AdaGeo: Adaptive Geometric Learning for Optimization and Sampling

Gabriele Abbati
Department of Engineering Science
University of Oxford
gabb@robots.ox.ac.uk

Alessandra Tosi
Mind Foundry
Oxford
alessandra.tosi@mindfoundry.ai

Michael A Osborne
Department of Engineering Science
University of Oxford
mosb@robots.ox.ac.uk

Seth Flaxman
Dept. of Mathematics and Data Science Inst.
Imperial College
s.flaxman@imperial.ac.uk

Abstract

Gradient-based optimization and Markov Chain Monte Carlo sampling can be found at the heart of a multitude of machine learning methods. In high-dimensional settings, well-known issues such as slow-mixing, non-convexity and correlations can hinder the algorithms' efficiency. In order to overcome these difficulties, we propose AdaGeo, a preconditioning framework for **ada**ptively learning the **geo**metry of parameter space during optimization or sampling. We use the Gaussian Process latent variable model (GP-LVM) to represent a lower-dimensional embedding of the parameters, identifying the underlying Riemannian manifold on which the optimization or sampling are taking place. Samples or optimization steps are consequently proposed based on the geometry of the manifold. We apply our framework to stochastic gradient descent and stochastic gradient Langevin dynamics and show performance improvements for both optimization and sampling.

1 Introduction

Within the field of statistical machine learning, the performances of a large number of methods rely heavily on the deployment of two main algorithms: gradient-based optimization (as in e.g. deep neural networks [5], kernel methods [11], variational methods [2]) and Markov Chain Monte Carlo (MCMC) sampling-based learning [4]. High-dimensional settings hinder these performances by arising issues such as non-convexity, strong correlations between parameters, or multimodality: convergence is then not reachable within a reasonable amount of time. Recent advancements that tackles these problems include AdaGrad [3], Adadelta [15], Adam [6] and RMSProp on the optimization side and Hamiltonian Monte Carlo [8] and Particle MCMC [1] on the sampling side. As shown by these methods, preconditioning can help overcoming high-dimensionality issues: we propose a new preconditioning method that exploits the advantages of probabilistic dimensionality reduction through Gaussian Process latent variable models (GP-LVM) [7]. The proposed concept operates as follows: after t steps of sampling or optimization, a set of samples of parameter vectors $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_t\}$ is obtained. By introducing a latent variable model, $\boldsymbol{\Theta}$ can be described through a lower-dimensional (latent) set $\boldsymbol{\Omega}$ (formally $\boldsymbol{\theta} = \mathbf{f}(\boldsymbol{\omega}) + \boldsymbol{\eta}$, where $\boldsymbol{\omega} \in \boldsymbol{\Omega}$). GP-LVMs assume a Gaussian Process form for the mapping $\mathbf{f}$ between the two sets: this generative model allows to capture the non-linearities hidden in the parameters and to describe them effectively. By making some assumptions of smoothness over the GP mapping $\mathbf{f}$, we can access the full distribution of the derivative process (also a Gaussian Process). Furthermore, these assuptions allow for the intrinsic structure of the topological space where the parameters lie to be learned as a lower-dimensional Riemannian manifold, equipped with a locally varying metric tensor which encapsulates the geometrical properties

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

of the space. We suggest to perform sampling and optimization exploiting the geometrical insights of the parameters space learned through GP-LVM. To the best of our knowledge, no generic approach based on dimensionality reduction techniques was previously applied to improve optimization and sampling methods. The contributions this paper brings along are the following: we develop a generic framework for combining dimensionality reduction techniques with optimization and sampling methods; we improve the performances of gradient descent and stochastic gradient descent, when training respectively a Gaussian Process and a neural network; we develop an AdaGeo version of stochastic gradient Langevin dynamics, where the information gathered through the latent space are employed to compute the steps of the Markov chain.

2 Methods

2.1 Gaussian Process Latent Variable Model

Latent variable models aim at relating a set of observed variables $\boldsymbol{\Theta} \subset \mathbb{R}^D$ to a set of lower-dimensional unobserved (or latent) variables $\boldsymbol{\Omega} \subset \mathbb{R}^Q$ (with $Q < D$). First, a prior distribution $p(\boldsymbol{\Omega})$ is introduced over the latent space, which we assume to induce a distribution over $\boldsymbol{\Theta}$. This relation is described by the probabilistic mapping

$$\boldsymbol{\theta} = \mathbf{f}(\boldsymbol{\omega}) + \boldsymbol{\eta}, \tag{1}$$

where $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, $\boldsymbol{\omega} \in \boldsymbol{\Omega}$ and $\boldsymbol{\eta}$ is a noise term. Gaussian Processes (GP) describe distributions over functions. They are defined as a collection of random variables, any finite number of which have a joint Gaussian distribution [9]. A Gaussian Process is fully specified by a mean function $m(\cdot)$ and a covariance function $k(\cdot, \cdot)$. A Gaussian Process $f(\cdot)$ will be denoted as $f(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot))$, where $m(\boldsymbol{\omega}) = \mathbb{E}[f(\boldsymbol{\omega})]$ and $k(\boldsymbol{\omega}, \boldsymbol{\omega}') = \mathbb{E}[(f(\boldsymbol{\omega}) - m(\boldsymbol{\omega}))(f(\boldsymbol{\omega}') - m(\boldsymbol{\omega}'))]$. Within the framework of latent variable models, Gaussian Process latent variable models (GP-LVM) assume a GP form for the mapping $\mathbf{f}$: the likelihood of the data $\boldsymbol{\Theta}$ given the latent $\boldsymbol{\Omega}$ is then computed by marginalizing the mapping and optimizing the latent variables. Note that for differentiable kernels the Jacobian $\mathbf{J}$ of $\mathbf{f}$ has a Normal distribution (if its columns are taken to be independent):

$$p(\mathbf{J} \mid \boldsymbol{\Theta}, \boldsymbol{\Omega}, \beta) = \prod_{i=1}^{D} \mathcal{N}\big(\mathbf{J}_{i,:} \mid \boldsymbol{\mu}_{\mathbf{J}_{i,:}}, \boldsymbol{\Sigma}_{\mathbf{J}}\big), \tag{2}$$

where $\boldsymbol{\mu}_{\mathbf{J}_{i,:}}$ and $\boldsymbol{\Sigma}_{\mathbf{J}}$ are respectively the posterior mean and covariance of the Jacobian given $\boldsymbol{\theta}$ and the mapping $\mathbf{f}$. Check for example [12] for the complete derivation.

2.2 AdaGeo Gradient-based Optimization

Optimization represents the most straightforward application of our approach. Consider the problem of finding the minimum of an objective function $g(\boldsymbol{\theta})$: $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} g(\boldsymbol{\theta})$. This could be tackled with an iterative gradient-based method of the form: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \Delta\boldsymbol{\theta}_t$, where $\Delta\boldsymbol{\theta}_t = \Delta\boldsymbol{\theta}_t(\nabla g)$. We propose, after t steps of optimization, to train the GP-LVM on the "samples" $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_t\}$, constructing this way the supporting latent space $\boldsymbol{\Omega}$. The gradients $\nabla_{\boldsymbol{\theta}} g(\boldsymbol{\theta})$ computed in the observed space can be estimated into the latent space through the expectation value of the Jacobian $\mathbf{J}_{\mathbf{f}}$ of the transformation $\mathbf{f}$:

$$\nabla_{\boldsymbol{\omega}} g\big(\mathbf{f}(\boldsymbol{\omega})\big) = \boldsymbol{\mu}_{\mathbf{J}} \nabla_{\boldsymbol{\theta}} g(\boldsymbol{\theta}). \tag{3}$$

At this point the updates of the algorithms can be computed in the latent space, exploiting its meaningful topological structures:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \Delta\boldsymbol{\omega}_t. \tag{4}$$

The mapping in eq. 1 can yield the actual update in the observed parameter space: $\boldsymbol{\theta}_{t+1} = \mathbf{f}(\boldsymbol{\omega}_{t+1})$. It is straightforward to plug in this framework into gradient-based optimization algorithms and formulate an AdaGeo version of the previously cited stochastic gradient descent and relative developments (e.g. AdaGrad [3], Adadelta [15] and Adam [6]). Empirically we find that, for the optimization case, the best strategy is to alternate phases of classic optimization updates to phases of latent updates (appendix A, algorithm 1). This is to overcome the exploration issues GP-LVMs suffer from: these models tend to warp onto a single point those portions of the space data did not cover. Alternating the updates means being able to explore the space through SGD (or one of its variants), codify it with GP-LVM and exploit the newly acquired topological information until exhaustion; the process is then repeated. This scheme becomes necessary as the GP-LVM optimizer starts to saturate and gets stuck somewhere along the way if the optimum is far from the starting point.

2.3 AdaGeo Stochastic Gradient Langevin Dynamics

In Bayesian sampling, we are given a dataset $\mathbf{X}$ and a generative model $p(\mathbf{X}, \boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta} \in \mathbb{R}^D$, on which we impose some prior $p(\boldsymbol{\theta})$. Inference consists in getting insights on the posterior $p(\boldsymbol{\theta} \mid \mathbf{X}) = p(\mathbf{X} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})/p(\mathbf{X})$, but because of the difficulties in computing the denominator an approximation is preferred, often through the form of posterior sampling. In particular, Metropolis-adjusted Langevin algorithms (MALA), are MCMC sampling methods that, coupled with Langevin dynamics [8], enable to sample from the posterior.

Stochastic gradient Langevin dynamics (SGLD) [13] implements this kind of sampling. The iterative algorithms belonging to the stochastic optimization category [10] behave as follows. At each step t, a mini-batch $\mathbf{X}_t = \{\mathbf{x}_{t1}, \dots, \mathbf{x}_{tn}\}$ is stochastically extracted from the dataset and the parameters are updated:

$$\Delta\boldsymbol{\theta}_t = \frac{\epsilon_t}{2}\left(\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}_t) + \frac{N}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}_i \mid \boldsymbol{\theta}_t)\right) + \boldsymbol{\eta}_t \tag{5}$$

where $\boldsymbol{\eta}_t \sim \mathcal{N}(\mathbf{0}, \epsilon_t \mathbf{I})$ and ϵ_t is a time-evolving learning rate. In particular, for large enough t the system will transition into Langevin dynamics, which converges to the true posterior distribution. When Langevin dynamics occur, the algorithm does not need acceptance/rejection steps [13]. As an example of how GP-LVMs could help in sampling from high dimensional distributions, we show how our contribution integrates SGLD. After t steps of sampling, we obtain a set of parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_t\}$. With GP-LVM, it is possible to construct the Q-dimensional latent space $\boldsymbol{\Omega}$ (where $Q < D$), which would condense the relevant features of $\boldsymbol{\Theta}$ in a simpler framework. Now we can perform the update in the latent space; subsequently the GP-LVM mapping $\mathbf{f}(\boldsymbol{\omega})$ in eq. 1 would be used to move back onto the original space and obtain the new observation. In the case of the SGLD sampler, the latent update can be written as:

$$\Delta\boldsymbol{\omega}_t = \frac{\epsilon_t}{2}\left(\nabla_{\boldsymbol{\omega}} \log p\big(\mathbf{f}(\boldsymbol{\omega}_t)\big) + \frac{N}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\omega}} \log p\big(\mathbf{x}_{ti} | \mathbf{f}(\boldsymbol{\omega}_t)\big)\right) + \boldsymbol{\eta}_t, \tag{6}$$

where $\boldsymbol{\eta}_t \sim \mathcal{N}(\mathbf{0}, \epsilon_t \mathbf{I})$. As before, we can compute the gradients of the likelihood with the Jacobian of the mapping $\mathbf{f}$ (eq. 6). We call this method AdaGeo stochastic gradient Langevin dynamics (AdaGeo-SGLD, appendix A, algorithm 2).

3 Results

3.1 AdaGeo Sampling: High-dimensional Banana Distributions

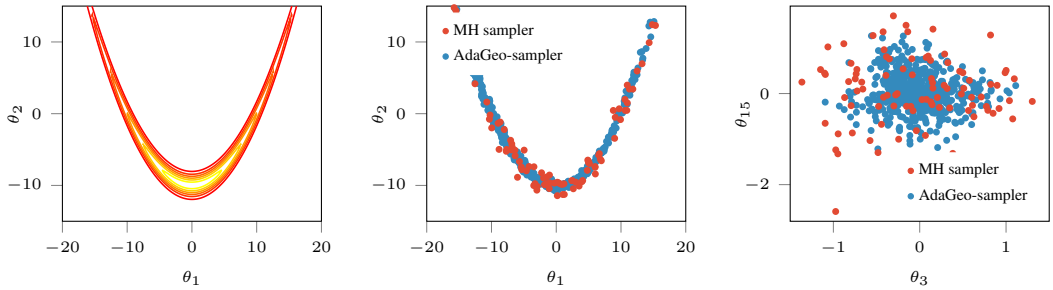
We employed the AdaGeo version of the SGLD sampler to sample from a probability density, the so-called "banana" distribution. In D dimensions this particular PDF $p(\boldsymbol{\theta})$ assumes the form

$$p(\boldsymbol{\theta}) \propto \exp\left(-\frac{\theta_1^2}{200} - \frac{(\theta_2 - b\theta_1^2 + 100b)^2}{2} - \sum_{j=3}^{D} \theta_j^2\right). \tag{7}$$

The main feature of this probability distribution is a sort of banana-shaped structure between the first two components of $\boldsymbol{\theta}$ (figure 1a) while the remaining components act as Gaussian noise. The first set of 100 samples is obtained with a vanilla Metropolis-Hastings Monte Carlo, with 10,000 burn-in steps and a thinning factor of 250. AdaGeo-SGLD is then employed to produce 250 samples, allowing for 1000 burn-in iterations and a thinning factor of 100. Results are shown in figure 1b and 1c: it is possible to see that the relevant areas of the distributions are well covered by the samples, and that the main characteristics are represented properly. Together with a better coverage of the distribution, it is important to note that a relevant speed up is obtained, as the sampler acts on a 5-dimensional latent space. Autocorrelation plots can be found in appendix B.

3.2 AdaGeo Optimization: MNIST with Neural Networks and Gaussian Process Training

First, we implement logistic regression on the MNIST dataset with a simple fully connected neural network, consisting of a single layer of 784 nodes featuring a sigmoid activation function. The 7850 weights are modeled through a 9-dimensional latent space. As a fair comparison, we choose to

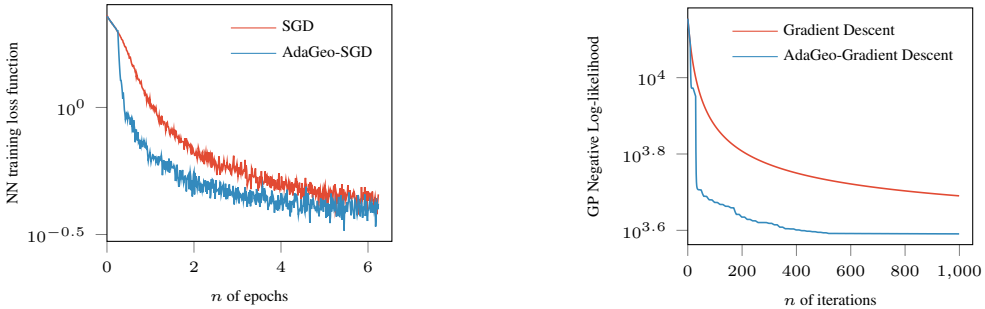


(a) Contour plot of a 50-dimensional banana distribution (θ_1 and θ_2).

(b) Samples on which the AdaGeo-SGLD sampler trains (red) and samples obtained subsequently (blue).

(c) Confounding noise happening between two randomly chosen components of $\boldsymbol{\theta}$.

Figure 1: Sampling from a 50-dimensional banana distribution with AdaGeo-SGLD.



(a) Neural network loss function during training, respectively with vanilla SGD (red) and AdaGeo-SGD (blue).

(b) Negative log-likelihood during training of a Gaussian Process, with AdaGeo and without.

Figure 2: Results obtained with AdaGeo when applied to gradient-based optimization.

train the network using vanilla stochastic gradient descent and the corresponding AdaGeo version, alternating $T_{\boldsymbol{\theta}} = 20$ normal updates to $T_{\boldsymbol{\omega}} = 30$ latent ones until convergence. Results are shown in figure 2a: with latent SGD updates we are able to outperform vanilla SGD and reach the local optimum almost 2 epochs in advance.

Finally, we train a Gaussian Process on the Concrete Compressive Strength Data Set [14]. The kernel is defined by a linear composition of different kernels [9]: radial basis function, Matérn kernel with parameter $\nu = 5/2$, another Matérn kernel with parameter $\nu = 3/2$, a linear kernel and some bias, yielding a total of 9 parameters. The idea is to use a simpler kernel (the GP-LVM one, squared exponential) to fit the hyper-parameters of a more complex one. We run gradient descent with Nesterov momentum and AdaGeo-gradient descent, with $T_{\boldsymbol{\theta}} = 15$, $T_{\boldsymbol{\omega}} = 15$, and learning rates respectively $\epsilon_{\text{latent}} = 10^{-3}$, $\epsilon_{\text{observed}} = 10^{-5}$. The dimension of the latent space is equal to 3. As we can see from figure 2b, AdaGeo successfully improves the performances of gradient descent and significantly speeds up training.

4 Conclusion

We propose a novel method that couples Gaussian Process latent variable models with optimization and sampling algorithms, in order to boost their performances and overcome the issues deriving from high dimensionality. In particular, we show how to include our approach in existing (gradient-based) optimization and sampling (stochastic gradient Langevin dynamics) algorithms. Furthermore, the methods that this work illustrates help sampling even in the simpler cases that don't require gradients (Metropolis-Hastings, Gibbs, etc.). Indeed, through dimensionality reduction the problems of sampling a high-dimensional space are efficiently tackled. Due to its modular nature, the approach presented in this paper is suitable for a large number of potential applications (more sophisticated gradient-based optimization algorithms, variational inference, Hamiltonian Monte Carlo).

References

[1] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72 (3):269–342, 2010.

[2] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, (just-accepted), 2017.

[3] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[4] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[6] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Neil Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of machine learning research*, 6(Nov):1783–1816, 2005.

[8] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11), 2011.

[9] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

[10] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[11] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization and beyond*. the MIT Press, 2002.

[12] Alessandra Tosi, Søren Hauberg, Alfredo Vellido, and Neil D Lawrence. Metrics for probabilistic geometries. *Uncertainty in Artificial Intelligence*, page 800, 2014.

[13] 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.

[14] I-C Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research*, 28(12):1797–1808, 1998.

[15] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

A Algorithms

Here we show the pseudo-code for AdaGeo gradient-based optimization and Adageo stochastic gradient descent.

Algorithm 1 AdaGeo gradient-based optimization

1: **while** convergence is not reached **do**
2: Perform $T_{\boldsymbol{\theta}}$ iterations with classic updates:

$$\Delta\boldsymbol{\theta}_t = \Delta\boldsymbol{\theta}_t(\nabla_{\boldsymbol{\theta}} g(\boldsymbol{\theta}))$$

3: Train the GP-LVM model on the samples got by optimization
4: Continue performing $T_{\boldsymbol{\omega}}$ using the AdaGeo optimizer:

$$\Delta\boldsymbol{\omega}_t = \Delta\boldsymbol{\omega}_t(\nabla_{\boldsymbol{\omega}} g(\mathbf{f}(\boldsymbol{\omega})))$$
$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \Delta\boldsymbol{\omega}_t$$

and moving back to the parameter space with

$$\boldsymbol{\theta}_{t+1} = \mathbf{f}(\boldsymbol{\omega}_{t+1}).$$

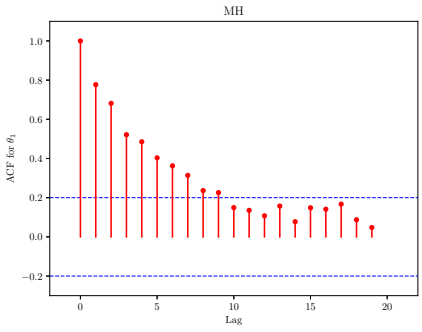
5: **end while**.

Algorithm 2 AdaGeo stochastic gradient Langevin dynamics

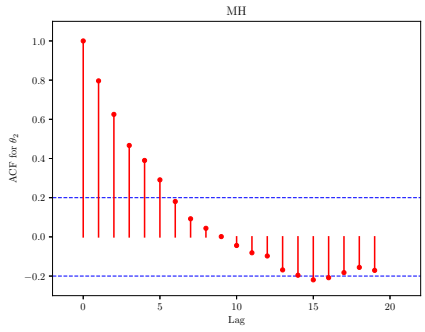
1: Sample t values in the original space using classic SGLD and construct the set $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_t\}$
2: Train the GP-LVM model on $\boldsymbol{\Theta}$, obtaining this way a mapping $\mathbf{f} : \boldsymbol{\Omega} \to \boldsymbol{\Theta}$
3: Choose $\boldsymbol{\omega}_0$ as the latent point corresponding to the last item in the dataset $\boldsymbol{\theta}_t$ and use it as next starting point
4: **for** $t_{\boldsymbol{\omega}} = 1 : N$ **do**
5: Compute the update in the latent space as described in eq. 6, which yields $\boldsymbol{\omega}_{t_{\boldsymbol{\omega}}+1}$
6: Project the new $\boldsymbol{\omega}_{t_{\boldsymbol{\omega}}+1}$ onto the observed space $\boldsymbol{\Theta}$ to get $\boldsymbol{\theta}_{t_{\boldsymbol{\omega}}+1}$
7: **end for**

B Autocorrelation Plots

In this section we report the autocorrelation plots for the experiment described in section 3.1, respectively for θ_1 and θ_2. Standard Metropolis-Hastings can be found on the left while our method, SGLD with AdaGeo, on the right. While Metropolis-Hastings suffers from high autocorrelation despite a long burn-in and large thinning factor, samples from SGLD-AdaGeo does not suffer from any autocorrelation.

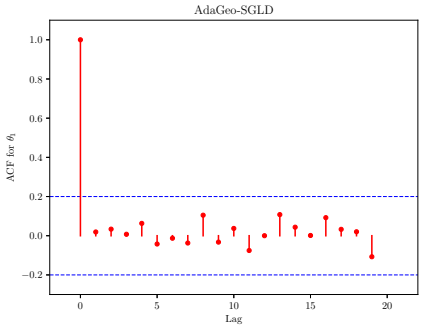


(a) Metropolis-Hastings: Autocorrelation function for θ_1

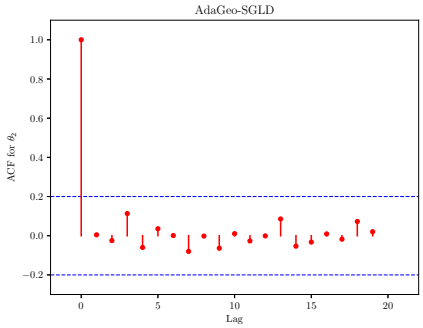


(b) Metropolis-Hastings: Autocorrelation function for θ_2

Figure 3: Autocorrelation functions for θ_1 and θ_2 for the samples obtained with Metropolis - Hastings.



(a) AdaGeo-SGLD: Autocorrelation function for θ_1



(b) AdaGeo-SGLD: Autocorrelation function for θ_2

Figure 4: Autocorrelation functions for θ_1 and θ_2 for the samples obtained with Metropolis - Hastings.