

Scalable Bayesian Inference via Stochastic Gradient Langevin Dynamics

Candidate Number: xxxxxx

Document: MSc Special Topic

Course: Machine Learning, HT 2025

Lecturer: Dr Lida Kanari

Institution: The University of Oxford

Department: Mathematical Institute

1 Introduction

Modern machine learning is dominated by optimisation based methods that scale efficiently to large datasets but typically provide only point estimates of model parameters. As a result, these approaches offer limited information about predictive uncertainty, which is critical in applications such as medical decision making [1] and autonomous systems [2]. Bayesian machine learning offers a principled alternative by explicitly modelling uncertainty in the learned parameters and reducing overfitting. However, exact Bayesian inference methods that rely on sampling from the posterior distribution, such as Markov Chain Monte Carlo (MCMC), scale poorly to large datasets, limiting their practical applicability.

To address this challenge, approximate Bayesian inference at scale has largely focused on deterministic approaches such as variational inference [3], which trade posterior fidelity for computational efficiency. Sampling based methods, however, remain attractive due to their asymptotic correctness. In this report, we focus on Stochastic Gradient Langevin Dynamics (SGLD), a scalable sampling based method for Bayesian inference. Section 2 introduces Bayesian inference and uncertainty quantification. Section 3 reviews sampling based methods, culminating in SGLD. In the later sections, these methods are applied to binary classification tasks on real datasets: Section 4 focuses on Bayesian logistic regression, while Section 5 extends the approach to Bayesian neural networks.

2 Preliminaries

In Bayesian inference, uncertainty about unknown model parameters is represented using probability distributions. Let θ denote a vector of unknown parameters.

The prior distribution $p(\theta)$ encodes beliefs about θ before observing any data. After observing data \mathcal{D} , these beliefs are updated through the likelihood function $p(\mathcal{D} | \theta)$, which quantifies how probable the observed data is under a given parameter value.

Bayes theorem, combines these qualities to give the posterior distribution of θ given the data:

$$p(\theta | \mathcal{D}) = \frac{p(\mathcal{D} | \theta) p(\theta)}{p(\mathcal{D})} \propto p(\mathcal{D} | \theta) p(\theta). \quad (1)$$

Assuming the data consist of N conditionally independent observations ($\mathcal{D} = \{x_i\}_{i=1}^N$), the likelihood of the full dataset factorizes as:

$$p(\mathcal{D} | \theta) = \prod_{i=1}^N p(x_i | \theta). \quad (2)$$

Throughout this report we work with log densities rather than probabilities. Taking logarithms converts products into sums:

$$\log p(\theta | \mathcal{D}) = \log p(\theta) + \sum_{i=1}^N \log p(x_i | \theta), \quad (3)$$

this simplifies both theoretical analysis and computation.

We denote the posterior distribution by $\pi(\theta)$. From an algorithmic perspective, $\pi(\theta)$ represents the target distribution that the sampling methods in this report aim to approximate. Because posterior distributions arising from the models considered are generally intractable to sample from directly, we turn to approximate sampling techniques.

2.1 Uncertainty

One of the main benefits of Bayesian machine learning is its ability to explicitly quantify uncertainty. In many real world applications, such as medical diagnosis, autonomous systems, and scientific modelling, it is not sufficient to produce accurate point predictions alone. Understanding how confident a model is in its predictions is crucial for decision-making, risk assessment, and identifying situations in which the model may be unreliable or operating outside the distribution of the training data.

In Bayesian modelling, uncertainty is represented explicitly by a probability distribution. After observing data \mathcal{D} , uncertainty about the model parameters θ is described by the posterior distribution $p(\theta | \mathcal{D})$. Predictions have uncertainty because they depend on θ . For a new input x_i , uncertainty in the predicted outcome is, given by the posterior predictive distribution:

$$p(y_i | x_i, \mathcal{D}) = \int p(y_i | x_i, \theta) p(\theta | \mathcal{D}) d\theta. \quad (4)$$

Exact evaluation of this integral is typically infeasible. Therefore, we make use of approximate sampling methods. These methods do not return closed form expressions

for the posterior distribution. Instead, they generate a finite collection of samples:

$$\{\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(S)}\}, \quad (5)$$

which approximate draws from the posterior distribution $p(\theta | \mathcal{D})$. Once posterior samples are available, the posterior predictive distribution can be approximated using Monte Carlo integration:

$$p(y_i | x_i, \mathcal{D}) \approx \frac{1}{S} \sum_{s=1}^S p(y_i | x_i, \theta^{(s)}). \quad (6)$$

This provides a feasible way to quantify predictive uncertainty.

3 Sampling Methods

3.1 Markov Chain Monte Carlo (MCMC)

Markov Chain Monte Carlo (MCMC) methods originated with the classic paper of Metropolis et al. (1953) [4]. MCMC algorithms produce a sequence of correlated, approximate samples from a target distribution $\pi(\theta)$ by constructing a Markov chain whose stationary distribution is $\pi(\theta)$. Proposed samples are accepted or rejected according to the Metropolis–Hastings algorithm. An example of how the algorithm operates with simple Gaussian proposals is shown below.

1. Propose a move:

$$\theta' \sim \mathcal{N}(\theta^t, \sigma^2). \quad (7)$$

2. Compute the acceptance probability:

$$\alpha(\theta^t, \theta') = \min\left(1, \frac{\pi(\theta')}{\pi(\theta^t)}\right). \quad (8)$$

3. Accept or reject the proposal:

$$\theta^{t+1} = \begin{cases} \theta' & \text{with probability } \alpha(\theta^t, \theta'), \\ \theta^t & \text{with probability } 1 - \alpha(\theta^t, \theta'). \end{cases} \quad (9)$$

MCMC methods are asymptotically exact meaning, as the number of iterations tends to infinity, the distribution of the generated samples converges to the target distribution $\pi(\theta)$. For this reason, MCMC is often considered the gold standard or Bayesian posterior sampling. We call the algorithm outlined above Random-Walk Metropolis (RWM) throughout the rest of this report.

3.2 Langevin Dynamics

One issue with simple Gaussian proposals is that they scale very poorly in high-dimensional parameter spaces. As the dimension of θ increases, such proposals require increasingly small step sizes to maintain reasonable acceptance rates, which leads to slow exploration of the target distribution.

To address this issue, we consider a class of MCMC technique based on Langevin dynamics [5]. These methods improve sampling efficiency by incorporating gradient information into the proposal, guiding proposals toward regions of higher probability density. The proposed update is now of the form:

$$\theta' = \theta_t + \frac{\epsilon_t}{2} \underbrace{\left(\nabla_{\theta} \log p(\theta_t) + \sum_{i=1}^N \nabla_{\theta} \log p(x_i | \theta_t) \right)}_{g_t} + \eta_t, \quad (10)$$

where $g_t = \nabla_{\theta} \log \pi(\theta_t)$ denotes the gradient of the log posterior at the current state, ϵ_t is the step size at each iteration (constant for this algorithm), and $\eta_t \sim \mathcal{N}(0, \epsilon_t)$ is the noise term. This proposal is inspired by the first order discretization of the Langevin diffusion equation.

As with random-walk proposals, a Metropolis-Hastings step is required. However, unlike simple Gaussian random-walk proposals, Langevin based proposals are asymmetric due to the inclusion of gradient information. As a result, the proposal density must be explicitly included in the acceptance probability. We define the proposal distribution as $q(\theta' | \theta_t) = \mathcal{N}(\theta' | \theta_t + \frac{\epsilon_t}{2} g_t, \epsilon_t)$. The resulting Metropolis–Hastings acceptance probability is given by:

$$\alpha(\theta^t, \theta') = \min\left(1, \frac{\pi(\theta') q(\theta^t | \theta')}{\pi(\theta^t) q(\theta' | \theta^t)}\right). \quad (11)$$

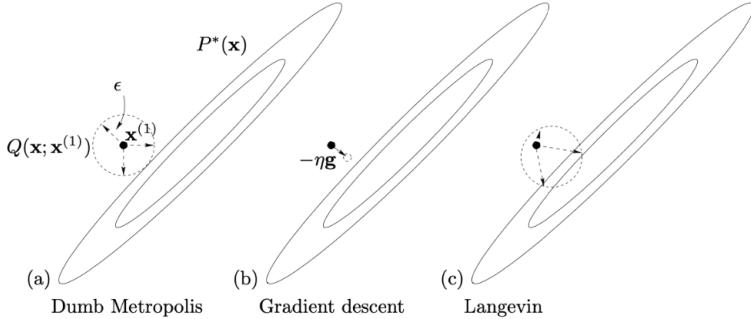


Figure 1: Illustration of Langevin dynamics in a two-dimensional target distribution, contrasted with random-walk Metropolis and deterministic gradient descent. Langevin dynamics combines gradient information with stochastic noise to enable efficient exploration of the target distribution. Reprinted from MacKay (2003) [6].

We refer to this method as the Metropolis–Adjusted Langevin Algorithm (MALA) throughout this report. A major limitation of MALA, and by extension RWM, is that each update step can be computationally expensive, particularly for large scale datasets. This is because both methods require evaluation of the full data log posterior at every iteration, and MALA additionally requires computation of its gradient. The per-iteration computational cost of these methods is discussed in detail later in the report.

3.3 Stochastic Gradient Langevin Dynamics (SGLD)

Introduced by Welling and Teh (2011) [7], Stochastic Gradient Langevin Dynamics (SGLD) seeks to address the poor scalability of traditional MCMC methods by combining ideas from stochastic optimization and Bayesian posterior sampling. The central idea of SGLD is to replace the exact gradient of the log posterior with a stochastic approximation computed using a randomly selected minibatch of the data.

Given a dataset of size N and a minibatch of size n , we use a minibatch estimate of the form:

$$\log \hat{\pi}_M(\theta) = \log p(\theta) + \frac{N}{n} \sum_{i=1}^n \log p(x_i | \theta). \quad (12)$$

This provides an unbiased estimate of the full data log posterior gradient $\nabla \pi(\theta)$. Our update step now looks like:

$$\theta_{t+1} = \theta_t + \frac{\epsilon_t}{2} \left(\nabla_\theta \log p(\theta_t) + \frac{N}{n} \sum_{i=1}^n \nabla_\theta \log p(x_i | \theta_t) \right) + \eta_t. \quad (13)$$

This introduces a new challenge, in that, calculating the Metropolis–Hastings acceptance ratio still requires evaluating the full data log posterior, eliminating the computational advantage of minibatching. The solution is to skip the Metropolis-Hastings accept-reject step entirely. Doing so, requires placing additional assumptions on the step size sequence $\{\epsilon_t\}$. The required conditions on the step size are as follows:

$$\sum_{t=1}^{\infty} \epsilon_t = \infty \quad \text{and} \quad \sum_{t=1}^{\infty} \epsilon_t^2 < \infty. \quad (14)$$

Typically, step sizes are chosen to decay polynomially $\epsilon_t = a(b+t)^{-\gamma}$ with $\gamma \in (0.5, 1]$. Welling and Teh (2011) [7] provide an intuitive argument for why θ_t converge in distribution to samples from the posterior as $t \rightarrow \infty$. As $\epsilon_t \rightarrow 0$, the injected Gaussian noise η_t dominates the stochastic noise from minibatch gradient estimation, so the update rule effectively reduces to Langevin Dynamics. The second observation is that as $\epsilon_t \rightarrow 0$, the discretization error of Langevin dynamics will be negligible so that the Metropolis-Hastings rejection probability approaches 0. Consequently, the accept reject step can be safely omitted, and all proposals are accepted.

3.4 Tuning Parameters

In order for comparisons between sampling algorithms to be meaningful, it is essential that the tuning parameters of each method are chosen carefully, as these parameters have a substantial effect on mixing behaviour and computational efficiency. For RWM this involves selecting the proposal variance σ , for the MALA the step size ϵ , and for SGLD the step size schedule $\{\epsilon_t\}$.

Suppose we are given a dataset on which we want to implement these sampling algorithms, the procedure goes as follows. The dataset is partitioned into three disjoint subsets: a training set, a validation set, and a test set. The training data is used to define the posterior distribution over model parameters. The validation data is used exclusively to tune algorithm specific parameters, while the test set is reserved for final performance evaluation and is not used during model fitting or sampler tuning.

Following standard MCMC practice, we tune these parameters by monitoring the Metropolis–Hastings acceptance rate on the validation set. In high-dimensional settings, the optimal acceptance rate is approximately 0.234 for RWM [8] and approximately 0.574 for MALA [9]. We therefore perform a grid search over plausible param-

eter values and select those that yield acceptance rates close to these optimal values.

Choosing an appropriate step size schedule for SGLD is a less well defined process, as the algorithm does not include an acceptance–rejection step. Following Welling and Teh (2011) [7], we adopt a polynomially decaying step size schedule of the form $\epsilon_t = \epsilon_0 t^{-\gamma}$ with $\gamma = 0.55$, which satisfies the standard conditions required for convergence.

While the decay rate γ is fixed, the initial step size ϵ_0 must be chosen carefully. Theoretical analyses of SGLD suggest that the algorithm should operate in two distinct regimes. In the initial phase, variability arising from stochastic minibatch gradients dominates the injected Langevin noise, and the algorithm behaves similarly to stochastic gradient ascent. In the later phase, as the step size decreases, the injected Gaussian noise dominates the stochastic gradient noise, so the algorithm will imitate the MALA algorithm. A desirable step size schedule allows the algorithm transition smoothly between the two.

To select a suitable value of ϵ_0 , we examine the relative magnitudes of the stochastic gradient noise and the injected Langevin noise over the course of the algorithm. In particular, we compare the variance from stochastic minibatch gradients, $\nabla \hat{\pi}_M(\theta)$, with the variance of the injected noise term η_t . We select ϵ_0 such that initially the stochastic minibatch noise dominates and at a later phase the injected noise dominates.

Finally, it is important to note that samples from SGLD are retained only after this transition has occurred. The initial period, during which the algorithm transitions from optimisation-dominated behaviour to Langevin dynamics, is treated as a burn-in phase and discarded in subsequent analysis.

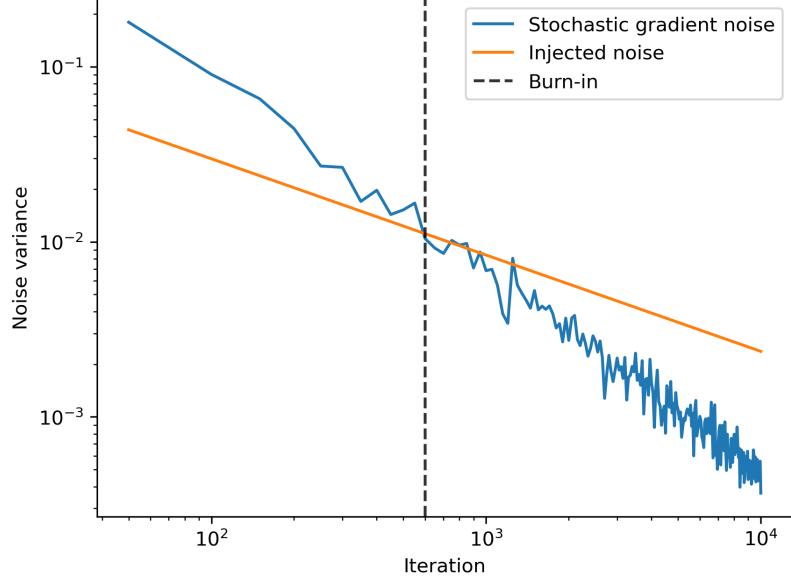


Figure 2: Total variance of the stochastic gradient noise (blue) and the injected noise (orange) during SGLD. As the step size decays, the injected noise eventually dominates the stochastic gradient noise. Samples are retained only after this crossover, which occurs after the burn-in period.

For completeness, we note that both MALA and RWM also require a burn-in period during which initial samples are discarded. For these methods, the burn-in period corresponds simply to the initial phase before the Markov chain reaches its target distribution. The length of the burn-in period in these cases is assessed by examining trace plots, as shown in Figure 3.

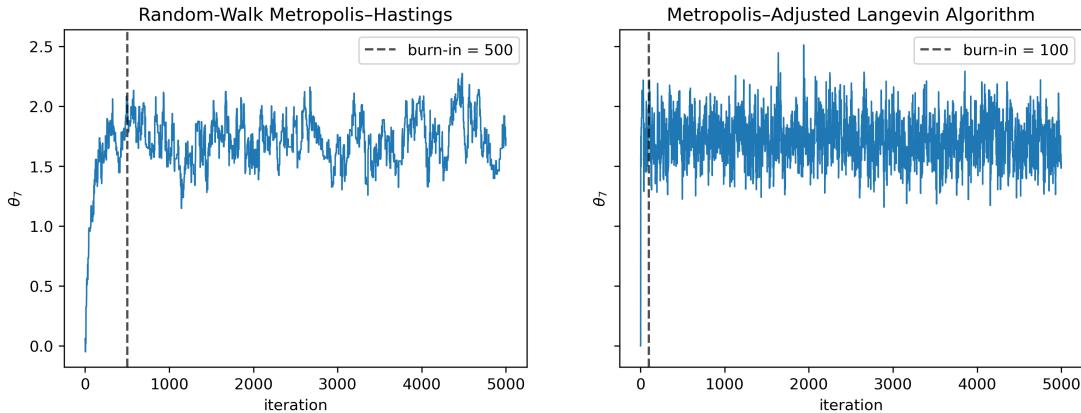


Figure 3: Trace plots of burn-in period for RWM and MALA

4 Logistic Regression

4.1 Implementation

To validate several properties of the sampling algorithms described above, we apply them to a binary logistic regression classification task. We use the Australian Credit Approval dataset, a standard binary classification benchmark originally from the UCI Machine Learning Repository [10]. The dataset contains $N = 690$ observations with $D = 14$ input dimensions and is sufficiently small to allow all algorithms to be implemented efficiently.

To implement the sampling algorithms, we require explicit expressions for the log posterior distribution and its gradient in the case of Bayesian logistic regression. The log posterior consists of a prior term and a log likelihood term. Throughout, we use the logistic (sigmoid) function defined as $S(z) = \frac{1}{1+e^{-z}}$.

Prior

We place a Gaussian prior on the parameters:

$$p(\theta) = \mathcal{N}(0, \sigma_0^2 I), \quad (15)$$

where $\theta \in \mathbb{R}^D$ and σ_0^2 denotes the prior variance. In our experiments, we set $\sigma_0 = 1$. The gradient of the log prior takes the simple form:

$$\nabla_\theta \log p(\theta) = -\frac{1}{\sigma_0^2} \theta. \quad (16)$$

Log Likelihood

We encode class labels as $y_i \in \{-1, +1\}$. For a single data point (x_i, y_i) , the logistic regression likelihood is given by:

$$p(y_i | x_i, \theta) = S(y_i \theta^\top x_i). \quad (17)$$

Taking the gradient of the log likelihood yields:

$$\nabla_\theta \log p(y_i | x_i, \theta) = (1 - S(y_i \theta^\top x_i)) y_i x_i. \quad (18)$$

Summing over all observations, the gradient of the log likelihood for the full dataset can be written compactly as:

$$\nabla_\theta \log p(y | X, \theta) = X^\top (y \odot (1 - \sigma(y \odot (X\theta)))), \quad (19)$$

where \odot denotes element-wise multiplication.

4.2 Results

For each sampling algorithm, model parameters were first fitted following the procedures described in the previous section. After tuning, the following hyperparameters were used. MALA employed a constant step size of $\epsilon = 2 \times 10^{-2}$. RWM used a Gaussian proposal with standard deviation $\sigma = 0.1$. For SGLD, the initial step size was set to $\epsilon_0 = 5 \times 10^{-2}$ with a minibatch size of 10.

For all sampling methods, a total of 5000 samples were generated. To mitigate the effect of initialisation bias, an initial burn-in period was discarded. The burn-in length was set to 100 samples for MALA, 500 samples for RWM, and 2000 samples for SGLD, this also reflects the differing mixing behaviour of the algorithms.

As a baseline for comparison, we also trained a conventional optimisation based logistic regression model using `scikit-learn`'s `LogisticRegression`. This model produces a single point estimate of the parameters via numerical optimisation. For the Bayesian sampling methods, classification was performed using the posterior mean parameter vector, obtained by averaging the retained samples after burn-in. As shown in Table 1, despite their fundamentally different inference procedures, all methods achieved comparable classification performance on the test dataset.

Method	Test Accuracy
Optimisation	0.8696
MALA	0.8696
RWM	0.8623
SGLD	0.8623

Table 1: Test set classification accuracy for optimisation based logistic regression and Bayesian sampling methods on the Australian Credit Approval dataset

4.3 Sample Quality

One of the most important checks when using approximate samplers is verifying that the sampler is producing representative samples from the target posterior distribution $\pi(\theta)$. If a sampler fails to explore the posterior adequately, any future analysis, such as uncertainty quantification may be misleading. In this context, mixing refers to how efficiently a Markov chain traverses the parameter space, with good mixing

indicating fast exploration of high probability regions.

As discussed previously, RWM can suffer from poor mixing in high-dimensional parameter spaces. In order to maintain reasonable acceptance rates, the proposal variance must often be chosen very small, which can result in slow exploration of the parameter space. Conversely, MALA uses gradient information to guide proposals, typically leading to more efficient mixing. SGLD introduces an additional source of approximation through the use of stochastic gradients, and as a result, SGLD samples may exhibit bias relative to exact MCMC methods. The decaying step size schedule may also lead to poor mixing.

While there exist quantitative diagnostics for assessing sampler performance, a useful initial check is to examine marginal posterior distributions of selected parameters. If different sampling algorithms are correctly targeting the same posterior distribution, their marginal distributions should broadly agree. Here, a parameter refers to a single component of the vector θ .

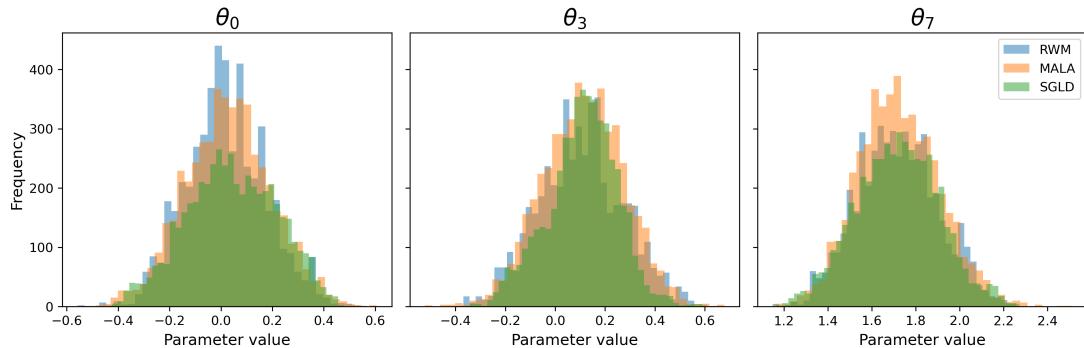


Figure 4: Comparison of the marginal posterior distributions of three parameters, θ_0 , θ_3 , and θ_7 obtained using RWM, MALA and SGLD

Figure 4 compares the marginal posterior distributions of three randomly selected parameters, θ_0 , θ_3 , and θ_7 , obtained using RWM, MALA, and SGLD. For all three parameters, the distributions produced by the different methods are closely aligned in both their mean and variance. This visual agreement provides evidence that the samplers are capturing a similar posterior structure.

Although such visual diagnostics are not sufficient to guarantee good mixing or convergence to the target distribution, they serve as an important sanity check before more detailed quantitative analyses are performed. A more rigorous way to assess

whether a sampler is producing effective samples is to examine its effective sample size (ESS).

Effective sample size accounts for the fact that samples generated by MCMC methods are correlated. An MCMC sampler produces a sequence of samples $\{\theta_1, \theta_2, \theta_3, \dots\}$, but successive samples are generally not independent, for example, θ_t is often similar to θ_{t+1} . As a result, a chain of 5000 MCMC samples typically contains less information than 5000 independent draws from the posterior distribution. ESS quantifies this by answering the question: *how many independent samples is the correlated chain equivalent to?*

ESS is defined as:

$$\text{ESS} = \frac{N}{1 + 2 \sum_{k=1}^K \rho_k}, \quad (20)$$

where N denotes the total number of samples and ρ_k is the autocorrelation, which measures the dependence between samples k iterations apart. If the chain exhibits strong autocorrelation, the values of ρ_k decay slowly as k increases, resulting in a large denominator and hence a small ESS. Conversely, rapidly decaying autocorrelation leads to an ESS close to N , indicating efficient exploration of the posterior distribution.

Method	median ESS	min ESS
MALA	627.24	103.06
RWM	85.84	27.26
SGLD	41.4	12.26

Table 2: Median and minimum effective sample size (ESS) across parameter dimensions for MALA, RWM, and SGLD, computed using 5000 post burn-in samples

We calculated ESS using the ArviZ python library [11] for the 5000 post burn-in samples generated by each sampling method. For each algorithm, ESS was evaluated separately for every parameter dimension, and the median and minimum ESS across dimensions are reported in Table 2. The median ESS reflects typical mixing behaviour, while the minimum ESS highlights the slowest mixing parameter, which often determines overall sampler efficiency.

The results are generally consistent with our theoretical expectations. Despite the relatively low dimensional parameter space ($D = 14$), MALA achieves substantially higher ESS values than RWM. The improved ability to efficiently explore the posterior distribution in higher-dimensional spaces was the primary advantage of MALA over RWM that we outlined at the beginning.

SGLD performs significantly worse than both MALA and RWM in terms of ESS. However, this comparison should be interpreted with caution. Unlike MALA and RWM, SGLD employs a decaying step size schedule, which causes the magnitude of parameter updates to shrink over time. As the step size decreases, successive samples become increasingly similar, resulting in strong autocorrelation and poor mixing, as reflected by the low ESS values. This behaviour is intrinsic to classical SGLD and does not necessarily indicate a failure of the algorithm, but rather a trade-off between asymptotic correctness and sampling efficiency.

While a polynomially decaying step size is theoretically required for convergence guarantees in SGLD, it is often overly conservative in practice. In many real world applications, alternative step size schemes or modified algorithms, are preferred as they can substantially improve mixing while retaining scalability. One such variant, cyclical SGLD, is discussed in subsequent sections.

Finally, it is worth noting that effective sample size per second is often a more informative metric when comparing samplers with different per-iteration computational costs. In principle, this metric better captures SGLD’s strengths, as individual SGLD updates are computationally cheap. However, in the present setting the dataset is sufficiently small that this advantage is limited, and SGLD does not outperform MALA or RWM even when accounting for runtime.

4.4 Uncertainty

We now examine predictive uncertainty, which is a central advantage of Bayesian sampling methods over optimisation based approaches.

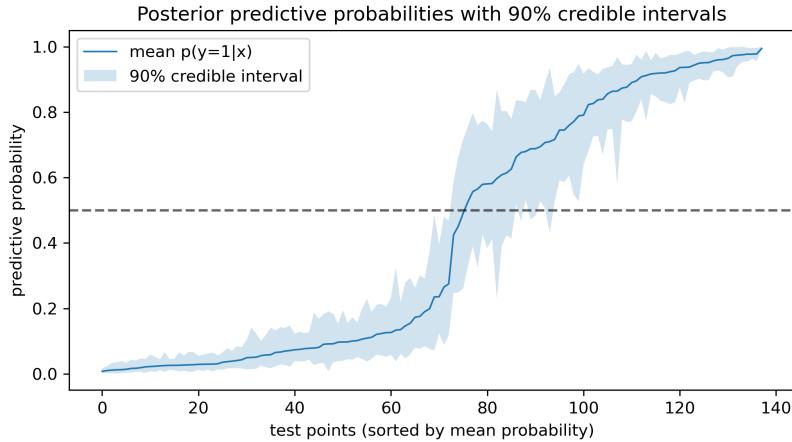


Figure 5: Posterior predictive probabilities for the test data obtained using SGLD. The solid line shows the mean predictive probability for each test point, while the shaded region denotes the 90% credible interval. Test points are ordered by increasing mean probability.

Figure 5 shows the posterior predictive probabilities for the logistic regression model evaluated on the test set. Since SGLD, MALA, and RWM produced nearly identical posterior distributions for this dataset, only the SGLD results are shown.

The key observation is that predictive uncertainty varies across test points, even when the mean predicted probabilities are similar. For some inputs, predictions are highly stable across posterior samples, resulting in narrow credible intervals. For others, small changes in the model parameters lead to large variations in predicted probability, producing wide credible intervals. This variability reflects uncertainty in the model parameters from the data and cannot be captured by point estimates alone. Standard optimisation-based methods, which rely on a single parameter estimate, cannot express this distinction between stable and unstable predictions.

4.5 Larger Datasets

The principal advantage SGLD has over RWM and MALA is in its ability to scale to large datasets. This advantage becomes clear by comparing the per-iteration computational cost of each algorithm.

Random-Walk Metropolis

At each iteration, RWM proposes a new parameter value θ' and accepts or rejects it using the Metropolis-Hastings acceptance probability. Evaluating this probability

requires computing the log posterior at the proposed state,

$$\log \pi(\theta') = \log p(\theta' | \mathcal{D}) = \log p(\theta') + \sum_{i=1}^N \log p(y_i | x_i, \theta'), \quad (21)$$

which involves a full pass through the dataset. Evaluating the log likelihood contribution of a single datapoint has a computation cost of $\mathcal{O}(d)$ where d is the dimension of the parameter vector, then the per-iteration cost of RWM is:

$\text{Cost per RWM iteration} = \mathcal{O}(Nd).$

Metropolis-Adjusted Langevin Algorithm

At each iteration, MALA proposes a new parameter value θ' using the update rule in Eqn 10. This involves computation of the gradient of the log posterior, given by:

$$\nabla_\theta \log p(\theta | \mathcal{D}) = \nabla_\theta \log p(\theta) + \sum_{i=1}^N \nabla_\theta \log p(y_i | x_i, \theta). \quad (22)$$

which requires a full pass through the dataset and therefore has a computation cost of $\mathcal{O}(Nd)$. Furthermore, the Metropolis-Hastings correction requires evaluating both the log posterior and the gradient at the proposed state θ' , each of which again involves a full data computation $\mathcal{O}(Nd)$. Consequently, one MALA iteration is more expensive than RWM as it requires multiple full passes through the dataset. However, up to constant factors, the overall computational complexity of one MALA step is:

$\text{Cost per MALA iteration} = \mathcal{O}(Nd).$

Stochastic Gradient Langevin Dynamics

At each iteration, SGLD updates the parameter θ_{t+1} using the update rule in Eqn 13. In contrast to MALA, the full data gradient of the log posterior is replaced with an unbiased stochastic estimate computed using a minibatch of data:

$$\nabla_\theta \widehat{\log p(\theta | \mathcal{D})} = \nabla_\theta \log p(\theta) + \frac{N}{B} \sum_{i \in \mathcal{B}} \nabla_\theta \log p(y_i | x_i, \theta), \quad (23)$$

where \mathcal{B} denotes a minibatch of size B . Evaluating this stochastic gradient requires only a single pass through the minibatch rather than the full dataset. Therefore the computational cost of one SGLD iteration is:

$\text{Cost per SGLD step} = \mathcal{O}(Bd).$

Importantly, after the burn-in stage we accept every sample produced by the SGLD algorithm, allowing us to skip the expensive Metropolis-Hastings step.

Comparing the per-iteration computational costs, we see that SGLD reduces the dependence on the dataset size from N to the minibatch size B , with $B \ll N$. When N is large, as is typical in modern applications such as Bayesian neural networks, this reduction is critical. Classical sampling methods such as RWM and MALA become computationally impractical due to their reliance on full data likelihood and gradient evaluations, whereas SGLD remains computationally feasible.

This scalability is the primary reason SGLD is preferred for large datasets. By contrast, when the dataset is small and full data gradients are computationally affordable, methods such as RWM and MALA are generally preferable, as they target the exact posterior distribution. Therefore, in the remainder of this report, when discussing applications to Bayesian neural networks, we focus exclusively on SGLD, as MALA and RWM have limited practical use in these settings unless the dataset is very small.

5 BNN

5.1 Cyclic Learning Rate

In our previous logistic regression example, the posterior distribution was largely unimodal and well behaved. In contrast, the posterior distribution over a neural network’s parameters (weights) is typically high-dimensional and highly multimodal. When using a standard decaying step size in SGLD, the sampler rapidly converges to a single mode and fails to adequately explore the posterior distribution. This raises the question: *how can SGLD be modified to efficiently explore highly multimodal parameter spaces?*

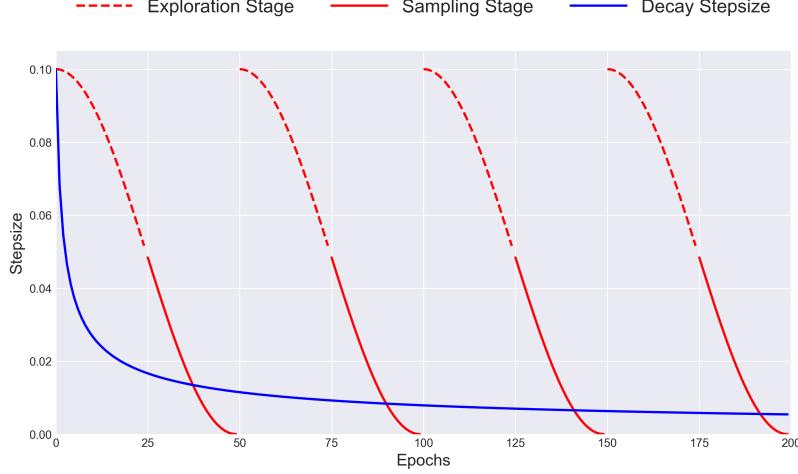


Figure 6: Illustration of cyclic step size schedule (red) compared with a traditional polynomially decaying step size schedule (blue), adapted from Zhang et al. (2019) [12].

Following the work of Zhang et al. (2019) [12], we introduce a cyclical learning rate schedule for SGLD, referred to as cyclical SGLD (cSGLD). The above Figure 6 shows cSGLD in action, which operates in two distinct stages:

- (i) **Exploration:** when the step size is large (dashed red curves), this stage acts as an effective burn-in, encouraging the sampler to take large moves and escape local modes.
- (ii) **Sampling:** when the step size is small (solid red curves), the sampler explores a single local mode more carefully. Samples collected during this stage are retained and used for local posterior distribution estimation.

In terms of the specific formula, we propose a cyclical cosine step size schedule. The step size at iteration k is defined as:

$$\epsilon_k = \frac{\epsilon_0}{2} \left[\cos\left(\frac{\pi \text{ mod}(k - 1, \lceil K/M \rceil)}{\lceil K/M \rceil}\right) + 1 \right], \quad (24)$$

where ϵ_0 is the initial step size, M is the number of cycles and K is the total number of iterations. The modulo operation resets the cosine schedule at the beginning of each cycle, causing the step size to vary smoothly between ϵ_0 and 0 within each cycle. This schedule preserves asymptotic correctness by ensuring a vanishing step size, while also mitigating collapse to a single mode by periodically reintroducing larger step sizes.

Using the code provided by Zhang et al.¹, in Figure 7 we demonstrate clearly where this approach is effective. We consider sampling from a highly multimodal two-dimensional target distribution: a mixture of 25 Gaussians arranged on a grid. This example serves as a controlled setting for evaluating the ability of different algorithms to explore multiple modes. With the same computational budget of 50,000 samples, standard SGLD discovered only one of the 25 modes. In contrast, cSGLD successfully explores a much larger portion of the target distribution, recovering multiple modes and producing a density that more closely resembles the true target.

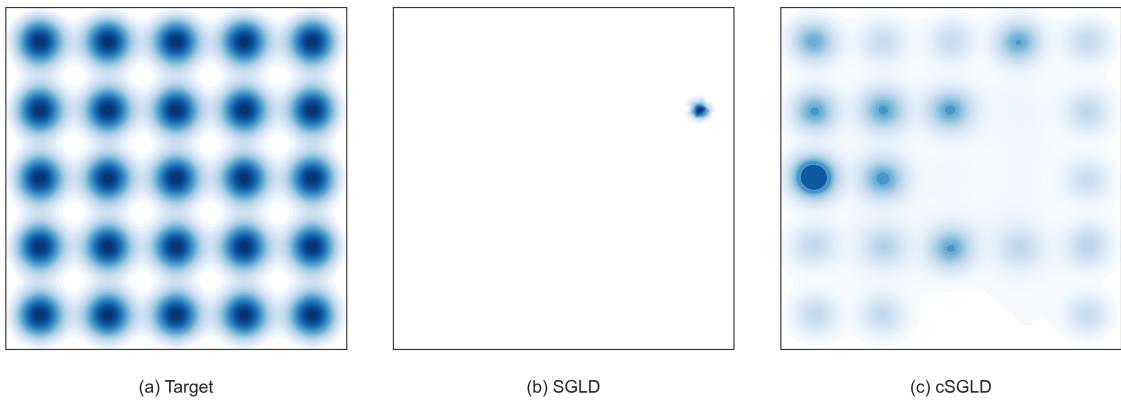


Figure 7: Sampling from a mixture of 25 Gaussians. With a budget of 50K samples, traditional SGLD discovers only one of the 25 modes, whereas the proposed cSGLD explores a substantially larger portion of the target distribution, inspired by Zhang et al. [12].

This experiment demonstrates the key advantage of cyclical step sizes in multimodal settings. By periodically reintroducing large step sizes, cSGLD avoids premature convergence to a single mode and achieves significantly improved exploration compared to standard SGLD. This behaviour motivates the use of cyclical SGLD in Bayesian neural networks, where multimodality and poor mixing are particularly severe.

5.2 Quick Result

We next apply SGLD to a simple feedforward neural network for a binary classification task. We generated a dataset using the `sklearn.datasets` function `make_moons`, which produces a two-dimensional, nonlinearly separable dataset. As a result, a linear model such as logistic regression is not suited for this problem.

¹<https://github.com/ruqizhang/csgmcmc>

To address this, we employ a fully connected feedforward neural network with ReLU activation functions. The network architecture consisted of two hidden layers with 100 and 10 neurons, respectively, followed by a single output unit. Despite its modest size, this network contains a total of 1,321 trainable parameters (weights and biases). Before performing Bayesian inference, we initialize the network parameters at a maximum a posteriori (MAP) estimate, obtained by minimizing the negative log posterior. This initialization places the parameters in a high probability region of the posterior distribution, from which the SGLD sampler can efficiently explore the surrounding parameter space.

The procedure of applying SGLD to a neural network, is very similar to the logistic regression case. The main difference is that the model output is no longer a linear function of the input, $\hat{y}_i = \theta^\top x_i$, but instead the output of a nonlinear neural network, $\hat{y}_i = f_\theta(x_i)$. Since we are performing binary classification, the underlying likelihood model remains the same as in logistic regression.

As before, we place a standard Gaussian prior over the model parameters, $p(\theta) = \mathcal{N}(0, I)$. While the resulting posterior has the same overall form as in logistic regression, the gradients required by SGLD are now obtained by differentiating through the neural network. In practice, this is handled automatically using backpropagation via the Python call `lp.backward()` on the minibatch log posterior.

After obtaining the MAP estimate, we run the SGLD algorithm to generate samples from the posterior distribution over network parameters. In our experiments, we use a batch size of 5 and use the cyclical learning-rate schedule defined in Eqn. 24, with $\epsilon_0 = 0.02$, 200 cycles, and 50 SGLD updates per cycle. One sample is collected at the end of each cycle, while the preceding 49 updates are treated as a within cycle burn-in. Although this strategy may appear conservative, it helps reduce discretization bias and autocorrelation between successive samples. The resulting 200 posterior samples are used to estimate predictive uncertainty and to visualise the posterior predictive distribution.

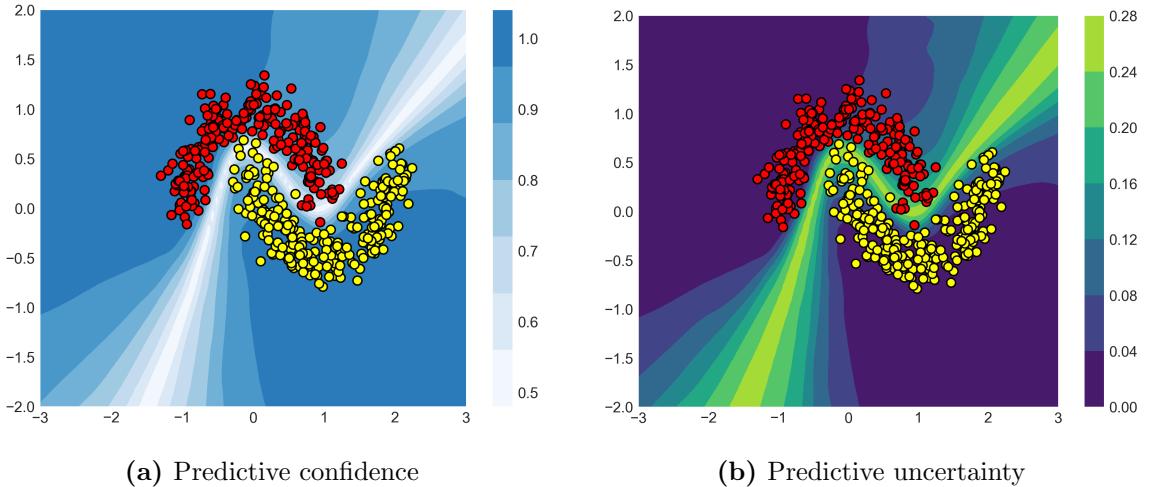


Figure 8: Posterior predictive behaviour of the Bayesian neural network on the two moons dataset.

Figure 8 shows the posterior predictive confidence and uncertainty for the two moons classification problem. These quantities are derived from the posterior predictive distribution 6, which is approximated using samples obtained via SGLD.

This two dimensional example provides a clear visual illustration of the key advantage of the Bayesian approach. Both the confidence and uncertainty plots show similar behaviour. Uncertainty is highest near the decision boundary and in regions where training data is sparse.

This contrasts, optimisation based methods, which often produce overly confident predictions even far from the observed data. By accounting for uncertainty in the model parameters, the Bayesian approach naturally expresses increased uncertainty in data scarce regions, reducing overconfidence and mitigating overfitting. We also applied SGLD to a three spiral multiclass classification task and it showed similar behaviour (see Appendix).

6 Discussion

In this report, we investigated Stochastic Gradient Langevin Dynamics (SGLD) as a scalable Bayesian inference method. Building on classical MCMC techniques derived from Langevin dynamics, SGLD replaces full data gradients with unbiased minibatch estimates, making posterior sampling feasible in settings where traditional methods

become computationally impractical. We then showed that, under a suitable step size schedule, SGLD produces samples that accurately approximate the target posterior distribution. Finally, we applied SGLD to Bayesian logistic regression and Bayesian neural networks, demonstrating its effectiveness for scalable Bayesian classification.

There are several possible directions for future work. One natural direction is to compare SGLD with other scalable Bayesian machine learning methods, particularly deterministic approaches such as variational inference, to better understand trade-offs in computational efficiency, uncertainty representation, and accuracy. Another direction is to investigate modern extensions of SGLD that aim to improve mixing, which was observed to be a limitation in some of our experiments. For example, preconditioned SGLD (pSGLD) [13] can improve mixing by adapting updates to ill-conditioned posteriors, while stochastic gradient Hamiltonian Monte Carlo (SGHMC) [14] introduces momentum and friction that enable better posterior exploration. Finally, applying SGLD to larger and more complex real world datasets would provide further insight into its practical strengths and limitations.

References

- [1] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, 76:243–297, 2021.
- [2] Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.
- [3] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.
- [4] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.
- [5] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.
- [6] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.
- [7] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.
- [8] Andrew Gelman, Walter R Gilks, and Gareth O Roberts. Weak convergence and optimal scaling of random walk metropolis algorithms. *The annals of applied probability*, 7(1):110–120, 1997.
- [9] Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling of discrete approximations to langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, 1998.
- [10] D. Dua and C. Graff. Australian credit approval dataset. UCI Machine Learning Repository, 2019.

- [11] Ravin Kumar, Colin Carroll, Ari Hartikainen, and Osvaldo Martin. Arviz a unified library for exploratory analysis of bayesian models in python. *Journal of Open Source Software*, 4(33):1143, 2019.
- [12] Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient mcmc for bayesian deep learning. *arXiv preprint arXiv:1902.03932*, 2019.
- [13] Chunyuan Li, Changyou Chen, David Carlson, and Lawrence Carin. Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.
- [14] Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691. PMLR, 2014.

7 Appendix

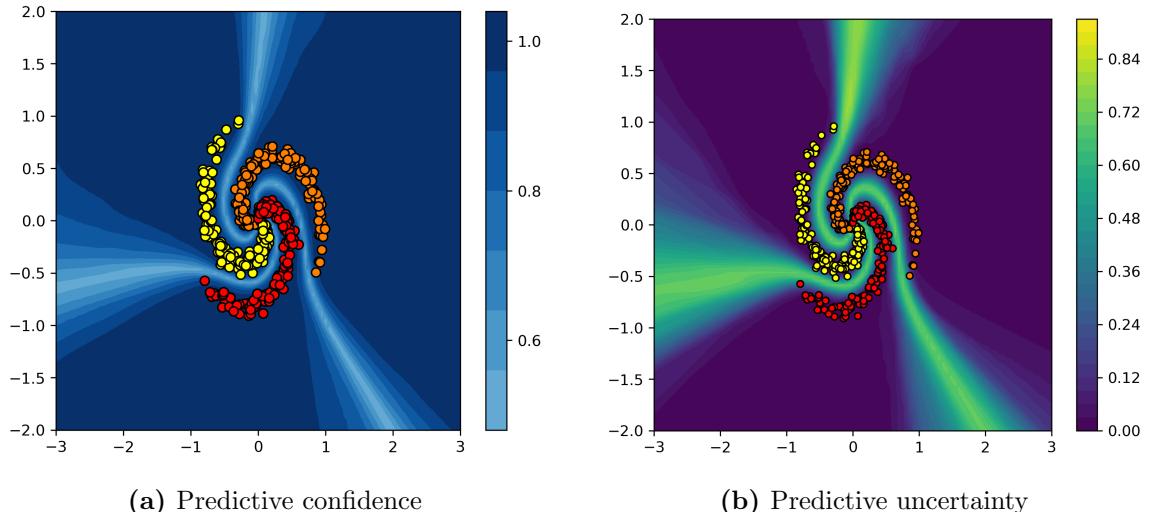


Figure 9: Posterior predictive behaviour of the Bayesian neural network applied to a three-spiral multiple classification task.

8 Extra (to be deleted)

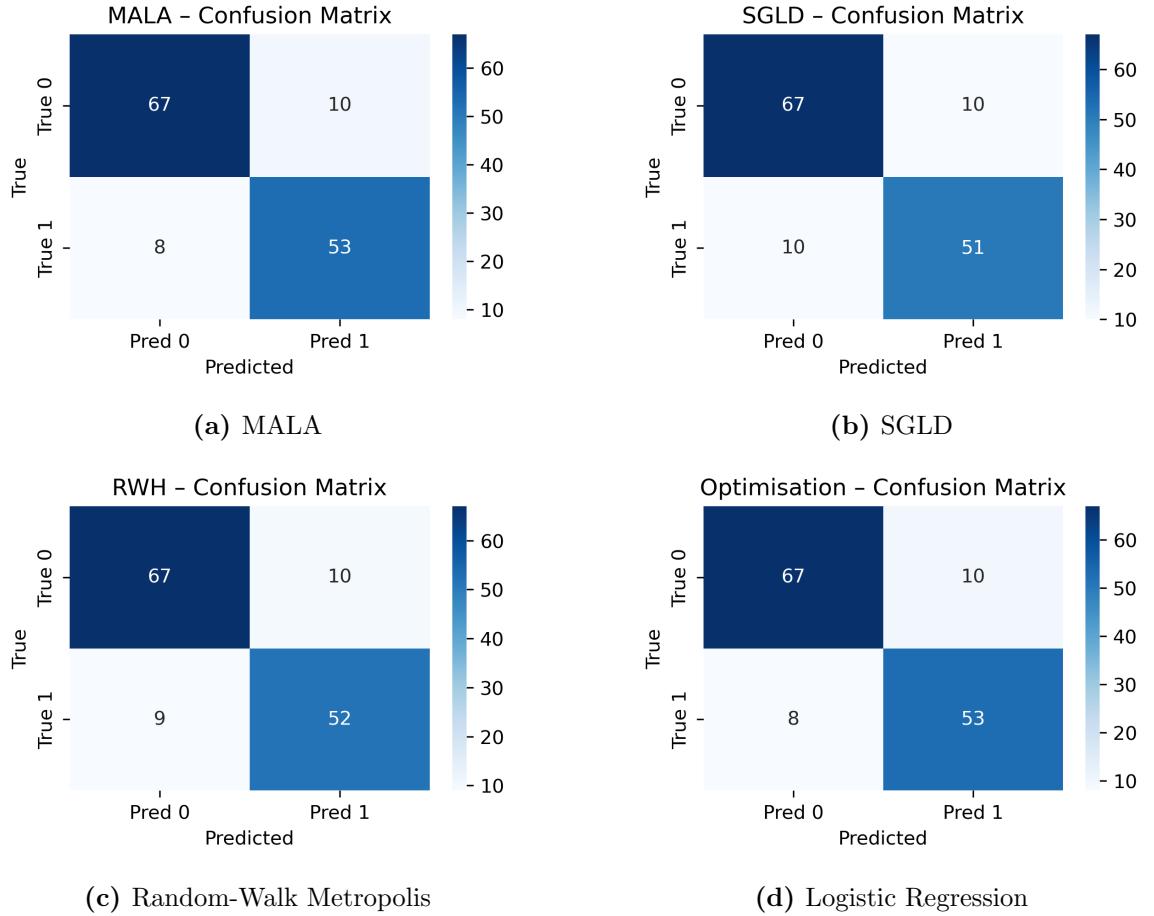


Figure 10: Confusion matrices for Bayesian sampling methods and optimisation-based logistic regression. Bayesian classifiers use the posterior mean parameter vector.