Machine Learning and Neural Networks (MATH3431) Epiphany term, 2023

Handout 4: Bayesian Learning via Stochastic gradient and Stochastic gradient Langevin dynamics

Lecturer & author: Georgios P. Karagiannis georgios.karagiannis@durham.ac.uk

Aim. To introduce the Bayesian Learning and Stochastic gradient Langevin dynamics (description, heuristics, and implementation).

Reading list & references:

- Welling, M., & Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In Proceedings of the 28th international conference on machine learning (ICML-11) (pp. 681-688).

This is subject to minor changes that will be decided based on the Lecture. It will be finalized around 1 day after the Lecture. It is given as guide before the lecture.

1. Bayesian learning and motivations

Remark 1. Bayesian methods are appealing in their ability to capture uncertainty in learned parameters and avoid overfitting. Arguably with large datasets there will be little overfitting. Alternatively, as we have access to larger datasets and more computational resources, we become interested in building more complex models (eg from a logistic regression to a deep neural network), so that there will always be a need to quantify the amount of parameter uncertainty.

Note 2. Consider a Bayesian statistical model with sampling distribution (statistical model) $f(z|w)$ labeled by an unknown parameter $w \in \Theta \subseteq \mathbb{R}^d$ that follows a prior distribution $f(w)$. Assume a dataset $\mathcal{S}_n = \{z_i; i = 1, ..., n\}$ of size n containing independently drawn examples. Let $L_n(w) := f(z_{1:n}|w)$ denote the likelihood of the observables $\{z_i \in \mathcal{Z}\}_{i=1}^n$ give parameter w. The Bayesian model is denoted as

$$\begin{cases} z_i|w & \overset{\text{ind}}{\sim} f(z_i|w), \text{ for } i = 1, ..., n \\ w & \sim f(w) \end{cases} \tag{1.1}$$

Remark 3. Bayesian learning (inference) relies on the posterior distribution density

$$f(w|z_{1:n}) = \frac{L_n(w) f(w)}{\int L_n(w) f(w) \, \mathrm{d}w} \tag{1.2}$$

which quantifies the researcher's belief (or uncertainty) about the unknown parameter w learned given examples $\{z_i \in \mathcal{Z}\}_{i=1}^n$. It is often intractable; hence there is often a need to numerically compute it. (Section 3)

Remark 4. Point estimation of a function h of w is often performed via computation of the posterior expectation w given the examples in $\mathcal{S}_n$

$$\mathrm{E}_f\left(h\left(w\right)|z_{1:n}\right)=\int h\left(w\right)f\left(w|z_{1:n}\right)\mathrm{d}w \tag{1.3}$$

It is often intractable; hence there is often a need to numerically compute it. (Section 3)

Remark 5. Point estimation of w can also be performed via maximum a-posteriori (MAP) estimator w^* of w that is the maximizer w^* of (1.2) i.e.

$$w^*=\arg\max_{\forall w\in\Theta}\left(f\left(w|z_{1:n}\right)\right) \tag{1.4}$$

$$=\arg\min_{\forall w\in\Theta}\left(\underbrace{-\log\left(L_n\left(w\right)\right)}_{\text{(I)}}-\underbrace{\log\left(f\left(w\right)\right)}_{\text{(II)}}\right) \tag{1.5}$$

Note that (I) may be interpreted as an empirical rist function, and (II) can be interpreted as a shrinkage term in terms of shrinkage methods (like LASSO, Ridge). It is often intractable; hence there is often a need to numerically compute it. (Section 2)

Note 6. In what follows, we first present the implementation of GD and SGD addressing MAP learning, and then we introduce the implementation of SGLD addressing posterior density and expectation learning.

2. Maximum A Posteriori (MAP) learning via GD and SGD

Problem 7. Given the Bayesian model (1.1), and rearranging (1.4), MAP estimate w^* of w can be computed as

$$w^*=\arg\min_{\forall w\in\Theta}\left(-\log\left(L_n\left(w\right)\right)-f\left(w\right)\right)=\arg\min_{\forall w\in\Theta}\left(-\sum_{i=1}^{n}\log\left(f\left(w|z_i\right)\right)-\log\left(f\left(w\right)\right)\right) \tag{2.1}$$

Remark 8. GD is particularly suitable to solve (2.1) when w has high dimensionality.

Algorithm 9. *Gradient descent (Algorithm 1 Handout 2) with learning rate $\eta_t\geq 0$ can be used to solve (2.1) by using the update rule as*

For $t=1,2,3,...$ iterate:

(1) *Compute*

$$w^{(t+1)}=w^{(t)}+\eta_t\left(\sum_{i=1}^{n}\nabla_w\log\left(f\left(w^{(t)}|z_i\right)\right)+\nabla_w\log\left(f\left(w^{(t)}\right)\right)\right) \tag{2.2}$$

Remark 10. The implementation of other GD variants (eg (3.2) in Handout 2) is straightforward, based on Algorithm 9.

Remark 11. SGD is particularly suitable to solve (2.1) when w has high dimensionality, and in big-data problems since the repetitive computation of the sum in (2.2) is prohibitively expensive. Yet consider the benefits of SGD against GD as discussed in Remarks 32 and 33 of Handout 3.

Algorithm 12. *Batch Stochastic Gradient Descent (Algorithm 26 in Handout 3) with learning rate $\eta_t \geq 0$ and batch size m can be used to solve (2.1) by using the update rule as*

For $t = 1, 2, 3, ...$ iterate:

(1) *generate a random set $\mathcal{J}^{(t)} \subseteq \{1, ..., n\}^m$ of m indices from 1 to n with or without replacement, and set a $\tilde{\mathcal{S}}_m = \left\{z_i; i \in \mathcal{J}^{(t)}\right\}$.*

(2) *compute*

$$w^{(t+1)} = w^{(t)} + \eta_t \left(\frac{n}{m} \sum_{j \in \mathcal{J}^{(t)}} \nabla_w \log \left(f \left(w^{(t)} | z_j \right) \right) + \nabla_w \log \left(f \left(w^{(t)} \right) \right) \right) \tag{2.3}$$

Remark 13. Recursion (2.3) is justified in terms of SGD theory as

$$\mathrm{E}_{\mathcal{J}^{(t)} \sim \text{simple-random-sampling}} \left(\frac{n}{m} \sum_{j \in \mathcal{J}^{(t)}} \nabla_w \log \left(f \left(w^{(t)} | z_j \right) \right) \right) = \sum_{i=1}^{n} \nabla_w \log \left(f \left(w^{(t)} | z_i \right) \right) \tag{2.4}$$

Remark 14. The implementation of the SGD variants (Algorithms 26, 38, 43, 49 in Handout 3)) is straightforward based on Algorithm 12 and (2.4).

3. Full Bayesian learning via SGLD

Problem 15. Fully Bayesian learning, computationally, is the problem of recovering the posterior distribution $f(w|z_{1:n})$ of w given $z_{1:n}$ that admits density (1.2). For a given Bayesian model (1.1), the Bayesian estimator of $h := h(w)$ can be computed as the posterior expectation of w given the data $\mathcal{S}_n$

$$\mathrm{E}_f \left(h(w) | z_{1:n} \right) = \int h(w) f(w|z_{1:n}) \, \mathrm{d}w \tag{1.3}$$

Remark 16. Monte Carlo integration aims at approximating (1.3), by using Central Limit Theorem or Law of Large Numbers arguments as $\hat{h} \approx \mathrm{E}_f \left(h(w) | z_{1:n} \right)$ where

$$\hat{h} = \frac{1}{T} \sum_{t=1}^{T} h \left(w^{(t)} \right) \tag{3.1}$$

where $\left\{ w^{(t)} \right\}$ are T simulations drawn (approximately) from the posterior distribution 1.2. This theory is subject to conditions we skip.

Remark 17. Stochastic gradient Langevin dynamics (SGLD) algorithm is able to approximately produce samples from the posterior distribution (1.2) of parameters w given the available data $z_{1:n}$. That allows to recover the whole posterior distribution (1.2) (hence account for the uncertainty in

parameters) and approximate posterior expectations (1.3) by averaging out (3.1) according to the Monte Carlo integration (Remark 16).

Remark 18. Stochastic gradient Langevin dynamics (SGLD) algorithm is able to generate a sample approximately distributed according to the posterior distribution (1.2). That allows to recover the whole posterior distribution (hence account for uncertainty in parameters) and approximate posterior expectations based on the Monte Carlo integration (Remark 16).

Note 19. SGLD relies on injecting the 'right' amount of noise to a standard stochastic gradient optimization recursion (2.2), such that, as the stepsize η_t properly reduces, the produced chain $\left\{w^{(t+1)}\right\}$ converges to samples that could have been drawn from the true posterior distribution.

Algorithm 20. *Stochastic Gradient Langevin Dynamics (SGLD) with learning rate $\eta_t > 0$, batch size m, and temperature $\tau > 0$ is*

---

(1) *Generate a random set $\mathcal{J}^{(t)} \subseteq \{1, ..., n\}^m$ of m indices from 1 to n with or without replacement, and set a $\tilde{\mathcal{S}}_m = \left\{z_i; i \in \mathcal{J}^{(t)}\right\}$.*

(2) *Compute*

$$w^{(t+1)} = w^{(t)} + \eta_t \left(\frac{n}{m} \sum_{i \in J^{(t)}} \nabla \log f\left(z_i | w\right) + \nabla \log f\left(w\right) \right) + \sqrt{\eta_t}\sqrt{\tau}\epsilon_t \tag{3.2}$$

where $\epsilon_t \overset{\text{iid}}{\sim} \mathrm{N}\left(0, 1\right)$.

(3) *Terminate if a termination criterion is satisfied; E.g., $t \leq T_{max}$ for a prespecified $T_{max} > 0$.*

Remark 21. The first few iterations from Algorithm 20 because they involve values generated at the beginning of the running algorithm while the chain have not yet converged to (or reached) an area of substantial posterior mass. Hence they are discarded from the output of the SGLD. These values are called burn-in.

Remark 22. The output of SGLD (Algorithm 20) $\left\{w^{(t)}\right\}$ includes the generated values of w produced during the last few iterations of the running algorithm (aka the tail of the generated chain).

Remark 23. SGLD (Algorithm 20) generates as output a random chain $\left\{w^{(t)}\right\}$ that is approximately distributed according to a distribution with density such as

$$f_\tau\left(w | z_{1:n}\right) \propto \exp\left(\frac{1}{\tau} \prod_{i=1}^{n} f\left(z_i | w\right) f\left(w\right) \right) \tag{3.3}$$

$$\propto \exp\left(\frac{1}{\tau} L_n\left(w\right) f\left(w\right) \right) \tag{3.4}$$

under regularity conditions. Conditions 11 on the learning rate are rather inevitable and should be satisfied.

Condition 24. Regarding the learning rate (or gain) $\{\eta_t\}$ should satisfy conditions

(1) $\eta_t \geq 0$,
(2) $\sum_{t=1}^{\infty} \eta_t = \infty$
(3) $\sum_{t=1}^{\infty} \eta_t^2 < \infty$

Remark 25. The temperature parameter $\tau > 0$ is user specified and aims at controlling (eg; inflating) the variance of the produced chain for instance with practical purpose to escape from local modes (otherwise energy barriers) in non-convex problems.

Remark 26. SGLD for $\tau = 1$ approximately simulates from the posterior (1.2).

Remark 27. The popular learning rates $\{\eta_t\}$ in Remark 9 in Handout 2 satisfy Condition 11 and hence can be used in SGD too. Once parametrized, η_t can be tuned based on pilot runs using a reasonably small number of data.

Remark 28. (Mathematically speaking) The stochastic chain in (3.2) can be viewed as a discretization of the continuous-time Langevin diffusion described by the stochastic differential equation

$$\mathrm{d}W(t) = -\nabla_w \left[-\log f\left(W(t)\,|z_{1:n}\right)\right] \mathrm{d}t + \sqrt{2\tau}\mathrm{d}B(t)\,,\ t \geq 0 \tag{3.5}$$

where $\{B(t)\}$ is a standard Brownian motion[1] in $\mathbb{R}^d$ (i.e.). Under suitable assumptions on f, it can be shown that a Gibbs distribution with PDF such as

$$f^*\left(w|z_{1;n}\right) \propto \exp\left(-\frac{1}{\tau}\left[-\log f\left(W(t)\,|z_{1:n}\right)\right]\right) \tag{3.6}$$

is the unique invariant distribution of (3.5), and that the distributions of $W(t)$ converge rapidly to f^* as $t \to \infty$.

Remark 29. (Heuristically speaking) In the initial phase of running, the stochastic gradient noise will dominate the injected noise ϵ_t and the algorithm will imitate an efficient SGD Algorithm 9 -but this is until η_t or $\nabla \log\left(L_n(w)\right)$ become small enough. In the later phase of running, the injected noise ϵ_t will dominate the stochastic gradient noise, so the SGLD will imitate a Langevin dynamics for the target distribution 1.2. The aim is for to the algorithm to transition smoothly between the two phases. Whether the algorithm is in the stochastic optimization phase or Langevin dynamics phase depends on the variance of the injected noise versus that of the stochastic gradient.

Remark 30. One can argue that, the output of SGLD is also an "almost" minimizer of the empirical risk for large enough t. A draw from the Gibbs distribution (3.6) is approximately a minimizer of (2.1). Also one can show that the SGLD recursion tracks the Langevin diffusion (3.5) in a suitable sense. Hence, both imply that the distributions of $W(t)$ will be close to the Gibbs distribution (3.6) for all sufficiently large t.

Remark 31. To guarantee the algorithm to work it is important for the step sizes η_t to decrease to zero, so that the mixing rate of the algorithm will slow down with increasing number of iterations t. Then, we can keep the step size η_t constant once it has decreased below a critical level.

[1] A a continuous-time stochastic process: (1) $B(0) = 0$; (2) $B(t)$ is almost surely continuous ; (3) $B(t)$ has independent increaments ; (4) $B(t) - B(s) \sim \mathrm{N}(0, t-s)$ for $0 \leq s \leq t$.

Remark 32. Expectation (1.3), can be estimated as an arithmetic average

$$\widehat{h_T(w)} = \frac{1}{T}\sum_{t=1}^{T} h\left(w^{(t)}\right) \tag{3.7}$$

as by LLN $\widehat{h_T(w)} \to \mathrm{E}_f\left(h(w)\,|z_{1:n}\right)$

Remark 33. Another more efficient estimator for the expectation (1.3) is the weighted arithmetic average

$$\widehat{h(w)} = \sum_{t=T_0+1}^{T} \frac{\eta_t}{\sum_{t=T_0+1}^{T}\eta_{t'}} h\left(w^{(t)}\right) \tag{3.8}$$

Because the step size decreases, the mixing rate of the chain $\left\{w^{(t)}\right\}$ decreases as well and the simple sample average (3.7) will overemphasize the tail end of the sequence where there is higher correlation among the samples resulting in higher variance in the estimator.

Remark 34. Certain dimensions may have a vastly larger curvature leading to much bigger gradients. In this case a symmetric preconditioning matrix $P_t > 0$ can transform all dimensions to the same scale. Hence the update (3.2) becomes

$$w^{(t+1)} = w^{(t)} + \eta_t P_t\left(\frac{n}{m}\sum_{i\in J^{(t)}} \nabla \log f\left(z_i|w\right) + \nabla \log f\left(w\right)\right) + \sqrt{\eta_t}\sqrt{\tau}P_t^{\frac{1}{2}}\epsilon_t \tag{3.9}$$

where $P_t^{\frac{1}{2}}$ is such that $P_t^{\frac{1}{2}}\left(P_t^{\frac{1}{2}}\right)^{\top} = P_t$.

4. Examples [2]

We continue the Example 33 in Handout 1, and Example 8 in Handout 2. Consider the Bayesian Normal linear regression model

$$\begin{cases} y_i|\beta,\sigma^2 \sim \mathrm{N}\left(x_i^{\top}\beta,\sigma^2\right) & \text{sampling distribution } f\left(y_i|\beta,\sigma^2\right) \\ \beta \sim \mathrm{N}\left(\mu=0,\Sigma=100I_d\right) & \text{prior } f\left(\beta\right) \\ \sigma^2 \sim \mathrm{IG}\left(\phi=1,\psi=1\right) & \text{prior } f\left(\sigma^2\right) \end{cases}$$

and $f\left(\beta,\sigma^2\right) = f\left(\beta\right)f\left(\sigma^2\right)$. As the SGD and SGLD can better handle cases when $w \in \mathbb{R}^d$, instead of $w = \left(\beta,\sigma^2\right) \in \mathbb{R}^2\times\mathbb{R}$, we consider a transformation $w = \left(\beta,\log\left(\sigma^2\right)\right) \in \mathbb{R}^2\times\mathbb{R}$. Hence, the

[2] Code is available from `https://github.com/georgios-stats/Machine_Learning_and_Neural_Networks_III_Epiphany_2023/tree/main/Lecture_handouts/code/04.Stochastic_gradient_Langevine_dynamics`

posterior is such that

$$\log\left(f\left(w_{1:3}|z_j=(x_j,y_j)\right)\right)=-\frac{1}{2}\log\left(2\pi\right)-\frac{1}{2}w_3-\frac{1}{2}\frac{\left(y_j-w_1-x_jw_2\right)^2}{\exp\left(w_3\right)}$$

$$\text{sampling distribution } \; y|w=\left(\beta,\log\left(\sigma^2\right)\right)$$

$$\begin{aligned}\log\left(f\left(w_{1:3}=\left(\beta,\log\left(\sigma^2\right)\right)\right)\right)=&-\frac{1}{2}\log\left(2\pi\right)-\frac{1}{2}100-\frac{1}{2}\frac{\left(w_1-0\right)^2}{100} \quad \text{prior of } w_1=\beta_1\\ &-\frac{1}{2}\log\left(2\pi\right)-\frac{1}{2}100-\frac{1}{2}\frac{\left(w_2-0\right)^2}{100} \quad \text{prior of } w_2=\beta_2\\ &-\left(\phi+1\right)w_3-\frac{\psi}{\exp\left(\psi\right)}+w_3 \quad \text{prior of } w_3=\log\left(\sigma^2\right)\end{aligned}$$

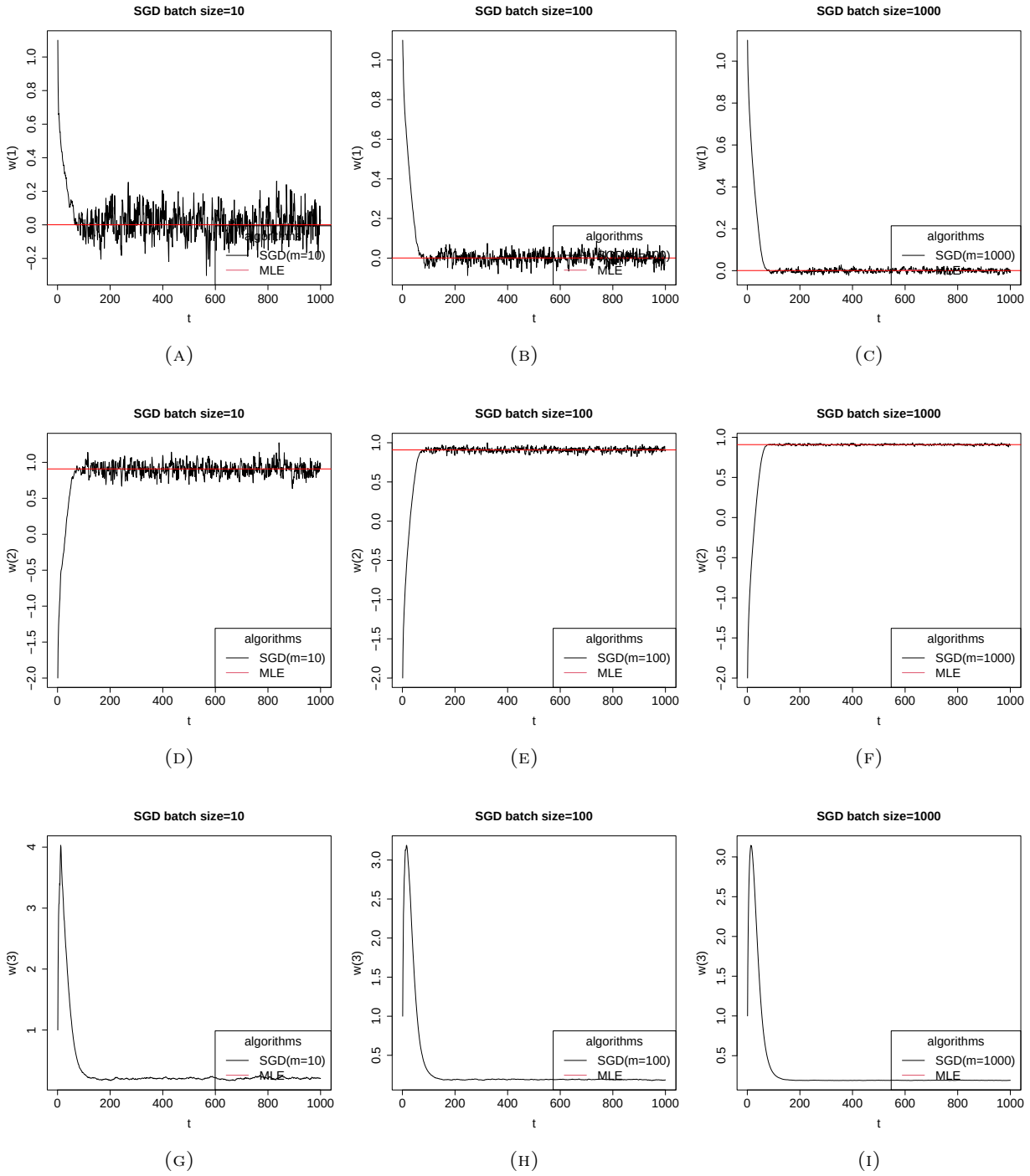

FIGURE 4.1. Bayesian learning via SGD (SGD vs (exact)MLE) Study on batch size m.

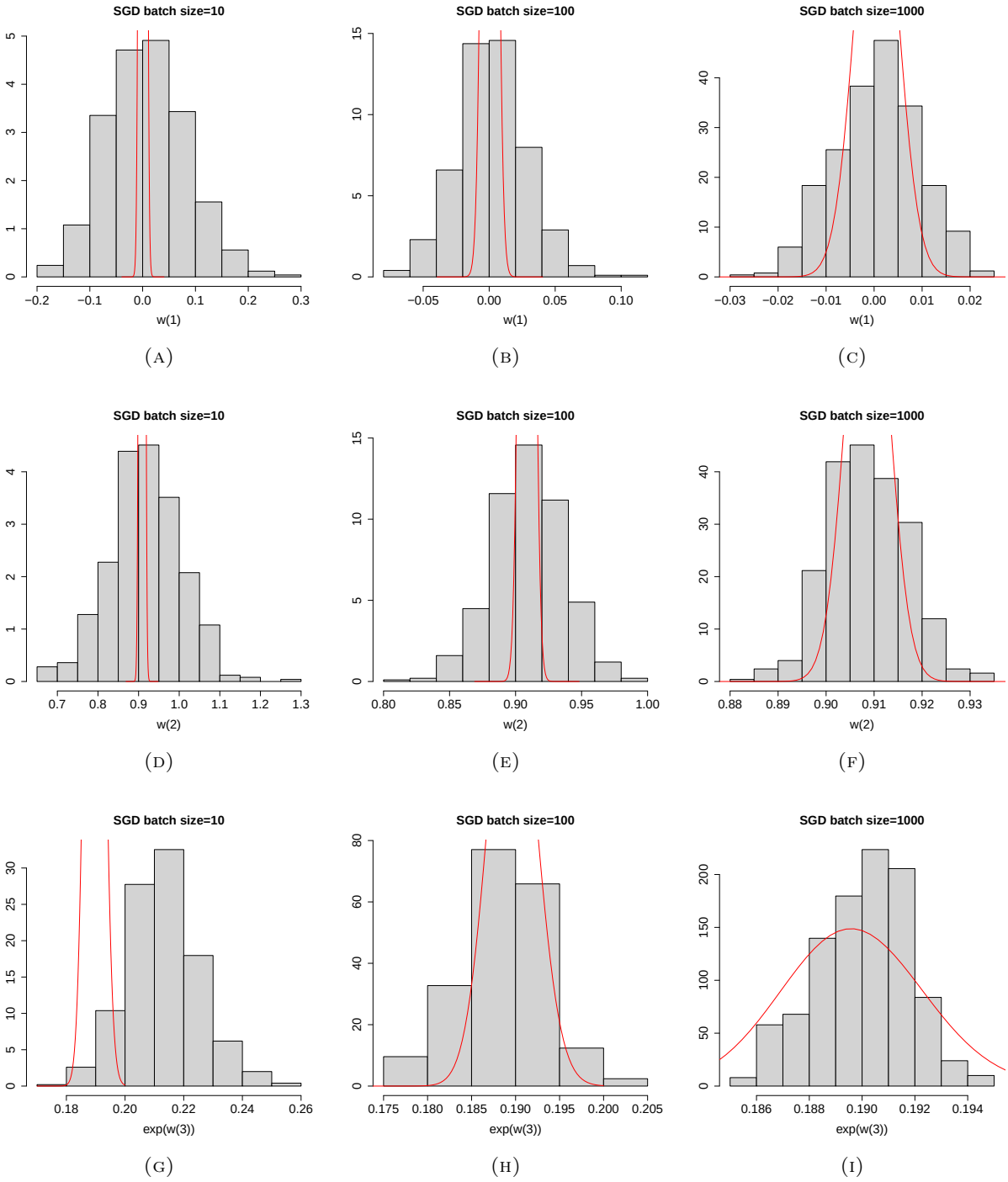

FIGURE 4.2. Bayesian learning: SGLD vs exact posterior (in red). Study on batch size m

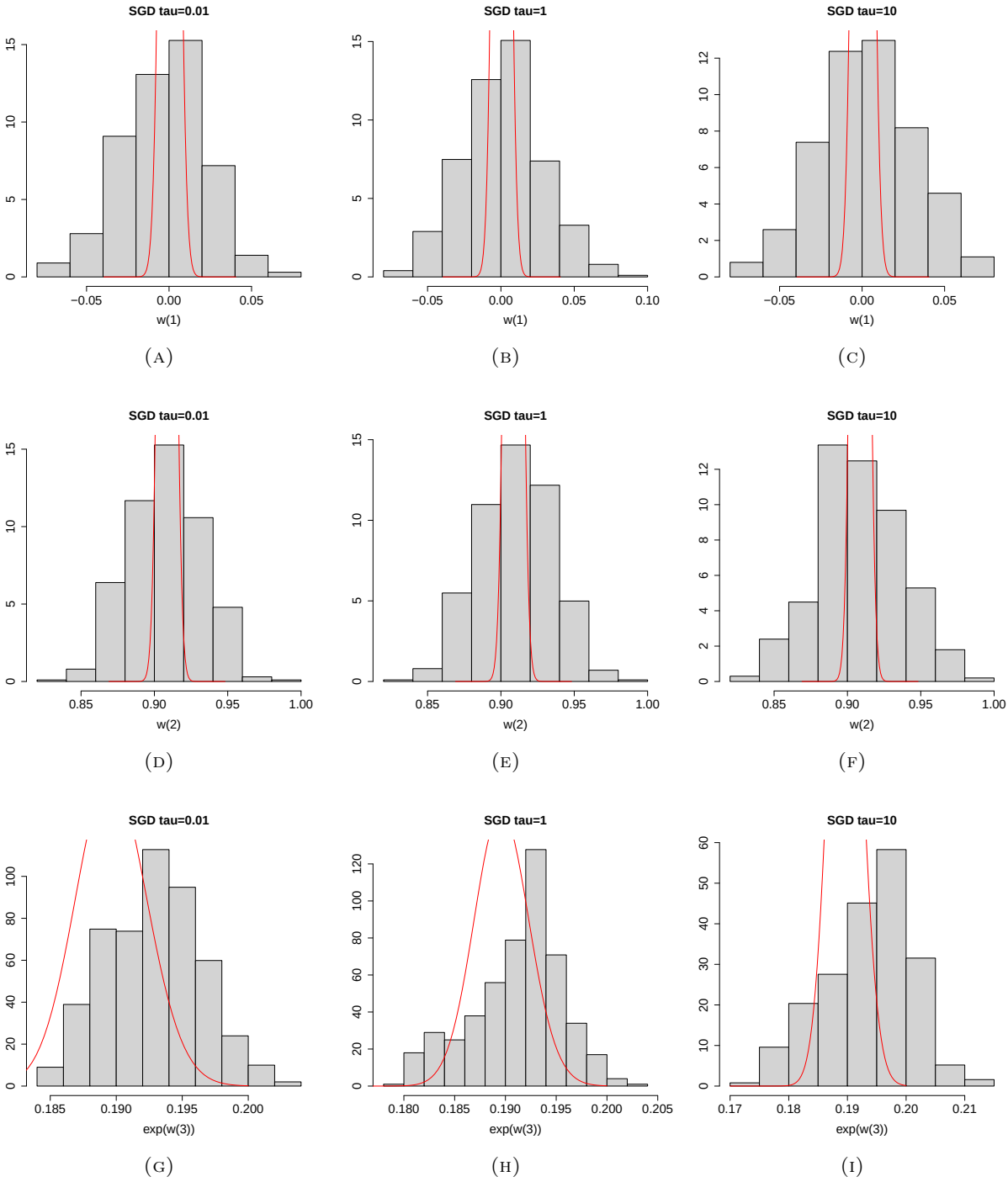

FIGURE 4.3. Bayesian learning: SGLD vs exact posterior (in red). Study on τ