Statistics Under Stochastic Metrics

Pedersen, Cilie Werner Feldager

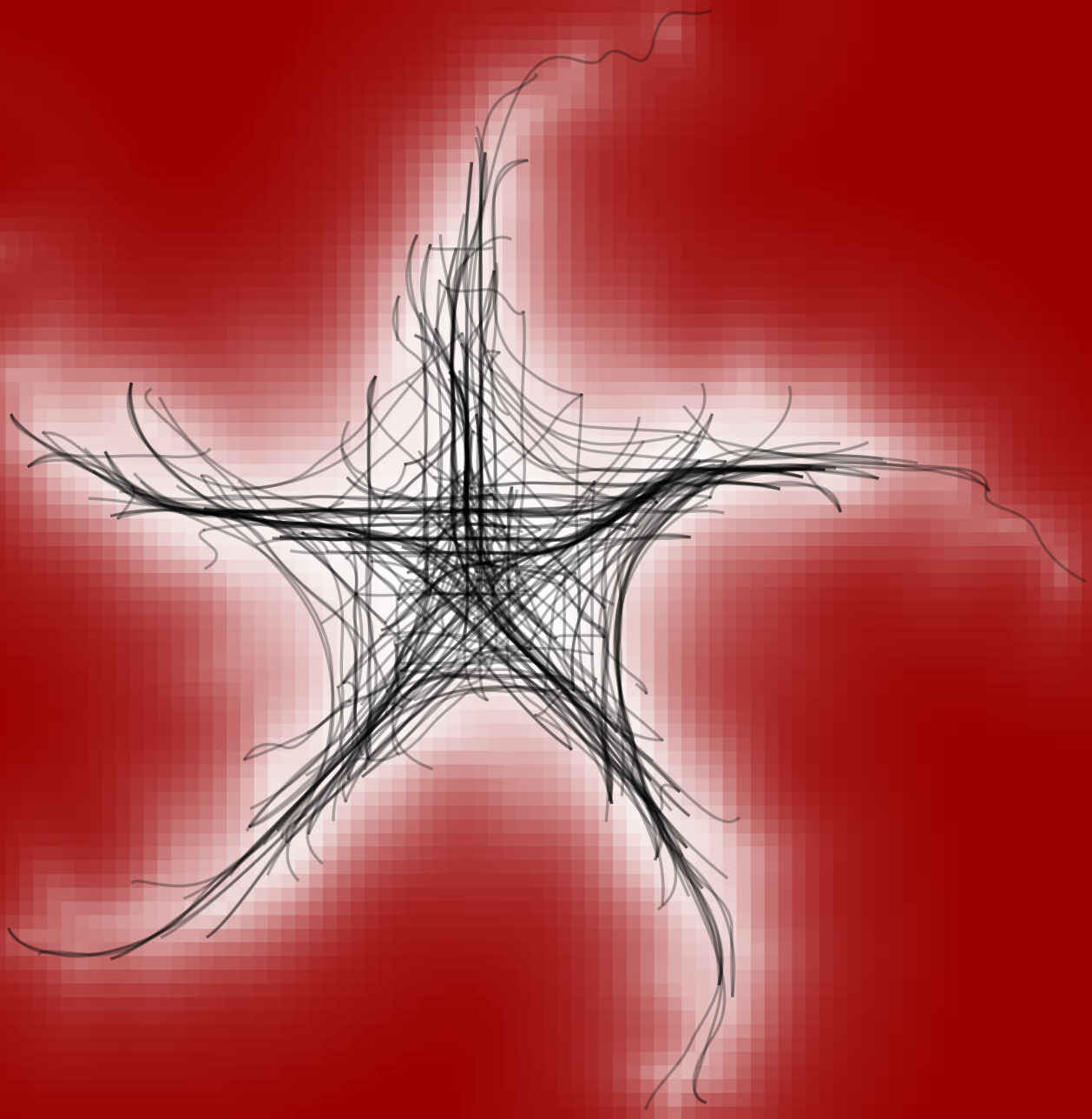
Publication date:
2023

Document Version
Publisher's PDF, also known as Version of record

DTU COMPUTE
DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTER SCIENCE

STATISTICS UNDER STOCHASTIC METRICS

CILIE W. FELDAGER
PH.D. THESIS



PH.D. THESIS
STATISTICS UNDER STOCHASTIC METRICS
BY CILIE W. FELDAGER

COGNITIVE SYSTEMS
DTU COMPUTE
TECHNICAL UNIVERSITY OF DENMARK

SUPERVISED BY
PROFESSOR SØREN HAUBERG
PROFESSOR LARS KAI HANSEN

To Jesper and Filippa

ABSTRACT

Probabilistic unsupervised learning aims to capture the generating distribution of data. In this setting, learning an interpretable model is challenging. Many models assume that the generating distribution is inherently Euclidean. We may achieve more insight by relaxing this constraint on the generating function. By considering, e.g. Riemaniann geometry, we can compute *more* distance. Our goal is to learn the geometry of data which allows for computing distances under Riemannian geometries to achieve more interpretable models.

Topology is the first step to capturing the geometry of data, and specifically, we employ symmetries in data as a proxy for topologies. We develop a simple workflow for detecting symmetries using tools from topological data analysis and investigate if the symmetry is preserved under dimensionality reduction in four models. Our quantitative study shows that these algorithms frequently break symmetry, highlighting current visualisation tools' shortcomings. This is somewhat concerning, but our results indicate that the likelihood is a good indicator for preserved topology in the Gaussian process latent variable model.

Decoders built on Gaussian processes (GPs) are enticing due to the marginalisation over the non-linear function space) which further motivates our work with the Gaussian process latent variable model (GPLVM). Such models are expensive and, therefore, often scaled with variational inference and inducing points, but these are challenging to train. We develop the stochastic active sets approximation, a scalable and robust training scheme for GPLVMs that leads to interesting latent representations with more structure than the Bayesian GPLVM and comparable variational autoencoders.

We attempt to build the geometry into the prior in a GPLVM. We develop the computational framework for this model, and we consider the Riemannian Brownian motion as a suitable choice fo prior for this purpose. We fit this to a GP manifold, and though we have the needed components, we fail to implement training of the model.

The intention of capturing geometry is to capture the essence of the generating process of observations. This has the potential to synthesise patterns in large amounts of data which humans would otherwise be unable to grasp and provides insights in a humanly interpretable format. This could enable humans to learn from machine learning.

DANSK RESUMÉ

Målet med probabilistisk usuperviseret læring er at fange den genererende distribution af data. I denne forbindelse er det en udfordring at lære en model, som kan fortolkes. Mange modeller antager, at den genererende distribution i sagens natur er Euklidisk. Vi argumenterer for, at vi kan opnå mere indsigt ved at lade den genererende function være ikke-Euklidisk- Ved at overveje f.eks. Riemannsk geometry kan vi beregne *mere* meningsfyldte afstande. Vores mål er at lære geometrien af data (ved at beregne afstande under Riemannske geometrier) for at opnå modeller, der i højere grad kan tolkes.

Topology er grundlaget for for geometri of specifikt anvender vi symmetrier i data som en proxy for topologi. Vi udvikler en simpel arbejdsgang til detektion af symmetrier ved hjælp af værktøjer fra topologisk dataanalyse og undersøger, om symmetrien er bevaret under dimensionalitetsreduktion i fire modeller. Vores kvantitative undersøgelse viser, at disse algoritmer ofte bryder symmetri, hvilket understreger bekymrende mangler ved nuværende visualiseringsværktøjer. Vores resultater tyder på, at likelihooden er brugbar indikator for hvorvidt topologi er bevaret i den Gaussiske proces latent variabel model.

Afkodere bygget på Gaussiske processer (GP'er) er tillokkende på grund af marginaliseringen over det ikke-lineære funktionsrum, hvilket yderligere motiverer vores videre arbejde med den Gaussiske proces latent variabel mode (GPLVM). Sådanne modeller er beregningsmæssigt dyre og skaleres derfor ofte med variationsslutning og inducerende punkter, men disse er udfordrende at træne. Vi udvikler den stokastiske approksimation af aktive sæt, som er en skalerbar og robust træningsalgoritme for GPLVM'er, der fører til interessante latente repræsentationer med mere struktur end både den Bayesianske GPLVM og sammenlignelige variationelle selvindkodere.

Vi forsøger at bygge geometrien ind i prioren i en GPLVM. Vi udvikler beregningsrammen for denne model, og vi betragter den Riemannske Brownske bevægelse som et passende valg af prior til dette formål. Vi tilpasser dette til en GP mangfoldighed, men på trods af at vi har de nødvendige komponenter, har vi endnu ikke formået at træne en model til konvergens.

Hensigten med at studere geometri er at fange essensen af den genererende proces. Dette har potentialet til at syntetisere mønstre i store mængder data, som mennesker ellers ikke ville være i stand til at forstå og give indsigt i et format, der kan tolkes af mennesker. Dette kan muliggøre at mennesker kan blive klogere af maskinlæring.

PREFACE

None of the work in this thesis has been done in isolation. Supervisors, colleagues and previous results have naturally all been instrumental and for this reason, I consistently use the pronoun *we* throughout the thesis.

This thesis touches upon a number of subjects and I have had to be selective in which background material to go through. I would have liked for this thesis to be self-contained, also the parts that might be unfamiliar to the reader, but this has not feasible. In particular, I would have expanded on Gaussian processes, tensor calculus and Riemannian geometry. I have tried to accommodate this shortcoming with introductory references where relevant.

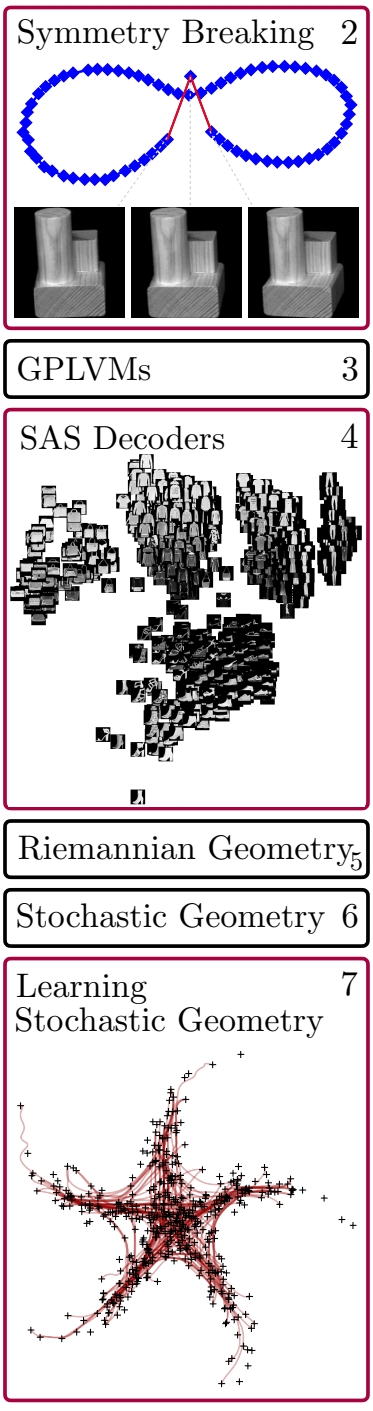
For full transparency, I list what I perceive as contributions here and this should also be clear from the text throughout the thesis. I have tried to display this visually in the figure where the red border indicates contributions. Chapters 2, 4, and 7 contain contributions though the project in chapter 7 is yet uncompleted. Chapter 2 is based on Feldager, Hauberg, and Hansen [1] and chapter 4 is based on Moreno-Muñoz, Feldager, and Hauberg [2].

Chapter 6 contains a correction to the derivation of expected Christoffel symbols first done by Adams [3] which was joint work with Alison Pouplin. The rest of chapter 6 is background material and so are chapters 3 and 5.

Let me end with an anecdote. Just before I started my PhD I explained that I would be looking at stochastic geometry to a then colleague who has a PhD in math. "That does not make any sense!" was her prompt response. To this day, I haven't decided if she was right or not.

New Orleans, 3rd of December 2022

Cilie W. Feldager



ACKNOWLEDGMENTS

I must admit that writing my acknowledgements feels somewhat pretentious because it suggests that I have made valuable contributions. This feeling is entirely overshadowed by the gratitude I feel: Many amazing people have had an implicit (and explicit) hand in the making of this thesis, and they deserve massive thanks.

I sincerely appreciate my great colleagues–former and current: Our secretary Anne–this department would never work without you, and I admire your immense patience with all of us. Georgios whose thesis [4] heavily inspired many of the figures in the geometry part. Martin and Nicki, thanks for helping with the fundamentals when I was a noob PhD student. Dimitris, thanks for your inspiring chaos. Pablo, thanks for working with me on SAS, it was a blast! Alison, thanks for the geometry discussions. Marco, for the philosophical discussions that I suspect, will continue for years. And Pola, for everything in between and in particular feminism. I feel the time was too short, and I will miss you all and the entire group.

I am immensely grateful that the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no° 757360) supported my project. My external stay was supported financially by *Reinholdt W. Jorck og hustrus Fond*, *Oberstløjtnant Max Nørgaard og hustru Magda Nørgaards Legat*, *the Niels Bohr Fondet*, and *Stibofondens rejselegat* for which I am also deeply grateful.

I have had the absolute pleasure of working with brilliant supervisors: I have been co-supervised by Professor Lars Kai Hansen which has been extraordinarily educational. I was fortunate enough that Professor Neil D. Lawrence from the University of Cambridge stuck with me for a year. I would like to thank Neil as well. Both for the opportunity to be part of his new group and to learn from one of the best in the field. In particular, my principal supervisor, Professor Søren Hauberg, has been absolutely amazing. Had it not been for him and his ways, I would not even have ventured into the endeavour: Despite having sworn never to do a PhD, Søren convinced me otherwise and I have never once regretted that decision. Thank you for that!

Finally, a huge thanks to friends and family. For bearing with me and supporting me in this endeavour. I am so privileged that you are too many of you to mention, but you know who you are. I do want to mention Filippa–you light up my days–and Jesper. You introduced me to machine learning and encouraged me to pursue a PhD. You are my rock and without you and your support, this would not have been possible. Thank you for everything. I love you both.

LIST OF PUBLICATIONS

Pablo Moreno-Muñoz*, Cilie W. Feldager*, and Søren Hauberg
Revisiting Active Sets for Gaussian Process Decoders. Sept. 2022.
DOI:10.48550/arXiv.2209.04636.arXiv:2209.04636 [cs, stat]
Poster at Neural Information Processing Systems, 2022.

* Equal contribution

Cilie W. Feldager, Søren Hauberg, and Lars Kai Hansen.
Spontaneous Symmetry Breaking in Data Visualization.
In: Artificial Neural Networks and Machine Learning.
Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)". Springer, 2021, pp. 435–446. isbn: 978-3-030-86339-5.
DOI: 10.1007/978-3-030-86340-1_35

CONTENTS

LIST OF FIGURES

INTRODUCTION 1

Machine learning delegates tedious tasks–often classification and regression problems–to computers that digest vast amounts of potentially high-dimensional data to reveal non-obvious patterns. If humans should learn from data, these patterns must be extracted as humanly interpretable insights; hence, machine learning models must be interpretable.

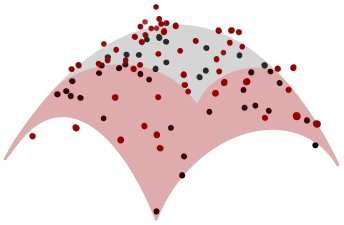


Figure 1.1: An illustration of data clustering around a manifold.

Interpretable models can be particularly challenging in unsupervised learning where the ground truth is unknown. Unsupervised learning often relies on the manifold assumption [5]–which, in simple terms, states that data lie near a manifold–and its goal is often to learn a representation of data. In this thesis, we aim to capture the underlying structure in data and learn from it by learning a representation of data in an unsupervised manner. This goal relies on the manifold hypothesis, and it assumes that our models *can* learn the manifold and that the learnt representation *actually* captures the generating process.

Figure 1.2: A meaningful distance (red) and a Euclidean distance shown in a two-dimensional representation of the globe (top) and a three-dimensional representation (bottom). The figures are made with Google Maps and Google Earth, respectively.

In this thesis, we measure distances between points directly on the manifold and argue that this is *more* meaningful than using Euclidean distances in the representation. To motivate our approach, consider a world map which is a representation of the approximately spherical Earth. On the flat two-dimensional map, the Euclidean distance (the black straight line in the top pane of figure 1.2) seems shorter than the red path, but this is an artefact of the representation. The same distances are shown on the globe, where the red curve appears shorter. In principle, both distances respect the geometry of Earth as they both follow the surface of the Earth but travelling the Euclidean distance corresponds to a detour as the red curve is indeed shorter: One would spend less time and energy to travel this path which–in this case–it makes it more meaningful.

Many models in machine learning inherently assume a Euclidean geometry as they rely on the Euclidean distance measure (e.g. variational autoencoders [6], Isomap [7], Gaussian processes [8] and normalising flows [9]) and with good reason as this is fast to compute. Accounting for non-Euclidean geometries is more challenging and often computationally taxing. But if the goal is to learn *from* data, the representation might benefit by considering non-Euclidean geometries as the models get the capacity to capture non-linear geometries, visualisations may

be more faithful, and a deeper understanding of representations might lead to new discoveries.

In addition to a geometric approach, we take a probabilistic view as this allows for estimating the manifold away from data: Models like isomap and neural networks [4] learn manifolds where there is support from data, but away from data deterministic models provide poor estimates of the manifold. The probabilistic approach allows for well-estimated uncertainty away from data. This uncertainty repels the distance curves so they follow the manifold [10, 11], which drives distances to follow the manifold, letting the uncertainty play the role of topology [12].

The goal of this thesis is to capture meaning from representations by considering geometry and uncertainty jointly. Before getting to that and to meaningful distances, we explore whether learnt representations recover the topology of data, as topology is a prerequisite for geometry. We investigate this in four dimensionality reduction models (chapter 2), including the Gaussian process latent variable model (GPLVM), which chapter 3 dives deeper into. Next, we develop a scalable GPLVM that learns interesting representations. Finally, after introducing elementary geometry (chapter 5), we discuss stochastic geometries (chapter 6), which enables doing *Statistics under Stochastic Metrics* (chapter 7).

SYMMETRY BREAKING IN VISUALISATIONS

2

> *Reality favours symmetry.*
>
> —Jorge Luis Borges

This chapter is based on the ICANN paper by Feldager, Hauberg, and Hansen [1]. Results on MNIST and high-dimensional features are left out.

In our endeavour to find a model suited for geometry, we search for a model that preserves topology, as topology is a prerequisite for geometry. In this chapter, we use a dataset with rotational symmetry and investigate if different visualisation models can recover the symmetry. If we can find a model that respects topology, the hope of finding a model with more meaningful distances persists.

2.1 SYMMETRIES IN VISUALISATIONS

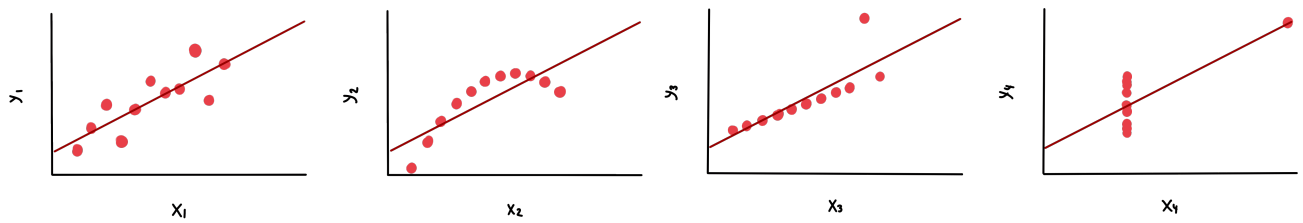


Figure 2.1: Four different datasets known as Anscombe's quartet [13] that have identical linear regressions.

Visualisation is often the first step to analysing a new dataset, which can shape the rest of the data analysis. Data visualisation is an integral part of the machine learning pipeline: From forming hypotheses [14] to communicating results [15, 16]. Visualisations are models that reduce the dimensionality of data to two (or three) dimensions. We cannot always trust models that reduce data to a few parameters: Anscombe's quartet consists of four different datasets with two variables with identical summary statistics. Fitting a linear model to each of the datasets yields the same coefficient and the same R^2. This illustrates that the visualisation of data is paramount.

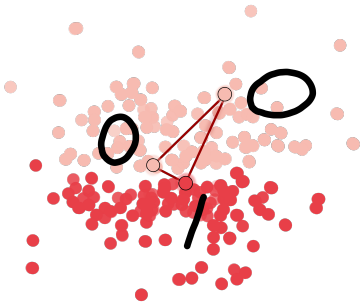


Figure 2.2: A representation of two classes where two zeros and one are shown in black. The Euclidean distance suggests that one of the zeros and the other are closer than the two zeros, but this is not the case. Recreated from Arvanitidis, Hansen, and Hauberg [17]

For example, the visualisation in figure 2.2 shows a representation of high-dimensional images of ones and zeros. We would expect the two zeros to be closer than either of the zeros to the one. The Euclidean distance suggests that one of the zeros and the one is closer, but this does not have to be the case under other distance measures [17].

In this chapter, we focus on visualisations (dimensionality reductions) which rely on the manifold assumption. Testing the manifold assumption is hard, so we focus on datasets with known symmetries and hence known manifolds to illustrate our point. Mathematically, continuous symmetries are represented by Lie groups [18]. We consider rotations in terms of the infinitesimal generator (an infinitesimal rotation), and this is the generating group action. The rotations give rise to a Lie group, a smooth differentiable manifold on which points are connected through a continuous group operation and its inverse. This means that we can view symmetries as manifolds and know that the manifold assumption is true by construction: There exists an underlying manifold that is the data generator. And more, we know which manifold. This enables the detection in a supervised manner, using standard visualisation techniques to see if symmetries are preserved. This is achieved by verifying if the topology of the group remains intact under visualisations.

This chapter focuses on situations where the governing group is known and investigates if standard visualisation techniques preserve the group structure. This is achieved by verifying if the symmetry remains intact under visualisations.

2.2 TOPOLOGICAL DATA ANALYSIS

This section introduces topological data analysis. Topological data analysis (TDA) offers a range of tools that can be used to study the breaking of symmetry. TDA extends topology to data. In the next section, we will use topological data analysis (TDA) to quantify high-dimensional symmetries. Feel free to skip this section if you are familiar with persistent homology, Betti numbers, the Vietoris-Rips complex, and barcodes. These notions are formalised in algebraic topology (Hatcher [19] gives a thorough introduction). This introduction follows Carlsson [20] and [21].

We employ tools from topological data analysis to verify that the topology of data is preserved under dimensionality reduction. TDA is an algebraic tool that allows for studying features of point clouds and their graph structures and provides qualitative information about data. The goal of homology is to quantify the topological features (e.g.

number of connected components, holes and voids) of point clouds in a principled manner. First, we introduce a scale-dependent simplicial complex (the Vietoris-Rips complex) that allows mapping the point cloud to a set of scalars that summarise the topology (Betti numbers). Finally, we discuss barcodes which rid us of the scale-dependency.

Definition 2.1 (Metric Space) *Given a distance $d : M \times M \to \mathbb{R}$, the ordered pair (M, d) is a metric space. The distance d is non-negative $d(y_i, y_j) \geq 0$ for all $y_i, y_j \in M$ and $d(y_i, y_j) = 0$ iff $y_i = y_j$. The distance is symmetric, $d(y_i, y_j) = d(y_j, y_i)$, and fulfils the triangle inequality $d(y_i, y_j) \leq d(y_i, y_k) + d(y_k, y_j)$.*

Consider the point cloud in figure 2.4 (left), denoted $Y \in \mathcal{Y}$ where $\mathcal{Y}$ is a metric space. This point cloud is an example of the statistical circle: A first glance suggests that the point are sampled from a circle. To quantify this, we start with building a graph by connecting the points closer to a distance ϵ, (figure 2.4, middle). The resulting graph (figure 2.4, right) captures the central hole but also has a number of smaller loops; topologically, we cannot distinguish one loop from another.

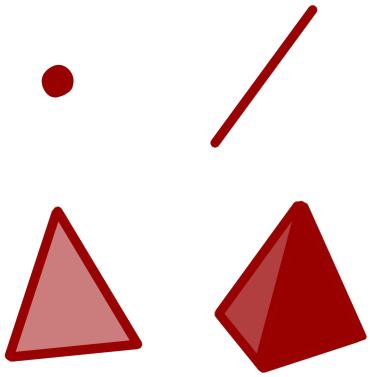


Figure 2.3: Simplices in zero (a point), one (a line segment), two (a triangle) and three dimensions (a solid pyramid).

This motivates the need for complices which helps distinguish smaller loops (noise) from holes that are a topological feature. Complices address this problem by filling gaps with simplices (figure 2.3). E.g. small loops in figure 2.4 (right) are filled with a two-simplex. This "filling in" can be done differently, but here we present the Vietoris-Rips complex. Other complices are not in the scope of this thesis; Ghrist [21] defines abstract simplicial complices, and Carlsson [22] discusses a number of these in detail.

Definition 2.2 (Vietoris-Rips Complex) *For any finite metric space $(\mathcal{Y}, d)$, and every scale parameter $\epsilon \geq 0$, let $\mathcal{V}_\epsilon(Y)$ denote the simplicial complex with vertex set equal to Y, and such that $\{y_0, \ldots, y_k\}$ spans a k-simplex iff $d(y_i, y_j) \leq \epsilon$ for all $0 \leq i < j \leq k$.*

VIETORIS-RIPS COMPLEX

In other words, we connect two points with the one-simplex when the two points are within the distance ϵ, and we connect three points with the two-simplex when the three points are pairwise within the distance ϵ and so forth. Given the scale parameter, the simplices compose the complex, $\mathcal{V}_\epsilon(Y)$.

Different scale parameters give rise to other topologies. A too-small ϵ fails to recognise the structure and a too-large ϵ leads to the trivial complex (i.e. the fully connected complex from which the centre hole cannot be recovered). We can obtain a quantitive characteristic of the

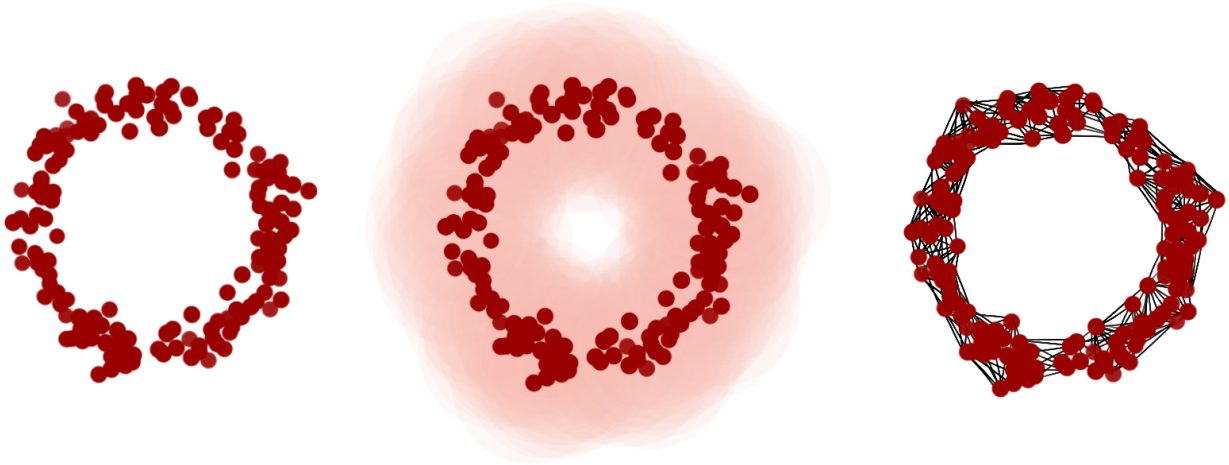


Figure 2.4: Left: A point cloud referred to as the statistical circle [20]. Middle: Points are connected if they are within a radius ϵ. Right: The graph G_ϵ.

homology of the point cloud using Betti numbers. Betti numbers allow us to map the topological structure to scalars.

Definition 2.3 (Betti Number) *Given the vector $H_k(X, F)$ over any field F, we write the* k*th Betti number as $\beta_k(X, F)$ with coefficients in F. Two homotopy equivalent spaces have equal Betti numbers β_k for all k.*

Informally, Betti numbers have an intuitive interpretation. The zeroth Betti number counts the number of connected components, the first Betti number the number of loops in the graph, and the second Betti number the number of closed spaces (or cavities). For example, a point has $\beta_0 = 1$ and $\beta_k = 0$ for $k \geq 1$, a circle has $\beta_0 = 1$, $\beta_1 = 1$ and $\beta_k = 0$ for $k \geq 2$, and a sphere has $\beta_0 = 1$, $\beta_1 = 0$ and $\beta_2 = 1$ and $\beta_k = 0$ for $k \geq 3$. The Betti numbers describe topological invariants and generalise to higher dimensions though the interpretation is less intuitive.

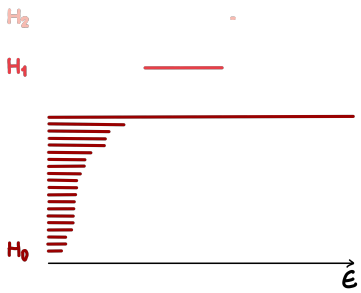


Figure 2.5: A barcode for the statistical circle as a function of the scale parameter ϵ.

For a given graph, we can compute the Betti numbers. This, however, depends on the scale parameter from which the graph was constructed. We are left with how to choose the scale parameter, and persistent homology answers that. The entire evolution of the Vietoris-Rips complex can be captured in a barcode[1]. Here, we detail the barcode only to the level of our purpose.

Persistence is the idea that topological features persist over an extensive range of ϵ, and the features that exist only in small ranges can be considered noise. The holes and voids that persist over a large range are considered topological features. Cohen-Steiner, Edelsbrunner, and Harer [25] proves that the barcode is robust to noise under

1. Edelsbrunner, Letscher, and Zomorodian [23] and Zomorodian and Carlsson [24] introduces barcodes rigorously

small perturbations to the point cloud. Persistence can be formally defined in terms of inclusion maps, but for our purpose, the barcode captures the persistence of Betti numbers as a function of the scale parameter. The example barcode in figure 2.5 captures the evolution of the homology as a function of the scale parameter as a vertical slice corresponds to the scale parameter. For small ϵ, the zeroth Betti number ($\beta_0 = \text{rank}(H_0)$) is large as each point is a connected component. As ϵ increases, β_0 decreases until all points are connected (i.e. the long bar). For Betti number of order 1, ($\beta_1 = \text{rank}(H_1)$), the barcode captures a hole in some range of ϵ and similarly for $\beta_2 = \text{rank}(H_1)$ but notice the length of the bars. In this case, we can interpret $\beta_1 = 1$ as a topological feature, but $\beta_2 = 1$ as noise due to the persistence theorem [25]. In the range where $\beta_1 = 1$, the point cloud has the homotopy type of a circle.

2.3 MEASURING SYMMETRY

We cannot detect the symmetry directly as this is a continuous object, but the observations constitute a noisy, discrete instantiation. Knowing the infinitesimal generator of the (continuous) symmetry informs a neighbourhood graph we can compare before and after dimensionality reduction for visualisation.

Specifically, we study rotational symmetry. For instance, we may connect rotated images in a graph if their rotation angles are similar to approximate the generating symmetry. In visualisation, we obtain a low-dimensional representation by coordinates $X = x_i \in \mathbb{R}^2$. We can then determine if a symmetry has been preserved by asking if the associated graph can be recovered from the low-dimensional representation.

Figure 2.6 shows an example of a visualisation model that fails to capture a rotation. The wooden toy is rotated, and the model learns a representation where images of only slight rotations are far apart. Ideally, all points should have been placed such that the loop had been closed and not self-intersecting. This work aims to develop a small experimental test to determine which visualisation models (if any) capture such rotations.

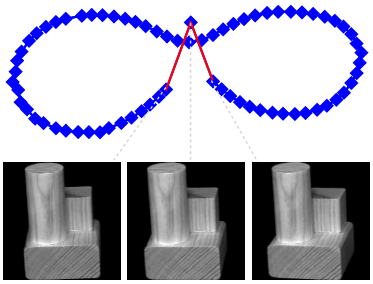


Figure 2.6: Example of symmetry breaking

We consider dimensionality reduction methods, and essentially we want the rotation in observation space, $\mathcal{Y}$, to be present in the low-dimensional representation, $\mathcal{X}$. We aim to describe that small changes in y should lead to small changes in x.

Definition 2.4 (Lipschitz Continuity) *[26] A function $f : \mathbb{R}^D \to \mathbb{R}^Q$ is Lipschitz continuous if there exists a constant K such that*

$$\left\|f(y_i) - f(y_j)\right\|_2 \leq K \left\|y_i - y_j\right\|_2 \tag{2.1}$$

for all $y_i, y_j \in \mathcal{Y}$

LIPSCHITZ CONTINUITY

We assume that the map f from the observation space to the latent representation exists[2]. Assuming no explicit representation but we define $f(y_i) = x_i$ and $f(y_j) = x_j$, which rephrase the Lipschitz continuity

$$\frac{\left\|x_i - x_j\right\|_2}{\left\|y_i - y_j\right\|_2} \leq K \tag{2.2}$$

2. Though we do not learn it explicitly

To our knowledge, there is no principled approach to determining the Lipschitz continuity constant, K, so we choose to use the median nearest neighbour distance as a proxy for the scale as it is a robust estimator [27]. For each set of neighbours, we can determine if the (Euclidean) distance agrees with the neighbourhood graph. We scale the neighbour distances with their median to compare different visualisation models.

$$\text{scale} = \underset{(x_i,x_j)\in G}{\text{median}}(\left\|x_i - x_j\right\|_2) \tag{2.3}$$

where G is the graph associated with the generating group. We, thus, measure

NEIGHBOUR DISTANCE
Scaled

$$B_{ij} = \frac{\left\|x_i - x_j\right\|_2}{\text{scale}} \tag{2.4}$$

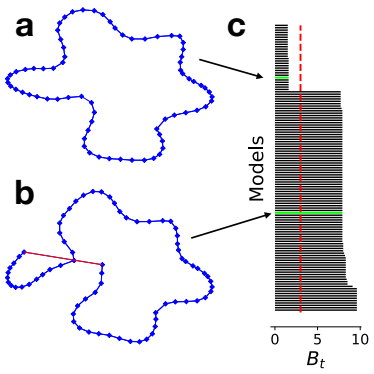


Figure 2.7: Examples of the imposed rotation being recovered (a) and not recovered (b) in the GPLVM. The barcode (c) allows for the inspection of numerous models simultaneously (stacked vertically), and $B_t = 3$ is indicated with a dashed red line.

Thresholding this measure $B_{ij} > B_t$ for any pair i, j indicate symmetry breaking: The symmetry has been broken if this is true and preserved if this is false. Note that this measure does not distinguish between one or multiple symmetry breaks.

We draw inspiration from topological data analysis and apply the tools somewhat differently to detect symmetry breaking. We already know what the graph should look like, so there is no need to detect it. Instead, we construct the graph (as we know it should be) and evaluate if two points *that are supposed to be* neighbours are, in fact, close enough to be neighbours.

The choice of B_t is somewhat arbitrary. To handle this, we draw inspiration from topological data analysis and persistent homology that explicitly avoids choosing a scale parameter. We know what Betti numbers we should expect for the rotational symmetry that we are imposing. In particular, it is b_1 that reveals whether the symmetry

has been broken. We need a $b_1 = 1$ for a preserved symmetry, and if $b_1 = 0$, then the symmetry has been broken. For this reason, we only focus on b_1. It turns out that the Betti number of order 1 is a practical indicator for symmetry breaking, and we estimate the Betti number of order one

$$b_1 = \begin{cases} 1 & B_{ij} > B_t \\ 0 & B_{ij} \leq B_t \end{cases}$$

where $b_1 = 1$ means that the symmetry is recovered and $b_1 = 0$ means that the symmetry is broken. We assume any cases of $b_1 > 1$ are noise.

The rotations give rise to a topology with Betti number of order one, $\beta_1 = 1$. This lets us search directly for topologies of this type rather than having to construct the Vietoris-Rips complex (section 2.2) and do persistent homology to identify it because the true graph is known from the data generation. The links are created based on the scale parameter, ϵ.

Instead, we know that a symmetry break is defined by $B_{\max} > B_t$, and this we can display visually, taking inspiration from the barcode. This means that the length of the bar (which is essentially a Betti number of the first kind) reduces the $B_{\max}$. In figure 2.7, the length of a bar represents this distance and stacking such bars yields the barcode [21]. Note that one bar corresponds to one model. This adapted barcode conveys the robustness of the conclusion qualitatively (break vs not breaking). This reduction may seem simplistic, but it allows for the visual comparison of numerous models simultaneously.

To quantify this, we chose a $B_t = 3$ and used that as a threshold in all experiments. This threshold is determined based on empirical evidence, but all results are accompanied by barcodes showing the sensitivity of our conclusions.

2.4 DIMENSIONALITY REDUCTIONS

This section briefly introduces four models for non-linear dimensionality reduction, representing different model classes. We will study symmetry breaking in detail in these models.

2.4.1 t-distributed Stochastic Neighbourhood Embedding

The method t-distributed Stochastic Neighbourhood Embedding (t-SNE) [28] is a popular visualisation technique, great for clustering. t-SNE is based on stochastic neighbourhood embedding (SNE) [29]. SNE matches an exponential distribution over pairwise distances in the observation space with an exponential distribution over pairwise distances in the latent space. These are matches by minimising the KL divergence between these two distributions. t-SNE is similar to SNE, the only difference being that t-SNE employs a t-distribution over pairwise distances in the latent space. A perplexity parameter controls the neighbourhood affinity in the exponential distribution.

2.4.2 TriMap

TriMap was proposed by Amid and Warmuth [30] as a randomised method which promises to *"preserve[...] the global accuracy of the data better than the other commonly used methods such as t-SNE, LargeVis, and UMAP"*. Triplets (three points i, j, k where i and j are closer) are weighted with their pairwise distance before obtaining the final triplet weight $\omega_{ijk} = \zeta_\gamma(\delta + \tilde{\omega}/\omega_{max})$ where $\zeta_\gamma(u) = \log(1 + \gamma u)$. The parameter γ emphasizes preservation of local structure (smaller γ) or global structure (larger γ). More details are given by Amid and Warmuth [30].

2.4.3 Kernel Principal Component Analysis

Kernel principal component analysis (kPCA) [31] extends PCA to a non-linear regime using the kernel trick. It deterministically maps a set of points into a higher-dimensional space using a kernel. In the following, we use the popular Gaussian kernel,

$$k(x_i, x_j) = \exp\left(-\frac{\left\|x_i - x_j\right\|_2^2}{\lambda}\right),$$

which is controlled by the scale parameter λ.

2.4.4 Gaussian Process Latent Variable Model

The Gaussian process latent variable model (GPLVM) is a probabilistic (but deterministic) generative model that uses a Gaussian process prior on the map from the latent space to the observation space [32].

A kernel decides the Gaussian process covariance. See section 3.3 for an elaboration of the GPLVM. As for kPCA, we use a Gaussian kernel,

$$k(x_i, x_j) = \theta \exp\left(-\frac{1}{2}\left\|x_i - x_j\right\|_2^2\right) + \sigma^2 \delta_{ij},$$

where x_i and x_j denote latent points and θ, and σ are the kernel and noise variances, respectively. The latent inputs are initialised using isomap [7] (or PCA [33, 34] in sensitivity analyses) and they are learnt by direct optimisation along with the kernel parameters.

2.5 EXPERIMENTAL RESULTS: SYMMETRY BREAKING IS PREVALENT

Figure 2.8: Preprocessed objects in COIL-20.

We study symmetries in COIL-20 [35]. The COIL-20 dataset is constructed with rotational symmetry in the images. The dataset contains 20 objects and 72 images of each object. Figure 2.8 shows five of the objects. Each object is placed on a turntable, and each image corresponds to a five-degree turntable rotation. The background in the images has been removed in the preprocessing.

For each of the four models, we identified the most significant parameter and studied symmetry breaking as a function of this. We sampled this parameter in an interval (determined by, e.g. SciKit Learn documentation for t-SNE or the TriMap paper [30]) and fixed other parameters at their default values. We emphasise that we did no parameter tuning. Figure 2.9 summarises our findings and shows the aggregated rates of symmetry breaking in each of the four models

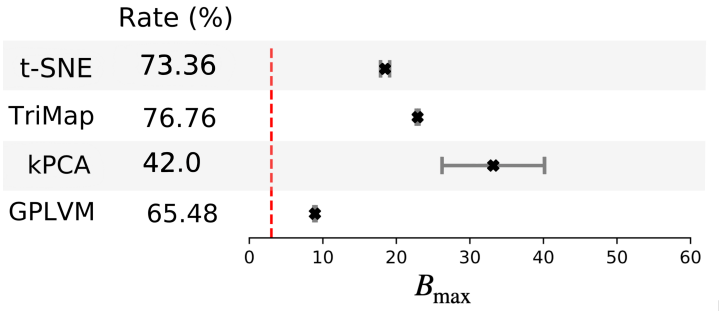


Figure 2.9: Prevalence of symmetry breaking in the four considered models using the threshold $B_t = 3$. The left-hand panel shows rates of symmetry breaking in COIL-20 The figure first appeared in Feldager, Hauberg, and Hansen [1].

in the two datasets, respectively, using the threshold $B_t = 3$. For instance, t-SNE broke the symmetry in 73% of the models trained on the twenty objects in COIL-20. The forest plot displays B_{mean} (the mean of $B_{\max}$) with its standard deviation. Interestingly, B_{mean} is well above our chosen threshold at $B_t = 3$ for all four models.

2.5.1 t-SNE

For each of the twenty objects in the COIL-20 dataset [36], we fit thirty t-SNE models using software from SciKit Learn [37]. We sample the perplexity parameter from $\mathcal{U}(5, 50)$ with the boundaries set according to SciKit Learn's documentation for t-SNE. In each of the thirty models (in each of the twenty objects), we measure $B_{\max}$ and report the average in semi-opaque blue (figure 2.10).

As perplexity increases, $B_{\max}$ decreases. This is unsurprising as the perplexity parameter controls the smoothness in t-SNE, and we expect a smaller rate of symmetry breaking the smoother model. We observe broken symmetry in 73% of t-SNE models. From the barcodes (figure B.1), it is clear that this number is sensitive to the choice of thresholds.

It seems that t-SNE reinforces noise in data. This means that if the distance between two points $d(x_i, x_{i+1})$ is larger than other pairwise distances, this distance is increased in the resulting model, which breaks the symmetry. We interpret this as t-SNE amplifying a gap in the underlying data manifold, which causes the manifold to be split into two or more pieces though the *true* manifold is whole. Like Amid and Warmuth [30], we refer to this as spurious clustering. Generally, random initialisation preserves symmetries better than PCA or isomap initialisation. Random initialisations require multiple restarts and picking the representation with the lowest KL divergence. The heavy tail, which makes t-SNE great for clustering, makes it less than ideal for non-clustering problems.

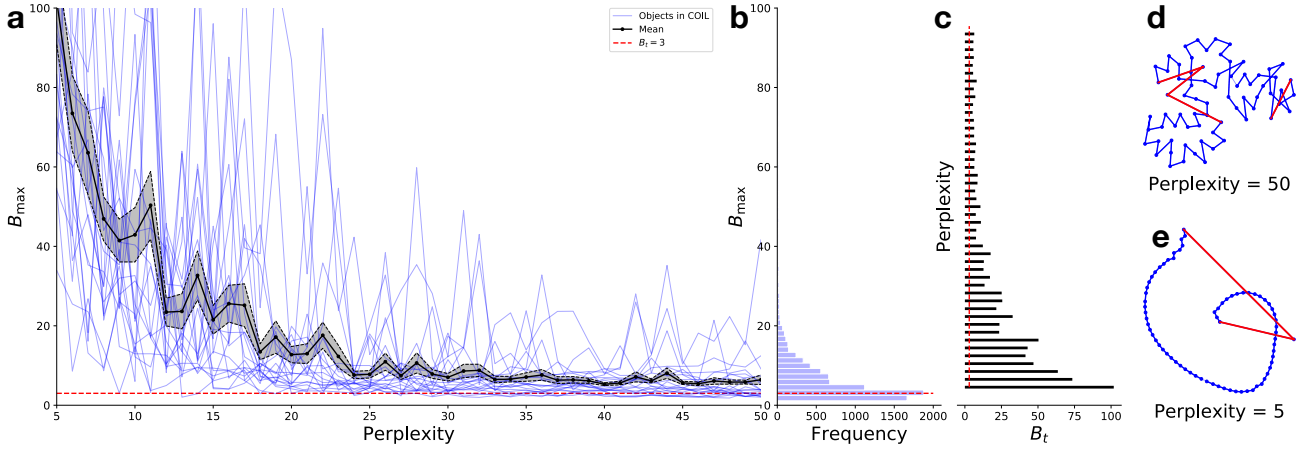


Figure 2.10: **a** shows the parameter space for t-SNE in the perplexity parameter. The blue lines each represent a mean of $B_{\max}$ from more than thirty models trained for each object in COIL-20, and the black line represents the mean with standard error of the mean (SEM). The dotted red line represents the threshold in **a**-**c**. **b** shows the distribution of $B_{\max}$ for all trained models. The barcode in **c** shows the mean $B_{\max}$ for each of the objects in COIL-20 as a function of γ. **d** and **e** show two learnt representations in models with perplexities 50 and 5, respectively. The figure first appeared in Feldager, Hauberg, and Hansen [1].

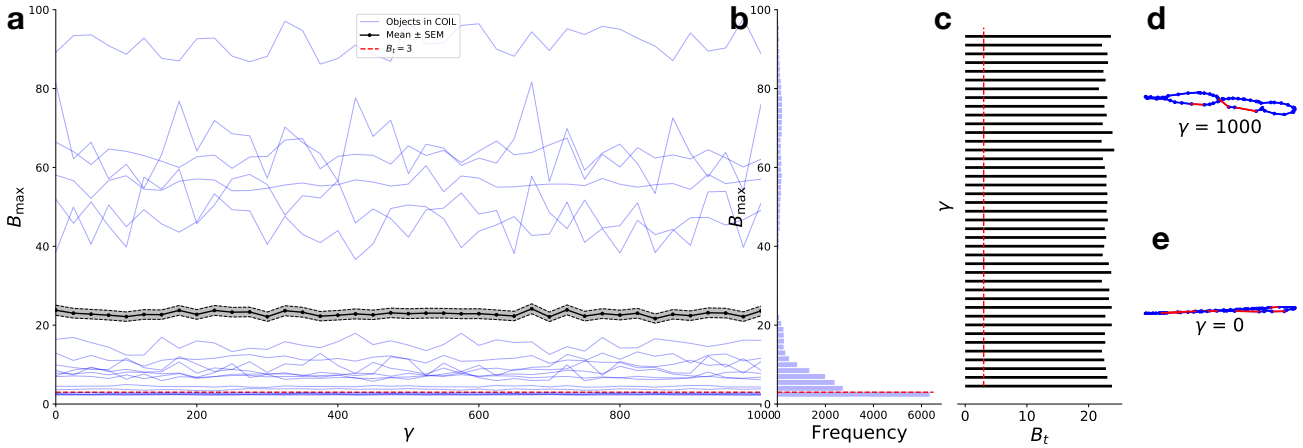


Figure 2.11: **a** shows the parameter space for TriMap in γ which places emphasis on local (small γ)or global structure (large γ). The blue lines each represent a mean of $B_{\max}$ from more than thirty models trained for each object in COIL-20, and the black line represents the mean with standard error of the mean (SEM). The dotted red line represents the threshold in **a**-**c**. **b** shows the distribution of $B_{\max}$ for all trained models. The barcode in **c** shows the mean $B_{\max}$ for each of the objects in COIL-20 as a function of γ. **d** and **e** show two learnt representations in models with $\gamma = 1000$ and $\gamma = 0$, respectively. The figure first appeared in Feldager, Hauberg, and Hansen [1].

2.5.2 TriMap

The spurious clustering in t-SNE led Amid and Warmuth [30] to develop TriMap to counter this. For this reason, we expected that TriMap would capture the symmetry to a large extent, but we found the opposite (figure 2.11). On average, TriMap breaks the symmetry in 77% of models across objects. For most objects, the conclusion is relatively

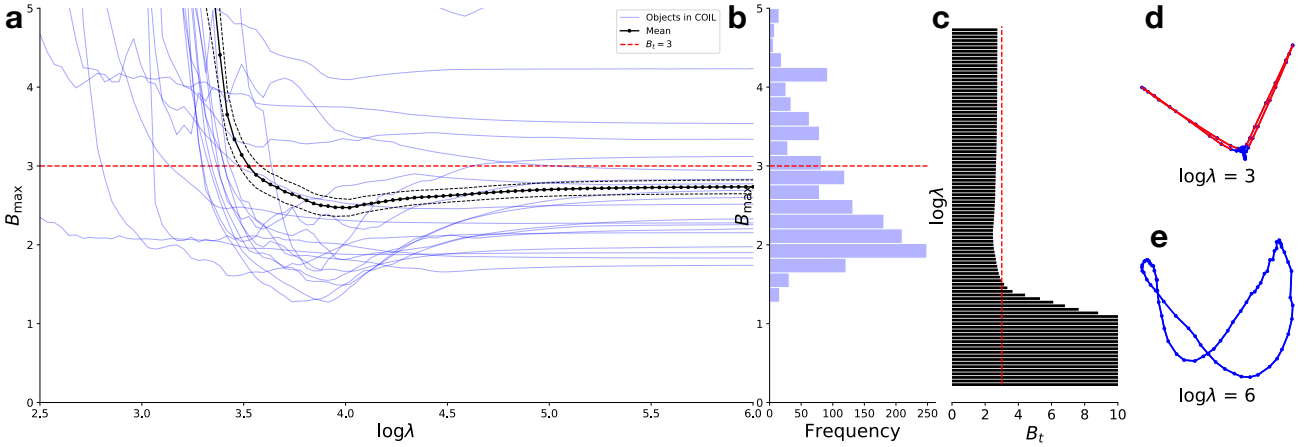


Figure 2.12: **a** shows the parameter space for kPCA in the scale parameter λ. The blue lines each represent a mean of $B_{\max}$ from more than thirty models trained for each object in COIL-20, and the black line represents the mean with standard error of the mean (SEM). The dotted red line represents the threshold in **a**-**c**. **b** shows the distribution of $B_{\max}$ for all trained models. The barcode in **c** shows the mean $B_{\max}$ for each of the objects in COIL-20 as a function of γ. **d** and **e** show two learnt representations in models with $\log \lambda = 3$ and $\log \lambda = 6$, respectively. The figure first appeared in Feldager, Hauberg, and Hansen [1].

robust (figure B.2). It is also somewhat surprising that the representations seem almost independent of γ, which emphasises local or global structure. TriMap is initialised with PCA, so we know that it is initialised without symmetry breaks, yet the overall rate of symmetry breaking is on par with t-SNE.

2.5.3 kPCA

The scale parameter for kPCA, λ, allows continuously examing symmetries from the linear regime (PCA, large λ) into the non-linear regime. Figure 2.12 summarises our findings for kPCA. In total, kPCA breaks the symmetry in 42% of models. In the non-linear regime (small values of λ), $B_{\max}$ seems to diverge. This is caused by clustering, meaning that B_{median} is small.

In the linear regime (large values of λ), we recover the PCA solution (figure 2.12e). Interestingly, some objects break the symmetry in the linear regime at $B_t = 3$. These objects correspond to the most symmetric objects (i.e. the object that looks the most like themselves after a π rotation), namely object 2 (wooden toy), object 16 (round bottle), object 16 (ceramic vase), object 18 (tea cup) and object 20 (round container). Four of these are rotationally symmetric in the plane of rotation, supporting our hypothesis that additional symmetry can induce symmetry breaking.

2.5.4 GPLVM

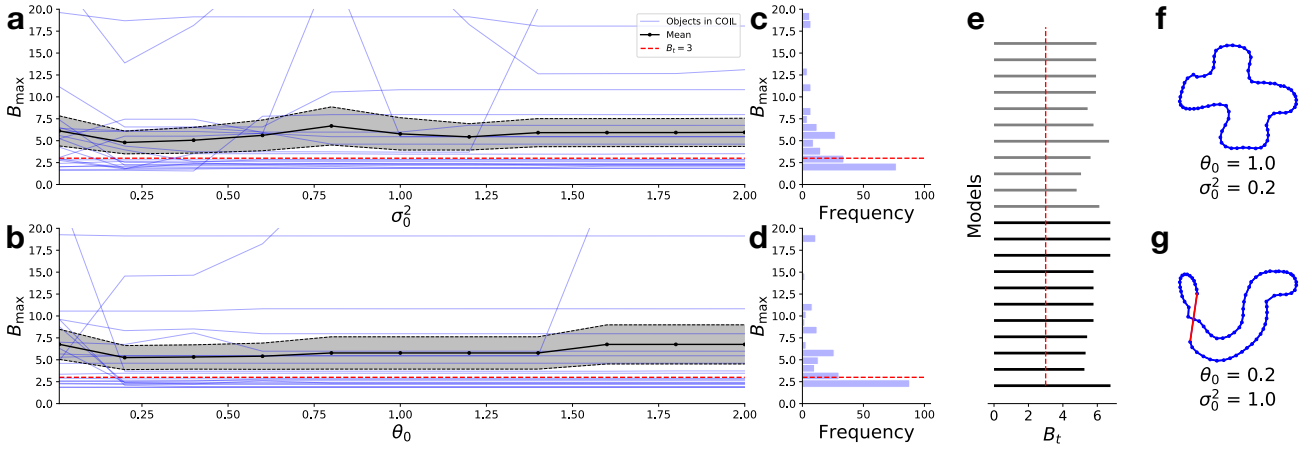


Figure 2.13: **a** and **b** show the parameter space for GPLVM of the initial values σ_0^2 (for fixed θ_0) and θ_0 (for fixed σ_0^2), respectively. The blue lines each represent a mean of $B_{\max}$ from more than thirty models trained for each object in COIL-20, and the black line represents the mean with standard error of the mean (SEM). The dotted red line represents the threshold in **a**-**e**. **c** and **d** show the distributions of $B_{\max}$ for all trained models in the parameter space corresponding. The barcode in **e** shows the mean $B_{\max}$ for each of the objects in COIL-20 in **a** (black) and **b** (grey). **f** and **g** show two learnt representations in models with $\theta_0 = 1.0, \sigma_0^2 = 0.2$ and $\theta_0 = 0.2, \sigma_0^2 = 1.0$, respectively. The figure first appeared in Feldager, Hauberg, and Hansen [1].

Figure 2.13 summarises our findings for GPLVM[3] in which symmetry breaking occurred in 65% of models. Like for TriMap, it seems that the mean $B_{\max}$ is approximately constant over a large range. We identified two important parameters for the GPLVM, but this figure shows the results of the varying parameter while keeping the other fixed. As this parameter space here is two-dimensional, we studied it in greater detail.

3. With isomap initalisations

Figures B.6 show a two-dimensional parameter space for the GPLVM and the barcode with the corresponding models. The models in this figure use isomap initialisation, and we do not observe any symmetry breaks for this object within the examined parameter ranges. Interestingly, we observe that regimes lead to seemingly identical latent representations, but the hyperparameters (green crosses) converge to the same values.

The GPLVM representation depends heavily on the initialisation [38]. Figures 2.14 and 2.15 compare the initialisations for one object. Each shows the parameter spaces in the initial parameters with learnt representations displayed at the location of the corresponding initialisation. Figure 2.14 shows no symmetry breaks in the parameters space for the initial parameters of the GPLVM.

The corresponding parameters space with PCA initialisation (figure 2.15) almost consistently breaks the symmetry. A small regime

exists for small θ_0 where the symmetry is not broken. Again, we generally observe that different regimes in initial hyperparameters lead to other latent representations, but the hyperparameters converge to the same values. We observe that if the initialisation breaks the symmetry, then the model cannot recover. On the other hand, if the initialisation preserves the symmetry, then the trained model can still break the symmetry. We display parameter space results for another object in B. The object in question also affects representation which figures B.5 and B.4 show. Like in kPCA, we find broken symmetries in the most symmetric objects.

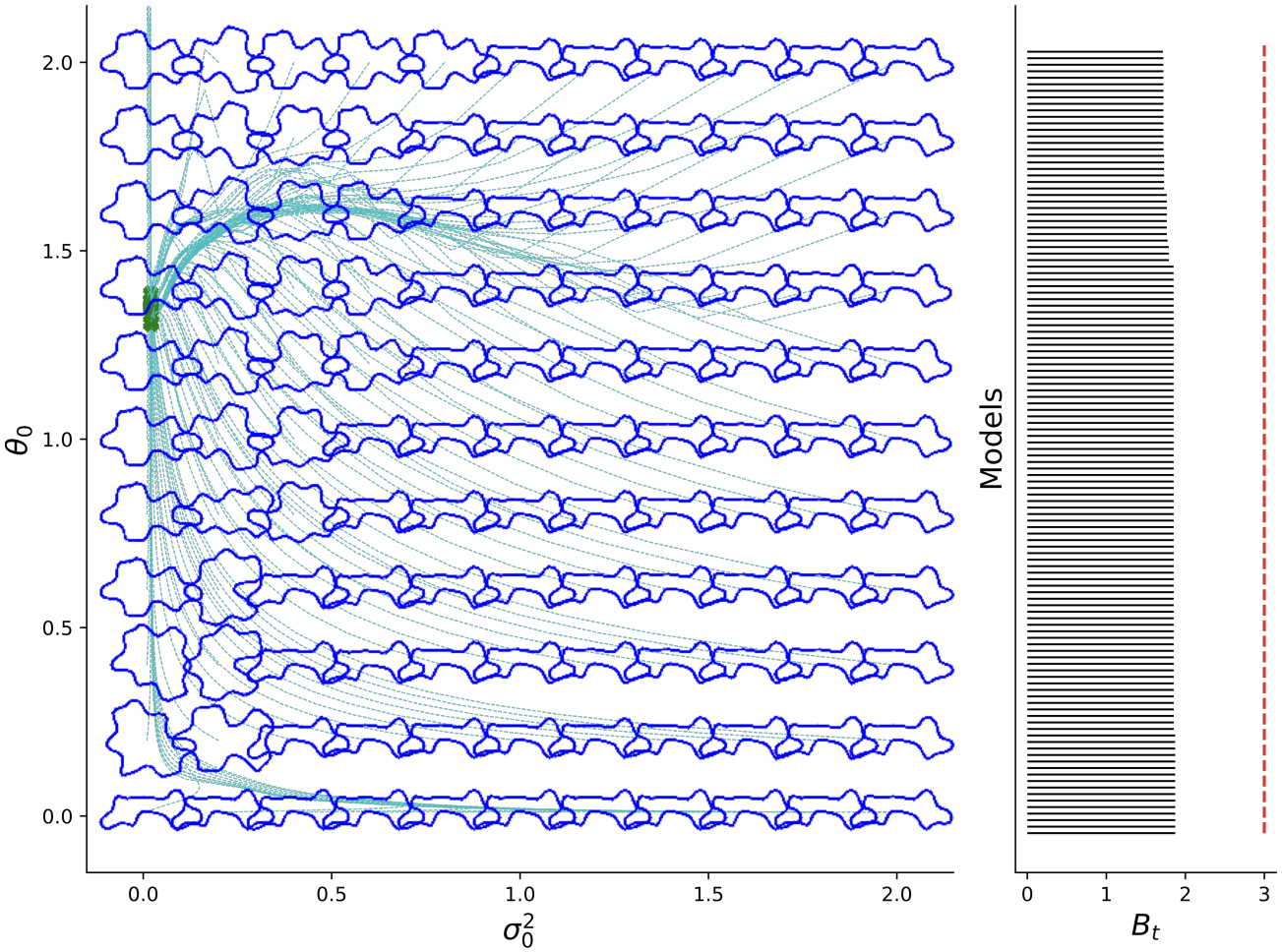


Figure 2.14: The left-hand side shows a phase space for GPLVMs in the initial values of hyperparameters θ_0 and σ_0^2 using isomap initialisation. Each location in the phase space corresponds to an initialisation of a model. As the model is trained, it follows the cyan trajectory through phase space until the green cross, which indicates the model's state at the end of training. The learnt latent space is shown at the initialisation location to emphasise what representation a given initialisation results in. The right-hand side shows the barcode for these models, where each bar corresponds to a model on the axis of thresholds. The red dotted line represented our chosen threshold for symmetry breaking.

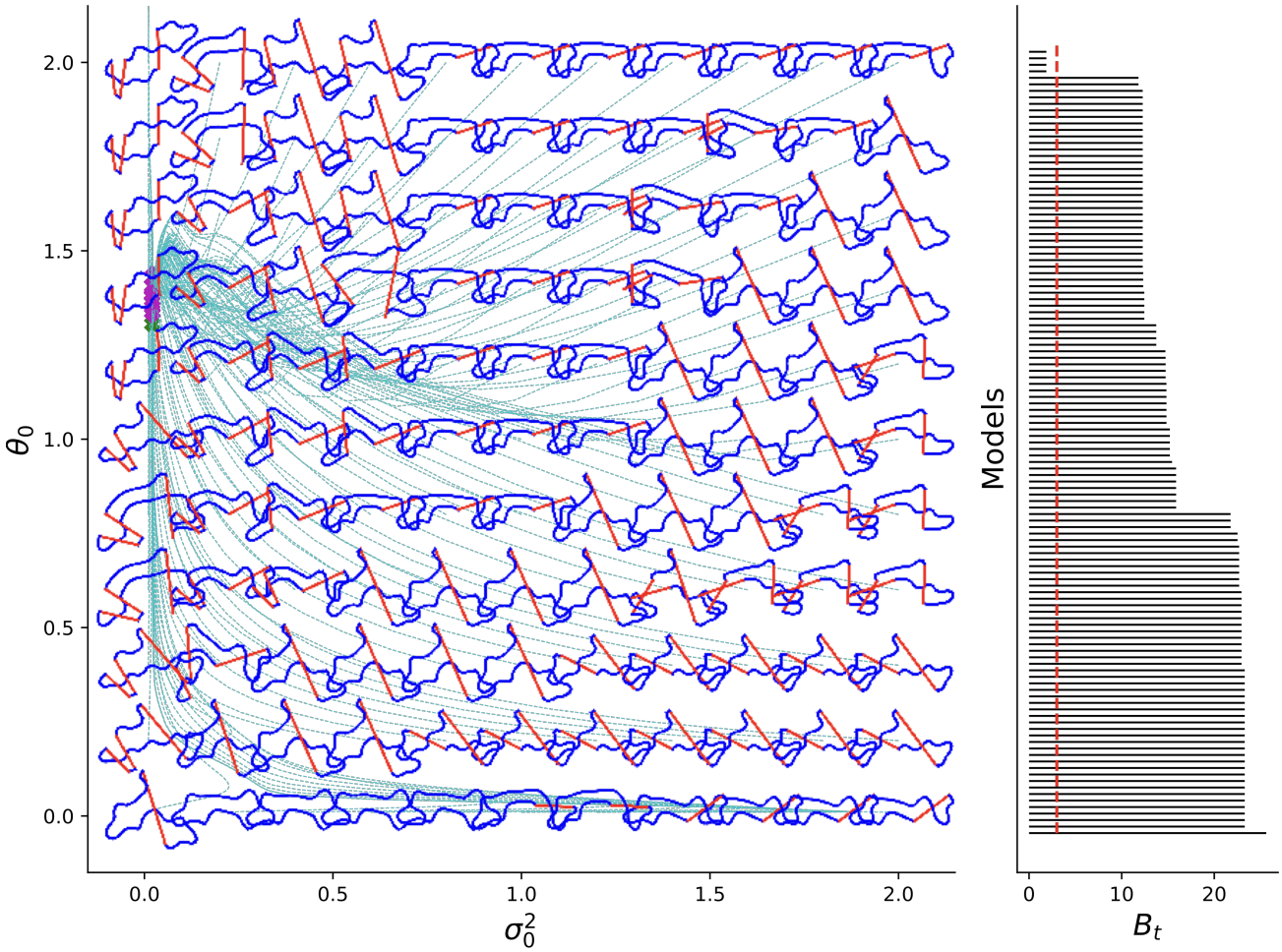


Figure 2.15: The left-hand side shows a phase space for GPLVMs in the initial values of hyperparameters θ_0 and σ_0^2 using PCA initialisation. Each location in the phase space corresponds to an initialisation of a model. As the model is trained, it follows the cyan trajectory through phase space until the green cross, which indicates the model's state at the end of training. The learnt latent space is shown at the initialisation location to emphasise what representation a given initialisation results in. The right-hand side shows the barcode for these models, where each bar corresponds to a model on the axis of thresholds. The red dotted line represented our chosen threshold for symmetry breaking.

2.6 RELATED WORK

Data visualisation is a particular case of dimensional reduction to two (or three) dimensions, making it crucial for working with high-dimensional data. Data visualisation is also often a vital step in the machine learning pipeline: From forming hypotheses [14] to communicating results [15, 16]. Our preliminary findings (data not shown) suggest that symmetry breaking is more prevalent in two dimensions compared to high dimensions.

Spurious clustering is well-known in t-SNE [39]. It was developed to capture both local and global structure–and as a solution to the

crowding problem in SNE [29] (the crowding problem covers visualisations without clear separation between clusters). t-SNE employs a heavy-tailed distribution to solve the crowding problem and cluster. Linderman and Steinerberger [40] proved this and showed that simple manifolds ripped apart clusters in line with our findings. It was the spurious clustering in t-SNE that motivated the development of TriMap [30]. Supposedly, TriMap should be able to counter the over-fitting, but our findings suggest otherwise.

Lawrence and Quiñonero-Candela [41] also considered spurious clustering in GPLVM, using an amortised model (i.e. encoding the latent means with a multi-layer perceptron). They explain that the GPLVM conserves dissimilarities and the neural network conserves similarities. More recent work supports this view [42].

More generally, Higgins et al. [43] argues that learning faithful representation (i.e. representations that reflect the underlying data structures) is an alternative to hard-coding inductive bias into models (e.g. CNNs are translation invariant).

2.7 DISCUSSION

Our experiments show that symmetry breaking is alarmingly prevalent in four visualisation models in a simple setting, systematically training more than 85.000 models. Generally, it is possible to manually tweak the parameter to learn a representation that preserves the symmetry. On the other hand, this is not feasible when the underlying symmetry is unknown. When the true manifold is unknown, symmetry breaking is hard to detect. Multi-starting the visualisation model and selecting the best model is a potential solution. However, our findings suggest that multi-starting five or even ten times might not recover the manifold due to the high prevalence of symmetry breaking.

Symmetry breaking may be particularly prevalent in visualisation. In our experiments, we have used a circular graph, a planar graph, i.e. a graph that can be embedded in $\mathbb{R}^2$. While only planar graphs can be embedded in $\mathbb{R}^2$, all graphs can be embedded in $\mathbb{R}^3$ [44]. This suggests visualisation may be more likely to break symmetries by forcing a two-dimensional view in a general setting.

Dimensionality reduction is a compression and a loss of information, and intuitively, it makes sense that the $\mathcal{Y}$ is a larger space than $\mathcal{X}$. We expect different observations to be mapped to the same small area in $\mathcal{X}$ but not vice versa. Two close observations getting mapping to two

distant points in the latent space is a breaking of the symmetry and, by extension, a violation of the Lipschitz continuity means (figure 2.16).

Lipschitz continuity, the basis of our symmetry measure, is also the foundation for individual fairness [45]. Individual fairness states that *similar individuals should be treated similarly*. A model that breaks symmetry is a model that violates Lipschitz continuity, and it is unclear if individual fairness can be ensured in downstream tasks.

As we argue in our paper [1], the driver for this study is the pursuit of *faithful representations*. These representations reflect the underlying physics of the generating process and allow for explainable models. This is also the objective of meaningful representation [46]. Symmetries (and other inductive biases) are custom-built into models [47, 48], and Higgins et al. [43] suggest learning a faithful representation as an alternative to hardwiring inductive biases.

Recovering topology is a crucial first step on the path towards a model for geometry. When the manifold is ripped apart, the true manifold cannot be recovered. While this is not possible for any of the four models investigated here, the GPLVM have a few benefits: Its likelihood correlates with how broken the symmetry is, and the transition between broken and preserved symmetry in the barcode seemed sharper. We choose to look into the GPLVM is further motivated by Jørgensen and Hauberg [12] and Hauberg [49] who argue that uncertainty can play the role of topology. We will get back to this in chapter 6 after the next part that introduces the GPLVM in more detail.

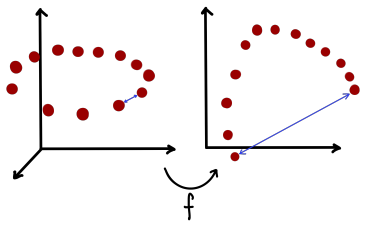


Figure 2.16: Two close observations are mapped distantly in the latent space, i.e. the blue distance is mapped from small to large. This can be considered a violation of the Lipschitz continuity.

INTRODUCTION TO LEARNING

3

> *If you just watch a teenager, you see a lot of uncertainty.*
>
> —Jamie Lee Curtis

In chapter 1, we discussed our undertaking of finding a probabilistic latent variable model that is suitable for learning geometries. This and the following two chapters encompass our search for such a model. After some housekeeping (section 3), we explore probabilistic principal component analysis (PPCA, section 3.1). PPCA is linear, and our ideal model should be able to capture non-linear manifolds. We look at an alternative to PPCA called dual PPCA, which is also linear, but non-linearities can readily be introduced (section 3.2). This leads to the Gaussian process latent variable model (GPLVM, section 3.3)–a non-linear, probabilistic latent variable model–and its sparse, variational extension, the Bayesian GPLVM (section 3.4). This will be a concise chapter that introduces aspects of GPLVMs as they are relevant to this thesis, and in particular relevant to chapter 4[1].

1. For an in-depth introduction to Gaussian processes, we refer to Rasmussen and Williams [8] and for great introductions to GPLVM, we suggest tutorials from the Gaussian Process Summer School.

BAYES THEOREM

Bayes' theorem is the fundamental principle for Bayesian learning. If we denote the parameters of a model ω and the observations Y, then Bayes' theorem [50, 51] states

$$\underbrace{p(\omega|Y)}_{\text{posterior}} = \frac{\overbrace{p(Y|\omega)}^{\text{likelihood}}\overbrace{p(\omega)}^{\text{prior}}}{\underbrace{p(Y)}_{\text{marginal likelihood}}}. \tag{3.1}$$

We refer to these distributions using specific terms: $p(\omega|Y)$ is the posterior, $p(Y|\omega)$ is the likelihood, $p(\omega)$ is the prior, and

$$p(Y) = \int p(Y|\omega)p(\omega)\mathrm{d}\omega,$$

is the marginal likelihood which is often also referred to as the evidence. In the Bayesian setting, learning is the same as inference: We aim to estimate the posterior using the marginal likelihood as this generalises well [52].

In general, fully Bayesian inference is not analytically tractable as the marginal likelihood requires integration over non-linearities, but

several alternative approaches exist; maximum likelihood, maximum aposteriori estimation, and variational Bayesian inference [53, 54]. Variational Bayesian inference estimates a variational distribution $q(\omega)$ that is an approximation to the true posterior distribution $p(\omega|Y)$. We get back to variational Bayesian inference in section 3.4 (when introducing the Bayesian GPLVM) and in chapter 4 that develops stochastic active sets for Gaussian process decoders.

NOTATION

We denote a set of observations $Y = [y_1, \ldots, y_N]^\top$ and $Y = y_{1:N}$ with $Y \in \mathbb{R}^{N \times D}$. We refer to one D-dimensional observation as y_n with $y_n \in \mathbb{R}^D$ and a subset of the observations as $y_{i:j}$ where i, j indexes observations (not dimensions) and $1 \leq i < j \leq N$. Sometimes we have to index dimensions and use $y_{n,d}$ or $y_{:,d}$ where the colon indicates that we also index dimensions. This will also be explicit in the text. y^* is a test point of dimensionality D so $y^* \in \mathbb{R}^D$. Each observation y_n has a corresponding input point x_n, and all input points are collected in the matrix $X = [x_1, \ldots, x_N]$ and we also use the notation $X = x_{1:N}$ for all the input variables. The dimensionality of the input space is Q such that $X \in \mathbb{R}^{N \times Q}$ and $x_n \in \mathbb{R}^Q$, and we refer to one input (or latent variable) as x_n with $x_n \in \mathbb{R}^Q$ and a subset of the observations as $x_{i:j}$ where i, j indexes observations (not dimensions) and $1 \leq i < j \leq N$. In the following, we will assume that $Q \ll D$, i.e., the observation space, is of much higher dimensionality than the latent space.

3.1 PROBABILISTIC PRINCIPAL COMPONENT ANALYSIS

We consider PPCA as a candidate for manifold learning. Here, we re-iterate essential aspects of PPCA needed to arrive at the GPLVM: 1) We introduce PPCA, where the weights are optimised, and the latent variables are marginalised. 2) We consider dual PPCA where the latent variables are optimised, and the weights are marginalised, and 3) we arrive at the GPLVM by introducing a non-linear kernel [32].

Probabilistic PCA was developed independently by Tipping and Bishop [55] and Roweis [56][2] and lays the ground for a Bayesian approach to PCA [51] so unlike PCA [33, 34], PPCA is a generative model. PPCA assumes that one observation from a set of D-dimensional, centred observations $Y = [y_1, \ldots, y_N]^\top$ is generated using a linear function of

2. More details on probabilistic PCA can be found there. Lawrence [32] and Bishop [51] also discuss probabilistic PCA.

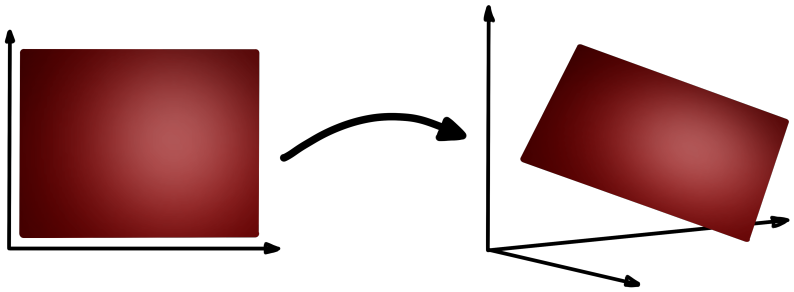


Figure 3.1: Illustration of dual probabilistic PCA as a two-dimensional, probabilistic plane that is embedded in a three-dimension space.

the Q-dimensional latent variables $X = [x_1, \ldots, x_N]^\top$ in the following way

$$y_n = Wx_n + \epsilon_n \tag{3.2}$$

where $Q \ll D$ and the random noise $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$ is Gaussian and W is a linear map, $W \in \mathbb{R}^{Q \times D}$.

Following the generative model (equation 3.2), we can write the likelihood of Y as

$$p(Y|X, W, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}(y_n|Wx_n, \sigma^2\mathbb{I}). \tag{3.3}$$

By assuming a unit Gaussian prior on the latent variables,

$$p(X) = \prod_{q=1}^{Q} \mathcal{N}(x_n|0, \mathbb{I}), \tag{3.4}$$

the marginal likelihood can be found analytically

$$p(y_n|W, \sigma^2) = \int p(y_n|x_n, W, \sigma^2)p(x)\mathrm{d}x = \mathcal{N}(y_n|0, WW^T + \sigma^2\mathbb{I}). \tag{3.5}$$

Using Gaussian conditional distributions, we can find the posterior of the latent $p(X|Y)$, as

$$p(X|Y) = \mathcal{N}(M^{-1}W^TY, \sigma^2M^{-1}), \tag{3.6}$$

with $M = W^\top W + \sigma^2\mathbb{I}$. Tipping and Bishop [55] derives analytical maximum likelihood solutions for W and σ^2.

PPCA extends PCA to the Bayesian domain by marginalising the latent variables X and optimising the weights, W. The learnt model maps from a latent space into a principal subspace; see figure 3.1.

3.2 DUAL PROBABILISTIC PRINCIPAL COMPONENT ANALYSIS

Dual PPCA is a linear, probabilistic model similar to PPCA and yields the same solution as PPCA. Rather than marginalising the latent variables and optimising the weights (as PPCA does), dual PPCA marginalises the weights and optimises the latent variables. The prior distribution of the weights is

$$p(W) = \prod_{d=1}^{D} \mathcal{N}(w_d|\, 0, \sigma^2 \mathbb{I}_Q), \tag{3.7}$$

where w_d refers to the dth column of W and W factorises over output dimensions. Marginalising the weights gives

$$p(Y|X, \sigma^2) = \prod_{d=1}^{D} p(y_{:,d}|X, \sigma^2) = \prod_{d=1}^{D} \mathcal{N}(y_{:,d}|0, XX^\top + \sigma^2 \mathbb{I}), \tag{3.8}$$

which yields the log marginal likelihood of Y,

$$L = \log p(Y|X, \sigma^2) = -\frac{DN}{2} \log 2\pi - \frac{D}{2} \log \det(K) - \frac{1}{2} K^{-1} YY^\top, \tag{3.9}$$

where $K = XX^\top + \sigma^2 \mathbb{I}_Q$. Lawrence [32] provides the maximum likelihood solution for the latent variables.

Lawrence [32] argues that marginalising both the weights and the latent variables would be ideal, but this is not tractable. He continues and says that it would be natural to marginalise the latent variables because $X \in \mathbb{R}^{N \times Q}$ and $W \in \mathbb{R}^{D \times Q}$ and often $N \ll D$ but the learnt principle subspace is equivalent in PPCA and dual PPCA.

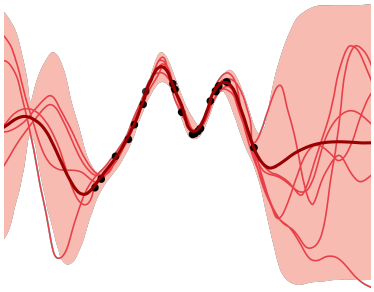


Figure 3.2: Illustration of a Gaussian process posterior. The black dots are the observations. The dark, red line is the mean of the Gaussian process, and the thin, lighter lines are samples. The shaded light area represents the covariance of the Gaussian process.

3.3 GAUSSIAN PROCESS LATENT VARIABLE MODEL

PPCA and dual PPCA assume linear, independent, and identically distributed output dimensions. The GPLVM challenges the assumption of linearity by replacing the linear map with a non-linear function and placing a Gaussian process prior on this map. Here, we introduce Gaussian processes kernels and GPLVM.

Gaussian process models are a flexible class of models where the marginal likelihood can be computed in closed form [8].

GAUSSIAN PROCESS

Definition 3.1 (Gaussian Process) *A Gaussian process is an infinite collection of random variables, any finite subset of which is jointly Gaussian distributed.*

This means that any two observations follow a multivariate Gaussian distribution jointly. This is nice because the conditional and marginal distributions are Gaussian distributions and can be found explicitly [51]. This provides posteriors in closed form.

Intuitively, we can view a Gaussian process as a distribution over functions, f, that maps from an input space to an output space (figure 3.3). The Gaussian process is determined by a mean $m(x)$ and a covariance $k(x, x')$

$$\mu(x) = \mathbb{E}[f(x)] \tag{3.10}$$
$$k(x, x') = \mathbb{E}[(f(x) - \mu(x))(f(x') - \mu(x'))], \tag{3.11}$$

and we write the Gaussian process

$$f(x) \sim \mathcal{GP}(\mu(x), k(x, x')). \tag{3.12}$$

When we model with Gaussian processes, we usually make (at least) two assumptions: We assume that the Gaussian process prior has a zero-mean, i.e. $\mu(x) \equiv 0$, and we assume that a kernel $k(x, x')$ determines the functional form of the covariance.

3.3.1 Kernels

It is standard to use kernels to estimate the covariance for Gaussian processes [8], and we follow this practice. Bishop [51] and Duvenaud [57] gives further details on kernels.

We can use kernels as covariance functions because they yield symmetric and positive, semi-definite matrices with real entries. Kernels are inner products defined in feature space such that

$$k(x, x') = \phi(x)^T \phi(x'), \tag{3.13}$$

and this allows us to compute pairwise relations in a higher dimensional space. The kernel computes this relation as an inner product in a high or infinite dimensional space. It returns the pairwise similarities directly without explicitly representing the observations in a higher dimensional space.

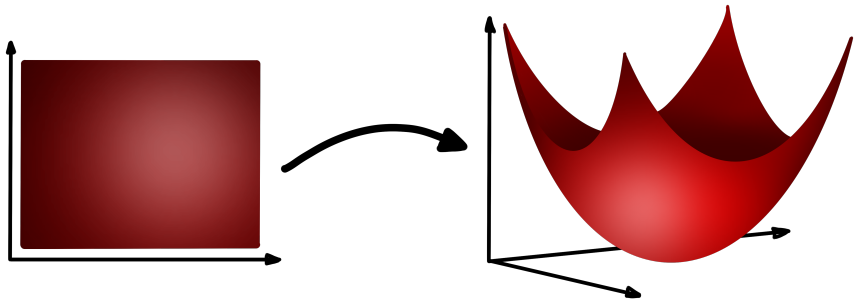


Figure 3.3: The Gaussian process decoder maps a lower dimensional latent space to a higher dimensional observation space from a non-linear function f, which is modelled as a Gaussian process.

In practice, kernels can be combined–added or multiplied–and still be valid kernels. We can optimise the kernel hyperparameters. The exponentiated quadratic (EQ) kernel[3] is a stationary kernel and one of the most common. The EQ kernel is given by

$$k(x, x') = \theta \exp\left(-\frac{||x - x'||^2}{2l}\right), \tag{3.14}$$

3. This is also called a radial basis function kernel, a Gaussian kernel, a squared exponential kernel.

where we refer to $\theta \in \mathbb{R}$ as the kernel variance and to $l \in \mathbb{R}$ as length scale. Alternatively, we can assume different length scales in different dimensions, leading to automatic relevance determination (ARD) for the EQ kernel.

$$k(x, x') = \theta \exp\left(-\frac{1}{2}\sum_{q=1}^{Q}\frac{||x_q - x'_q||^2}{l_q^2}\right). \tag{3.15}$$

This is a more flexible kernel that allows us to learn the dimensionality of the latent space. We will use the shorthand notation $k(x, x') = K_{xx'}$. In this thesis, we use the EQ kernel, a common covariance function choice. Further, the EQ kernel is C^∞ and doing geometry (which we get to later) requires the kernel to be at least C^2. We will discuss this choice in chapter 7.

Figure 3.4: Graphical model for GPLVM. Y is observed and the map F and the input X are latent.

We distinguish between the latent variables and the GP parameters. We refer to the following parameters as the GP parameters: The likelihood noise σ^2, the kernel length scale l and the kernel variance θ (assuming an EQ kernel) In the following, we leave out the GP hyperparameters of the conditioning for clarity.

The GPLVM can be viewed as non-linear PPCA and the graphical model is shown in figure 3.4. Lawrence [32] shows that using a linear kernel for the GPLVM yields the same solution as PPCA, and using a non-linear kernel allows for non-linear dimensionality reduction. The GPLVM (in its original formulation) inherits PPCA's assumptions of independent and identically distributed output dimensions.

We can also view the GPLVM as a GP regression problem [8] where the dimensionality of the input is not the same as the dimensionality

of the output, and rather than having observed the input, we learn the input.

Similarly to PPCA, the GPLVM is a generative model that maps a low-dimensional latent input, x_n, to an observation, y_n,

$$y_n = f(x_n) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2)$$

where the first term is the model and the second term is a noise corruption which is also assumed to be normally distributed. The noise is distributed as $\epsilon_n \sim \mathcal{N}(0, \sigma_n^2)$ with $\epsilon \in \mathbb{R}$. This means that a latent point x_n is mapped to a corresponding observation y_n with a map f and corrupted by some noise ϵ. We assume independent and identically distributed noise which leads to diagonal noise.

We place a Gaussian process prior on the mapping, $f_d : \mathbb{R}^Q \to \mathbb{R}^D$. The map refers to the Gaussian process mapping to the dth output dimension such that $F = [f_1(X), ..., f_D(X)]$.

$$p(f_d|x) = \mathcal{GP}(f_d|0, k(x, x'))$$

We assume a zero-mean prior but note that the posterior of the Gaussian process has a non-zero mean $m(x)$. The kernel, $k(x, x')$, captures our inductive bias on the data and gives the covariance between any two points. This means that the output dimensions are independent, i.e. we assume no covariance between output dimensions. The inputs are unobserved and therefore assumed to be latent. This scenario leads to a class of models called GPLVMs, unsupervised learning with Gaussian processes (section 3.3). We assume a unit Gaussian prior on the input

$$p(X) = \prod_{n=1}^{N} \mathcal{N}(x_n|0, \mathbb{I}_N)$$

where the factorisation over latent variables is due the assumption of independent and identically distributed noise. This assumption also factorises the likelihood, which is also Gaussian.

$$p(Y|F, X) = \prod_{d=1}^{D} p(y_d|F, X), \tag{3.16}$$

where y_d is a column in Y. The term $p(Y|f)$ is the likelihood which we assume to be Gaussian, $p(Y|F) = \mathcal{N}(Y|F, \sigma_n^2\mathbb{I}_N)$. There are numerous choices of likelihood, but other types of likelihoods are not in the scope of this thesis.

We want to predict the function value at a test point x_*. We name the predicted function value at the test point $f_* = f(x_*)$. We can find it by considering the joint distribution,

$$\begin{bmatrix} Y \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} k(X,X) + \sigma_n^2 \mathbb{I} & k(X, x_*) \\ k(x_*, X) & k(x_*, x_*) \end{bmatrix}\right). \tag{3.17}$$

The Gaussian conditionals lead to the predictive distribution in closed-form

$$f_* = k(x_*, X)[k(X,X) + \sigma_n^2 \mathbb{I}]^{-1} Y \tag{3.18}$$
$$\text{cov}(f_*) = K(x_*, x_*) - K(x_*, X)[k(X,X) + \sigma_n^2 \mathbb{I}]^{-1} k(X, x_*), \tag{3.19}$$

This assumes noisy observations; Rasmussen and Williams [8] gives more details.

3.3.2 Learning GPLVM

Learning a GPLVM amounts to estimating the posterior of mapping and learning the latent variables,

$$p(F|Y,X) = \frac{p(Y|F)p(F|X)}{p(Y)} = \frac{p(Y|F)p(F|X)}{\int p(Y|F)p(F|X)\mathrm{d}F}. \tag{3.20}$$

In general, solutions are not available in closed form, so we optimise both the inputs for the model and kernel hyperparameters. To obtain an ML estimate, we use the log-likelihood as the objective function,

$$\log p(Y|X) = -\frac{DN}{2}\log 2\pi - \frac{D}{2}\log |K_{NN}| - \frac{1}{2}\text{tr}(K_{NN}^{-1} Y Y^\top). \tag{3.21}$$

This can be optimised with respect to both latent variables and hyperparameters, but the ML approach is prone to overfitting, which Lawrence [32] pointed out in his seminal paper. He suggested training with MAP using the prior on the latent variables.

3.3.3 Challenges

The GPLVM is an elegant model which has proved itself in numerous applications, e.g. biological data [58] and missing data [42]. It is a flexible model that provides a generative, non-linear mapping,

which proves useful from a manifold learning perspective: Learning a manifold of fonts [59], path planning in robotics [60], human pose estimation [10] and in neural data [61]. These characteristics make the GPLVM a promising candidate for manifold learning, where we need a generative model to be able to do smooth interpolations.

While the GPLVM is elegant in concept, in practice, it has two main challenges: It tends to overfit [32], its complexity in time scales as $O(N^3)$ and its complexity in memory scales as $O(N^2)$ [62]. In his seminal paper, Lawrence [32] acknowledged these issues and suggested training with MAP using the prior on the latent variables rather than with MLE. MAP also has drawbacks [63, section 3.2.2]. This motivates the variational learning of inducing points–the Bayesian GPLVM (section 3.4)–which addresses both of these issues. Equation 3.21 shows the main challenge in scaling up decoder GPs: Inverting K_{NN} is expensive. The time requirement scales as $O(N^3)$ and the memory requirement scales as $O(N^2)$ [62]. Lawrence [32] used sparse kernel methods [8, 64] to scale the GPLVM and since numerous works have scaled GPs in various ways. Much of the effort for scaling GPLVM is downstream from the supervised setting, so it is not clear that these methods also work for the GPLVM. These are not in the scope of this thesis. Instead, we focus on active sets, which we revisit with a stochastic perspective in the next chapter, chapter 4.

Smola and Bartlett [64] introduced active sets to the GP community. An active set is an informative subset of the observations used for inference. This strategy reduces the time complexity to $O(N^2M)$ where M is the number of active points. The challenge of finding an *optimal* active set, which represents the entire data, is a combinatorially complex problem. Quiñonero-Candela and Rasmussen [65] summarises the discussion at the time.

> *Traditionally, sparse models have very often been built upon a carefully chosen subset of the training inputs [...] In sparse Gaussian processes it has also been suggested to select the inducing inputs $\mathbf{X}_u$ from among the training inputs. Since this involves a prohibitive combinatorial optimisation, greedy optimisation approaches have been suggested [...]. Recently, Snelson and Ghahramani [66] have proposed to relax the constraint that the inducing variables must be a subset of training/test cases, turning the discrete selection problem into one of continuous optimisation.*
>
> —Quiñonero-Candela & Rasmussen, 2006

The significant breakthrough by Snelson and Ghahramani [66] was introducing inducing variables as an additional set of inputs to the

model. Rather than selecting an optimal set of inducing variables, they proposed learning them.

3.4 BAYESIAN GAUSSIAN PROCESS LATENT VARIABLE MODELS

This section describes the most relevant aspects of the Bayesian GPLVM [67]. We build a Bayesian GPLVM as a suitable baseline for the Bayesian SAS decoder, which we cover in the next chapter. The Bayesian GPLVM is an umbrella term that covers a set of models based on the GPLVM, as there have been exciting extensions to the model initially proposed as the Bayesian GPLVM. The Bayesian GPLVM and some extensions are detailed by Damianou [63].

Snelson and Ghahramani [66] proposed inducing variables as an additional set of inputs to the model. Rather than selecting an optimal set of inducing variables, they proposed learning them and led the way for variational learning of inducing variables which Titsias [62] introduced in the supervised setting. This became (and remains) the foundational principle for scaling GPs. The year after, Titsias and Lawrence [67] extended this approach to GPLVM in which the inducing points are learnt using variational inference.

Bayesian GPLVM is built on sparse GPs which rely on inducing variables. Sparse GPs assume that m inducing variables f_u (additional variables evaluated at additional input, x_u, such that $u = f(x_u)$) represent the data well. We assume that f_u is a *sufficient statistic* for f, $p(x_u|f_u, f) = p(x_u|f_u)$.

JENSEN'S INEQUALITY

Theorem 3.1 *Jensen's inequality [68] Let x be a random variable and let f be a convex function, then $\mathbb{E}[f(x)] \geq f(\mathbb{E}[x])$ or*

$$\int f(x)p(x)\mathrm{d}x \geq f\left(\int xp(x)\mathrm{d}x\right)$$

Variational learning [69] approximates the true posterior with a variational distribution, q, using the Kullback-Leibler divergence through Jensen's inequality,

$$p(f|y) \sim q(f) = \mathcal{GP}(\mu(\cdot), \Sigma(\cdot,\cdot)) \tag{3.22}$$

The KL divergence is a statistical distance between two distributions and assesses how close the variational distribution is to the true posterior [51]. Titsias [62] and Damianou [63] both detail this derivation

and Blei, Kucukelbir, and McAuliffe [69] elaborate variational learning. Titsias and Lawrence [67] extended this approach for GPLVM to obtain a lower bound on the marginal likelihood. Titsias [70] proves that the bound is at least equally tight when adding an inducing point in the supervised case.

This bound is used with variational inference and does not allow for stochastic variational inference (SVI) [71]. The bound was extended to the SVI setting by Hensman, Matthews, and Ghahramani [72], who also elaborates on latent variable models.

Parametrising the variational distribution as a Gaussian, $q(u) = \mathcal{N}(u|m, S)$ with m and S as variational parameters, Hensman, Fusi, and Lawrence [73] writes the evidence lower bound (ELBO) as

$$\log p(Y) = \log \int p(Y|X)p(X)\mathrm{d}X \tag{3.23}$$

$$\leq \int q(X)(L + \log p(X) - \log q(X))\mathrm{d}X \tag{3.24}$$

with

$$L = \left[\sum_{n=1}^{N} \log p(y_n|f(x_n))\right] - \mathrm{KL}(q(u)||p(u)) \tag{3.25}$$

Here, m and S can be evaluated analytically rather than by gradient optimisation [67], and SVI applies.

The GPLVM in its original form can be viewed as a decoder without an encoder. Lawrence and Quiñonero-Candela [41] was the first to employ an encoder though the authors termed it a back constraint. This way beneficial because, in some sense, the GPLVM is dissimilarity preserving [41], and by using an encoder, similarity preservation can be introduced. This is similar to variational autoencoder [6, 9]. Lately, this is growing more common Bui, Yan, and Turner [74] and Ramchandran, Koskinen, and Lähdesmäki [75] also trained using an encoder and recently, Lalchand, Ravuri, and Lawrence [42] generalised this to different types of encoders. Shu et al. [76] showed empirically that amortisation improved generalisation performance. In the next chapter, we develop a model using an encoder to learn the latent variables. We refer to this as amortisation.

3.4.1 Challenges

While variational learning of inducing points has immensely progressed the Gaussian process research, learning inducing points remains challenging.

First, in the supervised scenario (where the input is fixed), Bauer, van der Wilk, and Rasmussen [77] notes that inducing points are "*not completely trivial to optimise, and often tricks [...] are required*". This is further supported by Dutordoir, Durrande, and Hensman [78], who points out that the inducing points only act locally; hence, a large number may be required to cover the input space. This effect is even more pronounced in high-dimensional data (e.g. images) due to the *curse of dimensionality*. The fundamental assumption for variational learning of inducing points is that the inducing variables are a sufficient statistic for the map. Titsias [62] notes that these are difficult to find in practice.

Second, while lots of effort has been made to scale GPs with inducing points, scaling in the unsupervised setting is a side effect of the supervised case, as the GPLVM can be viewed as a regression. The novel paper introducing the Bayesian GPLVM [67] points out that the optimisation is challenging and prone to local minima as the inducing points and the latent points are coupled. These reasons motivate scaling unsupervised Gaussian processes without inducing points. We suspect this could be because we are learning a second set of free parameters in addition to the latent variables. This unconstrains the learning problem, effectively making it more challenging.

Third, we want a model that learns structure in the latent space for our specific goal of doing geometry. Often the Bayesian GPLVM learns approximately spherical latent representations due to the Gaussian prior on the latent variables. See, for example, Bui, Yan, and Turner [74]. The hyperparameters influence the learnt representation and estimating hyperparameters properly can also be a challenge in sparse models. Bauer, van der Wilk, and Rasmussen [77] comments that though the lower bound can identify good solutions, the variational learning of inducing points has a tendency to underfit in practice, e.g. not properly estimating length scales.

4 STOCHASTIC ACTIVE SETS FOR GAUSSIAN PROCESS DECODERS

This chapter is based on the NeurIPS paper "Revisiting Active Sets for Gaussian Process Decoders" by Moreno-Muñoz, Feldager*, and Hauberg [2].*

* *Equal contribution*

Our overarching quest is for a GPLVM that is suitable for geometry. Often the GPLVM [32] is used when data is scarce as it scales poorly and is prone to overfitting (chapter 2). We also reviewed the Bayesian GPLVM and its extensions (section 3.4) which address some of these challenges. Though Bayesian GPLVM provides excellent theoretical guarantees, in practice, it is difficult to train, and the learnt representations are often approximately spherical due to the prior on the latent variables (section 3.4.1). We now set out to find a scalable model that learns structure in the latent space that we can use for geometry.

This chapter introduces *stochastic active sets* (SAS) for Gaussian process decoders [2], which scales training of GPLVMs [32] to many observations, challenging the view that GPLVMs are only useful for small datasets. SAS repeatedly samples a subset of the data (the so-called *active set*), which is assumed fully correlated, while the remaining data (the *remainder set*) are assumed independently conditioned on the active set. This means that only the covariance estimated on the active set has to be inverted, as the covariance estimated on the remainder set is diagonal. This effectively reduces the size of the covariance matrix to be inverted and enables scaling the model the more observations. First, we introduce the notation (section 4.1). Then we review a theoretical link between the marginal likelihood and exhaustive cross-validation [79] (section 4.2) to derive this for both a GP decoder (section 4.3) and a Bayesian GP decoder (section 4.4). Finally, we end this chapter with results (section 4.5) and a discussion of the proposed models (section 4.6).

4.1 STOCHASTIC ACTIVE SETS

This section introduces the notation for stochastic active sets. An active set is an informative subset of data to be used for inference:

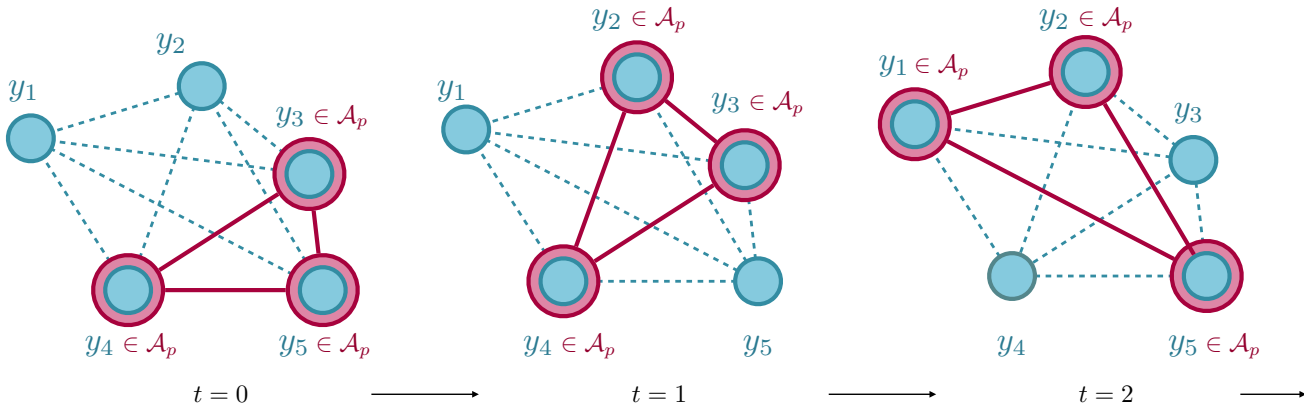


Figure 4.1: Illustration of SAS notation. The active set (red) is sampled from all observations (teal) at time zero. The solid red line indicates that a full covariance is estimated, and the teal dashed line indicates a conditional independence on the active set. The figure is slightly adapted from Moreno-Muñoz, Feldager, and Hauberg [2].

We split the N observations into two disjoint subsets; the active set y_A and the remainder set y_R. Similarly, we split the corresponding latent variables into two disjoint subsets as well $x_A \subset X$ and $x_R \subset X$ with $x_A \cup x_R = X$. We define A as the number of observations in the active set $A = |y_A|$, and R as the number of observations in the remainder set $R = |y_R|$. This gives the relation a $N = A + R$, which is true independently of how the observations are split.

Note, that $C = \binom{N}{A} = \binom{N}{R}$ by symmetry

There are many ways to split the set of observations. Given A, the binomial coefficient gives the number of possible active sets (or permutations), $C = \binom{N}{A}$. An indexing set indexes the active set denoted $\mathcal{A}_p$ where p refers to a specific permutation of the indices. The set

$$\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p, \ldots, \mathcal{A}_C\},$$

collects all the possible active sets. Similarly, we define the corresponding sets for the remainder set: The set

$$\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_p, \ldots, \mathcal{R}_C\},$$

collects all the possible remainder sets.

We adopt a shorthand notation for the kernel evaluation such that the subscript in the kernel indicates the set. This means that we denote the kernel evaluated on the active set as $k(x_A, x_A) = K_{x_A x_A} \equiv K_{AA}$ and similarly for the remainder set, i.e. $k(x_R, x_R) \equiv K_{RR}$.

4.2 MARGINAL LIKELIHOOD AND CROSS-VALIDATION

The SAS model relies on a theoretical result from Fong and Holmes [79]. This result relates the marginal likelihoods and Bayesian cross-validation[1]. This paper argues that if cross-validation and the marginal likelihood are equivalent, we should use the marginal likelihood for training. We flip this argument in the context of GPs: The marginal likelihood is hard to evaluate, so maybe we can gain something by formalising leave-p-out cross-validation. Fong and Holmes [79] shows that the marginal likelihood decomposes into two terms, and our goal is to relate that to active sets. This section focuses on understanding their result so we can develop the SAS decoder in the next section. We adapt the notation slightly to better suit our purpose. They focus on cross-validation and use the terminology hold-out set and training set. This is equivalent to active sets and remainder sets (due to symmetry).

1. In the following, we refer to the exhaustive, Bayesian cross-validation when we write cross-validation.

Definition 4.1 (Exhaustive, Bayesian cross-validation) *Leave-p-out Bayesian cross-validation is defined as*

$$S_{CV}(y_{1:n}; p) = \frac{1}{\binom{n}{p}} \sum_{t=1}^{\binom{n}{p}} \frac{1}{p} \sum_{j=1}^{p} \log p(y_j^{(t)} | y_{1:n-p}^{(t)}) \tag{4.1}$$

EXHAUSTIVE CROSS VALIDATION

The superscript t denotes the tth of the n-choose-p possible hold-out sets. The corresponding training set is $y_{1:n-p}^{(t)}$ such that the hold-out sets and the training set jointly constitute the entire dataset, $y_{1:n} = \{y_{1:p}^{(t)}, y_{1:n-p}^{(t)}\}$. We also refer to this as exhaustive cross-validation.

The cross-validation score S_{CV} is defined with the log posterior predictive probability. The interpretation of S_{CV} is the average predictive score on one test point.

Theorem 4.1 *The Bayesian marginal likelihood is equivalent to the exhaustive leave-p-out cross-validation score [79]*

$$\log p(y_{1:n}) = \sum_{p=1}^{n} S_{CV}(y_{1:n}; p) \tag{4.2}$$

using the log posterior predictive probability.

Theorem 4.1 states that the marginal likelihood is the sum of the cross-validation score, i.e. the sum of the average predictive score on one test point. Fong and Holmes [79] split the sum in theorem 4.1 at a

threshold P (which we will not discuss in further detail here, but Fong and Holmes [79, section 3.2] does),

$$\log p(y_{1:n}) = \sum_{p=1}^{n} S_{CV}(y_{1:n}; p) = \underbrace{\sum_{p=1}^{P} S_{CV}(y_{1:n}; p)}_{S_{CCV}} + \underbrace{\sum_{p=P+1}^{n} S_{CV}(y_{1:n}; p)}_{S_{PCV}}. \quad (4.3)$$

We refer to S_{CCV} as the cumulative cross-validation score or simply the cumulative scores; similarly, we refer to S_{PCV} as the preparatory cross-validation score or the preparatory score.

We can interpret S_{PCV} as the average log marginal likelihood over all possible training sets. Also, the S_{PCV} definition evaluates the log marginal likelihood on the whole training set. This corresponds precisely to constructing a full covariance matrix on the active set. We can interpret S_{CCV} as the cumulative average predictive score on the hold-out. To see how we need a few more steps.

As S_{CCV} and S_{PCV} each include a sum over p, this sum is determined if we know the limit P. For this reason, we write the sum given P instead of lowercase p,

$$\sum_{p=1}^{n} S_{CV}(y_{1:n}; p) = S_{CCV}(y_{1:n}; P) + S_{PCV}(y_{1:n}; P). \quad (4.4)$$

Fong and Holmes [79] proves that S_{PCV} and S_{CCV} can be expressed as

$$S_{PCV}(y_{1:n}; P) = \frac{1}{\binom{n}{P}} \sum_{t=1}^{\binom{n}{P}} \log p(y^{(t)}_{1:n-P}), \quad (4.5)$$

$$S_{CCV}(y_{1:n}; P) = \frac{1}{\binom{n}{P}} \sum_{t=1}^{\binom{n}{P}} \log p(y^{(t)}_{1:P} | y^{(t)}_{1:n-P}), \quad (4.6)$$

which are related to the original definition of exhaustive leave-p-out cross-validation score and sum to the log marginal likelihood (via theorem 4.1). This holds for all $1 \leq P < n$.

As mentioned, S_{CCV} can be interpreted as the cumulative average predictive score on the hold-out set. Still, for our purpose, we find a more convenient formulation by inserting the expression for the

cross-validation (equation 4.1) directly.

$$
\begin{aligned}
S_{CCV}(y_{1:n}; P) &= \sum_{p=1}^{P} S_{CV}(y_{1:n}; p) && (4.7)\\
&= \sum_{p=1}^{P} \frac{1}{\binom{n}{p}} \sum_{t=1}^{\binom{n}{p}} \frac{1}{p} \sum_{j=1}^{p} \log p(y_j^{(t)} | y_{1:n-p}^{(t)}) && (4.8)\\
&= \sum_{p=1}^{P} \frac{1}{p} \mathbb{E}_{\mathcal{R}} \left[\sum_{j=1}^{p} \log p(y_j | y_{1:n-p}) \right] && (4.9)
\end{aligned}
$$

Starting from the left, the sum over p is the sum over different sizes of hold-out sets. In the second equation, the sum over t (with the binomial coefficient) is a sum over all permutations of training (or remainder) sets. The sum over j collects contributions from all observations, i.e. one term in that sum corresponds to the average log predictive marginal likelihood for a single observation. More details can be found in Moreno-Muñoz, Feldager, and Hauberg [2, appendix].

4.3 GAUSSIAN PROCESS DECODER

Based on the insights from section 4.2, we can approximate the log marginal likelihood in the context of Gaussian processes. We will show that theorem 4.1 and its decomposition of the cross-validation can be formulated with active sets.

We can express S_{PCV} and S_{CCV} in the context of Gaussian processes using exhaustive cross-validation. For the non-Bayesian Gaussian GPLVM, we consider the marginal likelihood $p(y|x)$. Using the expression for S_{PCV} in equation 4.10 and putting it into the context of (non-Bayesian) Gaussian processes, we obtain

$$
S_{PCV}(y_{1:n}|x; P) = \mathbb{E}_{\mathcal{A}} \left[\log p(\boldsymbol{y}_A | \boldsymbol{x}_A) \right] \qquad (4.10)
$$

By symmetry, the expectation over permutation is equal for active sets (training sets) and remainder sets (hold-out sets). We replace the notation $y_{1:n-P}$ with y_A, which is the active set.

Likewise, we can reduce the expression for S_{CCV} (equation 4.9) to

$$S_{CCV}(y_{1:n}|x;P) = \sum_{p=1}^{P} \frac{1}{p} \mathbb{E}_{\mathcal{A}} \left[\sum_{i \in \mathcal{R}_p} \log p(y_i|y_A, x) \right] \tag{4.11}$$

$$= \mathbb{E}_{|A|} \mathbb{E}_{\mathcal{A}} \left[\sum_{i \in \mathcal{R}_p} \log p(y_i|y_A, x_A) \right]. \tag{4.12}$$

Here, we have three things. First, we assume that $p(y_A|x) = p(y_A|x_A)$. We can do this without loss of generality because the generative model states a one-to-one correspondence between latent variables and observation. Second, the expectation over permutation was rewritten from an expectation of permutations of the remainder set to an expectation permutation over the active set. This is due to symmetry. Finally, we used the expression we derived for S_{CCV} and S_{PCV} to rewrite the marginal likelihood.

$$\log p(y|x) = \mathbb{E}_{|A|} \mathbb{E}_{\mathcal{A}} \left[\sum_{n \in \mathcal{R}} \log p(y_n|y_A, x_A) \right] + \mathbb{E}_{\mathcal{A}} \left[\log p(y_A|x_A) \right] \tag{4.13}$$

This means that we construct a full covariance on the active set, K_{AA}, (see equations 4.14-4.17). Then each observation in the remainder set is predicted individually using K_{AA}. Equivalently, the observations in the remainder set are conditionally independent, given the active set. The terms in the sum can be evaluated using mini-batching and Gaussian predictive distributions.

$$p(y_A|x_A) = \mathcal{N}(y_A|0, K_{AA}), \tag{4.14}$$

$$p(y_R|y_A, x) = \mathcal{N}(y_R|m_R, C_R), \tag{4.15}$$

with posterior mean and variance [8]

$$m_R = K_{AR}^{\mathrm{T}} (K_{AA} + \sigma^2 \mathbb{I}_N)^{-1} x_A \tag{4.16}$$

$$C_R = K_{RR} + \sigma^2 \mathbb{I}_N - K_{AR}^{\mathrm{T}} (K_{AA} + \sigma^2 \mathbb{I}_A)^{-1} K_{AR}. \tag{4.17}$$

4.3.1 Learning the SAS Decoder

In practice, we have not gained much yet: Sampling A is quicker than inverting the full covariance matrix constantly (i.e. exact GPs), but we

Algorithm 1 Adam: Adaptive momentum estimation [80]. Learning of GP decoders with SAS. The algorithm is adapted from Moreno-Muñoz, Feldager, and Hauberg [2].

Input: Observed data y
Parameters: Initialise θ, ϕ // θ, z if non-amortized
for e **in** epochs **do**
  for b **in** batches **do**
    Sample $y_{\text{batch}} \sim y$
    Get $y_R, y_A \leftarrow$ `random_split`(y_{batch})
    if amortised **then**
      Get $\{x_R, x_A\} \leftarrow g(y_R, y_A|\phi)$
    end if
    Compute K_{AA}^{-1} // via Cholesky
    Evaluate $\log p(y_A|x_A)$
    Evaluate $\log p(y_n|x_A, z), \ \forall y_n \in y_R$
    $\hat{L}_{\text{approx}} = \sum_n \log p(y_n|y_A, x) + \log p(x_A|x_A)$
    do Adam(θ, ϕ) step for $\hat{L}_{\text{approx}}$
  end for
end for

would also sample a lot of large values of A, close to the number of observations. In the limit of $A = N$, this corresponds to an exact GP, so we are not on par with Bayesian GPLVM in terms of speed. To handle this, we fix $|A|$, which removes the expectation of $|A|$ and leads to a biased approximation of the marginal likelihood. We get back to this in section 4.6. Equation 4.18 states the implemented approximation of the log marginal likelihood for the GPLVM.

$$\log p(y|x) \approx \sum_{n \in \mathcal{R}} \log p(y_n|y_A, x_A) + \log p(y_A|x_A) \equiv \quad \hat{L}_{\text{approx}} \quad (4.18)$$

SAS APPROXIMATION
GP Decoder

Notice that the expectation over the set of permutations $\mathcal{A}$ is left out. We sample the active set in each mini-batch during training which corresponds to this expectation, so we do not have to compute it explicitly, see figure 4.1 and algorithm 1. The complexity reduces to $\mathcal{O}((N - A)A^2)$, which for small $|A|$ is cheaper than inducing points methods. This approach is summarised by algorithm 1.

4.3.2 Amortisation of the SAS Decoder

We employ a neural network to *amortise* or encode the latent variables discussed in section 3.4. We refer to this as the amortisation network, $g(Y) : \mathbb{R}^D \to \mathbb{R}^Q$ with weights $\phi = \{W_1, W_2, W_2\}$, which encodes the

latent variables X using hidden units Z.

$$Z_n = a_n(W_n Z_{n-1} + b_n) \tag{4.19}$$

where the activation functions in the hidden layers are ReLU [81, 82], the identity in the last, $a_n(y) = y$, $Z_n = X$ and $Z_0 = Y$. In practice, we use three layers.

4.4 BAYESIAN GAUSSIAN PROCESS DECODER

The SAS approximation also applies to the Bayesian GPLVM (section 3.4). Here, we aim to estimate the posterior $p(x|y)$ using the marginal likelihood $p(y)$. As Titsias and Lawrence [67] showed, this is not tractable in closed form because the kernel function is non-linear in the latent variables. Instead, we aim to derive a lower bound on the evidence, L_{ELBO} by considering the log of the marginal likelihood, marginalised with respect to the latent variables

$$\log p(y) = \log \int p(y, x) \mathrm{d}x \tag{4.20}$$

Here, we would like to introduce a variational distribution on the latent variables, $q(x)$. We do this through Jensen's inequality (theorem 3.1) which lets us write a bound on the marginal likelihood. This lets us rewrite the expression for the marginal likelihood to

$$\begin{aligned}
\log p(y) &= \log \int \frac{q(x)}{q(x)} p(y, x) \mathrm{d}x && (4.21)\\
&\geq \int q(x) \log \frac{p(y, x)}{q(x)} \mathrm{d}x && (4.22)\\
&= \int q(x) \log p(y|x) \mathrm{d}x + \int q(x) \log \frac{p(x)}{q(x)} \mathrm{d}x && (4.23)\\
&= \mathbb{E}_{q(x)}[\log p(y|x)] - \mathrm{KL}[q(x)||p(x)] && (4.24)\\
&\equiv L_{\text{ELBO}} && (4.25)
\end{aligned}$$

Generally, this expectation is hard to compute because the integration over the latent variables is non-linear through the Gaussian likelihood $p(y|x)$, equation 3.9. In equation 4.13, we derived an expression for $\log p(y|x)$, which we can insert in the expectation. Consider the expectation only

$$\mathbb{E}_{q(x)}[\log p(y|x)] = \mathbb{E}_{q(x)}\left[\mathbb{E}_{|A|}\mathbb{E}_{\mathcal{A}}\left[\sum_{n\in\mathcal{R}}\log p(y_n|y_A, x_A)\right] + \mathbb{E}_{\mathcal{A}}\left[\log p(y_A|x_A)\right]\right] \tag{4.26}$$

$$= \mathbb{E}_{|A|}\mathbb{E}_{\mathcal{A}}\mathbb{E}_{q(x)}\left[\sum_{n\in\mathcal{R}}\log p(y_n|y_A, x_A)\right] + \mathbb{E}_{\mathcal{A}}\mathbb{E}_{q(x)}\left[\log p(y_A|x_A)\right] \tag{4.27}$$

which gives the approximate lower bound under the SAS approximation

$$\hat{L}_{\text{ELBO}} = \mathbb{E}_{|A|}\mathbb{E}_{\mathcal{A}}\mathbb{E}_{q(x)}\left[\sum_{n\in\mathcal{R}}\log p(y_n|y_A, x_A)\right] + \mathbb{E}_{\mathcal{A}}\mathbb{E}_{q(x)}\left[\log p(y_A|x_A)\right] - \text{KL}[q(x)||p(x)]. \tag{4.28}$$

The first term is the expectation over S_{CCV} with respect to $q(x)$, and the second term is the expectation over S_{PCV} with respect to $q(x)$. When the Kullback-Leibler divergence goes to zero, the bound is tight, and the approximated ELBO reduces to the true ELBO.

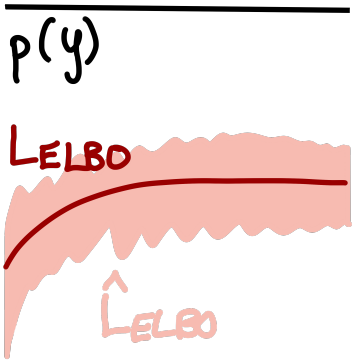


Figure 4.2: Illustration of the marginal likelihood, $p(y)$ (black), the true bound, L_{ELBO} (red) and the estimated bound, $\hat{L}_{\text{ELBO}}$ (light red).

4.4.1 Learning the Bayesian SAS Decoder

Like the SAS decoder, not much is gained in practice yet, and like the SAS decoder, the Bayesian SAS decoder fixes $|A|$. This is to avoid sampling large values of active set sizes, $A \sim N$, where the computational cost is the same as for the exact GPLVM. This removes the expectation of $|A|$ and leads to a biased approximation of the ELBO.

$$\hat{L}_{\text{ELBO}} \approx \mathbb{E}_{q(x)}\left[\sum_{n\in\mathcal{R}}\log p(y_n|y_A, x_A)\right] + \mathbb{E}_{q(x)}\left[\log p(y_A|x_A)\right] - \text{KL}[q(x)||p(x)] \tag{4.29}$$

$$= \sum_{n\in\mathcal{R}_p}\mathbb{E}_{q(x_n)}\left[\log p(y_n|y_A, x)\right] + \mathbb{E}_{q(x_A)}\left[\log p(y_A|x_A)\right] - \sum_{n=1}^{N}\text{KL}[q(x_n)||p(x_n)] \tag{4.30}$$

Again, notice that the expectation over the set of permutations $\mathcal{A}$ is left out. We sample the active set in each mini-batch during training which corresponds to this expectation, so we do not have to compute it explicitly, see figure 4.1 and algorithm 1. We use mean-field variational inference, which allows for mini-batching of the KL term [9], yielding the SAS approximation of the evidence lower bound, $\hat{L}_{\text{ELBO}}$. The complexity is the same as for the SAS decoder, $O((N-A)A^2)$ and the approach is summarised by algorithm 2 and figure 4.2.

4.4.2 Amortisation of the Bayesian SAS Decoder

As for the non-Bayesian SAS decoder, we amortise the latent means and variances with two neural networks: A neural network $g_\mu : \mathbb{R}^D \to \mathbb{R}^Q$ encodes the mean of the latent variables, and another neural network encodes the variance, $g_\sigma : \mathbb{R}^D \to \mathbb{R}^Q$ in which the last layer in includes a SoftPlus function given by $\log(1 + \exp X)$ which ensures is strictly positive variance.

The amortisation leads this variational distribution on X,

$$q(X) = \prod_{n=1}^{N} \mathcal{N}(x_n; g_\mu(y_n), g_\sigma(y_n)). \tag{4.31}$$

The amortisation is a parametrisation of the mean and covariance of the variational distribution [41, 74, 83], first suggested by Lawrence and Quiñonero-Candela [41] to learn GP representations that preserve local structures. The encoder also ensures that close observations are close in the latent space. The update of the network parameters also affects the other batches, and the network parameters are global [71], which enables stochastic variational inference [72].

We derived a lower bound on the evidence $p(y)$ that we call L_{ELBO} or the *true* lower bound. Using the SAS approximation, we derived an estimator for the true lower bound that we name the estimated lower bound, $\hat{L}_{\text{ELBO}}$. When training, we search for the parameters that bring $\hat{L}_{\text{ELBO}}$ (and therefore L_{ELBO} closer to the marginal likelihood $p(y)$. Algorithm 2 summarises the SAS approximation for Bayesian GP decoders. Operationally, this approach is straightforward and scales similarly to mini-batched inducing points methods [72].

Algorithm 2 Adam: Adaptive momentum estimation [80]. Learning of Bayesian GP decoders with SAS. The algorithm is adapted from Moreno-Muñoz, Feldager, and Hauberg [2].

Input: Observed data y
Parameters: Initialise θ, ϕ // θ, μ, σ^2 if non-amortized
for e **in** epochs **do**
  for b **in** batches **do**
    Sample $y_{\text{batch}} \sim y$
    Get $y_R, y_A \leftarrow$ `random_split`(y_{batch})
    if amortised **then**
      Get $\mu_x \leftarrow g_\mu(y_R, y_A|\phi)$
      Get $\sigma_x \leftarrow g_\sigma(y_R, y_A|\phi)$
    end if
    Sample $\{x_R, x_A\} \sim q(\mu_x, \sigma_x^2)$ // RT
    Compute K_{AA}^{-1} // via Cholesky
    Evaluate $\hat{L}_{\text{ELBO}}$ in equation 4.29
    do Adam(θ, ϕ) step for $\hat{L}_{\text{ELBO}}$
  end for
end for

4.5 RESULTS

This section investigates the SAS and the Bayesian SAS decoders (collectively referred to as the SAS models or the SAS decoders) and compares them to other models. After some experimental details, we first investigate the effect of the active set size on the approximation and the SAS models for more latent dimensions than two. Second, we introduce the baselines and look into both runtimes and structure of the latent space.

4.5.1 Experimental Setup

We evaluate the model shown here on datasets; MNIST[84], FASHION-MNIST [85], and CIFAR-10 [86]. These are three image datasets often used for supervised learning. Still, all the models in this section are unsupervised, i.e. they do not have access to the class labels (unless explicitly stated in the text).

The specifics of the SAS models are detailed in sections 4.3.1 and 4.4.1, implemented accordingly in PYTORCH [87] and run on CPU. The experiments used learning rates in the range $[10^{-4}, 10^{-2}]$ and trained for 300 epochs using the adam optimiser [80] in 64-bit precision. The SAS models only use batch sizes greater than active set sizes, $B > A$,

as the active set is subsampled from the batch. The batch size is 1024 unless otherwise stated.

4.5.2 Evaluation of The SAS Models

As more information is retained, we expect better representation with increasing active set sizes. Figure 4.3 shows this in terms of the approximated marginal likelihood for the trained model. We also used three metrics standard for GP regression (which unsupervised GP essentially is).

Assuming a test set $y^* = \{y_n^*\}_{n=1}^{N_*}$, we define the root mean square error (RMSE)

$$\text{RMSE}(y^*) = \sqrt{\frac{1}{N_*}\sum_{n=1}^{N_*}(y_n - \mu_n^*)^2}.$$

We define the mean absolute error (MAE)

$$\text{MAE}(y^*) = \frac{1}{N_*}\sum_{n=1}^{N_*}\left|y_n^* - \mu_n^*\right|.$$

RSME penalises observations far from mean harder the MAE does. The negative log predictive distribution (NLPD)

$$\text{NLPD}(y^*) = \frac{1}{2}\log(2\pi) + \frac{1}{2N_*}\sum_{n=1}^{N_*}\left[\log v_n^* + \frac{(y_n^* - \mu_n^*)^2}{v_n^*}\right],$$

Here, μ_n^* and v_n^* are the predictive mean and variance for the nth test point, respectively. The NLPD also accounts for the predictive variance, unlike RSME and MAE, which only considers the predicted mean.

Table 4.1 provides the metrics for the SAS decoders on Mnist, FashionMnist, and Cifar-10. For all three metrics, lower is better. For each model on each dataset, we consider three active set sizes. We observe smaller metrics with increasing active set sizes. This is consistent with the SAS approximation capturing the covariance structure with increasing active set sizes.

This behaviour is also evident in figure 4.3. The SAS decoder (top row) is consistent in smaller active set sizes converging faster and to higher values of the bound $\hat{L}_{\text{ELBO}}$. This is harder to tell for Cifar-10, where learning is more complex, and the training curve for $A = 400$

Table 4.1: RMSE: Root mean square. MAE: Mean absolute deviation. NLPD: Negative log predictive distribution. The table shows three metrics (RMSE, MAE, and NLPD) for the SAS decoder and the Bayesian SAS decoder, each for three datasets. For each metric, we trained five models to obtain the standard error of the mean. Lower metrics are better, and the lowest is in bold font. The batch size is $B = 1024$. The table is slightly adapted from Moreno-Muñoz, Feldager, and Hauberg [2].

		SAS			Bayesian SAS		
Model	Metric	$A = 100$	$A = 200$	$A = 400$	$A = 100$	$A = 200$	$A = 400$
MNIST	RMSE	2.55 ± 0.98	2.47 ± 0.98	**2.41 ± 0.93**	2.16 ± 0.02	2.08 ± 0.02	**1.99 ± 0.02**
	MAE	1.61 ± 0.97	1.55 ± 0.99	**1.51 ± 0.96**	1.11 ± 0.02	1.04 ± 0.02	**0.96 ± 0.01**
	NLPD	2.99 ± 1.41	2.92 ± 1.38	**2.84 ± 1.31**	2.33 ± 0.03	2.26 ± 0.02	**2.17 ± 0.02**
FMNIST	RMSE	2.37 ± 0.95	2.31 ± 0.94	**2.25 ± 0.90**	1.99 ± 0.17	1.88 ± 0.20	**1.85 ± 0.13**
	MAE	1.48 ± 0.91	1.42 ± 0.91	**1.39 ± 0.89**	1.11 ± 0.02	1.02 ± 0.03	**0.98 ± 0.02**
	NLPD	2.76 ± 1.33	2.71 ± 1.31	**2.65 ± 1.23**	2.16 ± 0.18	2.07 ± 0.19	**2.04 ± 0.12**
CIFAR-10	RMSE	2.66 ± 1.08	2.55 ± 1.06	**2.55 ± 1.03**	2.74 ± 1.07	2.64 ± 1.08	**2.57 ± 1.02**
	MAE	1.77 ± 1.06	1.69 ± 1.06	**1.69 ± 1.02**	1.84 ± 1.03	1.76 ± 1.05	**1.71 ± 1.03**
	NLPD	3.20 ± 1.55	**3.07 ± 1.44**	3.32 ± 1.89	3.24 ± 1.53	3.14 ± 1.53	**3.06 ± 1.45**

has not converged. For the Bayesian SAS decoder, large active set sizes converge to better bound for MNIST and FASHIONMNIST. For CIFAR-10, the trend is the opposite.

Higher Latent Dimensionality

To show that our model works for more than two latent dimensions, we tested three and four latent dimensions for MNIST and FASHIONMNIST. In all cases, the training curves seem to converge (figure 4.5), and we observed a similar performance to the experiments for two latent dimensions. This implies that the SAS decoders are not restricted to two latent dimensions.

4.5.3 Representation performance

We consider three baselines: The GPLVM [32] as a baseline for the SAS decoder, the Bayesian GPLVM [67] as a baseline for the Bayesian SAS decoder, and finally, a variational autoencoder (VAE) [6] which is standard in representation learning. As we described in section 3.4, numerous extension has been made to the Bayesian GPLVM accord-

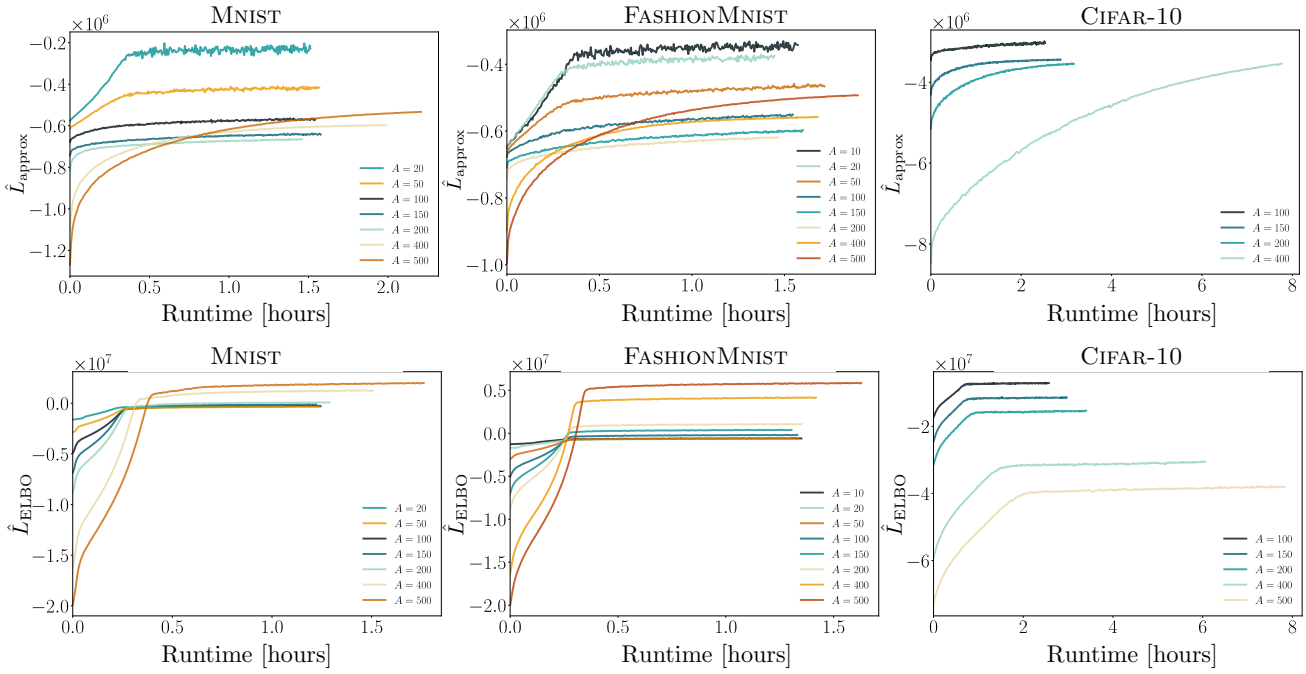


Figure 4.3: The figures show the approximated log-likelihood as a function of wall-clock time. We refer to these curves as training curves. Each plot shows the training curves for different active set sizes (indicated in the legend). The SAS decoder (top row) and the Bayesian SAS decoder (bottom row) were trained on MNIST (left column), FASHIONMNIST (centre column), and CIFAR-10 (right column). The figure is slightly adapted from Moreno-Muñoz, Feldager, and Hauberg [2].

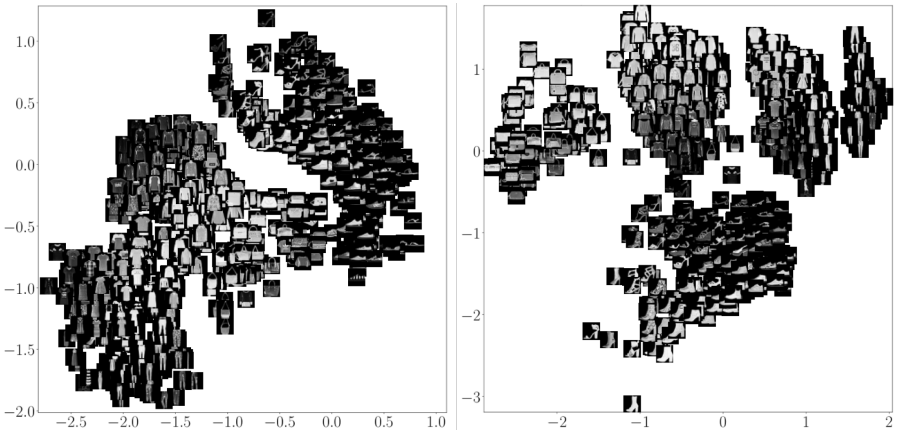


Figure 4.4: Two examples of learnt representation of FASHIONMNIST for the SAS decoder (left) and the Bayesian SAS decoder (right). The figure is slightly adapted from Moreno-Muñoz, Feldager, and Hauberg [2].

ingly, and we adapt both the GPLVM and the Bayesian GPLVM to enable a fairer comparison.

Chapter 2 showed that the initialisation affects the learnt representation for GP models. In amortised models, the initialisation of the latent variables is subject to the random initialisation. This initialisation is not directly comparable to either PCA or isomap initialisations. For this reason, we amortise both the GPLVM and the Bayesian GPLVM. The neural network encoding the latent variables in the GPLVM has the same architecture as the one for the SAS decoder(section 4.3.1).

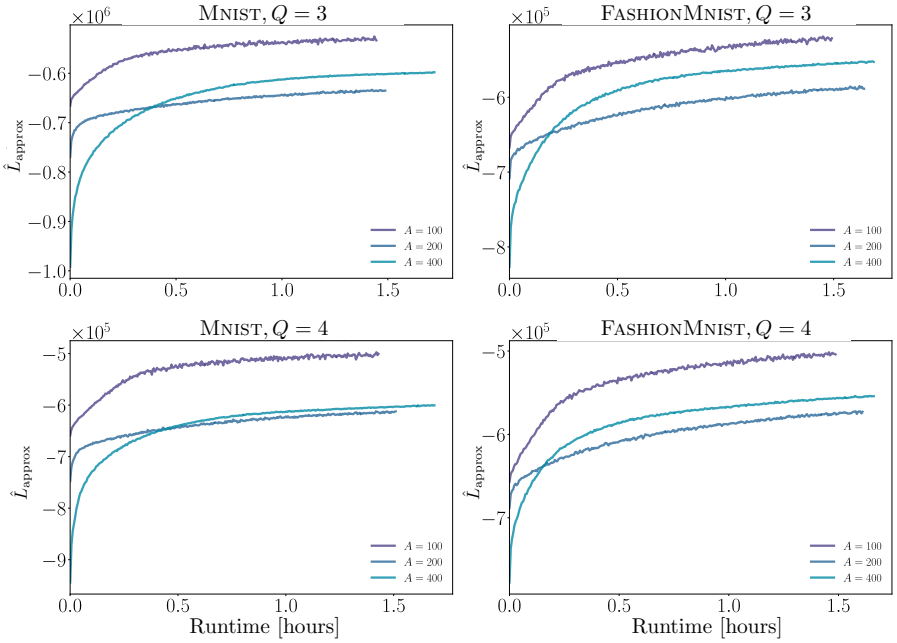


Figure 4.5: Training curves for different active set sizes A in three latent dimensions (top row) and four latent dimensions (bottom row) for MNIST (first column) and FASHION MNIST (second column). The figure is slightly adapted from Moreno-Muñoz, Feldager, and Hauberg [2].

The neural networks encoding the means and variances of the latent variables in the Bayesian GPLVM have the same architectures as the encoders for the Bayesian SAS decoder (section 4.4.1). We also use stochastic variational inference (see section 3.4) and implement these models in pyro [88].

Finally, the VAE employs the same encoder architectures as the encoders for the Bayesian SAS decoder. The decoder is a linear, fully connected layer with $H = 400$ hidden units $g_1 : \mathbb{R}^Q \to \mathbb{R}^H$, a SOFTPLUS activation function, another linear, fully connected layer $g_2 : \mathbb{R}^H \to \mathbb{R}^D$ and then a sigmoid activation.

Generally, for a fair comparison, we make several modelling choices: All models use Gaussian likelihood (though this is not optimal for images), and the four GP models all use the EQ kernel (with initial the initial kernel variance $\theta = 0.5$, initial length scale $l = 0.1$, and initial noise variance $\sigma_n^2 = 0.5$), and we only consider latent spaces of dimensionality two unless otherwise stated. All five models are optimised with respect to their respective approximation of the log marginal likelihood.

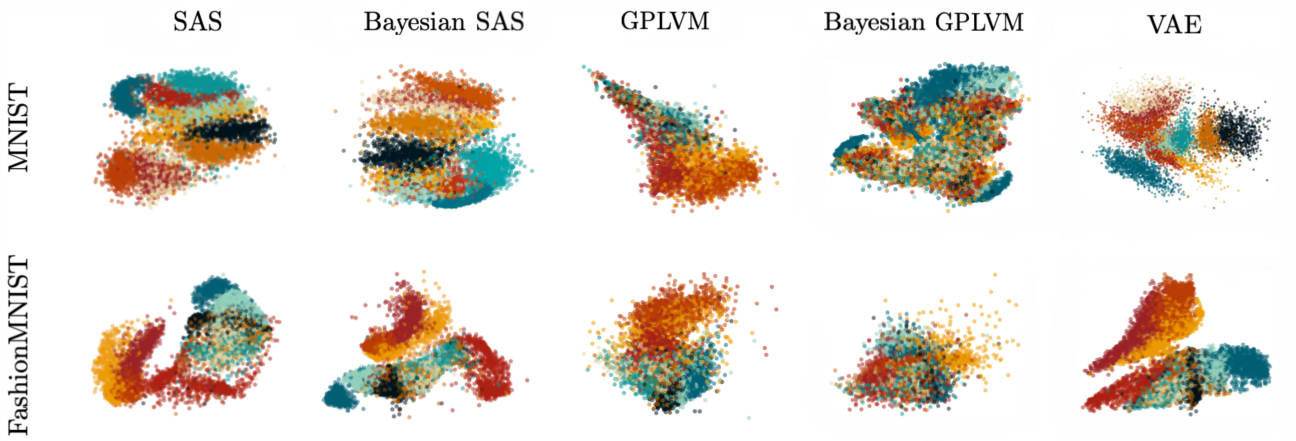


Figure 4.6: Learnt Representations for the SAS decoder, the Bayesian SAS decoder, the GPLVM, the Bayesian GPLVM, and the VAE on MNIST (top row) and FASHIONMNIST (bottom row).

Learnt Representations

The SAS models can lead to structured representations (figure 4.4). The choice of datasets is convenient because the images are labelled. Then we can define structure in the latent space as a representation that clusters images according to the class label. Representative learnt representations of MNIST and FASHIONMNIST are shown in figure 4.6.

One method to quantify structure is to define a downstream classifier, so we defined a nearest neighbour classifier [51] on the latent variables from the encoded test set. Figure 4.7 shows the classifier's accuracy on the test data as a function of the active set size; it is monotonic and flattens, suggesting that approximately $|A| \sim 150$ is sufficient for capturing structure. Table 4.2 shows the accuracies in the Bayesian SAS decoder, the Bayesian GPLVM and the VAE. The classifier trained on the test data encoded using the Bayesian SAS decoder obtained a higher classification accuracy than the Bayesian GPLVM and the VAE.

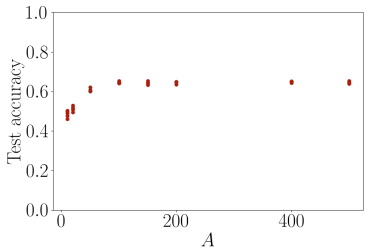


Figure 4.7: Test accuracy of downstream classifier of SAS decoder as a function of the active set size on FASHIONMNIST.

Table 4.2: Classification accuracy ± standard error of the mean on the encoded test set. Larger is better. The table is slightly adapted from Moreno-Muñoz, Feldager, and Hauberg [2].

	Dataset	
Model	MNIST	FASHIONMNIST
Bayesian SAS decoder	**0.63 ± 0.022**	**0.63 ± 0.020**
Bayesian GPLVM	0.18 ± 0.033	0.24 ± 0.043
VAE	0.54 ± 0.026	0.58 ± 0.008

Runtimes

The training curves for the Bayesian GPLVM and the Bayesian SAS decoder for different active sets sizes are shown in figure 4.8 (first two panes from the left). These show the runtime and convergence of the SAS approximation for comparable A (in Bayesian SAS decoder) and several inducing points (in Bayesian GPLVM). Qualitatively, we observe the same pattern: Smaller A leads (number of inducing points) to shorter convergence times. All the SAS models in figure 4.8 converge within two hours.

The right pane in figure 4.8 shows a smaller experiment in which SAS decoders were trained on a subset of MNIST, $N = 40,000$ samples. This enabled evaluating the true marginal likelihood, $p(y|z)$. First, we see that the SAS decoder indeed seems to be a lower bound, and second, the SAS model approximates the marginal likelihood well in figure 4.8.

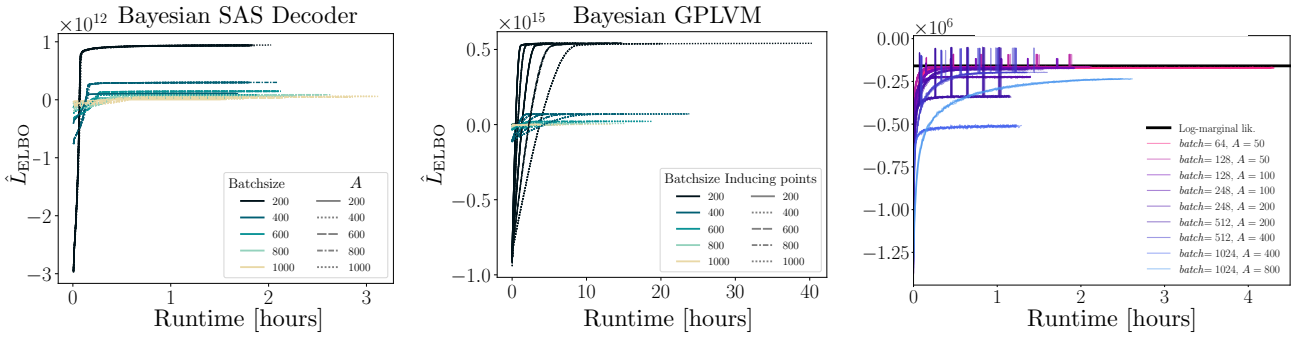


Figure 4.8: Left: Training curves for the Bayesian SAS decoder with different active sets and batch sizes. Centre: Training curves for the Bayesian GPLVM with different numbers of inducing points and batch sizes. Right: Training curves for Bayesian SAS decoder. The exact log-marginal likelihood is shown in black. All models were trained on 40,000 samples MNIST and repeated five times per batch size and active set size/number of inducing points. The figure is slightly adapted from Moreno-Muñoz, Feldager, and Hauberg [2].

4.6 DISCUSSION

Training GPs with inducing points is not straightforward (section 3.4.1) and unsupervised training often proves more challenging. We revisited active sets from a stochastic perspective and developed the SAS decoder and the Bayesian SAS decoder as an alternative to current methods. Our experiment illustrates the performance of our models compared to relevant baselines on image data. Here, the SAS models learn more structured representations (table 4.2) compared to both inducing points methods and, surprisingly, to a comparable VAE. The latter is surprising as Gaussian process representation generally cannot compete with VAEs for large datasets.

The SAS decoders are scalable training schemes for Gaussian processes with several benefits. First, the convergence of the Bayesian SAS decoder is on par or faster than its baseline. Second, the experiments presented have been done in 64-bit precision to enable a fair comparison of runtimes. Still, we observe that the SAS models run stable in 32-bit numerical precision with less jitter, unlike the baseline counterparts. Generally, we find that training SAS models require surprisingly few tricks (e.g. less tuning of initialisation and parameters) compared to other GP models. This robustness is partly due to the amortisation networks that the SAS models use. Appendix D contains an ablation study of the SAS models with and without amortisation.

The SAS models do not provide an approximated predictive distribution. Though SAS speeds up training, prediction is still $O(N^3)$. The Bayesian GPLVM provides scalable predictive distributions and theoretical guarantees (section 3.4) but is hard to train. We consider SAS a scalable training scheme (due to the lack of predictive distribution) that might be used for pre-training the Bayesian GPLVM.

Fixing the active set size introduces a bias to the approximation of the marginal likelihood. Fong and Holmes [79] suggests an unbiased estimator by sampling A, but this is more expensive, and the batch size limits the active set size. This bias is not concerning as we observe the training curves for SAS approximating the lower bound well. In figure 4.3 (right pane), the training curves for $A \sim B$ seem to converge to the true marginal likelihood, which is consistent with the findings of Chen et al. [89].

This work focuses on the stochastic active sets approximation in the unsupervised setting, but the idea is easy to generalise to the supervised setting. Extending to classification might prove more challenging. The stochastic active sets approximation relies on the Gaussian likelihood. This limits the type of data the model can use, e.g. it is not suitable for discrete targets. It would require a new derivation to implement the same idea.

This ends the discussion of the results from the original paper. In the next section, we consider SAS from the perspective of geometry.

4.6.1 Symmetry Breaking in the SAS Decoder

We discussed the problem of symmetry breaking in chapter 2 and have now developed the SAS decoder, which we observe to stabilise training. An obvious question is whether the amortisation could counter the symmetry breaking. Interestingly, it does the opposite: It seems that

the amortisation induces symmetry breaking, and it has not been possible to find a model in which the symmetry is preserved.

This may seem surprising as Lawrence and Quiñonero-Candela [41] fix breaks in the latent space using a back constraint. The amortisation requires more observations due to the training of the neural network, and the symmetry measure (equation 2.4) does not account for varying data points and consequently increases density, so this may not be surprising. With many observations, the median distance becomes relatively small; therefore, any more considerable distances are classified as a symmetry break. Our symmetry estimator did not account for that, so the result may not be directly comparable.

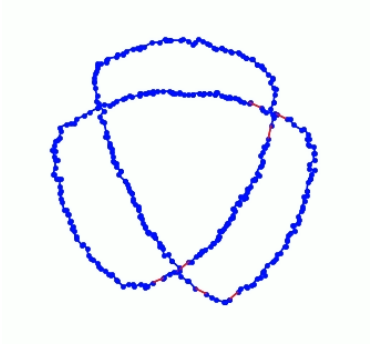


Figure 4.9: The SAS decoder consistently captures the reflection symmetry in a COIL object.

An alternative explanation is that the SAS decoder discovers the double loop, leading to symmetry breaking. Due to the encoder, it is initialised as noise but quickly recovers the double loop. Once the double loop is found, the model cannot recover. This is consistent with what we observed for GPLVM (chapter 2), where we observed a clear correlation between the prevalence of symmetry breaking and reflection symmetries within the COIL object. It remains unclear if the SAS model captures this additional symmetry or exhibits a strong tendency to clustering.

We set out to find a model that scales and learns structured representation. To achieve this, we developed the SAS decoders, but the SAS models lack a scalable predictive distribution. For this reason, we take the best of both worlds and use SAS as a pre-trained model with which we can initialise a GPLVM to continue into geometry.

RIEMANNIAN GEOMETRY 5

Manifold learning relies on the manifold hypothesis which states that data live on a noisy, low-dimensional manifold embedded in a high-dimensional space. There are numerous ways of recovering a manifold, e.g. the dimensionality reduction models, we considered in chapter 2 but non-probabilistic models like isomap [7] and t-SNE [39] recover only a representation of observed data, meaning that the manifold is only well defined where there are data. On the other hand, probabilistic models attempt to recover the distribution of data which is defined away from data. This is one of the arduous tasks in unsupervised learning [51].

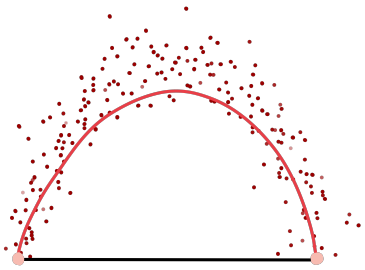


Figure 5.1: Observations (red) with a Euclidean (black) and Riemannian (light red) distances.

Recovering the generating distribution can have several purposes but here we aim to *learn* from the latent space. Assuming that a manifold actually exists, we should consider what meaningful distances are. We argue that distances are *more* meaningful when computed on the manifold. In this thesis, we assume that the manifolds are Riemannian and figure 5.1 shows an example of two distances computed on a dataset and only one respects the manifold, namely the Riemannian distance. By tackling distances on Riemannian manifolds, we are moving from topology (chapter 2) to geometry (this chapter).

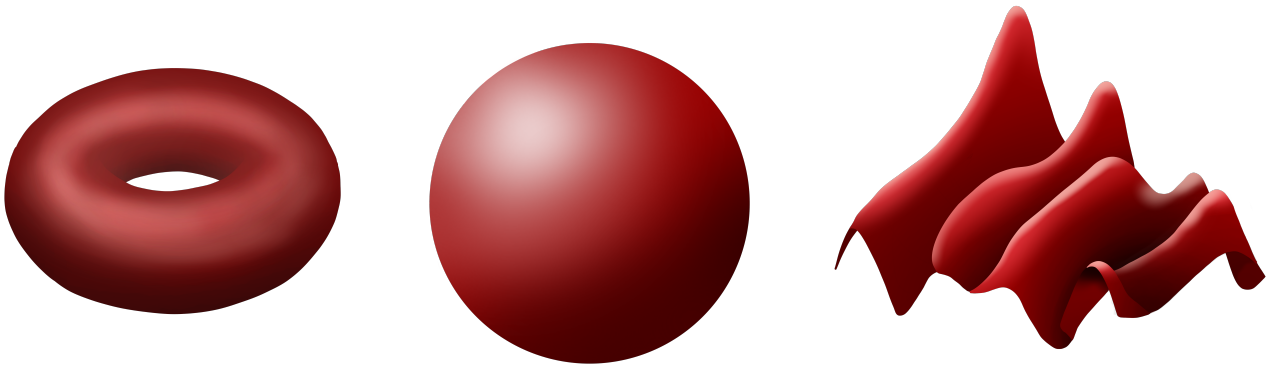


Figure 5.2: A torus, a sphere and an arbitrary, smooth surface are examples of Riemannian manifolds. The parametrisations of the torus and the sphere are known and the latter must be learnt.

Computing distances on a Riemannian manifold requires a Riemannian metric which can either be parametrised or not parametrised. Figure 5.2 shows three examples of smooth manifolds; the torus, the sphere and an arbitrary manifold. The first two are examples of *parametrised* manifolds where the metric is known. The latter is an example of a manifold where the parametrisation is not known so the

metric must be *learnt*. We consider these types of manifolds and this chapter introduces geometry context of manifold learning.

This chapter is based on a mixture of the works of Schutz [90] who introduces differential geometry for general relativity and Pressley [91] who introduces classical differential geometry in three dimensions and Pennec [92] who does statistics on Riemannian manifolds. The sources describe Riemannian geometry which provides tools for describing curved surfaces using tangent spaces (section 5.2.1), metric tensors (section 5.2.2), and geodesics (section 5.3), which are based on tensor calculus (section 5.1). This enables computing geodesic distances on manifolds.

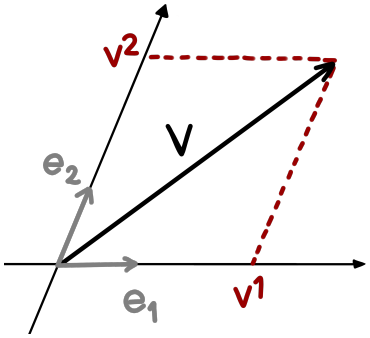


Figure 5.3: Components (red) of a contravariant vector (black).

5.1 TENSOR CALCULUS

At the heart of Riemannian geometry lies tensors. There are multiple ways of representing tensors: Like matrix notation (which we assume the reader is familiar with), Einstein notation is a compact way of writing multiple equations simultaneously and here, we briefly introduce Einstein notation [90] using an example.

The vector V in figure 5.3 can be expressed in terms of its contravariant components, V^1 and V^2, and its basis vectors, e_1 and e_2 as $V = V^1 e_1 + V^2 e_2$. This is the "usual" description of vectors which is contravariant because the components of V transform contrary to the basis, e.g. they decrease if the lengths of the basis vectors are increased.

Covectors can be thought of as linear maps from a vector space to its field of scalars. This is harder to visualise but we can think of covectors as a series of surfaces where the density of the surfaces corresponds to the magnitude of the covector (figure 5.4 left). The larger spacing, the smaller magnitude. The gradient of a function is a covector which means that $\frac{\partial}{\partial x^i} \equiv \partial_i$. We write covectors with a subscript. The product of a covector and a vector can be illustrated as the number of surfaces pierced by the vector (figure 5.4 middle and right). The smaller the covector, the smaller product as the vector pierces fewer surfaces.

Einstein notation is a representation of tensors that uses sub- and superscripts to distinguish algebraic objects, e.g. contravectors (or simply vectors) and covectors. The contra- and covariant vectors refer to how the vector transform and it should be clear that $V^i \neq V_i$, generally. These concepts let us define tensors.

TENSOR

Definition 5.1 (Tensor) *An $\binom{M}{N}$ tensor, T is a linear function of M covectors and N vectors into scalars.*

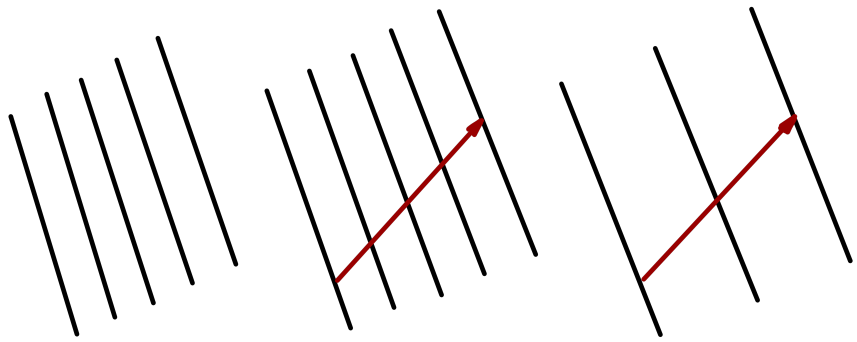


Figure 5.4: Left: A covector illustrated as a density of surfaces. Middle: The product of a covector and a vector illustrated as the number of surfaces pierced by the vector. Right: The product of the vector and a covector is smaller if the covector is smaller.

A tensor T^p_r is a (p, r) tensor or a tensor of rank (p, r) which distinguishes between co- and contravariant indices. We can use this to generalise for tensors of higher rank and define tensors as a collection of vectors and covectors that combine using a linear map. This definition of tensors is useful when considering the metric (section 5.2.2).

With the definition of tensor, we can outline a set of simple rules for Einstein notation.

1. Free indices (indices that only occur once per term) can take any value, and each value corresponds to an equation.

2. An index that occurs both as a subscript and as a superscript within a term implies summation though the sum is not stated explicitly. In the example above, this means $V = V^1 e_1 + V^2 e_2 = V^i e_i$.

3. Such a repeated index is also referred to as a dummy index and can be renamed for convenience. The vector V can be written as $V = V^i e_i = V^j e_j$.

In the following, we switch notation as convenient.

5.2 RIEMANNIAN MANIFOLDS

This section defines the tangent space and the metric (so we can define Riemannian manifolds), the covariant derivative ∇ (which provides a way of smoothly connecting tangent spaces) and the Christoffel symbols Γ^i_{jk} (which account for the corresponding change in basis vectors).

5.2.1 Tangent Spaces

Tangent spaces are tangent to the manifold and spanned by tangent vectors.

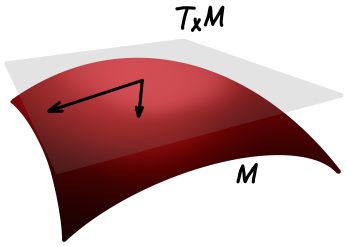


Figure 5.5: Illustration of the tangent space (grey) as a Euclidean space tangential to the manifold (red).

Definition 5.2 (Tangent Vector) *The tangent space is a D-dimensional, linear subspace that touches the manifold $\mathcal{M}$ tangentially at a point $x \in \mathcal{M}$. The tangent space $\mathcal{T}_x\mathcal{M}$ of the manifold $\mathcal{M}$ at the point x is the set of all tangent vectors to $\mathcal{M}$ at x.*

Figure 5.5 illustrates a flat tangent space on a two-dimensional manifold embedded in three dimensions. Notice, the tangent space has the same dimensionality as the manifold. Tangent spaces provide local Euclidean representations of curved surfaces where the usual (i.e. Euclidean) calculus applies and on which we can define a metric. We can obtain the (covariant) basis vectors of a tangent space at any point x on the manifold as the derivative of the manifold at x.

5.2.2 The Metric Tensor

The metric tensor is essential in Riemannian geometry as it lets us define geometric quantities like length, volumes, angles, and curvature. We can interpret the metric as a collection of basis vectors. These basis vectors are tangent vectors that span a tangent space and the metric is the inner product of the tangent vectors in the tangent space.

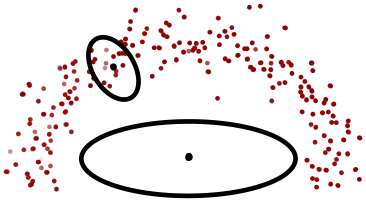


Figure 5.6: Illustration of the Riemannian metric tensor as an ellipsis that is larger away from data and smaller closer to data.

Definition 5.3 (Riemannian metric) *A Riemannian metric, g_{ij}, at a point x on a smooth manifold M is a smoothly varying inner product of two vectors in the tangent space $\mathcal{T}_xM$,*

$$g_{ij} = \langle \cdot, \cdot \rangle_x$$

where the Riemannian metric, g_{ij}, is a symmetric and positive definite (0,2)-tensor. This metric is sometimes called a pull-back *metric.*

The volume element of the metric, $\sqrt{\det g_i j}$, can be interpreted as how much volume a unit sphere corresponds to in the embedding space. It can be visualised accordingly (figure 5.6), where the size of the ellipsoid can be interpreted as the *size* of the metric tensor.

In Einstein notation, we can use the metric tensor for raising and lowering indices, $g^{kl}g_{lm} = \delta^m_k$, which is called contraction with the metric. Here, the covariant metric g^{kl} is the inverse of the contravariant g_{lm} but note, that this does not hold in general, i.e. $T^{-1}_{ij} \neq T^{ij}$.

We stress that the metric tensor is the foundation of Riemannian geometry. One of the reasons for its importance is that it encompasses the manifold; knowing the metric is knowing the manifold. To highlight its importance, we take a step back: High-dimensional data is represented in Euclidean space, $\mathbb{R}^D$ and the manifold hypothesis states that there exists a low-dimensional structure that captures data, i.e. a manifold from which data are assumed sampled. The Euclidean space is an *extrinsic* representation in which observations are not inherently constrained to the manifold. Revisiting the example of Earth from chapter 1, Earth exists in three spatial dimensions but world maps (its representations) are generally two-dimensional. Mapping a sphere to a plane induces an unavoidable distortion but this is due to the *extrinsic* description. Instead, to avoid this distortion of e.g. lengths, we can use the metric which leads to *intrinsic* representations.

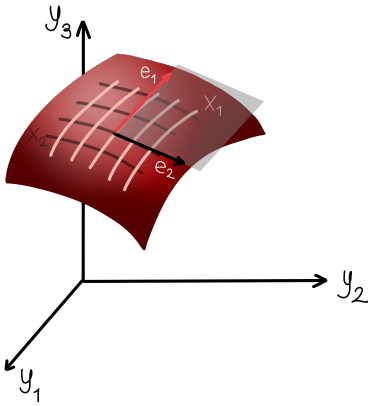


Figure 5.7: A manifold (red) described both with extrinsic coordinates (y_1, y_2, y_3) and intrinsic coordinates x_1, x_2 at a given point in terms of basis vectors e_1, e_2.

Intrinsic representations are convenient because they give rise to invariant quantities or simply invariances (quantities that do not change under transformation). We define *intrinsic* quantities as quantities that can be expressed solely in terms of the metric. This allows for an invariant characterisation of the manifold independent of the embedding space. In the context of machine learning, intrinsic quantities are invariant under reparametrisations of the latent space, (see section 5.4.

The arc lengths on manifolds are an example of such an invariant. Later, we discuss how to compute arc lengths or geodesics (section 5.3) but first, we consider a connection which connects tangent spaces. As the manifold (often) curves, the tangent plane and the metric tensor vary smoothly. We have not yet addressed how they change smoothly: The tangent spaces change via the Levi-Civita connection which connects the tangent spaces through the covariant derivative.

5.2.3 The Levi-Civita Connection

This section introduces the connection that ensures smoothly varying tangent spaces (figure 5.8). The fundamental theorem of Riemannian Geometry [93] ensures that there exists a unique connection, the Levi-Civita connection, for which the metric's torsion is zero. The torsion, $T(X, Y)$, is given by

$$T(X, Y) = \nabla_X Y - \nabla_Y X - [X, Y],$$

where X, Y are vector fields and $[X, Y]$ is the Lie bracket.

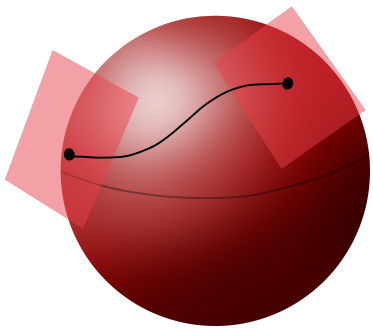


Figure 5.8: An affine connection can be viewed as a way of smoothly connecting tangent spaces.

Definition 5.4 (The Levi-Civita Connection) *An affine connection ∇ is called a Levi-Civita connection if*

- *It preserves the metric, i.e.* $\nabla g = 0$.
- *It is torsion-free, i.e.* $T(X, Y) = 0$.

The first condition is the metric compatability *condition and the second is called the* symmetry *condition.*

In our case, the Levi-Civita connection is a covariant derivative which describes the smooth change in tangent spaces. To understand how this works, we first consider the Euclidean setting, where the basis vectors do not change when differentiating and hence we usually do not write explicitly. This is not true on Riemannian manifolds where we explicitly have to handle the change in bases. The derivative of V contains both the derivatives of the components and the derivative of the basis,

$$\frac{\partial V}{\partial x^k} = \frac{\partial}{\partial x^k}(V^i e_i) = \frac{\partial V^i}{\partial x^k} e_i + V^i \frac{\partial e_i}{\partial x^k}. \tag{5.1}$$

The term covariant derivative is often used for the Levi-Civita connection. The components of this connection with respect to a system of local coordinates are called Christoffel symbols. In the local Euclidean basis, we define the Christoffel symbol as the magnitude of the change in the basis vector.

CHRISTOFFEL SYMBOLS
Euclidean basis

$$\frac{\partial e_i}{\partial x^k} = \Gamma^l_{ik} e_l \tag{5.2}$$

The Γ^l_{ik} gives the lth component of $\frac{\partial \bar{e}_i}{\partial x^k}$. i is the index of the basis vector that is being differentiated, k is the coordinate (or latent dimension), and this derivative gives the lth component of the resulting vector. It can be shown that if the metric is torsion-free, then the Christoffel symbols are symmetric in the two lower indices, $\Gamma^l_{ik} = \Gamma^l_{ki}$.

On a technical note, this is the Christoffel symbol of the second kind– which is not a tensor. The Christoffel symbol of the first kind [90] exists but this is not in the scope of this thesis. In the following, we refer to the Christoffel symbol of the second kind simply as the Christoffel symbol.

Revisiting the derivative of the vector, $V = V^i e_i$,

$$\begin{aligned} \frac{\partial V}{\partial x^k} &= \frac{\partial V^i}{\partial x^k} e_i + V^i \frac{\partial e_i}{\partial x^k} && (5.3) \\ &= \frac{\partial V^l}{\partial x^k} \cdot e_l + V^i \Gamma^l_{ik} \cdot e_l && (5.4) \\ &= (\partial_k V^l + \Gamma^l_{ik} V^i) \cdot e_l && (5.5) \end{aligned}$$

where the parenthesis is the covariant derivative, ∇, lets us define

$$\nabla_k V^i = \frac{\partial V^i}{\partial x^k} + \Gamma^i_{kl} V^l. \tag{5.6}$$

From this, we see that the covariant derivative consists of the usual derivative (the first term) and a correction term (second term) which include the Christoffel symbols.

5.2.4 Christoffel symbols

We obtain an *intrinsic* description of the Christoffel symbols (and therefore the covariant derivative) by expressing it in terms of the metric and its inverse, g^{il}. We defined the Christoffel symbols in the Euclidean basis to motivate the covariant derivative, but we can generalise this to Riemannian manifolds: By using the fact that the metric is torsion-free, we can express the Christoffel symbol intrinsically in terms of the metric [90]. Take the outer product the definition of the Christoffel symbols (equation 6.10) with a basis vector, e_l,

$$\Gamma^k_{ij} \boldsymbol{e}_k \cdot \boldsymbol{e}_l = (\partial_j \boldsymbol{e}_i) \cdot \boldsymbol{e}_l \tag{5.7}$$

Using the definition of the metric, $g_{ij} = \boldsymbol{e}_i \cdot \boldsymbol{e}_j$, we get

$$\begin{aligned} \Gamma^k_{ij} g_{kl} &= \partial_j (\boldsymbol{e}_i \cdot \boldsymbol{e}_l) - (\partial_j \boldsymbol{e}_l) \cdot \boldsymbol{e}_i && (5.8)\\ &= \partial_j g_{il} - \Gamma^k_{lj} \boldsymbol{e}_k \cdot \boldsymbol{e}_i && (5.9)\\ &= \partial_j g_{il} - \Gamma^k_{lj} g_{ki} && (5.10) \end{aligned}$$

After rearranging we are left with

$$\partial_j g_{il} = \Gamma^k_{ij} g_{kl} + \Gamma^k_{lj} g_{ki} \tag{5.11}$$

Here, k is a dummy index (it is being summed over) and i, j, k are all free indices. We could have started with any permutation of the indices so cyclic permutation of the indices leads to two more equations.

$$\begin{aligned} \partial_l g_{ji} &= \Gamma^k_{jl} g_{ki} + \Gamma^k_{il} g_{kj} && (5.12)\\ \partial_i g_{lj} &= \Gamma^k_{li} g_{kj} + \Gamma^k_{ji} g_{kl} && (5.13) \end{aligned}$$

We can combine these equations in a convenient way: We add equation 5.11 and equation 5.13 and subtract equation 5.12.

$$\begin{aligned} &\partial_j g_{il} + \partial_i g_{lj} - \partial_l g_{ji} && (5.14)\\ &= \Gamma^k_{ij} g_{kl} + \Gamma^k_{lj} g_{ki} + \Gamma^k_{li} g_{kj} + \Gamma^k_{ji} g_{kl} - \Gamma^k_{jl} g_{ki} - \Gamma^k_{il} g_{kj} && (5.15)\\ &= 2\Gamma^k_{ij} g_{kl} && (5.16) \end{aligned}$$

In the last step, we have exploited the symmetry in the lower indices in the Christoffel symbol, $\Gamma^k_{\ ij} = \Gamma^k_{\ ji}$, to rearrange the indices, to collect and cancel terms. To obtain an expression for the Christoffel symbol, we use the definition of the metric $g^{ij}g_{jk} = \delta^i_k$. We get

$$
\begin{aligned}
2\Gamma^k_{\ ij} g_{kl} g^{ml} &= g^{ml}(\partial_j g_{il} + \partial_i g_{lj} - \partial_l g_{ji}) && (5.17)\\
\Gamma^k_{\ ij}\delta^m_k &= \frac{1}{2} g^{ml}(\partial_j g_{il} + \partial_i g_{lj} - \partial_l g_{ji}) && (5.18)\\
\Gamma^m_{\ ij} &= \frac{1}{2} g^{ml}(\partial_j g_{il} + \partial_i g_{lj} - \partial_l g_{ji}) && (5.19)
\end{aligned}
$$

and we arrived at an intrinsic expression for the Christoffel symbols as they are expressed in terms of the metric tensor and its inverse. We will use this definition of the Christoffel symbol when we derive the geodesic equation (section 5.3).

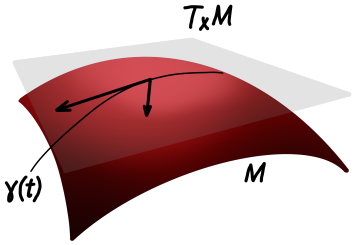


Figure 5.9: Illustration of the tangent space (grey) as a Euclidean space tangential to the manifold (red) and $\gamma(t)$ is a parametrised curve.

5.3 GEODESICS

We have defined Riemannian manifolds and this section introduces geodesics which allows for computing lengths and energies of curves and derives the geodesic equation which provides a mapping from the tangent space to the latent space. A geodesic is the notion of straight lines generalised to non-Euclidean spaces; the shortest path or the straightest path, locally.

A distance on a manifold follows a curve. All geodesics are curves but not all curves are geodesics.

Definition 5.5 (Curve) *A parametrised curve in $\mathbb{R}^D$ is a map, $\gamma(t) : \mathbb{R} \to \mathbb{R}^D$ for some $t \in [\alpha, \beta]$ with $\alpha = 0$ and $\beta = 1$.*

In metric spaces, a geodesic is a curve that minimises the length locally. Computing lengths (or distances) on a manifold requires a distance measure that takes curvature into account.

Definition 5.6 (Geodesic) *A curve $\gamma(t)$ on a manifold M is a geodesic if $\ddot{\gamma}$ is zero or orthogonal to the tangent space $\mathcal{T}_{\gamma(t)}$ at the point $\gamma(t)$ for all t and dot denotes differentiation with respect to t.*

Geodesics can be found by solving the geodesic equation,

GEODESIC EQUATION

$$\ddot{\gamma} + \Gamma^i_{\ jk}\dot{\gamma}^j\dot{\gamma}^k = 0 \qquad (5.20)$$

which is a second-order, ordinary differential equation. Appendix E contains a derivation of the geodesic equation from the Euler-Lagrange equations.

With the geodesic equation, we can map from the tangent space to the manifold. This is an initial value problem which can be solved with an initial tangent vector (figure 5.10). This is also called a shooting geodesic. The logarithmic map is the inverse map and is often defined as that; the inverse.

In the introduction (chapter 1), we motivated meaningful distances with an example of flying around the globe. The Earth is approximately a sphere and on spheres, geodesics are great circles of which there are infinitely many on a 2-sphere. The plane follows a great circle and it follows the great circles that have the shortest length. This makes it a meaningful distance.

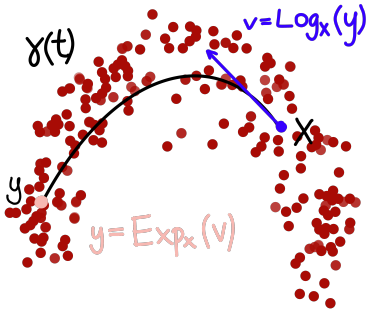


Figure 5.10: Illustration of the exponential map which corresponds to a shooting geodesic and of log map which corresponds to a map to tangent space

5.3.1 Length of a Curve

The example above links meaningful distances and lengths of geodesics so we must be able to compute the lengths of geodesics. First, we briefly discuss lengths of curves in general and then we relate to manifold learning.

We motivate the arc length of the curve γ in figure 5.11 starting at the point $\gamma(\alpha)$. The distance between two points, $\gamma(t')$ and $\gamma(t' + dt)$ is $\|\gamma(t' + \delta t) - \gamma(t')\|_2$ and $\|\dot{\gamma}\|_2$ for $\delta t \to 0$. The total length of the curve is given by the sum of all the length elements,

$$\text{Length}[\gamma] = \int_{t_0}^{t_1} \|\dot{\gamma}(t)\|_2 \, \mathrm{d}t.$$

As mentioned in the introduction, we choose to measure the distance between latent points on the manifold embedded in observation space: We define the meaningful length of a curve Length[γ] (where γ lives in the latent space) as the length of γ mapped to the manifold, Length[$f(\gamma)$] where f is the map, $f : \mathbb{R}^Q \to \mathbb{R}^D$.

The length of the curve measured in observation space is given by

$$\text{Length}[f(\gamma)] = \int_0^1 \left\| \frac{\partial f(\gamma)}{\partial t} \right\|_2 \mathrm{d}t = \int_0^1 \left\| \mathbf{J} \frac{\partial \gamma}{\partial t} \right\|_2 \mathrm{d}t \tag{5.21}$$

assuming that $t \in [0, 1]$ and where

$$\mathbf{J} = J_{ij} = \frac{\partial f_i}{\partial x_j}$$

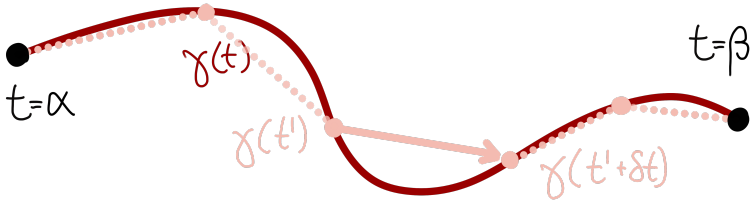


Figure 5.11: Illustration of a discretised curve where the sum of the discrete, Euclidean length elements approximates the total length of the curve. The arrow indicates the estimated tangent vector at $\gamma(t')$ which becomes more accurate as $\delta t \to 0$.

is the Jacobian of the map, f.

5.3.2 Energy of a Curve

In practice, we are searching for the shortest geodesic and minimising the length of a curve does not lead to unique geodesics and computationally, the norm can be challenging to optimise [4]. The energy of a curve provides an alternative. Similarly to the length of a curve (see section 5.3.1), the energy of a curve is defined as

ENERGY OF A CURVE

$$\text{Energy}(f(\gamma)) = \int_0^1 \left\| \frac{\partial f(\gamma(t))}{\partial t} \right\|_2^2 \mathrm{d}t = \int_0^1 \left\| \mathbf{J} \frac{\partial \gamma(t)}{\partial t} \right\|_2^2 \mathrm{d}t \tag{5.22}$$

Minimizing the length of a curve corresponds to minimizing the energy of a curve [4] (using the Cauchy-Schwartz inequality).

$$\underset{\gamma}{\text{argmin}} \int \|\dot{\gamma}(t)\|_2 \, \mathrm{d}t = \underset{\gamma}{\text{argmin}} \int \|\dot{\gamma}(t)\|_2^2 \, \mathrm{d}t$$

We can now find geodesics by minimising the energy of the curve.

5.4 MEANINGFUL DISTANCES

Finally, we can discuss meaningful distances–or *more* meaningful distances to be precise.

Consider a latent variable model, $f : \mathcal{X} \to \mathcal{Y}$, where f is a non-linear map from the latent space $\mathcal{X}$ to the observation space, $\mathcal{Y}$. The Euclidean distance between two latent points x_1 and x_2 is denoted $d(x_1, x_2)$. Now, consider a reparametrisation of this model, $\hat{f} : \mathcal{X} \to \mathcal{Y}$, where the latent space is reparametrised as $\hat{x} = \hat{f}(x)$ and the mapping

as $\hat{f}(\hat{x}) = f(g^{-1}(\hat{x}))$. The Euclidean distance in the reparametrised model between two latent points $\hat{x}_1 = g(x_1)$ and $\hat{x}_2 = g(x_2)$ is $d(\hat{x}_1, \hat{x}_2)$.

In general, these distances are not the same.

$$d(x_1, x_2) \neq d(\hat{x}_1, \hat{x}_2),$$

as g can be non-linear. This makes the Euclidean distance arbitrary.

Instead, we propose computing distances on a Riemannian manifold as a more meaningful measure of distance. The length of a curve on the manifold in the reparametrisation is

$$\begin{aligned}
\text{Length}[\hat{c}_t] &= \int \left\| \partial_t \hat{f}(c_t) \right\|_2 \mathrm{d}t && (5.23)\\
&= \int \left\| \partial_t f(g^{-1}(\hat{c}_t)) \right\|_2 \mathrm{d}t && (5.24)\\
&= \int \left\| \partial_t f(g^{-1}(g(c_t))) \right\|_2 \mathrm{d}t && (5.25)\\
&= \int \left\| \partial_t f(c_t) \right\|_2 \mathrm{d}t && (5.26)\\
&= \text{Length}[c_t] && (5.27)
\end{aligned}$$

This shows that the lengths of the reparametrised curve are identical to the length of the original curve, meaning that the distance is invariant under reparametrisation of the latent space. For example, two well-trained models learnt similar, yet different latent representations and the distances we measure on the manifold will be identical. We argue that measuring the distance in observation space is more meaningful as this is invariant to reparametrisations of the latent representation.

STOCHASTIC GEOMETRY 6

> *Geometry is the science of correct reasoning on incorrect figures.*
>
> —George Polya

This chapter contains mainly background on stochastic manifolds in GPLVMs. We contribute a correction to the derivation of apprximate Christoffel symbols originally done by Adams [3]. This correction was done jointly with Alison Pouplin.

At this point, we have seen that four visualisation models fail to preserve topology at an alarming rate which discouraged learning representations that to a greater extent reflect the geometry of data. We deep-dived into GPLVM. and developed a candidate model for learning such representations. As we have already introduced elementary Riemannian geometry (chapter 5), we are now almost ready to join geometry and GPLVMs (next chapter) but first, we introduce deterministic approximations of stochastic geometric objects that allow for using Riemannian geometry. This chapter develops the stochastic concepts needed to build a Riemannian GPLVM.

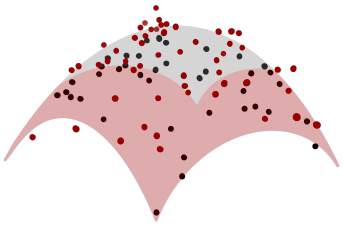


Figure 6.1: An illustration of data distributed around a mean manifold.

To understand why we take a stochastic approach, consider the manifold hypothesis which states that data are sampled from an underlying manifold. This means that data lie near the manifold as illustrated by figure 6.1. This implies that *off* the manifold is where there is no or little data. Topological data analysis (chapter 2) informs how to learn topological features, e.g. holes in the case of the statistical circle (figure 6.2). For a model to respect geometry, distances should be measured along the circle, and we saw how to do that using geodesics in chapter 5. So geodesics should stay near the manifold and thus near data.

Figure 6.2: The statistical circle.

As a motivating example, figure 6.3 shows geodesics in three learnt manifolds–two deterministic and a probabilistic. Only on the probabilistic manifold do the geodesics stay close to data. Hauberg [11] argues that it is key to estimate the manifold away from data. Deterministic models generally estimate the manifold well near data but it is more challenging away from data. This indicates that a probabilistic model is a good choice because we can explicitly control the behaviour away from data through the prior. This is one of the main motivations for considering geometry in probabilistic models: Uncertainty can play the role of topology [11, 12].

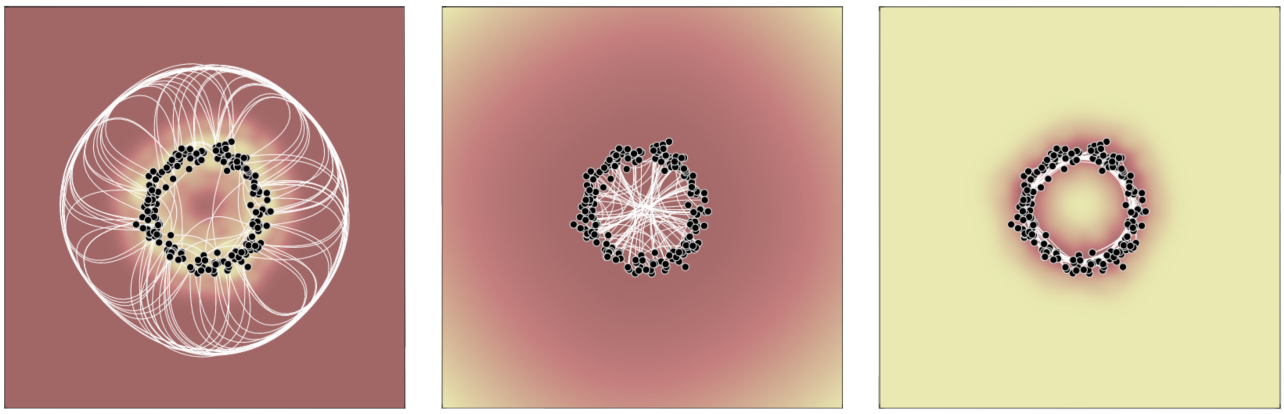


Figure 6.3: The figure illustrates geodesics on three different learnt manifolds. From the left, the figure shows a set of observations in black, geodesics in a Gaussian kernel ridge regression, geodesics in a Gaussian+linear, and geodesics in Gaussian process regression. The coloured backgrounds show the estimated uncertainty. The figure is from the paper *Only Bayes should learn a manifold* by Hauberg [11], figure 3.

Tosi et al. [10] were the first to consider geometry in the GPLVM (a probabilistic models). They questioned the Euclidean distance measure as the natural choice and developed a pull-back metric that reflects the data. From a manifold learning perspective, the GPLVM describes a stochastic manifold: The mean of the GP describes a deterministic manifold, the covariance provides uncertainty on this, and a single sample also corresponds to a deterministic manifold.

When the space itself is stochastic, all geometric objects are also stochastic. E.g., distances are computed between stochastic points which are also stochastic. In practice, we do not have the mathematical tools to handle stochastic manifolds but we might exploit the great tools for deterministic manifolds (chapter 5) by finding suitable deterministic approximations of stochastic manifolds.

6.1 STOCHASTIC MANIFOLDS

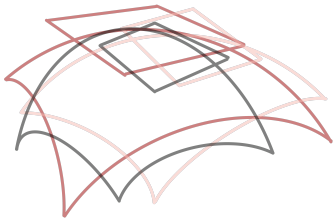


Figure 6.4: Illustration of three samples of tangent spaces.

This section develops the geometric foundation for treating stochastic manifolds as Riemannian manifolds. As we saw in the previous chapter, a Riemannian manifold requires a Riemannian metric defined on a tangent space.

This section derives the stochastic tangent space following Tosi et al. [10]. We consider the Jacobian (which we looked at in the deterministic setting in section 5.2.1) for GPLVM. In accordance with section 5.2.1, we define the tangent space as the Jacobian, J, of a Gaussian process,

$f(\cdot) \sim \mathcal{GP}(\mu(\cdot), k(\cdot,\cdot))$, with an element of the Jacobian given as

$$J_{ij} = \partial_j f_i = \frac{\partial f_i}{\partial x^j}$$

where we differentiate the ith GP with respect to the jth latent dimension. Both f and its derivative are Gaussian processes. The differentiability of the GP requires choosing a sufficiently differentiable kernel. The resulting Jacobian $J \in \mathbb{R}^{Q\times D}$ and the columns of J are independent and normally distributed. The prior distribution on J is the derivative of the Gaussian process, and we assume it factorises over the dimensions,

$$p(J|X, f, \sigma) = \prod_{d=1}^{D} \mathcal{N}(0, \partial^2 k(\cdot,\cdot)). \tag{6.1}$$

To evaluate the metric at any point on a manifold, we require the predictive distribution of J. The joint distribution of Y and J

$$\begin{bmatrix} Y \\ J \end{bmatrix} |X, x_* \sim \mathcal{N}\left(0, \begin{bmatrix} \text{cov}(Y,Y) & \text{cov}(Y,J) \\ \text{cov}(J,Y) & \text{cov}(J,J) \end{bmatrix}\right) \tag{6.2}$$

where the Jacobian is evaluated in x_* which is a point on the manifold, $x_* \in \mathcal{M}$.

The covariance of the joint distribution consists of four block matrices. The covariance of training data, $\text{cov}(Y,Y)$ is by definition $\text{cov}(Y,Y) = k(X,X)+\sigma^2\mathbb{I}_N$ and we denote is K_{XX} (absorbing the noise in K). We also know that $\text{cov}(Y,J) = \text{cov}(J,Y)^\top$ which means we have to consider two terms: $\text{cov}(Y,J)$ and $\text{cov}(J,J)$. Recall, the generative model, $y = f(X) + \epsilon$, which lets us rewrite,

$$\text{cov}\left(f(X) + \epsilon, \frac{\partial f(X)}{\partial x}\right) = \text{cov}\left(f(X), \frac{\partial f(X)}{\partial x}\right) + \text{cov}\left(\epsilon, \frac{\partial f(X)}{\partial x}\right).$$

The noise corruption ϵ is a Gaussian, random variable with zero mean and variances σ^2. The covariance can be computed by expectations,

$$\text{cov}\left(\epsilon, \frac{\partial f(X)}{\partial x}\right) = \mathbb{E}[\epsilon f(X)] - \mathbb{E}[\epsilon]\mathbb{E}[f(X)] = \mathbb{E}[\epsilon]\mathbb{E}[f(X)] = 0$$

because ϵ should be independent of Y and therefore the term is zero.

The key to the covariance between $f(X)$ and its derivative is equation 9.1 in Rasmussen and Williams [8]. It states that

$$\text{cov}\left(f_i, \frac{\partial f_j}{\partial x_{dj}}\right) = \frac{\partial k(x_i, x_j)}{\partial x_{dj}}, \qquad \text{cov}\left(\frac{\partial f_i}{\partial x_{di}}, \frac{\partial f_j}{\partial x_{ej}}\right) = \frac{\partial^2 k(x_i, x_j)}{\partial x_{di}\partial x_{ej}}$$

With the conditional Gaussian identities (chapter C), we obtain the predictive distribution of the Jacobian from the joint distribution (equation 6.2),

$$p(J|Y, X, x^*) = \prod_{d=1}^{D} \mathcal{N}(\mu_{J_{d,:}}, \Sigma_J),$$

where

$$\mu_{J_{d,:}} = \partial k_{x*}^T K_{xx}^{-1} Y_{:,d} \tag{6.3}$$
$$\Sigma_J = \partial^2 k_{**} - \partial k_{x*}^T K_{xx}^{-1} \partial k_{x*}, \tag{6.4}$$

The components of the differentials are given by

$$(\partial k_{x*})_{nq} = \frac{\partial k(x_n, x_*)}{\partial x^q}, \quad n = 1, \ldots, N \text{ and } q = 1, \ldots, Q \tag{6.5}$$
$$(\partial^2 k_{**})_{rq} = \frac{\partial^2 k(x_*, x_*)}{\partial x^q \partial x^r} \quad q, r = 1, \ldots, Q \tag{6.6}$$

To summarise, the Jacobian gives the tangent space, and a predictive distribution on the Jacobian leads to stochastic tangent space. It is straightforward to extend this to the Bayesian GPLVM, which Scannell [60] does.

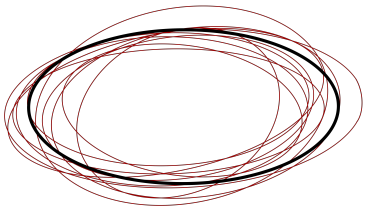


Figure 6.5: Illustration of stochastic metric. The black ellipsis represents the mean metric, and the red ellipses are draws of the metric.

6.2 STOCHASTIC METRICS

Chapter 5 introduced the metric as a smoothly changing inner product of basis vectors that is positive and semi-definite. Here, the basis is stochastic and given by J, which gives rise to a natural Riemannian metric

$$G = J^\top J \tag{6.7}$$

This metric is defined in the latent space, i.e. $G \in \mathbb{R}^{Q \times Q}$, and is also referred to as a pull-back metric.

The prior on J (equation 6.1) is a Gaussian process, and hence the metric is a Wishart process [94] as the product of two Gaussian processes is a Wishart process [95].

STOCHASTIC METRIC

$$G \sim \mathcal{W}(D, \Sigma_J, \mathbb{E}[J^\top]\mathbb{E}[J]) \tag{6.8}$$

This is a distribution of random positive and semi-definite matrices where D is the degrees of freedom.

The stochastic metric is not operational, so instead, we search for a deterministic approximation of the stochastic metric. The expected metric and the expectation of a Wishart distribution is [10, 95, 96]

THE EXPECTED METRIC

$$\mathbb{E}[G] = \mathbb{E}[J^T J] = \mathbb{E}[J]^T \mathbb{E}[J] + D\Sigma_J, \tag{6.9}$$

This yields a deterministic $Q \times Q$ matrix. With a deterministic approximation, we are all set, and we can use Riemannian geometry (chapter 5) directly.

This approximation includes the covariance of J directly, which means that the expected metric is larger for larger covariances of the Jacobian. Later, we discuss how this elongates curves in uncertain (high covariance) areas of the manifold and, eventually, how it forces geodesics to avoid uncertainty (section 6.4). This approximation applies to both the GPLVM and the Bayesian GPLVM as it only assumes normality of the Jacobian.

6.3 STOCHASTIC CHRISTOFFEL SYMBOLS

The Christoffel symbols play a crucial part in Riemannian geometry (chapter 5), and ideally, we would like a distribution over the Christoffel symbols. Extending the notation of derivatives $\frac{\partial g_{jk}}{\partial x^i} = \partial_i g_{jk} = g_{jk,i}$ gives the Christoffel symbols,

$$\begin{aligned} \Gamma^i_{jk}\frac{1}{2} &= \frac{1}{2} g^{ml}(\partial_j g_{il} + \partial_i g_{lj} - \partial_l g_{ji}) \\ &= \frac{1}{2} g^{ml}(g_{il,j} + g_{lj,i} - g_{ji,l}). \end{aligned} \tag{6.10}$$

The metric $g_{ij} \sim \mathcal{W}$ and $g^{ml} \sim \mathcal{W}^{-1}$ [97] so each of the terms is a product of a Wishart and an inverse Wishart, which is stochastic and therefore impractical. Instead, we may wish to approximate the Christoffel symbols using the deterministic approximation of the metric: The expectation of the Christoffel symbol should be easily obtained by

$$\mathbb{E}[\Gamma^i_{jk}] = \frac{1}{2}\mathbb{E}[g^{ml}\partial_j g_{il}] + \mathbb{E}[g^{ml}\partial_i g_{lj}] - \mathbb{E}[g^{ml}\partial_l g_{ji})]. \tag{6.11}$$

Each of the terms is $\mathbb{E}[\mathcal{W}^{-1}\mathcal{W}]$, and we are unaware of any analytical results for its expectation. This means that a deterministic approximation using the expectation of the Christoffel symbols is currently out of reach.

In line with our goal–an implementable expression for the Christoffel symbols–this section approximates the stochastic Christoffel symbols deterministically as the Christoffel symbol evaluated in the deterministic approximation of the metric. For now we ignore the inverse of the metric and focues on the metric derivatives. The expectations of the components of the metric, $G = J^\top J$, follows a Wishart distribution,

$$\begin{aligned}\mathbb{E}[G]_{ij} &= \mathbb{E}[J^\top J]_{ij} = [\mathbb{E}[J]^\top \mathbb{E}[J]]_{ij} + D\Sigma_{ij} && (6.12)\\ &= \mathbb{E}[J]_{mi}\mathbb{E}[J]_{mj} + D\Sigma_{ij} && (6.13)\\ &\equiv \mathbb{E}[g_{ij}] && (6.14)\end{aligned}$$

where $\mathbb{E}[J]_{ij}$ and Σ_{ij} are mean and covariance of the predictive Jacobian (equation 6.3) from section 6.1.

To ease the derivations, we decompose the derivatives of the expected metrics (equation 6.10) into mean and covariance contributions from the expected metric,

$$\Gamma^i_{jk}\Big|_{\mathbb{E}[G]} = \Gamma^i_{jk}\Big|_{\mathbb{E}[J]^\top\mathbb{E}[J]} + \Gamma^i_{jk}\Big|_{D\Sigma_J}. \tag{6.15}$$

For convenience, we derive these two terms separately,

$$\tilde{\Gamma}^i_{jk} = \Gamma^i_{jk}\Big|_{\mathbb{E}[J]^\top\mathbb{E}[J]} \quad \text{and} \quad \hat{\Gamma}^i_{jk} = \Gamma^i_{jk}\Big|_{D\Sigma_J}, \tag{6.16}$$

where $J \sim \mathcal{N}(\mu_J, \Sigma_J)$ so $\mathbb{E}[J] = \mu_J$ where μ_J and Σ_J are given by equation 6.3. We derive them individually and add them to obtain an expression for the expected Christoffel symbol.

We define the first term of the expectation in terms of a mean map, ϕ,

$$\mathbb{E}[J]^\top\mathbb{E}[J] = \frac{\partial\phi}{\partial x^i}\cdot\frac{\partial\phi}{\partial x^j} \equiv \phi_{,i}\cdot\phi_{,i} \tag{6.17}$$

where the derivative of ϕ is μ_J, and the index of the Gaussian process (the dimensionality) is summed over. Note that $\mathbb{E}[g^{il}] = \mathbb{E}[g_{il}]^{-1}$, which is straightforward to evaluate so we do not touch the inverse of the metric for now. With this, we consider,

$$\begin{aligned}\tilde{\Gamma}^i_{jk} &= \frac{1}{2}g^{il}\left(\frac{\partial\phi_{,l}\cdot\phi_{,j}}{\partial x^k} + \frac{\partial\phi_{,l}\cdot\phi_{,k}}{\partial x^j} - \frac{\partial\phi_{,j}\cdot\phi_{,k}}{\partial x^l}\right)\\ &= \frac{1}{2}g^{il}\left(\frac{\partial\phi_{,l}}{\partial x^k}\cdot\phi_{,j} + \phi_l\cdot\frac{\partial\phi_{,j}}{\partial x^k} + \frac{\partial\phi_{,l}}{\partial x^j}\cdot\phi_{,k} + \phi_{,l}\cdot\frac{\partial\phi_{,k}}{\partial x^j} - \frac{\partial\phi_{,j}}{\partial x^l}\cdot\phi_{,k} - \phi_{,j}\cdot\frac{\partial\phi_{,k}}{\partial x^l}\right)\\ &= \frac{1}{2}g^{il}\left(\cancel{\phi_{,lk}\cdot\phi_{,j}} + \phi_{,l}\cdot\phi_{,jk} + \cancel{\phi_{,lj}\cdot\phi_{,k}} + \phi_{,l}\cdot\phi_{,kj} - \cancel{\phi_{,jl}\cdot\phi_{,k}} - \cancel{\phi_{,j}\cdot\phi_{,kl}}\right)\\ &= g^{il}(\phi_{,l}\cdot\phi_{,jk}) && (6.18)\end{aligned}$$

where the cancelling happens due to symmetry in the indices; the order of differentiation does not matter.

Similarly, Σ_J is given by equation 6.3 and we define $[\Sigma_J]_{ij} = \sigma_{ij}$,

$$\begin{aligned}\sigma_{ij} &= \frac{\partial^2 k_{**}}{\partial x^i \partial x^j} - \frac{\partial k_*^\top}{\partial x^i} K_{XX}^{-1} \frac{\partial k_*}{\partial x^j} && (6.19)\\ &= \partial^2_{ij} k_{**} - \partial_i k_*^\top K_{XX}^{-1} \partial_j k_* && (6.20)\end{aligned}$$

where $k_* = k(x, X)$ and X is the training data (i.e. fixed). We can write the variance contribution of the Christoffel symbols as

$$\begin{aligned}\hat{\Gamma}^i_{jk} &= \Gamma^i_{jk}\Big|_{D\Sigma_J} && (6.21)\\ &= \frac{D}{2} g^{il} \left(\frac{\partial \sigma_{lj}}{\partial x^k} + \frac{\partial \sigma_{lk}}{\partial x^j} - \frac{\partial \sigma_{jk}}{\partial x^l} \right) && (6.22)\end{aligned}$$

We derive this in small steps, starting with the derivatives of σ_{ij},

$$\begin{aligned}\frac{\partial \sigma_{ij}}{\partial x^k} &= \frac{\partial^3 k_{**}}{\partial x^k \partial x^i \partial x^j} - \frac{\partial^2 k_*^\top}{\partial x^k \partial x^i} K_{XX}^{-1} \frac{\partial k_*}{\partial x^j} - \frac{\partial k_*^\top}{\partial x^i} K_{XX}^{-1} \frac{\partial^2 k_*}{\partial x^k \partial x^j} && (6.23)\\ \sigma_{ij,k} &= \partial^3_{kij} k_{**} - \partial^2_{ki} k_*^\top K_{XX}^{-1} \partial_j k_* - \partial_i k_*^\top K_{XX}^{-1} \partial^2_{kj} k_*. && (6.24)\end{aligned}$$

Before writing up the terms required by equation 6.21, we consider the last term separately,

$$\begin{aligned}\partial_i k_*^\top K_{XX}^{-1} \partial^2_{kj} k_* &= (\partial_i k_*^\top K_{XX}^{-1} \partial^2_{kj} k_*)^\top && (6.25)\\ &= (\partial^2_{kj} k_*)^\top (K_{XX}^{-1})^\top (\partial_i k_*^\top)^\top && (6.26)\\ &= \partial^2_{kj} k_*^\top K_{XX}^{-1} \partial_i k_*, && (6.27)\end{aligned}$$

as all terms in equation 6.21 are scalars. This trick reduces the second-order terms to just one term. Now consider the three terms for the variance in equation 6.21.

$$\begin{aligned}\sigma_{lj,k} &= \partial^3_{klj} k_{**} - \partial^2_{kl} k_*^\top K_{XX}^{-1} \partial_j k_* - \partial_l k_*^\top K_{XX}^{-1} \partial^2_{kj} k_* && (6.28)\\ \sigma_{lk,j} &= \partial^3_{jkl} k_{**} - \partial^2_{jl} k_*^\top K_{XX}^{-1} \partial_k k_* - \partial_l k_*^\top K_{XX}^{-1} \partial^2_{jk} k_* && (6.29)\\ \sigma_{jk,l} &= \partial^3_{ljk} k_{**} - \partial^2_{lj} k_*^\top K_{XX}^{-1} \partial_k k_* - \partial_j k_*^\top K_{XX}^{-1} \partial^2_{lk} k_* && (6.30)\end{aligned}$$

The variance contribution of the Christoffel symbols is

$$\begin{aligned}\hat{\Gamma}^i_{jk} &= \frac{D}{2} g^{il} \left(\sigma_{lj,k} + \sigma_{lk,j} - \sigma_{jk,l} \right) && (6.31)\\ &= \frac{D}{2} g^{il} \left(\partial^3_{klj} k_{**} - 2 \partial^2_{kj} k_*^\top K_{XX}^{-1} \partial_l k_* \right). && (6.32)\end{aligned}$$

Collecting the contributions yields

$$\Gamma^i_{jk}\Big|_{\mathbb{E}[G]} = g^{il}\left(\phi_{,l}\cdot\phi_{,jk} + \frac{D}{2}\partial^3_{klj}k_{**} - D\partial^2_{kj}k_*^\top K_{XX}^{-1}\partial_l k_*\right), \tag{6.33}$$

where g^{il} is the stochastic inverse metric but this is readily available with its deterministic approximation. This result holds for a general, sufficiently differential kernel.

The Christoffel symbols are integral to the covariant derivative which allows for generalising calculus to manifolds. This approach enables deriving expressions for approximate geometric quantities that rely on higher-order derivatives, e.g. Riemann curvature tensor–a tensor field capturing the curvature of manifolds.

6.3.1 Approximate Christoffel Symbols for the EQ Kernel

In this thesis, we use the EQ kernel which is C^∞ so the derivatives are defined and we can compute them explicitly though they are somewhat tedious to deal with. The derivatives are derived in appendix E,

$$\begin{aligned}
k(x_a, x_b) &= \alpha\exp\left(-\frac{\gamma}{2}\|x_a - x_b\|^2\right) && (6.34)\\
\frac{\partial k(x_a,x_b)}{\partial x^j} &= -\gamma(x_a^j - x_b^j)k(x_a,x_b) && (6.35)\\
\frac{\partial^2 k(x_a,x_b)}{\partial x^i\partial x^j} &= \gamma\left(\gamma(x_a^i - x_b^i)(x_a^j - x_b^j) - \delta_{ij}\right)k(x_a,x_b) && (6.36)\\
\frac{\partial^3 k(x_a,x_b)}{\partial x^l\partial x^i\partial x^j} &= \gamma^2\left(\delta_{lj}(x_a^i - x_b^i) + \delta_{li}(x_a^j - x_b^j)\right) && (6.37)\\
&- \gamma^2\left[\gamma(x_a^j - x_b^j)(x_a^i - x_b^i) - \delta_{ij}\right](x_a^l - x_b^l)k(x_a,x_b),
\end{aligned}$$

where γ is the inverse lengthscale of the EQ kernel, $\gamma = l^{-1}$ (section 3.3).

This approximation of the Christoffel symbols is not the expected Christoffel symbols, rather it is evaluated in the expected metric which ignores the product of the inverse Wishart and Wishart distributions. It is unclear how good an approximation but has the benefit of being quick to evaluate in practice: The derivatives of the kernel can be computed analytically and the expected metric can be inverted quickly as it is $Q \times Q$.

The derivation in the section follows the general idea introduced by Adams [3] who also provides an expression for the Christoffel

symbols evaluated in the expected metric. In principle, we could write our expression explicitly (by inserting the kernel derivatives into equation 6.33) but we settle for noting that our final expression deviates from that of Adams' [3] due to differences in the kernel derivatives.

6.4 STOCHASTIC GEODESICS

It is not straightforward to define stochastic geodesics and this section raises more questions than it answers. It was intended as the foundation for computing geodesics on stochastic manifolds; ideally, it would have answered what a stochastic geodesic is. In the deterministic case, a geodesic curve minimises the energy, which can be found by solving the geodesic equation (equation 5.20). In the stochastic case, we can interpret this in several ways. Here, we discuss a few options with varying degrees of stochasticity and with implementation in mind.

Li and Mukherjee [98] summarise the discussion of stochastic processes on Riemannian manifolds as *"there are basically two approaches to construct or model random processes on manifolds: one can randomise paths on the manifold or randomise the geometry that the paths follow."* and they pursue the path of random geometry. They study a random connection and random covariant derivatives, which lets them define stochastic geodesics rigorously as stochastic differential equations. This is indeed interesting, but we will not follow this path.

Our goal is to compute distance in the latent space that respects the geometry of the observation space. On a stochastic manifold, the space itself is stochastic, which means that all points along the geodesic are stochastic. The space itself and the mapping to the observation space contribute to the stochasticity, and ideally, we want to consider both.

In the previous chapter, we derived the geodesic equation which defines geodesics, i.e. the solution to the second-order ordinary differential equation is a geodesic. The most obvious choice is to consider a stochastic version of the geodesic equation to define a stochastic geodesic. We will not pursue this direction this would not be computationally too slow for our purpose.

We could sample the stochastic manifold and compute geodesics as deterministic curves. The metrics must be sampled jointly to ensure a smoothly varying manifold. Generally, a distribution can be approximated better than using one single sample. If using a deterministic approximation of the manifold is desirable, there are other routes to consider.

One route is sequentially sampling the geodesics. We could assume a

deterministic point in the latent space (e.g., in the maximum likelihood GPLVM) and obtain a predictive distribution over the next length element but this is slow when computing numerous geodesics.

Bewsher et al. [99] provides the distribution of arc lengths in posterior Gaussian processes by approximating the Nakagami distribution [100], but this is not sufficient for our purpose. Chapter 5 defines geodesics as length-minimising curves, but under a random metric, this is almost surely not the case [101]. Instead, we search for a deterministic approximation, where geodesics are locally length-minimising curves or, equivalently, energy-minimising curves (section 5.3). Then geodesics are curves that minimise the expected length. We used the energy in the deterministic case because the optimum is much easier to find for the energy than for the length and they lead to the same optima. This is not the case in the stochastic case, where the energy has a covariance term that contributes to the expectation, so

$$\operatorname{argmin} \mathbb{E}[\int \|\dot{c}\| \, \mathrm{d}t] \neq \operatorname{argmin} \mathbb{E}[\int \|\dot{c}\|^2 \, \mathrm{d}t]$$

Here, we assume that the curves that minimise the expected energy approximate the curves that minimise the expected length well. This gives a deterministic curve that is stochastic when mapped to observation space. We will consider this notion of stochastic geodesics in the next chapter.

Why geodesics avoid uncertainty

At the beginning of this chapter, we stated that *topology can play the role of uncertainty* which is a different way of stating that geodesics avoid uncertainty. This metric gives rise to geodesics that follow high-density regions in latent space [10], so defining stochastic geodesics as above has the benefit of incorporating the uncertainty such that geodesics avoid uncertainty. If the distance between two points is Δ, the expected squared distance,

$$\mathbb{E}[\|\Delta\|^2] = \mathbb{E}[\|\Delta\|]^2 + \operatorname{var}[\Delta].$$

This indicates that the variance of the length directly contributes to the expected length. Thus the expected length is shorter where the uncertainty is smaller which repels the geodesics as they are locally length-minimising.

We can understand this in a different way: Away from data, the mean function of the GP falls back to the mean of the prior, which is a constant, but it is the same constant in all points away from data. This

means that the volume away from data maps to the same point in observation space, which does not contribute to the geodesic length. This is what Hauberg [11] refers to as teleports. Accounting for the variance gives a finite volume in observation space which does contributes to the length of the geodesics, effectively preventing teleports.

LEARNING A STOCHASTIC GEOMETRY

7

> *One geometry cannot be more true than another; it can only be more convenient.*
>
> —Henri Poincaré

This chapter contains previously unpublished material.

As we have already introduced elementary Riemannian geometry (chapter 5) and deterministic approximations of stochastic geometric objects that allow for using Riemannian geometry (chapter 6), we are finally ready to build a Riemannian GPLVM.

In his seminal work, Lawrence [32] comments that the solution for the latent variables in the GPLVM will naturally not be unique as it is subject to an arbitrary rotation (chapter 3). We may avoid this with the Riemannian Brownian motion, which is invariant to reparametrisations unlike the usual Gaussian prior. The Brownian motion is a flexible distribution as it adapts to the data using only two parameters. Adapting the prior to data is somewhat controversial in Bayesian statistics but not new in the literature [102, 103] and this approach allows for capturing the prior belief that data is not Euclidean and do not lie on a specific parametrised manifold.

In this section, we introduce a computational framework for working with random manifolds (section 7.1) which we use for geometry in a toy dataset (section 7.2) and in the SAS model (section 7.3). This works surprisingly well, encouraging a model that incorporates geometry directly rather than being post hoc (after training). We introduce the Riemannian Brownian motion (section 7.4) with the intent of using this as a prior in a GPLVM or SAS model (section 7.5), but unfortunately, we have not yet succeeded.

7.1 STOCHMAN AND THE BUDDIES

This section describes an operational approach to working with random manifolds required to train models that respect the geometry of data. First, we introduce a framework for working with random manifolds directly from data (StochMan – Stochastic Manifolds Made Easy, section 7.1), and then, we detail additional, custom implementations (the Buddies) that speed up geodesic computations for the GPLVM (section 7.1.2).

7.1.1 StochMan: Stochastic Manifolds Made Easy

StochMan[1][104] is a python package for learning random manifolds from noisy, finite data. It employs PyTorch's automatic differentiation [87] and provides an interface for constructing manifolds, curves and geodesics. Figure 7.1 shows examples of geodesics computed on a manifold learnt from MNIST [84].

1. Code on Github https://github.com/MachineLearningLifeScience/stochman/

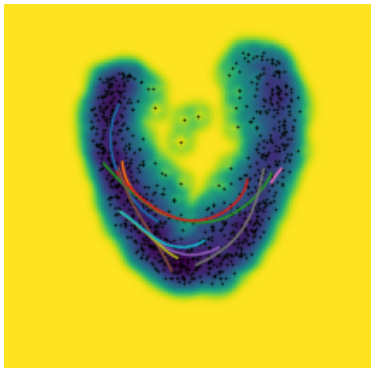


Figure 7.1: Black dots are latent points, the background is the log volume element of the metric (blue is smaller and yellow is larger), and the multicoloured curves are examples of geodesics computed on the manifold with StochMan. This example appears in the documentation of StochMan on Github

.

StochMan achieves this by assuming that a deterministic metric can approximate the stochastic metrics and that the expected energy-minimising curve approximates the expected length-minimising curve. The following paragraphs detail these assumptions.

First, StochMan requires the user to specify a deterministic metric, assuming that this is a suitable approximation of the stochastic metric. StochMan requires a deterministic metric as the foundation of its calculations, as the stochastic metric is not operational. For the GPLVM, StochMan implements the expectation of a Wishart distribution as a deterministic approximation as suggested in section 6.2.

Second, StochMan assumes that the curve c that minimises expected energy is also the curve that minimises expected length. In the deterministic case, the length l and energy ε of a curve will be minimised by the same curve (see section 5.3.1)

$$\operatorname*{argmin}_c \; l = \operatorname*{argmin}_c \; \varepsilon. \tag{7.1}$$

But this is not true in expectation in the stochastic setting,

$$\operatorname*{argmin}_c \; \mathbb{E}[l] \neq \operatorname*{argmin}_c \; \mathbb{E}[\varepsilon]. \tag{7.2}$$

In this work, we assume the expected energy-minimising curve approximates the expected length-minimising curve well.

$$\operatorname*{argmin}_c \; \mathbb{E}[l] \approx \operatorname*{argmin}_c \; \mathbb{E}[\varepsilon]. \tag{7.3}$$

Alternatively, geodesics can be computed straight-forwardly by solving the second-order ordinary differential equation in equation 5.20. Unfortunately, this is slow, so StochMan defines a geodesic as the curve c that minimises the expected energy

$$\underset{c}{\operatorname{argmin}} \ \mathbb{E}[\varepsilon] \tag{7.4}$$

GEODESIC
StochMan

StochMan uses several types of curves, but geodesics will be cubic splines for our purpose. These are initialised as Euclidean straight lines and optimised by minimising the expected energy to learn the parameters of the cubic spline.

7.1.2 The Buddies: Custom Implementations

Though Stochman provides a foundation for doing geometric computations, geometry in the GPLVM requires some custom functionalities. The Buddies include custom implementations of kernel derivatives to estimate Jacobians. This section also introduces cubic splines as an approximation to geodesics. This approximation uses the expected energy of a curve under the GPLVM expected metric. This is slow on the GPLVM, so the Buddies implements a discrete manifold to speed up computations.

Kernel Derivatives

In section 6.1, we showed that the predictive distribution of the Jacobian requires the derivatives of the kernel. We have implemented the analytic derivatives of the exponentiated quadratic kernel. We have also implemented the kernel derivatives using PyTorchs's automatic differentiation. This is slower but handles stationary and sufficiently differentiable kernels.

Expected Curve Energy

Starting from a discrete approximation of the curve energy for a curve on the Gaussian process manifold, this section details how this approximation can be implemented as a loss function for geodesic optimisation. After finding a discrete approximation of the curve energy, we derive the expectation and covariance of the line elements.

We introduced the energy of a curve ε in detail in section 5.3.2,

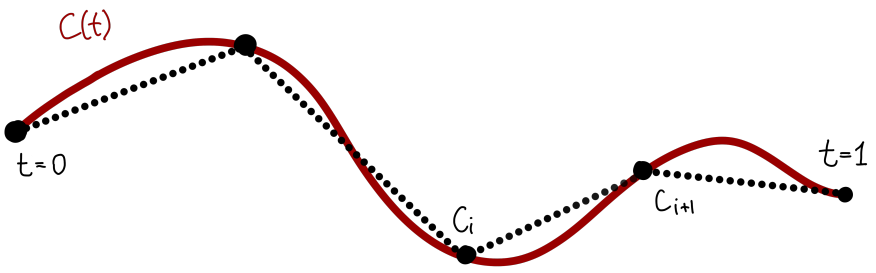


Figure 7.2: The figure shows a curve c_t (red) discretised in $N-1$ line elements, c_i, shown in dashed black lines. We refer to the endpoints of each line element with a subscript, e.g. c_{i+1} and c_i.

ENERGY OF CURVE

$$\varepsilon = \int_0^1 \|\partial_t f(c_t)\|_2^2 \, \mathrm{d}t, \tag{7.5}$$

where $t \in [0, 1]$ parametrises the continuous curve, c_t, and f is the map from the latent space to the data space, $f : \mathbb{R}^q \to \mathbb{R}^d$. Discretising the energy gives

$$\varepsilon = \sum_{i=1}^{N-1} \|f(c_{i+1}) - f(c_i)\|_2^2 \tag{7.6}$$

where i indexes the discrete curve element. This corresponds to dissecting the curve into N segments, computing the energy for each and summing them. The idea that the differential operator reduces to the difference assumes a Euclidean geometry which is a reasonable assumption in a local region where the space can be assumed Euclidean.

Taking the expectation of the curve energy, we obtain

$$\mathbb{E}[\varepsilon] = \sum_{i=1}^{N-1} \mathbb{E}\left[\|f(c_{i+1}) - f(c_i)\|_2^2\right] \tag{7.7}$$

and for brevity, we define $\Delta_i = f(c_{i+1}) - f(c_i)$. The expected energy is an inner product between two Gaussian random variables $x \in \mathbb{R}^D$, which follows the relation [51],

$$\mathbb{E}[x^T x] = \mathbb{E}[x]^T \mathbb{E}[x] + \mathrm{tr}(\mathrm{cov}[x]). \tag{7.8}$$

This gives the expected energy of a single line element Δ_i so summing over the $N-1$ expected energies gives the total expected energy

EXPECTED CURVE ENERGY

$$\mathbb{E}[\varepsilon_c] = \sum_{i=1}^{N-1} \mathbb{E}[\Delta_i]^T \mathbb{E}[\Delta_i] + \sum_{i=1}^{N-1} \mathrm{tr}(\mathrm{cov}[\Delta_i]) \tag{7.9}$$

This requires calculating the expectation of Δ_i and the covariance of the inner product of the Δ_is.

The expectation of Δ_i is straightforward. The mean of the distribution of Δ_i is given by the difference of the means of $f(c_{i+1})$ and $f(c_i)$,

$$\begin{aligned}\mathbb{E}[\Delta_i] &= \mathbb{E}[f(c_{i+1}) - f(c_i)] && (7.10)\\ &= \mathbb{E}[f(c_{i+1})] - \mathbb{E}[f(c_i)] && (7.11)\\ &= \mu_i - \mu_{i+1}. && (7.12)\end{aligned}$$

We consider the joint distribution to find the covariance of the two line segments. Both $f(c_i)$ and $f(c_{i+1})$ will follow a multivariate Gaussian distribution. In line with the GPLVM, we will assume that there is no covariance between output dimensions such that the joint distribution of two adjacent line segments factorises over dimensions.

$$\begin{bmatrix} f(c_i) \\ f(c_{i+1}) \end{bmatrix} = \prod_{d=1}^{D} \mathcal{N}\left(\begin{bmatrix} \mu_i \\ \mu_{i+1} \end{bmatrix}, \begin{bmatrix} \sigma_i^2 & \sigma_{i,i+1}^2 \\ \sigma_{i+1,i}^2 & \sigma_{i+1}^2 \end{bmatrix} \right) \tag{7.13}$$

Next, we exploit the Gaussian property of affine transformations (see section C) to obtain the distribution for $f(c_{i+1}) - f(c_i)$ by defining a transformation $A = [-1, 1]$.

$$\begin{aligned} f(c_{i+1}) - f(c_i) &= A \begin{bmatrix} f(c_i) \\ f(c_{i+1}) \end{bmatrix} && (7.14)\\ &= \prod_{d=1}^{D} \mathcal{N}\left(A \begin{bmatrix} \mu_i \\ \mu_{i+1} \end{bmatrix}, A \begin{bmatrix} \sigma_i^2 & \sigma_{i,i+1}^2 \\ \sigma_{i+1,i}^2 & \sigma_{i+1}^2 \end{bmatrix} A^\top \right) && (7.15)\end{aligned}$$

So writing up the new covariance is

$$\begin{aligned}\mathrm{cov}[\Delta_i] &= A\Sigma A^T = \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} \sigma_{i,i}^2 & \sigma_{i,i+1}^2 \\ \sigma_{i+1,i}^2 & \sigma_{i+1,i+1}^2 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} && (7.16)\\ &= \sigma_{i,i}^2 - \sigma_{i,i+1}^2 - \sigma_{i+1,i}^2 + \sigma_{i+1,i+1}^2 = 2(\sigma_{i,i}^2 - \sigma_{i,i+1}^2) && (7.17)\end{aligned}$$

In the last equality, we used that actually $f(c_i)$ and $f(c_{i+1})$ have the same variance so $\sigma_{i,i}^2 = \sigma_{i+1,i+1}^2$ and that the covariance matrix is symmetric. The assumption that the variance is the same as assumed in the original GPLVM paper by Lawrence [32]: The Gaussian processes have the same covariance in all dimensions but different mean functions. This remains a common assumption in the GPLVM literature to reduce complexity.

Substituting the mean and covariance of Δ into equation 7.9 to obtain

$$\mathbb{E}[\varepsilon] = \sum_{i=1}^{N-1} ||\mu_{i+1} - \mu_i||^2 + 2D(\sigma_{i,i}^2 + \sigma_{i,i+1}^2) \tag{7.18}$$

CURVE ENERGY
StochMan

In the latter term, the trace reduces to the dimensionality of the curve. The GPLVM uses this expression for the expected energy of a curve.

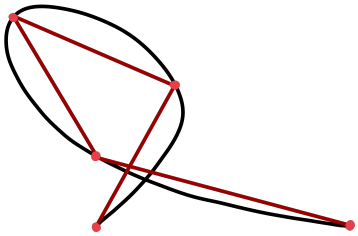


Figure 7.3: Illustration of estimated cubic spline (black) from small line segments (red).

Cubic Splines as Approximate Geodesics

StochMan employs cubic splines [105] to fit geodesics by minimising the expected energy (equation equation 7.18). This approach changes the geodesic computation from solving a second-order, ordinary differential equation to optimising the parameters of a cubic spline such that the energy is minimised. This approach allows re-using information from learnt geodesics for computing new. This section discusses the details of how cubic splines approximate geodesics.

Generally, splines are functions composed of L polynomials, each defined on a small domain. A cubic spline $\hat{C}(t)$ is a piece-wise, third-order polynomial [105] (spline of order three) and parametrised by t. We can consider a cubic spline on the interval $[a, b]$ where we segment the interval as $a = t_0 < t_1 < \ldots < t_{L-1} = b$. Then the cubic spline can be written as

CUBIC SPLINE

$$\hat{C}(t) = \hat{C}_{l-1}(t) = a_{l-1}t^3 + b_{l-1}t^2 + c_{l-1}t + d_{l-1}, \quad t_{l-1} \leq t \leq t_l \tag{7.19}$$

which yields four unknowns for each of the L pieces of the spline. This can be solved as a tridiagonal system of equations [106] with parameters we find by optimisation.

A general cubic spline has some undesirable characteristics for geodesics, e.g. a cubic spline is not necessarily continuous, which is a requirement for a geodesic. To accommodate the geodesic requirement, we impose four constraints: Three are continuity in the zeroth, first, and second order derivatives at the ends of each segment respectively, (which leads to natural splines), and the fourth constraint fixes the endpoints of the cubic spline.

NATURAL CUBIC SPLINE

If the endpoints are not fixed, and the coefficients are learnt using the expected energy as an objective function, the curve collapse to a point as a point has zero energy. This is not desirable.

We write the first three constraints of the continuity of derivatives as

$$\begin{aligned} \hat{C}^{(0)}_{l-1}(t_l) &= \hat{C}^{(0)}_{l}(t_l) \\ \hat{C}^{(1)}_{l-1}(t_l) &= \hat{C}^{(1)}_{l}(t_l) \\ \hat{C}^{(2)}l_{l-1}(t_l) &= \hat{C}^{(2)}_{l}l(t_l) \\ \hat{C}(0) &= p_0 \\ \hat{C}(1) &= p_1 \end{aligned} \tag{7.20}$$

where the superscript parentheses denote the order of the derivative. These constraints describe natural splines, ensuring a "natural" transition from one piece to the next. The fourth and fifth constraints arise from fixing the cubic spline to the endpoints.

This is a set of linear equations that is an underdetermined problem, so no unique solution exists. StochMan finds the null space using singular value decomposition. Here, we also revisit the fourth constraint: The endpoints should be fixed to avoid the curve collapsing to a point, $C_0(0) = 0$ and $C_{l-1}(1) = 1$. To ensure that the geodesic starts and ends in the desired points, p_0 and p_1, we define the geodesic as a

$$C(t) = p_0 t + (1 - t)p_1 + \hat{C}(t) \tag{7.21}$$

The first two terms correspond to a straight line between p_0 and p_1. By constraining the cubic spline to end in zero, we can ensure that the geodesic curve $C(t)$ starts in p_0 and ends in p_1. That is $C(0) = p_0$ and $C(1) = p_1$.

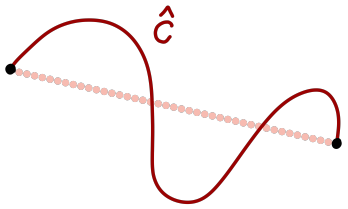


Figure 7.4: Illustration of a geodesic in StochMan as a sum of a straight line (light red) and a natural cubic spline, $\hat{C}(t)$ (red).

In StochMan, we determine the number of segments L , the start point p_0 and the end point, p_1, see figure 7.4. The cubic spline is initialised as a Euclidean straight line, and the parameters of the cubic spline are optimised to obtain a geodesic approximation. The resulting curve is reparametrised to have constant speed (see section 5.3). This results in local optima of geodesics. To compute multiple geodesics simultaneously, we batch the cubic splines.

This method conveniently exploits the geodesics computed between two points in a previous step. Assuming that the endpoints only move slightly, the previous geodesic is an excellent initialisation for the current though with updated endpoints. Unfortunately, this scales poorly as this essentially corresponds to l^2 regression problems for each geodesic.

Approximating Geodesics using a Discretized Manifold

Optimising cubic splines as geodesics, outlined in the previous section, is slow. This section contains an alternative that discretises the manifold and uses Dijkstra's algorithm [107, 108] to compute shortest paths [109]. The resulting discrete curves are smoothened by fitting cubic splines.

Dijkstra's algorithm finds a shortest path between nodes on a graph. Unlike the previously described method, Dijkstra's algorithm finds a global shortest path (though the shortest path might not be unique [110]). We will not go into the details of the algorithm here.

This is also the algorithm that empowers isomap[7]

To compute geodesics using Dijkstra's algorithm, we divide the latent space into a square grid of sufficiently small resolution (we refer to this as a discrete manifold). Computing the shortest path requires the distances between all neighbouring nodes, which requires the metric. We evaluate the metric at all points at a part of the initialisation of the discrete manifold.

Once that is found, we know all the distances, and we can put Dijkstra's algorithm to work to find discrete geodesics. Then we fit a cubic spline like previously but with this initialisation from the shortest path found using Dijkstra's algorithm. This is quick and leads to globally optimal geodesics. This approach eliminates the direct dependency on the number of data points in the metric computations.

Regarding implementation, there are four things to consider: 1) This procedure has a long initialisation time as the metric has to be evaluated at all points. Once the discrete manifold object is initialised, computing geodesics is fast. 2) The discretised manifold inherits from the manifold object such that the API is the same. 3) We have implemented boundaries on the DM to mitigate geodesics (or other points) that are "off" the manifold to ensure that the discrete manifold is defined everywhere. We extrapolate the metric on the boundary to a constant. 4) Practically, the discrete approach can only estimate manifolds up to three dimensions as the discrete lattice grows too large otherwise.

7.2 GEOMETRY IN GPLVMS: THE STARFISH

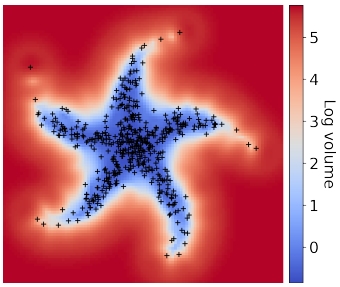


Figure 7.5: A manifold in the starfish dataset. The background is the log volume element. Blue indicates a smaller volume (less uncertainty), and red indicates a larger volume (greater uncertainty).

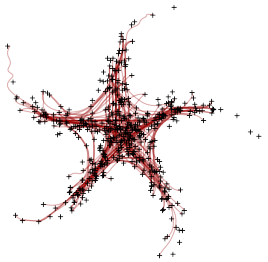


Figure 7.6: A manifold in the starfish dataset (black) with estimated geodesics (red).

As an illustrative example, we consider the starfish toy dataset: A two-dimensional starfish embedded in three dimensions with a linear mapping (appendix E). This simple dataset has the advantages that it is easy to learn, we know its true generating process, and it has an interesting shape in the latent space (even with PCA), making it well-suited for developing familiarity with learning manifolds.

The stochastic manifold is described by the Gaussian process that should capture the data's geometric nature. Qualitatively, there are two ways of verifying this: One way is to check that the volume element (section 5.2.2) corresponds to the latent representation. We expect a smaller volume close to data and a larger away from data [10], and this is also what we observe in figure 7.5. Alternatively, we can inspect that the resulting geodesics follow data (figure 7.6). The geodesics in red follow the latent variables in black nicely, and if the observations are removed, the shape of the starfish would still be visible from the geodesics.

Next, we want to understand which parameters lead to good geodesics, see figure 7.7. This plot allows for qualitative inspection of the geodesics as a function of the GP parameters. The noise variance is fixed, and the kernel variance θ and the length scale l are varied across columns and rows, respectively. For a considerable length scale, seemingly independent of variance (upper left corner in figure 7.7), the geodesics

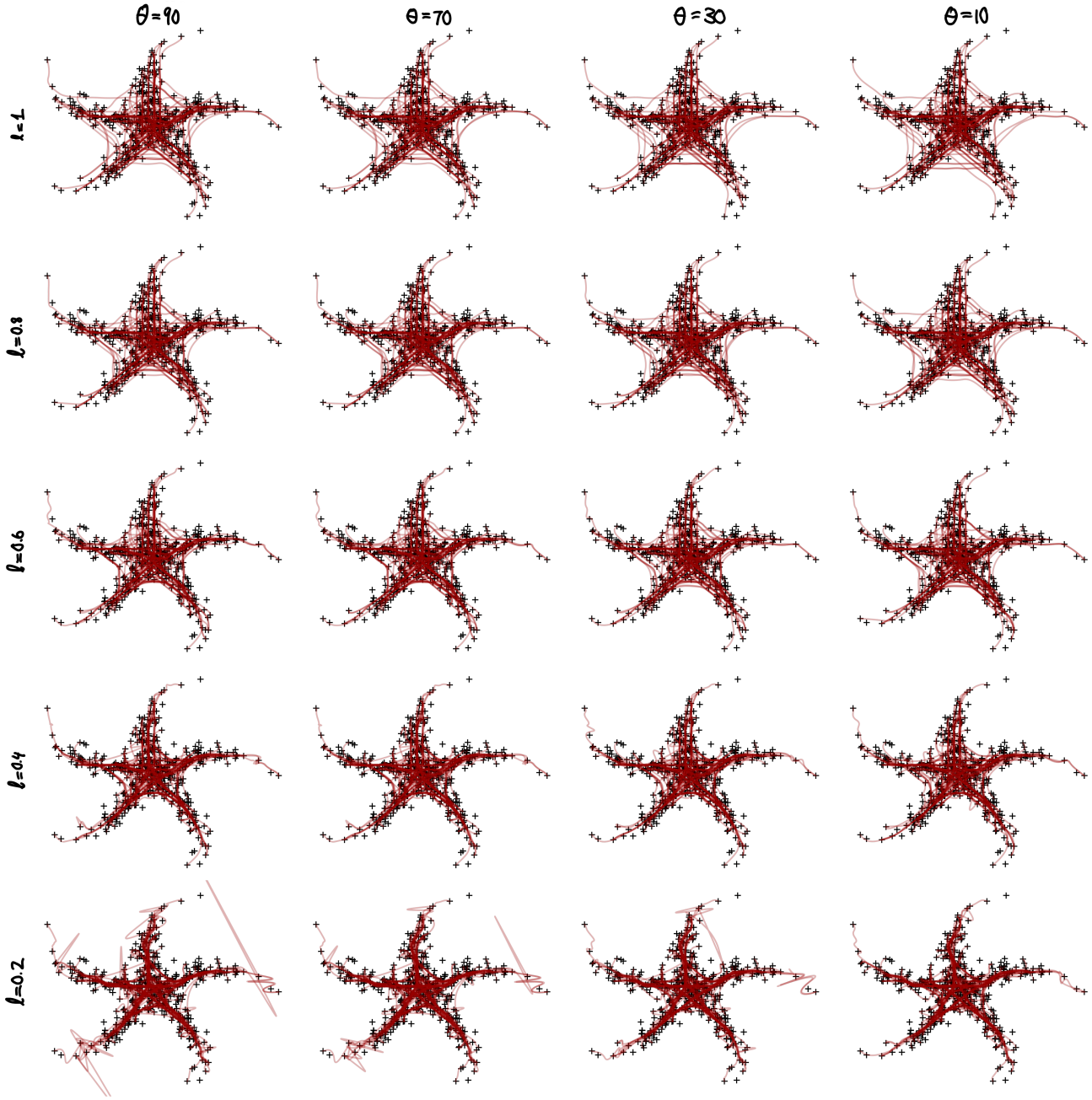


Figure 7.7: Geodesics in the starfish manifold as a function of the kernel variance θ (columns) and the length scale, l (rows) in the GPLVM. The likelihood noise is fixed, $\sigma = 0.05$.

do not properly follow the manifold, whereas the geodesics follow highways on the manifold for small variances and small length scale (lower right corner in figure 7.7). A large variance and a small length (lower left corner in figure 7.7) lead to poorly estimated geodesics. The geodesics (a proxy of the manifold) depend on the GP parameters.

In summary, we observe that the hyperparameters of the GP influence the geodesics and to get geodesics that follow the manifold, we must have the right balance between the GP hyperparameters. This links

back to the challenge of estimating hyperparameters (briefly mention in section 3.4.1); we should have well-estimated hyperparameters to useful geometry.

7.3 GEOMETRY IN THE SAS DECODER

We are ready to work on a more complicated model and for this, we consider the SAS decoders in chapter 4. These models learn structured latent representation but lack a predictive distribution. To accommodate this, we use the trained SAS decoder to initialise a GPLVM (section 3.3) which we do not train. Specifically, the SAS decoder was trained of FashionMnist (also used in chapter 4), which consists of 60,000 28×28 images in ten classes of clothing items. We encode the 1,000 points of the test points to obtain an initialisation of the latent variables for the GPLVM and use the GP hyperparameters directly from the SAS model. This yields a GPLVM with a latent representation, we might learn from and with a predictive distribution which is required for geometry.

Qualitatively, the SAS decoder captures structure (figure 7.8): Sandals, sneakers and ankle boots are similar and the models palce these three classes close. Trousers and bags are clustered by themselves, both separate from other categories. The models also seem to capture that the transitions between e.g. dresses and tops or t-shirts and long sleeves may be more smooth that suggested by a categorical label. The figure also shows Euclidean and Riemannian (geodesic) interpolations. The latter avoids areas with no data, so it travels through dresses, pullovers and sneakers to compute the distance between a pair of trousers and a sandal[2].

2. It raises the philosophical question of whether a meaningful smooth interpretation between pants and sneakers actually exists.

Figure 7.9 shows the same representation but colour-coded according to class labels, making it easier to distinguish classes and with the volume element in the background. It also shows reconstructions along the interpolations with the Riemannian in the top row of images and the Euclidean in the bottom. While neither is perfect, the Riemannian reconstructions are less blurry in this particular example. To show that the Riemannian approach captures the manifold, the bottom figure 7.9 shows five hundred geodesics in the same representation which all follow data and avoids uncertainty in the trained model.

Generally, obtaining nice geodesics (e.g. in the starfish) requires some tuning but (surprisingly) in this SAS model, it worked right out of the box without any extra tuning or hacks. Evidently, the model captured some geometry in this case. Given what we learnt in the previous section (for the geodesics to follow the manifold, the GP parameters

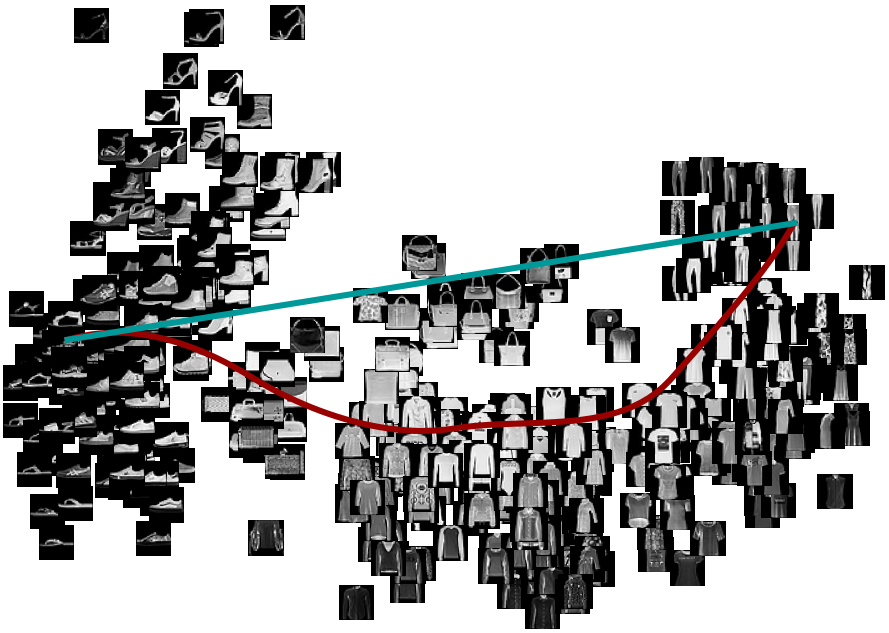


Figure 7.8: Latent representation in a SAS decoder on FashionMnist shown as an image manifold. The teal line is the Euclidean interpolation and the red is a geodesic.

must balance), figure 7.9 (bottom) suggests that the hyperparameters are well-estimated in the trained SAS decoder. This reignites the hope that we can exploit geometry *during* training to learn more meaningful representation.

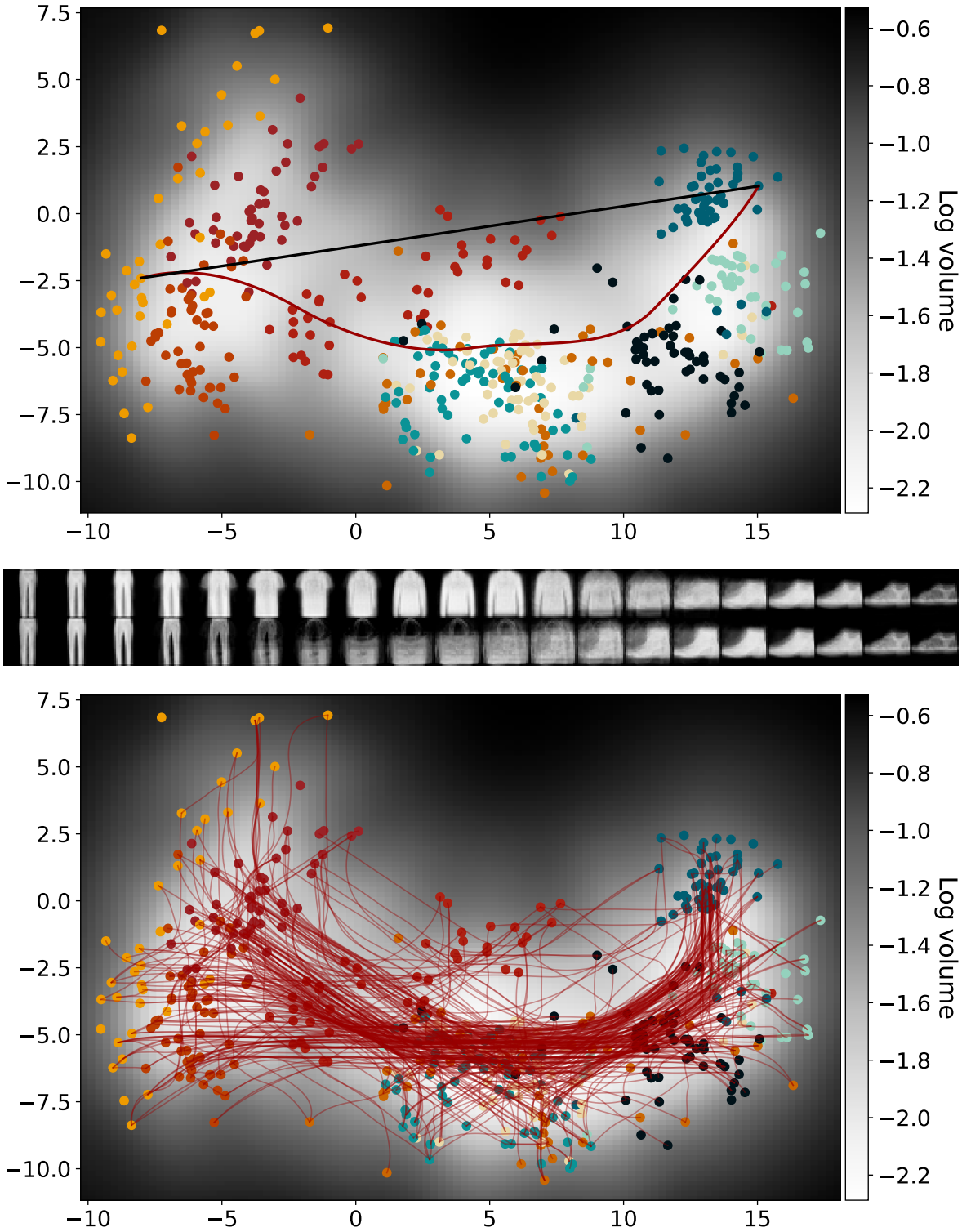


Figure 7.9: The top and bottom plots show the latent representation of a SAS decoder on FashionMnist colour-coded according to class label with the log volume element in the background. The top plot shows a Riemannian (red) and a Euclidean (black) interpolation and corresponding reconstructions along the interpolations with the Riemannian in the upper row and the Euclidean in the bottom row. The bottom plot shows five hundred geodesics.

7.4 ISOTROPIC BROWNIAN MOTION

The previous section demonstrated that the SAS decoder learns well-estimated manifolds and it would be interesting to capture this in a distribution. Such a distribution could be useful for exploration and visualisation *after* training, and it might prove useful *during*[3] training.

3. Spoileralert, it does not prove useful in this thesis as the joint training does not converge at the time of writing.

We intend to change the prior on the latent variables to the Riemannian Brownian motion as also considered by Kalatzis et al. [111], see more in Hsu [112]. This section introduces the Riemannian Brownian motion, a flexible distribution with only two parameters that can easily be adapted to respect a Riemannian geometry. We do this by first developing the intuition around Euclidean Brownian motion and second extending to Riemannian Brownian motion for which we consider an approximation of the log probability and a sampling scheme.

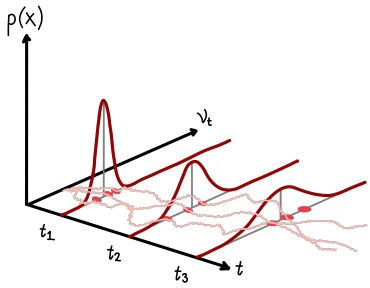


Figure 7.10: A one-dimensional Brownian motion where distribution (red) and the variance ν_t change as time passes. Samples of the Brownian motion are shown in pink.

7.4.1 Euclidean Brownian Motion

Brownian motion is a macroscopic physical phenomenon that occurs when particles in a medium bump into each other randomly. It is an ensemble of particle trajectories governed by microscopic dynamics called random walks. Keeping track of the net forces is tedious so we adopt a probabilistic view. Figure 7.10 illustrates how the density of a one-dimensional Brownian motion changes over time and how random walks are samples from these distributions–how the macroscopic behaviour changes as the random walks changes.

The Brownian motion fulfils the Markov property (that any current state depends only on the previous), and the differences in location (i.e. the subsequent steps) are normally distributed [113].

To later generalise this to Riemannian manifolds, it is useful to consider the two-dimensional case (figure 7.11). Each next step is sampled from multivariate, Gaussian distribution. Assuming an isotropic heat kernel leads to a diagonal covariance, and we can construct the full motion as

$$x_{t+1} \sim \mathcal{N}\left(x_t, \frac{\nu_t^2}{T}\mathbb{I}_N\right) \tag{7.22}$$

where ν_t is the variance of the Brownian motion at time $t \in [1, \ldots, T]$ where T is the duration of the Brownian motion.

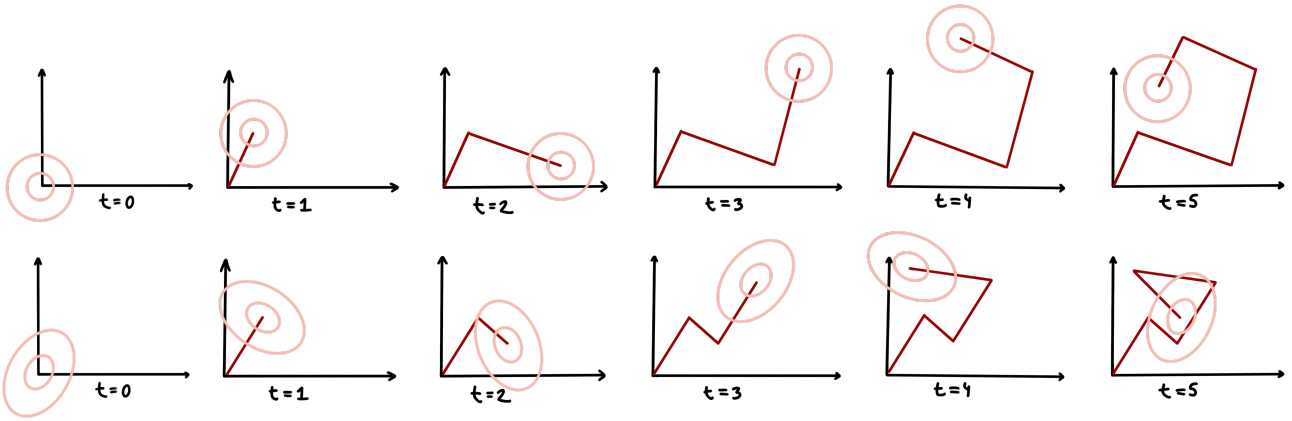


Figure 7.11: lllustrations of how Euclidean (top) and Riemannian (bottom) Brownian motions are constructed iteratively, starting from origo at $t = 0$. At each step, the next is sampled from a Gaussian distribution.

7.4.2 Riemannian Brownian Motion

The goal of this section is a distribution that incorporates geometry and the Riemannian Brownian motion integrates well with Riemannian geometry for several reasons; among these, its inherent construction of small steps means we can represent the Riemannian manifold as locally Euclidean. This section introduces Riemannian Brownian motion (following Hsu [112]), an approximation of its density and a sampling scheme for our specific purpose. In the following, we refer to the isotropic Riemannian Brownian motion simply as Brownian motion. When we mean isotropic Euclidean Brownian motion, this will be clear from the text.

The Brownian motion includes the metric, so diffusing particles respect the geometry and stay on the manifold. Intuitively, the Riemannian Brownian motion can be constructed the same way as the Euclidean,

$$x_{t+1} \sim \mathcal{N}\left(x_t, \frac{v_t^2}{T} G_{x_t}^{-1}\right) \tag{7.23}$$

using a scaled inverse metric as the covariance. Before we motivate this in section 7.4.4, we consider the implications. The distribution from which we sample the next step now depends on our location in space; see figure 7.11. We consider the distribution in the tangent space at each step and sample the next step from that. Intuitively, when a particle reaches the edge of the manifold, the metric increase due to uncertainty, the covariance decrease and sampled steps become smaller and smaller until the particle is stuck. This constrains the random walks to the manifold and is also demonstrated in figure 7.12, which compares a Euclidean and a Riemannian random walk; the Euclidean random walk ignores the geometry, and the Riemannian random walk respects the geometry by staying on the manifold.

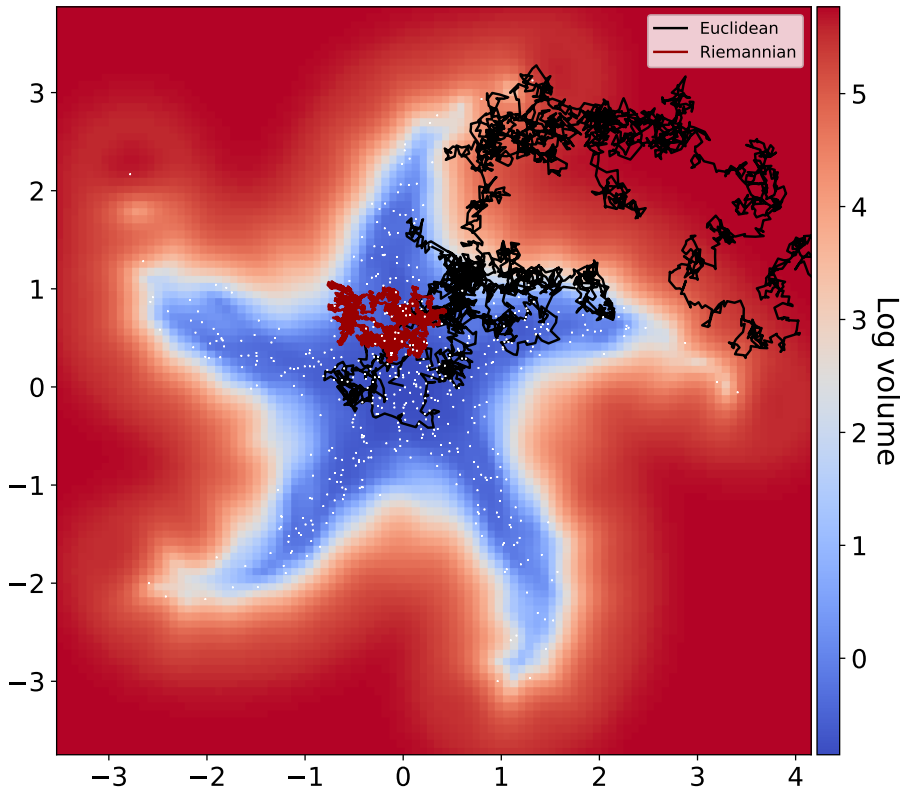


Figure 7.12: The latent representation of the starfish manifold with the logarithm of the volume element in the background with latent variables in white dots. The red curve is a representative Riemannian random walk and the black is a representative Euclidean, of equal duration.

7.4.3 Approximate Isotropic Riemannian Brownian Motion

The Brownian motion density, $p(x)$, may be described as the heat kernel associated with the Laplace-Beltrami operator (a generalisation of the Laplace operator to Riemannian manifolds). This gives rise to a stochastic differential equation which is challenging to work with [111, 112]. Instead, Hsu [112] suggests a parametrix expansion and following Kalatzis et al. [111], we use the zeroth order term as the density for the Brownian motion (see figure 7.13),

$$p(x_n) = \frac{1}{(2\pi t)^{Q/2}} H_0 \exp\left(-\frac{d^2(x_n, \mu)}{2t}\right) \tag{7.24}$$

where $t \in \mathbb{R}$ is the duration of the Brownian motion, Q is the dimensionality, $x_n \in \mathbb{R}^Q$ and μ is the centre of the Brownian motion distribution, $d(\cdot, \cdot)$ is the Riemannian distance (i.e. the geodesic distance between the mean of the Brownian motion and an input), and the H_0 term is computed as

$$H_0 = \left(\frac{\det G_{x_n}}{\det G_\mu}\right)^{1/2} \tag{7.25}$$

which is a ratio of log volume measures under Riemannian metrics evaluated at a point x_n and the mean of the Brownian motion μ.

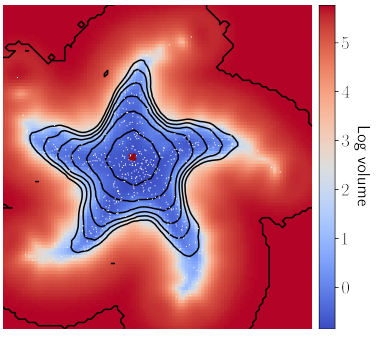


Figure 7.13: The latent representation of the starfish manifold with the logarithm of the volume element in the background with latent variables in white dots. The black contour lines show the approximated Brownian motion density.

The log probability as

$$\log p(x_n) = -\frac{Q}{2}\log(2\pi t) + \log H_0 - \frac{d^2(x_n,\mu)}{2t} \tag{7.26}$$

$$= \frac{1}{2}\left(-Q\log(2\pi t) + \log\frac{\det G_{x_n}}{\det G_\mu} - \frac{d^2(x_n,\mu)}{2t}\right) \tag{7.27}$$

where t is the square root of the Brownian motion variance v.

In practice, the primary computational hurdle is geodesic computations, which we have already discussed (see section 7.1.2), making this approximation relatively fast.

7.4.4 Sampling the Riemannian Brownian Motion

The idea with the Riemannian Brownian motion is stepping on the manifold without evaluating the expensive exponential map (which involves solving a second-order ordinary differential equation). Following Kalatzis et al. [111], we assume an isotropic heat kernel for the steps

$$\Delta \sim \mathcal{N}(0, \Sigma_M) \tag{7.28}$$

with $\Sigma_M = v^2\mathbb{I}$, and Δ is the step in the observation space [111]. We want to know the corresponding step ϵ in the latent space. The Taylor expansion of the map evaluated in a latent point $x + \epsilon$,

$$f(x+\epsilon) = f(x) + J_x\epsilon + \mathcal{O}(\epsilon^2), \tag{7.29}$$

with $f : \mathbb{R}^Q \to \mathbb{R}^D$ and J_x is the Jacobian of f at the point x. Then a step on the manifold is

$$\Delta = f(x+\epsilon) - f(x) = J_x\epsilon + \mathcal{O}(\epsilon^2) \tag{7.30}$$

Truncating at ϵ^2, yields $\Delta = J_x\epsilon$ which implies that $\epsilon = J_x^+\Delta$ where J_x^+ is the pseudo-inverse,

$$J_x^+ = (J_x^\top J_x)^{-1}J_x^\top \quad \in \mathbb{R}^{N\times Q}. \tag{7.31}$$

With this, we can obtain the distribution of steps in the latent space,

$$\epsilon \sim \mathcal{N}(0, J_x^+\Sigma_M(J_x^+)^\top). \tag{7.32}$$

With $\Sigma_M = v^2\mathbb{I}$, we have

$$\begin{aligned} J_x^+ \Sigma_M (J_x^+)^\top &= (J_x^\top J_x)^{-1} J_x^\top \Sigma_M J_x (J_x^\top J_x)^{-\top} && (7.33)\\ &= v^2 (J_x^\top J_x)^{-1} J_x^\top J_x (J_x^\top J_x)^{-\top} && (7.34)\\ &= v^2 (J_x^\top J_x)^{-\top} && (7.35)\\ &= v^2 G_x^{-1} && (7.36) \end{aligned}$$

where $-\top$ denotes the inverse transposed and G_x is the metric at x. Then the distribution of steps on the manifold is

$$\epsilon \sim \mathcal{N}(0, v^2 G_x^{-1}) \tag{7.37}$$

and the Riemannian Brownian motion for $t = [0, \ldots, T]$ is

$$x_{t+1} \sim \mathcal{N}\left(x_t, \frac{v_t^2}{T} G_{x_t}^{-1}\right). \tag{7.38}$$

This lets us step directly on the manifold using the Brownian motion: Sampling follows the simple intuition outlined above and requires a mean, μ, a variance, v^2, and a duration, T, of the Brownian motion, and a distance function, $d(\cdot, \cdot)$, which also requires a metric at each point. Implementing this as a distribution requires a method for computing the log probability (section 7.4.3) and a method for sampling from the distribution (this section). As we have both, we implement the Riemannian Brownian motion as a distribution in StochMan.

7.5 A RIEMANNIAN GPLVM

Finally, we have the components to build a model that incorporates geometry during training: We employ the Riemannian Brownian motion as a prior in the GPLVM, and we optimise the hyperparameter.

Similarly to the GPLVM, we assume that data are generated by mapping latent variables to observations with a map $f : \mathbb{R}^Q \to \mathbb{R}^D$, and we assume a Gaussian process prior $\mathcal{GP}(0, k(x, x'))$ on the map, where $k(x, x')$ is an EQ kernel (see section 3.3.1). Unlike the GPLVM, we assume an approximate Riemannian Brownian motion as the prior on the latent variables,

$$p(x_n) = \frac{1}{(2\pi t)^{Q/2}} H_0 \exp\left(-\frac{d^2(x_n, \mu)}{2t}\right), \tag{7.39}$$

in which the parameters are described in section 7.4.

Note that there are some challenges in this model: The prior depends on the metric, and the metric is estimated from data, making the prior implicitly dependent on data. The objective function for the model using maximum aposteriori estimation,

$$\log L = \log p(Y|X, v) + \log p(X) \tag{7.40}$$

$$= \sum_{n=1}^{N} \log p(Y|X, v) + \sum_{n=1}^{N} \log p(x_n), \tag{7.41}$$

where the prior factorises due to the assumption of isotropy in the Brownian motion. This model can then be trained using the sum-product algorithm [51].

7.6 DISCUSSION

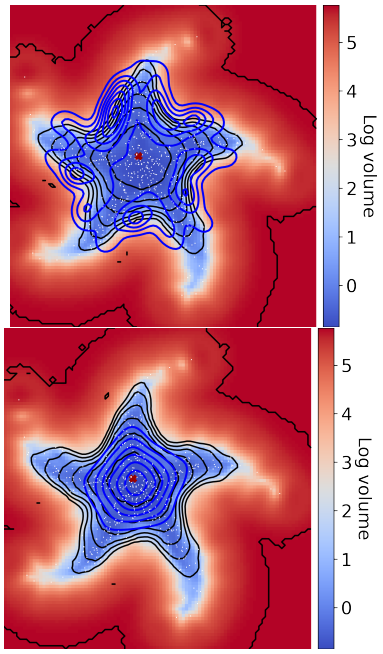


Figure 7.14: The latent representation of the starfish manifold with the logarithm of the volume element in the background with latent variables in white dots. The black contour lines show the approximated Brownian motion density. The blue contour lines show a kernel density estimate from 2,500 samples when inverting the metric (top) and not inverting (bottom).

Our goal was to develop a Riemannian GPLVM by placing a Riemannian Brownian motion prior on the latent variables. At this point, the model described above does not learn anything useful, so rather than presenting results, this section discusses our approach.

The Riemannian GPLVM (section 7.5) requires (at least) three steps: 1) Deterministic approximations of stochastic geometric quantities that make geometric computations feasible (chapter 6) and a computational framework (section 7.1) to compute these. 2) An implementation and pre-training of the approximated Riemannian Brownian motion (section 7.4) and 3) assigning the Riemannian Brownian motion as a prior in the GPLVM and training the model to convergence (section 7.5). We have completed steps one and two, but none of our training attempts has converged, leaving step three uncomplete at this point.

The are many possibilities for faulty steps. We have already discussed some of the theoretical challenges (chapter 6) and some of the practical challenges (section 7.1). Here, we focus on the challenges with the Brownian motion and the combination of the Brownian motion and the GPLVM.

7.6.1 Brownian Motion Sampling: To Invert or Not to Invert?

Assuming suitable GP parameters, we can train isotropic, Riemannian Brownian motions on manifolds. This allows us to estimate the density (equation 7.24) under the Riemannian Brownian motion and to sample random walks. We observe the pre-training of the Riemannian

Brownian motion is quite noisy and rarely converges. We can also estimate a density using the samples and kernel density estimate [114] (figure 7.13, top).

The resulting density estimate is unexpected as we observe the highest probability density just off the manifold where there is no data. This suggests that there is something we do not understand yet, as this behaviour is opposite to the theory's prediction. This is also not the desired behaviour.

On the other hand, we recover the desired behaviour when using the metric, G_x, rather than the inverse of the metric, G_x^{-1}, in sampling the next step,

$$\epsilon \sim \mathcal{N}(0, v^2 G), \tag{7.42}$$

in the covariance. We observe nicely estimated manifolds, and as an additional benefit, the training of the Brownian motions converges consistently. We choose to use the latter as we focus on the desired behaviour.

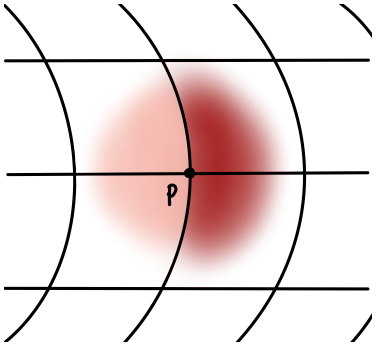


Figure 7.15: Coordinates that introduce a drift in the Brownian motion even in flat space. The illustration is recreated from StackExchange [115].

7.6.2 Brownian Motion Sampling: To Drift or Not to Drift?

The Riemannian Brownian motion can be written as a stochastic differential equation with a drift term [116],

$$\mathrm{d}X_t = \sqrt{g^{-1}}\mathrm{d}B_t - \frac{1}{2} g^{ij} \Gamma^k_{ij} \mathrm{d}t. \tag{7.43}$$

where $\mathrm{d}b_t$ is a Euclidean Brownian motion. Our approach in section 7.4 is aligned with previous approaches [17, 111], which do not implement the drift term.

The drift is an artefact of the coordinates, which is relevant in non-flat bases. This can be understood intuitively by considering a Brownian motion (without drift) in flat space but curved coordinates (figure 7.15). Starting from p, a particle is more likely to end its walk in the red area than in the light red area but a Brownian motion, $\mathrm{d}X_t = \sqrt{g^{-1}}\mathrm{d}B_t$, would not reflect this. The drift compensates for this.

This was the motivation for approximating the Christoffel symbol in section 6.3, and that approximation should be straightforward to implement in the sampler.

7.6.3 Related Works: Geometry in GPLVMs

There are numerous works considering geometry in the GPLVM [3, 10, 58–61, 117–119], so it seemed natural to build the geometry directly into the model. However, there are a few alternative routes for incorporating non-Euclidean geometry when considering GP-based models which we briefly review here.

An obvious choice is to use the Riemannian normal distribution as prior, which is well-studied [92, 120–123], but estimating its normalisation constant is computationally expensive. Instead, one might consider changing the distance function in the EQ kernel, but Feragen, Lauze, and Hauberg [124] proves that it cannot be guaranteed that Gaussian kernels with Riemannian distances are positive and semi-definite. Jayasumana et al. [125] proof that the EQ kernel can be generalised to Riemannian manifolds iff they are isometric to the Euclidean manifolds, which is a too-hard constraint for our purpose.

Developing new kernels that incorporate the geometry is an alternative approach; Jaquier et al. [126] constructs a kernel on hyperbolic manifolds and Jensen et al. [61] for tori, spheres, and SO(3). These are both exciting works, but neither allows for capturing non-parametrised manifolds as they constrain the manifold by assuming a parametrisation.

Finally, one might question the choice of the GPLVM over other probabilistic latent variable models (e.g. the variational autoencoder, VAE). Both models learn probabilistic representation though arguably the VAE learns better representations. The benefit of the GPLVM is the variance estimation away from data, which is not the case for neural networks which tend to extrapolate poorly [17, 111].

7.6.4 Future Directions

Our goal was to develop a proof of concept for the GPLVM using the Riemannian Brownian motion as a prior on the latent variables and succeeded partially. At this point, we have the theory and the computational framework to train the Riemannian GPLVM but currently, it does not learn anything useful. Going forward, the first step is to get the model to converge. Later, we would like to implement a mixture of Brownian motions to capture representations with clusters, e.g figure 7.8. Once we have a proof of concept, it would be interesting to extend to the Bayesian GPLVM. With a Riemannian Brownian motion prior on the latent and potentially with hyperpriors on the Brownian motion parameters.

OUTTRO

8

This concludes our efforts in finding a model in which distances are computed along the manifold. We started in topology (using symmetries as a proxy) and explored four dimensionality reduction models' ability to preserve the topology in visualisation (chapter 2). All four turned out to break the symmetry at alarming rates. Our results indicated two benefits of the GPLVM; the likelihood of the GPLVM is a useable measure for distinguishing broken from not broken symmetries and clear transitions from preserved to broken in our symmetry measure.

For these reasons, we explored GPLVMs and their challenges (chapter 3) and addressed some of them with the SAS decoders (chapter 4). Learnt latent spaces with this approximation tend to capture more structure in the latent which is suited for our pursued model. This revives the hope of finding a model that enables learning from the representation.

We introduced the foundation of Riemannian geometry (chapter 5) and stochastic geometry for GPLVM (chapter 6) with the intention of developing a new model. We do geometry in trained SAS models and consider Riemannian Brownian motions as priors for the latent variables in such models (chapter 7). Though we fail to complete this project, the hope of finding a model that captures (more of) the geometry persists.

The intention of capturing geometry is to capture the generating process of observations. This synthesizes patterns in large amounts of data that humans would otherwise be unable to grasp and provides insights in a humanly interpretable format. This is one way humans can learn from machine learning.

APPENDIX

PUBLICATIONS

A

Pablo Moreno-Muñoz*, Cilie W. Feldager*, and Søren Hauberg
Revisiting Active Sets for Gaussian Process Decoders.
DOI:10.48550/arXiv.2209.04636.arXiv:2209.04636 [cs, stat]
Poster at Neural Information Processing Systems, 2022.

* Equal contribution

Cilie W. Feldager, Søren Hauberg, and Lars Kai Hansen.
Spontaneous Symmetry Breaking in Data Visualization.
In: Artificial Neural Networks and Machine Learning.
Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)". Springer, 2021, pp. 435–446. isbn: 978-3-030-86339-5.
DOI: 10.1007/978-3-030-86340-1_35

Spontaneous Symmetry Breaking in Data Visualization

Cilie W. Feldager[1], Søren Hauberg[1], and Lars Kai Hansen[1]

Section for Cognitive Systems, Technical University of Denmark
{cife, sohau, lkai}@dtu.dk

Abstract. Data visualization tools should create low-dimensional representations of data that emphasize structure and suppress noise. However, such non-linear amplifications of structural differences can have side effects like spurious clustering in t-SNE (Amid and Warmuth [1]). We present a more general class of spurious structure, namely broken symmetry, defined as visualizations that lack symmetry present in the underlying data. We develop a simple workflow for detection of broken symmetry and give examples of spontaneous symmetry breaking in t-SNE and other well-known algorithms such as GPLVM and kPCA. Our extensive, quantitative study shows that these algorithms frequently break symmetry, thereby highlighting new shortcomings of current visualization tools.

1 Motivation

Data visualization is a core tool in the machine learning toolbox. Data sets are visualized for exploration, to formulate hypotheses and to make modeling decisions. Visualization is commonly used for interpretation of learned models, e.g. visualization of latent variables of a generative model to understand representations. Data visualization is also very useful for debugging. For these applications *faithfulness* is a concern — can we trust the structure revealed in a visualization?

Most data of interest is high dimensional, hence can not be directly visualized. Rather, some form of dimensionality reduction is required, which inevitably will lead to loss of information. Popular schemes such as t-SNE [27], aim at two or three-dimensional representations that capture both local and global structure in data. Fig. 1 shows a two-dimensional t-SNE visualization of images from the COIL-20 dataset [20]; the given example concerns a wooden object on a turntable that is viewed from multiple, equidistant angles forming full 360° rotation. Such an incremental physical rotation leads to a set of images with a simple topological structure which can be quantified by the neighborhood graph. More specifically, we form a graph with the images as nodes and connect neighboring nodes along the rotation path to obtain the graph of a circle. The neighborhood graph presents us with a strong physical symmetry and we naturally expect a visualization of the data to reveal this pattern by a structure which is topologically equivalent to a circle. Evidently, this does not happen: The visualization has broken the symmetry and "invented" a difference between neighboring points that is non-physical.

The significance of transformations and the ensuing question of symmetry preservation goes beyond the physical rotations of the COIL data set. Parameterized transformations are key to modern data augmentation strategies. The question of preservation of symmetries in augmented data sets is then related to whether given symmetries are successfully represented during learning.

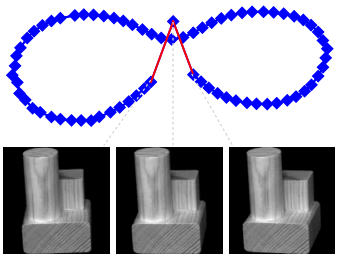


Fig. 1. We analyse a set of images of an object subject to a 360° rotation on a turntable. The nearest neighbor graph forms a simple circle, however when the set is visualized using t-SNE the symmetry is lost.

Our contribution is to identify a new, general class of spurious structures in data visualization, namely spontaneously broken symmetry, defined as representations that lack symmetry present in the underlying data. We provide a topological, quantitative measure to detect broken symmetry (Sec. 2) allowing for a systematic study. Our empirical studies (Sec. 3) show that widely used visualization techniques break simple symmetries like rotations, hence, challenging the notion that they conserve global structure.

2 Symmetries, Graphs, and Persistent Homology

Symmetry groups. We consider symmetries, i.e. a property of a system that remains unchanged under a given transformation. The images of the wooden toy in Fig. 1 are formally *equivariant* when the toy is rotated physically on the turntable, while the outputs of a deep network for image based object classification ideally would be invariant (symmetric) under rotation.

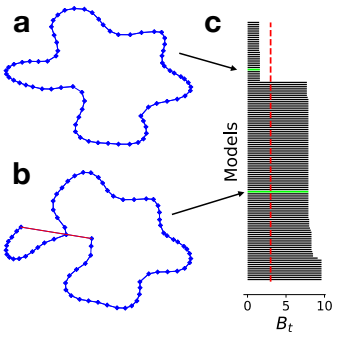


Fig. 2. The latent space of a model that preserves symmetry (a) and one that does not (b). (c) A *barcode* as a function of thresholds B_t.

Mathematically, such transformations and symmetries are described by Lie groups [11]. A real Lie group is a smooth differentiable manifold on which points are connected through a group operation and its inverse. For instance, rotation matrices form a smooth group with the matrix multiplication group operation. The unit circle can then be generated by a single unit vector and its multiplication with all members of the group of rotation matrices. If the rotation group governs a physical phenomenon then we expect to observe data along a path that topologically is a circle, disregarding observation noise.

This paper focus on situations where the governing group is known and investigate if its structure is preserved by common visualization techniques. This is achieved by verifying if the group topology remains intact under visualizations.

Discrete approximations. In practice, we only observe a finite number of data points, rather than the entirety of a group. We can, however, approximate the

path spanned by the observations with a graph, where points are connected if their generating group elements are close under the group metric. For instance, we may connect rotated images in a graph if their rotation angles are similar.

Measuring broken symmetry. For visualization, we map data to a low-dimensional space (typically $\mathbb{R}^2$); we let $X = \{x_i\}$ denote data coordinates in this low-dimensional space. We can now determine if a symmetry has been preserved under visualization by asking if the associated graph can be recovered from the low-dimensional coordinates. As the graph informs as to which points should be neighbors, we measure for each set of neighbors the radius of the ball needed to include one in the other's neighborhood graph. To compare across methods, we scale all distances by their median

$$B_{\text{median}} = \underset{(x_i,x_j)\in G}{\text{median}}(\|x_i - x_j\|), \tag{1}$$

where G denotes the graph associated with the generating group. We rely on the median due to its high breakdown point [15]. We, thus, measure

$$B_{ij} = \frac{\|x_i - x_j\|}{B_{\text{median}}}. \tag{2}$$

We can then threshold this measure such that, we say that a symmetry has been broken if $B_{ij} > B_t$ for any pair or equivalently, $\max(B_{ij}) > B_t$. We define $B_{\max} := \max(B_{ij})$. Note that this measure does not distinguish between one or multiple instances of broken symmetry.

Persistent homology. The measure above is linked with *persistent homology* [12]. This is a key mathematical tool in topological data analysis that has been shown to be robust to perturbations of the input data [6]. Following Carlsson [5], we place balls on each data point with radius ϵ and points falling within this ball defines a neighborhood. This defines a topological space Ω_ϵ. By varying ϵ, we can create multiple topological spaces and let the Betti numbers $b_i(\Omega_\epsilon)$ quantify the structure of the topological space. The number b_0 represents the approximate number of connected components and b_1 the number of circles or holes.

In persistent homology, we study a spectrum of neighborhood sizes. For a *known* generating group, we would know its Betti numbers, and may ask which (if any) ϵ yield the given Betti numbers in the visualization point set. This allows us to consider multiple thresholds of our measure (2) of symmetry.

Barcodes. A broken symmetry is defined by the maximum of the normalized pairwise distances $B_{\max}$ being greater than a threshold B_t. This we can represent by a bar ranging from zero to $B_{\max}$ that visualizes the birth and death of symmetry. Stacking such bars (as in Fig. 2) yields a *barcode*. This lets us inspect the sensitivity of a chosen threshold for multiple models visually as each bar corresponds to a model [10]. The 'sharper' the transition from short bars to long bars is, the more robust the conclusion is. The barcode in Fig. 2 suggests that a

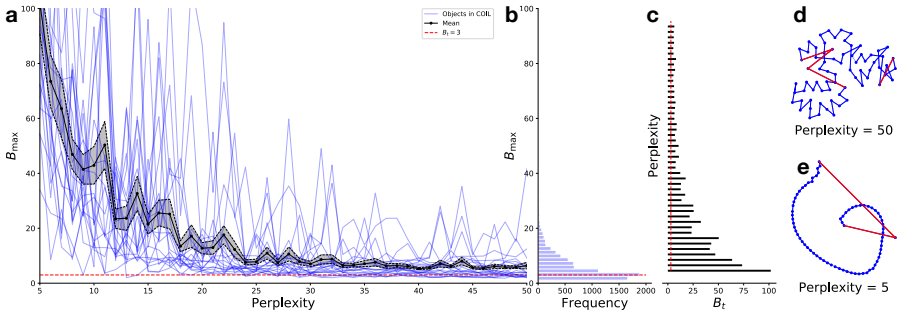


Fig. 3. (a) $B_{\max}$ vs. the perplexity for t-SNE. Each blue line represents the mean over 30 repeats for an object in COIL-20. The dotted, red line marks $B_t = 3$ and the black lines represent mean and standard error over all objects. (b) Histogram of models. (c) Barcode for the mean (black lines in (a) of objects. (d) Latent space in model with perplexities 50 ($B_{\max}$ is small). (e) Latent space in model with perplexities 5 ($B_{\max}$ is large).

choice of $B_t = 3$ is robust as any value in $B_t \in [2, 8]$ yields the same conclusions. For quantitative comparisons across experiments we consistently use $B_t = 3$ though this may be suboptimal for some models.

3 Experiments

We consider four methods representing the spectrum of visualization techniques:

t-SNE matches an exponential distribution of pairwise distances in data space with a t-distribution of pairwise distances in the latent space [27]. The visualization is controlled by a *perplexity* parameter that quantifies the effective number of neighbors used in the exponential distribution over pairwise distances. This is a randomized model as implemented in scikit learn [22].

TriMap [2] is a recent method that relies on an elaborate triplet weighting scheme such that point triplets are weighted with their pairwise distance before obtaining the final triplet weight $\omega_{ijk} = \zeta_\gamma\left(\delta + \tilde{\omega}_{ijk}/\omega_{\max}\right)$. Here $\zeta_\gamma(u) = \log(1 + \gamma u)$, where the the *locality* parameter γ is said to place focus on either local or global structure. The method is randomized and experiments were performed using software provided by Amid and Warmuth [2] where the default value is $\gamma = 500$.

Kernel principle component analysis (kPCA) [25] extends classic PCA through the kernel trick. We use the squared exponential kernel $k(x_i, x_j) = \exp\left(-||x_i - x_j||^2/\lambda\right)$, which is controlled by the *scale* parameter λ. The model is deterministic and experiments were performed using scikit learn [22].

Gaussian process latent variable model (GPLVM) [17] visualizes data using a latent representation with a Gaussian process prior with covariance function $k_{ij} = \theta \exp\left(-1/2\|x_i - x_j\|^2\right) + \sigma^2\delta_{ij}$, where x_i, x_j denote latent points.

The model is deterministic for a given *initial condition* of the hyperparameters θ and σ^2, θ_0 and σ_0^2. Experiments were performed using Pyro [4].

In all experiments, we vary method parameters over a large range, and randomized methods are repeated multiple times and reported numbers are averages. Experimentally, we focus on the most elementary symmetry of interest: *the rotation group*. We consider images from (1) *COIL-20* where objects are rotated 360° in 72 steps and (2) *MNIST* where we synthetically rotate images with up to 360° and 5% Gaussian noise is added to the pixel intensities. We perform a detailed analysis of each model's behavior, and quantitatively compare and summarize in Sec. 3.5.

3.1 t-Distributed Stochastic neighborhood Embedding (t-SNE)

To investigate possible symmetry breaking in t-SNE, we fit 30 t-SNE models to images of each COIL-20 object over a large span of *perplexity* parameters. We measure $B_{\max} = \max B_{ij}$ and report averages over the 30 models (the blue lines in Fig. 3a)[1]. As perplexity increase, $B_{\max}$ becomes smaller. This is to be expected as perplexity controls the smoothness of the t-SNE model. In 73% of all models, we observe broken symmetry ($B_{\max} > 3$). The barcodes reveal that this percentage is not particular sensitive to the choice of threshold (the red dotted line correspond to $B_t = 3$). On MNIST, we observe a similar pattern (omitted due to space constraints) with 96.5% of all models having a broken symmetry.

In our experience, t-SNE tends to amplify small gaps in the data, leading to broken symmetry. This is linked to the 'spurious clustering' effect observed by Amid and Warmuth [1]. We generally observe that random initialization of t-SNE seems to better preserve symmetries than initialization by other methods such as PCA or Isomap. This former approach requires multiple restarts and choosing the embedding with lowest KL divergence.

3.2 TriMap

Amid and Warmuth [2] developed TriMap motivated by the spurious clustering effect in t-SNE, and we hypothesized that TriMap would lead to less symmetry breaking. However, the evidence in Fig. 4 does not support that conclusion. As before, each blue line shows the average $B_{\max}$ for 30 randomly initialized models for each object in COIL-20 over a wide span of the γ parameter. Here 77% of all models are estimated to show broken symmetry, which is roughly on par with t-SNE. The barcode indicates that the choice of threshold is robust, though we find some inter-object variability (omitted). Our findings for MNIST are similar with 93.32% estimated symmetry breaking.

3.3 Kernel Principal Component Analysis (kPCA)

In kPCA, we examine symmetries as a function of the kernel scale parameter λ. The barcode (Fig. 5) shows the robustness of the conclusion of preserved

[1] $B_{\max}$ axis is cut off intentionally as the value for some object diverge

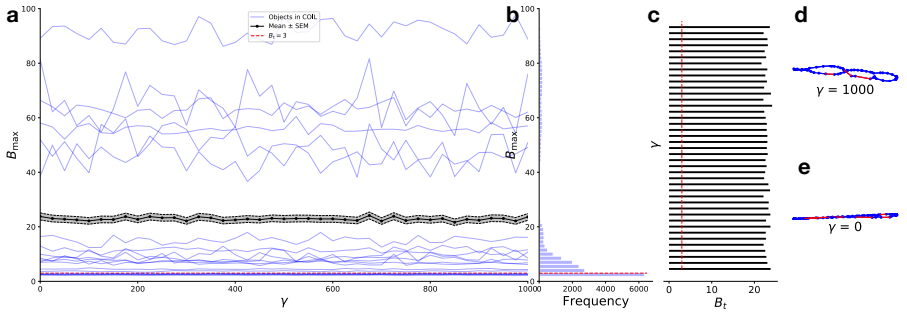


Fig. 4. (a) $B_{\max}$ vs. the locality parameter γ for TriMap. Each blue line represents the mean over 30 repeats for an object in COIL-20. The dotted, red line marks $B_t = 3$ and the black lines represent mean and standard error over all objects. (b) Histogram of models. (c) Barcode for the mean (black lines in (a) of objects. (d) Latent space in model with $\gamma = 1000$. (e) Latent space in model with $\gamma = 0$.

symmetry for the mean across COIL-20 objects. For large values of the scale parameter, the conclusion is robust as B_t can vary, but for smaller values, our conclusions become sensitive to the specific choice of B_t.

In the non-linear regime (small values of λ), B_{median} (1) is driven to small values (Fig. 5d) and $B_{\max}$ diverges. In the linear regime (large values of λ), the model approaches PCA which explains the flattening (Fig. 5e).

In 42% of models, we observe broken symmetry and note that five objects in COIL-20 give rise to broken symmetries: Object 2 (wooden toy), object 16 (round bottle), object 16 (ceramic vase), object 18 (tea cup) and object 20 (round container). Of these, four are rotationally symmetric in the plane of rotation, supporting our hypothesis that additional symmetry can induce symmetry breaking.

On MNIST data, the rate of broken symmetries was 7.23%. One possible explanation for this reduction, is that if PCA on the MNIST data does not induce symmetry breaking then fewer models will break the symmetry because kPCA converges to PCA in the linear regime.

3.4 Gaussian Process Latent Variable Model (GPLVM)

We investigated the GPLVM design space by varying the initial values of the kernel hyperparameters, θ_0 and σ_0^2 all with identical initialization of the latent space (isomap [26]). In Fig 6a, θ_0 is fixed and σ_0^2 is varying and in Fig 6b, σ_0^2 is fixed while θ_0 varies. An interesting thing to notice is while we mostly get consistent results, sometimes a small change in the initial condition induces a large change in the $B_{\max}$ leading to somewhat complex behavior.

The loss is often an indicator of broken symmetry as we saw with the KL divergence for t-SNE. If the parameter space contains symmetry-preserving models then these generally have lower loss than models that break symmetry.

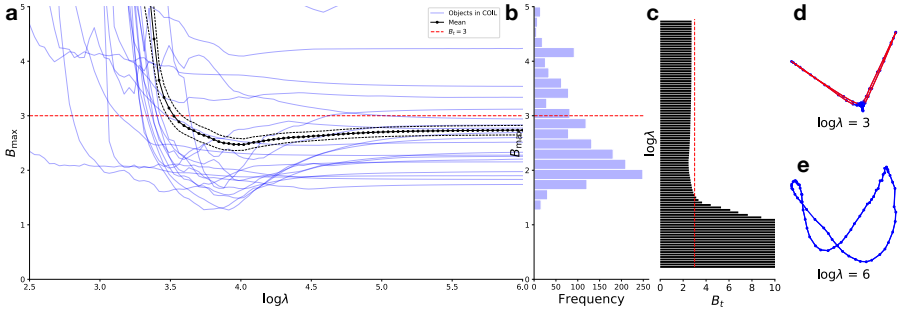


Fig. 5. (a) $B_{\max}$ vs. the scale parameter λ for kPCA. Each blue line represents an object in COIL-20. The dotted, red line marks $B_t = 3$ and the black lines represent mean and standard error over all objects. (b) Histogram of models. (c) Barcode for the mean of objects (black lines in (a)). (d) Latent space of model with $\log \lambda = 3$ ($B_{\max}$ is large). (e) Latent space of model with $\log \lambda = 6$ ($B_{\max}$ is small).

The hyperparameters θ and σ^2 converge to the final values independent of the model preserving the symmetry. This means that the difference in loss between symmetry-preserving and symmetry-breaking models must be accounted for by the latent variables. It also means that it is not possible to detect a broken symmetry from the optimized hyperparameters but rather, one have to consider the latent variables to detect a broken symmetry.

Like in kPCA, we find broken symmetries in the most symmetric objects. In the GPLVM, this is linked to the choice of initialization of the latent space. Overall, we found broken symmetries in 65.48% of the models and similarly in MNIST (46.32%).

3.5 Summary of experiments

We found broken symmetry in all models with a high prevalence as summarized in Fig. 7. Note that we did not tune the parameters but varied important parameters across large ranges and used default parameters for others.

All objects in COIL-20 are indeed symmetric in data space according to our estimator. One may expect that high-level features may be less susceptible to broken symmetry than raw data. To investigate we extracted features using ResNet18 [13] and found no broken symmetries in the extracted features and no consistent, significant difference when looking at symmetry in the models trained on extracted features. We noticed that the most symmetric objects generally experienced more broken symmetry across models.

4 Related Works

Data visualization is important at many steps in the machine learning process. Visualization is used exploratively to form hypotheses [3], for understanding latent representations in supervised learning [8] and generative models [9].

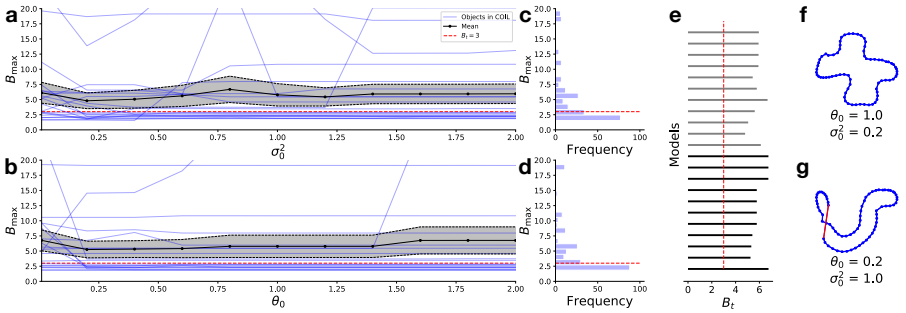


Fig. 6. (a) $B_{\max}$ vs. the initial value of the noise variance σ_0^2 for GPLVM. (b) $B_{\max}$ vs. the initial value of the kernel variance θ_0 for GPLVM. Each blue line represents an object in COIL-20. The dotted, red line marks $B_t = 3$ and the black lines represent mean and standard error over all objects. (a) Parameter space in σ_0^2 with fixed θ_0. (b) Parameter space in θ_0 with fixed σ_0^2. (c) Histogram of models in a. (d) Histogram of models in b. (e) Barcode for the mean of objects (black lines in (a) and (b)). (f) Latent space of model with $\theta_0 = 1$ and $\sigma_0^2 = 0.2$. (g) Latent space of model with $\theta_0 = 0.2$ and $\sigma_0^2 = 0.2$.

The desiderata of visualization are discussed by Kaski et al. [16] and Venna et al. [28], who argue that visualizations should be trustworthy, meaning that samples appearing similar (e.g., neighbors) in the visualization should be similar in a physical sense. Also, they point out that data points close in a physical sense should be close in visualization. They noted the similarity with the concepts of precision and recall in information retrieval. Our concept of broken symmetry is related to the "recall" dimension, i.e., data that are physical neighbors, should also be visualized as such. The precision and recall criteria together measure the faithfulness of the visualization, see also Najim [19] for a related quantitative measure of the preservation of neighborhood relations in visualizations.

The immensely popular visualization scheme t-SNE [27] is constructed with the aim of representing both global and local structure. The original motivation for t-SNE included a critique of its predecessor SNE [14] for creating crowded visualizations, i.e., visualizations that did not show a clear separation of known clusters. Crowding is closely related to the trustworthiness concept of [16, 28]. By using a long-tailed distribution of the representations, t-SNE aims to fix the crowding problem. However, this emphasis of local dissimilarity comes at a price as noted in [18], simple manifolds like lines and sheets are broken apart in clusters. These clustering problems are examples of broken symmetry in our definition. Motivated by the problem of over-fitting cluster structure Amid and Warmuth [2] proposed TriMap. We observed, however, that TriMap cannot heal the problem of broken symmetry.

For detecting symmetries, we used topological data analysis [5], specifically persistent homology. Using this, we examined all values of thresholds simultaneously rather than study just a single threshold. Conveniently, Cohen-Steiner

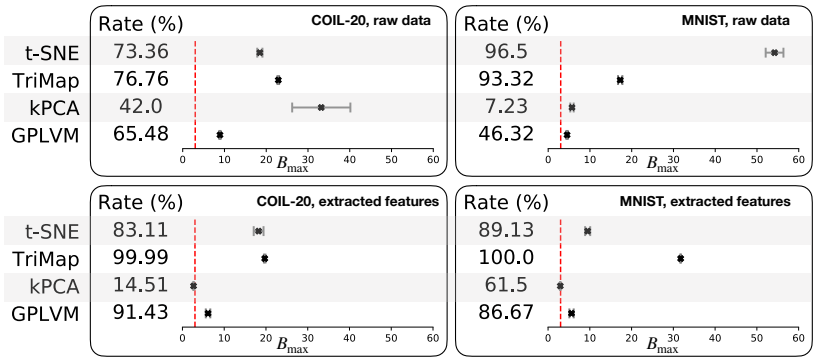


Fig. 7. Each panel shows the rate of broken symmetries in percent at $B_t = 3$ with the mean and standard error plotted displayed on the axis for t-SNE, TriMap, kPCA, and GPLVM. *Top, left pane*) Summary of results on COIL-20. *Top, right pane*) Summary of results on MNIST. *Bottom, left pane*) Summary of results on features extract from COIL-20 using ResNet-18. *Bottom, right pane*) Summary of results on features extracted from MNIST using ResNet-18.

et al. [6] showed that the persistent homology tool is robust under pertubations of the data. [23] used persistent homology in its classical form whereas we have adapted it slightly as we knew which Betti numbers were required to preserve the symmetry. Our work exploits the coordinate and deformation invariances in topology and these properties aid in detecting symmetries as various deformations of the "circle" graph.

5 Discussion

We have investigated to which extend common visualization techniques are able to preserve simple symmetries, and have largely found the answer to be negative.

5.1 Empirical findings

We have investigated four popular algorithms that also represent different branches of the literature, namely t-SNE [27], TriMap [2], kPCA [25] and the GPLVM [17]. We have performed a systematic study of the influence of parameter choices in these methods by training more than 85.000 models over a wide parameter span. To quantitatively summarize these models' performance, we have introduced a simple scheme for detecting whether known symmetries are broken. Tools from persistent homology verify that this scheme is generally reliable, with some deviations for kPCA (see below).

t-SNE was found to be particularly sensitive to local optima and generally we found a need for multiple restarts. Fortunately, we generally observe that smaller KL reported values imply less symmetry breaking. Even with such mechanisms

in place, we still see an overwhelming number of broken symmetries. Symmetry breaking can, to some extend, be reduced by increasing the perplexity parameter, but this also limits the flexibility and expressivity of the model.

TriMap, which was developed in part to alleviate problems with t-SNE, overall had comparable behavior to t-SNE with regards to broken symmetry. The γ parameter, that controls the trade-off between capturing local or global structure, was found to have practically no effect with regards to symmetry breaking. We did not expect this, but have manually verified that broken symmetry is prevalent across large spans of γ.

kPCA was in a sense the most successful method according to our estimator. Kernel PCA, however, has a tendency to collapse points on to each other when mapping only two latent dimensions in the non-linear regime leading to strong symmetry breaking. On the other hand, kPCA reduces to conventional PCA in the limit of large kernel length scales, showing less symmetry breaking.

GPLVM was generally found to be sensitive to choice of initial parameters. While we have found it helpful to consider multiple restarts and choosing the model with highest likelihood, broken symmetries remain rather prevalent.

High-level features. One could suspect that symmetries are broken more commonly when working with raw data than with high-level abstract features, e.g., as those extracted by deep neural networks. We found no broken symmetries directly in the high-level features though when applying visualization algorithms, the prevalence was indeed high.

Summary. Our general finding is that symmetries are broken consistently across the studied methods. It is generally possible to manually tweak parameters to enforce that a known symmetry remains intact, but such strategies are not possible when the symmetry is unknown[2], e.g. for knowledge discovery. We also note that default parameters of publicly available implementations of the studied methods generally perform poorly with regards to broken symmetry.

5.2 Faithful representations

At the heart of our study is the quest for *faithful representations*, i.e. representations that reflect the underlying physics of the data generating process. These have wider applicability than just visualization as studied here. For instance, a representation that is not faithful will most likely not result in a fair prediction. A broken symmetry can be viewed as model that violates the Lipschitz continuity condition. *Individual fairness* [7] can then no longer be ensured as *similar individuals should be treated similarly.*

Similar statements can be made for interpretable models, where 'almost discontinuous' models are generally difficult to interpret. From a purely predictive point of view, it is strictly not required that representations are faithful, though there is some evidence in that direction [24].

[2] It should be emphasized that while we consider known symmetries, we only do so in order to make quantitative statements.

Finally, we note that visualization may be particularly sensitive to symmetry breaking as we tend to embed onto $\mathbb{R}^2$. While it is well-known that only few graphs (namely the *planar* ones) can be embedded in $\mathbb{R}^2$, then *all* graphs can be embedded in $\mathbb{R}^3$ [21]. This suggests that symmetries are likely to be broken when data is forced onto a two-dimensional view (as is often the case in visualization), and indeed our experiments indicate that symmetry breaking is less frequent when embedding into three or more dimensions (omitted due to space constraints).

5.3 Concluding remarks

We have here pointed to a previously unnoticed problem in visualizations, namely *broken symmetries*. Through a systematic study of more than 85.000 trained models, we have found an alarming rate at which even the most simple symmetries are spontaneously broken during data visualizations. This suggest a need for both new methods that can reliably visualize high-dimensional data, but also for more systematic and quantitative evaluations of visualization techniques.

We have purposefully not investigated more complex symmetries as these raise complications that are beyond existing techniques; for instance, the two-dimensional torus is mathematically impossible to embed in $\mathbb{R}^2$ without breaking the underlying symmetry. This calls for visualization techniques that embed onto curved surfaces in order to preserve symmetries, just as we use a sphere when we visualize global geoinformatics patterns.

Acknowledgements. This work received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (757360). SH were supported in part by a research grant (15334) from VILLUM FONDEN.

References

1. Amid, E., Warmuth, M.K.: A more globally accurate dimensionality reduction method using triplets. arXiv:1803.00854 [cs] (Mar 2018)
2. Amid, E., Warmuth, M.K.: TriMap: Large-scale Dimensionality Reduction Using Triplets. arXiv:1910.00204 [cs, stat] (Oct 2019)
3. Arora, S., Hu, W., Kothari, P.K.: An analysis of the t-sne algorithm for data visualization. arXiv preprint arXiv:1803.01768 (2018)
4. Bingham, E., Chen, J.P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., Goodman, N.D.: Pyro: Deep universal probabilistic programming. The Journal of Machine Learning Research 20(1), 973–978 (Jan 2019)
5. Carlsson, G.: Topology and data. Bulletin of the American Mathematical Society 46(2), 255–308 (Jan 2009)
6. Cohen-Steiner, D., Edelsbrunner, H., Harer, J.: Stability of Persistence Diagrams. Discrete & Computational Geometry 37(1), 103–120 (Jan 2007)
7. Dwork, C., Hardt, M., Pitassi, T., Reingold, O., Zemel, R.: Fairness Through Awareness. arXiv:1104.3913 [cs] (Nov 2011)

8. Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M., Thrun, S.: Dermatologist-level classification of skin cancer with deep neural networks. Nature 542(7639), 115–118 (2017)
9. Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J., Greenspan, H.: Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. Neurocomputing 321, 321–331 (2018)
10. Ghrist, R.: Barcodes: The persistent topology of data. Bulletin of the American Mathematical Society 45(1), 61–75 (2008)
11. Hall, B.C.: Lie Groups, Lie Algebras, and Representations: An Elementary Introduction. Graduate Texts in Mathematics, Springer International Publishing, second edn. (2015)
12. Hatcher, A.: Algebraic topology. Cambridge University Press (2005)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. arXiv:1512.03385 [cs] (Dec 2015)
14. Hinton, G.E., Roweis, S.T.: Stochastic neighbor embedding. In: Advances in Neural Information Processing Systems. pp. 857–864 (2003)
15. Huber, P.J.: Robust statistics, vol. 523. John Wiley & Sons (2004)
16. Kaski, S., Nikkilä, J., Oja, M., Venna, J., Törönen, P., Castrén, E.: Trustworthiness and metrics in visualizing similarity of gene expression. BMC bioinformatics 4, 48 (Oct 2003)
17. Lawrence, N.D.: Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models. Journal of Machine Learning Research p. 34 (2005)
18. Linderman, G.C., Steinerberger, S.: Clustering with t-SNE, provably. arXiv:1706.02582 [cs, stat] (Jun 2017)
19. Najim, S.A.: Information visualization by dimensionality reduction: a review. Journal of Advanced Computer Science & Technology 3(2), 101 (2014)
20. Nene, S.A., Nayar, S.K., Murase, H.: Columbia Object Image Library (COIL-20). Technical Report CUCS-006-96 p. 6 (1996)
21. Nishizeki, T., Chiba, N.: Planar graphs: Theory and algorithms. Elsevier (1988)
22. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830 (2011)
23. Pokorny, F.T., Kjellström, H., Kragic, D., Ek, C.: Persistent Homology for Learning Densities with Bounded Support. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25. pp. 1817–1825. Curran Associates, Inc. (2012)
24. Rieger, L., Singh, C., Murdoch, W.J., Yu, B.: Interpretations are useful: penalizing explanations to align neural networks with prior knowledge. arXiv preprint arXiv:1909.13584 (2019)
25. Schölkopf, B., Smola, A., Müller, K.R.: Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Neural Computation 10(5), 1299–1319 (1998)

26. Tenenbaum, J.B.: A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science 290(5500), 2319–2323 (Dec 2000)
27. van der Maaten, L., Hinton, G.: Visualizing Data using t-SNE. Journal of machine learning research pp. 2579–2605 (2008)
28. Venna, J., Peltonen, J., Nybo, K., Aidos, H., Kaski, S.: Information Retrieval Perspective to Nonlinear Dimensionality Reduction for Data Visualization. Journal of Machine Learning Research 11(13), 451–490 (2010)

Revisiting Active Sets for Gaussian Process Decoders

Pablo Moreno-Muñoz* **Cilie W. Feldager*** **Søren Hauberg**
Section for Cognitive Systems
Technical University of Denmark (DTU)
{pabmo,cife,sohau}@dtu.dk

Abstract

Decoders built on Gaussian processes (GPs) are enticing due to the marginalisation over the non-linear function space. Such models (also known as GP-LVMs) are often expensive and notoriously difficult to train in practice, but can be scaled using variational inference and inducing points. In this paper, we revisit active set approximations. We develop a new stochastic estimate of the log-marginal likelihood based on recently discovered links to cross-validation, and we propose a computationally efficient approximation thereof. We demonstrate that the resulting stochastic active sets (SAS) approximation significantly improves the robustness of GP decoder training, while reducing computational cost. The SAS-GP obtains more structure in the latent space, scales to many datapoints, and learns better representations than variational autoencoders, which is rarely the case for GP decoders.

1 Introduction

Generative models can be viewed as regression models from unknown inputs. That is, we assume $\boldsymbol{x} = f(\boldsymbol{z})$, where f is an unknown mapping from latent variables $\boldsymbol{z}$ to observations $\boldsymbol{x}$. Given the inherent difficulty of this task, it is perhaps sensible to marginalize the unknown mapping f to avoid the brittleness of point estimates. This is the driving idea in the *Gaussian process latent variable* (GP-LVM) (Lawrence, 2005), which places a Gaussian process (GP) prior over the unknown mapping f and marginalizes accordingly. This contrasts contemporary generative models that predominantly operate with point-estimates of f (Kingma and Welling, 2013; Sohl-Dickstein et al., 2015). While conceptually elegant, the GP-LVM is, however, notoriously difficult to train and the conceptual benefits are often not realized in practice.

Exact inference involves computing the marginal likelihood, but (like other GP methods) its cubic complexity in the number of observations $\mathcal{O}(N^3)$ limits the scalability of the GP-LVM. However, the idea of marginalizing the decoder is sufficiently attractive to motivate the development of scalable and reliable training techniques: Its Bayesian formulation (Titsias and Lawrence, 2010) variationally integrates out the latent variables $\boldsymbol{z}$ to obtain an evidence lower bound. Using auxiliary inducing variables, Snelson and Ghahramani (2006) expanded GP regression, which is also applicable in the unsupervised learning setting (i.e. the GP-LVM).

However, considering inducing variables here involves dangers. First, the convergence of inducing points is well-studied in the supervised GP scenario, where *inputs are fixed*, but it differs from the unsupervised case where the *inputs are estimated*. Bauer et al. (2016) notes that even in the supervised setting, inducing points are "*not completely trivial to optimise, and often tricks [...] are required*", and we hypothesize that this is further complicated in the unsupervised setting where we optimize both latent and inducing variables while they interact.

In this paper, we revisit *active sets* for scaling GP decoders, a sparse approximation predominantly used before the seminal work of Snelson and Ghahramani (2006). From a practical viewpoint, active

*Equal contribution.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

sets are fixed inducing variables that belong to the training dataset. We make links between such active sets and cross-validation, allowing us to lean on a recent result from Fong and Holmes (2020) which, in turn, links cross validation and the log-marginal likelihood. We show how these links allow for a stochastic estimate of the log-marginal likelihood, and that active sets can be seen as a computationally efficient approximation of this. Practically, this amounts to repeatedly and randomly sampling active sets rather than trying to find the optimal active set. We denote this framework as *stochastic active sets (SAS)*. We demonstrate that SAS consistently results in significantly better-fitted GP decoders over models trained using inducing points.

Historical remarks. Methods based on subsets of data diminish the computational demand and were first introduced in a GP context in the foundational work on sparse approximations by Smola and Bartlett (2001). Back then, Quiñonero-Candela and Rasmussen (2005) had already pointed out the main difficulties behind the optimal selection of subsets:

> "*Traditionally, sparse models have very often been built upon a carefully chosen subset of the training inputs. [...] In sparse Gaussian processes it has also been suggested to select the inducing inputs* $\mathbf{X_u}$ *from among the training inputs. Since this involves a prohibitive combinatorial optimization, greedy optimization approaches have been suggested [...]. Recently, Snelson and Ghahramani (2006) have proposed to relax the constraint that the inducing variables must be a subset of training/test cases, turning the discrete selection problem into one of continuous optimization.*"

This explains how inducing variables reshaped the Gaussian process community, effectively banishing other subset-based methods. Our work builds on the advances made in stochastic optimization in the time since active sets were left behind. We show a third way over those considered by Quiñonero-Candela and Rasmussen (2005): instead of *optimizing* the active set, we *average* with respect to it. This simplifies matters notably and makes them more robust.

To justify our approach, we establish a link between active sets and cross validation (CV). The latter has a long history for model selection in GPs, dating at least to the seminal work of Wahba (1990). For probabilistic models, Rasmussen and Williams (2006) point to the utility of CV variants within negative log-probabilities. Building on results from Fong and Holmes (2020) linking CV and log-marginal likelihoods, we argue that, for GP-LVMs, active sets combine more gracefully with stochastic optimization. The remainder of this paper elaborates on this viewpoint and demonstrates it empirically.

2 Gaussian Process Decoders

The Gaussian process latent variable model (GP-LVM) (Lawrence, 2005) defines a *decoder* which is a non-linear mapping[2] $\boldsymbol{x} = f(\boldsymbol{z})$ from the latent space $\mathcal{Z} \in \mathbb{R}^Q$ to observation space $\mathcal{X} \in \mathbb{R}^D$. The prior on this map is a Gaussian process (GP) so it is drawn like $f \sim \mathcal{GP}(0, k_{\boldsymbol{\theta}}(\cdot,\cdot))$, where $k_{\boldsymbol{\theta}}$ is the covariance function or kernel and $\boldsymbol{K}_{NN}$ denotes the evaluated kernel function so the i,jth element of $\boldsymbol{K}_{NN}$ equals $k_{\boldsymbol{\theta}}(z_i, z_j)$. For clarity, we omit the dependence on covariance function parameters, $\boldsymbol{\theta}$.

The original version of the GP-LVM starts from one-dimensional observations $\boldsymbol{x} = \{\boldsymbol{x}_n\}_{n=1}^N$ and latent variables $\boldsymbol{z} = \{\boldsymbol{z}_n\}_{n=1}^N$, and factorises the joint distribution of the model as $p(\boldsymbol{x}, f|\boldsymbol{z}) = p(\boldsymbol{x}|f,\boldsymbol{z})p(f|\boldsymbol{z})$. Here the conditional distributions correspond to the likelihood model and the prior

$$p(\boldsymbol{x}|f,\boldsymbol{z}) = \prod_{n=1}^{N} \mathcal{N}(\boldsymbol{x}_n|f(\boldsymbol{z}_n), \sigma^2), \qquad p(f|\boldsymbol{z}) = \mathcal{N}(f(\boldsymbol{z})|0, \boldsymbol{K}_{NN}). \tag{1}$$

When the data dimensionality is $D>1$, the model factorises across dimensions, and we have different mappings f per d^{th} feature. One of the principal assumptions of the GP-LVM is that the prior $p(f)$ regularises the smoothness of all mappings equally, so we only consider one *kernel*. This assumption can be relaxed if needed, but at increased computational cost and with more learnable hyperparameters.

[2]We use $\{\boldsymbol{x}, \boldsymbol{z}\}$ to denote observations and latent variables respectively, since we do not consider inducing variables. Notice that Lawrence (2005) use the notation $\{\boldsymbol{y}, \boldsymbol{x}\}$.

Mapping marginalization. A GP prior over the non-linear decoder f allows for marginalisation of f to obtain a closed-form expression of the marginal likelihood of the GP-LVM

$$p(\boldsymbol{x}|\boldsymbol{z}) = \int p(\boldsymbol{x}|f,\boldsymbol{z})p(f|\boldsymbol{z})\mathrm{d}f = \mathcal{N}(\boldsymbol{x}|0,\boldsymbol{K}_{NN}+\sigma^2\mathbb{I}).$$

On a log-scale, this gives the following objective function (Lawrence, 2004), which can be optimized w.r.t. both hyperparameters $\boldsymbol{\theta}$ and latent variables $\boldsymbol{z}$

$$\mathcal{L} = -\frac{DN}{2}\log 2\pi - \frac{D}{2}\log|\boldsymbol{K}_{NN}+\sigma^2\mathbb{I}| - \frac{1}{2}\mathrm{tr}(\left(\boldsymbol{K}_{NN}+\sigma^2\mathbb{I}\right)^{-1}\boldsymbol{x}\boldsymbol{x}^\top). \tag{2}$$

The difficulties of training the GP-LVM using this objective function are evident above, as the evaluation cost grows cubically with the number of observations N. Furthermore, notice that observations $\boldsymbol{x}$ are no longer independent (in contrast with Eq. 1) once f is integrated out. The log-marginal likelihood will be the starting point for our approach in Sec. 3.

Bayesian extension. In the seminal works of Lawrence (2004, 2005), the GP-LVM is first derived as a non-linear extension of probabilistic principal component analysis (PPCA) (Tipping and Bishop, 1999). Considering the *isotropic* prior on the latent variables $\boldsymbol{z}$, such that $p(\boldsymbol{z}_n) = \mathcal{N}(\boldsymbol{z}_n|0,\mathbb{I})\ \ \forall \boldsymbol{z}_n \in \boldsymbol{z}$, the general idea is to optimize them rather than introducing marginalization. From a *full* probabilistic perspective, one could also be interested in the posterior distribution over $\boldsymbol{z}$, which leads to using Bayesian inference for the GP-LVM approach. This is the driving idea of Titsias and Lawrence (2010), where variational methods are introduced. In particular, latent variables are not easy to marginalize, mainly due to their presence in the kernel mappings, so a lower-bound on the log-marginal likelihood $\log p(\boldsymbol{x}) = \log \int p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})\mathrm{d}\boldsymbol{z}$ of the model is derived.

So far, the Bayesian GP-LVM model (Titsias and Lawrence, 2010) has been considered as the standard methodology to apply GPs to large unsupervised datasets with $10^4{-}10^6$ observations, e.g. in regression (Bui and Turner, 2015), classification (Gal et al., 2015) and representation learning (Märtens et al., 2019) tasks.

3 Stochastic Active Sets

Our key objective is a computationally efficient estimate of the log-marginal likelihood in Eq. 2, as this is known to be a good measure of generalization performance (Rasmussen and Ghahramani, 2000; Germain et al., 2016). Another popular measure of generalization performance is *cross validation* (CV) (Geisser and Eddy, 1979; Vehtari and Lampinen, 2002), which is arguably mostly used outside the realm of Bayesian models. Recently, Fong and Holmes (2020) linked these two measures, effectively showing that the marginal likelihood is equivalent to the average over exhaustive leave-R-out CV scores. In particular, the average is w.r.t. the size of the hold-out set. More precisely, let

$$\mathcal{S}_{\mathrm{CV}}(\boldsymbol{x}|R) = \frac{1}{\mathcal{C}}\sum_{p=1}^{\mathcal{C}}\frac{1}{R}\sum_{n\in\mathcal{R}_p}\log p(\boldsymbol{x}_n|\boldsymbol{x}_{\mathcal{A}_p},\boldsymbol{z}) = \frac{1}{R}\mathbb{E}_{\mathcal{A}_p}\left[\sum_{n\in\mathcal{R}_p}\log p(\boldsymbol{x}_n|\boldsymbol{x}_{\mathcal{A}_p},\boldsymbol{z})\right], \tag{3}$$

denote the leave-R-out CV using log-predictive scoring functions $\log p(\boldsymbol{x}_n|\boldsymbol{x}_{\mathcal{A}_p},\boldsymbol{z})$. Here $\mathcal{A}_p$ denotes the *active set* indices of the training data, such that $\mathcal{A}_p \subset \{1,2,\dots,N\}$ and $\mathcal{R}_p = \{1,2,\dots,N\}\backslash\mathcal{A}_p$ are the remaining hold-out samples. The subscript $p \in \mathcal{C}$ denotes the permutation, and we average over all $\mathcal{C} = \binom{N}{R}$ possible hold-out sets. We use use R to indicate the size of the hold-out set $\mathcal{R}_p$ ($R \equiv |\mathcal{R}_p|$) and let $A \equiv |\mathcal{A}_p| = N - R$. If we average $\mathcal{S}_{\mathrm{CV}}(\boldsymbol{x}|R)$ over all possible sizes of the hold-out set, then Fong and Holmes (2020) has shown that we recover the exact log-marginal likelihood,

$$\log p(\boldsymbol{x}|\boldsymbol{z}) = \sum_{r=1}^{N}\mathcal{S}_{\mathrm{CV}}(\boldsymbol{x}|r) = \mathcal{S}_{\mathrm{CCV}}(\boldsymbol{x}|R) + \mathcal{S}_{\mathrm{PCV}}(\boldsymbol{x}|R). \tag{4}$$

Here, $\mathcal{S}_{\mathrm{PCV}}(\boldsymbol{x}|R) = \mathbb{E}_{\mathcal{A}}[\log p(\boldsymbol{x}_{\mathcal{A}}|\boldsymbol{z}_{\mathcal{A}})]$ and $\mathcal{S}_{\mathrm{CCV}}(\boldsymbol{x}|R) = \sum_{r=1}^{R}\mathcal{S}_{\mathrm{CV}}(\boldsymbol{x}|r)$ is the cumulative CV score, which reduces to a sum of expectations over the predictive factors.[3] Further details on $\mathcal{S}_{\mathrm{PCV}}$ and $\mathcal{S}_{\mathrm{CCV}}$ can be found in the Appendix. Fong and Holmes (2020) used Eq. 4 to argue in favor of using the marginal likelihood over cross-validation for model selection.

[3] We drop the permutation subscript p in $\mathcal{A}$ and $\mathcal{R}$ to avoid cluttered notation.

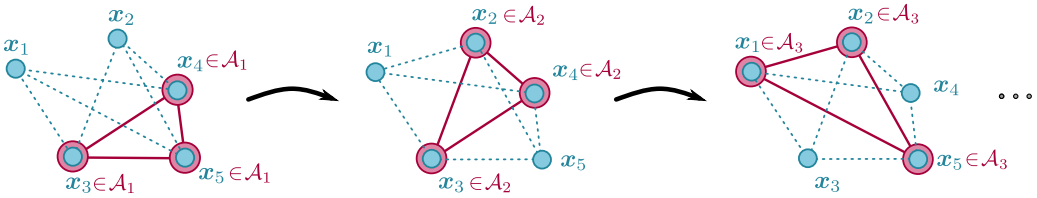


Figure 1: Schematic graphical model of the correlation structure given different permutations p of the active set $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_C$ for five observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_5\}$. Thick red lines indicate that we build *full* covariance densities between observations included in $\boldsymbol{x}_\mathcal{A}$. Only-blue variables are considered conditionally independent among them w.r.t. to the red colored ones. Dashed lines indicate the conditional probability factors $p(\boldsymbol{x}_n|\boldsymbol{x}_\mathcal{A}, \boldsymbol{z})$ that we can *easily* compute.

Stochastic approximation. We take a slightly different view than Fong and Holmes (2020) and argue that Eq. 4 can be the grounds for an efficient stochastic gradient (Robbins and Monro, 1951) of the log-marginal likelihood suitable for training. In the context of GPs, we have that conditional probabilities for $\mathcal{S}_{\text{PCV}}(\boldsymbol{x}|R)$ and $\mathcal{S}_{\text{CCV}}(\boldsymbol{x}|R)$ in Eq. 4 are

$$p(\boldsymbol{x}_\mathcal{A}|\boldsymbol{z}_\mathcal{A}) = \mathcal{N}(\boldsymbol{x}_\mathcal{A}|0, \boldsymbol{K}_{\mathcal{A}\mathcal{A}} + \sigma_n^2\mathbb{I}), \qquad p(\boldsymbol{x}_n|\boldsymbol{x}_\mathcal{A}, \boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{m}_{n|\mathcal{A}}, \boldsymbol{c}_{n|\mathcal{A}}), \tag{5}$$

where we used Eq. 2.22 from Rasmussen and Williams (2006) to obtain

$$\begin{aligned}\boldsymbol{m}_{n|\mathcal{A}} &= \boldsymbol{K}_{n\mathcal{A}}(\boldsymbol{K}_{\mathcal{A}\mathcal{A}} + \sigma_n^2\mathbb{I})^{-1}\boldsymbol{x}_\mathcal{A}, \\ \boldsymbol{c}_{n|\mathcal{A}} &= \boldsymbol{K}_{nn} + \sigma_n^2\mathbb{I} - \boldsymbol{K}_{n\mathcal{A}}(\boldsymbol{K}_{\mathcal{A}\mathcal{A}} + \sigma_n^2\mathbb{I})^{-1}\boldsymbol{K}_{n\mathcal{A}}^\top,\end{aligned}$$

and $\boldsymbol{K}_{\mathcal{A}\mathcal{A}} \in \mathbb{R}^{A\times A}$ has entries $k(\boldsymbol{z}_i, \boldsymbol{z}_j)$ with $\boldsymbol{z}_i, \boldsymbol{z}_j \in \boldsymbol{z}_\mathcal{A}$. The computational cost of $\mathcal{S}_{\text{PCV}}(\boldsymbol{x}|R)$ is $\mathcal{O}(A^3)$, while $\log p(\boldsymbol{x}_n|\boldsymbol{x}_\mathcal{A}, \boldsymbol{z})$ can reuse the matrix inversion $\boldsymbol{K}_{\mathcal{A}\mathcal{A}}^{-1}$ to only have an additional linear cost. Clearly, we can obtain an *unbiased* stochastic estimate of the log-marginal likelihood by *first* uniformly sampling R, *second* making a random split permutation into train and hold-out data $\mathcal{R}$ and *finally* by evaluating

$$\log p(\boldsymbol{x}|\boldsymbol{z}) \approx \sum_{n\in\mathcal{R}} \log p(\boldsymbol{x}_n|\boldsymbol{x}_\mathcal{A}, \boldsymbol{z}) + \log p(\boldsymbol{x}_\mathcal{A}|\boldsymbol{z}_\mathcal{A}), \tag{6}$$

where we remark that the summation can be mini-batched. This approach is equivalent to the decomposition $\log p(\boldsymbol{x}|\boldsymbol{z}) = \log p(\boldsymbol{x}_\mathcal{R}|\boldsymbol{x}_\mathcal{A}, \boldsymbol{z}) + \log p(\boldsymbol{x}_\mathcal{A}|\boldsymbol{z}_\mathcal{A})$ and it also assumes conditional independence among observations $\boldsymbol{x}_n$ for $n \in \mathcal{R}$. This is similar to the standard *active set* approximation (Seeger et al., 2003) previously discussed, and we may think of $\mathcal{A}$ as a *stochastic active set (SAS)*.

However, the estimate of $\log p(\boldsymbol{x}|\boldsymbol{z})$ still has the same computational complexity $\mathcal{O}(N^3)$ as the usual deterministic approach, since we may sample $A = N$ in the worst case. Instead, we propose to make a stochastic approximation where we choose the size of the active set deterministically through a user-specified parameter, such that the computational cost reduces to $\mathcal{O}(A^3)$, as in sparse approximations based on inducing points. This does not need to be unbiased; empirically we find that most often the approximation behaves as a lower bound to the true marginal likelihood, and that, in all instances, it is a rather close approximation. We include a longer discussion on this point in the Appendix and the training methodology using SAS is in Alg. 1.

3.1 Extension for Bayesian GP decoders

We next seek to extend the previous SAS approach to the Bayesian GP-LVM, where we aim to obtain the posterior distribution $p(\boldsymbol{z}|\boldsymbol{x})$. From this perspective, we are interested in marginalising latent variables $\boldsymbol{z}$ to obtain the marginal likelihood $p(\boldsymbol{x})$ of the model.[4] However, this integration is not possible, as latent variables appear non-linearly in the kernel function (Titsias and Lawrence, 2010). Alternatively, we consider the variational inference scheme, where an auxiliary distribution $q(\boldsymbol{z})$ is

[4]Notice that the probabilistic objective function changes between standard and Bayesian versions of the GP-LVM. In the former case, we usually look for $p(\boldsymbol{x}|\boldsymbol{z})$ as the marginal likelihood w.r.t. the mapping f. This is the one usually considered in supervised GP tasks. In the latter, we refer to $p(\boldsymbol{x})$ as the marginal likelihood of the model, since $\boldsymbol{z}$ are also integrated out.

introduced into the formulation. Therefore, we are able to build the evidence lower bound (ELBO) of the model using Jensen's inequality as

$$\log p(\boldsymbol{x}) \geq \int q(\boldsymbol{z}) \log p(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z}) \mathrm{d}\boldsymbol{z} = \mathbb{E}_{q(\boldsymbol{z})} \left[\log p(\boldsymbol{x}|\boldsymbol{z})\right] - \mathrm{KL}\left[q(\boldsymbol{z})||p(\boldsymbol{z})\right], \tag{7}$$

which is equivalent to the one obtained by Titsias and Lawrence (2010, Eq. 8). At this point, the computational cost is $\mathcal{O}(N^3)$, since the ELBO requires evaluating $\log p(\boldsymbol{x}|\boldsymbol{z})$, where we invert $\boldsymbol{K}_{NN}$. The expectation in Eq. 7 can also benefit from a stochastic SAS approximation, just as with inducing points (Hensman et al., 2013). Thus, the lower bound can be approximated as

$$\mathcal{L}_{\text{ELBO}} \approx \sum_{n \in \mathcal{R}} \mathbb{E}_{q(\boldsymbol{z}_n)} \left[\log p(\boldsymbol{x}_n|\boldsymbol{x}_\mathcal{A}, \boldsymbol{z})\right] + \mathbb{E}_{q(\boldsymbol{z}_\mathcal{A})} \left[\log p(\boldsymbol{x}_\mathcal{A}|\boldsymbol{z}_\mathcal{A})\right] - \sum_{n=1}^{N} \mathrm{KL}[q(\boldsymbol{z}_n)||p(\boldsymbol{z}_n)], \tag{8}$$

where we consider *mean-field* VI to factorize the distribution $q(\boldsymbol{z})$. The Bayesian GP-LVM shares high-level similarities with other generative models that marginalize the latent variable according to a simple prior (Rezende and Mohamed, 2015). The proposed SAS approximation (8) scales similarly to mini-batched inducing point approximations, but we will later see that SAS behaves notably better in practice. Algorithmically, the approach is simple, and the summary code is provided in Alg. 2.

Algorithm 1 SAS for GP decoders

Input: Observed data $\boldsymbol{x}$
Parameters: Initialize $\boldsymbol{\theta}$, $\boldsymbol{\phi}$ // $\boldsymbol{\theta}, \boldsymbol{z}$ if NA
for e **in** epochs **do**
  for b **in** batches **do**
    Sample $\boldsymbol{x}_{\text{batch}} \sim \boldsymbol{x}$
    $\boldsymbol{x}_\mathcal{R}, \boldsymbol{x}_\mathcal{A} \leftarrow$ `random_split`$(\boldsymbol{x}_{\text{batch}})$
    if amortized **then**
      Get $\{\boldsymbol{z}_\mathcal{R}, \boldsymbol{z}_\mathcal{A}\} \leftarrow g(\boldsymbol{x}_\mathcal{R}, \boldsymbol{x}_\mathcal{A}|\boldsymbol{\phi})$
    end if
    Compute $\boldsymbol{K}_{\mathcal{A}\mathcal{A}}^{-1}$ // via Cholesky
    Evaluate $\log p(\boldsymbol{x}_\mathcal{A}|\boldsymbol{z}_\mathcal{A})$
    Evaluate $\log p(\boldsymbol{x}_n|\boldsymbol{x}_\mathcal{A}, \boldsymbol{z}),\ \forall \boldsymbol{x}_n \in \boldsymbol{x}_\mathcal{R}$
    Evaluate Eq. 6
    do Adam$(\boldsymbol{\theta}, \boldsymbol{\phi})$ step for $\mathcal{L}$
  end for
end for

NA: Non-amortized.

Algorithm 2 SAS for Bayesian GP decoders

Input: Observed data $\boldsymbol{x}$
Parameters: Initialize $\boldsymbol{\theta}$, $\boldsymbol{\phi}$ // $\boldsymbol{\theta}, \mu, \sigma$ if NA
for e **in** epochs **do**
  for b **in** batches **do**
    Sample $\boldsymbol{x}_{\text{batch}} \sim \boldsymbol{x}$
    $\boldsymbol{x}_\mathcal{R}, \boldsymbol{x}_\mathcal{A} \leftarrow$ `random_split`$(\boldsymbol{x}_{\text{batch}})$
    if amortized **then**
      Get $\mu_{\boldsymbol{z}} \leftarrow g_\mu(\boldsymbol{x}_\mathcal{R}, \boldsymbol{x}_\mathcal{A}|\boldsymbol{\phi})$
      Get $\sigma_{\boldsymbol{z}} \leftarrow g_\sigma(\boldsymbol{x}_\mathcal{R}, \boldsymbol{x}_\mathcal{A}|\boldsymbol{\phi})$
    end if
    Sample $\{\boldsymbol{z}_\mathcal{R}, \boldsymbol{z}_\mathcal{A}\} \sim q(\mu_{\boldsymbol{z}}, \sigma_{\boldsymbol{z}})$ // RT
    Compute $\boldsymbol{K}_{\mathcal{A}\mathcal{A}}^{-1}$ // via Cholesky
    Evaluate $\mathcal{L}$ in Eq. 8
    do Adam$(\boldsymbol{\theta}, \boldsymbol{\phi})$ step for $\mathcal{L}_{\text{ELBO}}$
  end for
end for

NA: Non-amortized, RT: Reparametrization trick.

3.2 The Role of Amortization

Early after the emergence of the seminal GP-LVM (Lawrence, 2005), the lengthy optimization required obtaining a result in which all latent representations $\boldsymbol{z}_n$ became noticeable. An additional consideration is that, while most approaches to non-linear low-dimensionality reduction focus on preserving similarities, the GPLVM does the opposite. This property was initially discussed by Lawrence and Quiñonero-Candela (2006), since in some sense the GP-LVM is *dissimilarity preserving*, such that different observations will generally be represented far away from each other. In practice, we are often more interested in embeddings that reflect the *true* distance between the observed objects in their representations, particularly those that are close together. This observation inspired *back constraints* for locality preservation (Lawrence and Quiñonero-Candela, 2006), which enforces latent variables $\boldsymbol{z}$ to be an explicit function of observations $\boldsymbol{z} = g(\boldsymbol{x}|\boldsymbol{\phi})$ parameterized by $\boldsymbol{\phi}$. This is similar to the stochastic *encoder* applied in variational autoencoders (Kingma and Welling, 2013; Rezende and Mohamed, 2015). This idea was also extended to the VI framework in GP-LVMs by Bui and Turner (2015) using a recognition model, e.g. $q(\boldsymbol{z}_n) = \mathcal{N}(\boldsymbol{z}_n|g_\mu(\boldsymbol{x}_n|\boldsymbol{\phi}), g_\sigma(\boldsymbol{x}_n|\boldsymbol{\phi}))$ and more recently, to accelerate hyperparameter learning in GPs with hierarchical attention networks (Liu et al., 2020).

In our context, we assume the mappings to be neural networks (NNs) like Bui and Turner (2015), referring to them as *amortization* networks. We find such networks accelerate learning very nicely when used in conjunction with SAS. It is also worth noting that amortization has been empirically shown to improve generalization performance (Shu et al., 2018).

4 Related Work

Marginal likelihood approximations were used in GPs (Smola and Bartlett, 2001; Csató and Opper, 2002) before the apparition of pseudo-inputs (Snelson and Ghahramani, 2006) and the associated variational inference framework (Titsias, 2009). In the former case, stochastic approximations to the ELBO were first presented by Hensman et al. (2013, 2015), in line with the Bayesian counterpart of SAS. In terms of active sets, Seeger et al. (2003) empirically observed that the approximation was generally stable for optimisation, even if the size of $\mathcal{A}$ was a small fraction of the training size only. However, they also observed that random selection of active sets led to non-smooth fluctuations, making it difficult to converge through exact gradient ascent. Particularly, the issue with re-selecting of $\mathcal{A}$ motivated Snelson and Ghahramani (2006) as a *smoother* optimization alternative, and we find that SAS also circumvents this problem via stochastic estimates as shown in Sec. 5.

The connection between cross-validation and GPs was already described in Rasmussen and Williams (2006) as an alternative for model selection. However, the equivalence between *exhaustive* CV and the log-marginal likelihood provided in Fong and Holmes (2020) provides a novel perspective that we exploit. Additionally, the notion of *back constraints* has recently been considered in Lalchand et al. (2022) for GP-LVMs with inducing points, where a doubly stochastic formulation is used. More recently, amortization networks have been used to drastically reduce the number of inducing points in supervised GPs.

5 Experiments

In this section, we evaluate the performance of the SAS approach for stochastic learning of GP decoders, both the deterministic GP-LVM (Lawrence, 2004) and its Bayesian counterpart (Titsias and Lawrence, 2010). For this purpose, we consider three different datasets: MNIST (LeCun et al., 1998), FASHION MNIST (Xiao et al., 2017) and CIFAR-10 (Krizhevsky, 2009). For a *fair* comparison of our model with baseline methods, we use the same amortization across models, namely a neural network (three linear layers ReLU activation functions)[5] for all models in all experiments and all GPs use an quadratic exponentiated kernel. In all experiments, the learning rates are set in the range $[10^{-4}, 10^{-2}]$, the maximum number of epochs considered is 300 and we use the ADAM optimizer (Kingma and Ba, 2015). For SAS experiments, we only consider batch sizes greater than the active set size, as this is a requirement for SAS.

Performance metrics of the SAS-GP decoders are given in terms of the negative log-predictive density (NLPD), root mean-square error (RMSE) and mean absolute error (MAE). In all cases, we optimize w.r.t. an approximation to the log-marginal likelihood $\log p(\boldsymbol{x}|\boldsymbol{z})$ in the deterministic scenario or w.r.t. a lower bound on the $\log p(\boldsymbol{x})$ *of the model* in the Bayesian cases. We also provide PYTORCH code that allows for reproducing all experiments and models.[6] We monitor the run-time of convergence as we suspect that rotating active sets (see Fig. 1) across the dataset is a fast way to capture the correlation structure of the regression problem.

5.1 Representation performance

First, we analysed our SAS approach and the approximate optimization procedure on an *unsupervised* version of MNIST, FMNIST and CIFAR-10, where we took the full training corpus for learning two-dimensional latent representations of images. The approximation curves are included in Fig. 2, where we observe convergence in less that 2h runtime in CPU for most cases. For all experiments, we can observe that for larger active set sizes A, the SAS approaches take longer time to complete the 300 epochs, as the computational cost of inversion is higher.

Evaluation with SOTA methods. We also tested the performance of representation learning in the state-of-the-art methods with and without GPs for unsupervised scenarios. Results are shown in Fig. 4. We include a short description of the models considered:

- SAS-GP DECODER — It uses stochastic active sets to approximate the log-marginal likelihood $\log p(\boldsymbol{x}|\boldsymbol{z})$. The training methodology is described in Alg. 1.

[5] Please see the supplementary material for more details.

[6] The code is publicly available in the repository: `https://github.com/pmorenoz/SASGP/` including baselines.

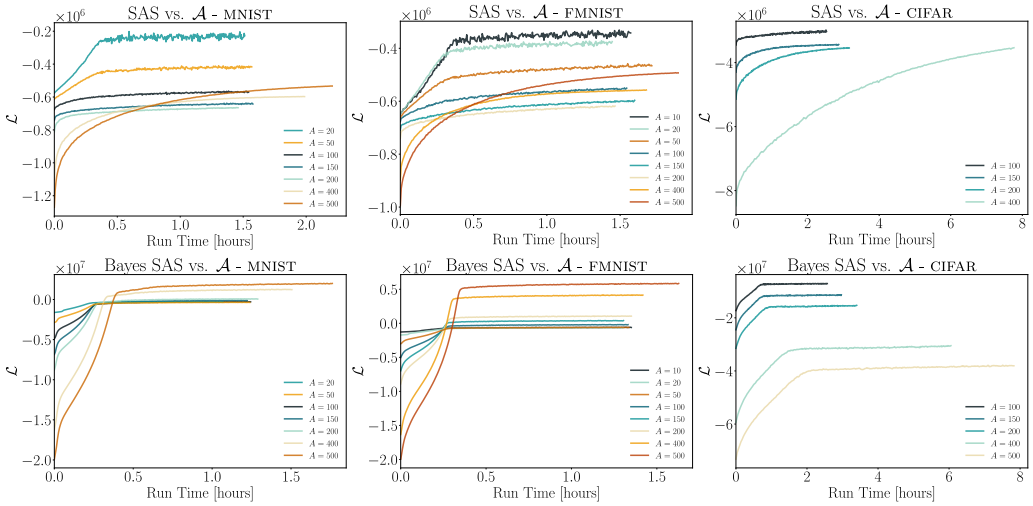


Figure 2: Approximate log-marginal likelihood (**upper row**) for SAS and ELBO curves (**lower row**) for Bayesian SAS. We fix the batch size in Alg. 1 to be $B = 1024$ and study the convergence for different *active set* sizes A. All values in the curves displayed are *per-epoch*.

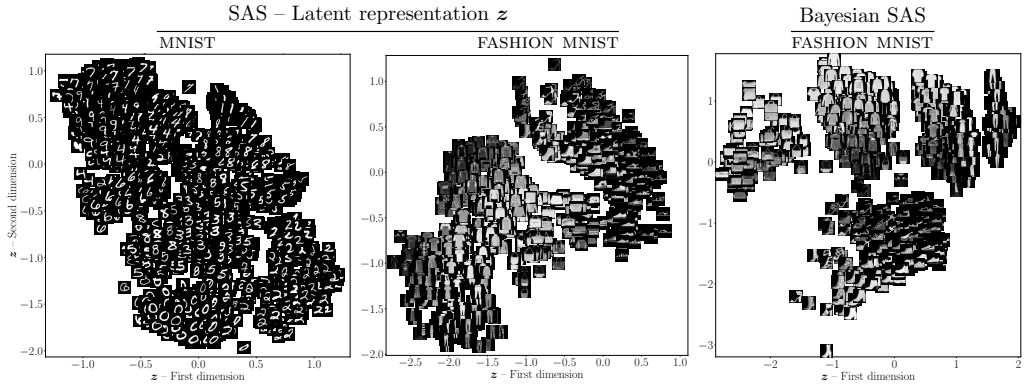


Figure 3: Latent representation in a 2-dimensional space $\mathcal{Z}$ for the 10 MNIST and FMNIST classes learnt with SAS and Bayesian SAS. Notice that the likelihood model of the GP decoder is controlled by a *vanilla* RBF kernel. The examples have been obtained using an *active set* size $A = 800$.

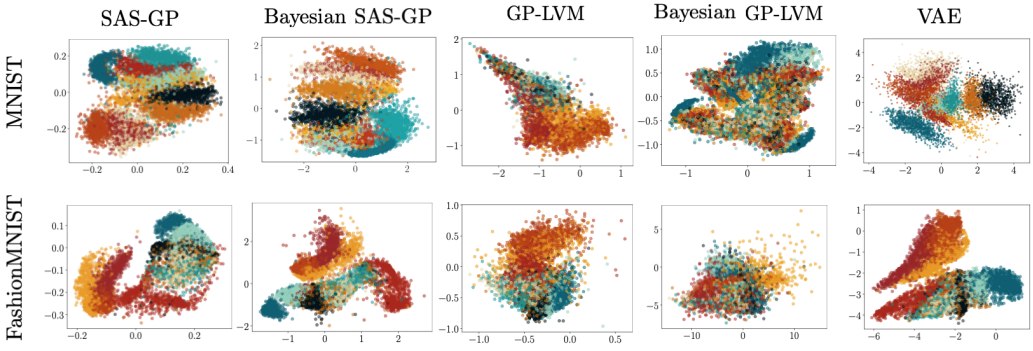


Figure 4: Illustration of latent space mappings $z_n \in \mathcal{Z}$ on test data for different SAS models and baselines (**rows**) and different datasets (**columns**). The models have been trained on *full* MNIST and FMNIST.

- BAYESIAN SAS-GP DECODER — It uses stochastic active sets to approximate the ELBO in Eq. (8) on $\log p(\boldsymbol{x})$. See Alg. 2.
- BAYESIAN GP-LVM — It is based on the model in Titsias and Lawrence (2010). Means and variance parameters are generated by an amortization NN as in Bui and Turner (2015). The model is trained using stochastic variational inference (Hensman et al., 2013).
- GP-LVM — We used the model proposed by Lawrence (2005) enhanced with an NN encoding to the latent space. The model is trained using maximum likelihood (ML).
- VARIATIONAL AUTOENCODER (VAE) — (Kingma and Welling, 2013) The NN encoder has the same architecture as the amortization function used in the SAS-GP models.

For the SAS-GP decoder and GP-LVM, a neural network encodes the latent variables. Likewise, for the Bayesian SAS-GP decoder and the Bayesian GP-LVM, two neural networks each encode the latent means and latent variances. We refer to this encoding as *amortization*. All models use a Gaussian likelihood[7].

Table 1: Comparative metrics for SAS and Bayesian SAS on MNIST, FMNIST and CIFAR-10.

MODEL	SAS			BAYESIAN SAS		
ACTIVE SET SIZE	$A = 100$	$A = 200$	$A = 400$	$A = 100$	$A = 200$	$A = 400$
MNIST / RMSE ↓	2.55 ± 0.98	2.47 ± 0.98	2.41 ± 0.93	2.16 ± 0.02	2.08 ± 0.02	1.99 ± 0.02
MNIST / MAE ↓	1.61 ± 0.97	1.55 ± 0.99	1.51 ± 0.96	1.11 ± 0.02	1.04 ± 0.02	0.96 ± 0.01
MNIST / NLPD ↓	2.99 ± 1.41	2.92 ± 1.38	2.84 ± 1.31	2.33 ± 0.03	2.26 ± 0.02	2.17 ± 0.02
FMNIST / RMSE ↓	2.37 ± 0.95	2.31 ± 0.94	2.25 ± 0.90	1.99 ± 0.17	1.88 ± 0.20	1.85 ± 0.13
FMNIST / MAE ↓	1.48 ± 0.91	1.42 ± 0.91	1.39 ± 0.89	1.11 ± 0.02	1.02 ± 0.03	0.98 ± 0.02
FMNIST / NLPD ↓	2.76 ± 1.33	2.71 ± 1.31	2.65 ± 1.23	2.16 ± 0.18	2.07 ± 0.19	2.04 ± 0.12
CIFAR10 / RMSE ↓	2.66 ± 1.08	2.55 ± 1.06	2.55 ± 1.03	2.74 ± 1.07	2.64 ± 1.08	2.57 ± 1.02
CIFAR10 / MAE ↓	1.77 ± 1.06	1.69 ± 1.06	1.69 ± 1.02	1.84 ± 1.03	1.76 ± 1.05	1.71 ± 1.03
CIFAR10 / NLPD ↓	3.20 ± 1.55	3.07 ± 1.44	3.32 ± 1.89	3.24 ± 1.53	3.14 ± 1.53	3.06 ± 1.45

All metrics are ($\times 10^{-1}$).

5.2 Evaluation metrics

In this section, we are interested in the evaluation of the GP decoder with the *standard* error metrics used for GPs. In Tab. 1 we provide RMSE, MAE and NLPD for the three datasets considered in this experiment. Interestingly, the performance usually improves with larger active set sizes A, as the SAS model captures the underlying correlation of datapoints better and this enhances the approximation. This happens for both deterministic and Bayesian cases. We observe a similar trend for the NLPD metric which is generally better for larger active set sizes A. This can also be noticed in Fig. 2, where loss curves have a better convergence for the lowest A.

Classification accuracy. We are interested in evaluating the structure of the representation. For this purpose, we trained a (one) *nearest neighbour* classifier on the encoded, two-dimensional latent variables and we tested the accuracy using encoded test data. Tab. 2 shows the mean and standard deviations of test the accuracy.

Table 2: Classification accuracy (↑) on 2-dim. latent space $\mathcal{Z}$.

MODEL	MNIST	FMNIST
BAYESIAN SAS-GP DEC. (**ours**)	0.63 ± 0.022	0.63 ± 0.020
BAYESIAN GP-LVM	0.18 ± 0.033	0.24 ± 0.043
VAE	0.54 ± 0.026	0.58 ± 0.008

Runtime and convergence of SAS approximation. The loss curves obtained for the (sparse) Bayesian GP-LVM model and the Bayesian SAS-GP decoder are shown in Fig. 5. We observe that the two methods scale similarly, but that SAS is faster by a notable constant within stochastic optimization. Additionally, we observe the performance of the SAS approximation to closely fit the exact log-marginal likelihood $\log p(\boldsymbol{x}|\boldsymbol{z})$. For the plot in Fig. 5, we computed the exact probability for a subset of MNIST $N{=}40,000$ samples.

[7]Pytorch (Paszke et al., 2019) and Pyro (Bingham et al., 2019) are available on public repositories.

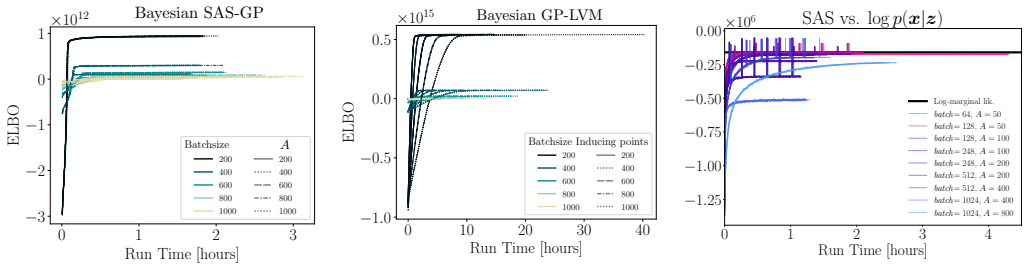


Figure 5: (**Left-center**) ELBO curves for Bayesian SAS-GP and Bayesian GPLVM for different batch-sizes and active set points. (**Right**) SAS loss function compared with the exact log-marginal likelihood computed. In all curves, N=40, 000 samples of MNIST were considered and five different initializations per *batch* and A setup.

6 Conclusion

State-of-the-art representation learning is generally based on neural networks, as this allows for scaling to large datasets. However, often we want reliable uncertainty estimates from the model and we can achieve these with Gaussian process decoders if we can scale them sufficiently. We have reviewed the main difficulties to obtain decent performances with GP-LVM approaches when applied to large-scale learning, even with inducing variables. Revisiting active set approximations, we considered a stochastic viewpoint to approximate the marginal likelihood while simultaneously keeping the model marginalized. We formulated our *stochastic active sets (SAS)* approach for both deterministic and Bayesian versions of GP decoders. We found that our approach works well with amortization, such that a neural network encoder approximately inverts the GP decoder. While amortization also helps when using inducing points, we found the combination with SAS to be particularly efficient and robust.

Empirically, we illustrated the advantage of our method first on image-based observations, where our approach learns better representations using fewer computational resources compared to inducing point methods. We further demonstrated that our approach easily scales to nearly 10^6 observations. In this experiment, we found that the learnt representations are qualitatively on par with those attained by a comparable autoencoder. This is an important finding as, beyond small datasets, GP decoders generally recovers less useful representations compared with models based on neural networks. From this result, we speculate that improvements in *training* might be enough to get state-of-the-art representations with GP decoders.

Additional benefits. Besides the empirical benefits demonstrated by SAS in the previous section, we have also observed other practical benefits worth reporting. *First*, we have observed that SAS easily runs in 32-bit numerical precision, unlike inducing point methods that generally require 64-bits of precision (when reporting running times we consistently used 64 bits). Similarly, the *jitter* usually added to the Cholesky factorization is of less importance in SAS. *Second*, we note that our implementation is surprisingly free of additional *tricks* and no numerical heuristics were needed to realize a reliable implementation.

Limitations and future work. Stochastic active sets rely on the Gaussian likelihood, and this is perhaps the strongest limitation. This works well for continuous data, but many data sources are inherently discrete and this requires a suitable likelihood, e.g. the discretized mixture of logistics (Salimans et al., 2017). Having more powerful likelihoods would surely improve the GP decoders, but this requires realization of further developments using SAS.

Future work will focus on applying the SAS approach in the supervised setting as well, and building SAS-like methods for discrete data. Other possible directions include extending the decoder with deep kernels (Wilson et al., 2016) to capture more features in the data and applying convolutional GPs (Van der Wilk et al., 2017) which are more suited to high-dimensional images.

Acknowledgements

The authors want to thank Simon Bartels for the fruitful discussions during the early stages of our investigation. This work was supported by research grants (15334, 42062) from VILLUM FONDEN. This project has also received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement 757360). This work was funded in part by the Novo Nordisk Foundation through the Center for Basic Machine Learning Research in Life Science (NNF20OC0062606). This work was further supported by the Pioneer Centre for AI, DNRF grant number P1.

References

M. Bauer, M. van der Wilk, and C. E. Rasmussen. Understanding probabilistic sparse Gaussian process approximations. In *Advances in neural information processing systems (NIPS)*, pages 1533–1541, 2016.

E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. A. Szerlip, P. Horsfall, and N. D. Goodman. Pyro: Deep universal probabilistic programming. *Journal of Machine Learning Research (JMLR)*, 20:28:1–28:6, 2019.

T. D. Bui and R. E. Turner. Stochastic variational inference for Gaussian process latent variable models using back constraints. In *Black Box Learning and Inference NIPS Workshop*, 2015.

L. Csató and M. Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

E. Fong and C. C. Holmes. On the marginal likelihood and cross-validation. *Biometrika*, 107(2):489–496, 2020.

Y. Gal, Y. Chen, and Z. Ghahramani. Latent Gaussian processes for distribution estimation of multivariate categorical data. In *International Conference on Machine Learning (ICML)*, pages 645–654. PMLR, 2015.

S. Geisser and W. F. Eddy. A predictive approach to model selection. *Journal of the American Statistical Association*, 74(365):153–160, 1979.

P. Germain, F. Bach, A. Lacoste, and S. Lacoste-Julien. PAC-Bayesian theory meets Bayesian inference. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016.

J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence (UAI)*, pages 282–290, 2013.

J. Hensman, A. Matthews, and Z. Ghahramani. Scalable variational Gaussian process classification. In *Artificial Intelligence and Statistics (AISTATS)*, pages 351–360, 2015.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

V. Lalchand, A. Ravuri, and N. D. Lawrence. Generalised GPLVM with stochastic variational inference. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 7841–7864, 2022.

N. Lawrence. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research (JMLR)*, 6(11), 2005.

N. D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 329–336, 2004.

N. D. Lawrence and J. Quiñonero-Candela. Local distance preservation in the GP-LVM through back constraints. In *International Conference on Machine Learning (ICML)*, pages 513–520, 2006.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

S. Liu, X. Sun, P. J. Ramadge, and R. P. Adams. Task-agnostic amortized inference of Gaussian process hyperparameters. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:21440–21452, 2020.

K. Märtens, K. Campbell, and C. Yau. Decomposing feature-level variation with covariate Gaussian process latent variable models. In *International Conference on Machine Learning (ICML)*, pages 4372–4381. PMLR, 2019.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: an imperative style, high-performance deep learning library. *Advances in neural information processing systems (NeurIPS)*, 32, 2019.

J. Quiñonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research (JMLR)*, 6(Dec):1939–1959, 2005.

C. Rasmussen and Z. Ghahramani. Occam's razor. *Advances in Neural Information Processing Systems (NIPS)*, 13, 2000.

C. E. Rasmussen and C. K. Williams. *Gaussian processes for machine learning*, volume 2. MIT Press, 2006.

D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning (ICML)*, pages 1530–1538. PMLR, 2015.

H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications. In *International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

M. W. Seeger, C. K. Williams, and N. D. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In *International Workshop on Artificial Intelligence and Statistics (AISTATS)*, pages 254–261. PMLR, 2003.

R. Shu, H. H. Bui, S. Zhao, M. J. Kochenderfer, and S. Ermon. Amortized inference regularization. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

A. J. Smola and P. L. Bartlett. Sparse greedy Gaussian process regression. In *Advances in Neural Information Processing Systems (NIPS)*, pages 619–625, 2001.

E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1257–1264, 2006.

J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning (ICML)*, pages 2256–2265. PMLR, 2015.

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

M. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics (AISTATS)*, pages 567–574, 2009.

M. Titsias and N. D. Lawrence. Bayesian Gaussian process latent variable model. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 844–851, 2010.

M. Van der Wilk, C. E. Rasmussen, and J. Hensman. Convolutional Gaussian processes. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2849–2858, 2017.

A. Vehtari and J. Lampinen. Bayesian model assessment and comparison using cross-validation predictive densities. *Neural Computation*, 14(10):2439–2468, 2002.

G. Wahba. *Spline models for observational data*. SIAM, 1990.

A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In *Artificial Intelligence and Statistics (AISTATS)*, pages 370–378. PMLR, 2016.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

In this appendix, we provide additional details about stochastic active sets (SAS) as an approximation of the log-marginal likelihood for GP decoders, widely known as the *Gaussian process latent variable model* (GP-LVM). We remark that SAS revisits *active sets*, a sparse approximation predominantly used before the seminal work of Snelson and Ghahramani (2006), in combination with stochastic optimization. The code for experiments is also included, and details on the data and initial setup of hyperparameters are included at the end of this appendix.

A Detailed derivation of Stochastic Active Sets

The construction of SAS approximations for the log-marginal likelihood $\log p(\boldsymbol{x}|\boldsymbol{z})$, builds on the connection between the *evidence* and *cross-validation* (CV) (Fong and Holmes, 2020). The equivalence between *leave-R-out* CV and the log-marginal likelihood is established by the use of predictive posterior scores, such that

$$\mathcal{S}_{\text{CV}}(\boldsymbol{x}|R) = \frac{1}{\mathcal{C}}\sum_{p=1}^{\mathcal{C}}\frac{1}{R}\sum_{n\in\mathcal{R}_p}\log p(\boldsymbol{x}_n|\boldsymbol{x}_{\mathcal{A}_p},\boldsymbol{z}) = \frac{1}{R}\mathbb{E}_{\mathcal{A}_p}\left[\sum_{n\in\mathcal{R}_p}\log p(\boldsymbol{x}_n|\boldsymbol{x}_{\mathcal{A}_p},\boldsymbol{z})\right], \tag{9}$$

where $\mathcal{A}_p$ denotes the *active set* indices of the training data, such that $\mathcal{A}_p \subset \{1,2,\ldots,N\}$ and $\mathcal{R}_p = \{1,2,\ldots,N\}\setminus\mathcal{A}_p$ are the remaining hold-out samples. The subscript $p \in \mathcal{C}$ denotes the permutation and we average over all $\mathcal{C} = \binom{N}{R}$ possible hold-out sets. We use use R to indicate the size of the hold-out set $\mathcal{R}_p$ and let $A = |\mathcal{A}_p| = N - R$. In particular, one might obtain the log-marginal likelihood in a cumulative manner by summing the scores $\mathcal{S}_{\text{CV}}(\boldsymbol{x}|R)$ in Eq. (9) over all possible lengths of R,

$$\log p(\boldsymbol{x}|\boldsymbol{z}) = \sum_{r=1}^{N}\mathcal{S}_{\text{CV}}(\boldsymbol{x}|r), \tag{10}$$

which is the main result presented in Fong and Holmes (2020). Notice that Eq. (10) has a similar computational cost as the exact calculus of $\log p(\boldsymbol{x}|\boldsymbol{z})$ for GP decoders, since for small values of r, i.e. $r = 1, 2, 3\ldots$, we need to invert large covariance matrices $\boldsymbol{K}_{\mathcal{A}\mathcal{A}}$, where $A \to N$. Here, we drop the permutation subscript p in $\mathcal{A}$ to avoid cluttered notation. Alternatively, we use the property that Eq. (10) can be factorised as

$$\log p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{S}_{\text{CCV}}(\boldsymbol{x}|R) + \mathcal{S}_{\text{PCV}}(\boldsymbol{x}|R), \tag{11}$$

where $\mathcal{S}_{\text{CCV}}(\boldsymbol{x}|R)$ is the *cumulative* CV score and $\mathcal{S}_{\text{PCV}}(\boldsymbol{x}|R)$ is defined as the *preparatory* CV. Additionally, Eq. (11) holds for every size of the hold-out data $R \in [1, 2, \cdots, N]$. This factorisation is of interest for us due to

$$\mathcal{S}_{\text{PCV}}(\boldsymbol{x}|R) = \sum_{r=R+1}^{N}\mathcal{S}_{\text{CV}}(\boldsymbol{x}|r) = \frac{1}{\mathcal{C}}\sum_{p=1}^{\mathcal{C}}\log p(\boldsymbol{x}_{\mathcal{A}_p}|\boldsymbol{z}_{\mathcal{A}_p}), \tag{12}$$

where the r.h.s. term is equivalent to $\mathcal{S}_{\text{PCV}}(\boldsymbol{x}|R) = \mathbb{E}_{\mathcal{A}_p}[\log p(\boldsymbol{x}_{\mathcal{A}_p}|\boldsymbol{z}_{\mathcal{A}_p})]$. We remark that the computational cost of Eq. (12) is *cheaper* than $\mathcal{S}_{\text{CCV}}(\boldsymbol{x}|R)$ when the choice of R is sufficiently large. Additionally, the *cumulative* CV is defined as

$$\mathcal{S}_{\text{CCV}}(\boldsymbol{x}|R) = \sum_{r=1}^{R}\mathcal{S}_{\text{CV}}(\boldsymbol{x}|r) = \sum_{r=1}^{R}\frac{1}{\mathcal{C}_r}\sum_{p=1}^{\mathcal{C}_r}\frac{1}{r}\sum_{n\in\mathcal{R}_p}\log p(\boldsymbol{x}_n|\boldsymbol{x}_{\mathcal{A}_p},\boldsymbol{z}), \tag{13}$$

where $\mathcal{C}_r = \binom{N}{r}$ are all the possible hold-out set for every value of r considered. We remark the convenience of Eq. (13) for stochastic optimization as it includes predictive posterior probabilities $p(\boldsymbol{x}_n|\boldsymbol{x}_{\mathcal{A}_p},\boldsymbol{z})$ which emerge from the factorization of hold-R-out CV. This expression can be also rewritten as

$$\mathcal{S}_{\text{CCV}}(\boldsymbol{x}|R) = \sum_{r=1}^{R}\frac{1}{r}\mathbb{E}_{\mathcal{A}_p}\left[\sum_{n\in\mathcal{R}_p}\log p(\boldsymbol{x}_n|\boldsymbol{x}_{\mathcal{A}_p},\boldsymbol{z})\right]. \tag{14}$$

The SAS approximation stochastically estimates both $\mathcal{S}_{\text{CCV}}(\boldsymbol{x}|R)$ and $\mathcal{S}_{\text{PCV}}(\boldsymbol{x}|R)$, with particular attention to the CCV score, which for large values of R induces the largest computational cost.

B Experiments, Algorithms and Metrics

The code for the experiments is written in Python 3.7 and uses the Pytorch syntax for the automatic differentiation of the GP models. It can be found in the repository `https://github.com/pmorenoz/SASGP`, where we also use the library Pyro for some baselines. In this section, we provide a detailed description of the experiments and the data used, the initialization of both latent variables $\boldsymbol{z}$, the parameters of the *amortization* network and hyperparameters $\boldsymbol{\theta}$. The training algorithms are provided in the main manuscript for both the *deterministic* and Bayesian approaches to the GP decoder. The performance metrics included in the main manuscript, e.g. the negative log-predictive density (NLPD), the root mean square error (RMSE) and the mean absolute error (MAE).

B.1 Detailed description and initialization

All the models have matching encoding network architecture: three linear, fully connected layers with ReLU activation functions. The first two layers have sizes of 512 and 256 hidden units and the network encodes to two dimensions. The variances of the latent variables $\boldsymbol{z}$ were encoded in a different way. In the VAE model, the variance was obtained by inputting the latent means to a soft-plus layer and the Bayesian SAS-GP and the Bayesian GP-LVM had separate network (with similar architecture) encoding the variance. The decoder in the VAE was built by a linear, fully connected layer with 400 hidden units, a softplus function and a sigmoid mapping. The GP-LVM baselines are implemented in Pyro (Bingham et al., 2019). Our implementation of the variational autoencoder is based on the official Pyro tutorial for VAEs. Importantly, the VAE and the SAS-GP implementations use standard *data loaders* whereas the Pyro code must keep all the data in memory. This has limited the scaling possibilities of experiments with baselines.

In most of our experiments, we use the *vanilla* RBF kernel, where we initially set the *amplitude* hyperparameter to $\sigma_a^2 = 0.5$, the *lengthscale* to $\ell = 0.1$ and the likelihood noise variance to $\sigma_n^2 = 0.5$. The initial location of latent variables $\boldsymbol{z}$ is subject to the initialization of the amortization networks, which is set up in a standard manner using the Pytorch `nn` module. Learning rates are set in the range $[10^{-4}, 10^{-2}]$ and the maximum number of epochs considered is 300.

B.2 Datasets

Our experiments make use of three well-known datasets: MNIST (LeCun et al., 1998), FMNIST (Xiao et al., 2017) and CIFAR10 (Krizhevsky, 2009). All of them are downloaded from the `torchvision` repository included in the Pytorch library (`https://pytorch.org/vision/stable/datasets.html`). These particular datasets are not subject to use constraints or they include licenses which allow their use for research purposes.

B.3 Error metrics

Having defined the test dataset as $\boldsymbol{x}_* = \{\boldsymbol{x}_n\}_{n=1}^{N_*}$, we use the following error metrics to test the performance of the SAS approximation for GP decoders

$$\text{RMSE}(\boldsymbol{x}_*) = \sqrt{\frac{1}{N_*}\sum_{n=1}^{N_*}(\boldsymbol{x}_n - \boldsymbol{\mu}_n^*)^2}, \tag{15}$$

$$\text{MAE}(\boldsymbol{x}_*) = \frac{1}{N_*}\sum_{n=1}^{N_*}|\boldsymbol{x}_n - \boldsymbol{\mu}_n^*|\,, \tag{16}$$

$$\text{NLPD}(\boldsymbol{x}_*) = \frac{1}{2}\log(2\pi) + \frac{1}{2N_*}\sum_{n=1}^{N_*}\left[\log \boldsymbol{v}_n^* + \frac{(\boldsymbol{x}_n - \boldsymbol{\mu}_n^*)^2}{\boldsymbol{v}_n^*}\right], \tag{17}$$

where $\boldsymbol{\mu}_n^*$ and $\boldsymbol{v}_n^*$ are the predictive mean and variance per nth test sample, respectively.

B.4 Additional experiment with latent spaces of dimension larger than two

We re-computed the experiments used for Table 2 using latent spaces of dimension *three* and *four*. The main outcome from these experiments is that training is as stable as in the former cases with two dimensions in the latent space. In general, we observed a similar performance as in the rest of experiments included in the main manuscript. So we remark that there is no limitation in our framework to accept $\dim(\mathcal{Z}) > 2$.

B.5 Ablation study

We did additional experiments as an ablation study based on Eq. 6. In particular, we ran the SAS model for $A = \{100, 200, 400\}$ using the MNIST and FMNIST datasets. The first ablation experiment shown in Figure 2 corresponds to using only the *second term* $\log p(\boldsymbol{x}_\mathcal{A}|\boldsymbol{z}_\mathcal{A})$ in Eq. 6. We can observe that the performance is not as good as in the results illustrated in the main manuscript. Alternatively, we also included another ablation experiment using the *first term* of Eq. 6, which is shown in Figure 3. In this last case, the performance is not good and the structure in the latent space is only provided by the amortization net.

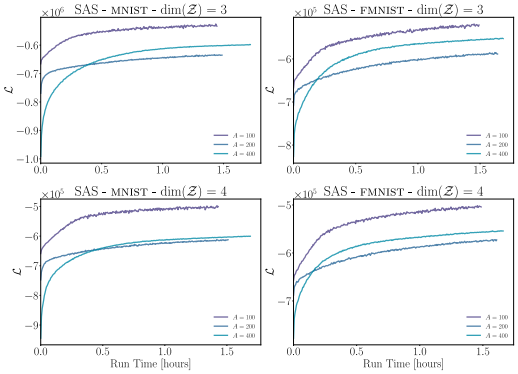


Figure 6: Training curves for different active set sizes A and dimensionalities of the latent spaces. Each row contains the results for MNIST and FMNIST, respectively.

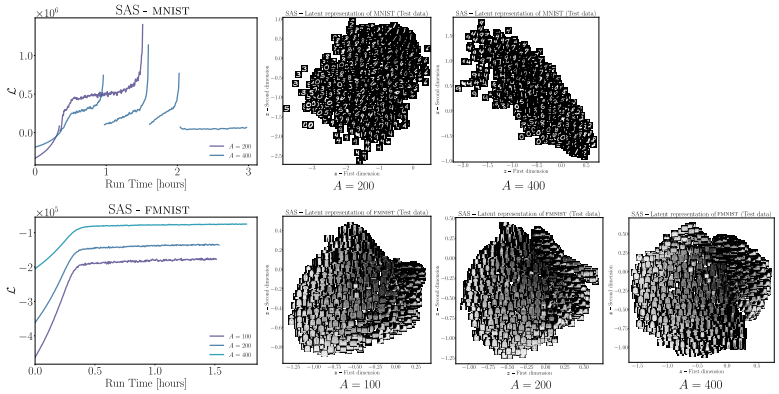


Figure 7: Ablation study for Eq. 6. The approximation of the log-marginal likelihood is only computed with the firm term (factorisation). Curves are computed for $A = \{100, 200, 400\}$ and rows indicate the dataset MNIST or FMNIST.

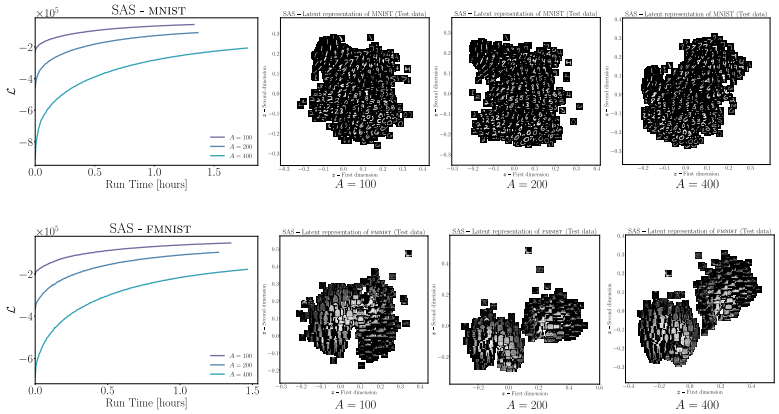


Figure 8: Ablation study for Eq. 6. The approximation of the log-marginal likelihood is only computed with the second term (full covariance). Curves are computed for $A = \{100, 200, 400\}$ and rows indicate the dataset MNIST or FMNIST.

SYMMETRY BREAKING

B

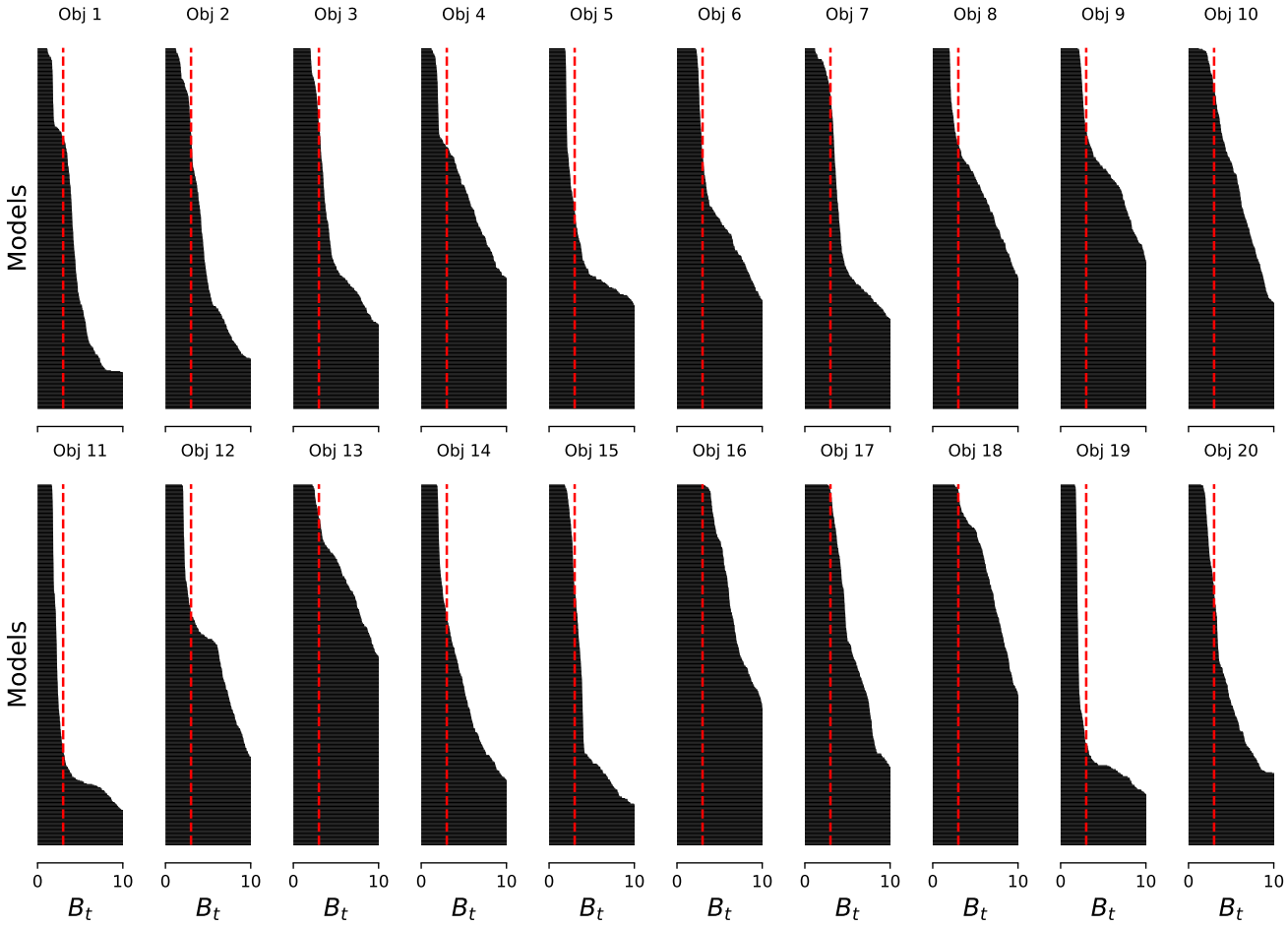


Figure B.1: Barcodes by object for t-SNE. The x-axis shows B_t and vertically, different TriMap models are displayed. The dotted, red line indicates $B_t = 3$. The object number in COIL-20 is listed above the corresponding barcode. Discussed in section 2.5.1. The figure first appeared in Feldager, Hauberg, and Hansen [1].

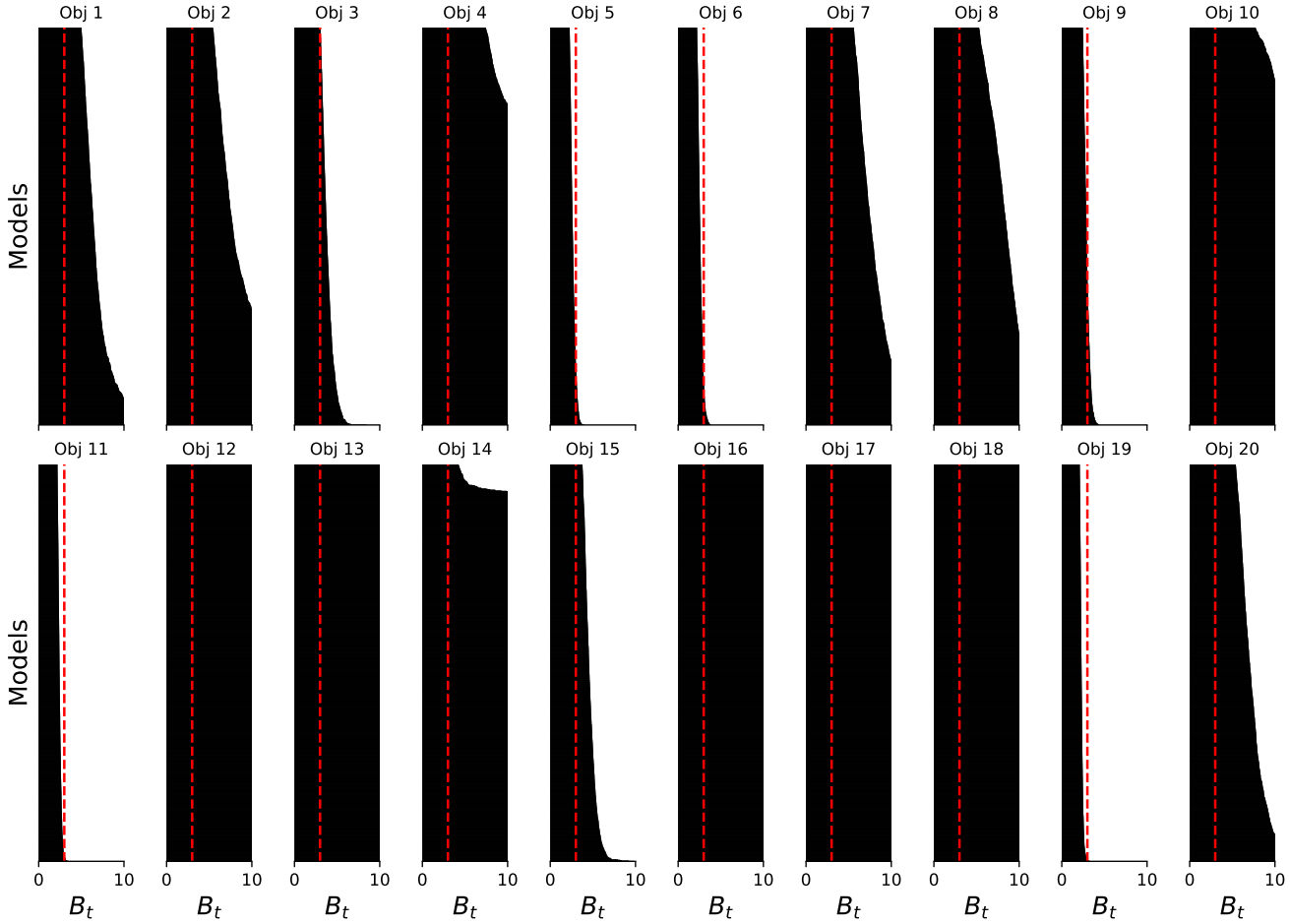


Figure B.2: Barcodes by object for TriMap. The x-axis shows B_t and vertically, different TriMap models are displayed. The dotted, red line indicates $B_t = 3$. The object number in COIL-20 is listed above the corresponding barcode. Discussed in section 2.5.2. The figure first appeared in Feldager, Hauberg, and Hansen [1].

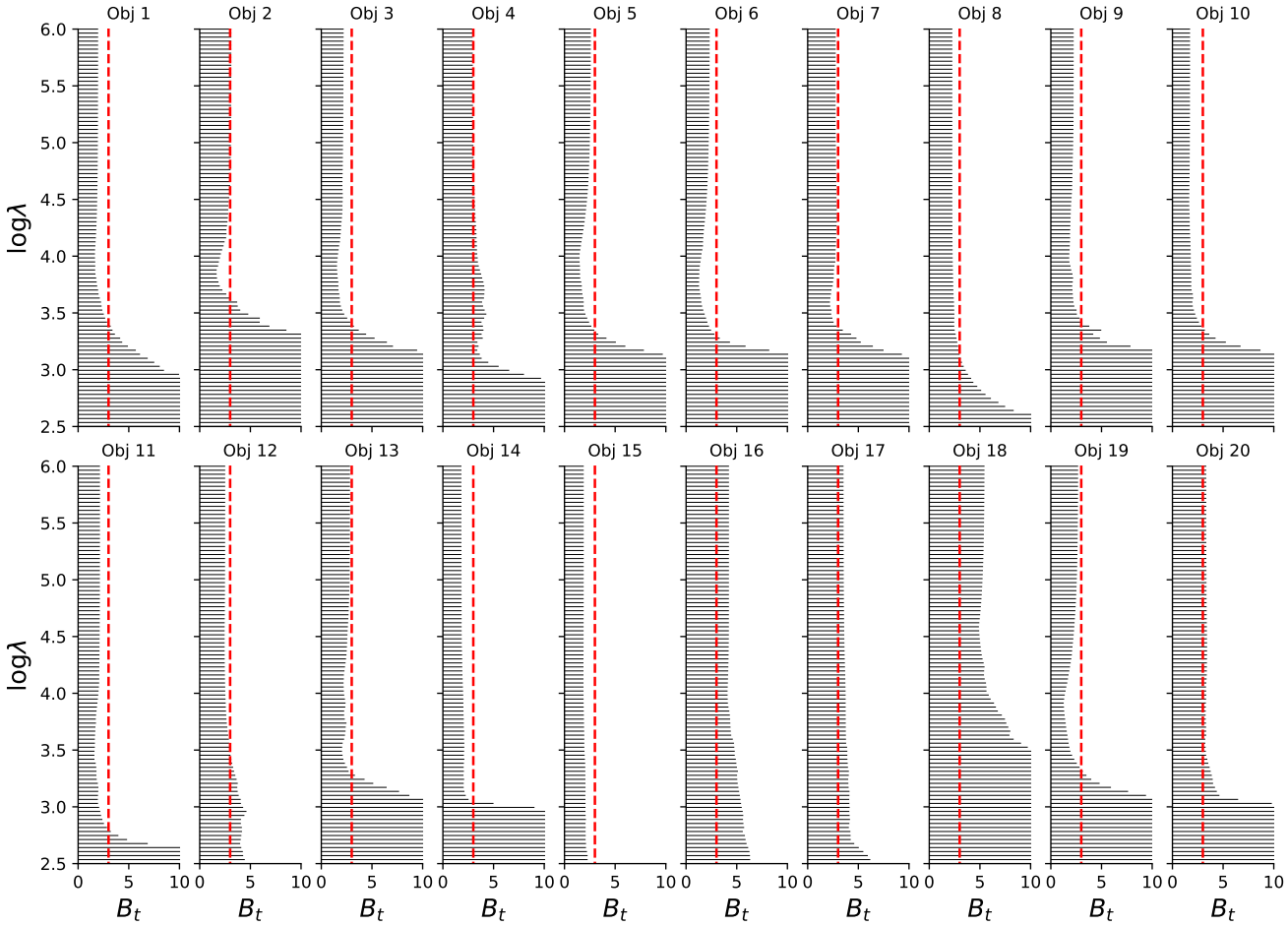


Figure B.3: Barcodes by object for kPCA. The x-axis shows B_t and vertically, different TriMap models are displayed. The dotted, red line indicates $B_t = 3$. The object number in COIL-20 is listed above the corresponding barcode. Discussed in section 2.5.3. The figure first appeared in Feldager, Hauberg, and Hansen [1].

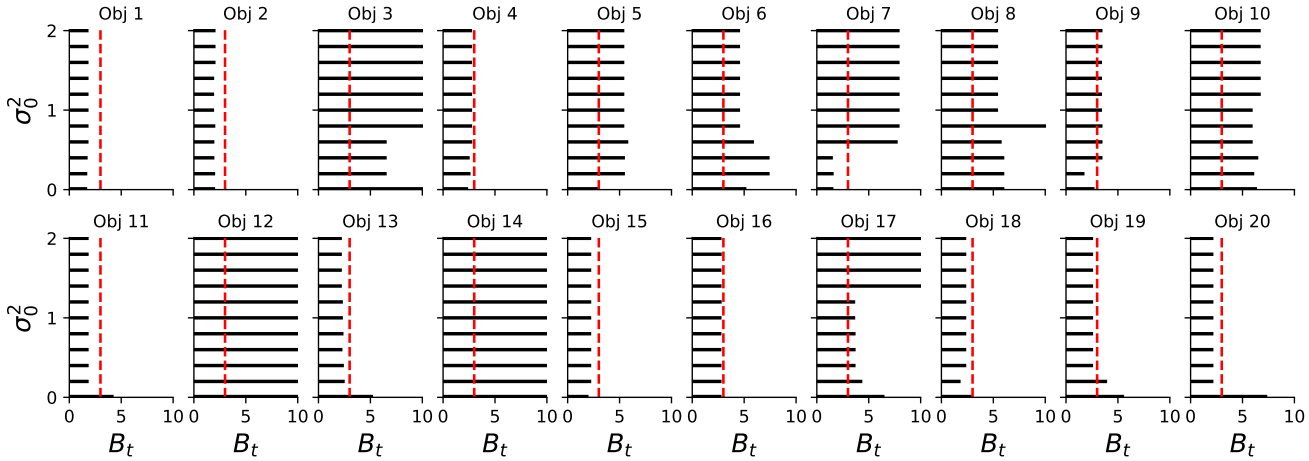


Figure B.4: Barcode by object for GPLVM with fixed initial values of θ_0. The x-axis shows B_t and vertically, different GPLVM models are displayed which corresponds to different initial values of σ_0^2. The dotted, red line indicates $B_t = 3$. The object number in COIL-20 is listed above the corresponding barcode. Discussed in section 2.5.4. The figure first appeared in Feldager, Hauberg, and Hansen [1].

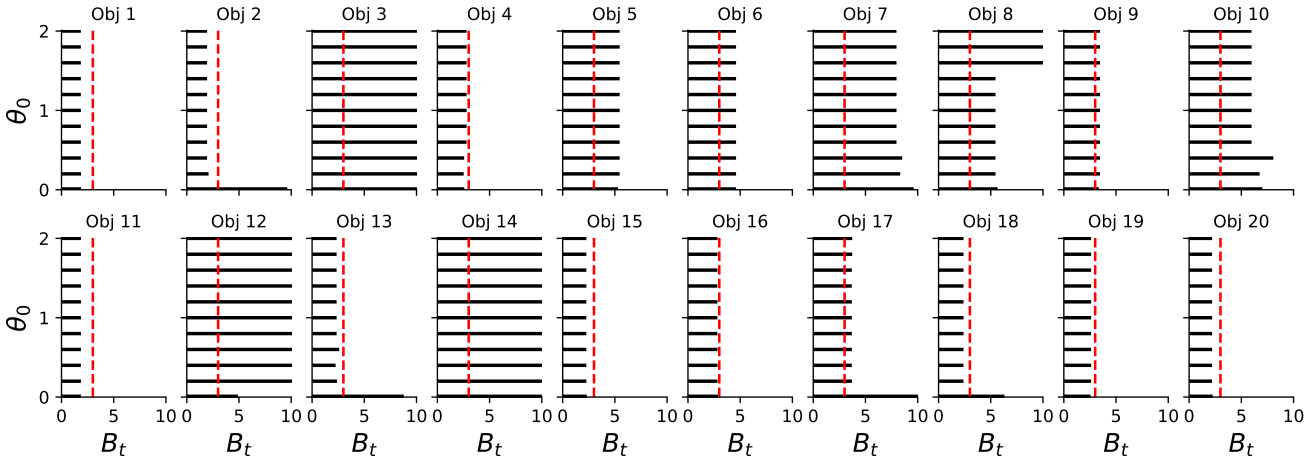


Figure B.5: Barcodes by object for GPLVM with fixed initial values of σ_0^2. The x-axis shows B_t and vertically, different GPLVM models are displayed which corresponds to different initial values of θ_0. The dotted, red line indicates $B_t = 3$. The object number in COIL-20 is listed above the corresponding barcode. Discussed in section 2.5.4. The figure first appeared in Feldager, Hauberg, and Hansen [1].

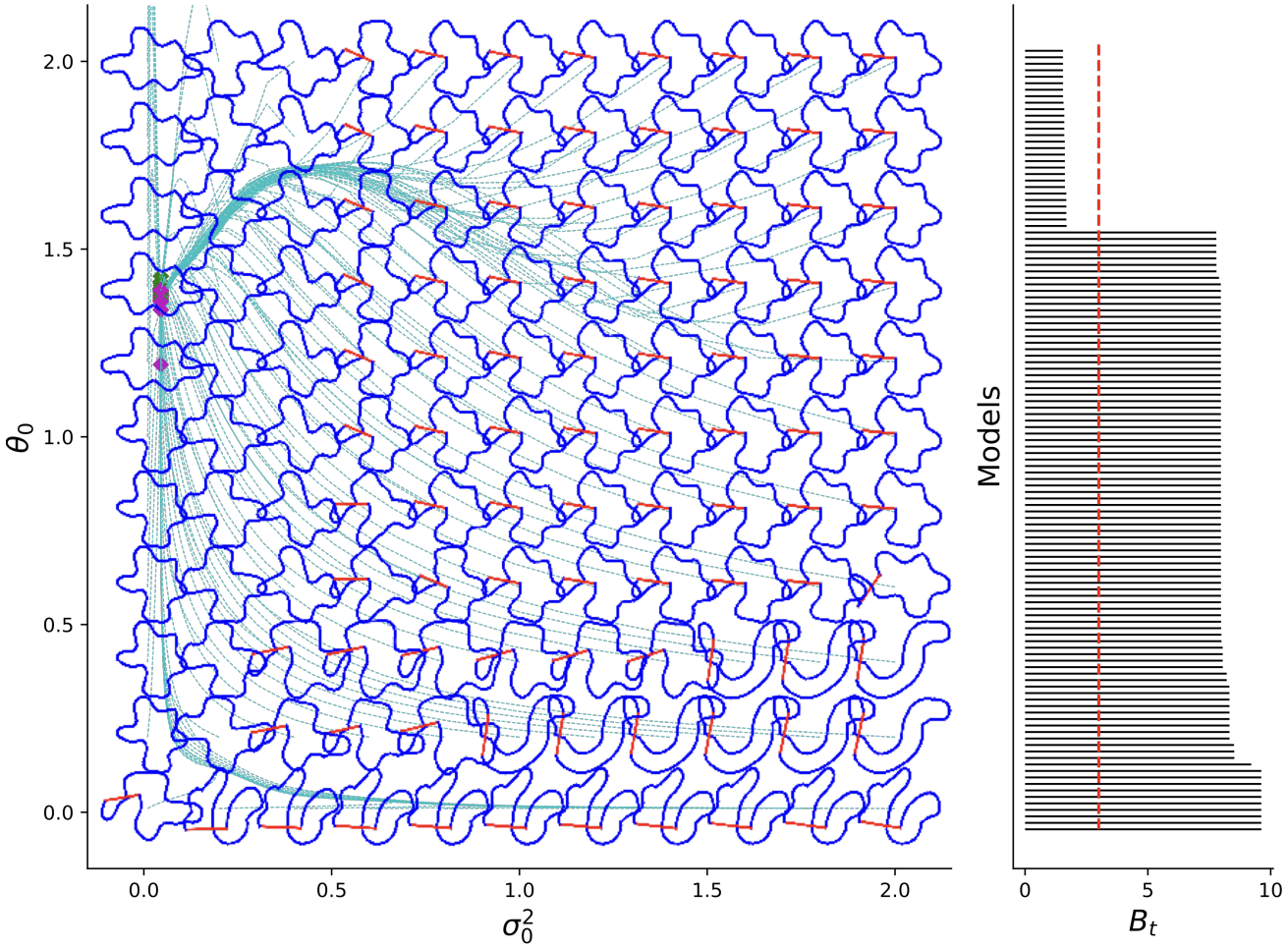


Figure B.6: The left-hand side shows a phase space for GPLVMs in the initial values of hyperparameters θ_0 and σ_0^2 using isomap initialisation. Each location in the phase space corresponds to an initialisation of a model. As the model is trained it follows the cyan trajectory through phase space and until the green cross which indicates the state of the model at the end of training. The learnt latent space is shown at the location of the initialisation to emphasise what representation a given initialisation results in. The right-hand side shows the barcode for these models where each bar corresponds to a model on the axis of thresholds. The red dotted line represented our chosen threshold for symmetry breaking. Discussed in section 2.5.4.

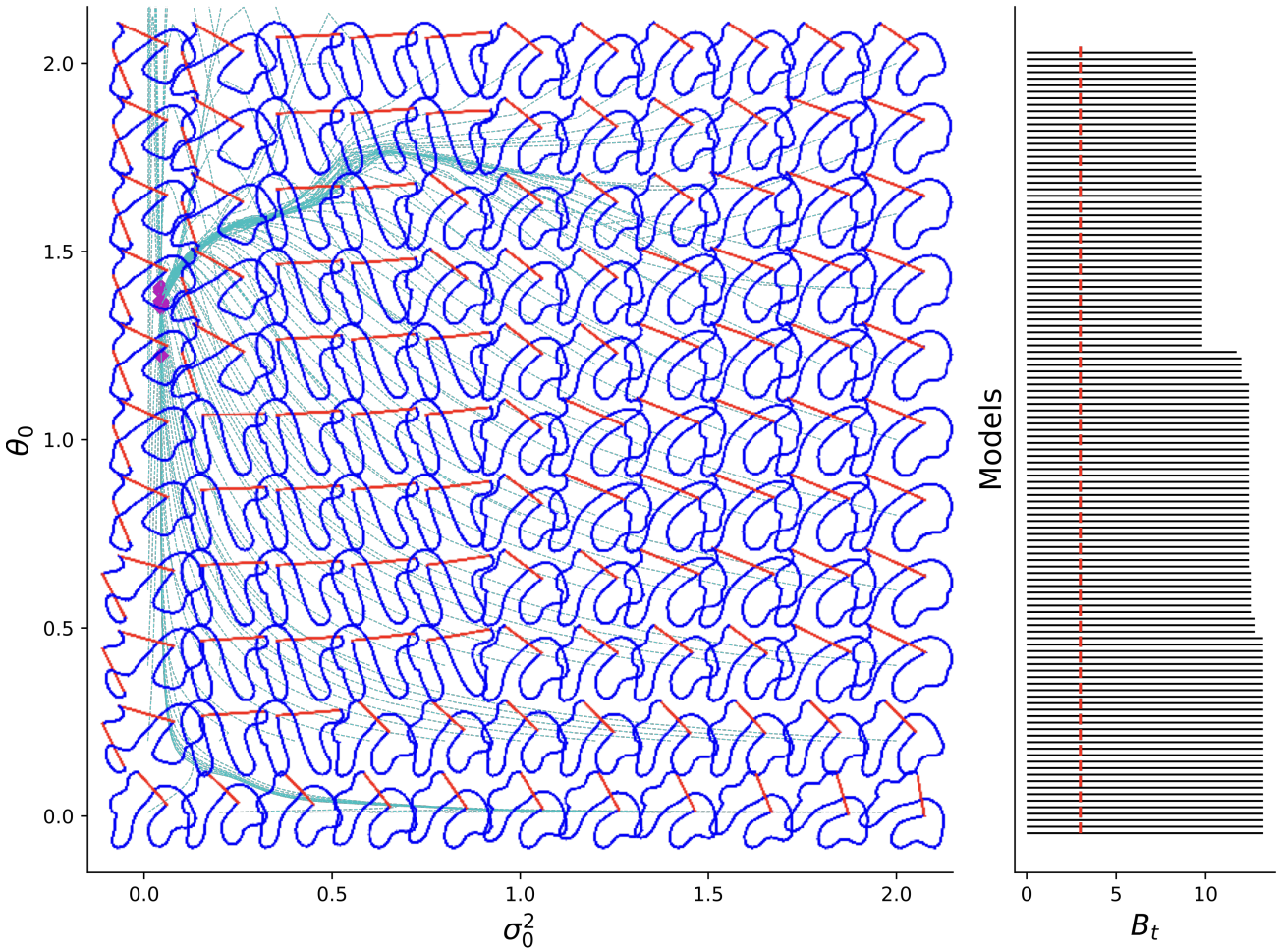


Figure B.7: The left hand side shows a phase space for GPLVMs in the initial values of hyper parameters θ_0 and σ_0^2 using PCA initialisation. Each location in the phase space corresponds to an initialisation of a model. As the model is trained it follows the cyan trajectory through phase space and until the green cross which indicates the state of the model at the end of training. The learnt latent space is shown at the location of the initialisation to emphasise what representation a given initialisation results in. The right hand side shows the barcode for these models where each bar corresponds to a model on the axis of thresholds. The red dotted line represented our chosen threshold for symmetry breaking. Discussed in section 2.5.4.

THE GAUSSIAN DISTRIBUTION

C

For easy reference, we repeat the properties of the multivariate Gaussian distributions [51].

Consider a set of N observations collected in a matrix $\mathbf{Y} = \{y_n\}_{n=1}^N$. Each observation is of dimensionality D such that $\mathbf{Y} \in \mathbb{R}^{N \times D}$. The Gaussian distribution, $\mathcal{N}$, is defined as

GAUSSIAN DISTRIBUTION
Multivariate

$$\mathcal{N}(\mathbf{y}|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^k \det(\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{y}-\mu)^\top \Sigma^{-1}(\mathbf{y}-\mu)\right)$$

This gives the probability of observations y under the multivariate Gaussian distribution of dimensionality k given a mean, μ, and a covariance, Σ. The distribution of a D-dimensional vector $\mathbf{x}$ has a D-dimensional mean μ and $D \times D$ covariance matrix Σ which must be symmetric and positive semi-definite and is given by

$$\mathcal{N}(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{-1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^\top \Sigma^{-1}(\mathbf{x}-\mu)\right)$$

with

$$\mathbb{E}[\mathbf{x}] = \mu \tag{C.1}$$
$$\text{cov}[\mathbf{x}] = \Sigma \tag{C.2}$$

The precision matrix is the inverse of the covariance matrix so $\mathbf{\Lambda} = \Sigma^{-1}$.

If we have a marginal Gaussian distribution of $\mathbf{x}$ $p(x)$,and a conditional distribution $p(\mathbf{y}|\mathbf{x})$ given in the form

$$p(x) = \mathcal{N}(x|\mu, \Lambda^{-1}) \tag{C.3}$$
$$p(y|x) = \mathcal{N}(y|Ax + b, L^{-1}) \tag{C.4}$$

then we can obtain the form of the marginal distribution $p(y)$ and the conditional $p(x|y)$

$$p(y) = \mathcal{N}(y|Ax + b, L^{-1} + A\Lambda^{-1}A^T) \tag{C.5}$$
$$p(x|y) = \mathcal{N}(x|\Sigma[A^T L(y-b) + \Lambda\mu], \Sigma) \tag{C.6}$$
$$\Sigma = (\Lambda + A^T LA)^{-1} \tag{C.7}$$

Alternatively, if we have a joint Gaussian distribution $\mathcal{N}(x|\mu, \Sigma)$ with $\Lambda \equiv \Sigma^{-1}$ so we can define the following identities

$$x = \begin{pmatrix} x_a \\ x_b \end{pmatrix}, \qquad \mu = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix} \tag{C.8}$$

$$\Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}, \qquad \Lambda = \begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix} \tag{C.9}$$

then the conditional $p(x|y)$ and the marginal $p(x_a)$ distributions are given by

$$p(x_a|x_b) = \mathcal{N}(x|\mu_{a|b}, \Lambda^{-1}) \tag{C.10}$$

$$\mu_{a|b} = \mu_a - \Lambda_{aa}^{-1}\Lambda_{ab}(x_b - \mu_b) \tag{C.11}$$

$$p(x_a) = \mathcal{N}(x_a|\mu_a, \Sigma_{aa}) \tag{C.12}$$

STOCHASTIC ACTIVE SETS

D

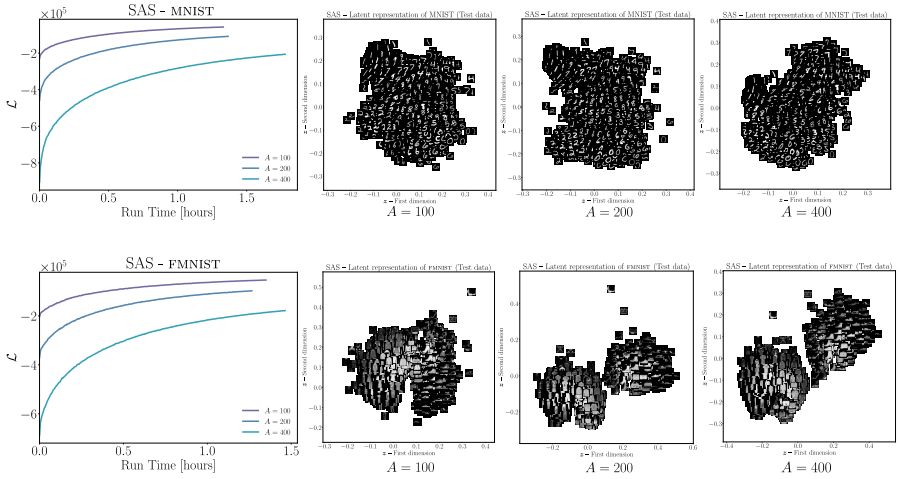


Figure D.1: Ablation study for the approximation of the log-marginal likelihood using the full covariance term (second term in equation 4.18). The training curves have $A = \{100, 200, 400\}$ and rows indicate the dataset MNIST (top) or FASHIONMNIST (bottom).

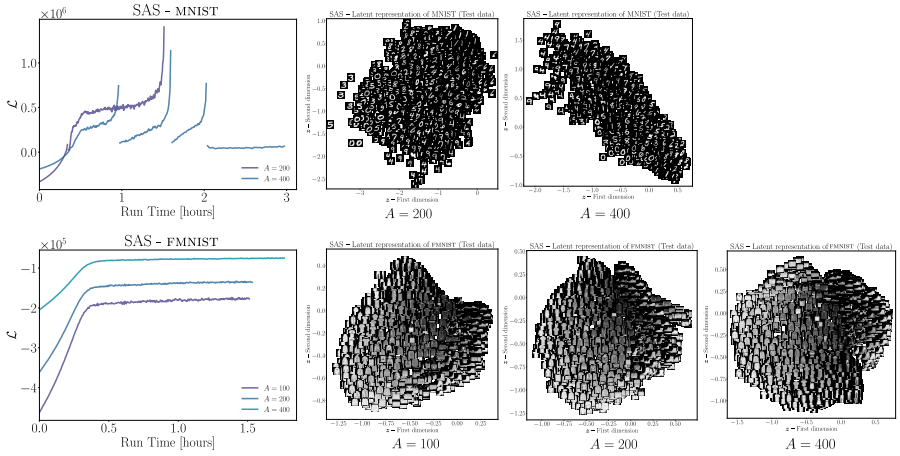


Figure D.2: Ablation study for the approximation of the log-marginal likelihood using the factorised term (first term in equation 4.18). The training curves have $A = \{100, 200, 400\}$ and rows indicate the dataset MNIST (top) or FASHIONMNIST (bottom).

GEOMETRY

E

E.1 DERIVATION OF THE GEODESIC EQUATION

Geodesics can be found by solving the geodesic equations, a second, ordinary differential equation.

$$\ddot{\gamma} + \Gamma^i_{jk}\dot{\gamma}^j\dot{\gamma}^k = 0 \tag{E.1}$$

GEODESIC EQUATION

In this section, we derive the geodesic equation through the principle of least action, which leads to the Euler-Lagrange equations,

$$\underbrace{\frac{\mathrm{d}}{\mathrm{d}t}\underbrace{\frac{\partial L}{\partial \dot{x}}}_{\text{Step two}} - \underbrace{\frac{\partial L}{\partial x}}_{\text{Step one}}}_{\text{Step three}} = 0,$$

where the dot denotes differentiation with respect to t, we note that we can employ a change of variables to get $d\tau = Ldt$. The derivation takes three steps: First, we compute $\frac{\partial L}{\partial x^\gamma}$, then $\frac{\partial L}{\partial \dot{x}^\gamma}$ and finally $\frac{d}{\mathrm{d}t}\frac{\partial L}{\partial \dot{x}^\gamma}$.

Step One

Starting with step one, we insert the expression for the Lagrangian and use the chain rule.

$$\frac{\partial L}{\partial x^\gamma} = \frac{\partial}{\partial x^\gamma}\left(\sqrt{g_{\alpha\beta}\dot{x}^\alpha\dot{x}^\beta}\right) = \frac{1}{2L}\frac{\partial}{\partial x^\gamma}\left(g_{\alpha\beta}\dot{x}^\alpha\dot{x}^\beta\right) \tag{E.2}$$

Differentiating using the product rule yields

$$\frac{\partial L}{\partial x^\gamma} = \frac{1}{2L}\left(\frac{\partial g_{\alpha\beta}}{\partial x^\gamma}\dot{x}^\alpha\dot{x}^\beta + g_{\alpha\beta}\frac{\partial \dot{x}^\alpha}{\partial x^\gamma}\dot{x}^\beta + g_{\alpha\beta}\dot{x}^\alpha\frac{\partial \dot{x}^\beta}{\partial x^\gamma}\right) \tag{E.3}$$

where we realise some terms are zero.

We note that the dot denotes a derivative with respect to s, so with a change of variables $\mathrm{d}\tau = L\mathrm{d}t$

$$\frac{\partial L}{\partial x^\gamma} = \frac{L}{2}\frac{\partial g_{\alpha\beta}}{\partial x^\gamma}\frac{\mathrm{d}x^\alpha}{\mathrm{d}t}\frac{\mathrm{d}x^\beta}{\mathrm{d}t} \tag{E.4}$$

we complete step one in the derivation of the geodesic equation.

Step Two

Step two requires computing the derivative of Lagrangian with respect to $\dot{x}^\gamma$. Again, we insert the expression for the Lagrangian.

$$\frac{\partial L}{\partial \dot{x}^\gamma} = \frac{\partial}{\partial \dot{x}^\gamma}\sqrt{g_{\alpha\beta}\dot{x}^\alpha\dot{x}^\beta} = \frac{1}{2L}\frac{\partial}{\partial \dot{x}^\gamma}\left(g_{\alpha\beta}\dot{x}^\alpha\dot{x}^\beta\right) \tag{E.5}$$

$$\tag{E.6}$$

We then differentiate and realise a term cancels (the metric $g_{\alpha,\beta}$ does not depend on the γth tangent vector), and the derivative further reduces to kronecker *delta*s as they are only non-zero when γ is either α or γ.

$$\frac{\partial L}{\partial \dot{x}^\gamma} = \frac{1}{2L}\left(\cancel{\frac{\partial g_{\alpha\beta}}{\partial \dot{x}^\gamma}}\dot{x}^\alpha\dot{x}^\beta + g_{\alpha\beta}\frac{\partial \dot{x}^\alpha}{\partial \dot{x}^\gamma}\dot{x}^\beta + g_{\alpha\beta}\dot{x}^\alpha\frac{\partial \dot{x}^\beta}{\partial \dot{x}^\gamma}\right) \tag{E.7}$$

$$= \frac{1}{2L}\left(g_{\alpha\beta}\delta^\alpha_\gamma\dot{x}^\beta + g_{\alpha\beta}\dot{x}^\alpha\delta^\beta_\gamma\right) \tag{E.8}$$

Due to the δs, the expression reduces and renaming the dummy index β to α yields

$$\frac{\partial L}{\partial \dot{x}^\gamma} = \frac{1}{2L}\left(g_{\gamma\beta}\dot{x}^\beta + g_{\alpha\gamma}\dot{x}^\alpha\right) = \frac{1}{2L}\left(g_{\gamma\alpha}\dot{x}^\alpha + g_{\alpha\gamma}\dot{x}^\alpha\right) \tag{E.9}$$

and with a change of variables, we obtain

$$\frac{\partial L}{\partial \dot{x}^\gamma} = \frac{1}{L}g_{\gamma\alpha}\dot{x}^\alpha \tag{E.10}$$

This concludes step two.

Step Three

For step three, we differentiate our result in step two with respect to s.

$$\frac{\mathrm{d}}{\mathrm{d}s}\frac{\partial L}{\partial \dot{x}^\gamma} = \frac{\mathrm{d}}{\mathrm{d}s}\left(\frac{1}{L}g_{\gamma\alpha}\dot{x}^\alpha\right) = L\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{1}{\cancel{L}}g_{\gamma\alpha}\cancel{L}\frac{dx^\alpha}{\mathrm{d}t}\right) \tag{E.11}$$

$$= L\left(\frac{\mathrm{d}g_{\gamma\alpha}}{\mathrm{d}t}\frac{\mathrm{d}x^\alpha}{\mathrm{d}t} + g_{\gamma\alpha}\frac{\mathrm{d}^2x^\alpha}{\mathrm{d}t^2}\right) \tag{E.12}$$

The first term in the parenthesis requires some massaging to get to the geodesic equation.

$$\frac{\mathrm{d}g_{\gamma\alpha}}{\mathrm{d}t}\frac{\mathrm{d}x^\alpha}{\mathrm{d}t} = \frac{1}{2}\frac{\mathrm{d}g_{\gamma\alpha}}{\mathrm{d}t}\frac{\mathrm{d}x^\alpha}{\mathrm{d}t} + \frac{1}{2}\frac{\mathrm{d}g_{\gamma\alpha}}{\mathrm{d}t}\frac{\mathrm{d}x^\alpha}{\mathrm{d}t} \tag{E.13}$$

$$= \frac{1}{2}\frac{\mathrm{d}g_{\gamma\alpha}}{\mathrm{d}t}\frac{\mathrm{d}x^\alpha}{\mathrm{d}t} + \frac{1}{2}\frac{\mathrm{d}g_{\gamma\beta}}{\mathrm{d}t}\frac{\mathrm{d}x^\beta}{\mathrm{d}t} \tag{E.14}$$

Here, we have split up the term and renamed the dummy index.

$$\frac{\mathrm{d}g_{\gamma\alpha}}{\mathrm{d}t}\frac{\mathrm{d}x^\alpha}{\mathrm{d}t} = \frac{1}{2}\frac{\partial g_{\gamma\alpha}}{\partial \mathrm{d}x^\nu}\frac{\mathrm{d}x^\nu}{\mathrm{d}t}\frac{\mathrm{d}x^\alpha}{\mathrm{d}t} + \frac{1}{2}\frac{\partial g_{\gamma\beta}}{\partial \mathrm{d}x^\mu}\frac{\mathrm{d}x^\mu}{\mathrm{d}t}\frac{\mathrm{d}x^\beta}{\mathrm{d}t} \tag{E.15}$$

$$= \frac{1}{2}\left(\frac{\partial g_{\gamma\alpha}}{\partial \mathrm{d}x^\beta} + \frac{\partial g_{\gamma\beta}}{\partial \mathrm{d}x^\alpha}\right)\frac{\mathrm{d}x^\alpha}{\mathrm{d}t}\frac{\mathrm{d}x^\beta}{\mathrm{d}t} \tag{E.16}$$

Here, we have used the chain rule and then rearranged the terms. This concludes step three.

Collecting Terms

We can now collect the terms into the Euler-Lagrange equations.

$$0 = \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{x}^\gamma} - \frac{\partial L}{\partial x^\gamma} \tag{E.17}$$

$$= \not{L}\left[g_{\gamma\alpha}\frac{\mathrm{d}^2x^\alpha}{\mathrm{d}t^2} + \frac{1}{2}\left(\frac{\partial g_{\gamma\alpha}}{\partial x^\beta} + \frac{\partial g_{\gamma\beta}}{\partial x^\alpha}\right)\frac{\mathrm{d}x^\alpha}{\mathrm{d}t}\frac{\mathrm{d}x^\beta}{\mathrm{d}t}\right] - \frac{\not{L}}{2}\frac{\partial g_{\alpha\beta}}{\partial x^\gamma}\frac{\mathrm{d}x^\alpha}{\mathrm{d}t}\frac{\mathrm{d}x^\beta}{\mathrm{d}t}$$

$$= g_{\gamma\alpha}\frac{\mathrm{d}^2x^\alpha}{\mathrm{d}t^2} + \frac{1}{2}\left[\frac{\partial g_{\gamma\alpha}}{\partial x^\beta} + \frac{\partial g_{\gamma\beta}}{\partial x^\alpha} - \frac{\partial g_{\alpha\beta}}{\partial x^\gamma}\right]\frac{\mathrm{d}x^\alpha}{\mathrm{d}t}\frac{\mathrm{d}x^\beta}{\mathrm{d}t} \tag{E.18}$$

Here we pause to do a bunch of renaming of dummy indices on the right-hand side.

$$0 = g_{\gamma\nu}\frac{\mathrm{d}^2x^\nu}{\mathrm{d}t^2} + \frac{1}{2}\left[\frac{\partial g_{\gamma\nu}}{\partial x^\beta} + \frac{\partial g_{\gamma\beta}}{\partial x^\nu} - \frac{\partial g_{\nu\beta}}{\partial x^\gamma}\right]\frac{\mathrm{d}x^\nu}{\mathrm{d}t}\frac{\mathrm{d}x^\beta}{\mathrm{d}t} \tag{E.19}$$

Next, we multiply with the inverse metric and use the property that $g_{\mu\nu}g^{\mu\nu} = \delta^\mu_\mu = 1$.

$$0 = g^{\gamma\nu}g_{\gamma\nu}\frac{\mathrm{d}^2x^\nu}{\mathrm{d}t^2} + \underbrace{\frac{1}{2}g^{\gamma\nu}\left[\frac{\partial g_{\gamma\nu}}{\partial x^\beta} + \frac{\partial g_{\gamma\beta}}{\partial x^\nu} - \frac{\partial g_{\nu\beta}}{\partial x^\gamma}\right]}_{\Gamma^\alpha{}_{\nu\beta}}\frac{\mathrm{d}x^\nu}{\mathrm{d}t}\frac{\mathrm{d}x^\beta}{\mathrm{d}t} \tag{E.20}$$

Here, we recognise the Christoffel symbol (section 5.2.4) as

$$\Gamma^\alpha{}_{\nu\beta} = \frac{1}{2}g^{\gamma\nu}\left[\frac{\partial g_{\gamma\nu}}{\partial x^\beta} + \frac{\partial g_{\gamma\beta}}{\partial x^\nu} - \frac{\partial g_{\nu\beta}}{\partial x^\gamma}\right] \tag{E.21}$$

we can write the geodesic equation as

$$0 = \frac{\mathrm{d}^2x^\alpha}{\mathrm{d}t^2} + \Gamma^\alpha{}_{\nu\beta}\frac{\mathrm{d}x^\nu}{\mathrm{d}t}\frac{\mathrm{d}x^\beta}{\mathrm{d}t} \tag{E.22}$$

$$0 = \ddot{x}^\alpha + \Gamma^\alpha{}_{\nu\beta}\dot{x}^\nu\dot{x}^\beta \tag{E.23}$$

which is the geodesic equation.

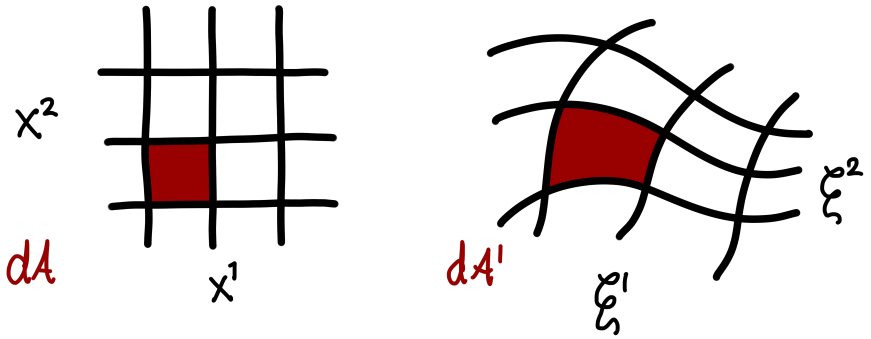


Figure E.1: This should illustrate how an area element dA in the latent space $\mathcal{X}$ in Cartesian coordinates gets distorted when mapped through f to another space $\mathcal{Y}$.

E.2 VISUALISING THE METRIC TENSOR

The entire metric tensor can be visualised as an ellipsoid (figure 5.6) where the size of the ellipsoid indicates the *size* of the metric tensor, corresponding to a unit space.

The metric tensor can be hard to visualise, so we introduce the volume element of the metric tensor. This scalar summarises the metric as it reflects a unit volume on the manifold. The volume measure is given by

VOLUME MEASURE

$$V = \sqrt{|G|}$$

Intuitively, we can think about the metric tensor as a transformation distorting a unit volume and the volume element as the volume of the unit. Bishop [127] used the volume element in two dimensions to visualise a distorted space. In two dimensions, the volume element is often called the magnification factor.

We denote the metric tensor as G in matrix notation and g_{ij} for the metric tensor in index notation. Both refer to the entire metric tensor. We assume the metric space to be complete and the tangent space at x_i to exist for all $x_i \in \mathcal{M}$.

E.3 KERNEL DERIVATIVES

A kernel $k(x_a, x_b)$ with input vectors $x_a, x_b \in \mathbb{R}^Q$,

$$\begin{aligned} k(x_a, x_b) &= \theta \exp\left(-\frac{\gamma}{2}\|x_a - x_b\|^2\right) \\ &= \theta \exp\left(-\frac{\gamma}{2}\sum_{k=1}^{Q}(x_a^k - x_b^k)^2\right), \end{aligned} \tag{E.24}$$

have the first derivative,

$$\begin{aligned}
\frac{\partial k(x_a, x_b)}{\partial x^j} &= k(x_a, x_b)\frac{\partial}{\partial x^j}\sum_k (x_a^k - x_b^k)^2 \\
&= -\frac{\gamma}{2}k(x_a, x_b)\frac{\partial}{\partial x^j}\sum_k (x_a^k - x_b^k)^2 \\
&= -\gamma k(x_a, x_b)\sum_k (x_a^k - x_b^k)\delta_{kj} \\
&= -\gamma(x_a^j - x_b^j)k(x_a, x_b), \qquad \text{(E.25)}
\end{aligned}$$

the second derivative,

$$\begin{aligned}
\frac{\partial^2 k(x_a, x_b)}{\partial x^i \partial x^j} &= \frac{\partial}{\partial x^i}\left[-\gamma(x_a^j - x_b^j)k(x_a, x_b)\right] \\
&= -\gamma\left[(\frac{\partial}{\partial x^i}(x_a^j - x_b^j)k(x_a, x_b) + ((x_a^j - x_b^j)\frac{\partial k(x_a, x_b)}{\partial x^i})\right] \\
&= -\gamma\left[\delta_{ij} - \gamma(x_a^j - x_b^j)(x_a^i - x_b^i)\right]k(x_a, x_b) \\
&= \gamma\left[\gamma(x_a^j - x_b^j)(x_a^i - x_b^i) - \delta_{ij}\right]k(x_a, x_b), \qquad \text{(E.26)}
\end{aligned}$$

and the third derivative,

$$
\begin{aligned}
\frac{\partial^3 k(x_a, x_b)}{\partial x^l \partial x^i \partial x^j} &= \frac{\partial}{\partial x^l}\left(\gamma\left[\gamma(x_a^j - x_b^j)(x_a^i - x_b^i) - \delta_{ij}\right] k(x_a, x_b)\right) \\
&= \gamma\left(\frac{\partial}{\partial x^l}\left[\gamma(x_a^j - x_b^j)(x_a^i - x_b^i) - \delta_{ij}\right] k(x_a, x_b) + \left[\gamma(x_a^j - x_b^j)(x_a^i - x_b^i) - \delta_{ij}\right]\frac{\partial k(x_a, x_b)}{\partial x^l}\right) \\
&= \gamma\Bigg(\left[\gamma\left(\frac{\partial}{\partial x^l}(x_a^j - x_b^j)(x_a^i - x_b^i) + (x_a^j - x_b^j)\frac{\partial}{\partial x^l}(x_a^i - x_b^i)\right) - \cancel{\frac{\partial \delta_{ij}}{\partial x^l}}\right] k(x_a, x_b) \\
&+ \left[\gamma(x_a^j - x_b^j)(x_a^i - x_b^i) - \delta_{ij}\right]\frac{\partial k(x_a, x_b)}{\partial x^l}\Bigg) \\
&= \gamma^2\left(\left[\delta_{li}(x_a^j - x_b^j) + \delta_{lj}(x_a^i - x_b^i)\right] - \left[\gamma(x_a^j - x_b^j)(x_a^i - x_b^i) - \delta_{ij}\right](x_a^l - x_b^l)\right) k(x_a, x_b)
\end{aligned}
\tag{E.27}
$$

In the derivatives, we have used the shorthand notation $\sum_{k=1}^{Q} = \sum_k$. We let $\gamma = l^{-1}$ but γ eases the notaiton. We leave the derivation of the fourth derivative as an exercise for the dedicated reader.

E.4 STARFISH CODE

Listing E.1: Two-dimensional Starfish (pinwheel dataset)

```
def make_pinwheel_data(radial_std, tangential_std, num_classes,
    num_per_class, rate):
    import numpy.random as npr
    rads = np.linspace(0, 2*np.pi, num_classes, endpoint=False)

    features = npr.randn(num_classes*num_per_class, 2)          * np
        .array([radial_std, tangential_std])
    features[:,0] += 1.
    labels = np.repeat(np.arange(num_classes), num_per_class)

    angles = rads[labels] + rate * np.exp(features[:,0])
    rotations = np.stack([np.cos(angles), -np.sin(angles), np.sin(
        angles), np.cos(angles)])
    rotations = np.reshape(rotations.T, (-1, 2, 2))

    return 10*npr.permutation(np.einsum('ti,tij->tj', features,
        rotations))
```

Listing E.2: Three dimensional Starfish (pinwheel dataset)

```
def make_starfish(num_classes = 5,num_per_class =50):

    v = make_pinwheel_data(0.75, 0.15, num_classes, num_per_class,
       0.025)
    z_noise = np.random.normal(0,0.1,size=[num_classes*
       num_per_class])

    xs = (v[:,0] - np.mean(v[:,0]))/np.std(v[:,0])
    ys = (v[:,1] - np.mean(v[:,1]))/np.std(v[:,1])

    vec = np.empty([num_classes*num_per_class,3])
    vec[:,0] = xs
    vec[:,1] = ys
    vec[:,2] = z_noise

    from scipy.spatial.transform import Rotation as R
    r_x = R.from_euler('x', 45, degrees=True)
    r_y = R.from_euler('y', 20, degrees=True)
    r_z = R.from_euler('z', 75, degrees=True)
    p = r_z.apply(r_y.apply(r_x.apply(vec)))
    return p
    #return torch.tensor(p,dtype=torch.float64)
```

E.5 IMPLEMENTATION DETAILS FOR RIEMANNIAN BROWNIAN MOTION

We have implemented this as a pyro distribution. It inherits from TorchDistribution, which allows pyro to name the distribution for internal tracking–essentially, it is just a simple wrapper. It implements a KL divergence between two IsotropicBrownianMotion distributions. The LogProb, Fit, and (r)Sample methods constitute the core of this distribution. The LogProb evaluation uses StochMan for computing geometric objects (e.g. metrics and geodesics), which is described in section 7.4.3 and the underlying machinery in section 7.1. The Fit method uses the LogProb to learn the mean and variance of the Riemannian Brownian motion.

The sampler is at the heart of the distribution, elaborated in the algorithm below. It implements a discretised Brownian motion (see section 7.5). Note that in this algorithm, the metric inverse is left out,

and we use the metric directly in the covariance of the subsequent step distribution as this leads to the desired behaviour.

Algorithm 3 The simple algorithm for a single Riemannian random walk. In practice, this can handle multiple samples in parallel.

Require: S // starting point for the random walk
Require: Metric M at S // positive, semi-definite, 2×2 matrix

```
dt = 1.0 / num_steps
t = scale**2
walk = list()
for _ in range(num_steps) do
  M = manifold.metric(S)
  C = t * dt * M
  S = MultivariateNormal(S, C).sample()
end for
return S
```

See also Arvanitidis, Hansen, and Hauberg [17, algorithm 2]

BIBLIOGRAPHY

[1] Cilie W Feldager, Søren Hauberg, and Lars Kai Hansen. "Spontaneous Symmetry Breaking in Data Visualization". In: *Artificial Neural Networks and Machine Learning*. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)". Springer, 2021, pp. 435–446. ISBN: 978-3-030-86339-5. DOI: 10.1007/978-3-030-86340-1_35 (Cited on pp. vii, 3, 12–15, 19, 129–132).

[2] Pablo Moreno-Muñoz, Cilie W. Feldager, and Søren Hauberg. *Revisiting Active Sets for Gaussian Process Decoders*. Sept. 2022. DOI: 10.48550/arXiv.2209.04636. arXiv: 2209.04636 [cs, stat] (Cited on pp. vii, 33, 34, 37, 39, 43, 45–49).

[3] Larin Cole Adams. "Gaussian Process Manifold Learning". In: (2021), p. 145 (Cited on pp. vii, 65, 72, 73, 96).

[4] Georgios Arvanitidis. "Geometrical Aspects of Manifold Learning". PhD thesis. Technical University of Denmark, 2019 (Cited on pp. ix, 2, 62).

[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation Learning: A Review and New Perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (Aug. 2013), pp. 1798–1828. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2013.50 (Cited on p. 1).

[6] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *arXiv:1312.6114 [cs, stat]* (May 2014). arXiv: 1312.6114 [cs, stat] (Cited on pp. 1, 31, 45).

[7] J. B. Tenenbaum. "A Global Geometric Framework for Nonlinear Dimensionality Reduction". In: *Science* 290.5500 (Dec. 2000), pp. 2319–2323. ISSN: 00368075, 10959203. DOI: 10.1126/science.290.5500.2319 (Cited on pp. 1, 11, 53, 83).

[8] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning.

Cambridge, Mass: MIT Press, 2006. ISBN: 978-0-262-18253-9 (Cited on pp. 1, 21, 24–26, 28, 29, 38, 67).

[9] Danilo Jimenez Rezende and Shakir Mohamed. "Variational Inference with Normalizing Flows". In: *arXiv:1505.05770 [cs, stat]* (June 2016). arXiv: 1505.05770 [cs, stat] (Cited on pp. 1, 31, 41).

[10] Alessandra Tosi et al. "Metrics for Probabilistic Geometries". In: *Uncertainty in Artificial Intelligence* (2014), p. 9 (Cited on pp. 2, 29, 66, 69, 74, 84, 96).

[11] Søren Hauberg. "Only Bayes Should Learn a Manifold (on the Estimation of Differential Geometric Structure from Data)". In: *arXiv:1806.04994 [cs, stat]* (Sept. 2019). arXiv: 1806.04994 [cs, stat] (Cited on pp. 2, 65, 66, 75).

[12] Martin Jørgensen and Søren Hauberg. "Isometric Gaussian Process Latent Variable Model for Dissimilarity Data". In: *arXiv:2006.11741 [cs, stat]* (June 2020). arXiv: 2006.11741 [cs, stat] (Cited on pp. 2, 19, 65).

[13] Francis J Anscombe. "Graphs in Statistical Analysis". In: *The American Statistician* 27.1 (1973), pp. 17–21 (Cited on p. 3).

[14] Sanjeev Arora, Wei Hu, and Pravesh K. Kothari. "An Analysis of the T-SNE Algorithm for Data Visualization". In: *arXiv:1803.01768 [cs]* (June 2018). arXiv: 1803.01768 [cs] (Cited on pp. 3, 17).

[15] Andre Esteva et al. "Dermatologist-Level Classification of Skin Cancer with Deep Neural Networks". In: *Nature* 542.7639 (Feb. 2017), pp. 115–118. ISSN: 1476-4687. DOI: 10.1038/nature21056 (Cited on pp. 3, 17).

[16] Maayan Frid-Adar et al. "GAN-based Synthetic Medical Image Augmentation for Increased CNN Performance in Liver Lesion Classification". In: *Neurocomputing* 321 (Dec. 2018), pp. 321–331. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2018.09.013 (Cited on pp. 3, 17).

[17] Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. "Latent Space Oddity: On the Curvature of Deep Generative Models". In: *arXiv:1710.11379 [stat]* (Jan. 2018). arXiv: 1710.11379 [stat] (Cited on pp. 4, 95, 96, 146).

[18] Brian C. Hall. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction.* Second. Graduate Texts in Mathematics.

Springer International Publishing, 2015.
ISBN: 978-3-319-13466-6. DOI: 10.1007/978-3-319-13467-3
(Cited on p. 4).

[19] A. Hatcher. *Algebraic Topology*.
Cambridge University Press, 2002 (Cited on p. 4).

[20] Gunnar Carlsson. "Topology and Data". In: *Bulletin of the American Mathematical Society* 46.2 (Jan. 2009), pp. 255–308.
ISSN: 0273-0979. DOI: 10.1090/S0273-0979-09-01249-X
(Cited on pp. 4, 6).

[21] Robert Ghrist. "Barcodes: The Persistent Topology of Data".
In: *Bulletin of the American Mathematical Society* 45.1 (2008), pp. 61–75. ISSN: 0273-0979, 1088-9485.
DOI: 10.1090/S0273-0979-07-01191-3 (Cited on pp. 4, 5, 9).

[22] Gunnar Carlsson.
"Persistent Homology and Applied Homotopy Theory".
In: *arXiv:2004.00738 [math]* (Apr. 2020).
arXiv: 2004.00738 [math] (Cited on p. 5).

[23] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian.
"Topological Persistence and Simplification". In: *Proceedings 41st Annual Symposium on Foundations of Computer Science*.
2000, pp. 454–463 (Cited on p. 6).

[24] Afra Zomorodian and Gunnar Carlsson.
"Computing Persistent Homology". In: *Proceedings of the Twentieth Annual Symposium on Computational Geometry*.
2004, pp. 347–356 (Cited on p. 6).

[25] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer.
"Stability of Persistence Diagrams". In: *Discrete & Computational Geometry* 37.1 (Jan. 2007), pp. 103–120.
ISSN: 1432-0444. DOI: 10.1007/s00454-006-1276-5
(Cited on pp. 6, 7).

[26] Mícheál O'Searcoid. *Metric Spaces*.
Springer Science & Business Media, 2006 (Cited on p. 8).

[27] Peter J. Huber. *Robust Statistics*. John Wiley & Sons, 2004.
ISBN: 978-0-471-65072-0 (Cited on p. 8).

[28] Lauren van der Maaten, Eric Postma, and Jaap van den Herik.
"Dimensionality Reduction: A Comparative Review".
In: (2009), p. 36 (Cited on p. 10).

[29] Geoffrey E. Hinton and Sam T. Roweis.
"Stochastic Neighbor Embedding".

In: *Advances in Neural Information Processing Systems*. 2003, pp. 857–864 (Cited on pp. 10, 18).

[30] Ehsan Amid and Manfred K. Warmuth. "TriMap: Large-scale Dimensionality Reduction Using Triplets".
In: *arXiv:1910.00204 [cs, stat]* (Oct. 2019).
arXiv: 1910.00204 [cs, stat] (Cited on pp. 10–13, 18).

[31] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. "Nonlinear Component Analysis as a Kernel Eigenvalue Problem".
In: *Neural Computation* 10.5 (July 1998), pp. 1299–1319.
ISSN: 0899-7667, 1530-888X. DOI: 10.1162/089976698300017467 (Cited on p. 10).

[32] Neil Lawrence.
"Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models".
In: *Journal of Machine Learning Research* (2005), p. 34 (Cited on pp. 11, 22, 24, 26, 28, 29, 33, 45, 77, 81).

[33] K. Pearson. "On Lines and Planes of Closest Fit to Systems of Points in Space". In: (1901), pp. 559–572 (Cited on pp. 11, 22).

[34] Harold Hotelling. "Analysis of a Complex of Statistical Variables with Principal Components".
In: *J. Educ. Psy.* 24 (1933), pp. 498–520 (Cited on pp. 11, 22).

[35] S. A. Nene, S. K. Nayar, and H. Murase.
CAVE | Software: COIL-100: Columbia Object Image Library.
https://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php. Feb. 1996
(Cited on p. 11).

[36] Sameer A Nene, Shree K Nayar, and Hiroshi Murase.
"Columbia Object Image Library (COIL-20)". In: (1996), p. 6
(Cited on p. 12).

[37] F. Pedregosa et al.
"Scikit-Learn: Machine Learning in {P}ython". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830
(Cited on p. 12).

[38] Sebastian Bitzer and Christopher K. I. Williams.
"Kick-Starting GPLVM Optimization via a Connection to Metric MDS".
In: *NIPS 2010 Workshop on Challenges of Data Visualization*.
Dec. 2010. DOI: http://cseweb.ucsd.edu/~lvdmaaten/workshops/nips2010/abstracts.html (Cited on p. 15).

[39] Lauren van der Maaten and Geoffrey Hinton.
"Visualizing Data Using T-SNE".
In: *Journal of machine learning research* (2008), pp. 2579–2605
(Cited on pp. 17, 53).

[40] George C. Linderman and Stefan Steinerberger.
"Clustering with T-SNE, Provably".
In: *arXiv:1706.02582 [cs, stat]* (June 2017).
arXiv: 1706.02582 [cs, stat] (Cited on p. 18).

[41] Neil D. Lawrence and Joaquin Quiñonero-Candela.
"Local Distance Preservation in the GP-LVM through Back Constraints". In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06.
Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, June 2006, pp. 513–520. ISBN: 978-1-59593-383-6.
DOI: 10.1145/1143844.1143909 (Cited on pp. 18, 31, 42, 51).

[42] Vidhi Lalchand, Aditya Ravuri, and Neil D. Lawrence.
"Generalised Gaussian Process Latent Variable Models (GPLVM) with Stochastic Variational Inference".
In: *arXiv:2202.12979 [cs, stat]* (Feb. 2022).
arXiv: 2202.12979 [cs, stat] (Cited on pp. 18, 28, 31).

[43] Irina Higgins et al.
"Towards a Definition of Disentangled Representations".
In: *arXiv:1812.02230 [cs, stat]* (Dec. 2018).
arXiv: 1812.02230 [cs, stat] (Cited on pp. 18, 19).

[44] T. Nishizeki and N. Chiba.
Planar Graphs: Theory and Algorithms. Vol. 32.
Annals of Discrete Mathematics, 1988 (Cited on p. 18).

[45] Cynthia Dwork et al. "Fairness Through Awareness".
In: *arXiv:1104.3913 [cs]* (Nov. 2011). arXiv: 1104.3913 [cs]
(Cited on p. 19).

[46] Nicki Skafte Detlefsen, Søren Hauberg, and Wouter Boomsma.
"What Is a Meaningful Representation of Protein Sequences?"
In: *arXiv:2012.02679 [cs, q-bio]* (Oct. 2021).
arXiv: 2012.02679 [cs, q-bio] (Cited on p. 19).

[47] Taco S. Cohen and Max Welling.
"Group Equivariant Convolutional Networks".
In: *arXiv:1602.07576 [cs, stat]* (June 2016).
arXiv: 1602.07576 [cs, stat] (Cited on p. 19).

[48] Daniel Kunin et al. "Neural Mechanics: Symmetry and Broken Conservation Laws in Deep Learning Dynamics".

In: *International Conference on Learning Representations*. Sept. 2020 (Cited on p. 19).

[49] Søren Hauberg. "Only Bayes Should Learn a Manifold". In: (2018), p. 15 (Cited on p. 19).

[50] Thomas Bayes. *An Essay towards Solv a Problem in the Doctrine of Chances*. Vol. 53. Philosophical Transactions of the Royal Society of London, 1763 (Cited on p. 21).

[51] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. New York: Springer, 2006. ISBN: 978-0-387-31073-2 (Cited on pp. 21, 22, 25, 30, 48, 53, 80, 94, 135).

[52] Carl Rasmussen and Zoubin Ghahramani. "Occam' s Razor". In: *Advances in Neural Information Processing Systems*. Vol. 13. MIT Press, 2000 (Cited on p. 21).

[53] Neil David Lawrence. "Variational Inference in Probabilistic Models". PhD thesis. Citeseer, 2001 (Cited on p. 22).

[54] Matthew James Beal. *Variational Algorithms for Approximate Bayesian Inference*. University of London, University College London (United Kingdom), 2003 (Cited on p. 22).

[55] Michael E. Tipping and Christopher M. Bishop. "Probabilistic Principal Component Analysis". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61.3 (1999), pp. 611–622. ISSN: 1467-9868. DOI: 10.1111/1467-9868.00196 (Cited on pp. 22, 23).

[56] Sam Roweis. "EM Algorithms for PCA and SPCA". In: *Advances in Neural Information Processing Systems*. Vol. 10. MIT Press, 1997 (Cited on p. 22).

[57] David Kristjanson Duvenaud. "Automatic Model Construction with Gaussian Processes". PhD thesis. University of Cambridge, 2014 (Cited on p. 25).

[58] Sumon Ahmed, M. Rattray, and A. Boukouvalas. "GrandPrix: Scaling up the Bayesian GPLVM for Single-Cell Data". In: *Bioinform.* (2019). DOI: 10.1093/bioinformatics/bty533 (Cited on pp. 28, 96).

[59] Neill D. F. Campbell and Jan Kautz. "Learning a Manifold of Fonts".

In: *ACM Transactions on Graphics* 33.4 (July 2014), pp. 1–11. ISSN: 07300301. DOI: 10.1145/2601097.2601212 (Cited on pp. 29, 96).

[60] Aidan Scannell. "Bayesian Learning for Control in Multimodal Dynamical Systems". PhD thesis. University of Bristol, 2022 (Cited on pp. 29, 68, 96).

[61] Kristopher T. Jensen et al. "Manifold GPLVMs for Discovering Non-Euclidean Latent Structure in Neural Data". In: *arXiv:2006.07429 [cs, q-bio, stat]* (Oct. 2020). arXiv: 2006.07429 [cs, q-bio, stat] (Cited on pp. 29, 96).

[62] Michalis K Titsias. "Variational Learning of Inducing Variables in Sparse Gaussian Processes". In: *Artificial Intelligence and Statistics* (2009), pp. 567–574 (Cited on pp. 29, 30, 32).

[63] Andreas Damianou. "Deep Gaussian Processes and Variational Propagation of Uncertainty". PhD thesis. University of Sheffield, 2015 (Cited on pp. 29, 30).

[64] Alex Smola and Peter Bartlett. "Sparse Greedy Gaussian Process Regression". In: *Advances in Neural Information Processing Systems*. Vol. 13. MIT Press, 2001 (Cited on p. 29).

[65] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. "A Unifying View of Sparse Approximate Gaussian Process Regression". In: *Journal of Machine Learning Research* (2005), pp. 1939–1959 (Cited on p. 29).

[66] Edward Snelson and Zoubin Ghahramani. "Sparse Gaussian Processes Using Pseudo-inputs". In: *Advances in Neural Information Processing Systems 18*. Ed. by Y. Weiss, B. Schölkopf, and J. C. Platt. MIT Press, 2006, pp. 1257–1264 (Cited on pp. 29, 30).

[67] Michalis K Titsias and Neil D Lawrence. "Bayesian Gaussian Process Latent Variable Model". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (2010), p. 8 (Cited on pp. 30–32, 40, 45).

[68] Johan Jensen. "Sur Les Fonctions Convexes et Les Inégualités Entre Les Valeurs Moyennes". In: Zenodo, Nov. 1906. DOI: 10.1007/bf02418571 (Cited on p. 30).

[69] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. "Variational Inference: A Review for Statisticians".

In: *Journal of the American Statistical Association* 112.518 (Apr. 2017), pp. 859–877. ISSN: 0162-1459, 1537-274X. DOI: 10.1080/01621459.2017.1285773. arXiv: 1601.00670 (Cited on pp. 30, 31).

[70] Michalis K. Titsias. *Variational Model Selection for Sparse Gaussian Process Regression.* 2009 (Cited on p. 31).

[71] Matt Hoffman et al. "Stochastic Variational Inference". In: *arXiv:1206.7051 [cs, stat]* (Apr. 2013). arXiv: 1206.7051 [cs, stat] (Cited on pp. 31, 42).

[72] James Hensman, Alex Matthews, and Zoubin Ghahramani. "Scalable Variational Gaussian Process Classification". In: *arXiv:1411.2005 [stat]* (Nov. 2014). arXiv: 1411.2005 [stat] (Cited on pp. 31, 42).

[73] James Hensman, Nicolo Fusi, and Neil D Lawrence. "Gaussian Processes for Big Data". In: (2013), p. 9 (Cited on p. 31).

[74] Thang D Bui, Josiah Yan, and Richard E Turner. "A Unifying Framework for Gaussian Process Pseudo-Point Approximations Using Power Expectation Propagation". In: (2016), p. 72 (Cited on pp. 31, 32, 42).

[75] Siddharth Ramchandran, Miika Koskinen, and Harri Lähdesmäki. "Latent Gaussian Process with Composite Likelihoods and Numerical Quadrature". In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics.* PMLR, Mar. 2021, pp. 3718–3726 (Cited on p. 31).

[76] Rui Shu et al. "Amortized Inference Regularization". In: *Advances in Neural Information Processing Systems.* Vol. 31. Curran Associates, Inc., 2018 (Cited on p. 31).

[77] Matthias Bauer, Mark van der Wilk, and Carl Edward Rasmussen. "Understanding Probabilistic Sparse Gaussian Process Approximations". In: *Advances in Neural Information Processing Systems 29.* Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 1533–1541 (Cited on p. 32).

[78] Vincent Dutordoir, Nicolas Durrande, and James Hensman. "Sparse Gaussian Processes with Spherical Harmonic Features". In: *Proceedings of the 37th International Conference on Machine Learning.* PMLR, Nov. 2020, pp. 2793–2802 (Cited on p. 32).

[79] E Fong and C C Holmes.
"On the Marginal Likelihood and Cross-Validation".
In: *Biometrika* 107.2 (June 2020), pp. 489–496. ISSN: 0006-3444.
DOI: 10.1093/biomet/asz077 (Cited on pp. 33, 35, 36, 50).

[80] Diederik P Kingma and Jimmy Ba.
"Adam: A Method for Stochastic Optimization".
In: *International Conference on Learning Representations (ICLR)* (2015) (Cited on pp. 39, 43).

[81] Kunihiko Fukushima.
"Cognitron: A Self-Organizing Multilayered Neural Network".
In: *Biological Cybernetics* 20.3 (Sept. 1975), pp. 121–136.
ISSN: 1432-0770. DOI: 10.1007/BF00342633 (Cited on p. 40).

[82] Vinod Nair and Geoffrey E Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *ICML*. 2010, p. 8 (Cited on p. 40).

[83] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic Backpropagation and Approximate Inference in Deep Generative Models".
In: *arXiv:1401.4082 [cs, stat]* (May 2014).
arXiv: 1401.4082 [cs, stat] (Cited on p. 42).

[84] Yann LeCun, Corinna Cortes, and Chris Burges.
MNIST Handwritten Digit Database.
http://yann.lecun.com/exdb/mnist/. 1999 (Cited on pp. 43, 78).

[85] Han Xiao, Kashif Rasul, and Roland Vollgraf.
Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. 2017-08-28, 2017.
arXiv: 1708.07747 [cs.LG] (Cited on p. 43).

[86] Alex Krizhevsky, Vinod Nair, and Hinton Geoffrey.
"Cifar-10 and Cifar-100 Datasets". In: *Technical Report* Canadian Institute for Advanced Research (2010), p. 60 (Cited on p. 43).

[87] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library".
In: *Advances in Neural Information Processing Systems 32*.
Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8026–8037 (Cited on pp. 43, 78).

[88] Eli Bingham et al.
"Pyro: Deep Universal Probabilistic Programming".
In: *The Journal of Machine Learning Research* 20.1 (Jan. 2019), pp. 973–978. ISSN: 1532-4435 (Cited on p. 47).

[89] Hao Chen et al. “Stochastic Gradient Descent in Correlated Settings: A Study on Gaussian Processes”. In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 2722–2733 (Cited on p. 50).

[90] Bernard Schutz. *A First Course in General Relativity*. Cambridge university press, 2009 (Cited on pp. 54, 58, 59).

[91] Andrew Pressley. *Elementary Differential Geometry*. Springer Science \& Business Media, 2010 (Cited on p. 54).

[92] Xavier Pennec. “Intrinsic Statistics on Riemannian Manifolds: Basic Tools for Geometric Measurements”. In: *Journal of Mathematical Imaging and Vision* (2006), pp. 127–154 (Cited on pp. 54, 96).

[93] Manfredo Perdig\tilde{a}o do Carmo. *Riemannian Geometry*. Mathematics. Theory & Applications. Boston: Birkhäuser, 1992. ISBN: 978-0-8176-3490-2 978-3-7643-3490-1 (Cited on p. 57).

[94] Andrew Gordon Wilson and Zoubin Ghahramani. “Generalised Wishart Processes”. In: (2010), p. 9 (Cited on p. 68).

[95] Robb J Muirhead. *Aspects of Multivariate Statistical Theory*. John Wiley & Sons, Inc., 1982. ISBN: 978-0-471-09442-5 (Cited on pp. 68, 69).

[96] Arjun K Gupta and Daya K Nagar. *Matrix Variate Distributions*. Chapman and Hall/CRC, 2018 (Cited on p. 69).

[97] KV Mardia. “JT Kent. and J. M. Bibby”. In: *Multivariate Analysis* (1979) (Cited on p. 69).

[98] Didong Li and Sayan Mukherjee. “Random Lie Brackets That Induce Torsion: A Model for Noisy Vector Fields”. In: *arXiv:2007.07309 [math]* (July 2020). arXiv: 2007.07309 [math] (Cited on p. 73).

[99] Justin D. Bewsher et al. “Distribution of Gaussian Process Arc Lengths”. In: *arXiv:1703.08031 [stat]* (Mar. 2017). arXiv: 1703.08031 [stat] (Cited on p. 74).

[100] Minoru Nakagami. “The M-Distribution—A General Formula of Intensity Distribution of Rapid Fading”. In: *Statistical Methods in Radio Wave Propagation*. Elsevier, 1960, pp. 3–36 (Cited on p. 74).

[101] Tom LaGatta and Jan Wehr.
"Geodesics of Random Riemannian Metrics".
In: *Communications in Mathematical Physics* 327.1 (Apr. 2014), pp. 181–241. ISSN: 0010-3616, 1432-0916.
DOI: 10.1007/s00220-014-1901-8. arXiv: 1206.4939
(Cited on p. 74).

[102] Jakub Tomczak and Max Welling. "VAE with a VampPrior".
In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. PMLR, Mar. 2018, pp. 1214–1223 (Cited on p. 77).

[103] Georgios Arvanitidis, Bogdan Georgiev, and Bernhard Schölkopf.
"A Prior-Based Approximate Latent Riemannian Metric".
In: *arXiv:2103.05290 [cs, stat]* (Mar. 2021).
arXiv: 2103.05290 [cs, stat] (Cited on p. 77).

[104] Nicki S. Detlefsen et al. *StochMan*. 2021 (Cited on p. 78).

[105] Isaac Jacob Schoenberg. "Contributions to the Problem of Approximation of Equidistant Data by Analytic Functions. Part B. On the Problem of Osculatory Interpolation. A Second Class of Analytic Approximation Formulae".
In: *Quarterly of Applied Mathematics* 4.2 (1946), pp. 112–141
(Cited on p. 82).

[106] Richard H Bartels, John C Beatty, and Brian A Barsky.
An Introduction to Splines for Use in Computer Graphics and Geometric Modeling. Morgan Kaufmann, 1995 (Cited on p. 82).

[107] Edsger W Dijkstra.
"A Note on Two Problems in Connexion with Graphs".
In: *Numerische mathematik* 1.1 (1959), pp. 269–271
(Cited on p. 83).

[108] T. Cormen, C. E. Leiserson, and R. L. Rivest.
Introduction to Algorithms. Cambridge, MA, 1990
(Cited on p. 83).

[109] Alessandra Tosi and Alfredo Vellido. "Local Metric and Graph Based Distance for Probabilistic Dimensionality Reduction".
In: *Proceedings of the Workshop on Features and Structures (FEAST 2014) International Conference on Pattern Recogni-tion (ICPR 2014), Stockholm, Sweden*. Citeseer, 2014
(Cited on p. 83).

[110] Aric A. Hagberg et al. "Exploring Network Structure, Dynamics, and Function Using NetworkX".

In: *Proceedings of the 7th Python in Science Conference*. Aug. 2008, pp. 11–15 (Cited on p. 83).

[111] Dimitris Kalatzis et al. "Variational Autoencoders with Riemannian Brownian Motion Priors". In: *arXiv:2002.05227 [cs, stat]* (Feb. 2020). arXiv: 2002.05227 [cs, stat] (Cited on pp. 89, 91, 92, 95, 96).

[112] Elton P Hsu. *Stochastic Analysis on Manifolds*. Vol. 38. American Mathematical Soc., 2002 (Cited on pp. 89–91).

[113] Jean-François Le Gall et al. *Brownian Motion, Martingales, and Stochastic Calculus*. Vol. 274. Springer, 2016 (Cited on p. 89).

[114] Lars Buitinck et al. "API Design for Machine Learning Software: Experiences from the Scikit-Learn Project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122 (Cited on p. 95).

[115] Anthony Carapetis. *Answer to "Why Does Brownian Motion Have Drift on Riemannian Manifolds?"* July 2017 (Cited on p. 95).

[116] Xavier Pennec, Stefan Sommer, and Tom Fletcher. *Riemannian Geometric Statistics in Medical Image Analysis*. Academic Press, 2019 (Cited on p. 95).

[117] Raquel Urtasun et al. "Topologically-Constrained Latent Variable Models". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: Association for Computing Machinery, July 2008, pp. 1080–1087. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390292 (Cited on p. 96).

[118] Anton Mallasto, Søren Hauberg, and Aasa Feragen. "Probabilistic Riemannian Submanifold Learning with Wrapped Gaussian Process Latent Variable Models". In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2019, pp. 2368–2377 (Cited on p. 96).

[119] Alexander Terenin. "Gaussian Processes and Statistical Decision-making in Non-Euclidean Spaces". PhD thesis. Apr. 2022. arXiv: 2202.10613 (Cited on p. 96).

[120] Josep M Oller. "On an Intrinsic Analysis of Statistical Estimation". In: *Multivariate Analysis: Future Directions 2*. Elsevier, 1993, pp. 421–437 (Cited on p. 96).

[121] Kanti V Mardia, Peter E Jupp, and KV Mardia. *Directional Statistics*. Vol. 2. Wiley Online Library, 2000 (Cited on p. 96).

[122] Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. "A Locally Adaptive Normal Distribution". In: (2016), p. 9 (Cited on p. 96).

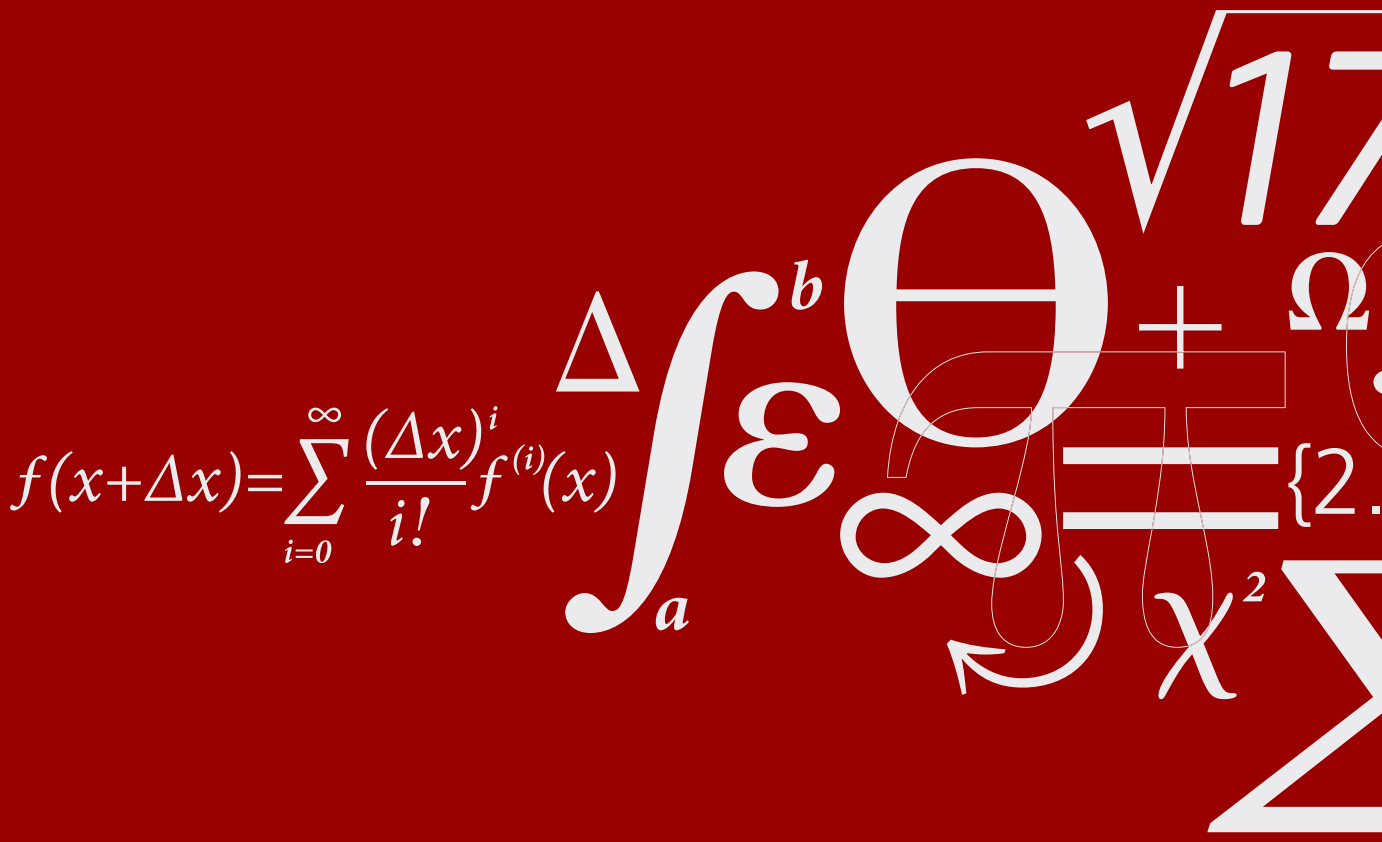
[123] Søren Hauberg. *On the Geometry of Latent Variable Models*. Technical Report. Technical University of Denmark, 2018, p. 3 (Cited on p. 96).

[124] Aasa Feragen, Francois Lauze, and Soren Hauberg. "Geodesic Exponential Kernels: When Curvature and Linearity Conflict". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, USA: IEEE, June 2015, pp. 3032–3042. ISBN: 978-1-4673-6964-0. DOI: 10.1109/CVPR.2015.7298922 (Cited on p. 96).

[125] Sadeep Jayasumana et al. "Kernel Methods on Riemannian Manifolds with Gaussian RBF Kernels". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.12 (Dec. 2015), pp. 2464–2477. ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2015.2414422. arXiv: 1412.0265 (Cited on p. 96).

[126] Noémie Jaquier et al. *Bringing Robotics Taxonomies to Continuous Domains via GPLVM on Hyperbolic Manifolds*. Oct. 2022. arXiv: 2210.01672 [cs] (Cited on p. 96).

[127] C.M. Bishop. "Magnification Factors for the GTM Algorithm". In: *Fifth International Conference on Artificial Neural Networks*. Vol. 1997. Cambridge, UK: IEE, 1997, pp. 64–69. ISBN: 978-0-85296-690-7. DOI: 10.1049/cp:19970703 (Cited on p. 142).



TECHNICAL UNIVERSITY OF DENMARK
COGNITIVE SYSTEMS

RICHARD PETERSENS PLADS, BUILDING 321
2800 KGS. LYNGBY
PHONE +45 4525 1700

WWW.COMPUTE.DTU.DK