Hamiltonian MC

Marco Szalay - BAT Meeting
MPP -12th Nov 2018

Max-Planck-Institut für Physik

Overview

- **Introduction**
  - What is HMC?
  - How does an HMC sampler work?
- **DynamicHMC.jl**
  - Bernoulli process
  - Sample from a Gaussian
- **Integration with BAT.jl**
- **Conclusions and Outlook**

What is HMC?

Hamiltonian MC (a.k.a hybrid MC):

- Algorithm to efficiently propose samples from a given distribution alternative to, e.g. Gibbs sampler or random walk in Metropolis-Hastings
- Developed in the '80s to efficiently compute lattice QCD
- Utilizes a powerful analogy to physics where the target distribution behaves like a potentital U(x) that can be efficiently "explored" by a particle with enough kinetic energy

$$T(\theta, \theta') = \mathcal{N}(\theta'|\theta, \sigma^2) min \left(\frac{f(\theta')}{f(\theta)}, 1 \right)$$

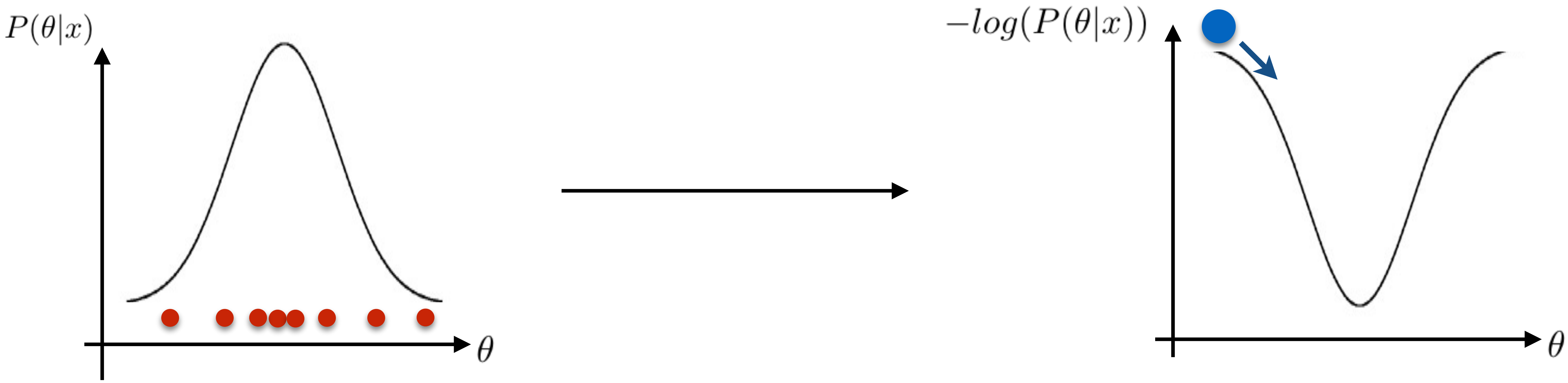
random diffusion

Metropolis acceptance

Substitute this with something more efficient

HMC sampler - I

Given some posterior $P(\theta|x)$



after introducing a new set of parameters M (our momentum), we can write an energy-like component:

$$E(\theta, M) = K(M) + U(\theta)$$

And then compute the probability of the system being in any given energy state:

$$P(E) \propto exp(-E(\theta, M)/T)$$

HMC sampler - II

$$P(E) \propto exp(-(K(M) + U(\theta))/T)$$

continuing with the particle analogy, we can choose K and U as such:

$$K(M) = \frac{M^2}{2m}$$

Euclidian HMC

$$U(\theta) = -log(P(x|\theta)P(\theta))$$

likelihood **prior**

for a probe of mass m=1 and at T=1 (no tempering) the probability becomes:

$$P(\theta, M) \propto e^{-\frac{M^2}{2}} P(x|\theta)P(\theta)$$

Normal distribution

our target distribution is the marginalization of this new joint density

we can sample from $P(\theta, M)$ and obtain a sample of $P(x|\theta)P(\theta)$

HMC sampler - III

Why is sampling from $P(\theta, M)$ beneficial?

Sample momentum from $\mathcal{N}(M|0,1)$

and then evolve the system with: $\frac{d\theta}{dt} = M$, $\frac{dM}{dt} = -\frac{\partial U}{\partial \theta}$

for some time T until:

$(\theta, M) \rightarrow (\theta^*, M^*)$

Now Metropolis-Hastings:

$$r = \frac{P(x|\theta^*)P(\theta^*)\mathcal{N}(M^*|0,1)}{P(x|\theta)P(\theta)\mathcal{N}(M|0,1)} \times \frac{P(\theta, M|\theta^*, M^*)}{P(\theta^*, M^*|\theta, M)}$$

Metropolis

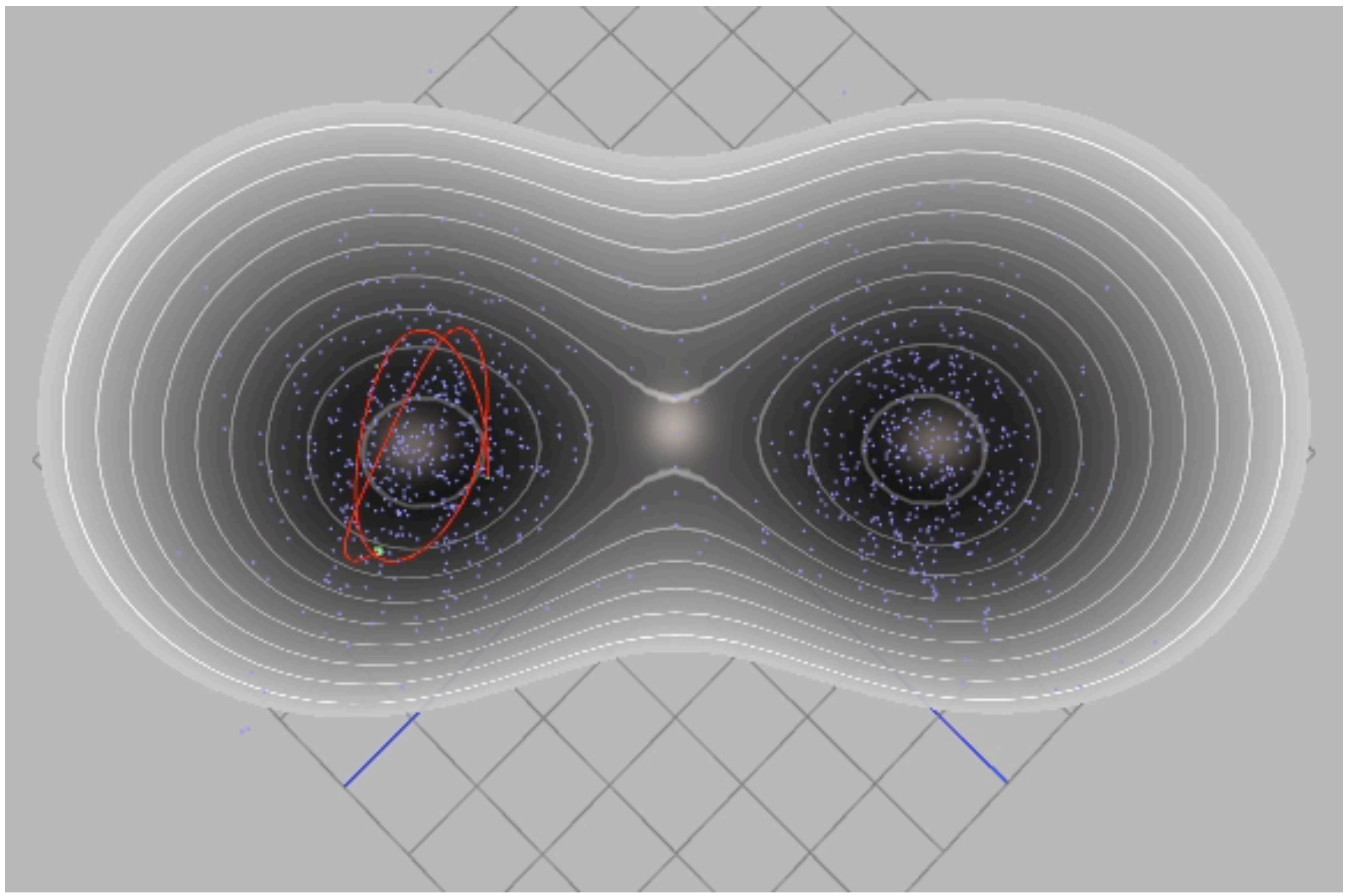
Hastings

= 1 if M → -M

if r > rnd[0,1] → accept new sample

HMC sampler - IV

Combining all together, the result is remarkable:



source: github, Alex Rogoznikov

- **Drawbacks of HMC:**
  - Needs the derivative of the target distribution
  - How to optimize length of trajectory?
  - mass of the "particle"?
  - Temperature?
  - Efficiently integrate the equations of motion (leapfrog tuning, error propagation…)
- **Solution:**
  Let someone else worry about it!

DynamicHMC.jl

DynamicHMC.jl is:

- Julia package from Tamas Papp (IAS Vienna)
- Robust HMC implementation (NUTS)
- Automatically tunes relevant parameters for HMC
- Good Documentation
- Well written & Maintained!

DynamicHMC.jl needs:

- Log probability of the target distribution
- Its gradient (can be made optional with autodifferentiation)

DynHMC examples: Bernoulli

Start with the simple example from the documentation:

```
struct BernoulliProblem
    "Total number of draws in the data."
    n::Int
    "Number of draws `==1` in the data"
    s::Int
end
```

← Define the problem

```
function (problem::BernoulliProblem)((α, )::NamedTuple{(:α, )})
    @unpack n, s = problem
    s * log(α) + (n-s) * log(1-α)
end
```

← and the log probability

```
p = BernoulliProblem(20, 10)
P = TransformedLogDensity(as((α = as𝕀,)), p)
∇P = ADgradient(:ForwardDiff, P)
```

← use autodifferentiation for the gradient

```
chain, NUTS_tuned = NUTS_init_tune_mcmc(∇P, 1000)
```

← run the HMC

```
posterior = transform.(Ref(∇P.transformation), get_position.(chain));
posterior_α = first.(posterior);
mean(posterior_α)

0.49908427887376866
```

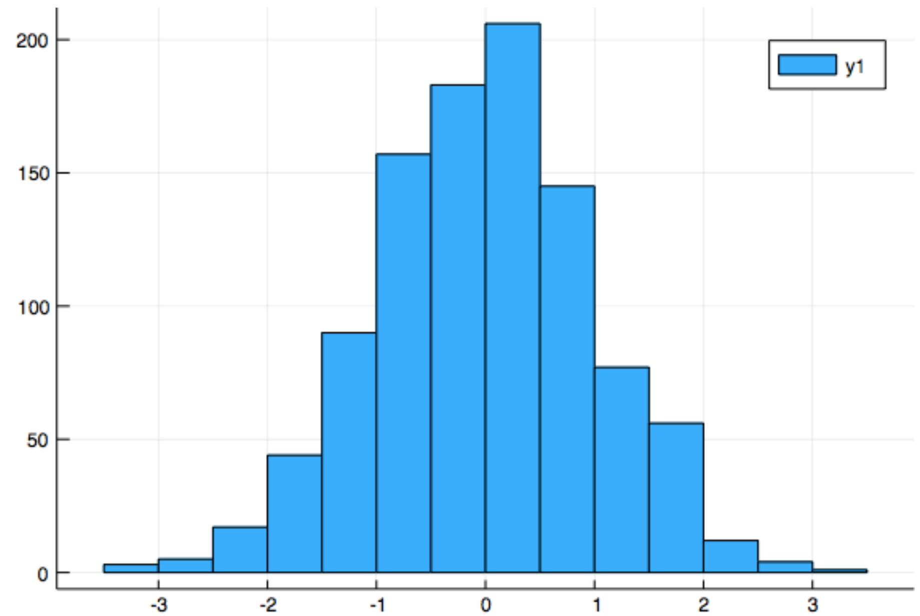
← check the result

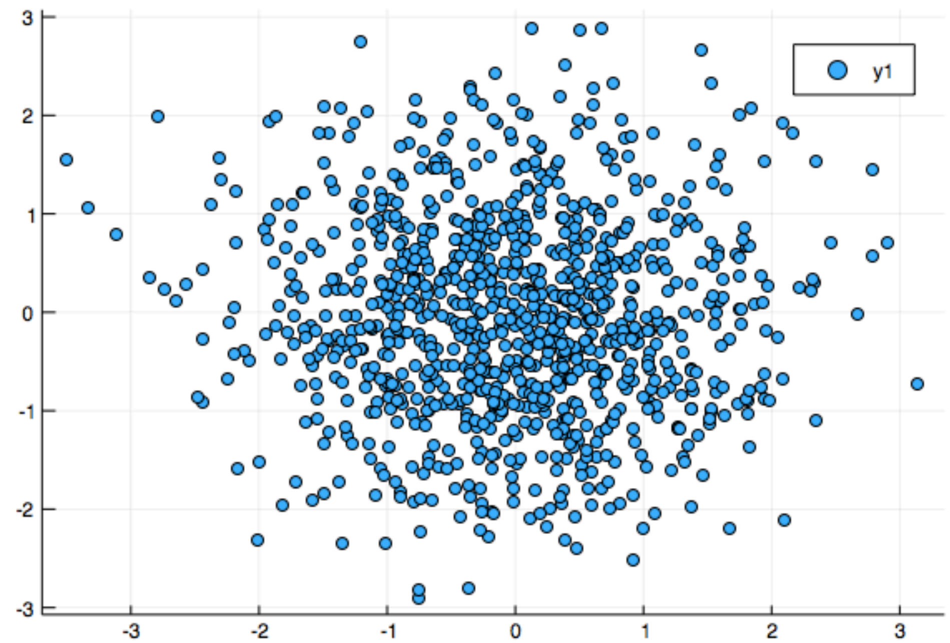
DynHMC examples: Gaussian

More involved:
we can use HMC with BAT models and BAT.unsafe_density_logval, use autodifferentiation and get samples from HMC

```
struct GaussianDistributionDensity<:AbstractDensity
    mean::Float64
    sigma::Float64
end
```

```
function BAT.unsafe_density_logval(target::GaussianDistributionDensity, params::Vector{Float64},
exec_context::ExecContext = ExecCapabilities())
    logprob = log(1.0/sqrt(2.0*pi*target.sigma^2) * exp(-(params[1]-target.mean)^2/(2*target.sigma^2)))
    return max(logprob, -800)
end
```





Integration with BAT.jl

So far:

- Wrapper to use DynamicHMC with some of the machinery of BAT
- Accepts BAT models and likelihood definition
- Automatically computes gradient from the likelihood

To do:

- Make HMC callable as any other BAT algorithm and wrap results in BAT sample struct (so that we have diagnostics, plotting…)
- Let the user define the logprob gradient
- Integrate Dynamic HMC with multi-threading capabilities of BAT (?)

Conclusion and Outlook

- Hamiltonian MC is a very efficient algorithm for sample proposal (when the problem doesn't have too many modes and particularly for many-dimensions problems)
- Implementation of HMC details and parameters tuning can be tricky
- One day in the library can save you 6 month in the lab (especially true when writing code)
- DynamicHMC seems to perform well and is well maintained from a reputable researcher
- Some work still needs to be done until using it in BAT is as simple as random-walk MH
- Great learning material on HMC from:
  Neal (2010), **Betancourt** (2017), **Rogoznikov**, **Lambert** (A Student's Guide to Bayesian Statistics)