Parameter Estimation of Gaussian Mixtures using HMC and MH

Özge Bozal , Abdullah Karaaslanlı, Korhan Polat

Problem

Our aim is to estimate parameters of synthetic Gaussian mixtures by using Metropolis-Hastings (MH) and Hamiltonian Monte Carlo (HMC) algorithms. We assume that the number of components of mixture is known.

Method

A mixture model is a convex combination of distributions [1]:

$$\sum_{i=1}^{k} p_i f_i(x), \quad \sum_{i=1}^{k} p_i = 1, \quad k > 1 \tag{1}$$

Where p_i are weights and f_i are parametric distribution with unknown parameters θ_i. Given n observed samples $\underline{x} = \{x_1, ..., x_n\}$ from mixture of distributions, unknown parameters $\underline{\theta}$ and weights $\underline{p}$ are estimated with Bayesian inference, such that posterior distribution of parameters is maximized:

$$\pi(\underline{\theta}, \underline{p}|\underline{x}) = \mathcal{L}(\underline{x}|\underline{\theta}, \underline{p})\pi(\underline{\theta}, \underline{p}) \tag{2}$$

Where $\pi(\underline{\theta}, \underline{p})$ is prior distribution of parameter and $\mathcal{L}$ is likelihood function defined as:

$$\mathcal{L}(\underline{x}|\underline{\theta}, \underline{p}) = \prod_{j=1}^{n} \sum_{i=1}^{k} p_i f_i(x_j|\underline{\theta_i}) \tag{3}$$

Maximization of Eq.2 is analytically intractable in most of the cases, especially when k is large, therefore a numerical estimation method is required.

In our problem f_i is a Gaussian distribution with unknown parameters $\underline{\theta} = \{\mu, \sigma\}$, with weight p_i's there are $3k$ unknown parameters in total. We use Metropolis-Hastings and Hybrid Monte Carlo algorithms to approximate their true values. Prior selection is as follows based on [1] and [3]:

$$\mu_i \sim \mathcal{N}(0, \eta)$$
$$\sigma_i \sim \mathcal{LN}(0, \psi)$$
$$(p_1, ..., p_k) \sim \mathcal{D}(\lambda_1, ..., \lambda_k)$$

Where $\mathcal{LN}$ and $\mathcal{D}$ are log-normal and Dirichlet distributions respectively and η, ψ, λ_i are hyper parameters that are predefined.

In MH, we propose μ in a random walk manner and σ is drawn from $\mathcal{LN}(log(\sigma^{(t-1)}, \zeta))$ where $\sigma^{(t-1)}$ is previous sample in Markov chain and ζ is scale of proposal that is tuned manually for a proper acceptance rate. Lastly, weights are proposed as σ and they are normalized as required by Gaussian mixture models.

One chain is run for both HMC and MH with number of iteration selected according to the convergence of approximations. Burn-in region is half of the number of iterations and thinning is selected as 2 to decrease the correlation between samples. We implemented MH and HMC in MATLAB and Python STAN respectively.

Hamiltonian MC

The main idea of HMC is to develop a Hamiltonian function $H(\mathbf{x}, \mathbf{p})$ such that the resulting Hamiltonian dynamics allow us to efficiently explore some target distribution $p(\boldsymbol{x})$.

Energy function for Hamiltonian dynamics is a combination of potential and kinetic energies.For any energy function $E(\theta)$ over a set of variables θ, we can define the corresponding canonical distribution as: $p(\theta) = \frac{1}{Z} e^{-E(\theta)}$ where $E(\theta) = H(\mathbf{x}, \mathbf{p}) = U(\mathbf{x}) + K(\mathbf{p})$ which has a corresponding canonical distribution:

$$\begin{aligned} p(\mathbf{x}, \mathbf{p}) &\propto e^{-H(\mathbf{x}, \mathbf{p})} \\ &= e^{-[U(\mathbf{x}) - K(\mathbf{p})]} \\ &= e^{-U(\mathbf{x})} e^{-K(\mathbf{p})} \\ &\propto p(\mathbf{x}) p(\mathbf{p}) \end{aligned}$$

Where $\boldsymbol{x}$ is position, whose distribution is our interest, and $\boldsymbol{p}$ is momentum. Because the canonical distribution for $\mathbf{x}$ is independent of the canonical distribution for $\mathbf{p}$, we can choose any distribution from which to sample the momentum variables.

Algorithm 1 Hamiltonian MC

1: Set $t = 0$ and initial position $\mathbf{x}^{(0)} \sim \pi^{(0)}$
2: **repeat**
3: $t \leftarrow t + 1$
4: Sample a new initial momentum variable from the momentum canonical distribution $\mathbf{p}_0 \sim p(\mathbf{p})$
5: Set $\mathbf{x}_0 = \mathbf{x}^{(t-1)}$
6: Run Leap Frog algorithm starting at $[\mathbf{x}_0, \mathbf{p}_0]$ for L steps and stepsize δ to obtain proposed states $\mathbf{x}^*$ and $\mathbf{p}^*$
7: Calculate the Metropolis acceptance probability α as:

$$\min\left\{1, \frac{\exp(U(\mathbf{x}_0) + K(\mathbf{p}_0))}{\exp(U(\mathbf{x}^*) + K(\mathbf{p}^*))}\right\}$$

8: Draw a random number u from $\mathcal{U}(0, 1)$
9: **if** $u \leq \alpha$ **then**
10: Accept the proposed state position $\mathbf{x}^*$ and set the next state in the Markov chain $\mathbf{x}^{(t)} = \mathbf{x}^*$
11: **else**
12: Set $\mathbf{x}^{(t)} = \mathbf{x}^{(t-1)}$
13: **end if**
14: **until** $t = M$

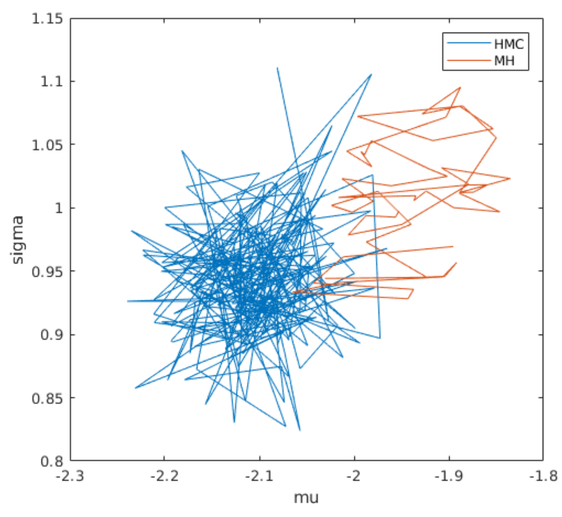

Figure 3: Random walk behaviour

Results

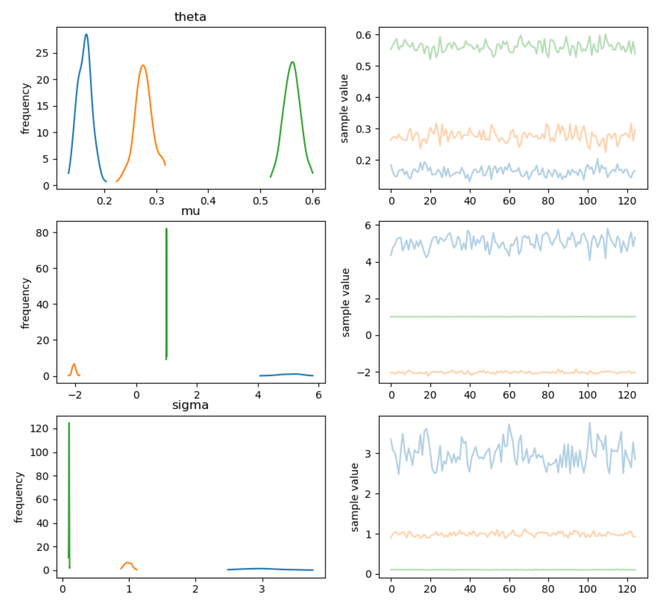

Figure 1: HMC for 3 components

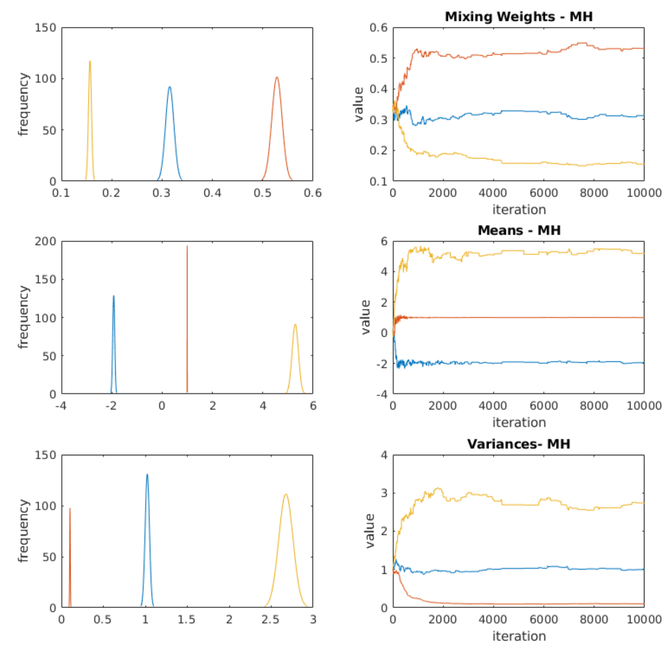

Figure 2: MH for 3 components

	True	HMC	MH
p_1	0.3000	0.3000	0.3100
p_2	0.1500	0.1400	0.1700
p_3	0.5500	0.5600	0.5200
μ_1	-2.0000	-2.0000	-1.9200
μ_2	5.0000	5.3900	5.1300
μ_3	1.0000	1.0100	0.9900
σ_1	1.0000	0.9400	1.0000
σ_2	3.0000	2.5500	2.7100
σ_3	0.1000	0.1000	0.1600

Table 1: Parameter approximations from HMC and MH with true parameters.

Conclusion

The MH algorithm converges slower than HMC. This results from the fact that consecutive samples of MH have much higher autocorrelation than samples drawn using HMC which uses Hamiltonian dynamics that provides distant proposals and efficient exploration of posterior distribution space.

References

[1] Marin, Jean-Michel, Kerrie Mengersen, and Christian P. Robert. "Bayesian modelling and inference on mixtures of distributions." Handbook of statistics 25 (2005): 459-507.

[2] Neal, Radford M. "MCMC using Hamiltonian dynamics." Handbook of Markov Chain Monte Carlo 2.11 (2011).

[3] Stan Development Team. 2017. Stan Modeling Language Users Guide and Reference Manual, Version 2.17.0. http://mc-stan.org