Natural Langevin Dynamics for Neural Networks

Gaétan Marceau-Caron[1] and Yann Ollivier[2]

[1] MILA, Université de Montréal, Canada
[2] CNRS, Université Paris-Saclay, France

Abstract. One way to avoid overfitting in machine learning is to use model parameters distributed according to a Bayesian posterior given the data, rather than the maximum likelihood estimator. *Stochastic gradient Langevin dynamics* (SGLD) is one algorithm to approximate such Bayesian posteriors for large models and datasets. SGLD is a standard stochastic gradient descent to which is added a controlled amount of noise, specifically scaled so that the parameter converges in law to the posterior distribution [WT11,TTV16]. The posterior predictive distribution can be approximated by an ensemble of samples from the trajectory.
Choice of the variance of the noise is known to impact the practical behavior of SGLD: for instance, noise should be smaller for sensitive parameter directions. Theoretically, it has been suggested to use the inverse Fisher information matrix of the model as the variance of the noise, since it is also the variance of the Bayesian posterior [PT13,AKW12,GC11]. But the Fisher matrix is costly to compute for large-dimensional models.
Here we use the easily computed Fisher matrix approximations for deep neural networks from [MO16,Oll15]. The resulting *natural Langevin dynamics* combines the advantages of Amari's natural gradient descent and Fisher-preconditioned Langevin dynamics for large neural networks.
Small-scale experiments on MNIST show that Fisher matrix preconditioning brings SGLD close to dropout as a regularizing technique.

Consider a supervised learning problem with a dataset $\mathcal{D} = \{(x_1, y_1), \dots, (x_N, y_N)\}$ of N input-output pairs, to be modelled by a parametric probabilistic distribution $y_i \sim p_\theta(y|x_i)$ ($x = \varnothing$ amounts to unsupervised learning of y). Defining the log-loss $\ell_\theta(y_i|x_i) := -\ln p_\theta(y_i|x_i)$, the maximum likelihood estimator is the value θ that minimizes $\mathbb{E}_{(x,y)\in\mathcal{D}}\ell_\theta(y|x)$, where $\mathbb{E}_{(x,y)\in\mathcal{D}}$ denotes averaging over the dataset.

Stochastic gradient descent is often used to tackle this minimization problem for large-scale datasets [BL03,Bot10]. This consists in iterating

$$\theta \leftarrow \theta - \eta\, \hat{\mathbb{E}}_{(x,y)\in\mathcal{D}}\, \partial_\theta \ell_\theta(y|x), \tag{1}$$

where η is a step size, ∂_θ denotes the gradient of a function with respect to θ, and $\hat{\mathbb{E}}_{(x,y)\in\mathcal{D}}$ denotes an empirical average of gradients from a random subset of the dataset $\mathcal{D}$ (a minibatch, which may be of size 1).

Estimating the model parameter θ via maximum likelihood, i.e., minimizing the training loss on $\mathcal{D}$, is prone to overfitting. Bayesian methods arguably offer a protection against overfitting ([Bis06, 3.4], [Mac03, 44.4]; see also [Nea96,Mac92]

for Bayesian neural networks). Arguably, the variance of the posterior distribution of θ represents the intrinsic uncertainty on θ given the data, and optimizing θ beyond that point results in overfitting [WT11]; sampling the parameter θ from its Bayesian posterior prevents using a too precisely tuned value.

Stochastic gradient Langevin dynamics (SGLD) [WT11,TTV16] modifies stochastic gradient descent to provide random values of θ that are distributed according to a Bayesian posterior. This is achieved by adding controlled noise to the gradient descent, together with an $O(1/N)$ pull towards a Bayesian prior:

$$\theta \leftarrow \theta - \eta \,\hat{\mathbb{E}}_{(x,y)\in\mathcal{D}}\, \partial_\theta \left(\ell_\theta(y|x) - \frac{1}{N} \ln \alpha(\theta) \right) + \sqrt{\frac{2\eta}{N}}\, \mathcal{N}(0, \mathrm{Id}) \tag{2}$$

where N is the size of the dataset, $\alpha(\theta)$ is the density of a Bayesian prior on θ, and $\mathcal{N}(0, \mathrm{Id})$ is a random Gaussian vector of size $\dim(\theta)$. [3] The larger N is, the closer SGLD is to simple stochastic gradient descent, as the Bayesian posterior concentrates around a single point. The Bayesian interpretation determines the necessary amount of noise depending on step size and dataset size. SGLD has the same algorithmic complexity as simple stochastic gradient descent.

Thanks to the injected noise, θ does not converge to a single value, but its *distribution* at time t converges to the Bayesian posterior of θ given the data, namely, $\pi(\theta) \propto \alpha(\theta) \prod_{(x,y)\in\mathcal{D}} p_\theta(y|x)$. A formal proof is given in [TTV16,CDC15] for suitably decreasing step sizes; the asymptotically optimal step size is $\eta_k \approx k^{-1/3}$ at step k, thus, larger than the usual Robbins–Monro criterion for stochastic gradient descent. The asymptotic behavior is well understood from [TTV16,CDC15], and [MDM17,DM16] provide sharp non-asymptotic rates in the convex case.

One can then extract information from the distribution of θ. For instance, the Bayesian posterior mean can be approximated by averaging θ over the trajectory. The full Bayesian posterior prediction can be approximated by ensembling [GBC16, 7.12] predictions from several values of θ sampled from the trajectory, though this creates additional computational and memory costs at test time.

We refer to [WT11,TTV16] for a general discussion of SGLD (and other Bayesian methods) for large-scale machine learning.

Practical remarks. For regression problems, the square loss $(y - \hat{y}(\theta))^2$ between observations y and predictions $\hat{y}(\theta)$ must be properly cast as the log-loss of a Gaussian model, $\ell = (y - \hat{y}(\theta))^2/2\sigma^2 + \dim(y) \ln \sigma$ for a proper choice of σ (such as the empirical RMSE). Just using $\sigma^2 = 1$ amounts to using a badly specified error model and will provide a poor Bayesian posterior.

The variance coming from computing gradients on a minibatch from $\mathcal{D}$, $\hat{\mathbb{E}}_{(x,y)\in\mathcal{D}} \partial_\theta \ell_\theta(y|x)$, adds up to the SGLD noise. For small step sizes, $\eta \ll \sqrt{\eta}$, so the SGLD noise dominates. [AKW12] suggest a correction for large η.

A popular choice of prior $\alpha(\theta)$ is a Gaussian prior $\mathcal{N}(0, \Sigma^2)$; the variance Σ^2 becomes an additional hyperparameter. In line with Bayesian philosophy we also

[3] Our convention for the step size η differs from [TTV16] by a factor $2/N$, namely, $\delta = \frac{2}{N}\eta$ where δ is the step size in [TTV16, (3)]: this allows for a direct comparison with stochastic gradient descent.

tested the conjugate prior for Gaussian distributions with unknown variance (a mixture of Gaussian priors for all Σ^2), the normal-inverse gamma, with default hyperparameters; empirically, performance comes close enough to the best Σ^2, without having to optimize over Σ^2.

Preconditioning the noise. SGLD as above introduces uniform noise in all parameter directions. This might hurt the optimization process. If performance is more sensitive in certain parameter directions, adapting the noise covariance can largely improve SGLD performance. This requires changing both the noise covariance *and* the gradient step by the same matrix [WT11,GC11,AKW12,LCCC16].

For any positive-definite symmetric matrix C, the *preconditioned SGLD*,

$$\theta \leftarrow \theta - \eta\, C\, \hat{\mathbb{E}}_{(x,y)\in\mathcal{D}}\, \partial_\theta \left(\ell_\theta(y|x) - \frac{1}{N} \ln \alpha(\theta) \right) + \sqrt{\frac{2\eta}{N}}\, C^{1/2} \mathcal{N}(0, \mathrm{Id}) \tag{3}$$

still converges in law to the Bayesian posterior (it is equivalent to a non-preconditioned Langevin dynamics on $C^{-1/2}\theta$). A diagonal C amounts to having distinct values of the step size η for each parameter direction, both for noise and gradient.

This assumes that C is fixed and does not depend on θ. [4] In practice, this means C should be adapted slowly in the algorithms (hence our use of running averages for C hereafter); the resulting bias is analyzed in [LCCC16, Cor. 2].

[LCCC16] apply preconditioned SGLD to neural networks, with a diagonal preconditioner C taken from the RMSProp optimization scheme, a classical tool to adapt step sizes for each direction of θ. [5]

Langevin preconditioners and information geometry. In order to provide a good or even optimal preconditioner C, it has been suggested to set C to the inverse of the Fisher information matrix [GC11,AKW12,PT13].

The Fisher information matrix $J(\theta)$ at θ, for a model p_θ, is defined by

$$J(\theta) := \mathbb{E}_{(x,y)\in\mathcal{D}}\, \mathbb{E}_{\tilde{y}\sim p_\theta(\tilde{y}|x)} \left[(\partial_\theta \ln p_\theta(\tilde{y}|x))\, (\partial_\theta \ln p_\theta(\tilde{y}|x))^\top \right] \tag{4}$$

(note that for supervised learning, we fix the distribution of the inputs x from the data but sample y according to the model $p_\theta(y|x)$). Intuitively, the entries of the Fisher matrix represent the sensitivity of the model in each parameter direction.

Using the inverse Fisher matrix as the SGLD preconditioner C has several theoretical advantages. First, this reduces Langevin noise in sensitive parameter directions (thanks to the Fisher matrix being the average of squared gradients).

Second, since C also affects the gradient term in (3), the gradient part of SGLD becomes Amari's *natural gradient*, known to have theoretically optimal convergence [Ama98]. The resulting algorithm is also insensitive to changes of variables in θ (for small learning rates) and makes sense if θ belongs to a manifold.

[4] If $C(\theta)$ depends on θ, the algorithm involves derivatives of $C(\theta)$ with respect to θ [GC11,XSL+14]. In our case (neural networks), these are not readily available.

[5] We could not reproduce the good results from [LCCC16]. Their code contains a bug which produces noise of variance $2\eta/N^2$ instead of $2\eta/N$ in (2), thus greatly suppressing the Langevin noise, and not matching the Bayesian posterior.

Third, the Bayesian posterior variance of the parameter θ is asymptotically proportional to the inverse Fisher information matrix $J(\theta^*)^{-1}$ at the maximum a posteriori θ^* (Bernstein–von Mises theorem [vdV00]). So with Fisher preconditioning, the noise injected in the optimization process has the same shape as the actual noise in the target distribution on θ. Thus, it is tempting to investigate the behavior of SGLD with noise covariance $C \propto J(\theta^*)^{-1}$.

Approximating the Fisher matrix for large models. The Fisher matrix $J(\theta^*)$ can be estimated by replacing the expectation in its definition (4) by an empirical average along the trajectory [AKW12]. This results in Algorithm 3 below.[6]

However, for large-dimensional models such as deep neural networks, the Fisher matrix is too large to be inverted or even stored (it is a full matrix of size $\dim(\theta) \times \dim(\theta)$). So approximation strategies are necessary.

Approximating the Fisher matrix does not invalidate asymptotic convergence of SGLD, since (3) converges to the true Bayesian posterior for any preconditioning matrix C. But the closer C is to the inverse Fisher matrix, the closer SGLD will be to a natural gradient descent, and SGLD noise to the true posterior variance.

One way of building principled approximations of the Fisher matrix is to reason in terms of the associated invariance group. The full Fisher matrix provides invariance under all changes of variables in parameter space θ: optimizing by natural gradient descent over θ or over a reparameterization of θ will yield the same learning trajectories (in the limit of small learning rates). Meanwhile, the Euclidean gradient descent does not have any invariance properties (e.g., inverting black and white in the image inputs of a neural network affects performance). We refer to [Oll15] for further discussion in the context of neural networks.

The diagonal of the Fisher matrix is the most obvious approximation. Its invariance subgroup consists of all rescalings of individual parameter components.

The *quasi-diagonal* approximation of the Fisher matrix [Oll15] is built to retain more invariance properties of the Fisher matrix, at a small computational cost. It provides invariance under all affine transformations of the *activities* of units in a neural network (e.g., shifting or rescaling the inputs, or switching from sigmoid to tanh activation function). The quasi-diagonal approximation maintains the diagonal of the Fisher matrix plus a few well-chosen off-diagonal terms, requiring to store an additional vector of size $\dim(\theta)$. Overall, the resulting algorithmic complexity is of the same order as ordinary backpropagation, thus suitable for large-dimensional models. [Oll15] also provides more complex approximations with a larger invariance group, suited to sparsely connected neural networks.

The resulting *quasi-diagonal natural gradient* can be coded efficiently [MO16]; experimentally, the few extra off-diagonal terms can make a large difference.

Natural Langevin dynamics for neural networks: implementation. Algorithm 1 presents the Langevin dynamics with a generic preconditioner C. For the ordinary SGLD, C would be the identity matrix. The internal setup of a preconditioner

[6] The Fisher matrix definition (4) averages over synthetic data $\tilde{y}$ generated by $p_\theta(\tilde{y}|x)$. In practice, using the samples y from the dataset is simpler (the OP variant in Alg. 3). This can result in significant differences [MO16,Oll15,PB13], even in simple cases.

decouples from the general implementation of SGLD optimization. A preconditioner C is a matrix object that provides the routines needed by Algorithm 1:

– Multiply a gradient estimate by C: $g \leftarrow Cg$;
– Draw a Gaussian random vector $\xi \sim \mathcal{N}(0, C) = C^{1/2}\mathcal{N}(0, \mathrm{Id})$;
– Update C given recent gradient observations;
– An initialization procedure for C at startup.

We now make these routines explicit for several choices of preconditioner.

The RMSProp preconditioner used in [LCCC16] divides gradients by their recent magnitude: C is diagonal, and for each parameter component i, C_{ii} is the inverse of a root-mean-square average of recent gradients in direction i (Alg. 2).

Algorithm 3 describes preconditioned SGLD with a preconditioner $C = J^{-1}$ using the full Fisher matrix J at the posterior mean θ^*. This is suitable only for small-dimensional models. The Fisher matrix is obtained as a moving average of rank-one contributions over the trajectory (Alg. 3). This moving average has the further advantage of smoothing the fluctuations of the parameter θ over the SGLD trajectory, ensuring convergence [AKW12].

Finally we consider SGLD using the *quasi-diagonal* Fisher matrix, the object of the tests in this article, applicable to large-dimensional models.

For a neural network, the parameters are grouped into blocks corresponding to the bias and incoming weights of each neuron, with the bias being the first parameter in a block. The Fisher matrix J is updated as in Algorithm 3, but storing only its diagonal and the first row in each block. Then a Cholesky decomposition $C = AA^\top$ is maintained for the preconditioner C, such that the axioms of the quasi-diagonal approximation are satisfied (Algorithm 4): in each block, A has non-zero entries only on its diagonal and first row, and is built such that $C^{-1} = (A^\top)^{-1}A^{-1}$ has the same first row and diagonal as the Fisher matrix J. The sparse Cholesky decomposition provides the operations of the preconditioner: multiplying by $C = AA^\top$ and sampling from $\mathcal{N}(0, C) = A\mathcal{N}(0, \mathrm{Id})$.

Experiments. We compare empirically four SGLD preconditioners: Euclidean ($C = \mathrm{Id}$, standard SGLD), RMSProp, Diagonal Outer Product (DOP) and Quasi-Diagonal Outer Product (QDOP) on the MNIST dataset. The Euclidean and RMSProp results widely mismatch those from [LCCC16], see footnote 5.

We compare SGLD to Dropout, a standard regularization procedure for neural networks. For SGLD we compare the performance of using a single network set to the posterior mean, and an ensemble of networks sampled from the trajectory (theoretically closer to the true Bayesian posterior, but computationally costlier).

The code for the experiments can be found at `https://github.com/gmarceaucaron/natural-langevin-dynamics-for-neural-networks` . We use a feedforward ReLU network with two hidden layers of size 400, with the usual $\mathcal{N}(0, 1/\text{fan-in})$ initialization [GBC16]. Inputs are normalized to $[0; 1]$. Step sizes are optimized over $\eta \in \{.001, .01, .1, 1\}$ for Euclidean and $\eta \in \{.0001, .001, .01, .1\}$ for the others, with schedule $\eta \leftarrow \eta/2$ every 10,000 updates [LCCC16]. Minibatch size is 100. The metric decay rate and regularizer are $\gamma_t = 1/\sqrt{t}$ and $\varepsilon = 10^{-4}$. The prior was a Gaussian $\mathcal{N}(0, \sigma^2)$ with $\sigma^2 \in \{0.01, 0.1, 1\}$. The Bayesian posterior ensemble is built by storing every 100-th parameter value of the trajectory after the first 500.

Method	NLL (train)	Accuracy (train)	NLL (test)	Accuracy (test)
SGD	0.0003	100.00	0.0584	98.24
Dropout	0.0006	100.00	0.0519	98.61
Ensemble, Euclidean	0.0357	99.63	0.0726	98.10
Ensemble, RMSProp	0.0415	99.47	0.0742	98.17
Ensemble, DOP	0.0292	99.69	0.0660	98.13
Ensemble, QDOP	0.0229	99.85	0.0591	98.38
PostMean, Euclidean	0.0281	99.12	0.1240	97.16
PostMean, RMSProp	0.0299	99.07	0.1134	97.21
PostMean, DOP	0.0243	99.20	0.1389	97.20
PostMean, QDOP	0.0292	99.60	0.3429	98.14

Table 1. Performance on the MNIST test set with a feedforward 400-400 architecture. Hyperparameters were selected based on accuracy on a validation set. The methods are SGD without regularization, Dropout, SGLD ensemble and SGLD posterior mean (PostMean) with a Gaussian prior ($\sigma^2 = 0.1$).

Table 1 shows that SGLD with a quasi-diagonal Fisher matrix preconditioner and Bayesian posterior ensembling outperforms other SGLD settings.

Bayesian theory favors the use of the full Bayesian posterior at test time, rather than any single parameter value. The results here are consistent with this viewpoint: using a single parameter set to the Bayesian posterior mean offers much poorer performance than either Dropout or a Bayesian posterior ensemble. (Dropout also has a Bayesian inspiration as a mixture of models [SHK+14].) This is also consistent with the generally good performance of ensemble methods.

All other preconditioners perform worse than QDOP or Dropout. In particular, the diagonal Fisher matrix offers no advantage over RMSProp, while the *quasi-diagonal* Fisher matrix does. This is consistent with [MO16] and may vindicate the quasi-diagonal construction via an invariance group viewpoint.

References

AKW12. Sungjin Ahn, Anoop Korattikara, and Max Welling. Bayesian posterior sampling via stochastic gradient Fisher scoring. In *ICML*, 2012.

Ama98. Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Comput.*, 10:251–276, February 1998.

Bis06. C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

BL03. Léon Bottou and Yann LeCun. Large scale online learning. In *NIPS*, volume 30, page 77, 2003.

Bot10. Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

CDC15. Changyou Chen, Nan Ding, and Lawrence Carin. On the convergence of stochastic gradient MCMC algorithms with high-order integrators. In *Advances in Neural Information Processing Systems*, pages 2278–2286, 2015.

DM16. Alain Durmus and Eric Moulines. High-dimensional Bayesian inference via the unadjusted Langevin algorithm. arXiv preprint arXiv:1605.01559, 2016.

GBC16. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Data: Dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ of size N;
probabilistic model $p_\theta(y|x)$ with log-loss $\ell(y|x) := -\ln p_\theta(y|x)$;
Bayesian prior $\alpha(\theta) = \mathcal{N}(\theta_0, \Sigma_0)$, default: $\theta_0 = 0$;
Learning rate $\eta_t \ll 1$. Preconditioner C (for simple SGLD: $C = \mathrm{Id}$).
Result: Parameter θ whose distribution approximates the Bayesian posterior $\Pr(\theta \mid D, \alpha)$. Approximation $\bar{\theta}$ of the Bayesian posterior mean of θ.
Initialization: $\theta \sim \alpha(\theta)$; $\bar{\theta} \leftarrow \theta_0$; initialize preconditioner;
while *not finished* **do**
  retrieve a data sample x and corresponding target y from $\mathcal{D}$;
  forward x through the network, and compute loss $\ell(y|x)$;
  backpropagate and compute gradient of loss: $g \leftarrow \partial_\theta \ell(y|x)$ (for a minibatch: let g be the *average*, not the sum, of individual gradients);
  incorporate gradient of prior: $g \leftarrow g + \frac{1}{N}\Sigma_0^{-1}(\theta - \theta_0)$;
  update preconditioner C using current sample and gradient g;
  apply preconditioner: $g \leftarrow Cg$;
  sample preconditioned noise: $\xi \sim \mathcal{N}(0, C) = C^{1/2}\mathcal{N}(0, \mathrm{Id})$;
  update parameters: $\theta \leftarrow \theta - \eta_t\, g + \sqrt{(2\eta_t/N)}\,\xi$;
  update posterior mean: $\bar{\theta} \leftarrow (1 - \mu_t)\bar{\theta} + \mu_t\theta$.
end

Algorithm 1: SGLD with a generic preconditioner C. For instance C may be Id (Euclidean SGLD), a diagonal preconditioner such as RMSProp, the inverse of a Fisher matrix approximation...

Data: Preconditioner $C = D^{-1/2}$ with D a diagonal matrix of size $\dim(\theta)$; decay rate γ_t; regularizer $\varepsilon \geq 0$.
Initialization: $D \leftarrow \mathrm{diag}(1)$;
Preconditioner update: $D_{ii} \leftarrow (1 - \gamma_t)D_{ii} + \gamma_t\, g_i^2$ with g_i the components of the gradient of the current sample;
Preconditioner application: $g_i \leftarrow (D_{ii} + \varepsilon)^{-1/2}\, g_i$;
Preconditioned noise: $\xi_i \leftarrow (D_{ii} + \varepsilon)^{-1/4}\mathcal{N}(0, 1)$.

Algorithm 2: RMSProp routines for SGLD, similar to [LCCC16].

Data: Preconditioner $C = J^{-1}$ with J the Fisher matrix; decay rate γ_t; regularizer $\varepsilon \geq 0$.
Initialization: $J \leftarrow \mathrm{diag}(1)$;
Preconditioner update: Synthesize output $\tilde{y} \sim p_\theta(\tilde{y}|x)$ given current model θ and current input x (OP variant: just use $\tilde{y} = y$ from the dataset);
Compute gradient of loss for $\tilde{y}$: $\tilde{v} \leftarrow \partial_\theta \ell(\tilde{y}|x)$;
Update Fisher matrix: $J \leftarrow (1 - \gamma_t)J + \gamma_t \tilde{v}\tilde{v}^\top$;
Preconditioner application: $v \leftarrow (J + \varepsilon\,\mathrm{Id})^{-1}v$;
Preconditioned noise: $\xi \leftarrow (J + \varepsilon\,\mathrm{Id})^{-1/2}\mathcal{N}(0, \mathrm{Id})$.

Algorithm 3: Routines for SGLD with full Fisher matrix.

Data: Symmetric positive matrix J of which only the diagonal and first row are known; regularizer $\varepsilon \geq 0$.
Result: Sparse matrix A whose non-zero entries lie only on the diagonal and first row, and such that $(A^\top)^{-1}A^{-1}$ has the same diagonal and first row as $J + \varepsilon \operatorname{Id}$.
$A \leftarrow 0$; $A_{00} \leftarrow \frac{1}{\sqrt{J_{00}+\varepsilon}}$ (Matrix indices start at 0);
$A_{ii} \leftarrow \frac{1}{\sqrt{J_{ii}-(A_{00}J_{0i})^2+\varepsilon}}$ for each index $i \neq 0$;
$A_{0i} \leftarrow -A_{00}^2 A_{ii} J_{0i}$ for each index $i \neq 0$;
return A;

Algorithm 4: Quasi-diagonal Cholesky decomposition.

GC11. Mark Girolami and Ben Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

LCCC16. Chunyuan Li, Changyou Chen, David E. Carlson, and Lawrence Carin. Preconditioned stochastic gradient Langevin dynamics for deep neural networks. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 1788–1794. AAAI Press, 2016.

Mac92. David JC MacKay. A practical Bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

Mac03. David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

MDM17. Szymon Majewski, Alain Durmus, and Błażej Miasojedow. 2017.

MO16. Gaétan Marceau-Caron and Yann Ollivier. Practical Riemannian neural networks. *arXiv*, abs/1602.08007, 2016.

Nea96. Radford M. Neal. *Bayesian learning for neural networks*. Springer, 1996.

Oll15. Yann Ollivier. Riemannian metrics for neural networks I: feedforward networks. *Information and Inference*, 4(2):108–153, 2015.

PB13. Razvan Pascanu and Yoshua Bengio. Natural gradient revisited. *arXiv*, abs/1301.3584, 2013.

PT13. Sam Patterson and Yee Whye Teh. Stochastic gradient Riemannian Langevin dynamics on the probability simplex. In *Advances in Neural Information Processing Systems*, pages 3102–3110, 2013.

SHK+14. Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

TTV16. Yee Whye Teh, Alexandre H Thiery, and Sebastian J Vollmer. Consistency and fluctuations for stochastic gradient Langevin dynamics. *Journal of Machine Learning Research*, 17(7):1–33, 2016.

vdV00. A.W. van der Vaart. *Asymptotic statistics*. Cambridge university press, 2000.

WT11. Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, 2011.

XSL+14. Tatiana Xifara, Chris Sherlock, Samuel Livingstone, Simon Byrne, and Mark Girolami. Langevin diffusions and the Metropolis-adjusted Langevin algorithm. *Statistics & Probability Letters*, 91:14–19, 2014.