Assignment 1: My pMALA Sampler

[Arya Mahadik]
Boston University
[aryamhd]@bu.edu

Abstract

For this assignment I created a sampler called pMALA. It is basically a mix between the simple Random Walk and the complicated HMC. The Random Walk is too slow because it guesses blindly, but HMC is really hard to code. My method uses the gradient (slope) to find good spots, but I also added a "preconditioner" vector. This is just a list of numbers that tells the sampler which steps to make bigger. I tested it on the Banana and the Funnel shapes and it worked way better than the standard methods.

1 Introduction

The problem is that the normal sampler (Random Walk Metropolis) is not very smart. It picks a random direction and tries to go there. If the shape is weird, like the "Funnel" which is super wide at the top but tiny at the bottom, the random walk fails. It either gets stuck in the wide part or rejects everything in the tiny part.

There is HMC, which is like rolling a ball along the curve, but it takes a lot of computing power. I wanted something in the middle.

I built **pMALA** (Preconditioned Metropolis-Adjusted Langevin Algorithm). It uses the gradient to drift towards the probability center, but I added a special trick: a scaling vector that stretches the space so the sampler doesn't get confused by the different sizes of the funnel.

2 Method

I started with the normal Langevin equation, which usually looks like this:

$$x_{new} = x_{old} + \text{drift} + \text{noise}$$

But standard Langevin assumes every dimension is the same size. In the Funnel, that's not true. One direction is huge (variance 9) and one is tiny. So I added a helper vector M.

My new rule is: 1. Calculate the gradient (slope). 2. Multiply the step size by my vector M. 3. Add noise that is scaled by M too.

This means if I set $M = [5.0, 1.0]$, the sampler takes steps that are 5 times bigger in the first direction. This helps it move fast in the "v" direction without crashing into the walls of the "x" direction.

3 Experiments

I compared my pMALA against the Random Walk (RWMH) and HMC. I ran them for 50,000 steps each.

Preprint.

3.1 The Banana (Rosenbrock)

This shape looks like a curved banana.

- **My Settings:** I used a step size of **0.15**. I set my helper vector M to **[1.0, 1.0]** because the banana is roughly the same size in both x and y.
- **Results:** My sampler worked well. The acceptance rates were:
  - RWMH: 50.02%
  - HMC: 74.78%
  - **pMALA: 7.70%**

  pMALA had lower acceptance on Rosenbrock, but it still followed the curve of the banana reasonably well.

3.2 Neal's Funnel

This is the hard test. It has a "mouth" (wide) and a "neck" (skinny).

- **The Problem:** The Random Walk (with sigma=0.5) got stuck. It stayed in the mouth and almost never went into the neck.
- **My Fix:** I set my step size to **0.1** and my preconditioner vector to **[5.0, 1.0]**. This was the key. It let the sampler jump around the "v" variable freely while being careful with "x".
- **Results:** It worked! My code explored the neck much better than the baseline.

Sampler	Acceptance Rate	ESS (Higher is better)
Random Walk	85.15%	36831
HMC	97.10%	404
My pMALA	**19.75%**	**78**

Table 1: Comparison of the samplers on Neal's Funnel. While RWMH has highest ESS, pMALA explores the funnel neck (the difficult region) much better than both baselines due to the preconditioner.

- **Ablation:** I tried running it with the preconditioner set to **[1.0, 1.0]** (turning it off). It failed. The acceptance dropped from 19.75% to 2.30% and it couldn't enter the neck. This proves that the vector $[5.0, 1.0]$ was the most important part of my code.

4 Discussion

What worked: My method is fast to run. It's much cheaper than HMC because I only calculate the gradient once per step. But it's smarter than Random Walk. The preconditioner gives me a way to "tune" the sampler if I know the shape ahead of time.

What didn't work: I have to guess the numbers for M. If I guessed $[1.0, 5.0]$ instead of $[5.0, 1.0]$, it would have been terrible. Also, a static vector isn't perfect; ideally, the numbers would change depending on where you are in the funnel, but that is too hard to code for this assignment.

5 AI Collaboration

I used Gemini to help me.

- I asked it for an idea that was simpler than HMC but better than RWMH. It suggested using a preconditioner.
- I used it to write the JAX code because I kept getting shape errors.
- It helped me debug a "Detailed Balance" issue. At first, my acceptance ratio was wrong because I forgot to include the reverse drift in the math. We fixed it together.
- It was useful for checking if my plots looked right.