Assignment 1: [Adaptive steps + temperature + noise]

Xiaolin(Eric) Tian
Boston University
`akemihl@bu.edu`

Abstract

A one-paragraph summary of your method, key results, and main takeaways. The main purpose of the assignment is to develop an innovative MALA that does a better job at random walk sampling methods. To test its effectiveness, two benchmarks are tested: the Rosenbroek distribution and Neal's Funnel. Compared to RWMH, the new methods show some superiority at difficult and tricky distributions. A few ideas were applied in the new method: adaptive step sizes, temperature, and adding noise. Compared to HMC, the innovative method doesn't show a huge increase in performance, but it does improve in certain cases. The trace plots, marginal distributions, acceptance rates and effective sample size were diagnosed at the end of the experiment.

1 Introduction

What is your method and why did you design it this way? What problem or limitation of existing methods does it address?

Random walks were widely used for sampling complex probabilities distributions, though there are many variant ways of applying them, it might not be always ideal when facing tricky distributions. To improve the traditional RWMH and HMC, this project meant to use a MALA sample that is more adaptive when facing uncommon distributions.

2 Method

Technical description of your algorithm. Include pseudocode if helpful.

We propose an adaptive Metropolis-Adjusted Langevin Algorithm (MALA) sampler. Given a target density $p(\theta)$, MALA constructs proposals using a discretization of Langevin dynamics:

$$\theta' = \theta + \tfrac{1}{2}\epsilon^2 \nabla \log p(\theta) + \epsilon z, \quad z \sim \mathcal{N}(0, I), \tag{1}$$

Because this proposal distribution is asymmetric, a Metropolis–Hastings correction is applied. The acceptance probability is given by:

$$\alpha = \min\left(1, \frac{p(\theta')q(\theta \mid \theta')}{p(\theta)q(\theta' \mid \theta)}\right), \tag{2}$$

where $q(\cdot \mid \cdot)$ denotes the Gaussian proposal density defined by the Langevin update.

To reduce sensitivity to hyperparameter tuning, we adapt the step size ϵ during an initial warmup phase using a Robbins–Monro update on $\log \epsilon$ to target a desired acceptance rate. After the warmup period, the step size is fixed and sampling proceeds with the tuned value. This ensures positivity of the step size and provides stable, diminishing adaptation.

To better help the model better at exploring the narrow end of the tunnel, a temperature less than 1 was used as a way to give the sampler more chance of sampling the narrow distribution.

In the sampler a new hyperparameter was set as Temperature. And since a temperature less than one would give you a flatter distribution, it is selected as 0.85 after several rounds of testing. def run$_m y_s ampler(key, log_p rob_f n, initial_p osition, n_s amples = 50_0 00, step_s ize = 0.15, adapt_s teps = 5_0 00, target_a ccept = 0.8, Temperature = 0.85):$

logp$_t heta = log_p rob_f n(theta)/Temperatureogp_p rop = log_p rob_f n(theta_p rop)/Temperature$

To prevent the model from being stuck, a small noise was being added. The noise gives the adaptive steps a certain level of randomness, so that the acceptance rate will not be stuck if the movement itself was not ideal.

3 Experiments

Results on both benchmarks. Include visualizations (samples, traces) and quantitative diagnostics (ESS, acceptance rates). Compare to vanilla MH and HMC baselines.

In the project, some experiments were done in order to improve the performance of the MALA when sampling the Neal's tunnel. To fit the model, the first thing needed to be done was to get an ideal step size. When testing the distribution in HMC, a few steps was put in a for loop as candidates, from 0.01 to 0.28, and after screening by a built in function that selects the most ideal step size, the 0.15 step size was selected for the HMC sampler, therefore 0.15 was used as a strong candidate for the first round of test for MALA. The result of step size 0.15 was not the most solid step size in the MALA sampler for the first several rounds of testing, before adding temperature, and after testing, 2.0 was selected as the initial step size. After seeing the result, I noticed that the left tail of the Neal's tunnel was not well sampled. To solve the problem a decision of adding temperature was made. The first round of testing was to test if the temperature should be bigger than one or smaller than one. After a few rounds of playing with the hyper parameter Temperature, a Temperature slightly smaller than one seems to be ideal at filling the gap on the left tail of the distribution. After playing with the temperature, i noticed that the temperature caused the spread on the right side less spread as before. To better sample the right wing, I try to set the accept rate more than 0.6(which was set before) to urge the model to take bolder steps so that it can be more spread to the right. After setting the acceptance higher the left tails became less well sampled than before. To adjust it I went back to the step sizes and set them lower. After a few mix and match, the sampler seems to do better than the beginning, and it's well balanced in the asymmetrical distribution. Looking at the Arviz analysis, the trace plot for MALA is not steady as HMC for both benchmarks. This suggests that the model is still stuck at a certain area, and might need further improvement, and the autocorrelation for the MALA model was still high. To solve this I first doubled the sample sizes, so that there is more room for the and adaptive steps to adjust the step size, and to reduce the autocorrelation, I went back reducing the accept rate, and finally set it to 0.4, this change will help reducing the autocorrelation. Then I increased the initial step size for the same purpose. The final autocorrelation was then lowered. Comparing the MALA model in the two distributions, it did not do a very good job on the banana distributions, it got stuck a lot, and the trace drifted a lot too. The autocorrelation was high, which indicated not very good sampling in that distribution.

3.1 Graphs

4 Discussion

Where does your method work well? Where does it struggle? What would you try next?

There is a lot that can be approved. The improvement here is that by applying temperature, it is more likely to explore the narrow area of the distribution. This can make the model better at exploring distributions like the Neal's Funnel. The Adaptive step sizes also help with smartly adjusting the step size, so that I can be more likely to be accepted. The application noise can help with getting out of potholes that stocks the model. There are a lot that need to be improved too. The first thing would be that it needs more sample size and can cause a lot more computational power if the distribution is complicated and the sample size is huge. To make the model more practical, The first thing that needs to be improved is lowering the autocorrelation further. One thing that can be improved on is to

output1.png

set the noise more random and in a maybe slightly higher distribution. Therefore there is less chance that the model will stick and get less autocorrelation. Another thing that can be improved is to use mixed stepsize, where it only becomes adaptive when the chance of being sample rejected is too high. Another thing that can help with exploring asymmetrical distribution would be applying adaptive temperature. Therefore if the sampler is sampling narrow areas can be more explorative and bolder when the distribution is spread.

5 AI Collaboration

Reflect on your use of AI assistants:

- Which AI tools did you use and for what purposes? Through what interfaces (e.g., chat, code completion, coding agents, . . .)? I used ChatGpt 5.2 and Gimini for improving my code and debugging, and I consulted how to properly apply adaptive steps and add noise. I specifically asked them to help me better at writing math functions and defining python functions.
- What prompting strategies were effective? What kind of context or instructions helped? I think giving them what I wish to improve and specific coding works well, showing them what I have helps with quicker debugging and structure improvement.
- Where did the AI provide useful ideas vs. where did you need to correct or guide it? It is useful at explaining concepts, and writing complete coding functions.
- What did you learn about working with AI on technical problems? What I learned is that they are great at explaining problems you don't know, and helpful at giving advice on coding.

output2.png

output4.png

output3.png

output5.png

output6.png

output7.png

output8.png

output9.png