Approximate Posterior Sampling via Stochastic Optimisation

Connie Trojan[1] Srshti Putcha[2]

[1]University of Durham [2]STOR-i, Lancaster University

excellence with impact

Background

Large scale machine learning models typically rely on stochastic optimisation techniques to learn parameters of interest, since they are designed to be computationally efficient even for large datasets.

However, stochastic optimisation methods only converge to a point estimate. It is often advantageous to understand the parameter uncertainty in the learning process using Bayesian inference. We usually simulate the Bayesian posterior using sampling algorithms known as Markov Chain Monte Carlo (MCMC). However, these samplers can be very slow for large datasets.

Stochastic gradient MCMC methods, like the stochastic gradient Langevin dynamics algorithm (SGLD), aim to capture parameter uncertainty more efficiently by combining stochastic optimisation methods with MCMC.

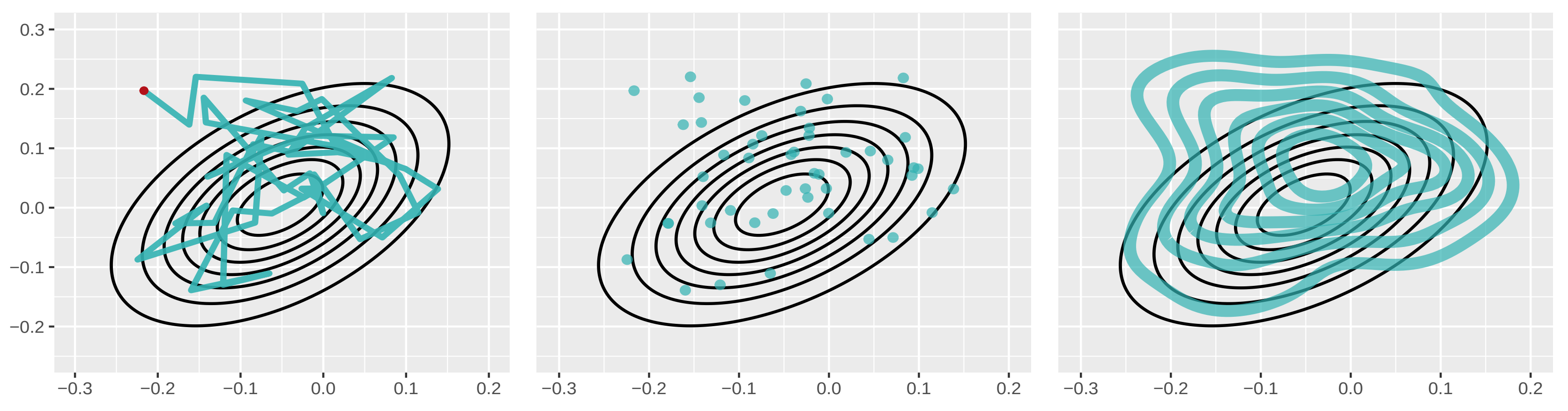

Figure 1: From left to right: MCMC random walk, scatterplot of samples, kernel density estimate. The target distribution (in black) is a 2D Gaussian.

Markov Chain Monte Carlo (MCMC)

Many problems for which Bayesian inference would be useful involve non-standard distributions and a large number of parameters, making exact inference challenging.

MCMC algorithms aim to generate random samples from the posterior, allowing us to estimate the integrals required for inference. These samplers construct a Markov chain, often a random walk, which converges to the desired stationary distribution.

Notation

The Bayesian posterior distribution $\pi(\theta)$ has the form:

$$\pi(\theta) \propto p(\theta) \prod_{i=1}^{N} \ell(x_i|\theta)$$

where the prior $p(\theta)$ describes initial beliefs about θ and $\ell(x_i|\theta)$ is the likelihood associated with observation i. In particular, gradient-based MCMC algorithms use the log posterior $f(\theta)$:

$$f(\theta) = k + f_0(\theta) + \sum_{i=1}^{N} f_i(\theta) \equiv k + \log p(\theta) + \sum_{i=1}^{N} \log \ell(x_i|\theta)$$

Metropolis-Adjusted Langevin Algorithm (MALA)

To propose candidate samples, MALA uses a discretised approximation of the overdamped Langevin diffusion. A Metropolis-Hastings accept/reject step is used to ensure convergence to the desired stationary distribution.

Set starting value θ_0 and step size σ^2. Iterate the following:

1. Set $\theta^* = \theta_t + \frac{\sigma^2}{2}\nabla f(\theta_t) + \sigma\eta_t$, where $\eta_t \sim N(0, I)$
2. Accept and set $\theta_{t+1} = \theta^*$ with probability $a(\theta^*, \theta_t) = \min\left\{1, \frac{\pi(\theta^*)q(\theta_t|\theta^*)}{\pi(\theta_t)q(\theta^*|\theta_t)}\right\}$, with $q(x|y) = P(\theta^* = x|\theta_t = y)$
3. If rejected, set $\theta_{t+1} = \theta_t$

Two calculations over the whole dataset are performed at each iteration, so the algorithm is very slow for large datasets.

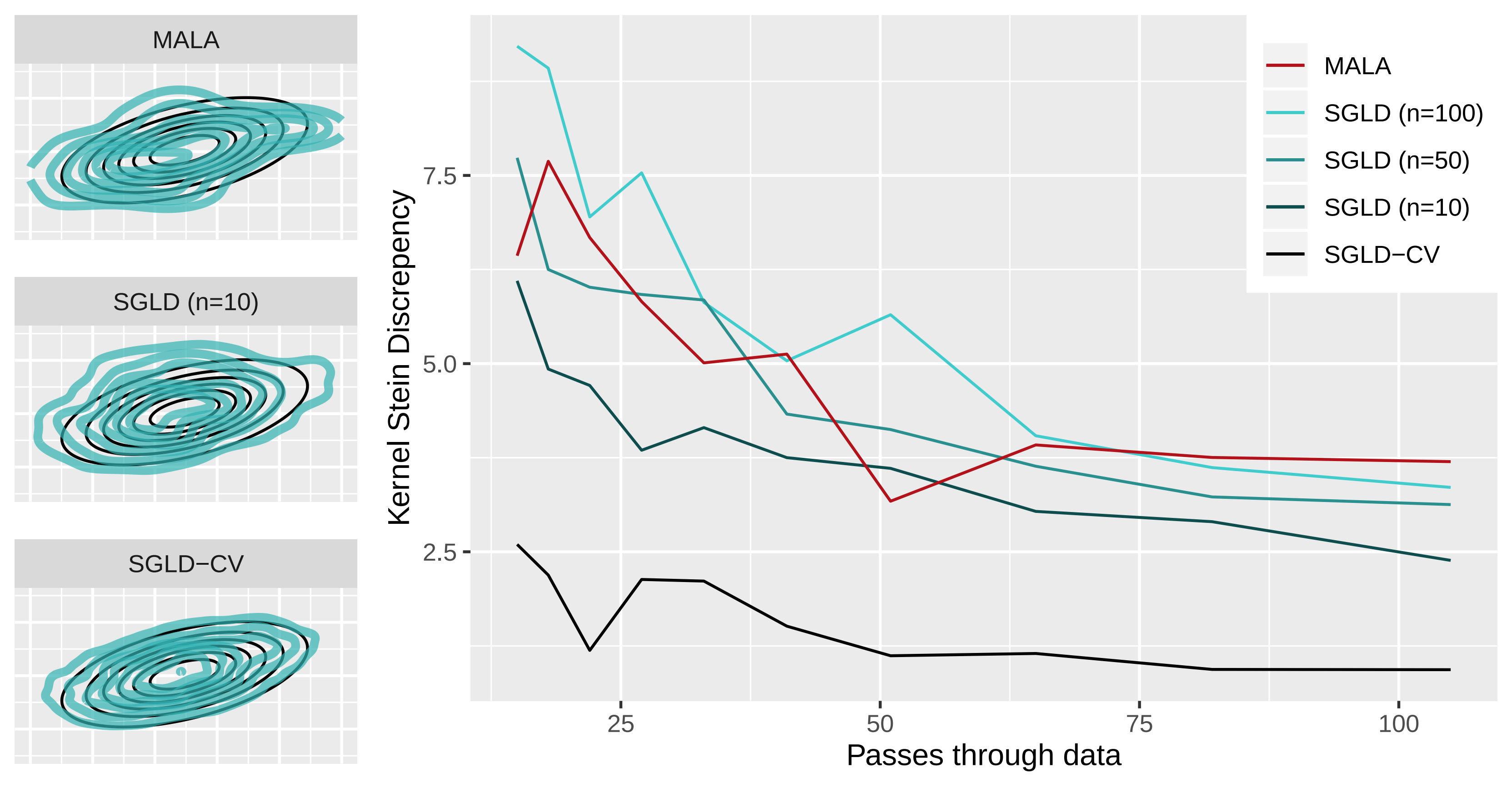

Figure 2: Left: Sampled densities after 100 passes through data. Right: Performance comparison.

Stochastic Gradient Langevin Dynamics (SGLD)

SGLD combines stochastic optimisation with MALA by using a stochastic gradient estimate and eliminating the accept/reject step.

Set starting value θ_0, batch size n, and step sizes ϵ_t. Iterate:

1. Take a subsample S_t of size n from the data
2. Estimate the gradient at θ_t by
$$\nabla\hat{f}(\theta_t) = \nabla f_0(\theta_t) + \frac{N}{n}\sum_{x_i \in S_t} \nabla f_i(\theta_t)$$
3. Set $\theta_{t+1} = \theta_t + \frac{\epsilon_t}{2}\nabla\hat{f}(\theta_t) + \sqrt{\epsilon_t}\eta_t$, where $\eta_t \sim N(0, I)$

Only a fraction of the dataset is used at each iteration, resulting in a lower computational cost. However, the step sizes require careful tuning.

SGLD with Control Variates (SGLD-CV)

SGLD-CV improves on SGLD by using a gradient estimate with lower variance. This is achieved by finding $\hat{\theta}$, a value of θ close to the mode, calculating the exact gradient $\nabla f(\hat{\theta})$ there, and conditioning on $\hat{\theta}$ in later gradient estimates.

Since $\nabla f(\theta_t) = \nabla f(\hat{\theta}) + \left[\nabla f(\theta_t) - \nabla f(\hat{\theta})\right]$, we can estimate $\nabla f(\theta_t)$ by:

$$\nabla\tilde{f}(\theta_t) = \nabla f(\hat{\theta}) + \left[\nabla f_0(\theta_t) - \nabla f_0(\hat{\theta})\right] + \frac{N}{n}\sum_{x_i \in S_t}\left[\nabla f_i(\theta_t) - \nabla f_i(\hat{\theta})\right]$$

$\hat{\theta}$ is found using stochastic optimisation and used as the starting value, replacing the burn-in time of standard SGLD. $\nabla\tilde{f}(\theta)$ has lower variance than $\nabla\hat{f}(\theta)$ because $\hat{f}(\hat{\theta})$ and $\hat{f}(\theta_t)$ are correlated for $\hat{\theta} \approx \theta_t$.

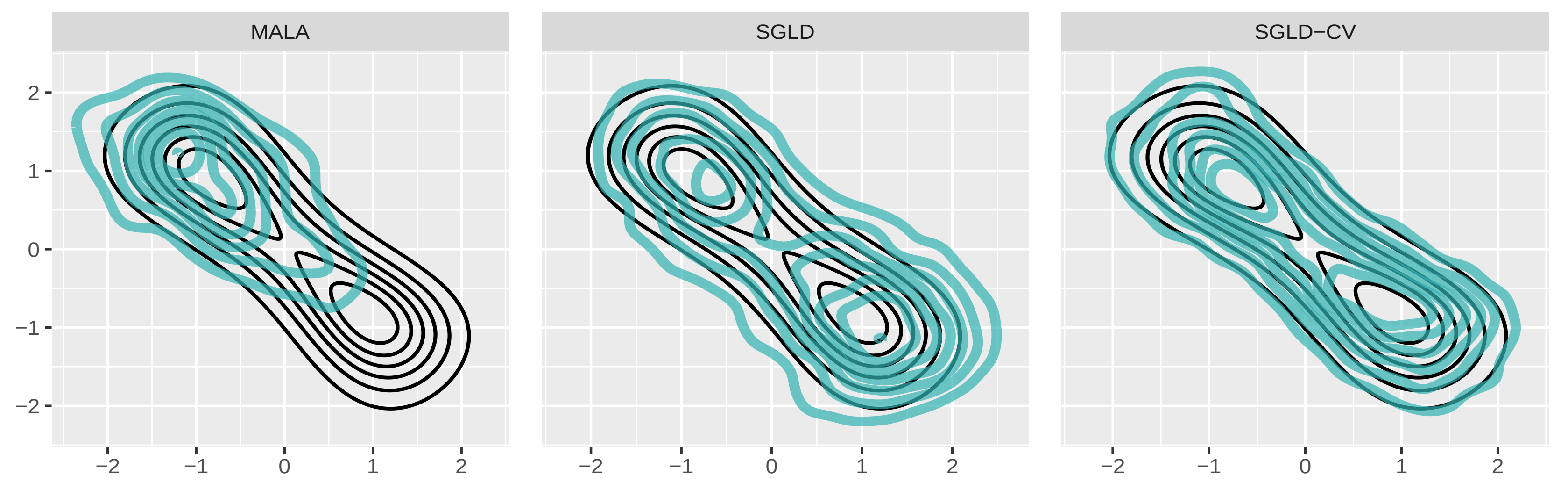

Figure 3: Comparison of the samplers for a more complicated multimodal target distribution. Each sampler was given 500 passes through the data and 20 passes of burn-in or optimisation.

Covertype Dataset

The sampling algorithms discussed above were used to fit a binary logistic regression model to the **covertype** dataset. The aim was to predict the class of tree cover from 54 forest terrain factors. The training dataset had 570 000 observations, and an additional 10 000 were used to test the model.

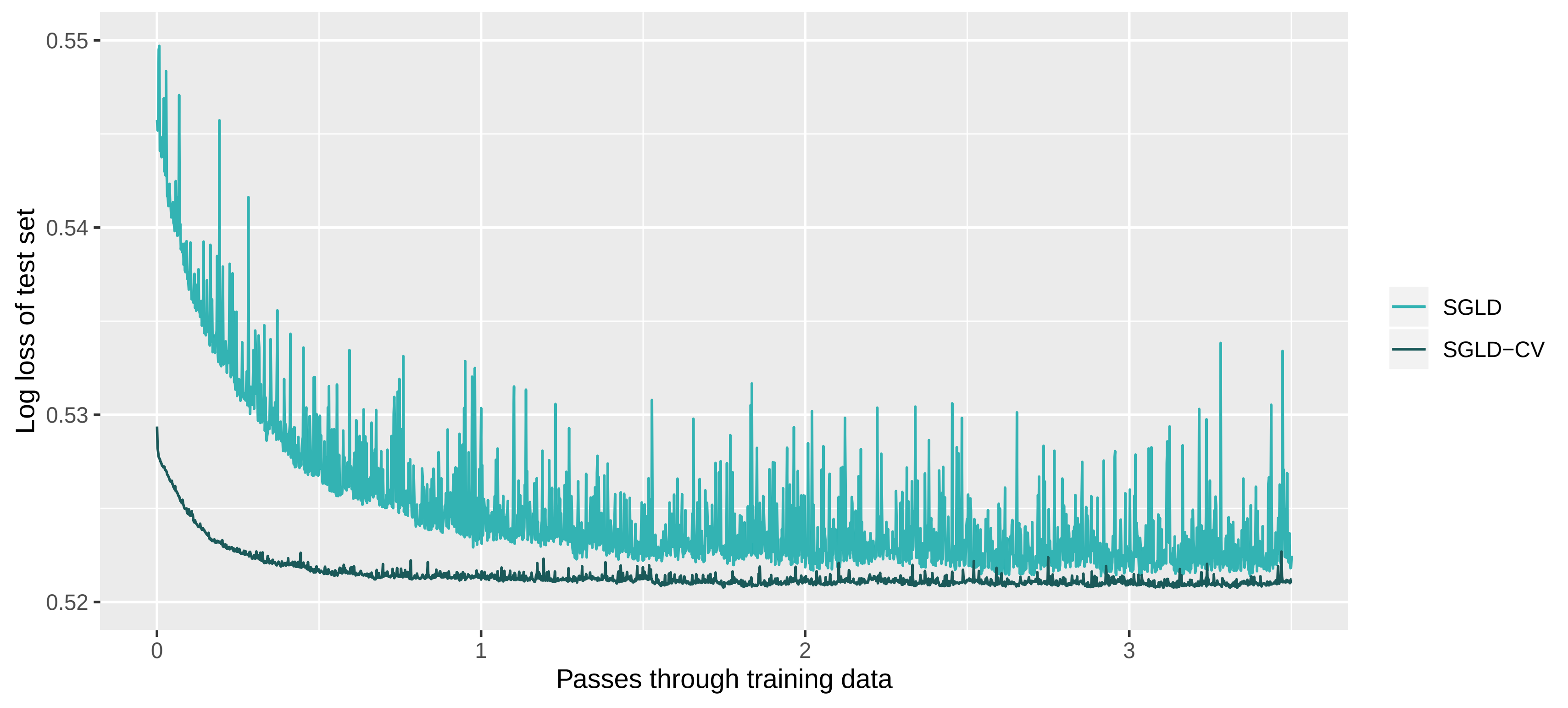

Figure 4: Logistic regression log loss comparison. Samplers given 1000 iterations of burn-in or optimisation.