AdaGeo: Adaptive Geometric Learning for Optimization and Sampling

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

Alessandra Tosi
Mind Foundry Ltd.
Oxford

Michael A Osborne
Dept. of Engineering Science
University of Oxford

Seth Flaxman
Dept. of Mathematics &
Data Science Institute
Imperial College London

Abstract

Gradient-based optimization and Markov Chain Monte Carlo sampling can be found at the heart of several 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 the parameter space during optimization or sampling. In particular, 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 is taking place. Samples or optimization steps are consequently proposed based on the geometry of the manifold. We apply our framework to stochastic gradient descent, stochastic gradient Langevin dynamics, and stochastic gradient Riemannian Langevin dynamics, and show performance improvements for both optimization and sampling.

1 INTRODUCTION

The performances of a large number of machine learning methods rely heavily on the deployment of two main algorithms: gradient-based optimization (e.g. deep neural networks [Goodfellow, Bengio, and Courville, 2016], kernel methods [Schölkopf and Smola, 2002], variational methods [Blei, Kucukelbir, and McAuliffe, 2017]) and Markov Chain Monte Carlo sampling-based learning [e.g. Robert, 2004; Gelman et al., 2013].

The advantages and results that these methods produce are well-known and established, but when the geometry of the parameter space becomes hard to explore (e.g. because of non-convexity, strong correlations between parameters, or multimodality), then a number of practical issues arise and convergence is not reachable within a reasonable amount of time. In this sense, preconditioning can help: by preconditioning, we mean the application of a transformation (the so-called preconditioner) to the parameter space, aimed at helping numerical solvers in their tasks. High-dimensional preconditioning can be crucial in this sense: by adaptively correcting the directions in which the algorithm explores the space, it is possible to obtain significant improvements in efficiency when trying to sample the relevant domain areas of e.g. a posterior distribution over the parameters. This yields a better characterization. In this context, preconditioning refers to the practice of scaling each coordinate differently.

Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84. Copyright 2018 by the author(s).

In this paper, we propose a new preconditioning algorithm that exploits the advantages of probabilistic dimensionality reduction through Gaussian process latent variable models (GP-LVM) [Lawrence, 2005]. The proposed concept operates as follows: after t iterations of sampling or optimization, a set of samples of parameter vectors $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_t\}$ is obtained. By defining a latent variable model, $\boldsymbol{\Theta}$ can be described through a lower-dimensional (latent) set $\boldsymbol{\Omega}$. Assuming the existence of a mapping $\mathbf{f}$ between the two sets,

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

where $\boldsymbol{\omega} \in \boldsymbol{\Omega}$ and $\boldsymbol{\eta}$ denotes a noise term. In particular, GP-LVMs assume a Gaussian process form for $\mathbf{f}$: this generative model can capture the non-linearities hidden in the parameters and can 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), and we are not limited to a point estimation of the natural gradient.

Under the assumption of a smooth mapping, the intrinsic structure of the topological space that contains the parameters is learned as a lower-dimensional Riemannian manifold, equipped with a locally varying metric tensor which encapsulates the geometrical properties of the space. In this work, we suggest performing sampling and optimization by exploiting the geometrical insights of the parameter space learned through GP-LVM. Finally, we show analytically how to take advantage of the distribution of the Riemannian metric tensor which such models yield [Tosi et al., 2014]. Code for our method is available online at `https://github.com/gabb7/AdaGeo`.

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 of this paper are listed as follows:

- we develop a generic framework for combining dimensionality reduction techniques with sampling and optimization methods;
- we contribute to gradient-based optimization methods by coupling them with appropriate dimensionality reduction techniques. In particular, we improve the performances of gradient descent and stochastic gradient descent, when training respectively a Gaussian Process and a neural network;
- we contribute to gradient-based Markov Chain Monte Carlo by developing an AdaGeo version of stochastic gradient Langevin dynamics; the information gathered through the latent space is employed to compute the steps of the Markov chain; and
- we extend the approach to stochastic gradient Riemannian Langevin dynamics, thanks to the geometric tensor naturally recovered by the GP-LVM model.

The paper is structured as follows: in section 2 we summarize the previous work that go into the same direction; section 3 contains a short review of GP-LVMs; in section 4 the novel methods and algorithms this work proposes are introduced and explained; in section 5 the results of some experiments are presented; finally in 6 we sum up the contributions of this paper and draw some conclusions.

2 RELATED WORK

Markov Chain Monte Carlo [Robert, 2004] methods are often the preferred choice for sampling-based Bayesian posterior inference. Recent developments include, among others, Hamiltonian Monte Carlo [Neal, 2011] and Particle MCMC [Andrieu, Doucet, and Holenstein, 2010]: both are at the core of probabilistic programming languages (respectively Stan [Carpenter et al., 2016] and Anglican [Tolpin et al., 2016]) and are deployed to perform Bayesian probabilistic inference. Recently, stochastic optimization algorithms [Robbins and Monro, 1951] were combined with Langevin dynamics [Neal, 2011] to develop a class of methods based on stochastic gradient descent capable of sampling from posterior distributions. The resulting algorithm is called stochastic gradient Langevin dynamics [Welling and Teh, 2011].

In an attempt to overcome the sampling issues introduced in the previous section, successful approaches include the results of Riemannian geometry [Do Carmo and Flaherty Francis, 1992], as in Riemannian Hamiltonian Monte Carlo [Girolami and Calderhead, 2011] and stochastic gradient Riemannian Langevin dynamics [Patterson and Teh, 2013].

On the optimization side, we focus our attention on gradient-based optimization algorithms, such as stochastic gradient descent (SGD). SGD processes only a small batch of data, chosen randomly within the total dataset, to approximate the gradients and propose computationally cheap updates. Recent developments mainly address how learning rates can be dynamically modified through time (e.g. AdaGrad [Duchi, Hazan, and Singer, 2011], Adadelta [Zeiler, 2012], Adam [Kingma and Ba, 2014] and RMSProp).

3 A BRIEF REVIEW OF GP-LVM

The aim of this work is to identify meaningful lower-dimensional representations of the parameter space in order to boost the performances of sampling and optimization algorithms. First of all, we introduce the idea of probabilistic dimensionality reduction with latent variable models. Later, we incorporate Gaussian processes into the framework and illustrate the formulation of Gaussian process latent variable models.

3.1 Probabilistic Dimensionality Reduction with Latent Variable Models

We would like to relate a set of observed variables $\boldsymbol{\Theta} \subset \mathbb{R}^D$ to a set of lower-dimensional latent (unobserved) variables $\boldsymbol{\Omega} \subset \mathbb{R}^Q$ (with $Q < D$) and define a joint distribution over them. By finding such representation we perform dimensionality reduction on $\boldsymbol{\Theta}$: such a model is known as a latent variable model.

First, a prior distribution $p(\boldsymbol{\Omega})$ is introduced over the latent space, which induces a distribution over $\boldsymbol{\Theta}$. This relation is described by the probabilistic mapping

$$\theta_{i,j} = f_j(\boldsymbol{\omega}_i) + \eta_j, \tag{2}$$

where $\boldsymbol{\omega}_i$ is the latent point associated with the ith observation $\boldsymbol{\theta}_i$, j is the index of the features of $\boldsymbol{\Theta}$ and $\boldsymbol{\eta}$ is a noise term (typically $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \beta^{-1}\mathbf{I})$).

The classic approach would suppose the latent variables to be marginalized out and the parameters to be optimized by maximizing the model likelihood. An alternative method proposes instead to marginalize the parameters and optimize the latent variables: this procedure (together with other considerations and assumptions) would lead to the formulation of Gaussian process latent variable model (GP-LVM) [Lawrence, 2005].

3.2 Gaussian process latent variable model

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 [Rasmussen and Williams, 2006]. A Gaussian process is fully specified by a mean function $m(\cdot)$ and a covariance function $k(\cdot,\cdot)$. If a real-valued stochastic process $f(\cdot)$ is a Gaussian process, it will be denoted as

$$f(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot)), \tag{3}$$

where $m(\cdot)$ and $k(\cdot,\cdot)$ are respectively defined as

$$m(\boldsymbol{\omega}) = \mathbb{E}\big[f(\boldsymbol{\omega})\big], \tag{4}$$
$$k(\boldsymbol{\omega},\boldsymbol{\omega}') = \mathbb{E}\Big[\big(f(\boldsymbol{\omega}) - m(\boldsymbol{\omega})\big)\big(f(\boldsymbol{\omega}') - m(\boldsymbol{\omega}')\big)\Big]. \tag{5}$$

From a GP, it is possible to instantiate a Gaussian distributed vector which has covariance matrix $\mathbf{K}$, where $K_{ij} = k(\boldsymbol{\omega}_i, \boldsymbol{\omega}_j)$. For simplicity of notation, the mean function is usually taken to be equal to zero, without loss of generality.

In the context of probabilistic dimensionality reduction, Gaussian process latent variable models use GPs to define a prior over the mapping $\mathbf{f}$ (appearing in eq. 2). As suggested above, the likelihood of the data $\boldsymbol{\Theta}$ given the latent $\boldsymbol{\Omega}$ is computed by marginalizing the mapping and optimizing the latent variables. The resulting likelihood is given by

$$\begin{aligned} p(\boldsymbol{\Theta} \mid \boldsymbol{\Omega}, \beta) &= \prod_{j=1}^{D} \mathcal{N}\big(\boldsymbol{\theta}_{:,j} \mid \mathbf{0}, \mathbf{K} + \beta^{-1}\mathbf{I}\big) \\ &= \prod_{j=1}^{D} \mathcal{N}\big(\boldsymbol{\theta}_{:,j} \mid \mathbf{0}, \tilde{\mathbf{K}}\big). \end{aligned} \tag{6}$$

Thus eq. 2 becomes

$$\theta_{i,j} = \tilde{\mathbf{K}}_{(\boldsymbol{\omega}_i,\boldsymbol{\Omega})} \tilde{\mathbf{K}}^{-1} \boldsymbol{\Theta}_{:,j} + \eta_j. \tag{7}$$

For differentiable kernels, it is possible to compute the Jacobian of the mapping $\mathbf{f}$ analytically. Let $\mathbf{J}$ be such Jacobian, defined as

$$J_{ij} = \frac{\partial f_i}{\partial \omega_j}. \tag{8}$$

In the context of GP-LVMs, the Jacobian follows a Gaussian distribution. We can see this if the rows of $\mathbf{J}$ are assumed to be independent:

$$p(\mathbf{J} \mid \boldsymbol{\Omega}, \beta) = \prod_{i=1}^{D} \mathcal{N}\big(\mathbf{J}_{i,:} \mid \boldsymbol{\mu}_{\mathbf{J}_{i,:}}, \boldsymbol{\Sigma}_{\mathbf{J}}\big), \tag{9}$$

and in the case of GP-LVM, for every latent point $\boldsymbol{\omega}_*$,

$$\begin{aligned} p(\mathbf{J} \mid \boldsymbol{\Theta}, \boldsymbol{\Omega}, \beta) &= \prod_{i=1}^{D} \mathcal{N}\left(\mathbf{J}_{i,:} \mid \boldsymbol{\mu}_{\mathbf{J}_{i,:}}, \boldsymbol{\Sigma}_{\mathbf{J}}\right) \\ &= \prod_{i=1}^{D} \mathcal{N}\Big(\mathbf{J}_{i,:} \Bigm| \partial\tilde{\mathbf{K}}_{\boldsymbol{\omega},*}^{\top} \tilde{\mathbf{K}}_{\boldsymbol{\omega},\boldsymbol{\omega}}^{-1} \boldsymbol{\Theta}_{:,j}, \\ &\qquad \partial^2\tilde{\mathbf{K}}_{*,*} - \partial\tilde{\mathbf{K}}_{\boldsymbol{\omega},*}^{\top} \tilde{\mathbf{K}}_{\boldsymbol{\omega},\boldsymbol{\omega}}^{-1} \tilde{\mathbf{K}}_{\boldsymbol{\omega},*}\Big) \end{aligned} \tag{10}$$

where:

$$\Big(\partial\tilde{\mathbf{K}}_{\boldsymbol{\omega},*}\Big)_{nl} = \frac{\partial k(\boldsymbol{\omega}_n, \boldsymbol{\omega}_*)}{\partial \boldsymbol{\omega}_*^{(l)}} \qquad \begin{matrix} n = 1,\ldots,N \\ l = 1,\ldots,Q \end{matrix} \tag{11}$$

$$\Big(\partial^2\tilde{\mathbf{K}}_{*,*}\Big)_{il} = \frac{\partial^2 k(\boldsymbol{\omega}_*, \boldsymbol{\omega}_*)}{\partial \boldsymbol{\omega}_*^{(i)} \partial \boldsymbol{\omega}_*^{(l)}} \qquad \begin{matrix} i = 1,\ldots,Q \\ l = 1,\ldots,Q \end{matrix}. \tag{12}$$

4 OUR APPROACH

In the next sections we show how our GP-LVM-based technique can be applied to optimization and sampling algorithms. First, in section 4.1 we take a look at gradient-based optimization. Second, we consider the framework of Bayesian sampling (section 4.2). Third, the theoretical framework for Riemannian extensions is described in section 4.3.

4.1 AdaGeo Gradient-based Optimization

Optimization represents the most straightforward application of our approach. Consider the problem of finding the minimum of an objective function $g(\boldsymbol{\theta})$:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} g(\boldsymbol{\theta}). \tag{13}$$

This could be tackled with an iterative gradient-based method of the form:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \Delta\boldsymbol{\theta}_t \tag{14}$$

where $\Delta\boldsymbol{\theta}_t = \Delta\boldsymbol{\theta}_t(\nabla_{\boldsymbol{\theta}} g)$. We propose, after t steps of optimization, to: firstly, train the GP-LVM on the samples $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_t\}$; secondly, build the supporting latent space $\boldsymbol{\Omega}$ and; finally, move the computation to this latent space to exploit its meaniningful topological structures,

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \Delta\boldsymbol{\omega}_t. \tag{15}$$

The gradients $\nabla_{\boldsymbol{\theta}} g(\boldsymbol{\theta})$, computed in the observed space, can be mapped into the latent space through the expected Jacobian $\mathbf{J}_{\mathbf{f}}$ of the transformation $\mathbf{f}$:

$$\nabla_{\boldsymbol{\omega}} g\big(\mathbf{f}(\boldsymbol{\omega})\big) = \mathbb{E}\big[\mathbf{J}_{\mathbf{f}}\big] \nabla_{\boldsymbol{\theta}} g(\boldsymbol{\theta}). \tag{16}$$

The mapping in eq. 2 can yield the actual update in the observed parameter space. It is straightforward to plug 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 [Duchi, Hazan, and Singer, 2011], Adadelta [Zeiler, 2012] and Adam [Kingma and Ba, 2014]).

Empirically, we find that the best strategy is to alternate phases of classic optimization updates to phases of latent updates. This is to overcome the exploration issues GP-LVMs suffer from: these models tend to map those portions of the space that data did not cover onto a single point [Duvenaud et al., 2014]. 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. The algorithm can be written as shown in algorithm 1. Here $T_{\boldsymbol{\theta}}$ and $T_{\boldsymbol{\omega}}$ refer to the number of iterations performed respectively in the parameter and latent spaces.

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.
```

4.2 AdaGeo Bayesian Sampling

Consider a dataset $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, modeled with a generative model whose likelihood is $p(\mathbf{X}, \boldsymbol{\theta}) = \prod_{i=1}^N p(\mathbf{x}_i, \boldsymbol{\theta})$ and parameterized by $\boldsymbol{\theta} \in \mathbb{R}^D$, on which we impose some prior $p(\boldsymbol{\theta})$. To perform statistical inference, we would like to be able to sample from the posterior

$$p(\boldsymbol{\theta} \mid \mathbf{X}) = \frac{p(\mathbf{X} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathbf{X})}. \tag{17}$$

However, the denominator (known as model evidence, or marginal likelihood) is often intractable and hence doesn't allow the posterior distribution to be expressed in a closed-form. An approximation is consequently required: Monte Carlo-based sampling is one of the most popular approaches used to estimate the posterior by drawing a set of iid samples $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_N$ from $p(\boldsymbol{\theta} \mid \mathbf{X})$.

We specifically focus on Metropolis-adjusted Langevin algorithms (MALA). Part of the family of Markov Chain Monte Carlo, these methods rely on two main mechanisms:

- Langevin Dynamics [Neal, 2011], where opportunely scaled Gaussian noise is used alongside the gradients of the target distribution;
- Metropolis-Hastings acceptance/rejection steps.

While Langevin dynamics push the system towards regions of high probability, the acceptance/rejection steps help mixing and convergence.

4.2.1 Stochastic Gradient Langevin Dynamics

Stochastic gradient Langevin dynamics (SGLD) [Welling and Teh, 2011] belongs to the class of samplers previously described and extends stochastic optimization [Robbins and Monro, 1951] results. Typically, the iterative algorithms belonging to this category 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 employed to estimate the gradients in the update rule

$$\Delta\boldsymbol{\theta}_t = \frac{\epsilon_t}{2}\left(\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}_t) + \frac{N}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}_{ti} \mid \boldsymbol{\theta}_t)\right), \tag{18}$$

where ϵ_t is a time-evolving learning rate. The mini-batch strategy allows a faster (albeit less precise) approximation of the gradient that doesn't need a pass over the whole dataset: this gives improved scalability.

Unfortunately MAP estimations would not capture the effects of uncertainty and lead to overfitting (and possibly to other troubles with e.g. multimodality). A classic Bayesian way to deal with uncertainty consists of using Markov Chain Monte Carlo [Metropolis et al., 1953] (MCMC). As mentioned above, by combining stochastic optimization updates with Langevin dynamics we get the SGLD algorithm:

$$\begin{aligned}\Delta\boldsymbol{\theta}_t &= \frac{\epsilon_t}{2}\left(\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}_t) + \frac{N}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}_i \mid \boldsymbol{\theta}_t)\right) + \boldsymbol{\eta}_t \\ \boldsymbol{\eta}_t &\sim \mathcal{N}(\mathbf{0}, \epsilon_t \mathbf{I}),\end{aligned} \tag{19}$$

where the step sizes decreases towards 0, with the learning rate satisfying

$$\sum_{t=1}^{\infty} \epsilon_t = \infty, \qquad \sum_{t=1}^{\infty} \epsilon_t^2 < \infty. \tag{20}$$

As $t \to \infty$ the samples will approach the posterior distribution. In particular, for large enough t the system will transition into Langevin dynamics and converge to the true posterior distribution. When Langevin dynamics occur, the algorithm does not need acceptance/rejection steps [Welling and Teh, 2011].

4.2.2 AdaGeo Stochastic Gradient Langevin Dynamics

Next, we show how our framework can be integrated into SGLD. After t steps of sampling, we obtain a set of parameter vectors $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_t\}$. 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 configuration. The update can be now performed in the latent space; subsequently the GP-LVM mapping $\mathbf{f}(\boldsymbol{\omega})$ in eq. 2 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:

$$\Delta\boldsymbol{\omega}_t = \frac{\epsilon_t}{2}\left(\nabla_{\boldsymbol{\omega}} \log p\big(\mathbf{f}(\boldsymbol{\omega}_t)\big) + \frac{N}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\omega}} \log p\big(\mathbf{x}_{ti}|\mathbf{f}(\boldsymbol{\omega}_t)\big)\right) + \boldsymbol{\eta}_t, \tag{21}$$

$\boldsymbol{\eta}_t \sim \mathcal{N}(\mathbf{0}, \epsilon_t \mathbf{I})$.

As before, we can compute the "latent" gradients of the likelihood with the Jacobian of the mapping $\mathbf{f}$ (eq. 16). The new algorithm we built, AdaGeo Stochastic Gradient Langevin Dynamics, as we call it, is described as algorithm 2.

Algorithm 2 AdaGeo Stochastic Gradient Langevin dynamics

```
1: Sample $t$ values in the original space using classic SGLD and construct the set $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_t\}$
2: Train the GP-LVM model on $\boldsymbol{\Theta}$, obtaining in 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 $t_\omega = 1 : N$ do
5:     Compute the update in the latent space as described in eq. 21, which yields $\boldsymbol{\omega}_{t_\omega+1}$
6:     Compute $\boldsymbol{\theta}_{t_\omega+1}$ through: $\boldsymbol{\theta}_{t_\omega+1} = \mathbf{f}(\boldsymbol{\omega}_{t_\omega+1})$
7: end for
```

4.3 Riemannian Extensions

In the following sections the theoretical framework necessary to plug GP-LVM into the Riemannian development of the stochastic gradient Langevin dynamics is shown and explained.

4.3.1 GP-LVM Local Metric

A trained GP-LVM can grant access to relevant information about the intrinsic geometry of the latent space, i.e. the metric tensor.

If the covariance function of a Gaussian process is differentiable (as it is the case, for example, for the widely-used squared exponential), it is straightforward to show that the mapping $\mathbf{f}$ in eq. 2 is also differentiable. Under this assumption, the pull-back metric acting from the latent space $\boldsymbol{\Omega}$ to the observed space $\boldsymbol{\Theta}$ can be computed [Do Carmo and Flaherty Francis, 1992]. Thus, we are able to define the Riemannian metric tensor over the manifold. The Riemannian metric $\mathbf{G}$ is defined as follows.

Definition (Riemannian metric) *A Riemannian metric* $\mathbf{G}$ *on a manifold* $\mathcal{M}$ *is a symmetric and positive definite matrix which defines a smoothly varying inner product:*

$$\langle \mathbf{a}, \mathbf{b} \rangle_x = \mathbf{a}^\top \mathbf{J}^\top \mathbf{J} \mathbf{b} = \mathbf{a}^\top \mathbf{G}(x) \mathbf{b} \tag{22}$$

in the tangent space $T_x\mathcal{M}$*, for each point* $x \in \mathcal{M}$*,* $\mathbf{a}, \mathbf{b} \in T_x\mathcal{M}$*, where* $\langle \cdot, \cdot \rangle_x$ *denotes the inner product. The matrix* $\mathbf{G}$ *is called the metric tensor.*

Intuitively, the metric tensor $\mathbf{G}$ gives information about the geometry (e.g. related to distances and geodetic lines) of the latent space. Let $\mathbf{J}$ be the Jacobian of $\mathbf{f}$. Then the tensor

$$\mathbf{G} = \mathbf{J}^\top \mathbf{J} \tag{23}$$

defines a local inner product over the latent space. Eq. 10, together with eq. 23, yields a distribution over the metric tensor $\mathbf{G}$ [Tosi et al., 2014]

$$\mathbf{G} \sim \mathcal{W}_Q\Big(D, \boldsymbol{\Sigma}_{\mathbf{J}}, \mathbb{E}\left[\mathbf{J}^\top\right] \mathbb{E}[\mathbf{J}]\Big), \tag{24}$$

where $\mathcal{W}$ denotes the non-central Wishart distribution [Anderson, 1946], which is a generalization to multiple dimensions of the chi-squared distribution.

The expected metric tensor can be computed as

$$\mathbb{E}\left[\mathbf{J}^\top \mathbf{J}\right] = \mathbb{E}\left[\mathbf{J}^\top\right] \mathbb{E}[\mathbf{J}] + D\boldsymbol{\Sigma}_{\mathbf{J}}. \tag{25}$$

Note that the second term in the last equation, depending on the covariance of the Jacobian, makes the metric tensor expand as the uncertainty over the mapping increases.

4.3.2 Stochastic Gradient Riemannian Langevin Dynamics

As mentioned above, issues in MCMC schemes may arise if the components of $\boldsymbol{\theta}$ have different scales or large correlations, as in these cases the isotropic proposal distribution of the stochastic gradient Langevin dynamics will lead to slow mixing. We propose to gather information about the geometry of the space though GP-LVM, but an alternative approach employing preconditioning tools consists in applying the Riemannian metric tensor $\mathbf{G}$ to the updates. In this way the issues encountered in the parameter space (e.g. different scales or large correlations) could be overcome and sampling improved.

The stochastic gradient Riemannian Langevin dynamics (SGRLD) sampler [Patterson and Teh, 2013] puts together the advantages of exploiting a known Riemannian geometry with the scalability of the stochastic optimization approaches. The update rule can be written as:

$$\begin{aligned} \Delta\boldsymbol{\theta}_t &= \frac{\epsilon_t}{2}\boldsymbol{\mu}(\boldsymbol{\theta}_t) + \mathbf{G}^{-\frac{1}{2}}(\boldsymbol{\theta}_t)\boldsymbol{\eta}_t \\ \boldsymbol{\eta}_t &\sim \mathcal{N}(\mathbf{0}, \epsilon_t\mathbf{I}), \end{aligned} \tag{26}$$

where the jth component of $\boldsymbol{\mu}(\boldsymbol{\theta})$ is given by

$$\begin{aligned} &\boldsymbol{\mu}(\boldsymbol{\theta})_j = \\ &\left(\mathbf{G}^{-1}(\boldsymbol{\theta})\Big(\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) + \frac{N}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}_{ti} \mid \boldsymbol{\theta})\Big)\right)_j \\ &\quad - 2\sum_{k=1}^{D}\left(\mathbf{G}^{-1}(\boldsymbol{\theta})\frac{\partial \mathbf{G}(\boldsymbol{\theta})}{\partial \theta_k}\mathbf{G}^{-1}(\boldsymbol{\theta})\right)_{jk} \\ &\quad + \sum_{k=1}^{D}\big(\mathbf{G}^{-1}(\boldsymbol{\theta})\big)_{jk}\mathrm{Tr}\left(\mathbf{G}^{-1}(\boldsymbol{\theta})\frac{\partial \mathbf{G}(\boldsymbol{\theta})}{\partial \theta_k}\right). \end{aligned} \tag{27}$$

With respect to the non-Riemannian version of the algorithm, it is possible to make some changes: the first term in $\boldsymbol{\mu}(\boldsymbol{\theta})$ is the gradient of the log-likelihood, already seen before, preconditioned with the matrix $\mathbf{G}(\boldsymbol{\theta})$ and hence taking into account the geometry of the space; the second and third term describe the derivatives of the curvature of the manifold. Notice the noise term in (26) is appropriately preconditioned as well. Typically the Fisher information matrix is used as $\mathbf{G}$, but of course an analytical form is necessary (and rarely obtainable in the majority of problems).

4.3.3 AdaGeo Stochastic Gradient Riemannian Langevin Dynamics

Beyond the classic formulation of SGRLD, we now describe our proposal for incorporating dimensionality reduction into its framework. Analogous to the SGLD case, the update in the observed space in eq. 26 can be converted into the one acting in the latent space:

$$\begin{aligned} \Delta\boldsymbol{\omega}_t &= \frac{\epsilon_t}{2}\boldsymbol{\mu}(\boldsymbol{\omega}_t) + \mathbf{G}_{\boldsymbol{\omega}}^{-\frac{1}{2}}(\boldsymbol{\omega}_t)\boldsymbol{\eta}_t \\ \boldsymbol{\eta}_t &\sim \mathcal{N}(\mathbf{0}, \epsilon_t\mathbf{I}), \end{aligned} \tag{28}$$

where the jth component of $\boldsymbol{\mu}(\boldsymbol{\omega})$ is given by

$$\begin{aligned} &\boldsymbol{\mu}(\boldsymbol{\omega})_j = \\ &\left(\mathbf{G}_{\boldsymbol{\omega}}^{-1}(\boldsymbol{\omega})\Big(\nabla_{\boldsymbol{\omega}} \log p\big(\mathbf{f}(\boldsymbol{\omega})\big) + \frac{N}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\omega}} \log p\big(\mathbf{x}_{ti} \mid \mathbf{f}(\boldsymbol{\omega})\big)\Big)\right)_j \\ &\quad - 2\sum_{k=1}^{Q}\left(\mathbf{G}_{\boldsymbol{\omega}}^{-1}(\boldsymbol{\omega})\frac{\partial \mathbf{G}_{\boldsymbol{\omega}}(\boldsymbol{\omega})}{\partial \omega_k}\mathbf{G}_{\boldsymbol{\omega}}^{-1}(\boldsymbol{\omega})\right)_{jk} \\ &\quad + \sum_{k=1}^{Q}\big(\mathbf{G}_{\boldsymbol{\omega}}^{-1}(\boldsymbol{\omega})\big)_{jk}\mathrm{Tr}\left(\mathbf{G}_{\boldsymbol{\omega}}^{-1}(\boldsymbol{\omega})\frac{\partial \mathbf{G}_{\boldsymbol{\omega}}(\boldsymbol{\omega})}{\partial \omega_k}\right). \end{aligned} \tag{29}$$

The matrix $G_{\boldsymbol{\omega}}$ is computed by the GP-LVM according to eq. 24. This is an important theoretical contribution of our paper: if the update is performed in the latent space, the Riemannian metric tensor can be accessed and exploited to hopefully increase the quality of the updates and the exploration of the (original) observed space. The proposed algorithm (AdaGeo Stochastic Gradient Riemannian Langevin Dynamics) is described in algorithm 2, with AdaGeo-SGRLD updates instead of AdaGeo-SGLD ones.

5 RESULTS

This section contains the experimental results: in section 5.1 AdaGeo-SGLD is used to sample from a "banana" distribution; section 5.2 shows how a AdaGeo-powered version of SGD can be used to speed up training of a small neural network; in section 5.3 it is possible to find an application of AdaGeo to gradient descent to train a Gaussian Process.

5.1 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

$$p(\boldsymbol{\theta}) \propto \exp\left(-\frac{\theta_1^2}{200} - \frac{(\theta_2 - b\theta_1^2 + 100b)^2}{2} - \sum_{j=3}^{D}\theta_j^2\right), \tag{30}$$

which is a particularly interesting density to sample from: the only observable interaction occurs between the first two components of $\boldsymbol{\theta}$ (visible in figure 1), masked by the remaining components which act as Gaussian noise. First, a vanilla Metropolis-Hastings sampler is run for 35,000 steps, with 10,000 samples discarded as burn-in and a thinning factor of 250, yielding 100 samples. Due to the high dimensionality of the problem, samplers relying on Langevin dynamics would require prohibitive amounts of time to reach convergence, hence the choice of the simple Metropolis-Hastings. Afterwards, a GP-LVM model is trained on the 100 samples, using squared exponential kernel and latent dimension $Q = 5$. In this experiment

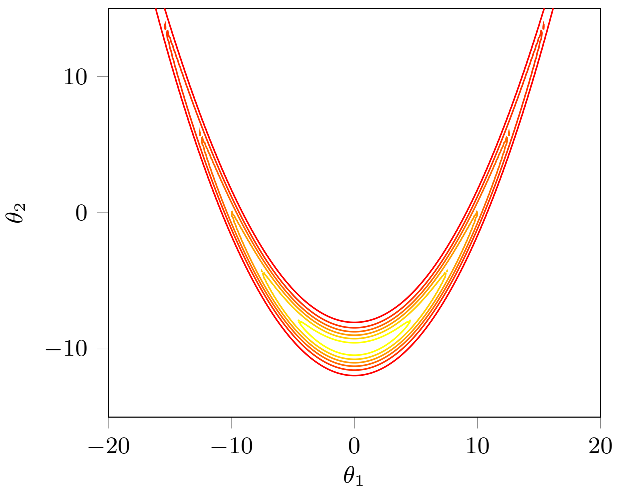

Figure 1: Contour plot showing the level curves of the density generated by a 50-dimensional banana distribution. The figure displays the interaction between the first and second components of $\boldsymbol{\theta}$.

and in the following ones, Q is chosen based on the best empirical results (an alternative to explore would be using an ARD kernel to identify the meaningful latent components). At this point it is possible to sample directly from the 5-dimensional latent space, using then the probabilistic mapping to move back to the real "observed" space. The update in the latent space is computed using the SGLD rule (section 4.2.1), whose functioning is made possible by the reduced dimensionality. 1000 burn in iterations are allowed before starting to sample 250 samples, with a thinning factor of 100 (here the total number of steps is 26,000). Notice that in this case we don't alternate classic and latent updates. Results are shown in figure 2 and 3: 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 (e.g. the banana-shaped structure in fig. 2 and the noise in fig. 3). 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 the supplementary material (appendix A) and show quantitatively the improvement in samples quality.

5.2 Logistic Regression on MNIST with Neural Networks

The second experiment aims at speeding up the training algorithm for a neural network used for classification on the MNIST dataset, while showing the advantages that our method could bring to optimization. Logistic regression on MNIST is implemented through a simple fully connected neural network, consisting of a single layer of 784 nodes featuring a sigmoid activation function. The total num-

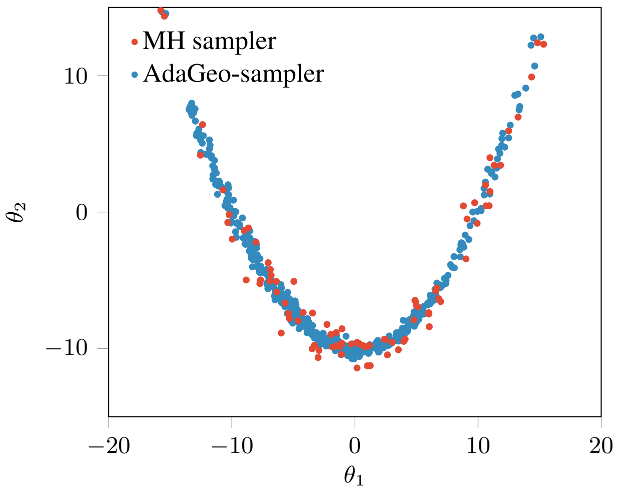

Figure 2: First, our AdaGeo-SGLD sampler trains on the samples returned by a first run of a Metropolis-Hastings algorithm (samples in red). Thanks to those, a lower-dimensional latent space is identified and then sampled from using a SGLD method. The samples obtained using a 5-dimensional latent space are plotted in blue.

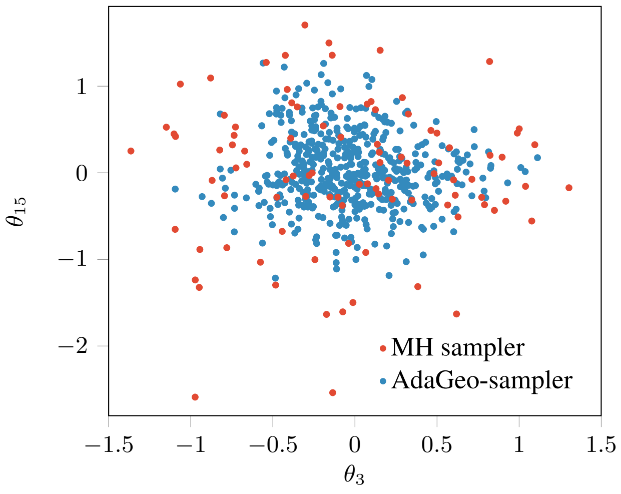

Figure 3: Confounding noise happening between the components of $\boldsymbol{\theta}$. As an example, we show the samples of θ_3 against θ_{15}, but this kind of plot would be similar for every pair of components.

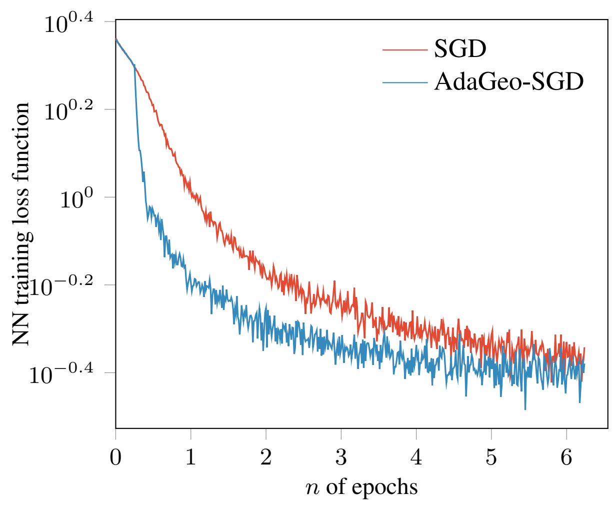

Figure 4: Neural network loss function during training, obtained with vanilla SGD (red) and AdaGeo-SGD (blue), as described in section 5.2.

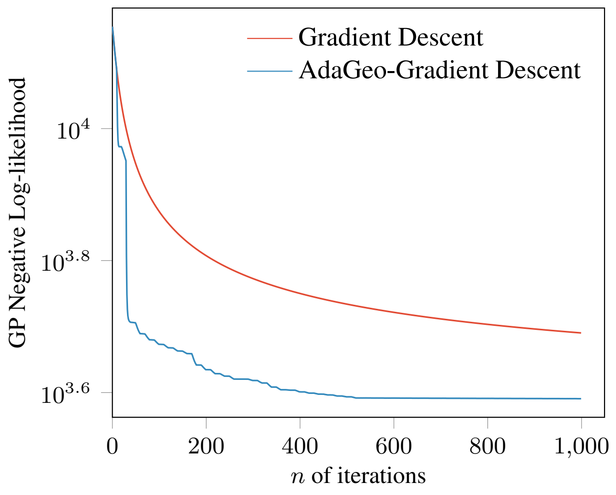

Figure 5: Negative log-likelihood during training of a Gaussian Process, as described in section 5.3. A latent space of dimension 3 is used here to optimize a function whose parameters lie in a 9-dimensional space.

ber of weights that such a network has is 7,850. Out of the 60,000 MNIST images, 50,000 are used for training and 10,000 for testing. During training, cross-entropy loss is minimized. As a fair comparison, we choose to train the network using vanilla stochastic gradient descent and the corresponding AdaGeo version, described as algorithm 1. For the MNIST experiment we have chosen $T_{\boldsymbol{\theta}} = 20$ and $T_{\boldsymbol{\omega}} = 30$, while we have used a surprisingly low-dimensional latent space with $Q = 9$. Results are shown in figure 4: with AdaGeo-SGD updates we are able to outperform vanilla SGD and reach the local optimum almost 2 epochs in advance.

5.3 Gaussian Process Training

The third and last experiment demonstrates further the advantages that AdaGeo could bring to optimization. Here we make use of the Concrete Compressive Strength Data Set [Yeh, 1998]: the model trains on 8 quantitative variables in order to predict a scalar real quantity. The idea is to use a simpler kernel (the GP-LVM one, squared exponential) to fit the hyperparameters of a more complex one. The latter is composed by a linear composition of different kernels [Rasmussen and Williams, 2006]: 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.

We run both gradient descent with Nesterov momentum and AdaGeo-gradient descent for 1000 iterations, 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 5, AdaGeo successfully improves the performances of gradient descent and significantly speeds up training.

6 CONCLUSION

We propose a novel method that introduces dimensionality reduction techniques, namely Gaussian process latent variable models, into optimization and sampling algorithms, in order to boost their performances and overcome the issues deriving from high dimensionality (non-convexity, correlations, different scales etc.). In particular, we show how to include our approach in existing (gradient-based) optimization and sampling (stochastic gradient Langevin dynamics, stochastic gradient Riemannian Langevin dynamics) algorithms. We prove that in this way, by identifying the underlying topological structure of the variable space, sampling and optimization performances are significantly improved. 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, Riemannian stochastic gradient descent). The discussion of the particular formulations of those fall beyond the scope of this paper and will be the subject of future research efforts.

ACKNOWLEDGMENTS

The project was funded by the NCSML PDRA grant 2016/2017. GA was supportd by University of Oxford, Google DeepMind. SRF was supported by the ERC (FP7/617071).

References

Anderson, T. W. (1946). “The Non-Central Wishart Distribution and Certain Problems of Multivariate Statistics”. In: *The Annals of Mathematical Statistics* 17.4, pp. 409–431.

Andrieu, Christophe, Arnaud Doucet, and Roman Holenstein (2010). “Particle markov chain monte carlo methods”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72.3, pp. 269–342.

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

Carpenter, Bob et al. (2016). “Stan: A probabilistic programming language”. In: *Journal of Statistical Software* 20, pp. 1–37.

Do Carmo, Manfredo Perdigao and J Flaherty Francis (1992). *Riemannian geometry*. Vol. 115. Birkhäuser Boston.

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

Duvenaud, David et al. (2014). “Avoiding pathologies in very deep networks”. In: *Artificial Intelligence and Statistics*, pp. 202–210.

Gelman, Andrew et al. (2013). *Bayesian data analysis*. CRC press.

Girolami, Mark and Ben Calderhead (2011). “Riemann manifold langevin and hamiltonian monte carlo methods”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.2, pp. 123–214.

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

Kingma, Diederik and Jimmy Ba (2014). “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980*.

Lawrence, Neil (2005). “Probabilistic non-linear principal component analysis with Gaussian process latent variable models”. In: *Journal of machine learning research* 6.Nov, pp. 1783–1816.

Metropolis, Nicholas et al. (1953). “Equation of state calculations by fast computing machines”. In: *The journal of chemical physics* 21.6, pp. 1087–1092.

Neal, Radford M et al. (2011). “MCMC using Hamiltonian dynamics”. In: *Handbook of Markov Chain Monte Carlo* 2.11.

Patterson, Sam and Yee Whye Teh (2013). “Stochastic gradient Riemannian Langevin dynamics on the probability simplex”. In: *Advances in Neural Information Processing Systems*, pp. 3102–3110.

Rasmussen, Carl Edward and Christopher KI Williams (2006). *Gaussian processes for machine learning*. Vol. 1. MIT press Cambridge.

Robbins, Herbert and Sutton Monro (1951). “A stochastic approximation method”. In: *The annals of mathematical statistics*, pp. 400–407.

Robert, Christian P (2004). *Monte carlo methods*. Wiley Online Library.

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

Tolpin, David et al. (2016). “Design and Implementation of Probabilistic Programming Language Anglican”. In: *Proceedings of the 28th Symposium on the Implementation and Application of Functional Programming Languages*. ACM, p. 6.

Tosi, Alessandra et al. (2014). “Metrics for probabilistic geometries”. In: *Uncertainty in Artificial Intelligence*, p. 800.

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

Yeh, I-C (1998). “Modeling of strength of high-performance concrete using artificial neural networks”. In: *Cement and Concrete research* 28.12, pp. 1797–1808.

Zeiler, Matthew D (2012). “ADADELTA: an adaptive learning rate method”. In: *arXiv preprint arXiv:1212.5701*.