Bayesian Learning via Stochastic Gradient Langevin Dynamics

Théo Uscidda
Master MVA
Télécom Paris
theo.uscidda@telecom-paris.fr

Roman Castagné, Louis Bouvier
Master MVA
Ecole des Ponts Paristech
firstname.name@eleves.enpc.fr

Abstract

In this project, we focus on the broadly cited paper Bayesian Learning via Stochastic Gradient Langevin Dynamics (SGLD) from Welling and Teh [9]. We review the main assets of the method, taking advantage to some extent both of the Bayesian inference framework and of the relative speed of the Robbins–Monro algorithms. We implement it and compare its outcomes on simple experiments to those of other Bayesian methods. In this way, we emphasize the limits of the posterior estimation via SGLD. We also suggest some possible variations of the algorithms, motivated by the Hamiltonian Markov Chain Monte Carlo (MCMC) literature.

1 Introduction

In their article Bayesian Learning via Stochastic Gradient Langevin Dynamics [9], Welling and Teh introduce a method to sample from a posterior distribution given a prior and a likelihood, based on two main concepts: Langevin MCMC, a particular case of Hamiltonian MCMC [8] and stochastic optimization. This idea is motivated by a simple assessment: when we are in a case of small scale data sets or complex models, not only parameters inference is crucial, but also a metric of uncertainty on those estimates - notably to avoid over-fitting. This can be achieved using Bayesian models leading to the full posterior instead of only a Maximum A Posteriori (MAP). Nonetheless, computations with classical Metropolis-Hastings updates require to evaluate a quantity over the whole data set at each iteration, which is often costly. Taking advantage of a stochastic approximation paves the way for applications of Bayesian inference in large scale data sets and complex models.

In this report, we introduce the theoretical concepts that underpin the SGLD method in section 2. We highlight the main assets and liabilities of this latter in section 3. We then emphasize our implementation to compare SGLD with other Bayesian methods with respect to three simple applications in section 4. We eventually highlight our results and discuss them in section 5.

2 Theoretical setting

Let a data set $X = (x_i)_{1 \leq i \leq N} \in (\mathbb{R}^d)^N$ of size $N \in \mathbb{N}^*$ and a parameter $\beta \in \mathbb{R}^q$. The hypothesis of a continuous state space for β is important here, but it can be extended to constrained continuous spaces. We assume we have defined a model with a prior $p(\beta)$, inducing the posterior: $p(\beta|X) \propto p(\beta) \prod_{i=1}^{N} p(x_i|\beta)$ we want to sample from.

2.1 Langevin Dynamics and MCMC

Langevin MCMC is a particular version of Hamiltonian MCMC, a method inspired by the dynamics of a system in a potential field in physics, involving a position $q \in \mathbb{R}^d$ and a momentum $p \in \mathbb{R}^d$, presented in details by Neal [8]. The position is taken as the variable of interest - i.e. the one we want to sample, in our case β. The momentum is created artificially. Then, an Hamiltonian is defined: $\mathcal{H}(q, p) = U(q) + K(p)$,

where $U(q) \propto_+ -log(p(\beta|X))$ and K is often taken quadratic: $K(p) = p^T M p$, where M is the definite positive mass matrix. We can discretize the Hamiltonian dynamics equations: $\frac{dq}{dt} = \frac{\partial \mathcal{H}}{\partial p}$ and $\frac{dp}{dt} = -\frac{\partial \mathcal{H}}{\partial q}$, involving a time step parameter ϵ and a number of steps L, leading to a horizon $T = \epsilon \times L$. The integration of the discretized system over T and starting at the current point by a reversible method conserving volume called leapfrog enables to get a proposal for a Metropolis-Hastings step. We alternate this latter followed by the negation of the momentum with a step of sampling merely p from the induced distribution $p(p) \propto exp(-K(p))$ - Gaussian with the quadratic definition of K. Langevin MCMC corresponds to a number of leapfrog steps $L = 1$. We see that this choice entails successive samples potentially close to each other in the U field.

2.2 Stochastic Optimization

Stochastic optimization is an approach to address the maximization of the log-posterior $log(p(.|X))$. Instead of computing gradients $\nabla_\beta log(p(\beta_t|X))$ at a step $t \in \mathbb{N}$ of the gradient ascent over the whole data set X, we can estimate this latter over a batch of size $n \leq N$ and define the update:

$$\Delta\beta_t = \frac{\epsilon_t}{2}\left(\nabla_\beta \log(p(\beta_t)) + \frac{N}{n}\sum_{i=1}^{n} \nabla_\beta \log(p(x_{ti}|\beta_t))\right) \tag{1}$$

Where $(\epsilon_t)_{t\geq 1}$ is a sequence of step sizes that has to satisfy the following conditions to provide convergence guarantees:

$$\sum_{t\geq 1} \epsilon_t = \infty \quad \sum_{t\geq 1} \epsilon_t^2 < \infty \tag{2}$$

2.3 Stochastic Gradient Langevin Dynamics

The main idea of the paper of Welling and Teh is to create a kind of "hybridized" update of the parameter β, using Langevin MCMC formalism with a diagonal mass matrix scaled with the step size ϵ_t at time t, a stochastic approximation of the derivative $\frac{\partial \mathcal{H}}{\partial q}$ with a batch in the wake of equation 1. This leads to the following update:

$$\Delta\beta_t = \frac{\epsilon_t}{2}\left(\nabla_\beta \log(p(\beta_t)) + \frac{N}{n}\sum_{i=1}^{n} \nabla_\beta \log(p(x_{ti}|\beta_t))\right) + \eta_t \tag{3}$$

Where $\eta_t \sim \mathcal{N}(0, \epsilon_t)$. We can extend the analogy with physics interpreting the scaling of the momentum as a friction effect.

3 Technical points

Transition between optimization and sampling phases SGLD induces a mix of stochastic optimization and Bayesian sampling, with a smooth transition emphasized by some coefficients denoted by α in the article, computed via the empirical covariance $V_{s,t}$ of the scores $s_{t_i} = \nabla \log p(x_{t_i}|\beta) + \frac{1}{N}\nabla \log p(\beta)$. Intuitively, the transition between the optimization phase and the sampling phase is expressed as a domination of the noise w.r.t the stochastic gradient in the update (3), whose variations are normally distributed with covariance $\frac{N^2}{n} V_{s,t}$, when t is large, by central limit theorem. The domination of the noise in all dimensions can then be expressed: $\alpha_t = \epsilon_t \frac{N^2}{4n} \lambda_{max}(V_{s,t}) \ll 1$.

Estimation of posterior expectation To estimate the posterior expectation, the article suggests to use the step-sizes at each iteration. The mixing rate of the Markov chain decreases during sampling since $\epsilon_t \to 0$, which allows to weight the contribution of each sample to the approximation of the expectation according to the corresponding step-size. For example, in numerical experiments, we approximated predictive distributions using, for x^* new unseen example: $\mathbb{E}_{p(\beta|X,y)}\left[p(y^*|\beta, x^*)\right] \approx \frac{1}{(T-t_0)\sum_{t=t_0}^{T} \epsilon_t} \sum_{t=t_0}^{T} \epsilon_t\, p(y^*|\beta_t, x^*)$, where t_0 determines the *burn-in*, i.e the transition to the sampling phase.

Main potential limits Given the method, we expect a worse sampling efficiency than some clever variants of Metropolis-Hastings (HMC, adaptive SRW) steps using the whole data set (successive samples are bound to be strongly correlated). Besides, it can not be applied to discrete parameters.

4 Implementation and applications

4.1 Bayesian Logistic Regression

As in the paper under study, we first test the Stochastic Gradient Langevin Dynamic on a Bayesian Logistic Regression with a Laplace prior of scale $b = 1$. Bayesian Logistic Regression is a very often used model in the litterature to test the validity and efficiency of sampling methods in Bayesian learning, see for instance [2] or [3].

4.1.1 The experiments

In order to explore all the SGLD features, we apply it to three datasets: two synthetic datasets containing 1,500 samples in $\mathbb{R}^2$ - a linearly separable and a non-linearly separable - and the *HTRU2 Dataset* [1], which contains 17,898 samples with 8 features and a binary class. In the case of the non-linearly separable dataset in $\mathbb{R}^2$, we use a polynomial kernel of degree three to separate the classes, so the sampling is done in $\mathbb{R}^{10}$.

On these three datasets, we compare SGLD to two other sampling methods: Hamiltonian Monte Carlo, whose principles have been described previously, and an Adaptive Symmetric Random Walk. An Adaptive Symmetric Random Walk (ASRW) [6] is a Symmetric Random Walk with a Gaussian distribution proposal, where the covariance of the proposal distribution is learned during the sampling via a stochastic approximation. This adaptive setting allows to efficiently walk the support of the posterior distribution, without having too much rejection. All these methods have been re-implemented from scratch in Python.

These two methods are intrinsically non-stochastic, in the sense that each iteration requires to manipulate the whole training set, without batch computation. Indeed, they require to compute, at each iteration, the acceptance probability of the candidates which is a posterior distribution ratio. We thus compare the speed of SGLD to the speed of these two methods to generate the same number of samples. Afterwards, we examine the sampling quality in order to see how SGLD pays the price of its speed.

4.1.2 Some properties of the predictive distribution

The great advantage of Bayesian methods comes from the fact that they help to control overfitting, providing estimates of the uncertainty about the predictions via the predictive distributions. A well-calibrated Bayesian model then has a maximum uncertainty when it makes bad predictions. We thus investigated some properties of the predictive distribution with each method (HMC, ASRW, SGLD) to compare the obtained calibration.

For the real dataset, we first classify the testing examples using their predictive distribution by predicting for each x^* new unseen example $\hat{y^*} = 1_{p(y^*|X,y,x^*)\geq 0.5}$. Then, we compute the averaged binary entropy of the predictive distribution on the misclassified testing examples. As the entropy of a random variable X with value in $\{-1, 1\}$ is maximal when $\mathbb{P}(X = 1) = \mathbb{P}(X = -1) = 0.5$, this criterion allows to measure the calibration of the model. The larger this criterion is, the more uncertain the model is when it makes bad predictions.

For the synthetic datasets, which are perfectly classified, we study the sub-level in $\mathbb{R}^2$ of the predictive distribution in order to visualize its evolution when we move away from the training data.

4.2 Bayesian Linear Regression

In order to assess the viability of SGLD to optimize a wider range of Bayesian models, we conduct experiments on a Bayesian Linear Regression setting. For a complete picture on Bayesian Linear Regression, see for instance [7]. In short, assuming that our data points $\mathbf{X}$ are centered, labels $\mathbf{y}$ and parameters $\mathbf{w}$, our likelihood is given by $\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2 \sim \mathcal{N}(\mathbf{X}\mathbf{w}, \sigma^2 I)$. We choose to use a Laplace prior of parameter $b = 1$.

We then proceed to test our Bayesian Linear Regression on two regression datasets. The first one is a toy dataset in $\mathbb{R}$ that we map to features in $\mathbb{R}^{10}$ using polynomials transformation in a similar fashion to the Logistic Regression. The second dataset is called *Superconductivity Dataset*[2]. It contains 21,623 samples with each 81 features. We reserve 25% of these samples for testing and use the rest for training.

In order to highlight some of the flaws of SGLD and compare it to a different optimization strategy, we test SAGA-LD for the Bayesian Linear Regression. Implementing SAGA-LD does not require much changes to the algorithm. It consists in reducing the variance of the stochastic gradient updates by adding an estimation of the gradients on the remaining points in the dataset. The update becomes :

[1] `https://archive.ics.uci.edu/ml/datasets/HTRU2`

[2] `https://archive.ics.uci.edu/ml/datasets/superconductivty+data`

$$\Delta\theta = \frac{\epsilon_t}{2}\left(\nabla_\beta \log(p(\beta_t)) + \frac{N}{n}\sum_{i=1}^{n}\left(\nabla_\beta \log(p(x_{ti}|\beta_t) - \nabla_{\alpha_t^i} \log(p(x_{ti}|\alpha_t^i)) + g_\alpha\right)\right) + \eta_t \tag{4}$$

$$\alpha_{t+1}^i \leftarrow \beta_t \text{ if } i \text{ belongs to the batch, } \alpha_t^i \text{ otherwise;}$$

$$g_\alpha \leftarrow g_\alpha + \sum_{i=1}^{n}\left(\nabla_{\alpha_{t+1}^i} \log(p(x_{ti}|\alpha_{t+1}^i) - \nabla_{\alpha_t^i} \log(p(x_{ti}|\alpha_t^i))\right)$$

with $\eta \sim \mathcal{N}(0, \epsilon_t)$, and g_α the approximation of the gradient on all the points in the dataset.

4.3 Bayesian Neural Networks

Neural Networks have seen great success in Machine Learning recently thanks to their expressiveness which lets them model highly complicated functions. However, they are overly confident in their predictions [5] and overfit easily. Bayesian Neural Networks estimate the posterior distribution on their parameters which allows them to quantify epistemic uncertainty and solve the overconfidence issue. However, computing an exact Bayesian posterior is intractable. Several techniques exist to approximate the posterior distribution on the weights of a model, such as Monte Carlo dropout [4] or Bayes-by-Backprop [1]. Another solution is to use SGLD which allows us to transition to a sampling phase after optimization and construct statistics over our parameters.

We use two different models that we optimize with SGLD on the same toy experiments than for the Bayesian Logistic Regression. The first model is a simple linear layer mapping inputs to a class probability, which is equivalent to a Logistic Regression. The second model is a Multi Layer Perceptron with a single hidden layer of size 4. Both models are very simple, with 3 and 33 parameters respectively.

5 Results

Bayesian Logistic regression For this application - exhaustively detailed in the corresponding notebook - we emphasize the results of three methods to assess the efficiency of SGLD: SGLD, ASRW and a stochastic version of Hamiltonian MCMC (SGHD)- that is to say using batch estimation, systematic acceptance and reduced step size like in equation 2. Nonetheless, we have chosen to keep the mass matrix constant to foster exploration. This method can be seen as an extension of the SGLD. We see that the three methods manage to properly separate the points figure 1. The main difference is that since ASRW needs an initialisation and has a MH step compared with the two other methods, it has less uncertainty estimation visible on the predictive colormap (exploring less). We also highlight the transition between the phases evoked section 3 with figure 2, with a preliminary optimization phase and a sampling one. Last, we compute the entropy of the predictive distribution over the misclassified samples by the three methods in table 1. We see that taking several steps $L = 10$ for SGHD leads to a better robustness.

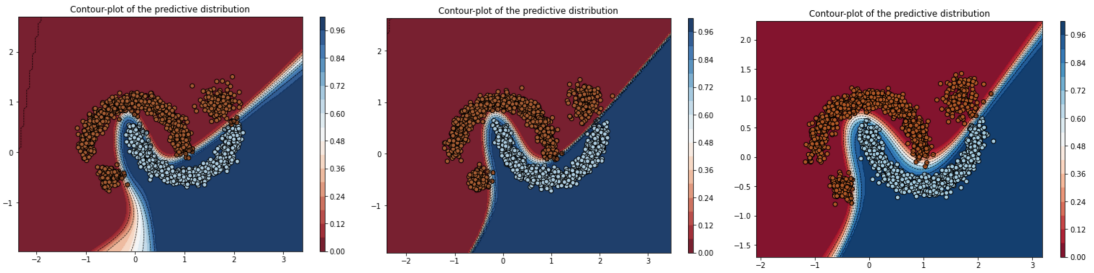

Figure 1: Results of three methods in the attempt to estimate the predictive distribution in the second synthetic dataset: SGLD (left), ASRW (middle), and SGHD (right).

Bayesian Linear Regression We ran SGLD several times with restarts -taking as initialisation the previous outputs- to improve the convergence. We found it to be relatively stable and working for different sets of hyperparameters. We present a comparison of the Mean Squared Errors (MSE) on the test set of the *Superconductivity dataset* between SGLD and SAGA-LD in figure 3. As can be seen, SAGA-LD converges faster, and to lower values than SGLD does. In addition, SAGA-LD has a greatly reduced variance in its

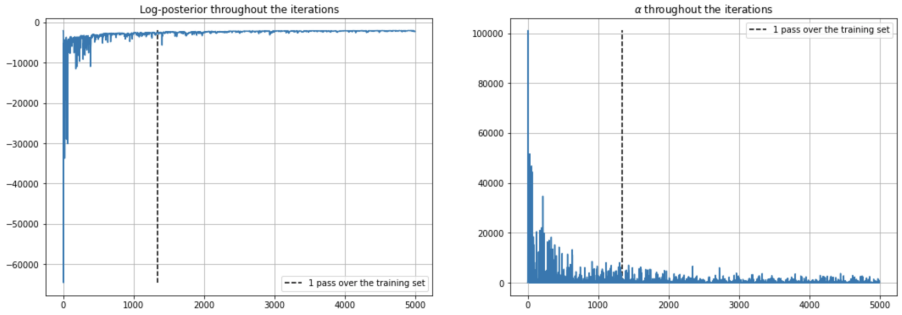

Figure 2: Evolution of the posterior evaluated at the samples and of the α coefficient with sampling iterations (*HRTU2 Dataset*[4]).

MSE compared to SGLD. This difference is not as large for the toy dataset (results not presented here). Indeed, points in that set are much more representative of the dataset than points in the real dataset, which means that the variance of the stochastic gradient should be smaller.

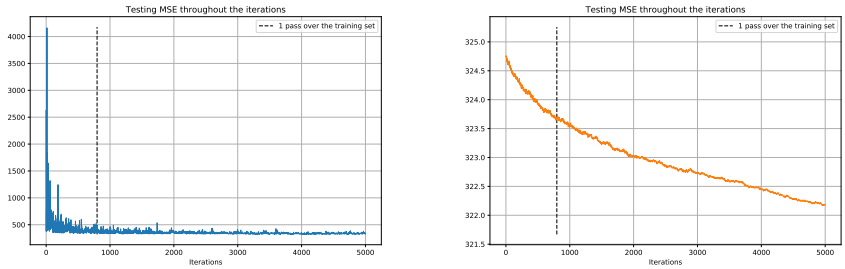

Figure 3: MSE of linear regressions trained with SGLD (left) and SAGA-LD (right). Notice the very different scale of the data, which made it impossible to plot the two errors on the same graph.

Bayesian Neural Networks Bayesian Neural Networks were particularly difficult to optimize correctly. We found that the loss used, a binary cross entropy, did not converge in general, even after extensive tuning of the learning rate and step size scheduler. For the two moons dataset, SGLD failed repeatedly and never succeeded in optimizing the model. We can make some hypothesis about this failure : by multiplying the number of parameters in the model, it is possible that our Neural Network is more sensitive to hyperparameters; stacking layers multiplies the variance over each parameter, which reduces stability of the training.

Table 1: Value of the robustness criterion evaluated for 3 methods on data set HRTU2, over 30 runs

Method	SGLD	ASRW	SGHD
Robustness	0.53174 ± 0.00988	0.52727 ± 0.00224	0.65727 ± 0.00165

6 Conclusion

In this project, we study SGLD, an algorithm at a crossroad between optimization and posterior sampling. By applying it to a range of different tasks, we are able to compare the performance and characteristics of SGLD with other algorithms such as Hamiltonian Monte Carlo sampling, Adaptive Symmetric Random Walks or SAGA-LD.

In future work, we would like to explore Stochastic Hamiltonian Monte Carlo algorithms further, which greatly improve the sampling efficiency compared to SGLD in section 5. Pushing the physics analogy further, an evolution to Hamiltonian MC sampling algorithms would be to simulate several particles in a potential field, which could improve convergence speeds by parallelising computations and would have links to statistical physics theory.

References

[1] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

[2] N. Brosse, A. Durmus, and E. Moulines. The promises and pitfalls of stochastic gradient langevin dynamics. page 11.

[3] A. Dubey, S. J. Reddi, B. Póczos, A. J. Smola, E. P. Xing, and S. A. Williamson. Variance reduction in stochastic gradient langevin dynamics. *Advances in neural information processing systems*, 29:1154, 2016.

[4] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[5] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

[6] H. Haario, E. Saksman, J. Tamminen, et al. An adaptive metropolis algorithm. *Bernoulli*, 7(2):223–242, 2001.

[7] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[8] R. M. Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[9] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.