a strong correlation between the mass of the two objects. To leading post-Newtonian order, this is captured using the chirp mass and mass ratio

$$M = \frac{(m_1 m_2)^{\frac{3}{5}}}{(m_1 + m_2)^{\frac{1}{5}}}$$
$$q = \frac{m_2}{m_1}$$
$$\pi(M, q) = M \frac{(1+q)^{2/5}}{q^{6/5}} \pi(m_1, m_2)$$

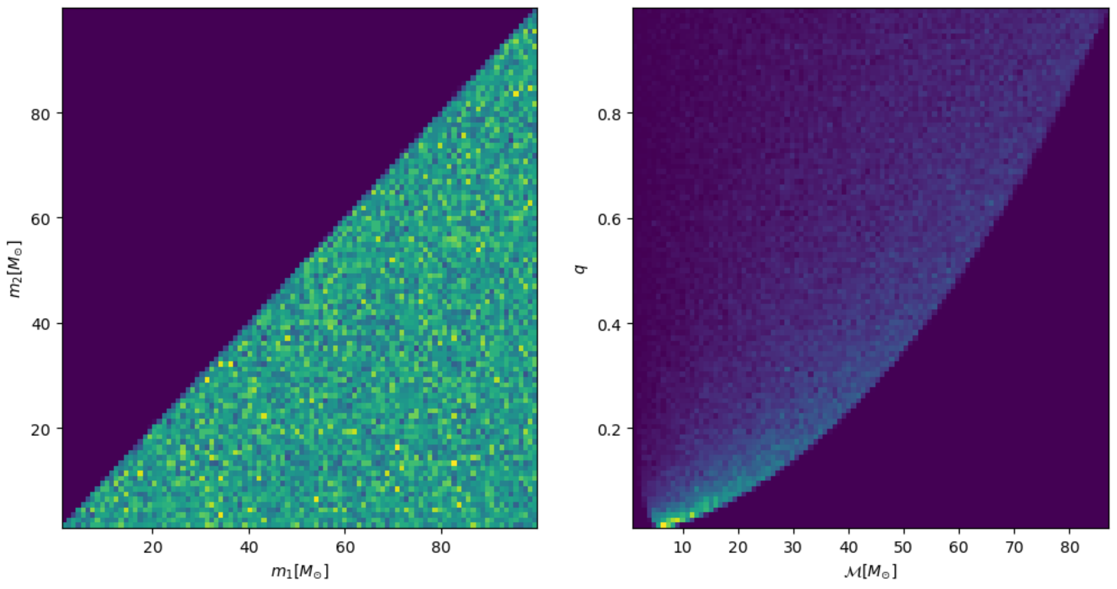
1.5.2 Example: masses

There is a strong correlation between the mass of the two objects. To leading post-Newtonian order, this is captured using the chirp mass and mass ratio

$$M = \frac{(m_1 m_2)^{\frac{3}{5}}}{(m_1 + m_2)^{\frac{1}{5}}}$$
$$q = \frac{m_2}{m_1}$$
$$\pi(M, q) = M \frac{(1+q)^{2/5}}{q^{6/5}} \pi(m_1, m_2)$$

- There are more complex combinations of mass and spin that can further reduce the complexity of the sampled space, e.g., Lee+ arXiv:2203.05216.
- Deep learning methods can also in principle determine arbitrary changes of reparameterizations using, e.g., normalizing flows.

[2]:
```
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline

mass_2, mass_1 = np.sort(np.random.uniform(1, 100, (2, 100000)), axis=0)
chirp_mass = (mass_1 * mass_2)**0.6 / (mass_1 + mass_2)**0.2
mass_ratio = mass_2 / mass_1

fig, axes = plt.subplots(ncols=2, figsize=(12, 6))

axes[0].hist2d(mass_1, mass_2, bins=100)
axes[0].set_xlabel(r"$m_1 [M_{\odot}]$")
axes[0].set_ylabel(r"$m_2 [M_{\odot}]$")

axes[1].hist2d(chirp_mass, mass_ratio, bins=100)
axes[1].set_xlabel(r"${\cal M} [M_{\odot}]$")
axes[1].set_ylabel("$q$")

plt.show()
plt.close()
```



1.5.3 Example: extrinsic parameters

- Geometric arguments can define a "natural" basis for the problem that significantly simplifies the sampling
- E.g., defining the zenith direction aligned with the vector separating interferometers instead of declination

- See Roulet+ arXiv:2207.03508 for a nice exposition

2 Stochastic sampling

2.1 Problem

- We have a target distribution $p(\theta|d)$ that we want samples from*†.
- We can evaluate p, but not sample it directly.

* Why do we want to draw samples from p?

† We may also want to calculate a "Bayesian evidence".

2.2 Why Stochastic Sampling

2.2.1 Why don't we just evaluate $p(\theta|d)$ or $(d|\theta)$ directly every time we need it?

- Visualization requires marginalizing over all but one or two parameters
- Population analyses (currently) require samples
- Evaluating the per-event likelihood is expensive with standard waveform models

2.3 An Alternative? - RIFT (Lange+ arXiv:1805.10457)

- Evaluate the likelihood on a grid of intrinsic parameters marginalized over all other parameters (in practice only extrinsic parameters)
- Fit a smooth model to the likelihood at each of those points, e.g., using a Gaussian process
- Can also be used to generate samples

2.4 Stochastic Sampling Methods

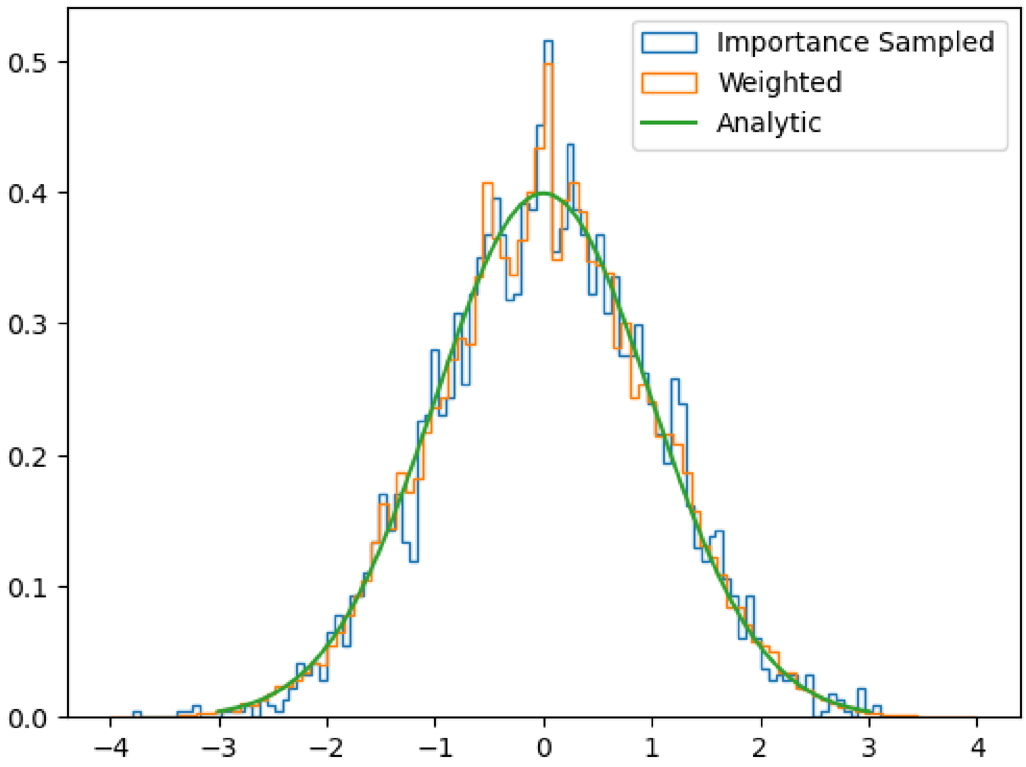
2.4.1 Monte Carlo

- Define some fixed proposal distribution, possibly conditioned on the data or a summary of the data, $\psi(\theta|d)$.
- Draw samples from $\varphi \sim \psi$
- Calculate weights $w = \frac{p(\varphi|d)}{\varphi|d}$
- Importance sample using the weights

[3]:
```python
proposal = uniform(loc=-4, scale=8)
target = norm(loc=0, scale=1)

proposed = proposal.rvs(10000)

weights = target.pdf(proposed) / proposal.pdf(proposed)

keep = weights > np.random.uniform(0, max(weights), len(weights))

posterior = proposed[keep]

print(f"Rejection sampling efficiency: {np.mean(keep):.2f}")
print(f"Importance sampling efficiency: {np.mean(weights)**2 / np.
    ↪mean(weights**2):.2f}")
print(f"Integral: {np.mean(weights):.2f} +/- {np.std(weights) / len(weights)**0.
    ↪5:.2f}")
```

```
Rejection sampling efficiency: 0.31
Importance sampling efficiency: 0.44
Integral: 1.00 +/- 0.01
```

[4]:
```
plt.hist(posterior, bins=100, density=True, histtype="step", label="Importance␣
    ↪Sampled")
plt.hist(proposed, bins=100, weights=weights, density=True, histtype="step",␣
    ↪label="Weighted")
plt.plot(np.linspace(-3, 3, 1000), target.pdf(np.linspace(-3, 3, 1000)),␣
    ↪label="Analytic")
plt.legend()
plt.show()
plt.close()
```



Examples

- prior sampling:

    - simple to implement
    - often very inefficient
    - does not strongly depend on the data
- "SimplePE" (Fairhurst+ (arXiv:2304.03731)):
    - Use physical arguments to generate an efficient proposal based on summary statistics of the data
- deep learning proposals, e.g., `DINGO-IS`
    - I was told not to talk about this

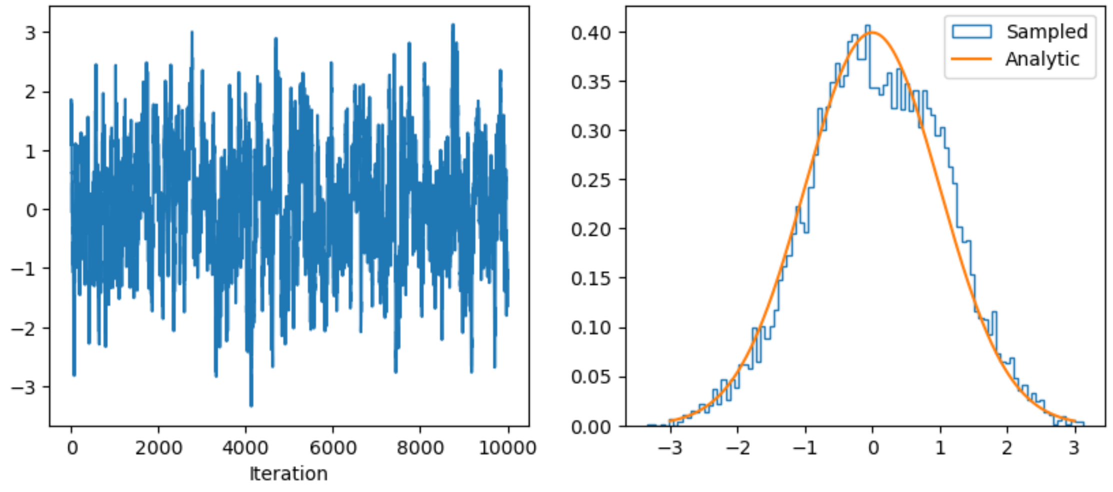
2.4.2 Markov Chain Monte Carlo (MCMC)

- Start from some initial state θ_t (or ensemble).
- Define a transition kernel $K(\theta_{t+1}|\theta_t)$ and an acceptance function $A(\theta_t, \theta_{t+1}|d)$
    - many choices for the transition kernel:
        * ensemble methods
        * parallel tempering
        * gradient-based sampling, e.g., NUTS
        * deep-learning rejection sampling
        * …
- Under fairly weak conditions $\{\theta_i\} \sim p(\theta|d)$

[5]:
```python
proposal = norm(loc=0, scale=0.3)
likelihood = norm(loc=0, scale=1)
prior = uniform(loc=-4, scale=8)

point = np.random.uniform(-3, 3)
post = likelihood.pdf(point) * prior.pdf(point)

chain = list()

naccept = 0

for _ in trange(10000):
    proposed = point + proposal.rvs()
    proposed_post = likelihood.pdf(proposed) * prior.pdf(proposed)

    accept_prob = proposed_post / post
    if accept_prob > np.random.uniform(0, 1):
        point = proposed
        post = proposed_post
        naccept += 1

    chain.append(point)

print(f"Acceptance rate: {naccept / len(chain):.2f}")
```

```
  0%|          | 0/10000 [00:00<?, ?it/s]

Acceptance rate: 0.90
```

```
[6]: fig, axes = plt.subplots(ncols=2, figsize=(10, 4))

     axes[0].plot(chain)
     axes[0].set_xlabel("Iteration")

     axes[1].hist(chain, bins=100, density=True, histtype="step", label="Sampled")
     axes[1].plot(np.linspace(-3, 3, 1000), target.pdf(np.linspace(-3, 3, 1000)),␣
      ↪label="Analytic")
     axes[1].legend()

     plt.show()
     plt.close()
```



Examples

- emcee: ensemble sampling
- ptemcee: parallel-tempered emcee (not currently maintained, looking for a new maintainer!)
- numpyro: (mostly) gradient-based sampling
- FlowMC: gradient/deep-learning sampling
- bilby_mcmc): supports ensemble, parallel-tempering, and deep-learning proposals
- …

2.4.3 Nested Sampling

- Start with a set of "live" samples from the prior
- Iteratively replace the lowest likelihood live point with a point with a higher likelihood point from the prior $\theta_{t+1} \sim \pi(\theta)_{L(d|\theta_{t+1})>L_*}$
  - new samples are typically generated using Monte Carlo or MCMC
- Nested sampling chain can be importance sampled to obtain posterior samples

Variations

- “dynamic” nested sampling uses a variable number of live points
- “Importance Nested Sampling” (INS) turns nested sampling into an optimized MC proposal

[7]:
```
# initialize the nested sampler

likelihood = norm(loc=0, scale=1)
prior = uniform(loc=-4, scale=8)

nlive = 1000
live = prior.rvs(nlive)
live_likelihood = likelihood.pdf(live)

remaining_volume = 1
compression = np.exp(-1  / nlive)

chain = list()
prior_volume = list()
chain_likelihood = list()
```

[8]:
```
# main nested sampling loop

for ii in trange(10000):
    worst = np.argmin(live_likelihood)
    worst_point = live[worst]
    worst_likelihood = live_likelihood[worst]

    new_likelihood = -1

    bound = (-2 * np.log(worst_likelihood * (2 * np.pi)**0.5))**0.5
    new_point = np.random.uniform(-bound, bound)
    new_likelihood = likelihood.pdf(new_point)

    chain.append(worst_point)
    prior_volume.append(remaining_volume * (1 - compression))
    remaining_volume *= compression
    chain_likelihood.append(worst_likelihood)
    live[worst] = new_point
    live_likelihood[worst] = new_likelihood
```

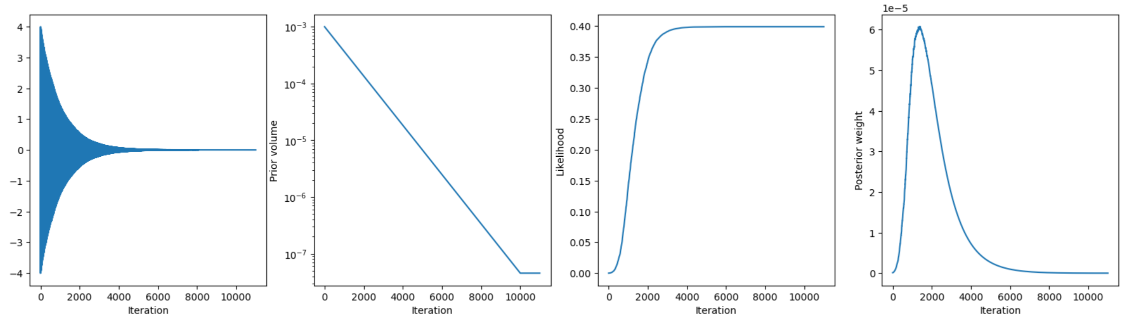
```
  0%|          | 0/10000 [00:00<?, ?it/s]
```

[9]:
```
# add final live points

remaining_volume = np.exp(-len(chain) / nlive)
for point, like in zip(live, live_likelihood):
    prior_volume.append(remaining_volume / nlive)
    chain.append(point)
    chain_likelihood.append(like)
```

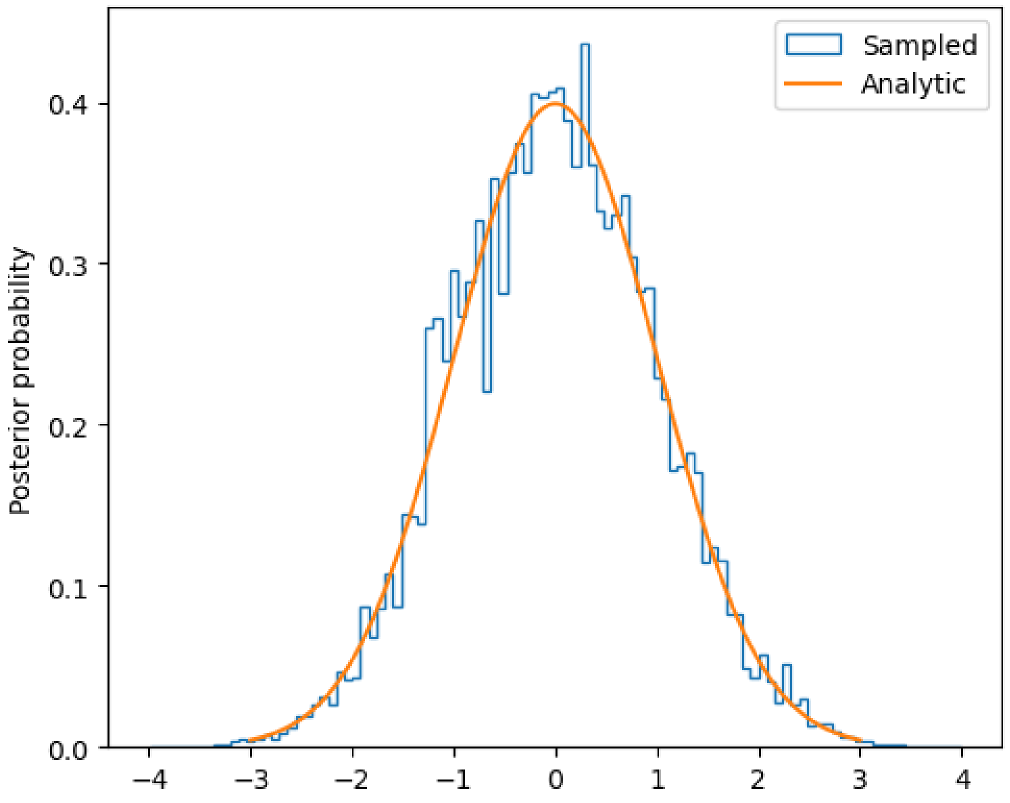
```
weights = np.array(chain_likelihood) * np.array(prior_volume)
print(f"Evidence: {np.sum(weights):.2f}")
print(f"Expected: {0.125}")
```

```
Evidence: 0.13
Expected: 0.125
```

[10]:
```
fig, axes = plt.subplots(ncols=4, nrows=1, figsize=(20, 5))

axes[0].plot(chain)
axes[0].set_xlabel("Iteration")
axes[1].plot(prior_volume)
axes[1].set_xlabel("Iteration")
axes[1].set_ylabel("Prior volume")
axes[1].set_yscale("log")
axes[2].plot(chain_likelihood)
axes[2].set_xlabel("Iteration")
axes[2].set_ylabel("Likelihood")
axes[3].plot(weights)
axes[3].set_xlabel("Iteration")
axes[3].set_ylabel("Posterior weight")
plt.show()
plt.close()
```



[11]:
```
fig = plt.figure(figsize=(6, 5))

plt.hist(chain, bins=100, weights=weights, density=True, histtype="step",␣
 ↪label="Sampled")
plt.plot(np.linspace(-3, 3, 1000), target.pdf(np.linspace(-3, 3, 1000)),␣
 ↪label="Analytic")
plt.ylabel("Posterior probability")
plt.legend()
plt.show()
plt.close()
```



Examples

- `dynesty`
  - MCMC and MC proposals via mutliple ellipsoids
- `nestle` (not currently maintained)
  - MCMC and MC proposals via mutliple ellipsoids
- `pymultinest`
  - MCMC and MC proposals via mutliple ellipsoids
  - supports INS
- `nessai`
  - MC proposals via neural networks
  - supports INS
- `nautilus`
  - MC proposals via neural networks
  - INS only
- `jaxted` (not production ready, but talk to me if you're interested)
  - `JAX`-compatible, parallelised MCMC proposals
- …

2.4.4 Sequential Monte Carlo (a.k.a. Particle Filter)

- Transition a set of samples from one distribution to another via a series of small steps

- At each iteration, the current points are "mutated" using e.g., MCMC or Monte Carlo

Examples

- Sequential tempering (`pocomc`, `coix`)
  - Step through $\pi(\theta)L(d|\theta)^{\beta}$ with $\beta \in 0, \beta_1, \beta_2, \dots, 1$
- Nested sampling via sequential Monte Carlo (Salome+ arXiv:1805.03924, `jaxted`)
  - Step through $\pi(\theta)_{L(d|\theta)>L_*}$ for $L_* \in 0, L_1, L_2, \dots, L_{\max}$