AdaGeo: Adaptive Geometric Learning for Optimization and Sampling

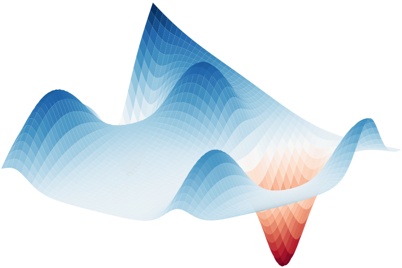
Gabriele Abbati[1], Alessandra Tosi[2], Seth Flaxman[3], Michael A Osborne[1]

[1]University of Oxford, [2]Mind Foundry Ltd, [3]Imperial College London

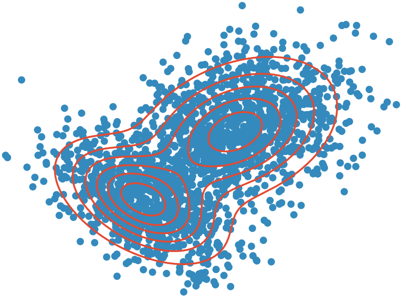
Afternoon Meeting on Bayesian Computation 2018
University of Reading

High-dimensional Problems

- **Gradient-based optimization**



- **MCMC Sampling**



Issues arising from high dimensionality:

- non-convexity
- strong correlations
- multimodality

Related Work

Gradient-based optimization

- AdaGrad
- AdaDelta
- Adam
- RMSProp

MCMC Sampling

- Hamiltonian Monte Carlo
- Particle Monte Carlo
- Stochastic gradient Langevin dynamics

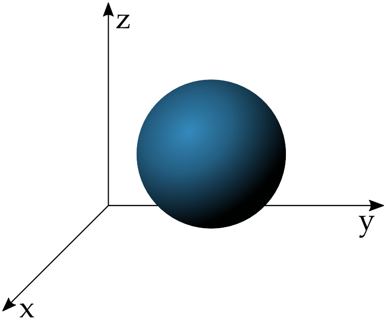
All of these methods focus on computing clever updates for optimization algorithms or for Markov chains.

Novelty: to the best of our knowledge, no dimensionality reduction approaches were applied in this direction before.

The Manifold Idea

After t steps of optimization or sampling, we assume the obtained points in the parameter space to be lying on a **manifold**.

We then feed them to a dimensionality reduction method to find a **lower-dimensional representation**.



3D example: if the sampler/optimizer algorithm keeps on returning proposals on a sphere surface, that information might be used to our advantage

Can we perform better if the algorithm acts with **knowledge** of the manifold?

Latent Variable Models

Latent Variable Models describe a set $\boldsymbol{\Theta}$ through a lower-dimensional latent set $\boldsymbol{\Omega}$

Latent Variable Models

$$\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N \ \in \mathbb{R}^D\} \xleftarrow[\text{with } Q < D]{\text{map } f} \boldsymbol{\Omega} = \{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_N \ \in \mathbb{R}^Q\}$$

where:

- $\boldsymbol{\theta}$: observed variables/parameters
- $\boldsymbol{\omega}$: latent variables
- f: mapping
- D, Q: dimensionalities of $\boldsymbol{\Theta}$ and $\boldsymbol{\Omega}$ respectively

Latent Variable Models

Latent Variable Models describe a set $\boldsymbol{\Theta}$ through a lower-dimensional latent set $\boldsymbol{\Omega}$

Latent Variable Models

$$\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_N \in \mathbb{R}^D\} \xleftarrow[\text{with } Q < D]{\text{map } f} \boldsymbol{\Omega} = \{\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_N \in \mathbb{R}^Q\}$$

mapping: $\boldsymbol{\theta} = f(\boldsymbol{\omega}) + \boldsymbol{\eta}$ with $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \beta^{-1}\mathbf{I})$

Dimensionality reduction $\longrightarrow$ **Manifold identification**

The lower-dimensional manifold on which the samples lie is characterized through the latent set

Gaussian Process Latent Variable Model

The choice of the dimensionality reduction method fell on the **Gaussian Process Latent Variable Model**[1].

GPLVM: Gaussian Process prior over mapping f in

$$\boldsymbol{\theta} = f(\boldsymbol{\omega}) + \boldsymbol{\eta}$$

Motivation:

- Analytically sound mathematical tool
- Full distribution over the mapping f
- Full distribution over the derivatives of the mapping f

[1]Lawrence, N., Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of machine learning research* (2005)

Gaussian Process

Gaussian Process[2]: a collection of random variables, any finite number of which have a joint Gaussian distribution.

If a real-valued stochastic process f is a GP, it will be denoted as

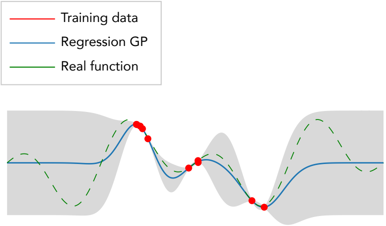
$$f(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot))$$

A Gaussian Process is fully specified by

- a mean function $m(\cdot)$
- a covariance function $k(\cdot, \cdot)$

where

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



[2]Rasmussen, C. E., Williams, C. K. I., Gaussian Processes for Machine Learning, *the MIT Press* (2006)

Gaussian Process Latent Variable Model

GPLVM: Gaussian Process prior over mapping f in

$$\boldsymbol{\theta} = f(\boldsymbol{\omega}) + \boldsymbol{\eta}$$

The likelihood of the data $\boldsymbol{\Theta}$ given the latent $\boldsymbol{\Omega}$ is given by

1. marginalizing the mapping
2. optimizing the latent variables

Resulting likelihood:

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

With the resulting noise model being: $\theta_{i,j} = \tilde{\mathbf{K}}_{(\boldsymbol{\omega}_i, \boldsymbol{\Omega})} \tilde{\mathbf{K}}^{-1} \boldsymbol{\Theta}_{:,j} + \eta_j$

Gaussian Process Latent Variable Model

GPLVM: Gaussian Process prior over mapping f in

$$\boldsymbol{\theta} = f(\boldsymbol{\omega}) + \boldsymbol{\eta}$$

For differentiable kernels $k(\cdot, \cdot)$, the Jacobian $\mathbf{J}$ of the mapping f can be computed analytically:

$$J_{ij} = \frac{\partial f_i}{\partial \omega_j}$$

But as previously said, GPLVM can yield the full (Gaussian) distribution over the Jacobian. If the rows of $\mathbf{J}$ are assumed to be independent:

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

Recap

1. After t iterations the optimization or sampling algorithm has yielded a set of observed points $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N \in \mathbb{R}^D\}$ in the parameter space

2. A GPLVM is trained on Θ in order to build a latent space Ω that describes the lower-dimensional manifold on which the optimization/sampling is allegedly taking place. We can:

   - move from the latent space Ω to the observed space Θ:

     $$\theta = f(\omega) + \eta$$
     $$\Theta \leftarrow \Omega$$

     but not viceversa (f is not invertible)

   - bring the gradients of a generic function $g : \Theta \to \mathbb{R}$ from the observed space Θ to the latent space Ω:

     $$\nabla_\omega g(f(\omega)) = J \nabla_\theta g(\theta)$$
     $$\Omega \leftarrow \Theta$$

     In this case a punctual estimate of J is given by the mean of its distribution.

Recap

1. After t iterations the optimization or sampling algorithm has yielded a set of observed points $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N \in \mathbb{R}^D\}$ in the parameter space

2. A GPLVM is trained on $\boldsymbol{\Theta}$ in order to build a latent space $\boldsymbol{\Omega}$ that describes the lower-dimensional manifold on which the optimization/sampling is allegedly taking place. We can:
   - **move** from the latent space $\boldsymbol{\Omega}$ to the observed space $\boldsymbol{\Theta}$:

     $$\boldsymbol{\theta} = f(\boldsymbol{\omega}) + \boldsymbol{\eta}$$
     $$\boldsymbol{\Theta} \leftarrow \boldsymbol{\Omega}$$

     but not viceversa (f is not invertible)

   - bring the **gradients** of a generic function $g : \boldsymbol{\Theta} \to \mathbb{R}$ from the observed space $\boldsymbol{\Theta}$ to the latent space $\boldsymbol{\Omega}$:

     $$\nabla_{\boldsymbol{\omega}} g(\mathbf{f}(\boldsymbol{\omega})) = \boldsymbol{\mu}_{\mathbf{J}} \nabla_{\boldsymbol{\theta}} g(\boldsymbol{\theta})$$
     $$\boldsymbol{\Omega} \leftarrow \boldsymbol{\Theta}$$

     In this case a punctual estimate of $\mathbf{J}$ is given by the mean of its distribution.

Recap

1. After t iterations the optimization or sampling algorithm has yielded a set of observed points $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N \in \mathbb{R}^D\}$ in the parameter space

2. A GPLVM is trained on $\boldsymbol{\Theta}$ in order to build a latent space $\boldsymbol{\Omega}$ that describes the lower-dimensional manifold on which the optimization/sampling is allegedly taking place. We can:
   - **move** from the latent space $\boldsymbol{\Omega}$ to the observed space $\boldsymbol{\Theta}$:

$$\boldsymbol{\theta} = f(\boldsymbol{\omega}) + \boldsymbol{\eta}$$
$$\boldsymbol{\Theta} \leftarrow \boldsymbol{\Omega}$$

     but not viceversa (f is not invertible)

   - bring the **gradients** of a generic function $g : \boldsymbol{\Theta} \to \mathbb{R}$ from the observed space $\boldsymbol{\Theta}$ to the latent space $\boldsymbol{\Omega}$:

$$\nabla_{\omega} g(\mathbf{f}(\omega)) = \boldsymbol{\mu}_{\mathbf{J}} \nabla_{\theta} g(\theta)$$
$$\boldsymbol{\Omega} \leftarrow \boldsymbol{\Theta}$$

     In this case a punctual estimate of $\mathbf{J}$ is given by the mean of its distribution.

Recap

1. After t iterations the optimization or sampling algorithm has yielded a set of observed points $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N \ \in \mathbb{R}^D\}$ in the parameter space

2. A GPLVM is trained on $\boldsymbol{\Theta}$ in order to build a latent space $\boldsymbol{\Omega}$ that describes the lower-dimensional manifold on which the optimization/sampling is allegedly taking place. We can:
   - **move** from the latent space $\boldsymbol{\Omega}$ to the observed space $\boldsymbol{\Theta}$:

     $$\boldsymbol{\theta} = f(\boldsymbol{\omega}) + \boldsymbol{\eta}$$
     $$\boldsymbol{\Theta} \leftarrow \boldsymbol{\Omega}$$

     but not viceversa (f is not invertible)

   - bring the **gradients** of a generic function $g : \boldsymbol{\Theta} \to \mathbb{R}$ from the observed space $\boldsymbol{\Theta}$ to the latent space $\boldsymbol{\Omega}$:

     $$\nabla_{\boldsymbol{\omega}} g\big(\mathbf{f}(\boldsymbol{\omega})\big) = \boldsymbol{\mu}_{\mathbf{J}} \nabla_{\boldsymbol{\theta}} g(\boldsymbol{\theta})$$
     $$\boldsymbol{\Omega} \leftarrow \boldsymbol{\Theta}$$

     In this case a punctual estimate of $\mathbf{J}$ is given by the mean of its distribution.

AdaGeo Gradient-based Optimization

Minimization problem:

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

Iterative scheme solution (e.g. (stochastic) gradient descent):

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

We propose, after having learned a latent representation with GPLVM, to move the problem onto the latent space Ω

Minimization problem:

$$\omega^* = \arg\min_{\omega} g(f(\omega))$$

Iterative scheme solution (e.g. (stochastic) gradient descent):

$$\omega_{t+1} = \omega_t - \Delta\omega_t(\nabla_{\omega} g)$$

AdaGeo Gradient-based Optimization

Minimization problem:

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

Iterative scheme solution (e.g. (stochastic) gradient descent):

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

We propose, after having learned a latent representation with GPLVM, to move the problem onto the latent space $\boldsymbol{\Omega}$

Minimization problem:

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} g(f(\boldsymbol{\omega}))$$

Iterative scheme solution (e.g. (stochastic) gradient descent):

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

AdaGeo Gradient-based Optimization

Algorithm 1 AdaGeo gradient-based optimization (minimization)

1: **while** convergence is not reached **do**

2: Perform $T_{\boldsymbol{\theta}}$ iterations with classic updates on the parameter space $\boldsymbol{\Theta}$:

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

3: Train the GP-LVM model on the samples $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{T_{\boldsymbol{\theta}}}\}$

4: Continue performing $T_{\boldsymbol{\omega}}$ using the AdaGeo optimizer:

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

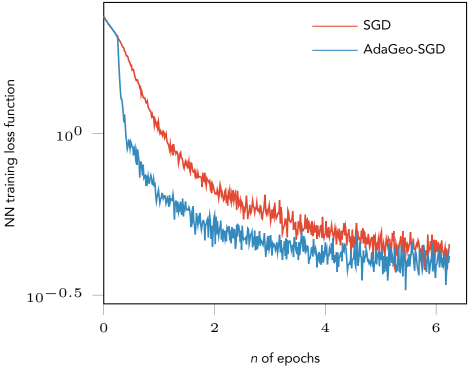
and moving back to the parameter space with

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

5: **end while**.

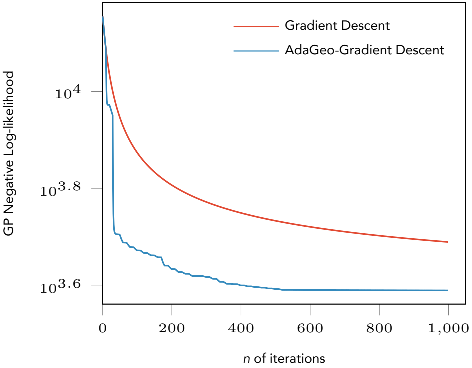
Experiment: Logistic Regression on MNIST

- Neural Network with a single hidden layer implementing logistic regression on MNIST
- Dimension of parameter space: $D = 7850$
- Dimension of latent space: $Q = 9$
- Iterations: $T_{\boldsymbol{\theta}} = 20$ and $T_{\boldsymbol{\omega}} = 30$



Experiment: Gaussian Process Training

- Concrete compressive strength dataset[3]: regression task with 8 real variables
- Composite kernel (RBF, Matérn, linear and bias)
- Dimension of parameter space: $D = 9$
- Dimension of latent space: $Q = 3$
- Iterations: $T_{\boldsymbol{\theta}} = 15$ and $T_{\boldsymbol{\omega}} = 15$



[3]Lichman, M., UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science (2013)

AdaGeo Bayesian Sampling

Bayesian sampling framework:

- a **dataset** $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is given
- $\mathbf{X}$ is modeled with a **generative model** whose likelihood is

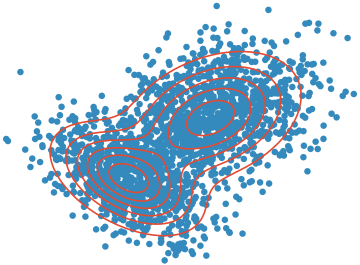
$$p(\mathbf{X}, \boldsymbol{\theta}) = \prod_{i=1}^{N} p(\mathbf{x}_i, \boldsymbol{\theta}),$$

parameterized by the vector $\boldsymbol{\theta} \in \mathbb{R}^D$, with prior $p(\boldsymbol{\theta})$.

Performing statistical inference means getting insights on the **posterior distribution**

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

analytically or **approximately** through **sampling**.



AdaGeo Bayesian Sampling

Bayesian sampling framework:

- a **dataset** $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is given
- $\mathbf{X}$ is modeled with a **generative model** whose likelihood is

$$p(\mathbf{X}, \boldsymbol{\theta}) = \prod_{i=1}^{N} p(\mathbf{x}_i, \boldsymbol{\theta}),$$

parameterized by the vector $\boldsymbol{\theta} \in \mathbb{R}^D$, with prior $p(\boldsymbol{\theta})$.

Unfortunately the denominator is often intractable. One possible approach is to approximate the integral

$$p(\mathbf{X}) = \int_{\boldsymbol{\Theta}} p(\mathbf{X}, \boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} = \int_{\boldsymbol{\Theta}} p(\mathbf{X}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}$$

through **Markov Chain Monte Carlo** or similar methods.

Stochastic Gradient Langevin Dynamics

Stochastic gradient Langevin dynamics[4] combines **stochastic optimization** and the physical concept of **Langevin dynamics** to build a posterior sampler

At each time t a mini-batch is extracted and the parameters are updated as:

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

with the learning rate ϵ_t satisfying:

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

[4]Welling, M., Teh, Y. W., Bayesian learning via stochastic gradient Langevin dynamics. *Proceedings of the 28th International Conference on Machine Learning (ICML)* (2011)

AdaGeo - Stochastic Gradient Langevin Dynamics

Analogously as before, we propose to:

1. Pick your favourite sampler and produce the first t samples to build the set $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N\}$
2. Train a GPLVM on $\boldsymbol{\Theta}$ to learn the latent space $\boldsymbol{\Omega}$
3. Move the updates onto the latent space with **AdaGeo - Stochastic Gradient Langevin Dynamics**:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t + \Delta\boldsymbol{\omega}_t,$$

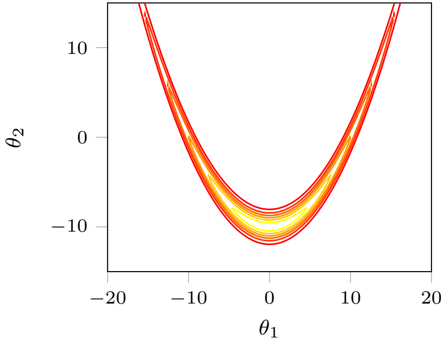
$$\Delta\boldsymbol{\omega}_t = \frac{\epsilon_t}{2}\left(\nabla_\omega \log p\big(f(\boldsymbol{\omega}_t)\big) + \frac{N}{n}\sum_{i=1}^{n} \nabla_\omega \log p\big(\mathbf{x}_{ti}|f(\boldsymbol{\omega}_t)\big)\right) + \boldsymbol{\eta}_t,$$

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

Experiment: Sampling from the Banana Distribution

The "banana" distribution has this formula:

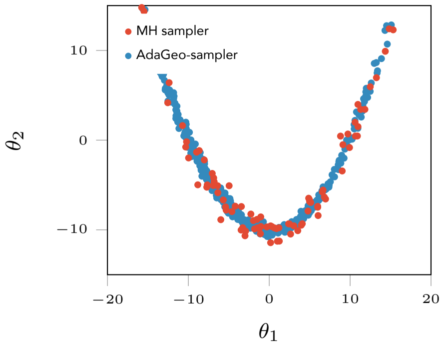
$$p(\boldsymbol{\theta}) \propto \exp\left(-\frac{\theta_1^2}{200} - \frac{(\theta_2 - b\theta_1^2 + 100b)^2}{2} - \sum_{j=3}^{D} \theta_j^2\right)$$

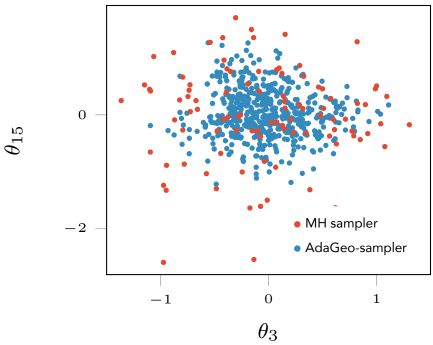


θ_1 and θ_2 present the interaction shown on the left, while the other variables produce Gaussian noise

Experiment: Sampling from the Banana Distribution

- A Metropolis-Hastings returns the first 100 samples drawn from a 50-dimensional banana distribution
- AdaGeo-SGLD is then employed to sample from a 5-dimensional latent space





Bonus Round: Riemannian Extensions (theory only)

If a **covariance function** of a Gaussian Process is **differentiable**, then it is straightforward to show that the **mapping** f is also **differentiable**.

Under this assumption we can compute the latent **metric tensor G**, which will give further information about the **geometry** of the latent space (distances, geodetic lines etc.)

If **J** is the Jacobian of the mapping f, then

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

This yields a distribution over the metric tensor[5]:

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

and a punctual estimate can be obtained with

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

[5]Tosi, A., Hauberg, S., Vellido, A., Lawrence, N. D., Metrics for probabilistic geometries. *Uncertainty in Artificial Intelligence* (2014)

Bonus Round: Stochastic Gradient Riemannian Langevin Dynamics

Stochastic gradient Riemannian Langevin dynamics[6] puts together the advantages of exploiting a known **Riemannian geometry** with the scalability of the **stochastic optimization** approaches:

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

where

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

[6] Patterson, S., Teh, Y. W., Stochastic gradient Riemannian Langevin dynamics on the probability simplex. *Advances in Neural Information Processing Systems* (2013)

Bonus Round: AdaGeo - Stochastic Gradient Riemannian Langevin Dynamics

Analogously as the SGLD case, we can move now the update in the latent space with **AdaGeo - Stochastic gradient Riemannian Langevin dynamics**:

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

where

$$
\boldsymbol{\mu}(\boldsymbol{\omega})_j = \left(\mathbf{G}_{\omega}^{-1}(\boldsymbol{\omega})\left(\nabla_{\omega} \log p(\mathbf{f}(\boldsymbol{\omega})) + \frac{N}{n}\sum_{i=1}^{n} \nabla_{\omega} \log p(\mathbf{x}_{ti} \mid \mathbf{f}(\boldsymbol{\omega}))\right)\right)_j
$$

$$
-2\sum_{k=1}^{Q}\left(\mathbf{G}_{\omega}^{-1}(\boldsymbol{\omega})\frac{\partial \mathbf{G}_{\omega}(\boldsymbol{\omega})}{\partial \omega_k}\mathbf{G}_{\omega}^{-1}(\boldsymbol{\omega})\right)_{jk} + \sum_{k=1}^{Q}\left(\mathbf{G}_{\omega}^{-1}(\boldsymbol{\omega})\right)_{jk}\mathrm{Tr}\left(\mathbf{G}_{\omega}^{-1}(\boldsymbol{\omega})\frac{\partial \mathbf{G}_{\omega}(\boldsymbol{\omega})}{\partial \omega_k}\right)
$$

Conclusions

- 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 **Markov Chain Monte Carlo** by developing a AdaGeo version of stochastic gradient Langevin dynamics; the information gathered through the latent space are employed to compute the steps of the Markov chain
- We extend the approach to stochastic gradient **Riemannian** Langevin dynamics, thanks to the geometric tensor naturally recovered by the GP-LVM model

Thank you

Reference: Abbati, G., Tosi, A., Flaxman, S., Osborne, M. A. (2018). AdaGeo: Adaptive Geometric Learning for Optimization and Sampling. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS), to appear*.