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). Stochastic gradient descent will be implemented in the Bayesian learning too.

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).
- Bottou, L. (2012). Stochastic gradient descent tricks. In Neural networks: Tricks of the trade (pp. 421-436). Springer, Berlin, Heidelberg.
- Sato, I., & Nakagawa, H. (2014). Approximation analysis of stochastic gradient Langevin dynamics by using Fokker-Planck equation and Ito process. In International Conference on Machine Learning (pp. 982-990). PMLR.
- Teh, Y. W., Thiery, A. H., & Vollmer, S. J. (2016). Consistency and fluctuations for stochastic gradient Langevin dynamics. Journal of Machine Learning Research, 17.
- Vollmer, S. J., Zygalakis, K. C., & Teh, Y. W. (2016). Exploration of the (non-) asymptotic bias and variance of stochastic gradient Langevin dynamics. The Journal of Machine Learning Research, 17(1), 5504-5548. –>further reading for theory
- Nemeth, C., & Fearnhead, P. (2021). Stochastic gradient Markov chain Monte Carlo. Journal of the American Statistical Association, 116(533), 433-450.

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

$$\begin{cases} z_{1:n}|w & \sim f\left(z_{1:n}|w\right) \\ w & \sim f\left(w\right) \end{cases} \tag{1.1}$$

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

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

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. To describe the learning algorithms Gradient Descent (GD), Stochastic Gradient Descent (SGD), and Stochastic Gradient Langevin Dynamics (SGLD), we consider that the examples (data) in (1.1) are independent realizations from the sampling distribution i.e. $z_i|w \sim f\left(\cdot|w\right)$ for $i = 1, ..., n$ and that w is continuous (not discrete).

Note 7. 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 8. 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 9. GD is particularly suitable to solve (2.1) when w has high dimensionality.

Algorithm 10. *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*

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

Remark 11. The implementation of other GD variants (eg (3.2) in Handout 2) is straightforward, based on Algorithm 10.

Remark 12. 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 13. *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*

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

Remark 14. Recursion (2.3) is justified in terms of SGD theory as

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

Remark 15. The implementation of the SGD variants (Algorithms 26, 38, 43, 49 in Handout 3)) is straightforward based on Algorithm 13 and (2.4).

3. Fully Bayesian learning via SGLD

Problem 16. 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 17. Monte Carlo integration aims at approximating (1.3), by using Central Limit Theorem or Law of Large Numbers arguments as $\hat{h} \approx \mathrm{E}_f\left(h\left(w\right)|z_{1:n}\right)$ 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 18. 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 17).

Remark 19. 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 17).

Note 20. 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 21. *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 22. The first few iterations from Algorithm 21 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 23. The output of SGLD (Algorithm 21) $\left\{w^{(t)}\right\}$ includes the generated values of w produced during the last few iterations of the running algorithm (aka the end tail of the generated chain).

Remark 24. SGLD (Algorithm 21) 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 25 on the learning rate are rather inevitable and should be satisfied.

Condition 25. 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 26. 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 27. SGLD for $\tau = 1$ approximately simulates from the posterior (1.2).

Remark 28. The popular learning rates $\{\eta_t\}$ in Remark 9 in Handout 2 satisfy Condition 25 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 29. (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

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

where $\{B\left(t\right)\}$ 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

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

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

Remark 30. (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 10 -but this is until η_t or $\nabla \log\left(L_n\left(w\right)\right)$ 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

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

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 31. 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 32. 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.

Remark 33. 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 $\widehat{h_T(w)} \to \mathrm{E}_f\left(h(w)\,|z_{1:n}\right)$ based on LLN arguments.

Remark 34. 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 η_t 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 35. 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; this is similar to the SGD case in Handout 3. Hence the update (3.2) becomes

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

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

Remark 36. 'Exploding gradients' is the practical phenomenon in which large updates to weights during training can cause a numerical overflow or underflow due to the machine error of the computer. Practical solutions involve, at each iteration t, checking the magnitude of the gradient v_t, (e.g., Euclidean norm $\|v_t\|$), and instantly changing it (e.g., truncating it) if it is gonna result an overflow.

To be explain
the computer
tical and Lec

Gradient scaling: involves normalizing the gradient vector such that vector norm (magnitude) equals a defined value. More formally, given any gradient v on an example, gradient

clipping can be used in a standard recursion

$$w^{(t+1)} = w^{(t)} + \eta_t v_t$$

as

$$w^{(t+1)} = w^{(t)} + \eta_t \text{clip}\left(v_t, c\right)$$

where

$$\text{clip}\left(v, c\right) = v \min\left(1, \frac{c}{\|v\|}\right)$$

and c is a clipping threshold impaling that clipping will take place at iteration t if $\|v_t\| > c$.

Gradient clipping: involves forcing the gradient values (element-wise) to a specific minimum or maximum value if the gradient exceeded an expected range.

Beware that unreasonable clipping may introduce significant bias and hence it should not be applied unnecessarily.

4. Examples [2]

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

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

and $f\left(\beta, \sigma^2\right) = f\left(\beta|\sigma^2\right) f\left(\sigma^2\right)$, $\beta \in \mathbb{R}^d$, and $\sigma^2 \in \mathbb{R}_+$. Note given densities

$$\text{N}\left(x|\mu, \Sigma\right) = \left(\frac{1}{2\pi}\right)^d \frac{1}{|\Sigma|} \exp\left(-\frac{1}{2}\left(x-\mu\right)^\top \Sigma^{-1}\left(x-\mu\right)\right)$$

$$\text{IG}\left(x|a, b\right) = \frac{b^a}{\Gamma\left(a\right)} x^{-a-1} \exp\left(-\frac{b}{x}\right) 1\left(x \geq 0\right)$$

Because SGD (Algorithm 13) and SGLD (Algorithm 21) can handle cases where $w \in \mathbb{R}^d$ in a straightforward manner than what they do when $w = \left(\beta, \sigma^2\right) \in \mathbb{R}^d \times \mathbb{R}_+$ which requires an additional projection step; we consider a transformation $w = (\beta, \gamma) \in \mathbb{R}^d \times \mathbb{R}$ with $\gamma = \log\left(\sigma^2\right)$. Hence, the Bayesian model becomes

$$\begin{cases} y_i|\beta, \sigma^2 \sim \text{N}\left(x_i^\top \beta, \exp\left(\gamma\right)\right) & \text{sampling distribution } f\left(y_i|\beta, \gamma\right) \\ \beta|\sigma^2 \sim \text{N}\left(\mu = 0, \exp\left(\gamma\right) V\right) & \text{prior } f\left(\beta|\gamma\right) \\ \gamma \sim f_\gamma\left(\gamma\right) & \text{prior } f\left(\gamma\right) \end{cases}$$

where $f_\gamma\left(\gamma\right)$ is computed according to the method of bijective transformation of random variables; i.e.

$$f_\gamma\left(\gamma\right) = \text{IG}\left(\exp\left(\gamma\right)|\phi, \psi\right) \left|\frac{\text{d}}{\text{d}\gamma} \exp\left(\gamma\right)\right| = \text{IG}\left(\exp\left(\gamma\right)|\phi, \psi\right) \exp\left(\gamma\right)$$

[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`

Then we can compute the required gradients in order to run the SGD, and SGLD with respect to $w = (\beta, \gamma)$ with $\gamma = \log\left(\sigma^2\right)$.

$$\begin{aligned}
\log\left(f\left(z_i = (x_i, y_i)\,|w\right)\right) =& -\frac{1}{2}\log\left(2\pi\right) - \frac{1}{2}\gamma - \frac{1}{2}\left(y_j - x_i^\top \beta\right)^2 \exp\left(-\gamma\right) \\
\log\left(f\left(w = (\beta, \gamma)\right)\right) =& \log\left(f\left(\beta|\gamma\right)\right) + \log\left(f\left(\gamma\right)\right) \\
\log\left(f\left(\beta|\gamma\right)\right) =& -\frac{d}{2}\log\left(2\pi\right) - \frac{d}{2}\gamma - \frac{1}{2}\left|V\right| - \frac{1}{2}\exp\left(-\gamma\right)\left(\beta - \mu\right)^\top V^{-1}\left(\beta - \mu\right) \\
\log\left(f\left(\gamma\right)\right) =& \psi\log\left(\phi\right) - \log\left(\Gamma\left(\phi\right)\right) - \left(\phi + 1\right)\gamma - \psi\exp\left(-\gamma\right) + \gamma
\end{aligned}$$

Hence for the log sampling PDF we have

$$\begin{aligned}
\nabla_w \log\left(f\left(z_i|w\right)\right) =& \left(\frac{\mathrm{d}}{\mathrm{d}\beta}\log\left(f\left(z_i|w\right)\right)\,;\,\frac{\mathrm{d}}{\mathrm{d}\gamma}\log\left(f\left(z_i|w\right)\right)\right) \\
\frac{\mathrm{d}}{\mathrm{d}\beta}\log\left(f\left(z_i|w\right)\right) =& \left(y_i - x_i^\top\beta\right)x_i\exp\left(-\gamma\right) \\
\frac{\mathrm{d}}{\mathrm{d}\gamma}\log\left(f\left(z_i|w\right)\right) =& -\frac{1}{2} + \frac{1}{2}\left(y_j - x_i^\top\beta\right)^2\exp\left(-\gamma\right)
\end{aligned}$$

...so

$$\nabla_w \log\left(f\left(z_i|w\right)\right) = \begin{pmatrix} \left(y_i - x_i^\top\beta\right)x_i\exp\left(-\gamma\right) \\ -\frac{1}{2} + \frac{1}{2}\left(y_j - x_i^\top\beta\right)^2\exp\left(-\gamma\right) \end{pmatrix} \tag{4.1}$$

Hence for the log a priori PDF we have

$$\begin{aligned}
\nabla_w \log\left(f\left(w\right)\right) =& \left(\frac{\mathrm{d}}{\mathrm{d}\beta}\log\left(f\left(w\right)\right)\,;\,\frac{\mathrm{d}}{\mathrm{d}\gamma}\log\left(f\left(w\right)\right)\right) \\
\frac{\mathrm{d}}{\mathrm{d}\beta}\log\left(f\left(w\right)\right) =& -\exp\left(-\gamma\right)V^{-1}\left(\beta - \mu\right) \\
\frac{\mathrm{d}}{\mathrm{d}\gamma}\log\left(f\left(w\right)\right) =& -\frac{d}{2} + \frac{1}{2}\exp\left(-\gamma\right)\left(\beta - \mu\right)^\top V^{-1}\left(\beta - \mu\right) - \left(\phi + 1\right) + \psi\exp\left(-\gamma\right) + 1
\end{aligned}$$

...so

$$\nabla_w \log\left(f\left(w\right)\right) = \begin{pmatrix} -\exp\left(-\gamma\right)V^{-1}\left(\beta - \mu\right) \\ \frac{d}{2} + \frac{1}{2}\exp\left(-\gamma\right)\left(\beta - \mu\right)^\top V^{-1}\left(\beta - \mu\right) - \left(\phi + 1\right) + \psi\exp\left(-\gamma\right) + 1 \end{pmatrix} \tag{4.2}$$

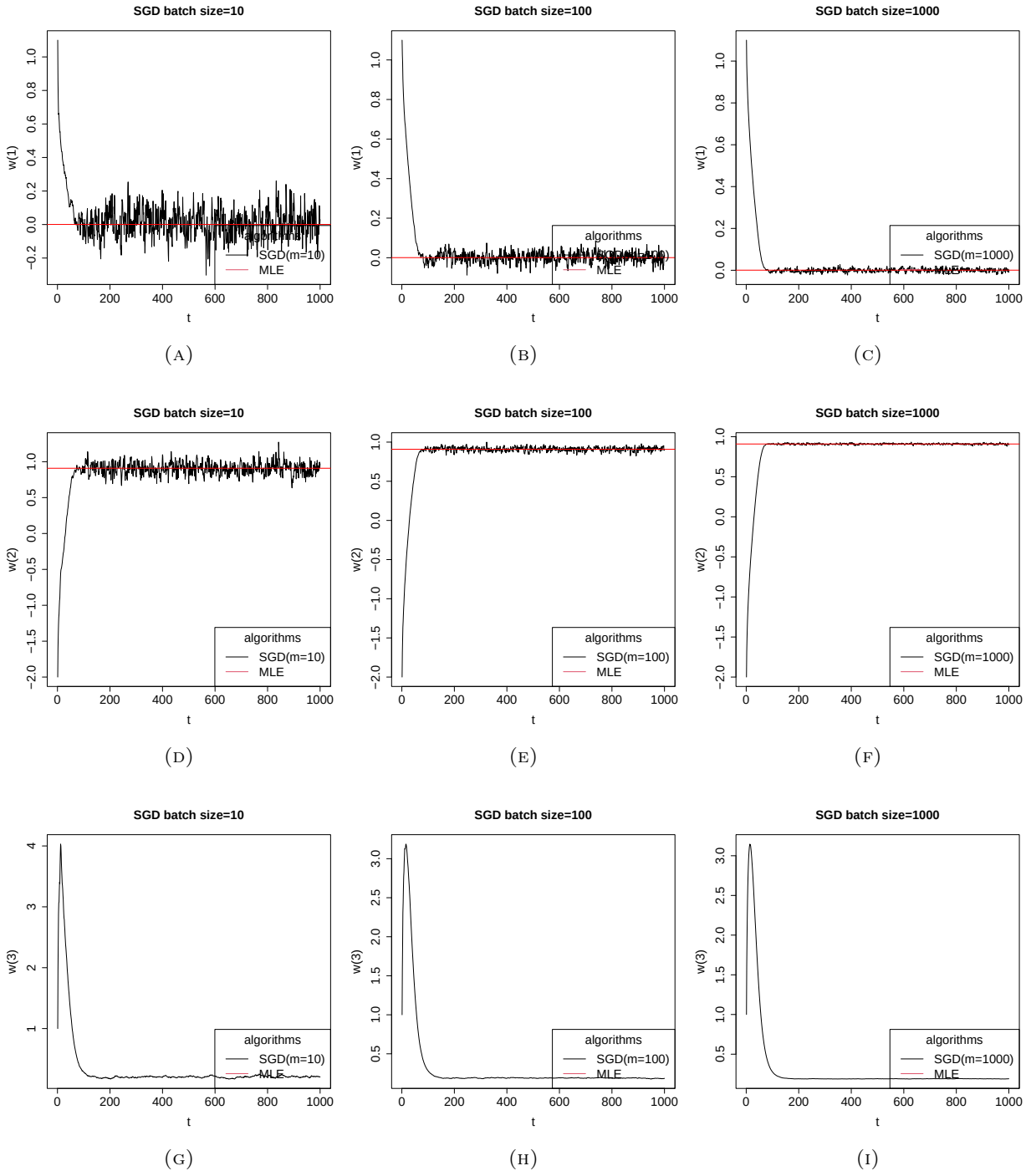
To implement SGD (Algorithm 13) and SGLD (Algorithm 21) , we just need to plug in the computed gradients (4.1) and (4.2) for $w = (\beta, \gamma) \in \mathbb{R}^d \times \mathbb{R}$ with $\gamma = \log\left(\sigma^2\right)$. After running SGD and SGLD with the computed gradients with respect to $w = (\beta, \gamma) \in \mathbb{R}^d \times \mathbb{R}$ with $\gamma = \log\left(\sigma^2\right)$, and obtaining chains $\left\{w^{(t)} = \left(\beta^{(t)}, \gamma^{(t)}\right)\right\}_{t=1}^T$, we can just perform transformation $\left\{\left(\sigma^{(t)}\right)^2 = \exp\left(\gamma^{(t)}\right)\right\}_{t=1}^T$ if we are interested in learning $\left(\beta, \sigma^2\right)$.

We consider $\mu = 0$, $\phi = 1$, $\psi = 1$, and $V = 100 I_d$, for our simulations below.

- In Figures 4.1, we ran the SGD for different batch sizes m and compared it against the exact MLE. We observe that SGD trace converges to the exact MLE. The oscillations are due to the stochastic gradient (ie, noise in the gradient).
- In Figures 4.2, we ran the SGLD for different batch sizes m but the same temperature $\tau = 1$ and compared it against the exact posterior densities. We observe that the histograms of

$\left\{w^{(t)}\right\}$ produced from SGLD are closer to the curves representing the exact posteriors when the batch size m is bigger. As we said this is not a panacea; if the landscape of the exact psterior density was multimodal (aka not convex but with had several maxima), then the SGLD using smaller batch sizes could have performed better, in the sense that the inflated noise from the stochastic gradient could accidentally make the generated chain to pass the low mass barrier and visit a different mode, unlike the one with larger batch-size and hence smaller variation.

- In Figures 4.3, we ran the SGLD for different temperatures τ but the same batch sizes $m = 100$ and compared it against the exact posterior densities. We observe that increasing the temperature τ may increase the variation of the produced chain. We can use a large τ at the beginning of the run of the algorithm to perform an exploration of the space (this is particularly useful for non-convex/multimodal densities as it allows visiting different modes), and later on we can use a smaller temperature such as $\tau = 1$.

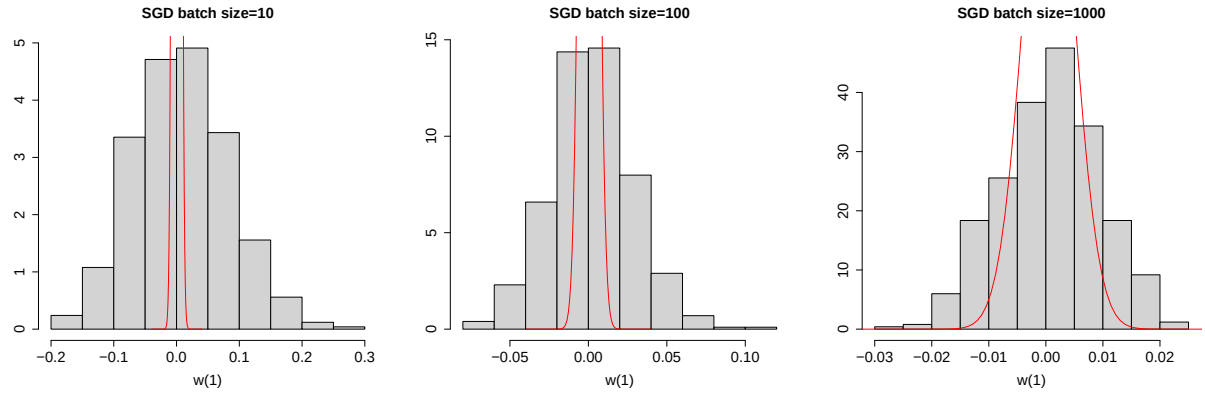


(A) (B) (C)

(D) (E) (F)

(G) (H) (I)

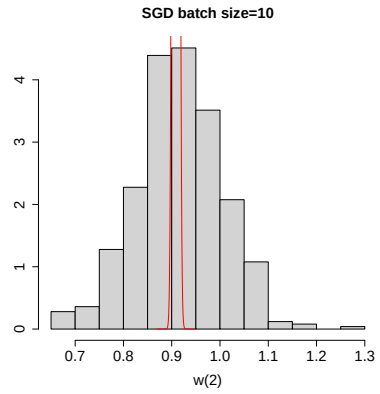
FIGURE 4.1. Bayesian learning via SGD (SGD vs (exact)MLE) Study on batch size m.



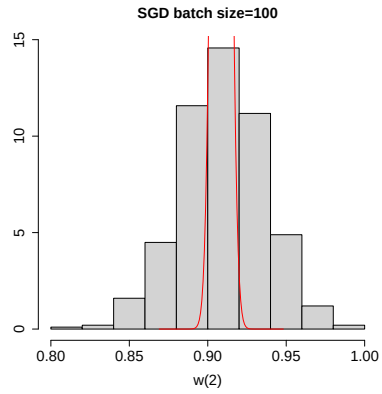
(A)

(B)

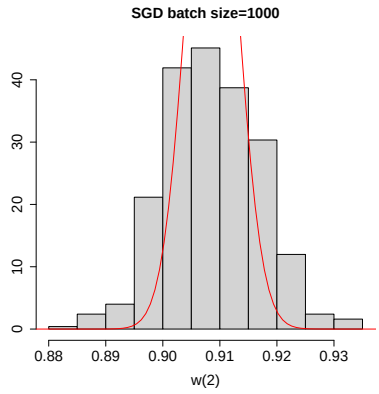
(C)



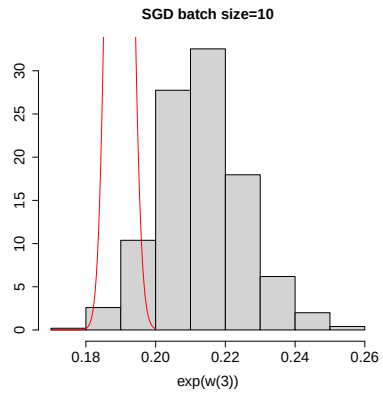
(D)



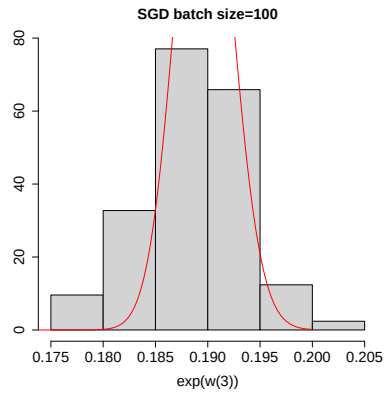
(E)



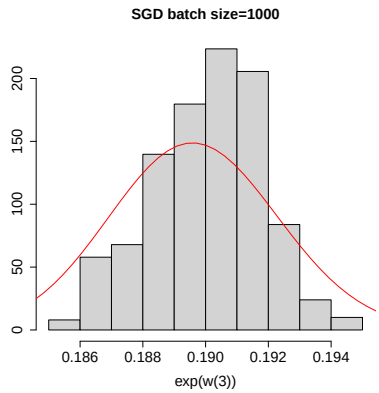
(F)



(G)



(H)



(I)

FIGURE 4.2. Bayesian learning: SGLD vs exact posterior (in red). Temperature $\tau = 1$. Study on batch size m

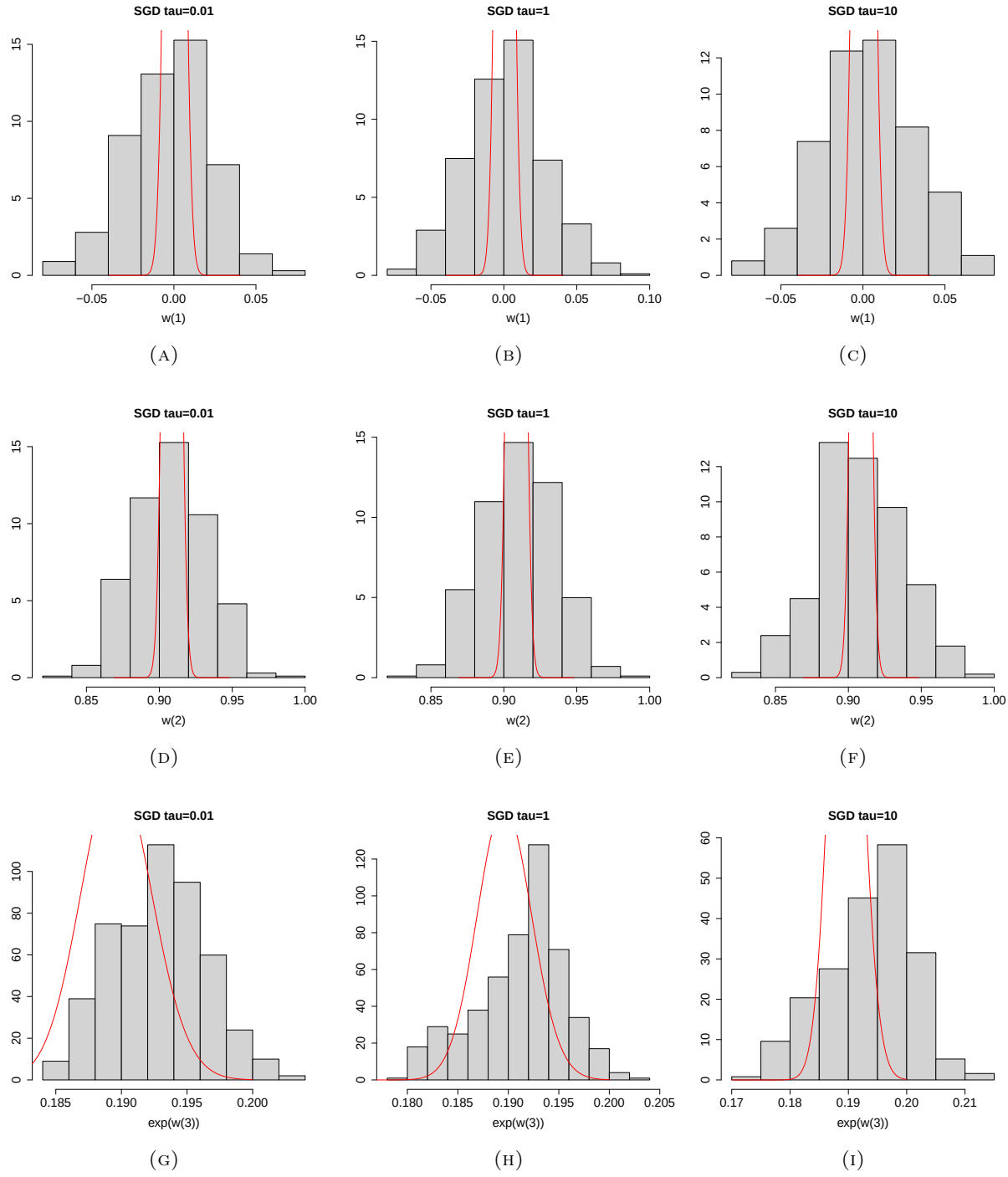


FIGURE 4.3. Bayesian learning: SGLD vs exact posterior (in red). Batch size = 100. Study on τ